DEVELOPMENT OF A FLIGHT AVIONICS SYSTEM FOR AN AUTONOMOUS MICRO AIR VEHICLE

By

JASON PLEW

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2004

This work is dedicated to my wife, Shalomoth Plew. Her love and support during this research allowed me to not only accomplish my goals for this project, but also ensured that my life was filled with happiness. It is also dedicated to my father, Richard Plew, who made all of this possible. It was his support and guidance through the many years of high school research projects that eventually led me to robotics, the Machine Intelligence Lab, and eventually the MAV project.

ACKNOWLEDGMENTS

I would like to thank the professors affiliated with the Machine Intelligence Lab, including Dr. Keith Doty, Dr. Antonio Arroyo, Dr. Eric Schwartz, Dr. Michael Nechyba, Dr. Karl Gugel, and Dr. Michael Lynch, for providing me with the knowledge necessary for this research. Their lessons both in the classroom and outside it were invaluable. I would also like to thank the members of the AVCAAF research group. This of course includes Dr. Pete Ifju and his students for developing the Micro Air Vehicles that we would use for our platforms. Without their amazing vehicles, none of this would have been possible. Thanks also go out to the controls team, led by Dr. Andrew Kurdila and Dr. Rick Lind, who developed the navigational controller for the MAV128 R4. Among their students, I would especially like to thank Mujahid Abdulrahim, for developing the inertial-based controller for the MAV128 R5 to verify its ability to control a MAV, and for sharing his knowledge and experience in RC aircraft and control systems. I would also like to thank Dr. Nechyba and the other students in the MIL MAV group, and most importantly Jason Grzywna for his development of the ground station system and participation in this research.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

Figure page

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

DEVELOPMENT OF A FLIGHT AVIONICS SYSTEM FOR AN AUTONOMOUS MICRO AIR VEHICLE

By

Jason Plew

December 2004

Chair: Antonio Arroyo
Major Department: Electrical and Computer Engineering

The goal of this thesis was to develop a complete avionics system for a Micro Air Vehicle. This system must support the research UF is currently conducting in demonstrating an autonomous MAV capable of operations in an urban environment. To support this ability, an onboard system was needed that included both an Inertial Measurement Unit (IMU) for stabilization of the MAV and GPS for navigational support. There was also a need for an RF transceiver for linking the MAV to a Ground Station for telemetry and control. A flight computer, designated the MAV128, was required to provide an interface between all of these systems, and to eventually allow for an onboard controller. Combined with an onboard camera, video transmitter, and vision-processing algorithms on the ground station, this system was to allow us to begin developing a MAV capable of the advanced maneuvers required for flying among buildings.

A prototype flight system was developed for a 24" MAV but could not demonstrate autonomous control and navigation due to significant noise in the data from the IMU and

the overall weight of the onboard system. In the development of the next system, the weight was reduced and onboard autonomous navigation was demonstrated, with flight stabilization being handled by a ground-based horizon tracking system already developed. We then integrated an IMU into the MAV128, and though development in both a mechanical mounting system for the system and advanced control filters are required to ensure valid inertial data, flight test results have been encouraging. Once this version of the flight system has demonstrated autonomous flight, we will turn to incorporating inertial, vision, and GPS control onboard, moving us even closer to our goal of demonstrating autonomous urban operation of a MAV.

CHAPTER 1
INTRODUCTION

A major focus of the United States Air Force (USAF) in the next decade is the development of Unmanned Air Vehicles (UAVs) that can be deployed in tactical scenarios, as opposed to pure strategic operations, such as the Predator [1]. This includes the ability for soldiers in the field to deploy Micro Air Vehicles (MAVs) for onsite surveillance. The military also wishes to use these aircraft in complex environments, such as urban areas, in multiple scenarios. Of course, as is the case with most military technology, there are also numerous applications in the civilian sector.

The Micro Aerial Vehicle Lab at the University of Florida's Department of Mechanical and Aerospace Engineering has been involved in the research and design of such aircraft for several years, and has become very proficient in their development of MAVs [2, 3]. Their technology in the development of these airplanes has led them to win the International MAV competition for the past five years in a row because of their small size and relatively long range. The airplanes of the MAV lab have traditionally been controlled by off-the-shelf RC airplane equipment. To increase their range, the airplanes often flew with a forward-looking camera and a video transmitter, allowing the pilot to fly the aircraft when it was beyond the pilot's visual range. Given the size of the airplane, the MAV pilots had to become quite skilled to keep the planes in the air.

The research of Scott Ettinger helped change the great skill requirement of the pilots; he developed the first autonomous MAVs [4, 5]. In his research he took advantage of the fact that many of the MAVs were sending a video signal to the ground, and

developed a system to analyze these images to find the horizon. A PID controller was developed to take the position of the horizon in the image, and determine the necessary commands to the servos to adjust the control surfaces to keep the horizon level. It could also keep the horizon at a specific angle, allowing the aircraft to hold a specific roll angle and thus orbit a position. The necessary servo commands were then sent through a device that would convert them to signals understood by a standard RC controller, which would then transmit the commands to the plane back to the airplane. Once it was working in real time, the vision-based flight stability system could keep a MAV in the air without any input from a human pilot. Further work led to the ability for the controller to take input from a joystick, allowing the MAV to be flown by any untrained pilot.

The success of Ettinger's work led to the initiation of a new project – Active Vision for Control of Agile Autonomous Flight (AVCAAF) [6]. The purpose of this program, which began in 2003 and is to last for five years, is to develop Micro Air Vehicles that are capable of autonomous flight within complex environments, such as urban settings. However, there are multiple challenges that must be overcome to accomplish this goal. Though autonomous flight had been demonstrated through the horizon-tracking system, this alone was not sufficient for performing the complex maneuvers required in any environment beyond an open field, such as a forest or city.

To be able to achieve the control necessary to fly a MAV autonomously in an urban environment, we needed to tackle the problem at three levels. Traditionally, control of aerial robots has been accomplished through an Inertial Measurement Unit (IMU). Such devices, which often include accelerometers and gyroscopes, are very good at determining the instantaneous movement of the airplane. Therefore, they can be very

accurate at tracking the movement of the vehicle over a short period of time. However, over longer periods of time the error of the devices increases to the point that they are ill suited to measuring the movement of the airplane. GPS has often been used to provide the necessary data to recalibrate the IMU data. A controller can start with a known position from the GPS, and track the vehicle's progress with inertial sensors, updating the position to correct for errors on every GPS position update. However, the GPS unit can only determine the position of the aircraft in relationship to the earth, and potentially previously known fixed objects, if such data is available. It is unable to provide any real information about the changing environment around the airplane. The vision system, however, can be placed directly between the Inertial System and the GPS, as yet another sensor to provide data to the airplane control system. A vision system's ability, for example, to track a moving target or determine that an obstacle is ahead of the aircraft allows it to provide information that would not be available to the more traditional methods of airplane control. By using GPS, Vision, and an IMU, the inadequacies of any one system are covered by the other two [7].

The first component of the vision system had already been developed, in the form of the Horizon Tracking System. While the 900 MHz x86-based computer used in the original tests were adequate for running the horizon tracking system, the transfer of the horizon-tracking program to new computer technology ensured that enough processing power was available for more complicated vision processing tasks. The technology for GPS and IMU sensors that could be used for aircraft on the scale of even the largest of MAVs was only in the initial levels of development. No off-the-shelf systems existed that could be both flown on MAV, and integrated with the vision system already being

developed. Therefore, we had to develop the technology ourselves, i.e., develop a complete flight system capable of demonstrating autonomous flights. We began developing such a flight system in the spring of 2003.

While the Department of Defense defines a Micro Air Vehicle as an airplane with a wingspan of less then six inches, the current level of technology was not advanced enough to be used in such a platform. Therefore, we continued to use the 24-inch wingspan systems used in the earlier vision-based flight stability experiments. Two initial attempts to develop a flight system for a MAV were focused on identifying the proper devices to use and to then develop a means of transferring telemetry from the airplane to the ground. These systems also helped us to understand the problems we faced in developing a MAV with the capability of performing autonomous urban operations, which led us to the requirements for the third revision of the flight hardware. This third system culminated in our first attempt at autonomous flight.

CHAPTER 2
REVISION 3: PROTOTYPE FLIGHT SYSTEM

System Requirements

The goals of the first year of the AVCAAF project were based on the realization that the autonomous MAVs required Inertial and GPS capabilities in addition to vision. The primary effort was to develop an onboard flight system for a two-foot wingspan MAV that included both an IMU and a GPS receiver. This setup was developed to work in conjunction with a new ground station including an enhanced vision-guidance system based on Ettinger's work. The overall system requirement were to autonomously fly the MAV through multiple GPS waypoints, and be ready for demonstration at Eglin AFB in July 2003. Initially, the desire was to build as much possible functionality into the system as possible. This included the ability to both send sensor data to the ground as well as store it in the onboard system for later retrieval. Multiple locations of the controller (both on the ground station and in the onboard system) were considered. We also wanted the ability to control the servos through both the onboard flight system and through standard RC equipment, allowing the computer to control the airplane and yet insure that a human pilot could step in if necessary. As experimentation began, some of this functionality was determined to be unnecessary. Other components proved to be inadequate for accomplishing the task of developing a controller for the airplane, and were abandoned. As a result, the prototype flight system that was developed by the end of the summer of 2003 had gone through several iterations, and the relationship between the different components often changed.

There were multiple challenges in developing the onboard system. The major issue was the weight constraint. A MAV with a wingspan of two feet, once outfitted with motors, servos, batteries, and RC equipment, could only handle payloads of less then 100 grams. Furthermore, there was the difficulty in developing a system that was located onboard the airplane – access to the device when flying in the air was limited, and it would be difficult to replicate these conditions in the lab. This was especially the case when it came time to develop a controller to link the onboard sensors to the control surfaces of the MAV.

Initially, it was hoped that a controller could be developed to function onboard the aircraft. And as the research progressed, this remained the end goal of the flight system. However, any time an onboard controller was in need of serious modification, either we would have to land the airplane so that the onboard system could be reprogrammed, or we would have to develop a robust method of transmitting a new controller to the airplane. Since both of these options were considered to be unsuitable for the development phase, we decided to develop the controller on a ground station. This approach made it easier for the controls team to develop the necessary algorithms to fly the airplane autonomously, allowing the fusion of the vision, inertial, and GPS data to occur in a single location. The airplane needed to send all of its sensor data to the ground station, and either process servo commands through standard RC equipment or through the onboard flight system. Once the process of developing a controller for the MAV on the ground had been refined, we began work on the development of transferring the system to the onboard hardware.

Flight System Components

The major goal of the flight system was to have onboard inertial sensors and a GPS receiver. Due to weight restrictions, very small sensors had to be found, even if the tradeoff was in reduced accuracy. Furthermore, we needed to include the ability for the flight system to transmit at least some of this sensor data to the ground. Finally, some form of a processing system was required to tie all these individual components together, and link them to the servo motors controlling the plane.

Inertial Sensors: Microstrain 3DM-G

Instead of attempting to build an IMU from discrete parts, we tried to obtain an integrated component light in weight yet able to give us the accurate data necessary for stable flight. The solution was the 3DM-G, an IMU from Microstrain (Figure 2-1). This device included accelerometers in all three directions to give us the instantaneous movement of the airplane, as well as gyroscopes for all three axes to provide roll, pitch and yaw rates. Also included were magnetometers to give the orientation of the aircraft in relationship to the Earth's magnetic field, and other assorted sensors. An onboard chip would process the sensor data to provide not only filtered sensor data, but also the orientation results (e.g., Euler Angles) that are useful for controllers. The 3DM-G could be communicated to through a Universal Asynchronous Receiver Transmitter (UART), a standard communications device found on most

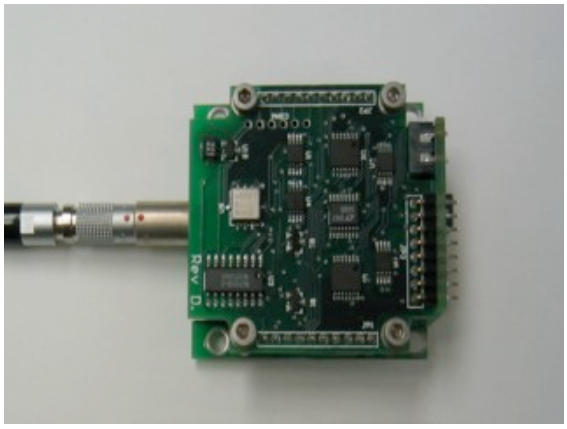

Figure 2-1: 3DM-G IMU

embedded processors, as well as on the serial ports of personal computers [8].

Global Positioning System Receiver: Axiom Swift A2

The other major system was the GPS unit. Once again weight was a major issue in deciding on which device to select. However, in this case, the accuracy of the data was not as important a factor. No GPS receiver currently available could return data accurate enough to be useful in the flight stability problem, and therefore, the GPS was only used for navigation. Because of this, a GPS unit that was only accurate to within several feet was not an issue.

To save on weight, we looked for a GPS with an integrated antenna. Though using an integrated antenna results in less capability in detecting the GPS satellites, especially in extreme maneuvers, we decided that the savings in weight made such a choice worth the degradation in navigational data. The resulting solution was the Axiom Swift A2 GPS Receiver [9]. This device, shown with an interface in Figure 2-2, communicates both the position and course of the aircraft through a UART, and had been used in previous work that focused on exploring the possibility of integrating the horizon tracking system with GPS [10]. Given the receiver's low weight of 20 grams and the fact that we already had some of these units available made this choice for a GPS unit nearly ideal.

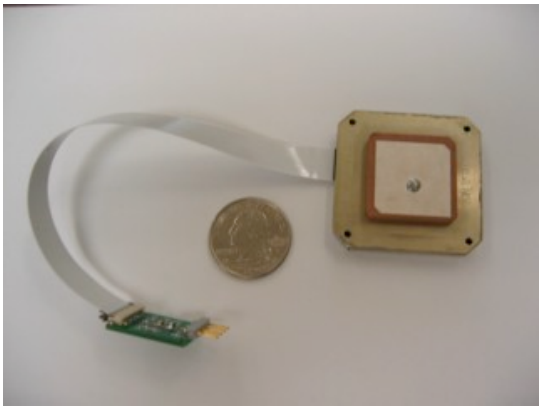

Figure 2-2: Axiom GPS Radio

By default, the Swift A2 communicates using the industry standard NMEA Protocol, which can provide multiple navigational data, including

Longitude, Latitude, and Course Speed and Bearing. The drawback was that the NMEA protocol uses ASCII to transfer data, which takes up much more bandwidth then using a raw binary format. The onboard system first had to convert the GPS data into binary if it was to be sent to the ground. While this was feasible, we wished to avoid having to waste processing time parsing the ASCII stream. Fortunately, the Swift A2 could also be set to communicate using the raw SiRF standard protocol. The interface to the GPS using this standard proved to be much simpler. One minor issue was that the SiRF protocol, unlike NMEA, communicated position data using a 3D axis system with the origin at the center of the earth, the z-axis pointing to the north pole, and the x-axis along the equatorial plane and perpendicular to the prime meridian [11]. For the development of navigational controller to be straight forward, the format of the position needed to be in the latitude, longitude, and altitude format. However, the conversion proved to be just a matter of converting from the Cartesian coordinate system to a Spherical Polar system, a process simple to carry out on the ground station computer.

RF Transceiver: Microhard MHX-2400

We needed some way of being able to communicate with the flight system when it was in the air. The bandwidth could range between a few bytes for sending servo commands up to the plane (bypassing the RC Radio System) and getting simple status information back to receiving several bytes of telemetry including Inertial and GPS data. The necessary data had to be sent at a rate of 30 Hz for the controller on the ground station to function properly. We also had to be concerned with the range of the transceiver. We needed to maintain the radio link for at least a few miles. One benefit of the MAV system was that with one of the transceivers being in the air, we were operating in close to Line of Sight (LOS) conditions.

Weight and size were as always a concern in determining the requirements for a transceiver. We also needed to look for a device that operated in one of the unlicensed bands of spectrum, so that we would not have to spend time getting approval from the necessary government agencies to operate the system. These frequencies included 900 MHz, 2.5 GHz, and 5 GHz. The latter spectrum was only just beginning to be utilized, and so the options for using a device at that frequency band were limited. We therefore decided to use either a 900 MHz or 2.5 GHz. Preliminary tests showed that using a data transceiver and a video transmitter operating at the same frequency resulted in too much video noise on the ground. We did have the ability to transmit video at either 2.5 GHz or 900 MHz, so it was an easy matter to keep the two devices from interfering with one another.

We decided that it would take too long to develop our own transceiver, even using the RF chipsets such as Intersil's Prism that have become popular in the past few years. Therefore, an off the shelf solution was investigated. Initial work had been undertaken in getting data to the ground from some of our initial flight systems using the CompactRF Radio Transceiver from Microhard Corp. This device operates at 900 MHz, and can, under optimal conditions, transfer data at up to 19.2Kbits for a range of 20 miles. [12] Theoretically, we should then be able to send a packet of 50 bytes at a rate of about 20 Hz. However, even with the rather ideal conditions of having one of the radios in the air, we were only able to send inertial data to the ground at around 10 Hz. Since this did not meet our targeted data rate of 30 Hz, another radio needed to be found.

After an exhaustive search for other off the shelf solutions, we settled for another radio made by Microhard. The MHX-2400 (Figure 2-3) advertised a 115.2K data transfer rate at the same range as the CompactRF radio. For that matter, they offered the MHX-900 radio as well, which was identical to the MHX-2400 except that it operated at 900 MHz [13]. Using the Microhard radios was attractive because it gave us flexibility in determining which frequency bands the Datalink and Video systems would use. However, a major drawback was the size and weight – whereas the CompactRF was 2" x 1.5" at 20g, the MHX-2400 was 3.5" x 2" at 75g. However, there appeared to be no other viable options at that time, and so this device was chosen to allow telemetry to be sent from the MAV to a controller on the ground station.

Figure 2-3: MHX-2400 RF Transceiver

The MAV128 R3 Onboard Computer

It was evident from the beginning that some sort of embedded processor was required onboard the aircraft. There was initially some thought given to using an off the shelf Power PC embedded system, running ucLinux. Such a device could be interfaced to inertial sensors and a GPS, and it was powerful enough that it could easily run a controller onboard, and possibly even part of the vision system. However, the size, weight, and power requirements of this device led us to drop this as a solution, as did the fact that its version of ucLinux required over a minute to boot. There was still the desire to build a system that could handle the GPS and IMU systems for now, but eventually be

capable of supporting the horizon tracking system onboard. Therefore, a DSP system was considered. However, we only had a few months to develop a flight system capable of using the vision and inertial sensors to keep the MAV stable, and using GPS to navigate. Therefore, it was decided to just focus on the IMU and GPS for the onboard systems. The vision processing systems would remain on the ground station computer for now. For this situation, a normal embedded microcontroller could be used.

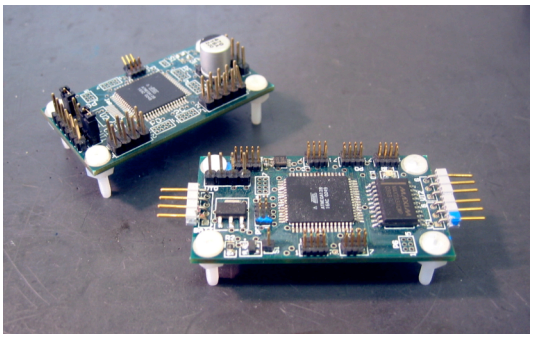

Figure 2-4: MAV128 R1, R2

An Atmel AVR Mega128 microcontroller was selected, a device we had used in past projects. This was the most powerful processor in the AVR family at the time, capable of running at up to 16 MHz, and giving up to 16 MIPS of throughput. It contains 128K of Flash Memory for Programs, and 4K of RAM for data [14]. Besides its performance and the fact that we were already familiar with this device, the Mega128 was well suited to the project for the following features. It had up to eight PWM outputs, which were necessary for controlling servos, as well as an eight-channel 10-bit ADC for reading analog sensors. While most microcontrollers have only one UART, the Mega128 has two, allowing control of two separate serial devices. A c compiler was available for this device, allowing us to avoid assembly and therefore more quickly develop the code for the flight system.

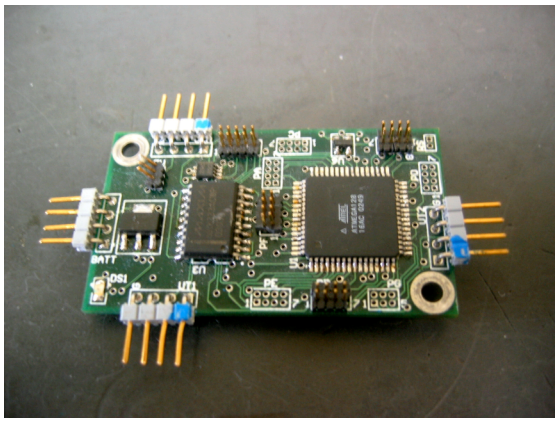

Figure 2-5: MAV128 R3

Development of an onboard flight computer began in the spring of 2003. Due to the fact that it was intended for small aircraft and built around a Mega128, the flight computer was designated as the MAV128. The first two versions, shown in Figure 2-4, were engineering prototypes and targeted for general research into MAV technologies as we determined the requirements of agile autonomous flight. The MAV128 R3 was specifically targeted for the AVCAAF system demonstrated in July (Figure 2-5).

We were unsure as to as to how well we would be able to isolate the flight system from the motors, and keep the electrical noise they generated from interfering with the system. Therefore, a ground and power plane were used with all of the flight computers. Using these planes also allowed us to cut down on the number of traces that we needed to layout on the design, and as a result minimize the size of the circuit board. We also saved space by using small electrical trace widths and clearance requirements. Aside from power signal traces, whose high currents required wider paths, all trace widths and clearances were at most 8 mils (.008 in.), allowing us to keep the devices close together. Even this specification proved to be too large, and we have since moved to 6 mils clearance and trace widths.

Due to weight and size considerations, surface mount devices and other small components were used whenever possible. While this was easy to accomplish in

procuring resistors, capacitors, and even most of the IC's, the matter of the connectors was more difficult. To conserve space, we desired small micro-headers of .05" pitch Some connectors had to remain at the standard header size of .10" pitch due to the fact that the smaller connectors could not handle the current flowing through the cables. There was also the possibility that external components such as the GPS, IMU, and RF Transceiver would need to be connected in the field, and this was more difficult with smaller headers. We were also unsure if they could handle the vibration of the plane. Therefore, we continued to use standard headers for power and connections to all the external components of the flight system (IMU, GPS, and transceiver). The ports on the Mega128 itself used the micro-headers, significantly reducing the board size of the MAV128. Small cables could then be made to connect to these headers, giving us access to the internal peripherals of the Mega128. This method was used for the programming port, but the cables were found to not be very secure, and therefore were not suitable for flight.

A later development was that of daughter boards, which used the micro-header ports to provide added functionality to the MAV128. Since multiple headers were used, the daughter boards were far more secure then individual cables. Two such boards were developed (Figure 2-6). The first was to allow the MAV128 to store the telemetry in onboard external flash for later retrieval. The daughter board connected four separate banks of 64 Mbit Atmel DataFlash to the SPI port of the Mega128. The major focus of the project shifted towards getting telemetry to the ground for developing a controller, and thus the data-logging capability was never used. The second daughter board allowed the Mega128 to control up to four servos, and interfaced to their internal potentiometers

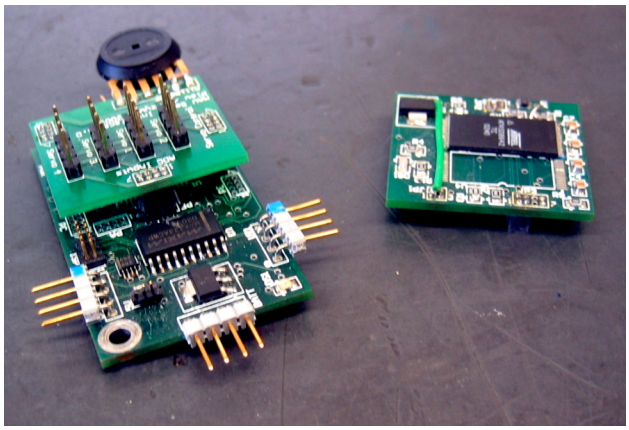

Figure 2-6: MAV128 R3 Daughter Boards

so that the MAV128 could record the actual servo positions. This allowed us to support the MAV used for the research, which included three actual servos as well as the drive motor, whose speed was controlled in a similar manner to a servo. The board also included the Motorola MPX4115A, a pressure sensor for measuring altitude, which could be read by the Mega128's internal ADC.

In designing the MAV128, we also had to consider the power requirements of this board. The principle power source for MAVs is currently Lithium Polymer batteries. The large three cell batteries provide around 12 volts, while two cells provide 7.4 volts. The major load on the battery, however, is the drive motor. The motor in our system requires a three-cell battery. This results in a flight time of around 20 to 30 minutes. Since the current draw of the onboard electronics is insignificant compared to the flight motor, we did not have to worry about whether the electronics would be the critical factor in the duration of a flight. Therefore, once again, weight was the primary factor. All of the electronics on the MAV128 operated at 5V, and so the battery voltage had to be regulated. If we were to use the same three-cell LiPoly battery as the motor, then the difference between the input and output voltages would have been significant enough that the thermal dissipation of the regulator could potentially result in the device overheating and shutting down.

While the problem of thermal dissipation could have been completely eliminated by using a Switching Regulator, such a device required external components, such as large inductors and capacitors, and therefore increased the size of the board. We had to keep the board as small as possible, and so instead we decided to power the electronics on a separate two-cell battery instead, allowing us to use a standard voltage regulator in a surface mount package, even though this resulted in a system with an efficiency of only 67%. Instead of a Linear Regulator, we used a LDO Voltage Regulator, which could operate even when the battery input voltage dropped to less then 6V. This allowed us to use the same electronics battery for multiple flights, and only have to replace the motor battery to get the plane back in the air. The choice for the primary voltage regulator of the system was a National Semiconductor's LM2940C. This LDO regulator could take the battery input and produce up to 1A of current at 5V.

Flight System and Integration

In the end, all of the individual components had to be combined to work together, and any issues ironed out so that a controller could successfully keep the airplane in the air and navigate properly. A major focus was on the data and power interface between the MAV128, 3DM-G IMU, Swift A2 GPS Radio, and the MHX-2400 RF Transceiver. A major issue was the fact that all three devices connecting to the MAV128 required a UART, but the Mega128 only had two UARTS available. Varying sources of power were also an issue. The 3DM-G required at least 7V, as it uses its own regulators. The MHX-2400 required 5V to operate, while the Swift A2 could only handle 3.3V. In the end, the RF Transceiver was used on UART 0, and the 3DM-G on UART1. Code was written for the Mega128 to allow one of the timer's output compare and input capture functions to operate as a software UART. This pseudo communications port could only operate at low

speeds of 4800 bits per second or less, but this was not an issue for the GPS, which could only update it's position every second. A standard connector was specified for all flight hardware using a UART to make it easy to debug the system. Pin 1 contained data from the MAV128, and Pin 2 data to the processor. Power and ground were designed to be on pins 3 and 4, respectively.

We hacked the 3DM-G's cable so that it could connect to this standard UART port. Interface boards were developed for the MHX-2400 and Swift A2 GPS so that they could also interface to the port. The Swift A2 only required 150 mA at 3.3V. Therefore, 5V from the MAV128 Regulator was sent through the UART2 port, and a LM3940IMP-3.3 voltage regulator from National Semiconductor was used on the interface board to convert it to the GPS operating voltage. The LM3940 was chosen because it was in the same package as the MAV128 voltage regulator, and was specifically meant for converting 5V to 3.3V. The interface board also had the special connector necessary to mate with the ribbon cable coming out of the Swift A2. As this cable was very small and flat, we decided it was too difficult to hack, and left this cable in the system.

While it was felt that the MAV128 regulator could handle the current requirements of the Swift A2 in addition to its own systems, this was not the case with the MHX-2400. The RF Transceiver could consume more then 550 mA, over half of the current that could be sourced by the LM2940. Therefore, another such regulator was used on the MHX-2400 Interface Board. The MAV128 therefore sourced the raw battery current to the IMU and RF Transceiver, and provide up to 1A at 5V to the processor systems and the GPS Radio.

With the connections between the main components of the flight system finalized, the code-base (written entirely in C) was developed to allow the Mega128 to interface to both the GPS and IMU, and send this data to the ground in a binary format, as well as receive commands from the ground station and adjust the servo values. The main program was called datamav. A single global structure, called mav_data, was developed that contained all of the data that was to be recorded from the sensors, and sent to the ground station. This allowed all of the various software modules to access one location for manipulating and transferring data, and make management of the code-base far easier. We developed a standard interface for interfacing to the UART, which was then used for all three communications ports in the system (including the software UART). The IMU, GPS, and RF Transceiver software modules all went through this interface to transfer data. Code was also written to read the servo commands from the RC system, and if necessary, use these commands to control the servo motors of the airplane. However, this function was not initially used for safety reasons, as it was considered necessary to completely bypass the flight hardware at this stage of development. The end result was the architecture seen in Figure 2-7.

After the flight system and code had been thoroughly tested in the lab, we began flight tests. The MAV128, 3DM-G, and Swift A2 GPS were mounted into a 24” MAV. As the effort to develop the ability to store telemetry onboard had been abandoned, the MHX-2400 was also used to send the Inertial and GPS data to the ground. The total weight of this system including the battery and Datalink antenna was 212 grams. The flight system sent telemetry to the ground station through the RF transceiver. A controller on the ground system was developed to use this data to send servo commands back to the

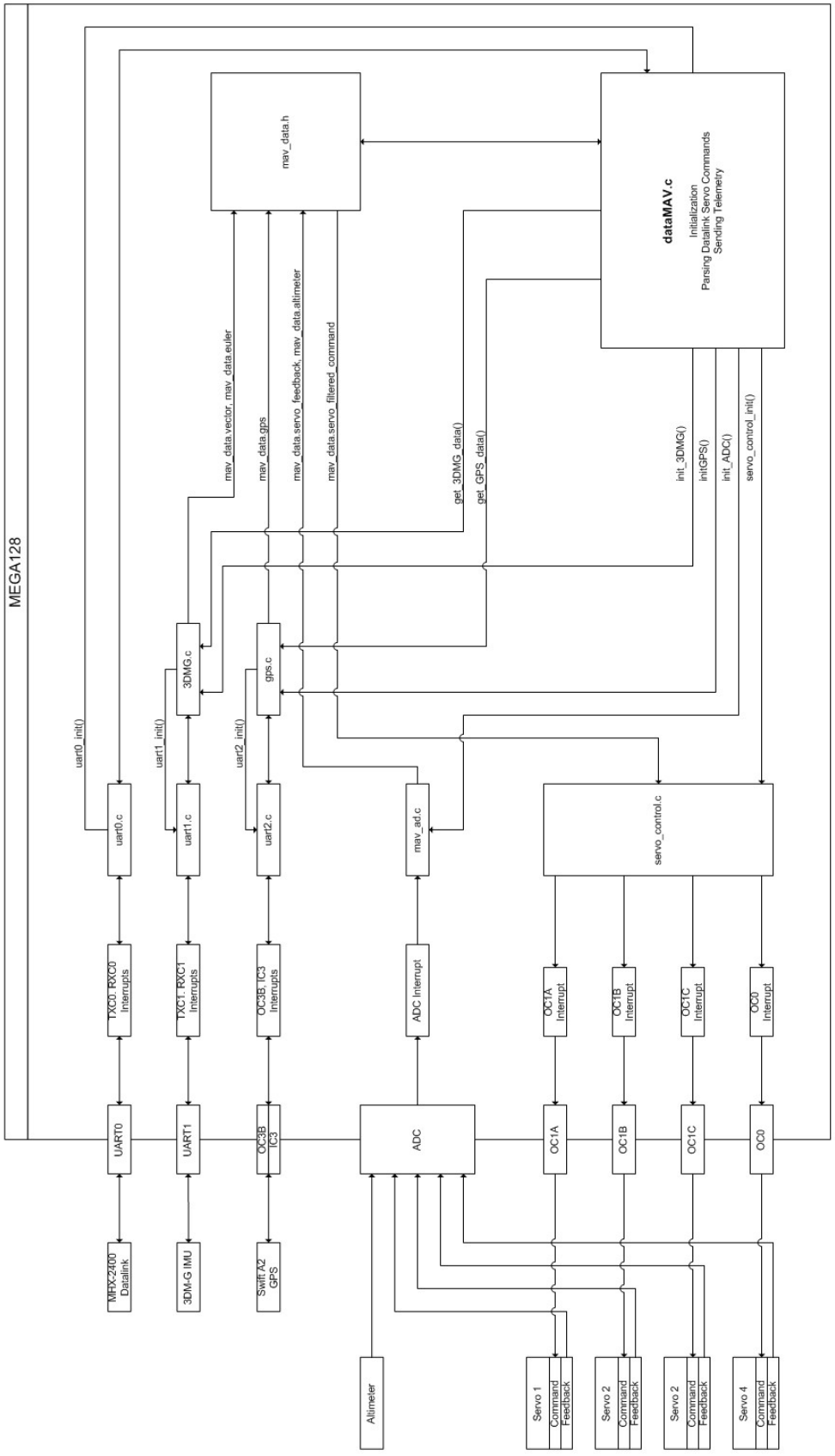

Figure 2-7: Revision 3 Flight System Software Architecture

MAV128, which controlled the servos. While the controller was being developed, the ground station passed through Servo Commands from a human pilot, allowing for open looping testing

Even with open loop control, we experienced problems. Though some of this was due to delays in controlling the aircraft through the ground station, another factor was the over all weight of the flight system. It was at the very edge of the payload weight that the 24” MAV could handle, and as such, the airplane was extremely difficult to control. Therefore, so that we could quickly start working on a controller, a 30” wingspan MAV (see Figure 2-8) was developed for testing the prototype flight system. Once we had successfully demonstrated autonomous flight, we could begin to focus on decreasing the weight of the flight system.

It was at this point that the method of sending commands to the servos was also changed, to eliminate the delays in controlling the airplane. The MAV128 just send data to the ground, while the Controller on the Ground Station sent Servo Commands through the RC System. This modification, along with using the 30” MAV platform, eliminated the controllability problems, and open loop testing began. Inertial Telemetry and GPS data were both sent back at about 30 Hz, which was fast enough for the Controls Group led by Professor Rick Lind to begin work on developing a controller.

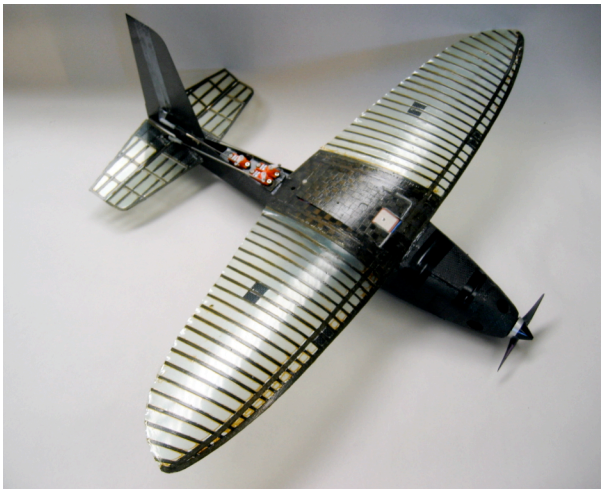

Figure 2-8: 30" MAV

More issues arose as open loop testing continued. The 3DM-G was set to output both instantaneous acceleration rates and the Euler Angles that determine the orientation of the plane. The data coming back from the IMU was incredibly noisy. It was initially thought that earlier crashes that had resulted from the controllability issues had damaged the 3DM-G, but when it was replaced with a second unit the same issues occurred. Efforts at filtering were unsuccessful in cleaning up the data to the point that a controller could keep the aircraft stably flying. There was a possibility that the mounting of the 3DM-G might be the issue. All of the flight system components were just placed inside the fuselage, with foam keeping them from jostling around in the middle of a flight. We thought that this might be too unstable for the 3DM-G, so wood inserts were used to physically mount the 3DM-G to the airplane. However, there was still significant noise in the 3DM-G data, such that the sensor was unusable.

This trend continued for the next few months as we approached the July 24th demo. All efforts to develop a controller using the 3DM-G for inertial control proved unsuccessful. Sources for this noise was thought to include EMF emissions from the data and video antennas, or possibly from the drive motor of the MAV itself. The latter seemed likely when bench tests showed perfect Euler Angle estimations from the 3DM-G until the throttle was turned on. However, moving the 3DM-G away from these possible sources did not help the matter.

Analysis by Jason Grzywna, Mujahid Abdulrahim, and myself found that a good deal of the problem was the vibration of the airplane itself, and its affect on the 3DM-G sensors. The MAV was put into a rig that allowed the throttle to be set to 100%, and yet hold the airplane in place. The plots in Figure 2-9 show the 3DM-G sensor outputs both

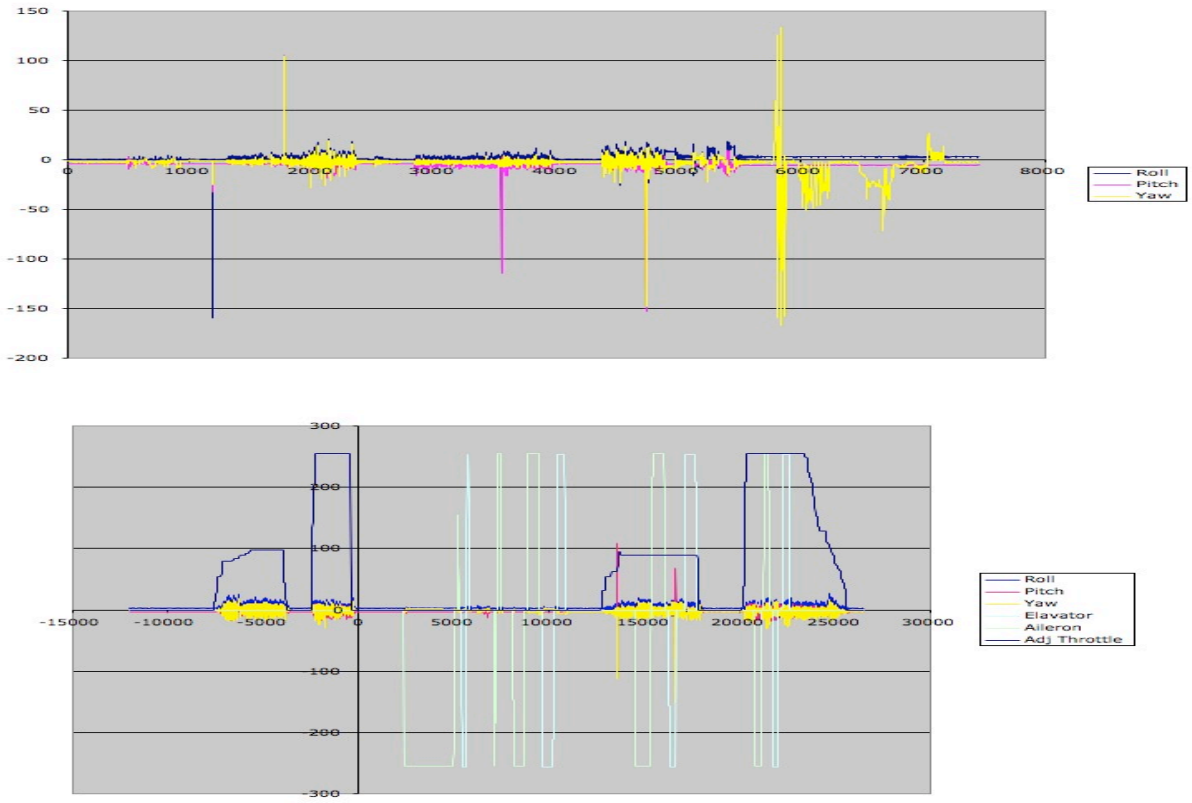

Figure 2-9: 3DM-G Benchtop Analysis

with the throttle active, and with the MAV undergoing vibration from external movement, with the motor inactive. The vibration appears in both scenarios.

We tried to develop some form of advanced filter that would enable us to get useful data out of the IMU, but with little success. However, a discovery was made that the 3DM-G did have the ability to produce filtered data. The IMU had the ability to use the onboard gyroscopes to stabilize the data. As the plot in Figure 2-10 shows, doing so immediately cleaned up the noise on the Euler Angles.

Even with this modification to the system, there were still issues with noise in the inertial data. Another major issue discovered was that the accelerometers on the 3DM-G could only handle gravitational force up to 2Gs. However, any extreme maneuvering

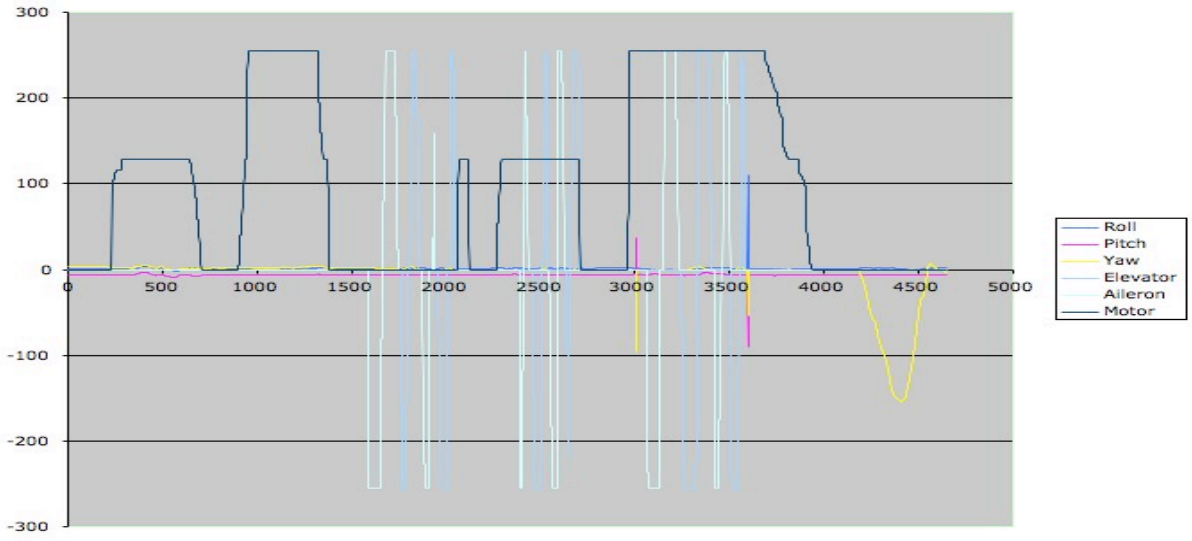

Figure 2-10: 3DM-G Benchtop Analysis with Data Stabilization

causes the aircraft to exceed these forces, and the 3DM-G accelerometer's to saturate. These excessive forces were mostly happening during takeoff, but also occurred during extreme maneuvers. While the instantaneous data would return to nominal values, the process by which the 3DM-G calculates Euler Angles uses integration, and therefore each successive case of high gravitational forces further distorted the IMU data. An attempt to reset the calculation process both manually and at a set rate proved ineffective.

The process of experimenting with the 3DM-G was not helped by the conditions. Multiple airplanes were destroyed during test flights, resulting in one 3DM-G being destroyed. The rest of the hardware fared somewhat better, though the wear and tear resulted in the small cable for the Swift A2 detaching from its connector inside of the GPS radio. As a result, the metal case had to be peeled away (there was no way to open the device) and wires soldered directly to pins on the internal circuit board of the GPS. The MAV128, however, came away from all crashes unscathed.

Prototype Flight System Development: Conclusions

Our inability to get adequate data out of the 3DM-G made it impossible to develop a controller by the time of the demo at Eglin AFB on July 24th 2004. This was the primary reason that we were unable to demonstrate autonomous flight. Instead, parts of the different MAV technologies currently being developed were shown, including a static display of the prototype flight system.

There were other issues as well. The Swift A2 GPS had proved to be inadequate for our requirements. Though it had the same weight as the 3DM-G, its large form factor made it difficult to handle, especially since it had to be mounted on the airplane so that it's integrated antenna could lock onto GPS satellites. Furthermore, the cable that it used to connect to the other devices was too fragile, and extremely difficult to replace.

Besides the IMU problems, we also had to consider the overall weight of the flight system. At almost 250 grams, it was simply too heavy for a 24" wingspan MAV to handle. Though a great deal of this weight was the RF Transceiver and antenna, we still needed to look into decreasing the weight of the other components as well. With aircraft at this scale, every gram counted.

The one major benefit to come out of the Prototype Flight System was the MAV128. The Flight Computer had proved to be more then adequate in handling the multiple tasks placed on it, and had been durable enough to survive crashes in multiple airplanes. As such, it was the one major component from this first system that continued to be developed, with new versions remaining at the heart of our future flight systems. Indeed, the MAV128 R3 has since been used in other MAV projects at the University of Florida. An R3 board was used in the ground station to provide an interface between the Ground Station Computer and an RC Controller. It also has been used for servo control

Figure 2-11: μMAV128 Flight Controller

and transmitting telemetry in larger aircraft for flight tests supporting MAV research. The technology was even moved into the PocketMAV, a small 12” wingspan MAV that is to allow for augmented control of a Micro Aerial Vehicle. The initial system actually used a MAV128 R3 with the addition of a daughter board containing a two-axis accelerometer, and the integration of the two PCBs and the removal of unnecessary components for the system led to the μMAV128 (Figure 2-11), a circuit board with dimensions .7” x .7” [15].

CHAPTER 3
REVISION 4: FLIGHT SYSTEM WITH ONBOARD GPS

With the July demo complete, attention turned to the next demonstration, which was supposed to occur in late October. We still had the goal of developing a flight system to enable autonomous flight and navigation of a 24" MAV. To accomplish this, we needed to both replace the 3DM-G and dramatically reduce the weight of the flight system.

It appeared that the only way to create a controller capable of stabilizing the aircraft during flight was to develop our own IMU system, directly integrated with the MAV128. However, doing so in less then three months was near impossible, especially since we also needed to also find replacements for the RF Transceiver and the GPS to cut the weight of the overall flight system, and develop the Revision 4 MAV128 to support the new system requirements. A temporary solution was to use the vision-guided stability system. Instead of having an IMU onboard for the October demo, the controller used an augmented version of the horizon tracking system to obtain the instantaneous roll and pitch angles. This data could then be used to keep the aircraft stable while flying. The new onboard flight system used the MAV128 R4 to obtain GPS data and send it to the ground for the controller to use in navigating the plane. Furthermore, the ground station also was used to develop new vision algorithms for objectives such as target testing. This allowed the Revision 4 system to be used as a testbed for both advanced vision processing systems and GPS-based navigational control [16, 17]. Once the October demo was complete, we could then start working on developing on the MAV128 R5, which

would include Inertial Sensors and be a drop in replacement for the new system. The Revision 5 board would allow the controller to be modified to take advantage of GPS, Vision, and INS, and fully demonstrate autonomous flight.

Flight System Components

Global Positioning System Receiver: Furuno GH-80D

There were multiple reasons for replacing the Axiom Swift A2 GPS Receiver. A major issue was it's weight, which was 28 grams. A related problem was the form factor. At 1.65" by 1.65", the Swift A2 took up a major portion of the top surface area of the MAV. Finally, it's connector had been proven to be unsuitable for the sometimes rough conditions of MAV flight testing, with the result being that there were several trips to the field where a lose connection resulted in no navigational data.

A search for a new GPS unit was begun. Another MAV grant at the University of Florida had come across a new GPS unit, the GH-80 (Figure 3-1) by a company called Furuno, and was having some success with it. After some investigation, we settled on the GH-80 for the Revision 4 Flight System. At .8" x .8" size, and with a weight of only 12 grams, its physical specifications were well suited for our needs. Furthermore, there was a version available with 10 small pins protruding from the bottom of GH-80 that could be used for interfacing [18]. A simple circuit was designed that allowed the GPS to be directly soldered to a PCB, and then connected to the MEGA128 through the standard 4 Pin UART Header. This completely eliminated the connector problems we had been experiencing with the Swift A2.

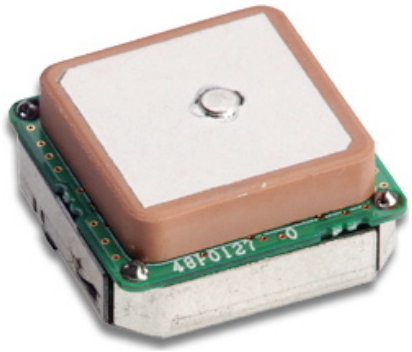

Figure 3-1: GH-80D

The protocol used to interface to the GH-80 was a manufacturer-defined binary specification, but was similar enough to SiRF that it was quite easy to rewrite the GPS interface code from the Swift A2 to work with the Furuno GPS instead. One issue that was discovered was that the GH-80D ignored any commands sent to it for the first few seconds after power up. Unfortunately, this information was not included in the preliminary datasheet, and as such it took awhile to determine why the GPS units were ignoring the commands being sent that set which data format the device should send. Once this issue was resolved by continuously sending the commands until there was a change in what data was g sent, the code necessary to interface the GH-80D to a MAV128 flight computer was complete.

RF Transceiver: Aerocomm AC4490-500

The Microhard MHX-2500 had been selected primarily because of their advertised ability to transmit at up to 115.2 kbps at 20 miles. Though they operated adequately, the excessive weight was a critical issue. However, after an exhaustive search nothing else could be found that met the requirements, and so the Microhard Transceivers were chosen for the Revision 3 Flight System in April 2003.

It turns out that we should have continued looking for viable alternatives. One month later, a company called Aerocomm started publicizing its new 900 MHz RF Transceiver, the AC4490 (Figure 3-2), which could also communicate at 115.2K. The device came in a few different versions capable of transmitting at different power levels. The AC4490-500, which was the most powerful, had a range of 20 miles, more then adequate for our needs. Furthermore, the device was 1.9” x 1.65” at a weight of only 12 grams, and therefore less then half the size and 1/7 of the weight of the MHX-2400 [19].

There were of course issues. One was the fact that the AC4490 was a very new device. Fully detailed specifications on the device were not yet available, and so some of our work in making the transceivers function in our system was trial and error, since the necessary information was not always at hand. Furthermore, the AC4490-500 was not yet available. While it was initially supposed to be ready in the summer of 2003, it ended up being delayed and we were not able to get the first two modules until late August. As a result, we initially started using AC4490-200 RF Transceivers instead. They were identical to the AC4490-500 modules except that they transmit at a lower power, which results in a maximum range of four miles. For initial testing, this was not an issue.

Figure 3-2: AC4490 RF Transceiver

We started with development kits for the transceivers. However, we ran into some problems in testing the devices, which did not operate very well in the lab. We assumed that the issues when testing the radios under these conditions was the result of interference from other devices in the lab also running at 900 MHz, as well as from the fact that we were working in confined rooms with walls that probably contained some metal shielding. As a result, radio reflections were likely to occur. In any event, we found we had better results when we kept the transceivers in separate rooms, and much better performance once we moved outdoors.

There was the matter of determining how to set up the modules. Fortunately, Aerocomm preset most of the necessary parameters for optimum performance depending on the scenario the AC4490's were operating under. There was still the issue of setting up the network. First of all, one AC4490 had to be set as a server, and the other as the client. We also had to determine whether the server radio should be on the MAV, or stay with the ground station. Furthermore, we needed to determine whether the master radio should just communicate to the client AC4490, or broadcast to the world. If we chose the former, every time we changed clients, we would have to modify the configuration of the server radio with the new client address.

There was also the matter of whether the radios should be set to stream mode, or acknowledge. Stream mode could actually result in a faster throughput, but as a result, data packets were sometimes broken up. This latency in the received data appeared to cause issues in remaining in synch with the ground station, and result in corrupted telemetry. Furthermore, the radios would not retry sending corrupted packets in this mode, and so errors could occur [20]. Therefore, acknowledge mode was selected. We also adjusted the baud rate down to 19.2K, as sending so much data at higher baud rates appeared to result in dropped packets as well.

Broadcast mode also seemed to result in dropped packets, and so we set the server radio to communicate with the single client. We also noticed issues where the client radio occasionally lost synch with the server. While the device usually reacquired the synch, at other times the client needed to be reset. As this was impossible to do with the MAV AC4490 when it was in the air, we chose to locate the client device in the ground station.

We had begun to gain an understanding of the devices when we received our first AC4490-500s. However, we then ran into problems because of the fact that the higher power radios were not compatible with our development boards. The AC4490-200 was available in both 5V and 3.3V versions. We were using the 5V versions to have less of a voltage drop when converting from the battery voltage to the RF Transceiver voltage. However, it was then announced that the AC4490-500 would only be available at 3.3V. The development boards Aerocomm had originally sent us were only an initial design that could not support 3.3V modules. Fortunately, we were able to obtain new development boards from Aerocomm that had much greater functionality, including the ability to interface to our new RF transceivers. We tested the range of the AC4490-500s and found that they were roughly equivalent to the old MHX-2400s. Satisfied, we now focused on the development of the MAV128 R4, and the integration of all of these components into the flight system necessary for the October demo.

The MAV128 R4 and Flight System Integration

The MAV128 R4, whose design began in early July 2003, grew out of a desire to combine into a single design the MAV128 R3 and the daughter board that added servo motor support and an altimeter, as well as integrating any necessary interface boards for the new GPS and RF transceiver devices. Since the first two prototypes (there were three different versions of the MAV128 for this version of the flight system – R4A, 4B, and 4C) were created before it had been finally decided to drop the 3DM-G, both still supported the IMU. The final version of the MAV128 R4 removed 3DM-G support. Thereafter the RF transceiver was located on UART0, and the GPS on UART1. This had the added benefit of removing the need for the software-based UART2, and as a result the output compare and input capture timing resources formerly reserved for the UART

became available for other functions. The latter was especially important, since it reopened the possibility of reading servo commands from the onboard RC equipment.

The daughter board had been created specifically so that the servos would not have to connect to the micro headers. Therefore, the servo ports on the R4 also used standard .1" pitch headers. Servo feedback was once again available using the Mega128's 10-bit ADC. The converter also supported the MPX4115A pressure sensor for measuring altitude, which was also integrated into the R4.

The program that we developed for the MAV128 R4 was actually rather similar to that of the Revision 3 system. The 3DM-G code was of course removed, and the GPS interface software modified to process the Furuno Packets instead of SiRF. With the removal of the software UART, the Mega128's input capture 3 hardware was available, and code that had been developed earlier to read servo commands from an RC aircraft receiver was reinserted into the program. In an attempt to eliminate synchronization problems that sometimes occurred between the MAV and the ground computer, the MAV128 only sent a packet of telemetry when it received a request from the Ground Station. However, the software mostly remained the same, as seen in Figure 3-3.

Most of the changes through the series of MAV128 Boards resulted from determining how to supply the AC4490 with power. Much of these issues were due to the fact that we had access only to preliminary data. The AC4490 included both 5V and 3.3V versions. We therefore designed the first MAV128 R4 board to supply 5V from the main voltage regulator onboard (Figure 3-4). A connector located on the bottom of the board allowed the AC4490 to connect directly underneath the MAV128, and communicate directly to the ATmega128.

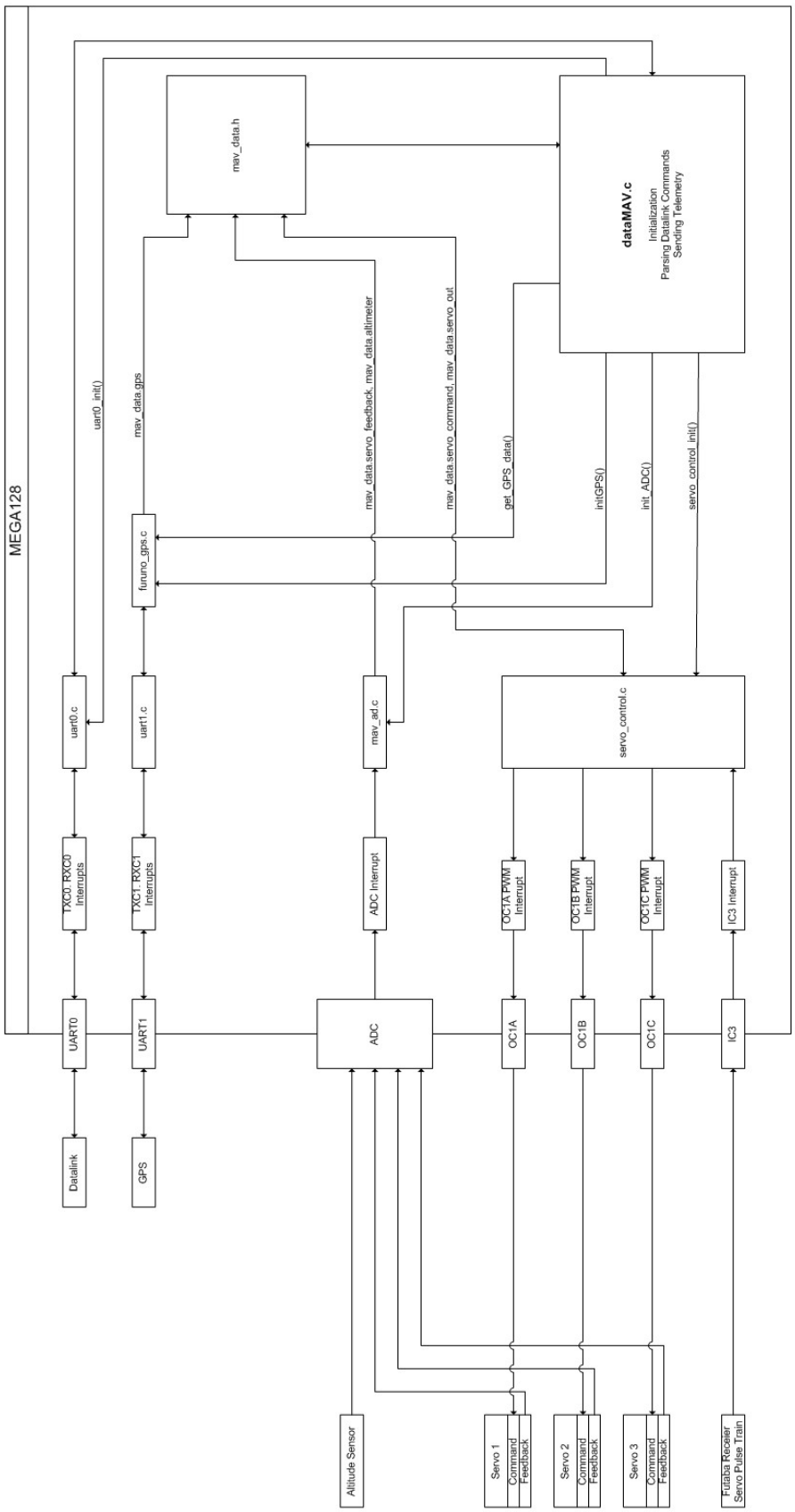

Figure 3-3: Rev. 4 Flight System Software Architecture

This procedure worked rather well for the AC4490-200, and the onboard supply was able to power the AC4490, the MAV128, and the GH-80, whose 3.3V supply was derived from the same source. However, it was then announced that only 3.3V version of the AC4490-500 were available. Given the fact that no power consumption data was available, we assumed that transmitting .5W at 3.3V consumed about 150 mA. We had already selected a voltage regulator for the GPS to replace the old LM3940, and decided to use it for the AC4490-500 as well. Selected for its very small size of 3 x 3 mm, the REG113NA-3.3 could source up to 400 mA. As the GPS itself could consume almost 90 mA, we thought it better to use separate regulators for the two 3.3V devices on the MAV128 R4B (Figure 3-4).

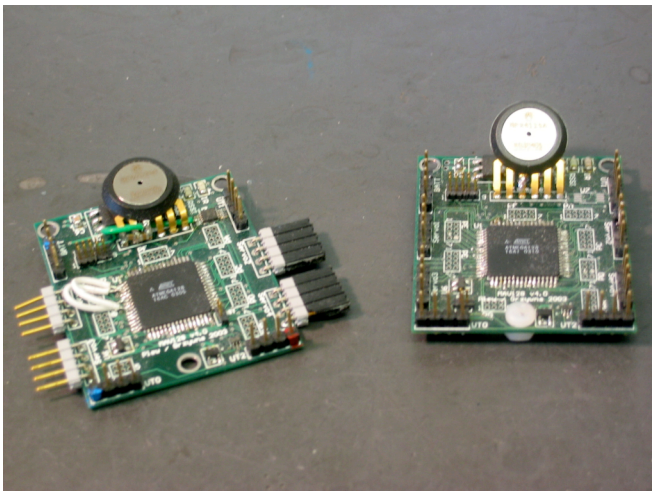

Figure 3-4: MAV128 R4A, R4B

However, we immediately had problems transmitting data through the AC4490-500 when using the MAV128 R4B to source the power. It only worked for a minute or two, and then only with intermittent communication. The In Range Signal on the Client AC4490, which should assert when it is synched with a Server, constantly switched on and off. We immediately suspected that the device was actually consuming more power then originally thought. An interface board was created using the old LM2940 / LM3940 system from the Revision 3 System to convert a battery source to 5V and then to 3.3V. However this system had the same problems.

Our attention then turned to whether placing the board underneath the MAV128 could be resulting in some type of electrical noise interfering with the operation of the AC4490. We occasionally saw indications that this could be a factor, such as moving the transceiver away from the MAV128 resulting in the device transmitting again. However, in most cases the position of the device appeared to have no affect on whether the RF transceiver was operating, and we decided that any indications of interference from the MAV128 being a factor was a coincidence. We were also unsure if maybe some signals that were dealt with on the development board, such as RTS and CTS, needed to be interfaced on the MAV128 Board as well for the device to operate properly. However, the AC4490-200 had not demonstrated any problem in this area, and the datasheet said that they could be left floating if unused. To verify, we made sure all of the inputs to the AC4490 other then TXD, transmit data, were deasserted. Doing so had no affect.

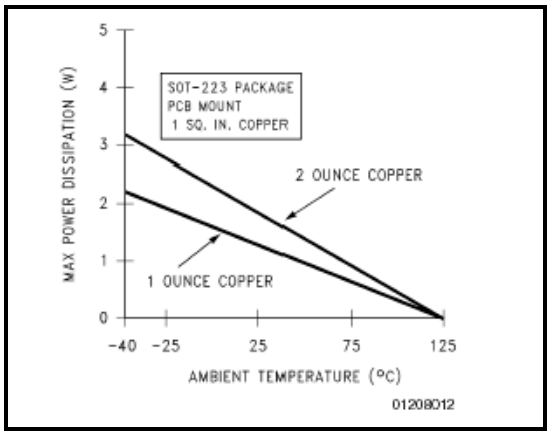

Figure 3-5: SOT-223 Maximum Power Dissipation[1]

Eventually, feedback from Aerocomm as to the actual power consumption of the device determined that the problem had all along been providing enough power to the AC4490-500. When transmitting 100% of the time, the radio consumed up to almost .5A. Though this explained why the RF Transceiver was not working when powered by the MAV128 R4B, it failed to explain why the interface board did not work.

[1] LM2940 Datasheet, National Semiconductor, 2003

We soon determined the cause however – we had failed to ensure the conditions used for the LM2940 / LM3940 Power System on the interface board were the same as the LM2940 / REG113 system on the MAV128 R4B. We had always powered the flight computer on a two-cell battery, with a source of around 7.4V. When we powered the interface board, however, we had used a bench-top power supply sourcing around 12V. While the LM2940 can handle an input voltage of over 20V, there is also the matter of being able to dissipate the thermal energy that is generated by the differential voltage between the input and output multiplied by the current being sourced. In this case, the LM2940 needed to dissipate about 3.5W. As seen in Figure 3-5, this was beyond the capabilities of the SOT-223 surface mount package we were using, especially since the device is meant to dissipate heat through the ground plane of the PCB. We were using a prototyping device to quickly create the interface boards, instead of having them fabricated and sent to us after a weeks delay. The drawback was that the circuit boards used only one or two layers, with no internal planes. As a result, the voltage regulator had little ability to dissipate thermal energy, and the load we were placing on it caused the device to shutdown to protect itself, and only turn back on when its internal temperature had returned to normal levels. Once we used a source of 7V for the AC4490-500 interface board, it worked properly.

With the proper information and the analysis of the power issues shown in Table 3-1, we determined the requirements of the AC4490, and what devices were required to provide power. Therefore, the LM2940 / LM3940 system was also used in the MAV128 R4C, the final version of the flight computer intended for the Revision 4 flight system (Figure 3-6). As always, a 2-Cell LiPoly battery was sued to power the electronics, and as

a result the AC4490-500 was able to transmit without the power system going into thermal shutdown. Later flight tests verified that the RF Transceiver could send data to the ground, though transmission speeds when sending about 50 bytes of data were limited to around 10 to 15Hz.

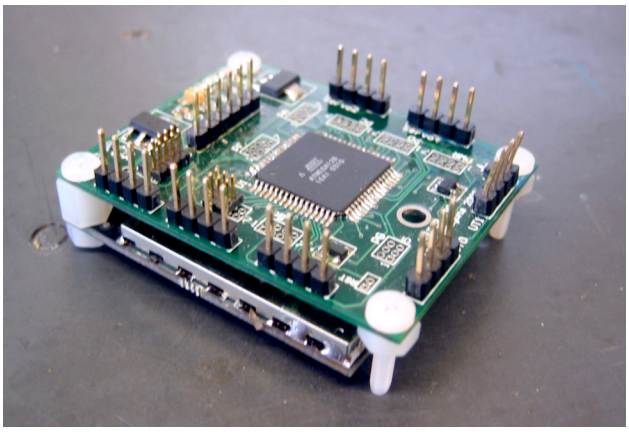

Figure 3-6: MAV128 R4C With AC4490

Table 3-1: Analysis of Thermal Dissipation Issues in Powering the R4 Flight System

Voltage Regulator Thermal Dissipation	$P_T = I * (V_{OUT} - V_{IN})$
AC4490-200 (5V) Current Consumption	106 mA
AC4490-500 (3.3V) Current Consumption	492 mA
MAV128R4 (5V) Current Consumption	10 mA
GH-80 (3.3V) Current Consumption	88 mA
Two-Cell Battery, MAV128R4A, AC4490-200, GH-80	
LM2940 – 5.0V	(7.4V – 5.0V) * (106 mA + 10 mA + 88 mA) = .4896W
GH-80 REG113NA – 3.3V	(5.0V – 3.3V) * 88 mA = .1496W
Two-Cell Battery, MAV128R4B, AC4490-500, GH-80	
LM2940 – 5.0V	(7.4V – 5.0V) * (492 mA + 10 mA + 88 mA) = 1.416W
AC4490 REG113NA – 3.3V	(5.0V – 3.3V) * 492 mA = .8364W
GPS REG113NA – 3.3V	(5.0V – 3.3V) * 88 mA = .1496W
12V Power Supply, AC4490-500 Interface Board	
LM2940 – 5.0V	(12V – 5.0V) * 492 mA = 3.44W
LM3940 – 3.3V	(5.0V – 3.3V) * 492 mA = .8364W
Two-Cell Battery, MAV128R4C, AC4490-500, GH-80	
LM2940 – 5.0V	(7.4V – 5.0V) * (492 mA + 10 mA + 88 mA) = 1.416W
AC4490 LM3940 – 3.3V	(5.0V – 3.3V) * 492 mA = .8364W
GH-80 REG113NA – 3.3V	(5.0V – 3.3V) * 88 mA = .1496W

With the power issues concerning the AC4490 complete, our attention turned to conducting flight tests of the integrated flight system, which included the MAV128 R4C, the Aerocomm AC4490-500, and the Furuno GH-80D GPS. Direct control of the servos

through the MAV128 was possible in two ways, with commands either sent from the Ground Station through the AC4490 datalink, or by reading the servo commands from the onboard RC Receiver through the Input Capture on the Mega128. However, we instead used the setup shown in Figure 3-8, with the servos remaining on the standard RC control system, and the ground station flying the MAV through the RC Controller.

Therefore, all the MAV128 R4C had to do was read in GPS Data, as well as take readings from the altimeter through the ADC. All of this data was then sent to the ground station through the datalink, where the controller combined it with the pitch and roll angle estimations from the horizon tracking system to fly the plane.

Given the fact that we had significantly reduced the weight of the flight system, it was decided to target a 24" platform again. A MAV that had been specifically designed for the AVCAAF project (Figure 3-7) was therefore used in our flight tests. We did notice some issues with the GH-80 GPS receiver once we began flying. At times, it took up to a few minutes to lock on to enough satellites to be able to determine a position. It was obvious that the issue was entirely with the GPS set, and not the MAV128 or AC4490, because of the fact that we were getting telemetry on the ground at all times in these situations. Furthermore, the timer on the GPS was being retrieved and sent to the ground, and it was constantly increasing, indicating that the GPS was operational and communicating with the MAV128.

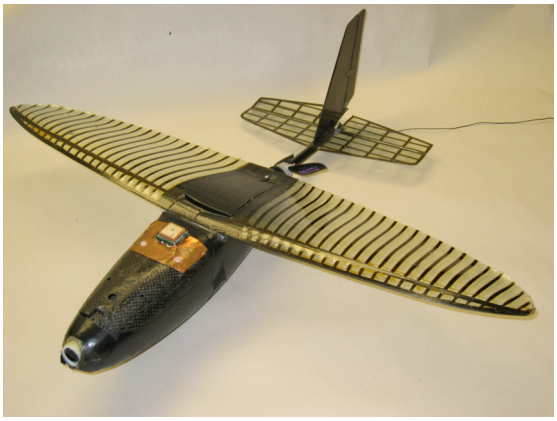

Figure 3-7: AVCAAF MAV Platform 1.0

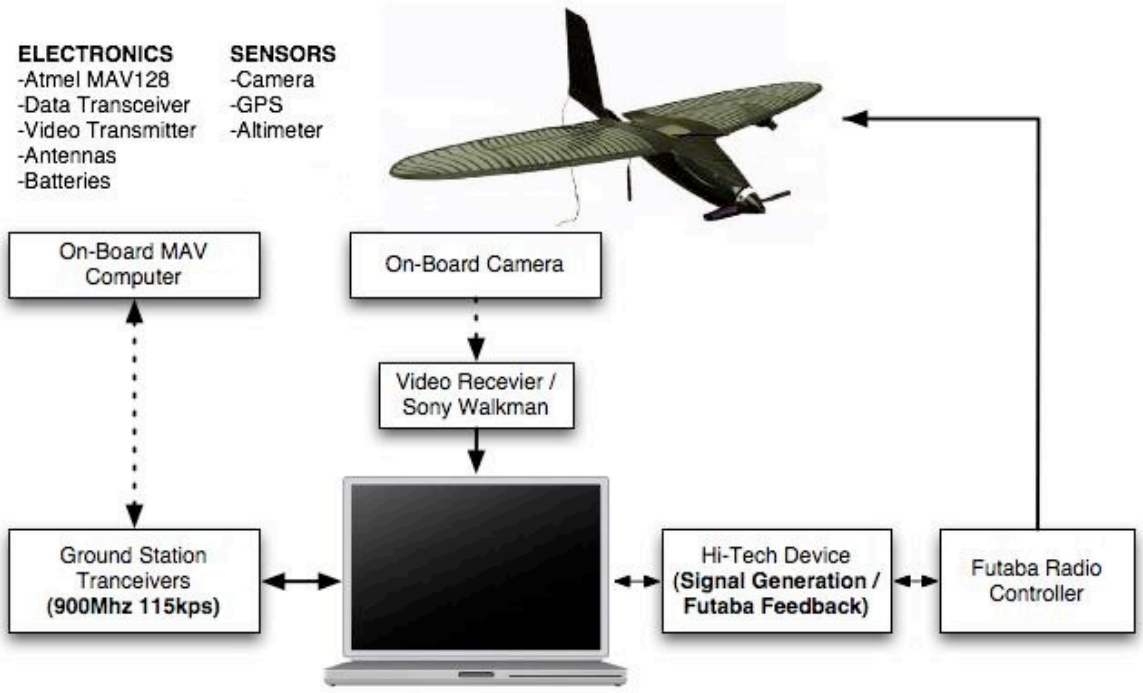

Figure 3-8: Revision 4 Flight Testing Setup

Since the time it took for the GH-80 to start providing accurate position data varied, we thought it was possible that the issue was the locations of the GPS satellites in orbit, and that at certain times they were positioned in such a matter to make it more difficult for the device to obtain a lock. Communication with Eglin AFB, where the device was also being used, included observations that the GH-80D needed a sizable ground plane beneath it to improve the performance of it's integrated antenna. However, a Furuno Field Application Engineer claimed that such a modification was not necessary, and though we did add copper plating to the hatch where the GPS Receiver was mounted, no real improvement was detected. In any event, it often took no more then a few minutes for the GPS to acquire a lock, and so the matter was dropped. As a precaution, the copper plating remained on the hatch for all of the Revision 4 flight system testing.

The other issue that was observed as flight testing continued was the sensitivity of the altitude sensor. This device registers the difference in pressure between vacuum and the outside environment, and could range from 15 to 115 kilopascals (kPa). However, this translated to being able to register an altitude in the range of several thousand feet, whereas we were only concerned with registering changes on the order of a few hundred feet at most. It was thought that the 10-bit ADC of the Mega128 was capable of giving the system the necessary resolution of a change in a few feet so that the MAV could hold an altitude. However, initial results indicated a resolution of around 40 feet.

Given the fact that the Mega128's 10-bit ADC had a resolution of about 5mV when using a 5V reference, we had the ability to register a change of about .1 kPa in pressure. However, this only translated to an accuracy of around 30 feet, as shown in Table 3-2. Furthermore, this was assuming that the ADC in the Mega128 performed with absolute accuracy. In actuality, noise from the processor core could induce errors in the results, and as such, our ability to read changes in altitude was reduced even further unless we were to shut down the core while taking analog readings. Obviously, this was not the solution.

Table 3-2: Pressure Sensor Conversion Data

ADC Sensitivity	$\left(Vref_{High} - Vref_{Low}\right) * \frac{1}{2^{10}} = .00488V$
Pressure Sensor Sensitivity	45.9 mV / kPa
ADC Pressure Sensitivity (APS)	$\frac{45.9mV}{1kPa} = \frac{5mV}{APS} \Rightarrow APS = .1089kPA$
Altitude vs. Pressure	Increase of 1 foot = Decrease of .003559 kPa
ADC Altitude Sensitivity (AAS)	$\frac{1ft}{.003559kPa} = \frac{APS}{.1089kPA} \Rightarrow APS = 30.598feet$

Instead, it was decided to modify the altimeter circuit. The MPX4115A was initially soldered directly to the board. One was removed from a MAV128 R4C, and a

header installed that could connect to a prototype board whose purpose was to increase the sensitivity of the altimeter system.

A few different methods were tried. One of the first attempts was to use operational amplifiers to multiply the signal. Before it was multiplied, a set bias voltage was subtracted from the sensor output so that the pressure range of interest (sea level to at least a few hundred feet) existed between 0 and 5V. The trouble was that as the gain factor increased, range of pressure detectable by the system decreased. The pressure at ground level could vary depending on location and weather conditions, meaning that if we set the gain factor too large, we risked being unable to detect changes in altitude under certain conditions. Though several gain factors were considered and used during experiments, a factor of eight was used in all flight testing, since it was the minimum factor to produce a theoretical resolution of 4 feet, and we were concerned that a further increase in gain would result in the ADC saturating.

To ensure that the system was simple, and that we could achieve the maximum range possible, we required small single supply operation amplifiers that could produce a true rail-to-rail output. Though such devices were difficult to find, we did have success with National's LM324 and Linear Technology's LT1006. There was also an attempt to bypass the Mega128 ADC and use a TI TLC3545 14-Bit Serial ADC, which then communicated with the processor through a SPI Port. It was hoped that the fact that this device was isolated from the Processor Core combined with its greater accuracy would result in better altitude data. However, using the TLC3545 did not provide any observable increase in accuracy over the original operation amplifier test system, even when the altimeter input signal was preamplified by the op-amps.

We therefore retuned to working with just the operational amplifier system of subtracting the bias voltage from the sensor output and then multiplying the result by eight. The initial design used a single operational amplifier to keep the size of the circuit down. However, we found that this approach allowed noise on the bias to have a major affect on the output. As we were using a resistor divider network to generate the bias, this was the signal most susceptible to noise. Therefore, we began using other op-amps to buffer the bias voltage, the sensor output, and eventually even the analog input to the Mega128 ADC, all in attempt to isolate the noise from the sensor system. However, we still had at most a performance of around 20 feet.

Neither amplifying the circuit by a factor of 8 through the use of the operational amplifiers, nor increasing the accuracy of the ADC by a factor of 16 (through the TLC3545), had been adequate in increasing the performance of the Altitude Sensor. Even combining the two systems had had no discernible affect. It was decided that if we were to assume that the sensor was capable of providing the resolution necessary, then we needed to further protect the altitude sensor from possible interference. The next step was to isolate the power for the Mega128 from the sensors. One 5V voltage regulator had to be the source for digital power and ground, another for analog. However, such a power system required a complete redesign of the MAV128. Therefore, it was decided to not spend time making this change for the Revision 4 Flight System, and instead this was incorporated into the many changes necessary to turn the MAV128 into a complete IMU system for Revision 5.

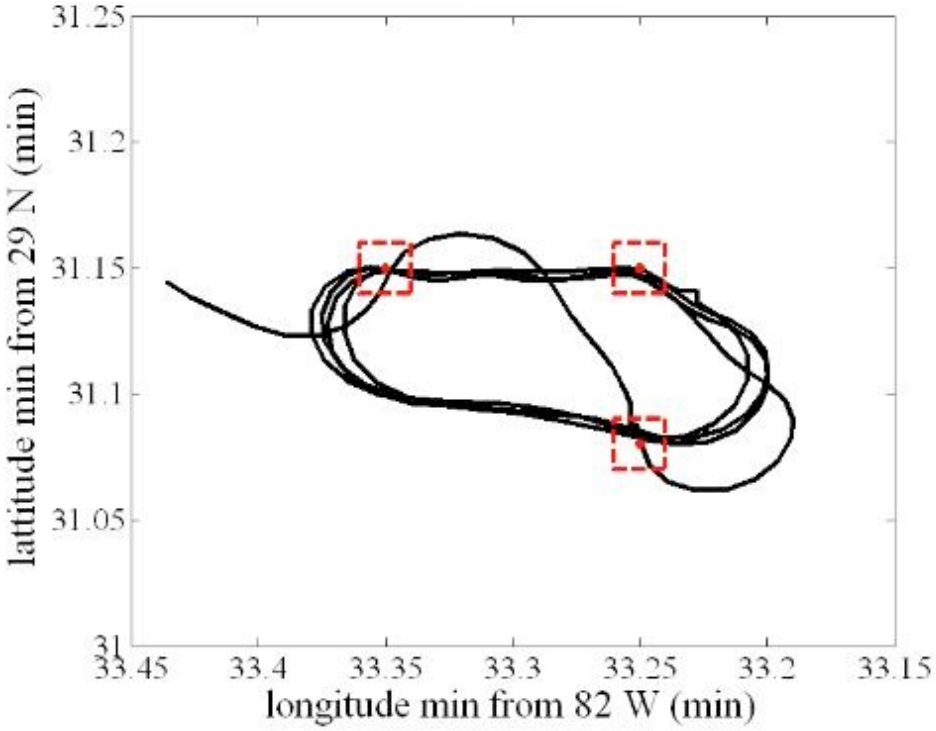

Figure 3-9: Rev. 4 Flight System Autonomous Waypoint Navigation

While the experiments on the altitude sensor system were being conducted with one MAV128 R4C, the other was being used in flight tests for developing the controller. Initial attempts to use the ground station based horizon tracking system to keep the aircraft stable were successful [21]. The controls team then moved on to creating the outer loop of the controller so that the GPS data could be used for waypoint navigation, or flying between different set points [22]. As seen in Figure 3-9, this controller allowed the MAV to continuously fly through a series of GPS waypoints, and therefore demonstrated our first autonomous MAV capable of performing missions.

Revision 4 Flight System Conclusions

We were able to develop a flight system that, when used with the pitch and roll estimated derived from the ground, demonstrated autonomous control of the MAV. All of the major systems used – the MAV128 R4C, the Furuno GH-80D GPS, and the Aerocomm AC4490 RF Transceiver – had proved to be more then adequate for the job. We had also successfully cut the weight of the overall flight system by more then half – from 212 grams in July 2003 to 86 grams by October (Table 3-3). Furthermore, the hardware was a major step towards developing the Revision 5 system with an onboard IMU.

Table 3-3: Flight Systems Weight Distribution (Grams)

System	Prototype		MAV128 / GPS		MAV128 / INS / GPS	
Flight Computer	MAV128 R3	16	MAV128 R4	10	MAV128 R5	16
INS	3DM-G	30	-			
GPS	Swift A2	28	GH-80D	12	GH-80D	12
Datalink	MHX-2400	86	AC4490-500	12	AC4490-1000	12
Datalink Antenna		26		26		26
Battery	Two Cell Li-Poly	52	Two Cell Li-Poly	52	Motor Battery	-
Total	Total	238	Total	112		66

One major issue that still remained was the problems we were having with the MPX4115A altitude sensor. We were never able to reliably increase its sensitivity, even when using the amplification circuits. This was a problem that had to be addressed as we moved on to development of Rev 5. For one thing, we needed to determine a way to detect small changes in the altitude of the MAV on the order of a few feet to be able to develop a robust controller. It appeared that when powered on the same supply as the digital components, the 10-b ADC on the Mega128 was not adequate for reading the sensor output of the MPX4115A. This meant that we might also run into issues as we integrated analog inertial sensors into the system as well. Therefore, the analog system of

the MAV128 was a major issue that would have to be addressed as we moved into the development of the Revision 5 Flight System.

CHAPTER 4
REVISION 5: FLIGHT SYSTEM WITH ONBOARD IMU, GPS, AND CONTROLLER

The Revision 4 system had been debugged and was being used in test flights, and development of an autonomous controller utilizing the vision system and GPS was now underway. We therefore turned our attention to developing the Revision 5 flight system. The overall architecture of the system remained the same, with the AC4490-500 being used for the Datalink, and the GH-80D providing navigational data. However, the MAV128 received a major upgrade, with the necessary changes made to support an IMU system onboard as well as continue with all of its other functions.

The issues we faced in developing the MAV128 R5 actually were rather similar to those faced in developing the Revision 4 system. And as before, it took three versions of the MAV128 R5 before we got all of the major problems sorted out. We had been asked to further decrease the weight of the overall flight system. The only solution was to no longer use a separate battery to power the electronics, but instead run off of the same battery that powered the MAV motors. The major issue with this was that the input voltage to all regulators handling the battery supply was now 12V instead of 7.4. This resulted in much higher thermal dissipation requirements then had been experienced with the two-cell LiPoly batteries.

We also wanted to ensure that the MAV128 R5 remained the same size as the R4C. However, we were adding a large number of new components to support the IMU. Therefore, some changes to the overall layout were made. Whereas previous flight computers used the ATMEGA128-16-AI, which came in a Thin Quad Flat Pack (TQFP)

package, we started using the ATMEGA128-16-MI with the Revision 5 computers. This device came in a Micro Lead Frame (MLF) package 1/3 the size of the QFP. To further conserve board space, the micro headers providing access to all of the individual ports of the MEGA128 were removed. We felt that they no longer had any real purpose, due to the fact that we had not really used them after abandoning the daughter board concept after the MAV128 R3. Since then, redesigns of the boards had been used to incorporate new functionality into the MAV128. Furthermore, there was also the fact that many of the port functions were already being utilized by the MAV128 hardware. The number of available peripherals inside the processor was rapidly decreasing, and so there was never any real reason to connect a device to most of the microheaders. We also eventually began to use the JST ZH/ZR connector system for the servo cables. Using these ports on the MAV128 saved a lot of room due the fact that they were half the size of the old connectors, though the drawback was that any MAV that was to be flown by the MAV128 had to have it's servos modified to use the JST connectors instead of the standard 0.1" pitch headers.

Even after we handled all of the problems with these transitions, there was still the matter of developing an analog processing system for the inertial sensors that was accurate enough for the controller. During the development of the Revision 4 flight computer, we had immense difficulty in creating a processing system just for the MPX4115A altitude sensor, and were not able to obtain the sensitivity that was needed. Now, instead of just one sensor to deal with, we had to deal with ten – acceleration in all three directions (x, y, z), angular rates around all three axes (roll, pitch, and yaw) and pressure sensors to obtain both altitude and airspeed.

The MAV128 R5 Power System

The initial goal of the power system for the MAV128 R5 was to separate the analog and digital power systems in the hopes of improving the performance of the analog processing system. Therefore, the MAV128R5A, shown in Figure 4-1, used separate voltage regulators for the digital and analog systems. The TI REG113NA regulator used in the MAV128 R4C for GPS power was utilized once again, since it was available in 5V versions as well. One regulator powered all of the 5V digital systems and the altimeter while both the gyroscopes and accelerometers each had their own 5V REG113 to provide power. This was a precaution to try and keep the sensors isolated from one another. A REG113 3.3V regulator once again provided power for the GPS. However, all of the regulators received their input voltage direct from the battery. Though the REG113 was a small device and not capable of dissipating more then a watt of thermal requirements, it could still handle the 7.4V of the two-cell battery.

The exception was the AC4490. Due to the fact that it required almost 0.5A there was no way a REG113 could provide it power. Furthermore, the LM3940 used previously could not handle 7.4V as an input, as it was targeted only for converting 5V to 3.3V. A new voltage regulator had to be found. The solution was the LMS8117A. Also available in a SOT-223 package, the device was barely capable of dissipating the heat we needed. However, we increased it's ability to transfer heat to the rest of the board and

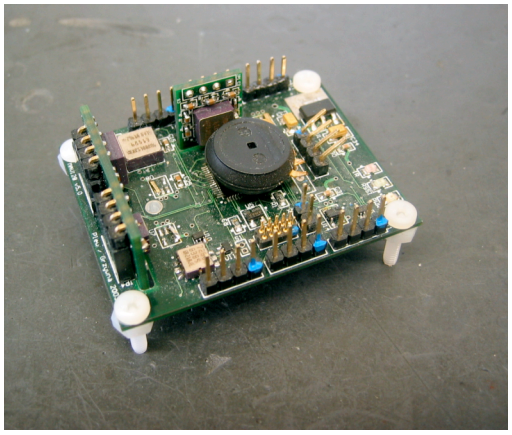

Figure 4-1: MAV128 R5A

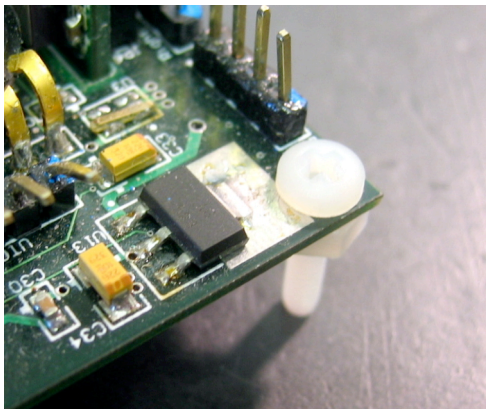

Figure 4-2: MAV128 R5A AC4490 Regulator

the surrounding air by increasing the size of the pad that it's main ground tab is connected to, and then removing the protective covering over that pad from the PCB design (Figure 4-2).

We then sent out the MAV128 R5A design to be fabricated. The first boards came back and were assembled right before we were told that we had to cut the weight of the flight system even more in R5. However, this was extremely difficult. A few grams might be trimmed from the MAV128, or we might possibly be able to find a smaller GPS unit (though using one might result in decreased performance), but neither attempt would really make a difference in the overall weight of the flight electronics. The 26-gram antenna used for data transmission was a generic device obtained from Aerocomm, and had a rubber shroud covering that was not necessarily needed for our application. We might have been able to reduce the weight by getting a custom antenna designed, but even then we would only be saving about 10 grams.

Therefore, our solution was to remove the dedicated electronics battery from the equation, which took up almost half of the weight of the flight system. However, as a result the electronics had to be powered off of the three-cell motor battery. The power system had never been designed to be capable of handling a battery input of 12V. As a test, we hooked up a 12V battery to the MAV128 R5A board, which had already been verified to work at 7.4V. Within a minute all of the voltage regulators connected to the

battery went into thermal shutdown. We therefore had to completely redesign the power system on the MAV128 R5A to support operation off of the motor battery.

We decided to use the same regulator for both the accelerometers and the gyroscopes, as it didn't appear that any real performance gain was being produced from separating the two sensors. Therefore, we needed two 5V regulators, one for analog devices, and one for digital. The 3.3V regulator that was to provide power to the GPS could exist underneath the digital 5V regulator, essentially acting as a second stage device. We still had the issue of the 3.3V regulator for the AC4490. If it were a challenge to enable a 5V regulator to receive 12V and source a large amount of current, then it would be even more difficult to do so with a 3.3V regulator. Therefore, it was decided that the voltage regulator for the AC4490 should not be connected to the battery directly, but instead receive its power from another voltage regulator, essentially acting as a second stage device.

Even with this layout, we still required several voltage regulators capable of dissipating a lot of thermal energy, including both the analog 5V regulator since it would have the battery for an input, and the AC4490 regulator since it would have to source a large amount of current. The digital 5V regulator would have to handle both of these situations, since it would be providing the AC4490 voltage regulator with power. However, the only way that we could ensure that these devices could operate and not go into thermal shutdown was to move to larger packages then the SOT-223 currently being used. Using three such voltage regulators would take up a lot of real estate on the MAV128 board.

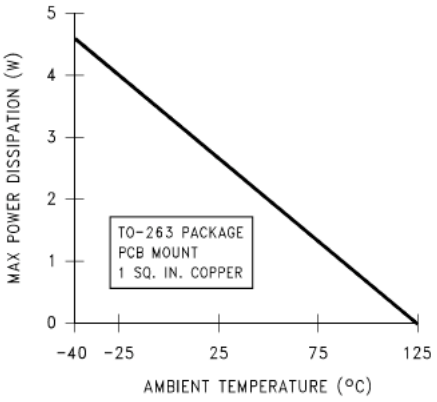

Figure 4-3: TO-263 Maximum Power Dissipation[2]

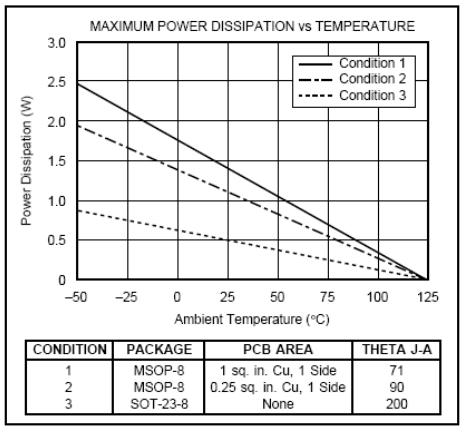

Figure 4-4: SOT-23 Maximum Power Dissiaption[3]

The solution was to use one single first stage voltage regulator to supply all current for the other devices on the MAV128 R5B. It took in the 12V source from the battery and produced 5.5V. The REG113 regulator, which has a dropout voltage of less then 400 mA, can easily produce 5V or 3.3V under these conditions. By moving to the REG113EA, which used a MSOP package, there was an increase in size of 167%, but a significant increase in thermal dissipation, as can be seen in Figure 4-4. Connecting all of the ground pins of the device to a small ground plane that was then exposed to the air was further insurance against the devices overheating. Under this setup, the REG113 could power the GPS, digital, and analog systems.

The AC4490 Voltage Regulator, however, required far too much current to use the REG113 Voltage Regulator, or even the LM3940 or LMS8117A used in previous systems. Now that the input to the regulator was 5.5V, even the latter device was incapable of dissipating the thermal energy resulting from providing power to the RF

[2] LM1085 Datasheet, National Semiconductor, 2003

[3] REG113 Datasheet, Texas Instruments, 2003

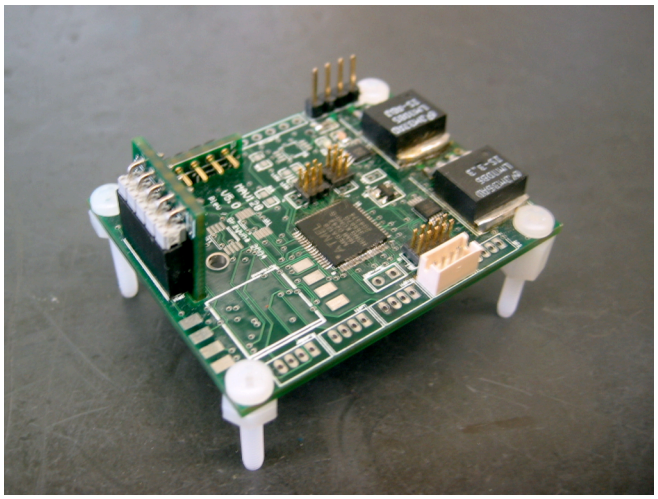

Figure 4-5: MAV128 R5B

Transceiver. Therefore, there were two voltage regulators that needed to be upgraded to a larger package. Using the analysis in Table 4-1, we determined that the solution was to use devices from National's LM1085 and LM1086 series of regulators. These devices, which met voltage and current requirements, were available in the TO-263 surface mount package. The package is over three times the size of the SOT-223 used previously, but it is also able to dissipate twice as much power as well (Figure 4-3, Figure 3-5). Therefore, a LM1086 3.3V regulator, which could source up to 1.5A, was used to power the AC4490. The LM1085 could source up to 3A, and was therefore well suited to power the rest of the MAV128. Since the first stage regulator required a non-standard output of 5.5V, an adjustable version of the LM1085 was used.

The evolution of the power system over the development of the Revision 5 MAV128 can be seen in Figure 4-7. With these changes, the Revision 5B was successful at powering the system when connected to a three cell battery (Figure 4-5). The first stage and RF transceiver regulators could dissipate the heat generated, though the temperature of the devices increased to near 100°C. The devices themselves could handle this heat, but they increased the temperature of the air around them significantly. The heat was too much for the AC4490, which was connected to the MAV128 on the bottom of the board

and therefore right next to the LM1086IS-3.3. Long-term operation resulted in damage to the RF transceiver, which no longer communicated and had to be replaced.

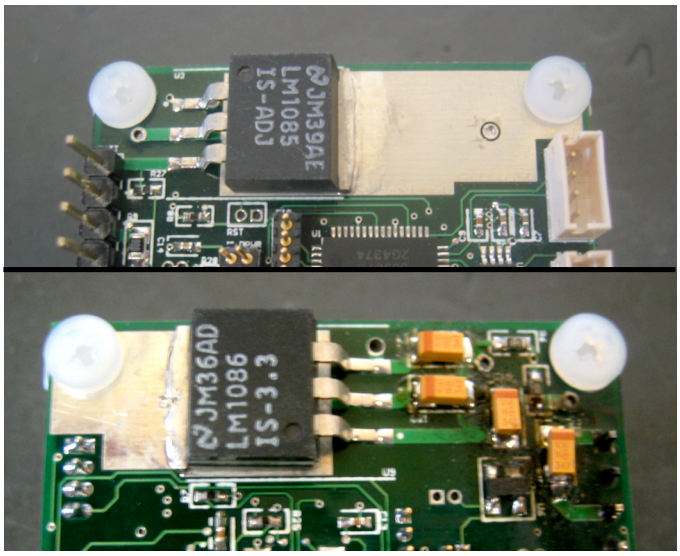

Figure 4-6: MAV128 R5C Power System: First Stage and AC4490 Regulators

Because of this issue, the layout of the power system was changed for the MAV128 R5C. The connector for the AC4490 was moved to the opposite side of the board, to keep the RF transceiver away from the heat of the voltage regulators. Significant surface area of the MAV128 was also used to create exposed copper planes connected to the main tab of the TO-263 regulators. (Figure 4-6) This increased the ability of the devices to transfer heat to the board and ambient air, and therefore lowered their internal temperature. This layout of the power system was verified on the benchtop to successfully operate for long periods of time with out shutting down due to heat. Though there is a detectable increase in temperature around the board, it has never risen to the point where the AC4490 is affected. The MAV128 R5C (Figure 4-8) board has since been successfully flown on a 24” MAV, and the power system was able to operate under flight conditions.

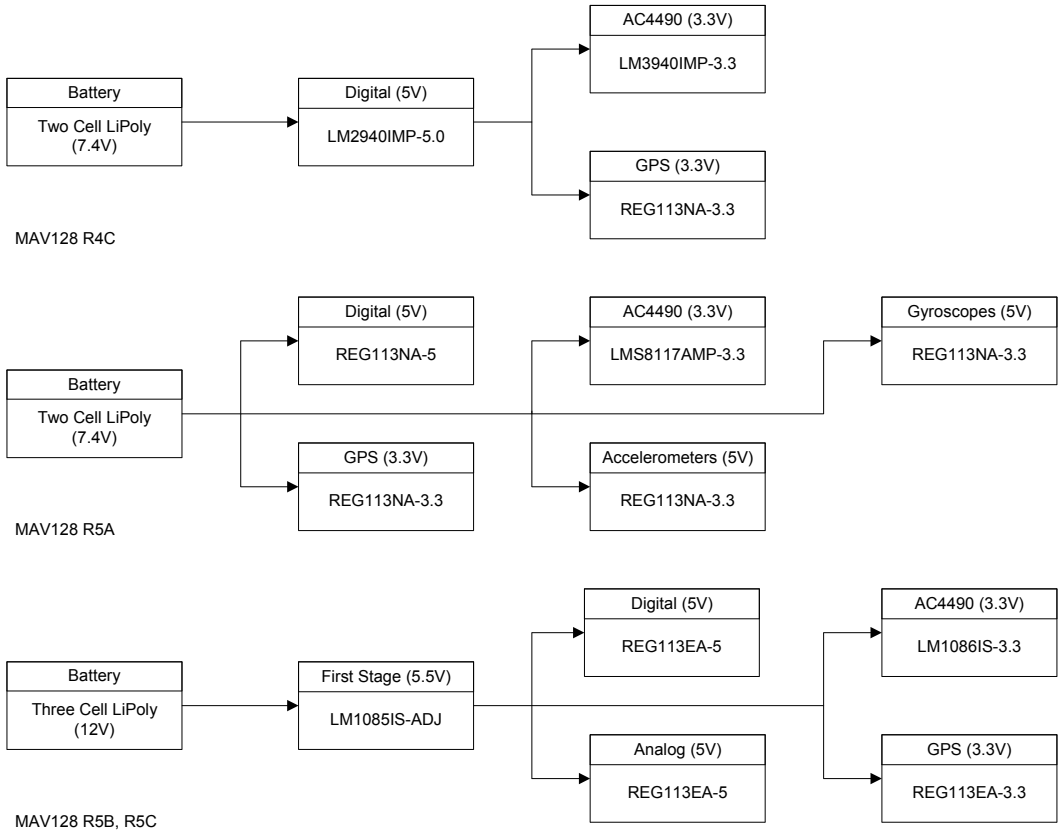

Figure 4-7: MAV128 Power Systems

Table 4-1: Analysis of Thermal Dissipation Issues in Powering the MAV128R5, GH-80, and AC4490

Voltage Regulator Thermal Dissipation	$P_T = I * (V_{OUT} - V_{IN})$
AC4490-500 (3.3V) Current Consumption	492 mA
MAV128R5 (5V) Current Consumption (Digital)	10 mA
MAV128R5 (5V) Current Consumption (Analog)	50 mA
GH-80 (3.3V) Current Consumption	88 mA
Three-Cell Battery, MAV128R5B/C, AC4490-500, GH-80	
First Stage LM1085 – 5.5V	(12V – 5.5V) * (492 mA + 10 mA + 50 mA + 88 mA = 4.16W
R5 Digital REG113EA – 5.0V	(5.5V – 5.0V) * 10 mA = .005W
R5 Analog REG113EA – 5.0V	(5.5V – 5.0V) * 10 mA = .025W
AC4490 LM3940 – 3.3V	(5.5V – 3.3V) * 492 mA = 1.0824W
GH-80 REG113EA – 3.3V	(5.5V – 3.3V) * 88 mA = .1936W

Development of the Inertial and Analog Conversion Systems

While we were determining the best way to ensure that the MAV128 R5 could survive on a three-cell battery, we were also focusing on developing the inertial measurement system that needed to be integrated into the flight computer. The major components that were required were accelerometers to measure the instantaneous acceleration of the aircraft in all three directions, and gyroscopes to determine the angular rates of movement about all three axes. The MPX4115A also needed to be migrated to the new system. A request was also given to include in R5 a sensor to measure the airspeed of the MAV. After some research, it was determined that the Motorola MPXV4006 pressure sensor was being used for this purpose in other autopilots [23]. We still needed to determine a way to accurately convert the analog signals of all of these devices into data accurate enough to be used in our controller.

Since as always size was an issue, we decided to focus on new accelerometers and gyroscopes developed using MEMS technology. Many IMUs that were intended for small platforms used accelerometers from Analog Devices, including the 3DM-G. We initially focused on using the ADXL210, which had a range of ±10Gs. The ADXL210, like most of the Analog Devices accelerometers, was dual-axis, and therefore we only needed two sensors to measure acceleration in all three directions. The ADXL210 provided digital pulses for outputs, with the duty cycle determining the acceleration it was measuring. However, the Mega128 only had two input capture signals, and so therefore we were forced to use a different method for recording the data. Fortunately, the ADXL210 could also provide an analog signal with the addition of a capacitor and an op-amp buffer.

There were a few options for the gyroscopes. The 3DM-G used the ENC-03J gyroscopes from Murata. However, we were informed that they were not available for military applications. Tokin also had SMD gyroscopes available, but they were not MEMS based. In the end, we again went with Analog Devices, using their ADXRS300 gyroscope. The device measures angular rates at up to 300°/s about one axis. One major issue was the fact that the device was only available in a small ball grid array (BGA). It was very difficult to attach this device to the board using the equipment we had available. We had some success in using a SMD Soldering Station, focusing its hot hair on the underside of the BGA as we lowered it onto the board, and allowing the solder on the bottom of the device to flow onto the pads. However, this process was never completely reliable.

Using these devices meant we needed two accelerometers and three gyroscopes to cover all three axes. One accelerometer was placed directly on the main board, and provide acceleration in the x and y directions. A gyroscope also was placed on the board to provide the yaw rate. There were also two daughter boards, with one containing an accelerometer and gyroscope to prove acceleration along the z-axis as well as the yaw rate. The other daughter board contained just a gyroscope to provide the pitch rate.

We had designed the MAV128 R5A with the goal of keeping the accelerometer and gyroscope sensors isolated from the digital power system. At the time we were also still working on the MAV128 R4C, attempting to develop a circuit board using op-amps to increase the sensitivity of the sensor. Therefore, like the Revision 4C, the new system included a connector to a separate circuit for amplifying the altitude data.

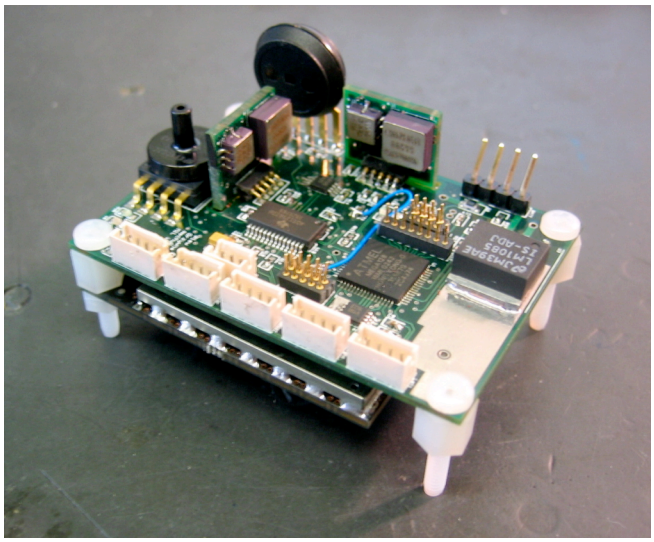

Figure 4-8: MAV128 R5C

By the time we started developing the MAV128 R5B to correct the issues in the power system, we had also realized through our experimentation with the altitude sensor that part of the problem was the affect the Mega128 processor core had on the internal ADC. Therefore, when we redesigned the power system for the MAV128 R5B, we also made an attempt to further isolate the analog systems from the digital. The ATmega128 did provide separate power and ground connections for the internal ADC. These signals were tied to analog power and ground, as were all of the inertial sensors. As discussed earlier, the digital and analog power systems used different voltage regulators. And starting with the R5B system, we kept the analog and digital grounds separate from one another. As soon as ground entered into the board at the main power connector, it connected directly to the digital ground. It also connected through a 0Ω Resistor to Analog Ground. This was the only connection between the two signals. This way, fluctuations on the digital ground plane due to the processor operation were less likely to affect the analog systems.

Another change made during the development of the MAV128 R5B was a modification of the layout of the inertial sensors. In the previous designs, one daughter board included an accelerometer and gyroscope, and other board just had a gyro. To reduce the complexity of the system and make manufacturing easier, a single daughter

board was designed containing both an accelerometer and a gyroscope. Two such daughter boards were used with the system, and provide all of the inertial data except the yaw rate. This was again covered by one gyroscope that remained on the main board.

We also went from using the ADXL210 accelerometers to the more accurate ADXL203s, a new sensor from Analog Devices. It also provided analog outputs instead of PWM signals, meaning the op-amp buffers were no longer needed. However, while the ADXL210 had a range of ±10Gs, the 203 only had a range of ±1.7Gs. Some of the issues with the 3DM-G in the earlier tests had been the saturation of its accelerometers under certain flight conditions, and those devices had a similar range to the ADXL203. However, the sensor was now being used on the PocketMAV Controller with some success. Therefore, we decided to use these sensors for the MAV128 R5B as well. One benefit of the new layout of the inertial sensors was that if there were problems with the ADXL203, it would be easy to go back to the old accelerometers by just replacing the daughter boards.

With the analog and digital systems completely isolated, we were hopeful that the amplifier system developed previously would be enough to obtain a sensitivity of a few feet from the MPX4115A. Therefore, the amplification circuit was migrated directly into the MAV128 design. However, tests of the MAV128 R5B showed no further improvement in the resolution of the altitude sensor. We were getting data from the inertial sensors, but there was no real way to tell if the results were accurate enough for autonomous control. It was therefore possible that the same problems we were facing with the altitude sensor were occurring with the other sensors as well.

Because of these issues, we decided to search for a high resolution external ADC that would interface to the Mega128. We hoped to further isolate the analog systems from the Mega128, and achieve greater sensitivity in all of the sensor data. The MAV128 R5C was designed to use this new ADC to convert the analog signals of the inertial sensors, as well as the altitude and airspeed pressure sensors. The Mega128 ADC was still used for low-resolution devices. This included the servo feedback systems, as well as a simple resistor divider circuit to drop the battery to a level that could be read by the Mega128. This allowed us to monitor the battery voltage and land the MAV if the battery was low. Finally, the Mega128 also read a temperature sensor that was included in the gyroscope. Since the inertial and pressure sensors were affected by temperature conditions, it was decided to include the signal in case it was needed for future controller development.

Our requirements for an external ADC included a serial interface through either the SPI Port or the I2C Bus. We did not want to deal with having to connect several traces between the Mega128 and the ADC, which could make the layout of the board more difficult, and so ADCs with parallel interfaces were not considered. We needed at least 8 channels to support the accelerometers, gyroscopes, and the pressure sensors, though the sensors could potentially be divided into multiple ADCs if necessary. It was felt that using a 14-bit or less ADC would most likely result in not enough sensitivity for our needs based on the previous experiments. Furthermore, it was found that often companies offer the same device in 12, 14, or 16 bit resolutions, with the only difference being the cost of the devices. Therefore, we only looked for ADCs with at least 16 bits of resolution.

In the end, we went with even higher resolution in choosing the TI ADS1256, an 8 Channel 24-bit Serial ADC, for the MAV128 R5C. With this much accuracy, any issues in the data were most likely to result from the sensors themselves or interference from other devices, not the ADC. Hopefully, the MAV128 R5C design had already dealt with these issues though the selection of the sensors and the isolation of the power systems. The ADS1256 interfaced to the Mega128 through the SPI Port. It included multiple features, including a low pass filter, a programmable gain amplifier, and also an input buffer that helped produce the large resolution for the ADC. However, the input buffer could only function with inputs of less then 3V. The accelerometers and gyroscopes produce 2.5V under null measurements, but could easily go above this limit. Furthermore, the measurement of the altitude sensor on the ground was about 4.1V. We therefore had to disable the input buffer, though we only sacrificed one or two bits of resolution as a result.

Once the details of interfacing the ADS1256 to the inertial sensors and the Mega128 had been determined, the MAV128 R5C design was sent to be fabricated. When the boards were returned, the parts were soldered onto the system. However, our method of attaching the gyro BGAs to the PCBs failed this time. The Yaw Gyroscope only produced a signal when it was heated up by the air gun of the SMD soldering station, and then only for a few minutes. Obviously, there was an intermittent connection on the bottom of the device. The solution was to instead have two complete MAV128 R5C systems assembled by a contractor. This ensured that all of the components were accurately placed on the boards.

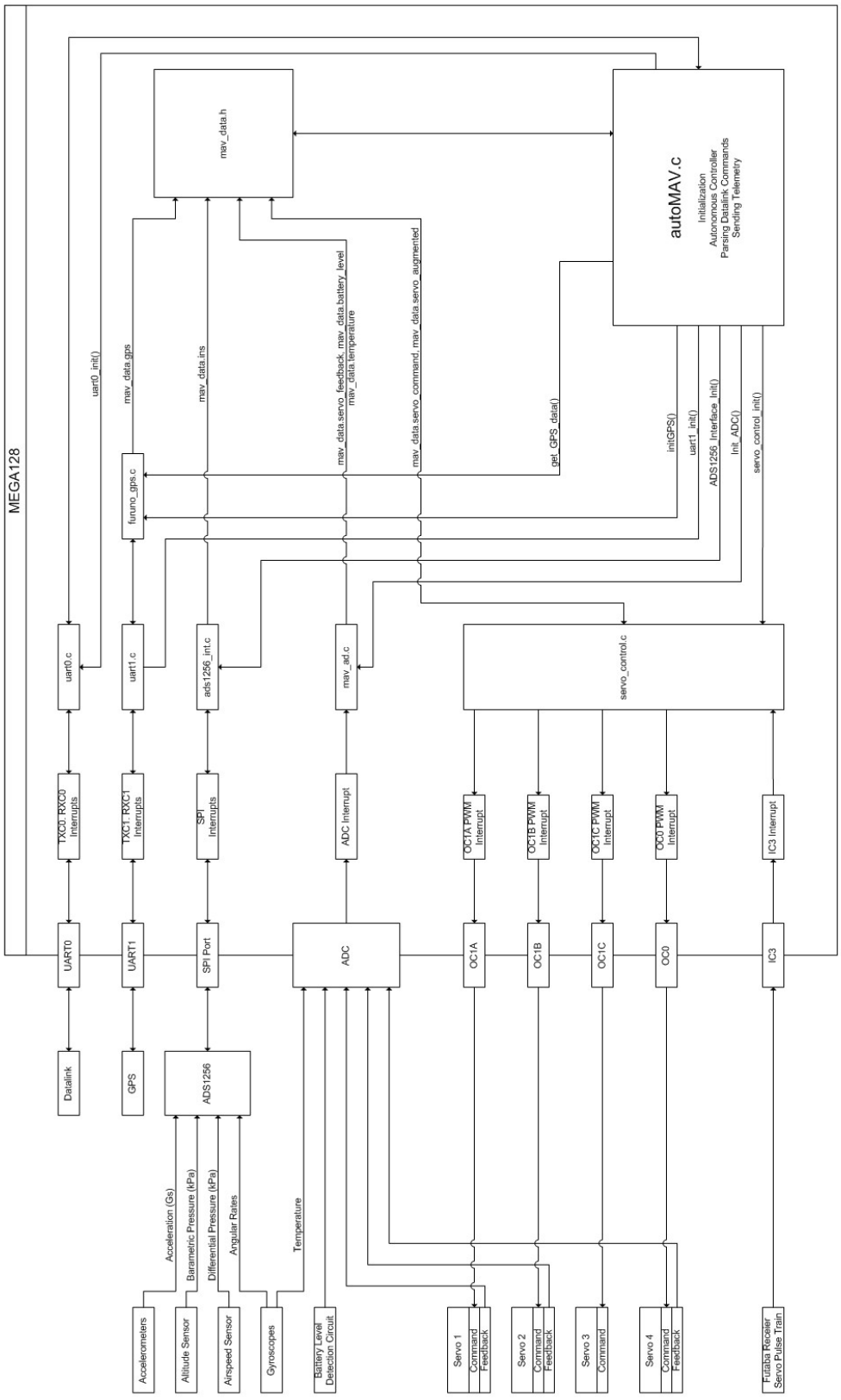

Figure 4-9: MAV128 R5C Software Architecture

With the MAV128 R5C PCBs fabricated, assembled, and returned to us, we set out to modify the code so that not only GPS data but also inertial telemetry could be recorded and sent down to the ground station upon request. The major change was the addition of the ads1256_int module, which used the Mega128's SPI Port to control the ADS1256, access the inertial sensor data, and store it to the mav_data structure. The overall program was changed from dataMAV to autoMAV, because for the first time since the early experiments of the prototype flight systems, we were going to attempt to have an onboard controller fly the airplane directly through the MAV128. The ground station was only to be used to monitor the situation.

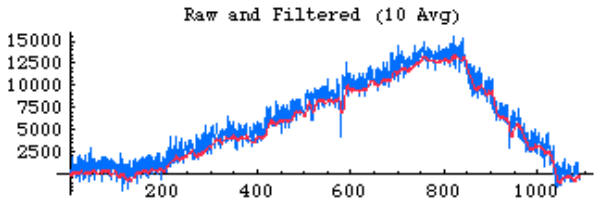

Figure 4-10: MAV128 R5C Altitude Data

Once we had finished developing the autoMAV program (Figure 4-9), we set out to test the MAV128 R5C on the ground. We already knew that the sensors were sending data, with the gyros showing no angular rates under static conditions. Furthermore, the accelerometers could only detect the force of gravity. However, until flight-testing began we did not know whether the ADS1256 and isolated analog power system would be enough to provide accurate inertial data for autonomous control. We did have the ability to test the sensitivity of the altitude sensor though. We carried the MAV128 R5C board up and down several flights of stairs, and then analyzed the telemetry using a low pass filter to determine the sensitivity of the system. Both the raw and filtered data are shown in Figure 4-10. We determined that the sensor now had a sensitivity of around 2.5 feet, due to the fact that we could actually determine the discrete changes in altitude that

occurred for every step taken. It appeared that our issues in converting the analog data were solved. We now had to test the system under actual flight conditions to determine whether the MAV128 R5C could be used as an IMU to autonomously fly a MAV

Flight Testing and Onboard Controller Development

By the time we were ready to begin flight-testing of the MAV128 R5C, the second version of the AVCAAF plane was ready (Figure 4-11). Therefore, the plane was modified so that the servos could be controlled directly by the MAV128. This method of control had not been attempted since the initial tests of the prototype flight system, and had been abandoned because of controllability issues. However, this functionality was necessary for developing an onboard controller, so our first flight tests were to verify the operation of the "fly-by-wire" system.

We soon ran into issues, however, with the servos intermittently rotating the control surfaces to maximum deflection. After some analysis, it was determined that the servo pulse, which should occur at a frequency of 50 Hz with a duty cycle of 5% to 10%, was occasionally not switching correctly. The reason for this was that the servo channels were using the output compare functions of the Mega128. When a Timer in the Mega128 reached a point equal to the compare register, the servo control pin toggled, and the

Figure 4-11: AVCAAF 2.0

processor would interrupt so that an interrupt service routine (ISR) function could run. This function would then update the compare register so that the system would activate again on the next transition of the servo command signal. However, with so many different interrupt-based systems now being used in the Mega128, occasionally an ISR would not have a chance to run before the next transition, and so the servo pulse would remain at its current level, causing the uncommanded movements.

The solution was to use the Mega128's PWM Channels, a different part of the Timer System, so that the processor could generate the servo pulses without relying on code. This had previously not been used because using the PWM system resulted in the loss of an input capture device. However, the peripheral in question was not being used, and it was necessary to move to PWM control if the fly-by-wire system was to work properly. Therefore, the necessary changes to the servo_control module were made. The move to PWM control eliminated the issues with the system, and the fly-by-wire system has since been verified.

With the MAV128 now flying the airplane, we began sending inertial data to the ground. With the addition of a 900 MHz gain antenna to the ground station, we were able to improve the reception of the datalink, though still only at a rate of about 25 Hz. This allowed us to have some view into the operation of the controller, which was now being coded inside of the autoMAV control loop. The autoMAV code was modified to convert the results from the ADS1256 to floating point numbers representing the actual voltage present at the inputs of the ADC. It could then be converted to the actual units of the sensor in question, as shown in Table 4-2.

Due to the fact that each gyroscope varied in its null voltage by ±.2V, autoMAV on startup also averaged the first 10 initial readings of each gyroscope, average them together, and used the results to calculate the null voltage of each device. However, this also meant that on power up, the MAV128 R5C needed to remain stationary.

Table 4-2: Conversion Formulas for Inertial Sensors

ADS1256 ADC	
0V	8388607 (0x7FFFFF)
2.5V	0
5V	-8388608 (0x800000)
Float Voltage Representation Conversion	$\frac{2.5V}{2^{23}-1} * (ADC_{result}) + 2.5V$
Inertial Sensors	
Accelerometers	
Null (0G Acceleration)	2.5V
Sensitivity	1 V/G
Gyroscopes	
Null (0° / s Rotation)	2.5V
Sensitivity	.005V / ° / s
Altitude Sensor	
Null (0 Feet)	≈ 4.1V
Conversion (Voltage to kPa)	$P = \frac{\left(\frac{ADC_{result}}{5V} + 0.095\right)}{.009}$
Airspeed Sensor	
Conversion (Voltage to kPa)	$P = \frac{\left(\frac{ADC_{result}}{5V} - 0.045\right)}{.1533}$

A simple controller was developed by Mujahid Abdulrahim , and then ported into the autoMAV program. A more advanced butterworth lowpass filter was used to further filter the data. Then, the results from the accelerometers were used to determine the gravitational vector, and from that data a state estimator could obtain the current pitch and roll of the MAV. The controller then used a simple proportional controller to order the servos to positions that would adjust the pitch and roll angles to 0, using the angular

rates from the gyros to control the movement. One initial result that came out of this work was that we now knew for certain that the Mega128 could run an inertial-based controller onboard. Whereas at minimum any control algorithm needs to run at about 30 Hz, the Mega128 was running the control loop at over 250 Hz.

Static tests in the lab showed that the state estimator could accurately determine the orientation of the airplane. However, the first flight tests were unsuccessful in that the MAV128 controller could not stabilize the MAV. Telemetry on the ground indicated that the issue was the accelerometers, which were showing extremely noisy data, even with the Butterworth low pass filter.

The accelerometers had a hardware low pass filter built in, with the cutoff frequency determined by the value of an external capacitor at the output of the sensor. We had initially used a .001μF capacitor, resulting in a 5 kHz bandwidth. As we were seeing a lot of high frequency noise in the accelerometer signals, the capacitor was changed to a .1μF capacitor instead, giving us a cutoff frequency of 50 Hz. The noise still remained, however. There was some though that the issues with the accelerometers was that they were being affected by EMF generated by the MAV drive motor. However, it was considered more likely to be caused by the vibrations of the airplane as it moved through the air. The MAV was therefore hooked back up into the jig previously used in the 3DM-G tests. Once again, these tests showed that vibration was a key factor, since the accelerometer results rapidly oscillated whenever the plane was being shaken, whether due to the drive motor or induced motion (Figure 4-12).

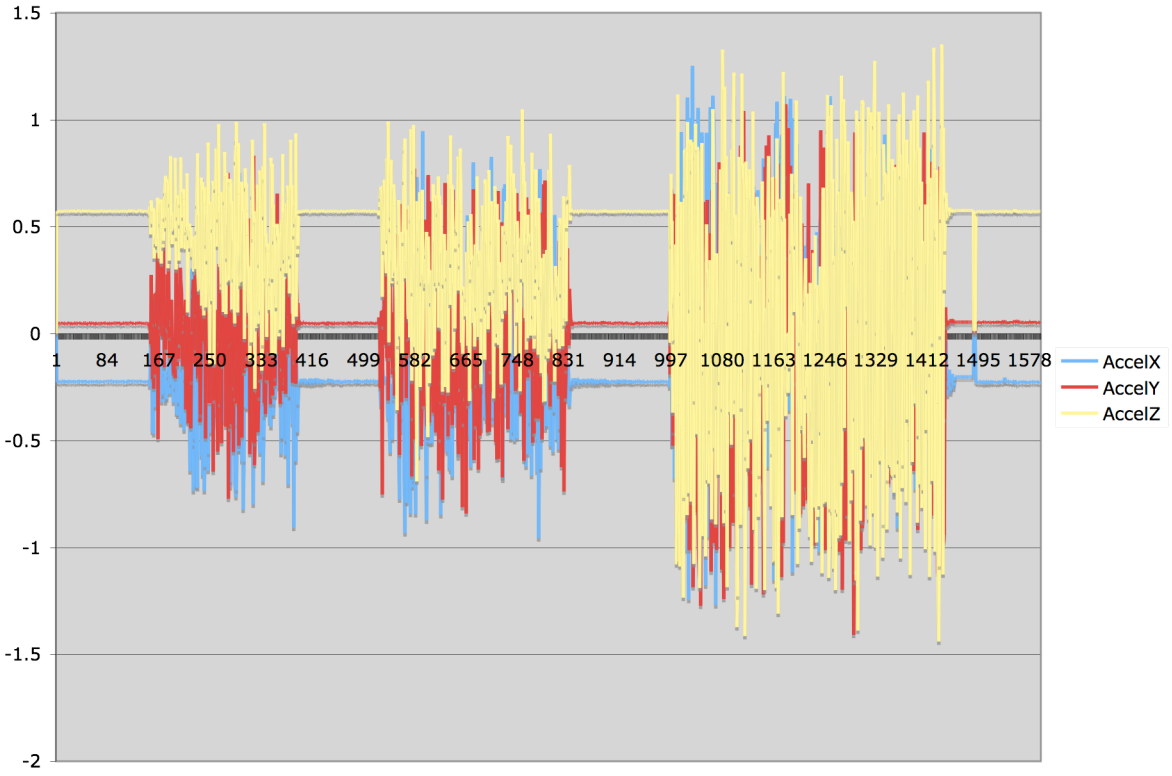

Figure 4-12: Accelerometer Data

Adjustments to the cutoff frequency of the Butterworth filter had little affect. We then came to the realization that the filter itself was not running properly. This was due to the fact that the results of the Mega128-based filter differed from that of a PC-based filter when the same raw data was supplied to both. We eventually determined that the issue was with the frequency of the controller on the Mega128. The control loop ran as quickly as possible, with the rate slightly varying as we inserted different instructions to try and debug the system. Furthermore, the rate was also affected during those cycles where a packet had been requested and was being sent to the ground station. We decide to modify the program so that the control loop always ran at a rate of 50 Hz. The timer in the Mega128 that was being used for capturing servo commands from the RC system was

used as a reference to stall the control loop at the end of a cycle until a full 20 ms had passed. Once this modification was made, the results of the Mega128-based Butterworth filter matched that of the ground. However, we still could not filter out the high frequency oscillations in the data caused by the vibration of the MAV. AS a result, the controller would not see the lower frequency changes in the gravitational vector resulting from the movement of the plane. The filters we are employing are still unable to provide us with the necessary data to allow the system to track the gravitational vector.

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

At this point we are still attempting to develop a simple onboard controller for the MAV128 R5 that will enable inertial-based autonomous control. A few test flights have been flown on a much larger RC aircraft. The resulting accelerometer telemetry, shown in Figure 5-1, shows none of the high frequency noise we have noticed on the MAV flights. Therefore, the issue with vibration appears to be directly related to the size of platform that we must control. After investigation, we have discovered that we are not alone in our difficulties to isolate the vibration of the aircraft. Other projects attempting

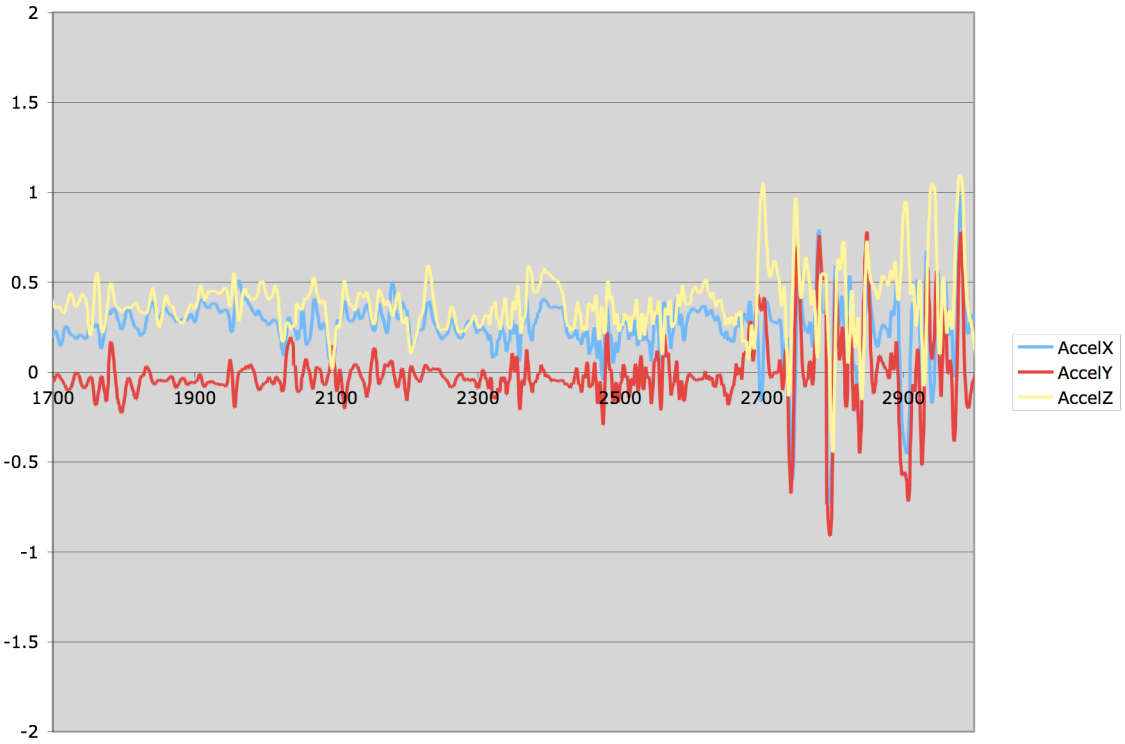

Figure 5-1: Accelerometer Results, 6 Foot RC Aircraft Platform

to develop autopilots for aircraft have often used a mixture of special mechanical mounts to dampen the vibrations of the avionics system, as well as Kalman filters to retrieve the necessary data from the accelerometers [24]. However, the former is dependent on the design of the aircraft, and the latter must be dealt with from a controls perspective. At this point, it appears that there is nothing more that can be done from the perspective of hardware development to further develop the MAV128. Given the right mounting system and the proper control methods, it will be possible for the MAV128 R5C to use its integrated IMU to stabilize the MAV in the air, and use the GPS system already developed in the Revision 4 platform to navigate.

With the combination of the MAV128 R5C Inertial and GPS Systems and the vision processing capabilities of the ground station that are even now starting to come online, the AVCAAF aircraft should be able to begin to perform increasingly complex maneuvers in open fields. This would not, however, necessarily translate to being able to do so in an urban environment. With the inertial based control onboard, the ground station is no longer necessary for aircraft stabilization. Furthermore, if the control algorithms for the GPS-based navigational systems are moved into the Mega128, the ground station interface could be simplified to just provide status information about the state of the aircraft, and allow the GPS waypoints used for navigation to be updated. However, since the MAV128 will never be able to provide the computational power for vision processing, this would have to remain in the ground station. As such, the MAV would still need to send a video signal to the ground to be analyzed, and receive some controller commands.

In the flight tests of the MAVs that have been conducted in an open field, video noise occasionally occurs, but the system is often able to compensate. However, in an urban environment, there may be several seconds or more of dropouts that could very well be fatal for the survival of the airplane. It will be very difficult to ensure a reliable link between the ground station and the MAV once the vehicle is flying amongst buildings.

Therefore, the next step in the development of the avionics flight system must be to start the process of migrating the vision processing functions of the ground station into the MAV. The functionality of the MAV128 is of vital importance to the ability of a MAV to survive in an urban environment, but it now must be relegated to one component of an overall MAV flight system that will take data from all three systems – inertial, vision, and GPS – and determine the necessary servo commands to continue its mission.

The complexity of this system will be far beyond that of the MAV128. The horizon tracking system, the first component of the MAV vision processing system developed, could successfully run on a 1 GHz processor only through the use of a hardware framegrabber. A DSP, or a processor with special functions for image processing, would most likely have to be used to have any chance of running the horizon tracking system onboard. Even by reducing the complexity of the algorithm, we would still face immense challenges in developing a traditional processing system that could accomplish this, given the size, weight, and power limitations of a MAV platform. Furthermore, having horizon tracking onboard will allow us to easily augment the inertial-based stabilization system with vision and produce a more robust controller, Similarly, there are far more complex vision processes that are just now coming online that must eventually be moved onboard

if we are to succeed in developing an autonomous MAV capable of urban operations [25, 26].

Fortunately, we have more options available to us then the traditional processing systems. While initially the PLD market was focused on replacing simple logic systems with a single device, the advances in Field Programmable Gate Arrays (FPGAs) over the past few years has led to growing interest over the ability of the devices to function as custom designs in multiple applications. The newest devices are so advanced that they are now being used to not only augment DSP systems, but also replace them entirely. Because of this, FPGA-based system-on-a-chip (SoC) technology is being using in many applications, including vision processing. Just recently, FPGAs have been used in Sony humanoid robots as a means for supporting stereo vision by running the necessary algorithms to combine the two camera signals for processing [27]. The ability to develop a custom digital system in an FPGA, as well as have this design perform multiple operations in a single clock cycle, make FPGAs well suited for performing the intensive pixel operations necessary in most vision processing applications.

Therefore, the next stage of flight computer development will be heavily focused on designing an FPGA-based system capable of performing horizon tracking. It will also have the processing power and capacity to handle more then just the vision-based stability system, so that other components of the ground station vision systems may be ported to the system. The same challenges faced in the development of the MAV128 in the past year and a half will still be major factors in the development of this new system, but emerging technology developed by the industry allowed us to move beyond those challenges and develop a flight avionics system capable of autonomous control for such a

small aircraft. As such, new developments coming in the next year or so will also allow us to add vision processing capabilities to the system, and move us closer to our goal of developing an autonomous MAV capable of urban operations.

LIST OF REFERENCES

[1] J. M. McMichael and Col. M. S. Francis, "Micro Air Vehicles – Toward a New Dimension in Flight," World Wide Web, http://www.darpa.mil/tto/mav/mav_auvsi.html, August 1997

[2] P. G. Ifju, S. Ettinger, D. A. Jenkins, Y. Lian, W. Shy, and M. R. Waszak, "Flexible-wing based Micro Air Vehicles," *40th AIAA Aerospace Sciences Meeting*, Reno, NV, AIAA 2002-0705

[3] P. G. Ifju, S. Ettinger, D. A. Jenkins, and L. Martinez, "Composite Materials for Micro Air Vehicles," *SAMPE Journal*, vol. 37, no. 4, pp. 7-13, July/August 2001

[4] S. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided Flight Stability and Control for Micro Air Vehicles," *Proc. IEEE Int. Conf. on Intelligence Robots and Systems*, vol. 3, pp. 2134-40, 2002

[5] S. Ettinger, M. C. Nechyba, P. G. Ifju and M. Waszak, "Vision-guided Flight Stability and Control for Micro Air Vehicles," *Advanced Robotics*, vol. 17, no. 7, pp. 617-40, 2003

[6] A. Kurdila, "Vision-Based Control of Agile, Autonomous Micro Air Vehicles and Small UAVs in Urban Environments Project Overview," World Wide Web, http://www.mil.ufl.edu/mav/presentations/kickoff_mtg/overview.pdf, AVCAAF Kickoff Meeting, UF-GERC, 27 October 2003

[7] L. Armesto, S. Chroust, M. Vincze, and J. Tornero, "Multi-rate Fusion with Vision and Inertial Sensors," *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 193-99, 2004

[8] 3DM-G User Manual, Microstrain, Inc., World Wide Web, http://www.microstrain.com/usermanuals/3DMG_usermanual.pdf, April 2003

[9] Swift A2 GPS Receiver Product Specification, Axiom Navigation, Inc., Costa Mesa, CA, 2002

[10] J. Grzywna, S. Kanowitz, P. Ifju, and M. Nechyba, "Integrating GPS with Micro Air Vehicles," World Wide Web, http://www.mil.ufl.edu/~number9/mav/, December 2002

[11] GPS Receiver Message Set Specification, Axiom Navigation Inc., Costa Mesa, CA, 2002

[12] CompactRF User's Guide, Microhard Corp., Calgary, AB, Canada, 2003

[13] MHX-910/2400 User's Guide, Microhard Corp., Calgary, AB, Canada, 2003

[14] ATmega128(L) Datasheet, Atmel Corp., San Jose, CA, 2004

[15] J. W. Grzywna, D. MacArthur, J. Plew, and M. C. Nechyba, "Evaluation of a MAV Inertial-based Flight Stability System using Vision Feedback," accepted to AIAA Intelligent Systems Conference, Chicago, IL, September 2004.

[16] J. W. Grzywna, A. Jain, J. Plew, and M. C. Nechyba, "Rapid Development of Vision-Based Control for MAVs through a Virtual Flight Testbed," submitted to IEEE Int. Conf. on Robotics and Automation, Barcelona, Spain, April 2005.

[17] J. W. Grzywna, "A Flight Testbed With Virtual Environment Capabilities for Developing Autonomous Micro Air Vehicles," Master's Thesis, University of Florida, 2004

[18] Specifications for GPS Receiver Model GH-80, Furuno Electric Co., Ltd. System Products Division, Camas, WA, 2002

[19] AC4490 Datasheet, Aerocomm, Lenexa, KS, 2003

[20] AC4490 User's Guide V1.7, Aerocomm, Lenexa, KS, 2004

[21] S. Jung, K. Lee, P. A. Barnswell, P. G. Ifju, J. W. Grzywna, J. Plew, A. Jain, and M. C. Nechyba, "Vision-based Control for a Micro Air Vehicle : Part 1 : Testbed," submitted to AIAA Conf. for Guidance, Navigation, and Control, Providence, RI, August 2004.

[22] J. Kehoe, J. W. Grzywna, R. S. Causey, J. Plew, M. Abdulrahim, M. C. Nechyba, and R. Lind, "Waypoint Navigation for a Micro Air Vehicle using Vision-Based Attitude Estimation," submitted to European Micro Air Vehicle Conf., Braunschweig, Germany, July 2004.

[23] Kestrel Autopilot 1.45 Description, Procerus Technologies, Provo, UT, 2004

[24] "autopilot: Do it yourself UAV," World Wide Web, http://autopilot.sourceforge.net/, November 2004

[25] S. Todorovic and M. C. Nechyba, "A Vision System For Intelligent Mission Profiles of Micro Air Vehicles," accepted by *IEEE Trans. On Vehicular Technology*, In press

[26] S. Todorovic, M. C. Nechyba, and P. G. Ifju, "Sky/Ground Modeling for Autonomous MAVs," *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, pp. 1422-7, September 2003.

[27] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle Avoidance and Path Planning for Humanoid Robots Using Stereo Vision," *Proc. IEEE Int. Conf. on Intelligence Robots and Systems*, vol. 4, pp. 3488-93, 2004

BIOGRAPHICAL SKETCH

Jason Plew was born in Bedford, IN, in 1979. He first began research into robotics when entering high school in Palm Bay, FL. Upon graduating high school in 1998, Jason began attending the University of Florida, where he started working at the Machine Intelligence Lab. During this time, he has also worked through summer internships at both P-Com, Inc., and Centerpoint Broadband Technologies, Inc. Jason later began to work at Prioria Robotics, Inc., a small startup company located in Gainesville, FL. In 2003, he graduated from UF with Bachelor of Science degrees in both computer and electrical engineering. He then began pursing a Master of Science in electrical engineering, using the MAV research he was conducting as the basis for this thesis.