CUAir: Cornell University Unmanned Air Systems
Stabilized Gimbal Control System

Phillip Tischler

Fall 2013

Contents

Executive Summary

This document describes the development of a Stabilized Gimbal Control System for the CUAir team. The Stabilized Gimbal Control System will help the CUAir team compete at the Association for Unmanned Vehicle System International (AUVSI) Student Unmanned Air Systems (SUAS) Competition [2]. The system developed will help the team accomplish the off-axis target imaging task, accomplish the egg-drop task, and improve the task of imaging targets below the aircraft.

This project consisted of the design and construction of the Gimbal Control Board, and the development of the software that executes both on the micro-controller and on the aircraft's payload computer. This project did not consist of building the physical gimbal that the Gimbal Control Board will actuate. The Gimbal Control Board contains an onboard 6-Axis Inertial Measurement Unit (IMU), headers for a GPS module, 5 servo connections with optical isolation, voltage regulation, and an Atmega128 micro-controller. The software developed provides a high performance asynchronous serial communication library, software to read and parse data from the GPS and IMU, software to control the servo signal lines that command the servo to actuate to certain angles, software to stabilize the gimbal system's pointing angle, and software to point the gimbal at a specified GPS position.

The first section of the document describes the motivation for the Stabilized Gimbal System. This section describes the AUVSI SUAS competition, the CUAir team, and why the Stabilized Gimbal System would be beneficial. The second section describes the overall system design. This includes the project requirements, an overview of the components and how they integrate, and how the system will eventually be integrated into the main CUAir system. The third section describes the design of the hardware. This includes the custom physical gimbal and the custom Gimbal Control Board. The fourth section describes the software that was developed for the micro-controller and for the host-computer. The fifth section describes the results of the project. Finally, the references and appendices are provided. The appendices include the printed circuit board schematics, the printed circuit board layout, images of the assembled gimbal board, a cost breakdown for the Gimbal Control Board, statistics on the code that was developed, and a breakdown of what tasks were completed by which members of the development team.

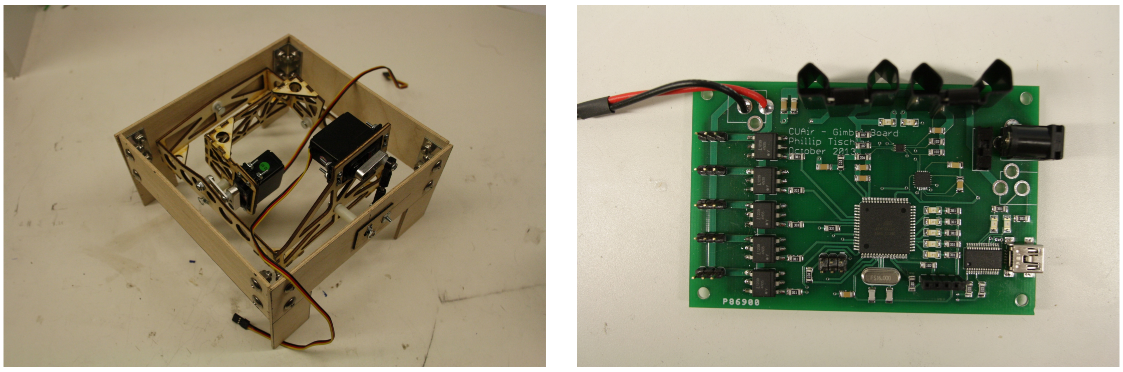


Figure 1: The physical gimbal and Gimbal Control Board.

1 Project Motivation

This section describes the motivation for the Stabilized Gimbal System Project. The first subsection describes the Association for Unmanned Vehicle System International (AUVSI) Student Unmanned Air Systems (SUAS) Competition [2]. The second subsection describes the CUAir Project Team, an engineering team at Cornell University that competes in the AUVSI SUAS Competition [10]. The Stabilized Gimbal System described in this document was developed for the CUAir team. The final subsection describes the motivation for the Stabilized Gimbal System, and how it relates to the AUVSI SUAS Competition and the CUAir team.

1.1 Association for Unmanned Vehicle Systems International (AUVSI) Student Unmanned Air Systems (SUAS) Competition

The AUVSI SUAS Competition is an annual competition in June that takes place at the Webster Field Annex of the Patuxent River Naval Base. The competition focuses on building unmanned air systems that can complete reconnaissance missions with real-world constraints. College teams from around the world attend this competition. Last year over 40 teams registered to compete, about 35 teams attended the competition, and about 32 teams flew their aircraft at competition.

Figure 2: CUAir at the 2013 AUVSI SUAS Competition

Competition Components. The competition is broken into three components: the technical journal paper, the flight readiness review presentation, and the simulated mission. The components are worth 25%, 25%, and 50% respectively. The journal paper is a 20 page paper that focuses on the design and testing of the air system. The flight readiness review presentation focuses on why the team is confident the air system will perform a safe and successful mission. The simulated mission is a sample mission where students are given mission parameters and expected to perform the mission within the given time-frame and mission parameters.

Figure 3: 2013 AUVSI SUAS Competition: CUAir at the Flight Readiness Review

The Simulated Mission. The simulated mission is broken into eight sections: mission setup, takeoff, waypoint navigation, search grid, emergent target, Simulated Remote Intelligence Center (SRIC), landing, and cleanup. Mission setup involves transporting the competition gear to a designated site within 5 minutes, unpacking and settings up the gear within 15 minutes, and then beginning the mission. Teams are not allowed to turn on aircraft components or wireless gear during this setup phase. After the 15 minutes of setup the mission starts, which means students have 30 minutes to complete all simulated mission components. The first component is autonomous takeoff, which is usually attempted once ground checks have been completed. The second component is waypoint navigation, where the aircraft must follow a specified flight path within a certain tolerance. The third component is the search grid, which is an area of abnormal shape which may contain targets of interest. The air system must stay within the search grid at all times, and must enter and leave the grid at a specified location. The judges will at some point indicate the location of an emergent target, which is a target of interest with an approximate location. Teams are required to dynamically re-route to this location. The teams then attempt the SRIC task, which involves downloading information from a remote WIFI network and relaying this information back to the ground system. Finally, the air system lands and data is given to the judges. The air system is tasked with identifying and classifying targets of interests that are located within the mission area during the course of aforementioned flight operations. Teams are judged based on targeting performance and level of system autonomy. Once the mission is complete, teams have 15 minutes to cleanup the area and leave the mission site.

Figure 4: 2013 AUVSI SUAS Competition: CUAir preparing for transport to the Mission Site

Ground Focused Imaging. A core focus of the competition is the ability for the air system to take pictures of the ground and identify targets of interest. The air systems must achieve good imagery ground coverage as the target identities and locations are not known to the teams beforehand. Most teams bring fixed-wing aircraft to the competition due to their better performance in long range flight, stabilized flight in wind, and flight with heavy payload components. These aircraft change attitude by actuating flight surfaces to roll or pitch the aircraft. When the aircraft changes attitude, the payload also changes attitude. This means that cameras that point at the ground to take images are no longer pointed at the ground when the aircraft changes attitude from a level position. The aircraft changes attitude during turns, and in response to wind that moves the aircraft off desired flight-path. This means that during these events the air system does not take pictures of the ground, which could cause the system to miss targets entirely and hurt mission performance. A stabilized gimbal system can actuate the camera in response to these attitude changes to keep the camera focused on the ground where targets are located.

Off-Axis Target Imaging. During the waypoint navigation phase of the simulated mission the air system is required to image a target with known location that does not appear directly below the air system's flight path. That is, a camera system which only points directly down cannot image this off-axis target. There are two ways to solve this problem: use multiple cameras that are fixed so as to cover any off-axis target, roll the

Figure 5: 2013 AUVSI SUAS Competition: CUAir settings up for the simulated mission

Figure 6: 2013 AUVSI SUAS Competition: Hyperion (CUAir's Aircraft) preparing for takeoff

Figure 7: 2013 AUVSI SUAS Competition: Hyperion Flying the Waypoint Navigation Section

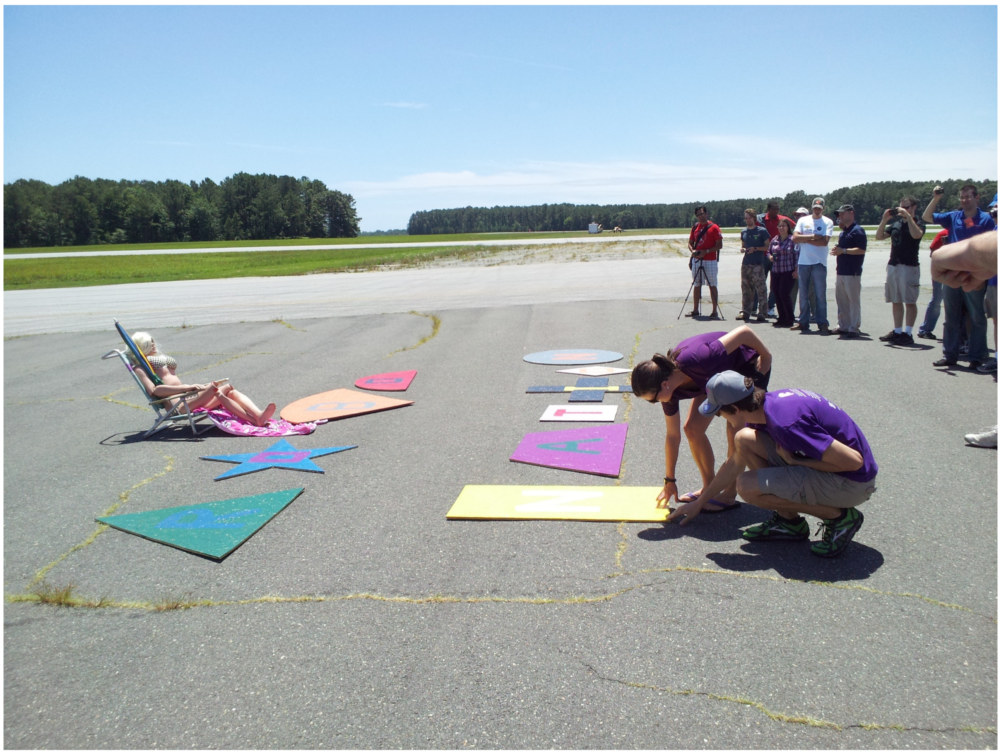


Figure 8: 2013 AUVSI SUAS Competition: Targets that were placed in the mission area

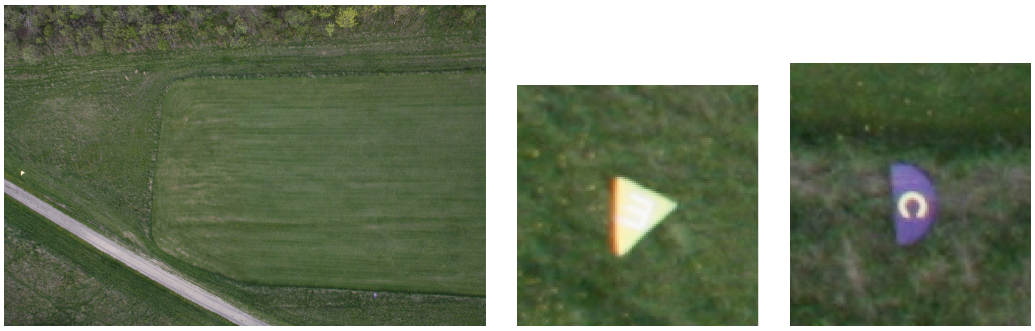


Figure 9: Image taken from Hyperion (left) during a test flight and the targets seen in the image (center, right).

aircraft to align the camera with the target, or use a stabilized gimbal system that can point at the known GPS location as the aircraft is passing the target. Multiple cameras are usually prohibitive due to size, weight, and cost. Rolling the aircraft will cause it to deviate from the required flight-path and may cause the aircraft to exceed the path tolerance. Thus, a stabilized gimbal system is best for a fixed-wing aircraft to image this required off-axis target.

New Competition Components. The 2014 AUVSI SUAS Competition have a few new components than previous years. The two major components are infrared imaging and egg drop. The infrared imaging task is to identify a target of interest that requires an infrared camera to be seen. Infrared cameras are significantly more expensive at higher resolutions, which will require teams to purchase lower cost cameras that have the ability to zoom. Cameras that zoom require a gimbal system to be effective as the probability of a target appearing perfectly below the flight path is very small. The egg drop task is to drop a plastic egg with flour inside to hit a known target. There are many ways to accomplish this task, but most require the actuation and positioning components that are inherent to a stabilized gimbal system.

1.2 CUAir: Cornell University Unmanned Air Systems Team

CUAir is Cornell University's Unmanned Air Systems Engineering Project Team. The team consists of about 40 undergraduate students, up to 1 graduate student, and a faculty advisor. The faculty advisor for the team is Professor Thomas Avedisian of Cornell University's MAE School. The team is structured into five subteams: airframe, autopilot, software, electrical, and business. There is a team lead that manages the entire team, and five subteam leads that manage their respective subteam. Phillip Tischler is a member of the CUAir team. From 2012-2013 he was the full team lead, from 2011-2012 he was the software subteam lead, and 2010-2011 was his first year on the team. Phillip Tischler is the single graduate student for the 2013-2014 year.

Team History and Goals. CUAir, Cornell University Unmanned Air Systems, is an interdisciplinary project team working to design, build, and test an autonomous unmanned aircraft system capable of autonomous takeoff and landing, waypoint navigation, and reconnaissance. Some of the team's research topics include airframe design and manufacture, propulsion systems, wireless communication, image processing, target recognition, and autopilot control systems. The team aims to provide students from all majors at Cornell with an opportunity to learn about unmanned air systems in a handson setting. The team was founded in 2002. In 2012 the team placed 2nd Overall, and 1st in Mission Performance. In 2013 the team placed 1st Overall, 1st in Mission Performance, and 1st in Technical Journal Paper. The team hopes to continue this success in the 2014 competition.

Overall System Design. Figure 10 shows the design of CUAir's full system. It is quite complicated and out of scope for this document. From this figure you can see that there is a message passing layer that can carry data and command messages from various nodes in the system. The aircraft node is the software process located on the aircraft's payload computer that controls all payload components. This software node can communicate with any onboard gimbal control systems and be a middle-man for any data or command messages that come from the ground system. Any gimbal control system simply needs to be integrated into the aircraft node, and then the gimbal is fully integrated into the rest of the system.

1.3 Stabilized Gimbal System

A Stabilized Gimbal System would greatly improve the mission performance of CUAir's air system at the AUVSI SUAS Competition. Such a system would increase the amount of imagery ground coverage by focusing the camera system on the ground during attitude corrections that roll and pitch the aircraft. Such a system would also allow the air system to image off-axis targets reliably. Commercial systems are not a viable option for the CUAir team. The commercial systems the team investigated indicated that such systems were either lacked performance, were prohibitively expensive, or did not allow the required system integration the team would need. As such, the team decided it is best to build a custom Stabilized Gimbal System.

CUAir previously attempted to build a Stabilized Gimbal System, but that effort was mostly unsuccessful. However, the lessons learned during that effort guided this new system to success. Previously, the mechanical system relied on gears that allowed the gimbal to move in between commanded set points. This greatly decreased the accuracy and reliability of the camera actuation. The new gimbal system developed uses mechanical arms that remove this "slop" in the gimbal mechanism. Second, the gimbal suffered from twitching servos due to a noisy power supply as well as signal corruption. Signal corruption was caused when the servos produced back-emf on the same line as the pulse signal controlling the servo's position. The new gimbal system uses linear voltage regulators to smooth the power supply, and optical isolation chips to prevent signal corruption. Third, the accelerometer and gyroscope chips used to calculate the gimbal's state suffered from axis misalignment and mis-calibration. The new gimbal system has a single chip with internal axis alignment, factory-calibrated axis alignment, and internal calibration. Finally, the communication interface was previously selected to be Ethernet, a choice which drastically complicated the project and caused the gimbal control boards to take up much more space. The new gimbal system uses regular serial communication with a USB connection.

This project will provide CUAir with a gimbal system that can actuate a camera accurately and reliably, communicate with a host computer to convey state information and receive command settings, to calculate its state from on-board sensors, to stabilize the camera system so as to keep it pointed at the ground, and finally to point the camera system at a known GPS location. Such a system will help CUAir excel at the 2014 AUVSI SUAS Competition.

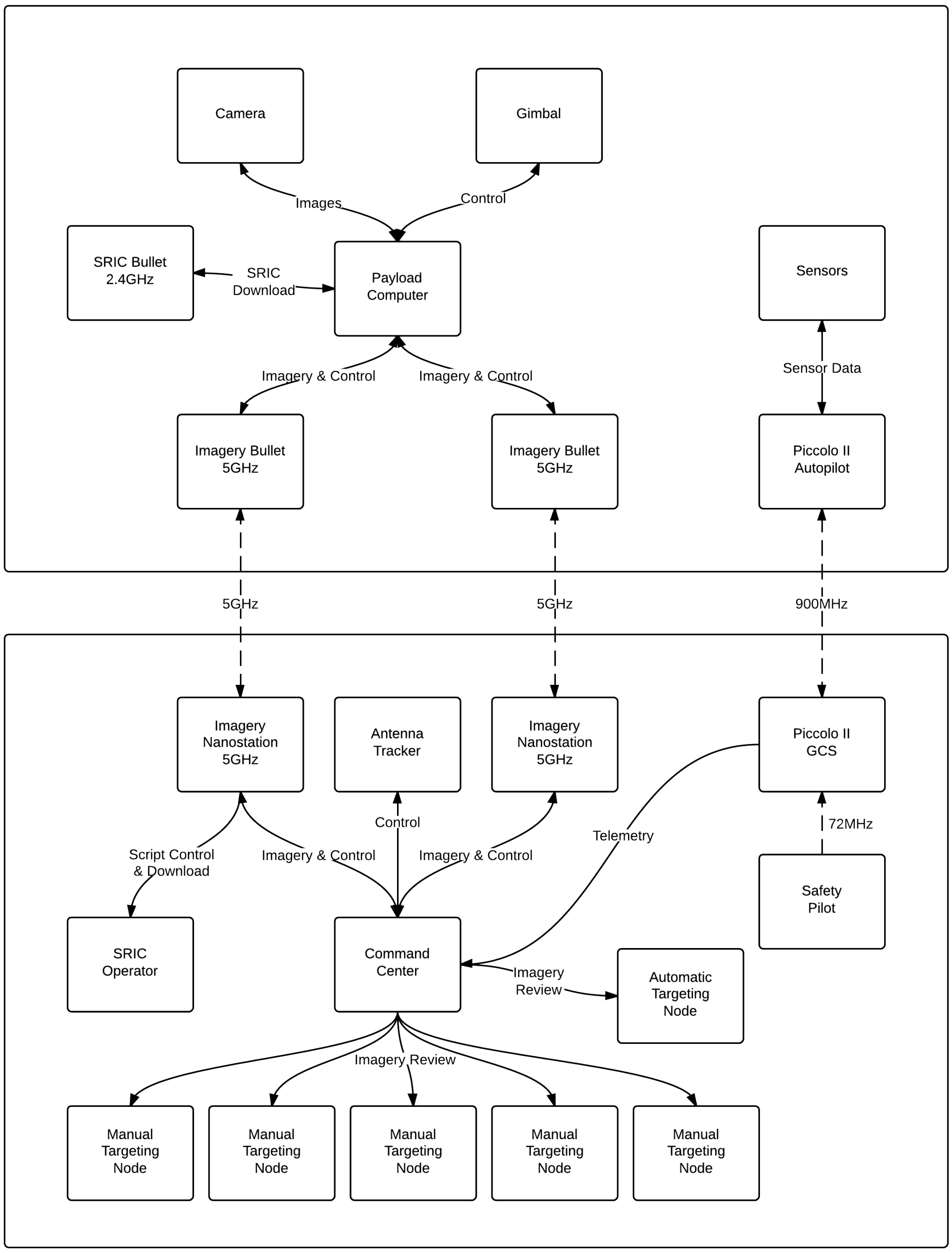


Figure 10: CUAir's Overall System Design. The top portion of the diagram shows system components that are located in the aircraft. The bottom portion of the diagram shows system components that are located in the ground station. As shown, the gimbal control system is controlled by the payload computer which executes the aircraft node software process.

2 System Design

This section describes the systems level design of the Stabilized Gimbal System. The first subsection describes the requirements of the system. The second subsection describes the overview of how the gimbal system works. The final subsection describes how the system is integrated into the rest of the CUAir system.

2.1 Project Requirements

The Stabilized Gimbal System has defined features that it should support to optimally perform its role in the AUVSI SUAS Competition's simulated mission. These requirements include those for communication with the gimbal and control of it, means to actuate servos, the ability to sense and estimate the system's current state, the ability to stabilize and point at a GPS position, and the ability to support future tasks like the egg drop.

Communication and Control. The Stabilized Gimbal System must be controllable by the aircraft's payload computer, which would allow it to be controlled via interfaces on the ground system or through autonomy software on the aircraft. The communication interface should support commanding the gimbal into different modes of operation, and allow the gimbal to relay state information at high speed. Additionally, the communication must be reliable and robust to errors. The AUVSI SUAS mission is a realistic one, and during the mission there is no time to debug problems, reconnect cables, or restart devices. The software must be able to survive and recover from any potential problems. This means that lost bytes, corrupted data, unplugged cables, or any other such problems cannot cause complete degradation of performance and should resume optimal operation immediately once the problem is restored. Finally, the communication system must be easily reconfigurable. End-users should be able to define new message types with relative ease, so that the system may be expanded upon.

Servo Actuation. The Stabilized Gimbal System must be able to control at least 2 servos, and preferably up to 5. Two servos are needed to control the two axis mechanism that the camera system is mounted to. One to two servos may be needed to retract the gimbal in future designs of the CUAir system where the aircraft lands differently. One servo will also be needed to perform the egg-drop task and any future tasks.

State Sensing and Estimation. The Stabilized Gimbal System must be able to receive accelerometer, gyroscope, and GPS data so that it can sense its state. From that data, the system should also be able to estimate the current state of the gimbal system. This is necessary in order to perform the stabilization and point at GPS tasks. State estimation can be performed two ways: it can be done off-chip on hardware internal to the sensors themselves, or it can be done on the micro-controller itself.

Stabilization and Point at GPS. The Stabilized Gimbal System should be able to take its estimated state and compute how to actuate the servos in order to point the camera system in a desired direction. For stabilization, this means computing the angular offset that is necessary to counter the roll and pitch of the aircraft. For point at GPS, this means to compute the difference in position, the subsequent roll and pitch necessary to point at the GPS position, and then the actuation of the servos to achieve that amount.

Egg-Drop and Future Tasks. The Stabilized Gimbal System will likely be expanded to perform future roles. One such task is the Egg-Drop. The hardware and software should be designed to promote the Egg-Drop tasks, as well as any reasonable future design revisions to the CUAir system.

2.2 Stabilized Gimbal System Overview

Figure 11 shows an overview of the Stabilized Gimbal System that was developed and will be integrated into the CUAir system. The diagram is broken into four sections: the Gimbal Control System, the Physical Gimbal, the Payload Computer, and the Personal Laptop. The Gimbal Control system consists of a printed circuit board (PCB) that contains a 6 Axis MEMS device, a GPS header which connects to a GPS device, a FTDI chip which converts serial to the USB connection, and the micro-controller. Each individual component is discussed in greater detail in later sections. The MEMS device and the GPS continuously relay state to the micro-controller. The micro-controller then estimates its state, computes the required actuation amount to achieve a desired functionality based on the current state, and then outputs the pulse signal to actuate the servos. The micro-controller then sends its state from the Gimbal Control System to the Payload Computer via the USB connection.

The Payload Computer receives the state information from the USB connection and can use that connection to send commands. Example commands include to point at a specific GPS position, to stabilize to a relative attitude, or to set specific servo angles. The communication interface, state messages, and command messages are packaged into a C++ software library that was developed as part of this project. This makes integration extremely easy. The existing aircraft node software process uses the Stabilized Gimbal Control System Library to receive the state information and send command settings. The aircraft node acts as a middle-man between the Stabilized Gimbal System and either the ground station or other autonomy software. To demonstrate the ease of integration, the team built a test client GUI application which shows the data received through the software library and allows the user to send command messages.

2.3 System Integration

System integration is a relatively straightforward task. The new Stabilized Gimbal System contains a library that makes interfacing with the gimbal simple. The user instantiates and instance of *GimbalCommManager*, a class which controls the serial communication to the gimbal control board. The user implements a callback function and provides it to the communication manager. The communication manager will then call the callback function whenever messages are received from gimbal control board. The user can also use the communication manager to send command messages to the gimbal. No other work is necessary to interface with the new Stabilized Gimbal System.

During the mission the CUAir operators will need to command the Stabilized Gimbal System to enter different modes of operation. The CUAir ground station already allows the operators to select different gimbal modes of operations and their parameters. This data is then sent to the aircraft via the wireless link. Once received the aircraft node then provides the messages to the software responsible for controlling the gimbal. Currently, this is the gimbal control software from the previous gimbal which had an Ethernet interface. Integrating the new Stabilized Gimbal System software into the CUAir system would simply mean replacing the old Ethernet interface code with the new software library. This work is schedule to occur during winter break before return for Spring 2014 semester. Once integrated, the CUAir team will test-fly the entire system to validate its operation. Figure 12 shows last year's plane, Hyperion, which will be used to evaluate the Stabilized Gimbal System until this year's plane is ready for competition.

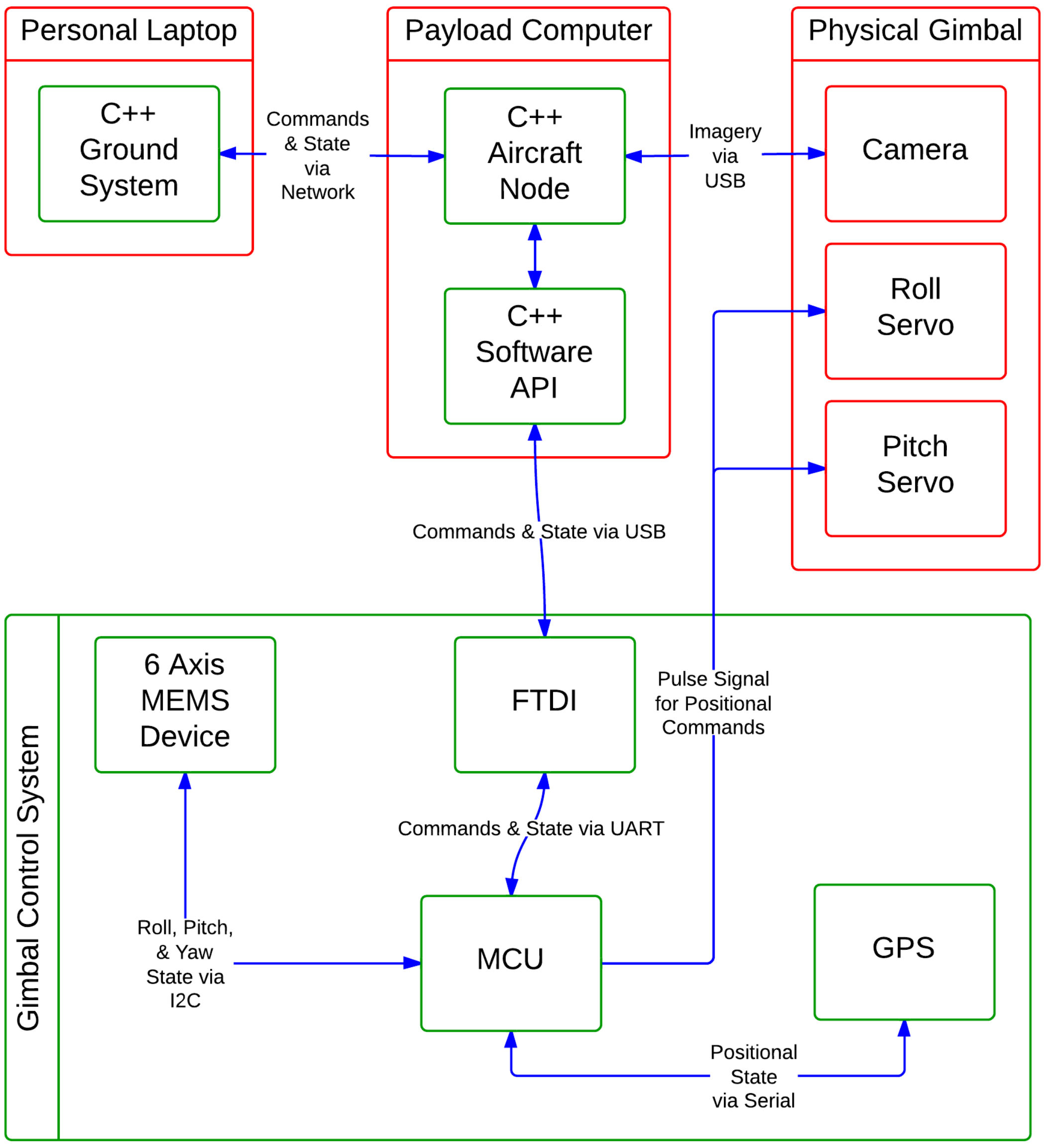


Figure 11: Stabilized Gimbal System Block Diagram Overview

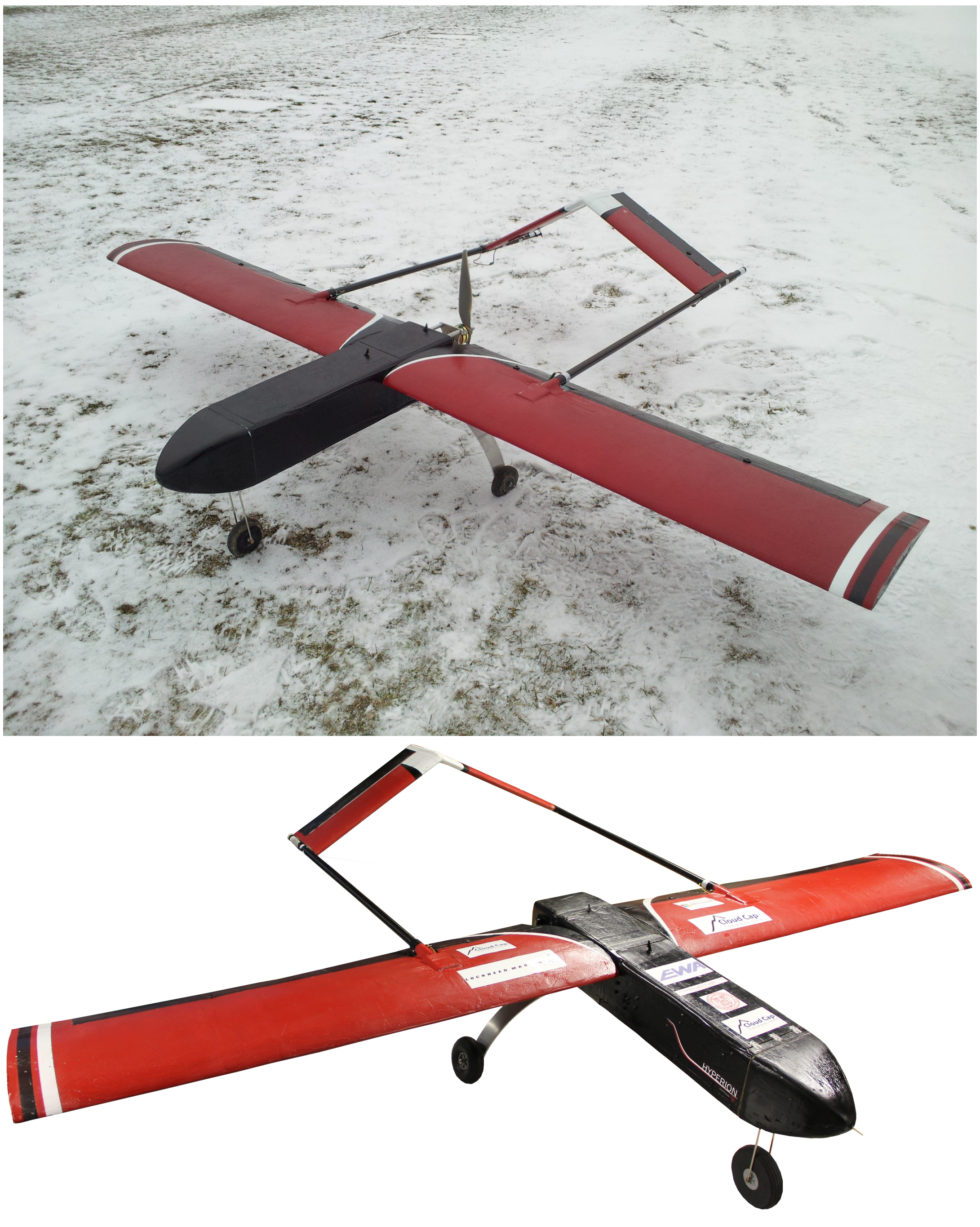


Figure 12: CUAir's 2013 Competition Plane: Hyperion

3 Hardware Design

This section describes the design of the hardware for the Stabilized Gimbal System. The first section describes the design of the physical gimbal that the control system was designed to actuate. The second section describes the design of the Stabilized Gimbal Control Board, which contains all of the hardware work performed by the members of the project.

3.1 Physical Gimbal

The ECE 4760 final project was worked on by Phillip Tischler and Kelly Glynn. The final project consisted of the gimbal control board and the gimbal control software. It did not consist of the physical gimbal that the control system is designed to actuate. The physical gimbal was built by Derek Faust, a member of the CUAir team. This section describes the design of the physical gimbal as it motivates the design of the rest of the system.

Actuation Properties. There are many types of gimbal systems that could be used for an autonomous reconnaissance aircraft, but only two are usually used. The first is a pan-tilt gimbal which is best used when a human operator has to view a video feed and manually control a gimbal to visualize a target. However, this is not the use case that CUAir is concerned with. CUAir's system needs optimal stabilization but does not care about orientation because algorithms process the imagery. Thus, CUAir uses a roll-pitch gimbal system.

Mechanical Frame & Servos. The mechanical frame and servos form a two-axis roll-pitch gimbal system. One servo controls the actuation in the roll axis, and one servo controls the actuation in the pitch axis. The gimbal supports up to 60 degrees of actuation in the roll axis which allows the gimbal to view the farthest off-axis target. The gimbal supports up to 30 degrees of actuation in the pitch axis which allows the gimbal to stabilize for any normal ascending or descending the aircraft will perform during a mission. Figure 13 show the gimbal's mechanical frame and servos.

3.2 Electrical Control Board

This section describes the design of the Electrical Control Board for the Stabilized Gimbal Project. The following describes the various components that were selected for the control board, how the control board was designed, and how the control board was fabricated.

3.2.1 Optical Isolation & Servo Connection

Electric motors and servos are usually controlled by an input line, a line which is frequently controlled by a micro-controller. For the Stabilized Gimbal System, the Atmega128 micro-controller is used to control up to 5 servos. When a load is applied to a motor or servo it produces an back electromagnetic force (emf) which can corrupt a control signal. The solution is to electrically isolate the control signal from the line that contains the back emf. This is accomplished using optical isolation. An optical isolation chip works like a transistor. Electricity flowing through one side of the isolator will create a light beam that activates a photodiode which turns on a transistor. Essentially, current through one side of the isolator allows current through the other side to flow, without the

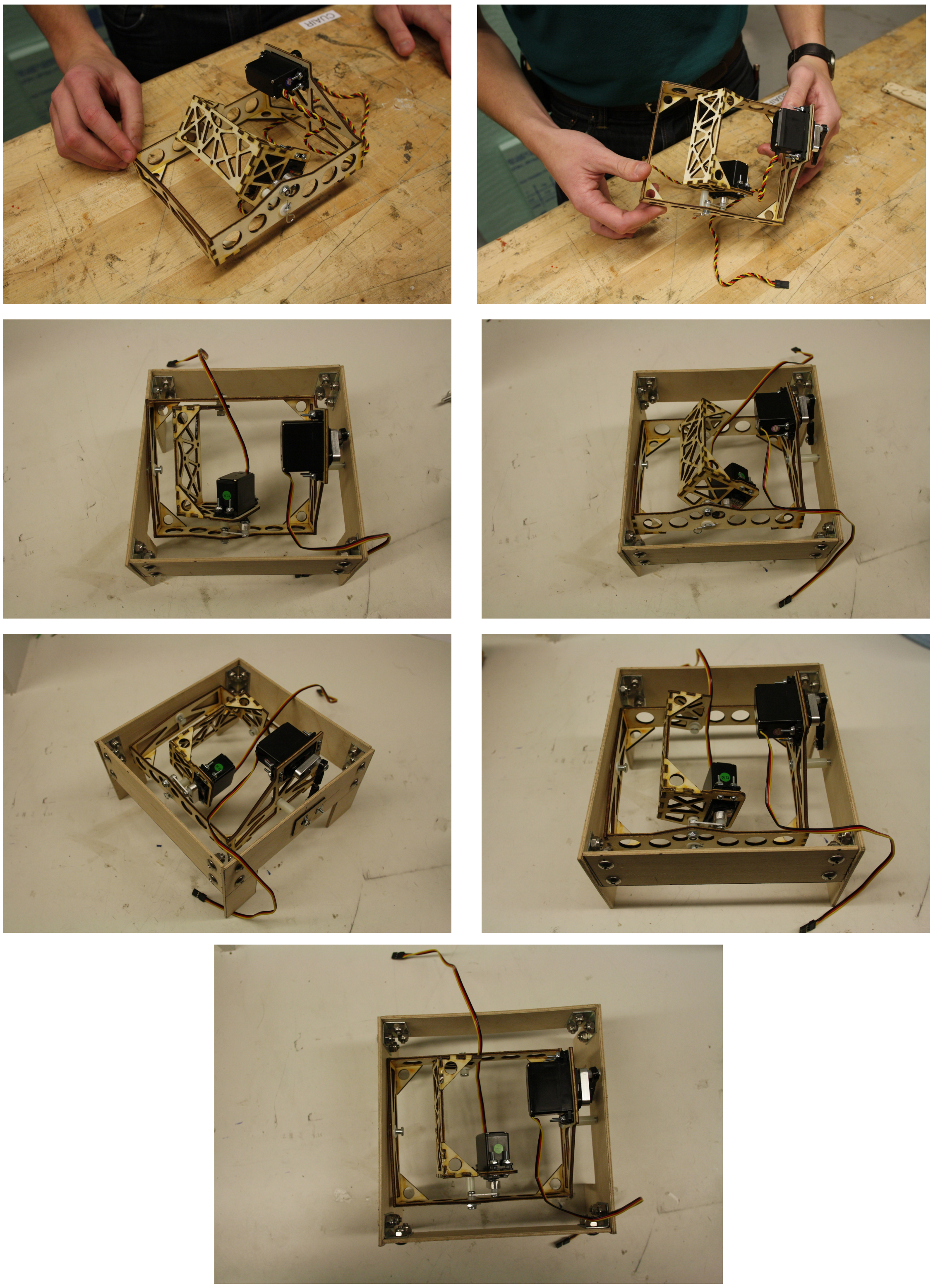


Figure 13: Physical Gimbal for the Stabilized Gimbal System. The top 2 images show the gimbal without being mounted. There is an outer tray which is actuated by a single servo. The inner tray is mounted to the outer tray and actuated by a separate servo. The other images show the gimbal in a test-mount, which has the same mounting interface as the aircraft it will be integrated into. The camera is mounted to the inner tray and is thus actuated by the two servos in the roll and pitch axis of the gimbal.

need for the two sides to be electrically coupled in any way. Figure 14 shows the optical isolation chip used by the Gimbal Control Board.

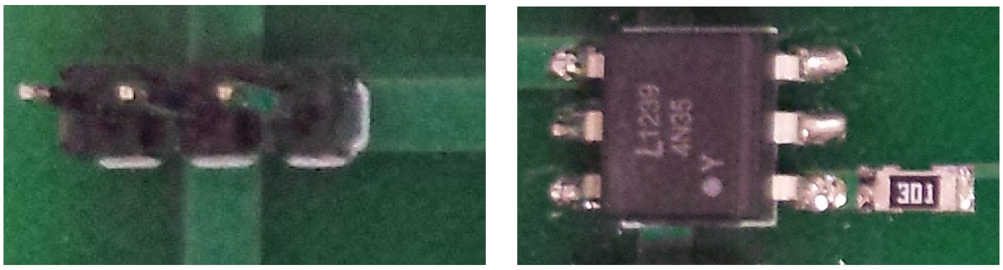


Figure 14: The left image shows the servo pin header used to connect the gimbal servos. The right image shows the optical isolation chip used to electrically isolate the servo control lines from the micro-controller control lines. Each servo output line has one optical isolator and one servo pin header.

3.2.2 GPS Unit

The GPS unit provides the Stabilized Gimbal Controller the ability to sense its absolute position on earth. This is necessary for the point at GPS functionality. That is, the control system needs to know its position to know the relative pointing angle to command so that the camera points at the desired GPS position. The GPS also provides an absolute clock which can be used to synchronize time between components.

Figure 15 shows the GPS unit that was selected for the project and how it connects to the board. The GPS unit chosen was selected because of its wide use in open source autopilots, other CUAir members previously researched GPS units and figured out it is a high quality device, and subsequently the CUAir team already owns the unit. The GPS unit gives position (latitude, longitude, absolute altitude (msl), relative altitude (agl)) at 1Hz and can be configured to run at up to 10Hz. It also gives velocity which can be used to extrapolate where the plane is going and point the gimbal between GPS updates. [13]

3.2.3 6-Axis Inertial Measurement Unit

The MPU-6000 is a 6 Axis IMU which consists of a 3 axis accelerometer and 3 axis gyroscope. The IMU also has a Digital Motion Processor (DMP) which can provide a state estimate as a quaternion at rates up to 200Hz [15]. A quaternion is a 4 dimensional vector where one dimension represents the cosine of half the angle, and the other three dimensions represent the vector axis of rotation times the sine of half the angle. This is discussed in more detail in later sections. The unit was selected because it provides on chip axis alignment of the accelerometer and gyroscope, superior factory and internal calibration relative to separate chips, availability and popularity, and because of the DMP. The DMP provides a state estimate which could potentially save the Micro-controller from having to compute the state itself. Figure 16 shows the IMU chip on the Gimbal Control Board.

The downside to the MPU-6000 is that the chip is extremely small which makes handling difficult, the chip is lead-less which makes board assembly extremely difficult and error prone, the chip interfaces through I2C which makes receiving data slightly

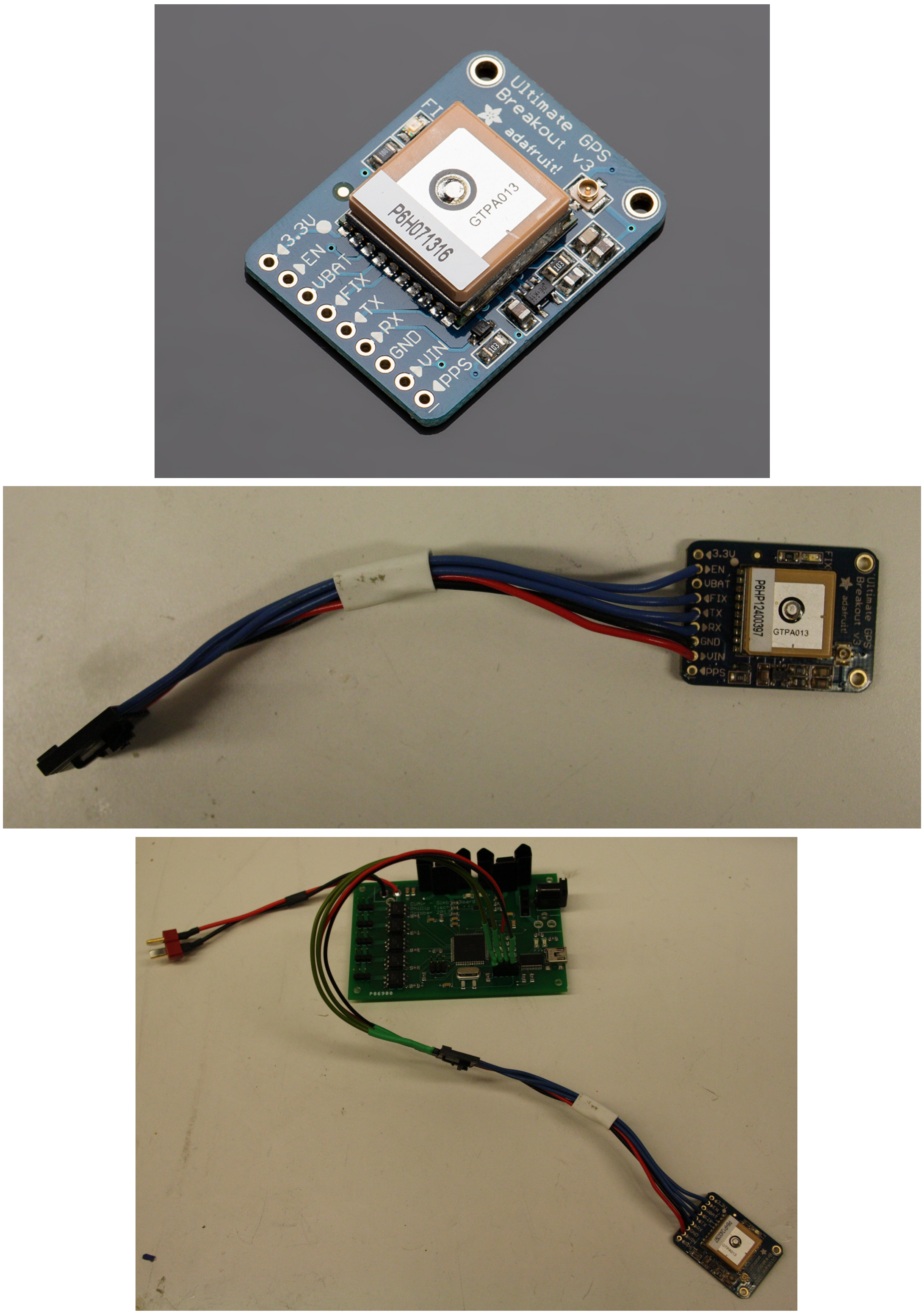
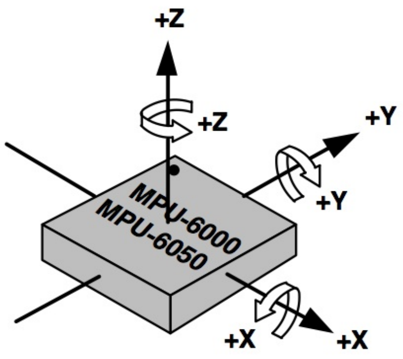


Figure 15: Stabilized Gimbal System's GPS Module. The top image shows the GPS unit as sold by Adafruit [13]. The middle image shows the GPS unit with a connector soldered onto it. The bottom image shows the GPS unit connected to the Gimbal Control Board that was developed by the team.



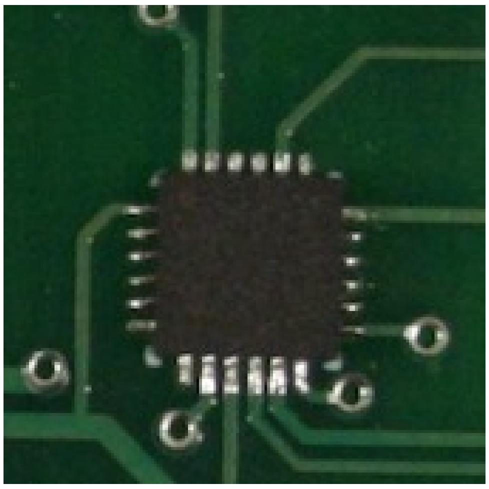


Figure 16: The 6 Axis Inertial Measurement Unit. The left image shows the axis alignment [15] and the right image shows the IMU chip on the control board.

more difficult, the chip runs on a 3.3V power supply and logic signal which requires extra components, and the software to interface with the chip is very closed-source. All of the potential downsides to the chip were resolved during the course of the project, except for the closed-source nature of the DMP. The DMP is completely closed-source and it is still quite difficult to interface with or debug.

3.2.4 Logic Level Converter

The logic level converter is used to translate a 5V I2C signal to a 3.3V I2C signal. The Atmega128 micro-controller operates at 5V and outputs a 5V I2C signal. The MPU-6000 IMU chip operates at 3.3V and outputs a 3.3V I2C signal. This difference prevents communication between the two devices. The logic level converter connects to the 5V and 3.3V signal lines to convert the voltage levels. This creates a communication channel between any 5V and 3.3V devices connected to the two I2C lines. Figure 17 shows the logic level converter on the Gimbal Control Board. It is by far the smallest component that is on the board, and is the second hardest to solder.

When building another board, it should be noted that there are multiple manufacturers that make this chip, each with the same name but with different pin-outs. This lead to an early mistake where a TI chip was purchased instead of the NXP chip. This caused the component to start smoking on the board. Once replaced with the proper chip everything worked perfectly.

3.2.5 FTDI & USB Connection

The Gimbal Control Board needs to communicate with the payload computer on the aircraft in order to relay state information and accept commands. There are two possible interfaces: Ethernet and USB. Ethernet simplifies development from the payload computer software side, but it drastically increases the complexity of the Gimbal Control Board hardware and software. USB is a much easier interface to implement and can achieve the required communication speeds. The FTDI chip converts the serial that is transmitted and received from the micro-controller into the USB interface needed to

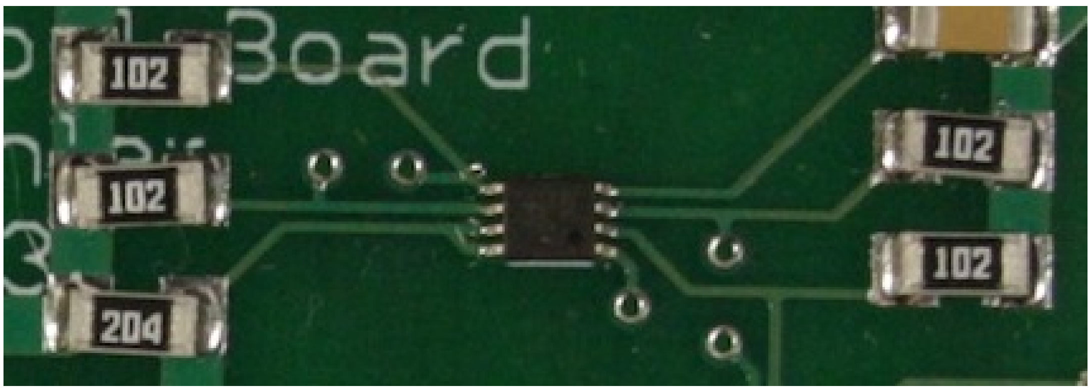


Figure 17: The logic level converter used to interface the 6 Axis IMU (3.3V logic signal) with the Micro-controller (5V logic signal).

communicate with a host computer. This chip requires the host computer to install the FTDI driver which makes the USB connection appear as a serial communication interface. Figure 18 shows the FTDI chip and the USB connection.

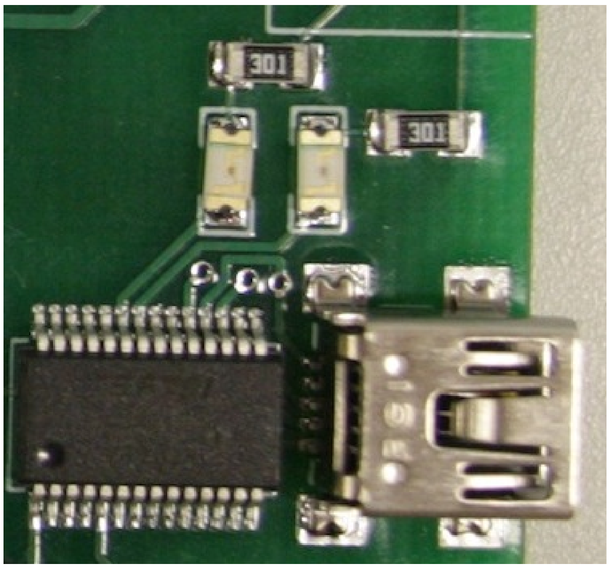


Figure 18: The FTDI chip and USB connection used to interface the Gimbal Control Board with the Payload Computer.

3.2.6 Atmega128 Micro-controller

The Atmega128 micro-controller was selected because of the number of timers, PWM output lines, and serial communication interfaces it has. The Gimbal Control Board needs to control 5 servos, receive data from a GPS, and communicate with the payload computer. It was also desired to have a smaller surface mount chip that would make the entire footprint of the Gimbal Control Board smaller. This is important for a device that is mounted into a small autonomous aircraft. Figure 19 shows the Atmega128 on the Gimbal Control Board.

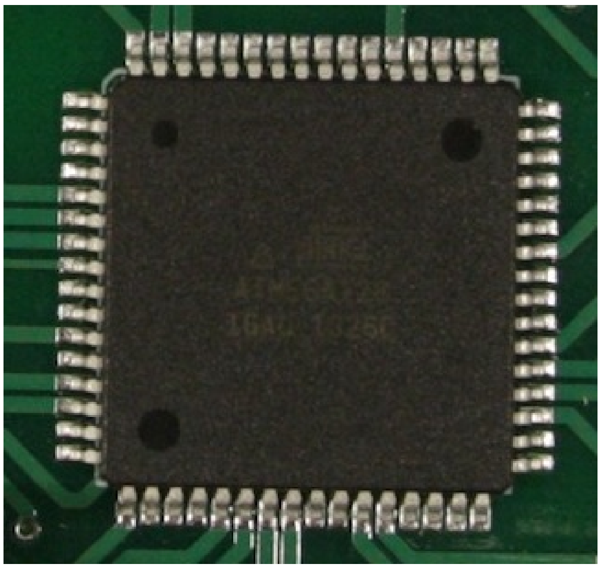


Figure 19: The Atmega 128 Micro-controller chip used to control the gimbal control board.

The choice of the Atmega128 did have one downside, the amount of available data memory. The MPU-6000 needs to be loaded with its firmware upon startup, which needs to be stored on the Atmega128 and sent via I2C. The code to control the chip and the firmware for the MPU-6000 is bigger than the data memory available on the chip. This required the firmware to be stored in program memory and for the MPU-6000's software library to be modified to load from program memory. This is discussed in more detail in later sections.

3.2.7 Voltage Regulation

Voltage regulation drops a higher voltage and noisy input signal into a specific voltage with low noise. There are two types of voltage regulation: switching and linear. Switching regulation is much more efficient than linear, but it introduces a ripple into the output line which increases the noise. Certain components can deal with this noise which makes switching regulation ideal. However, previous designs for the Stabilized Gimbal System indicate that the gimbal is not one of those components. Linear regulation is less efficient and generates more heat, but it provides a lower noise output signal.

The Gimbal Control Board uses two linear regulators: one to provide a 5V power line and another to provide a 3.3V power line. The 5V line powers almost all components except the IMU and the Logic Level Converter which are powered by the 3.3V line. The regulators were chosen to easily cover the power requirement so as to minimize heat generated. The aircraft payload compartment is mostly sealed and does not necessarily get great airflow. Good heat dissipation is important for such an environment, which is why powerful regulators were used with heat sinks. Figure 20 shows the voltage regulators on the Gimbal Control Board.

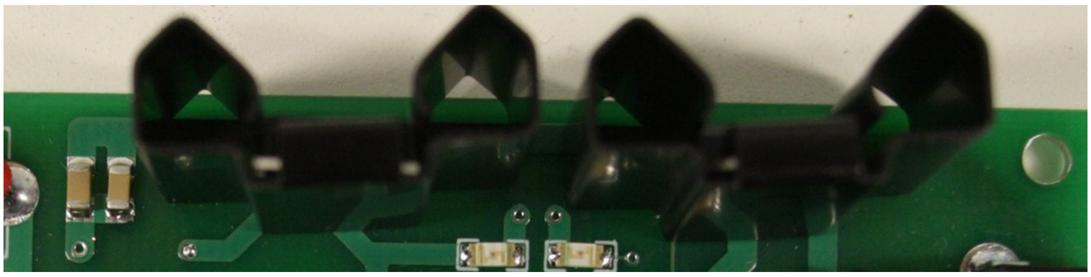


Figure 20: The 5V and 3.3V linear voltage regulators with heat-sinks used to supply power to the Gimbal Control Board components. These regulators do not power the servos, which have a separate power line and connection. The black housings are heat sinks to dissipate any heat that may be generated.

3.2.8 Printed Circuit Board Design & Re-flow Soldering Assembly

The printed circuit board (PCB) was designed using Eagle CAD and then sent to Advanced Circuits to be manufactured. This was done early enough in the semester so that the boards were manufactured in time to start the software. Phillip Tischler, who designed the boards, did not know how to design or manufacture PCBs prior to this project and learned in the beginning of the semester. Figure 37 shows the PCB schematic that was designed. Figure 47 shows the PCB layout that was sent to be manufactured. Figure 21 shows what the boards looked like when they arrived, before they were soldered.

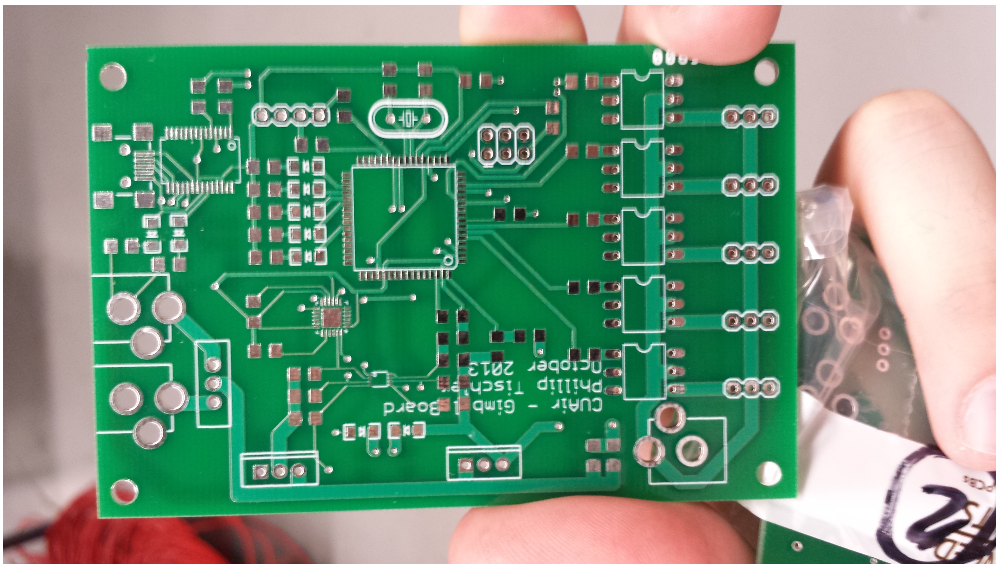


Figure 21: The Gimbal Control Board before it was reflow soldered.

Most of the components on the Gimbal Control Board are surface mount. The logic level converter is so small it is practically impossible to solder by hand. The MPU-6000 IMU is lead-less and thus impossible to solder by hand. The team used re-flow soldering to assemble most of the board. Phillip Tischler did not know re-flow soldering prior to the start of the project. The team first purchased re-flow tutorials on Sparkfun which served as an introduction to the technique. The team then used a stencil cutter to cut stencils to reflow solder the Gimbal Control Board. The team applied solder paste and used a griddle to reflow the board. The process proved to be very easy, once practiced, and the team was able to assemble two Gimbal Control Boards quickly. Figure 22 shows the assembled boards.

The board did have design flaws that were fixable. The first mistake was that the programming header was not given power. A jumper wire connected the programming header to power and fixed this problem. The second mistake was that connection of the micro-controller programming lines to the header. In system programming (ISP) usually uses SPI for communication. The Atmega128 also uses SPI for ISP, but it has separate SPI lines from the main lines for programming. These lines are not labeled as SPI lines, but rather PDI/PDO lines and is referenced in one page of the datasheet. Once this mistake was caught, two jumper wires were soldered onto the programming lines and the problem was fixed. A total of three mistakes were made, with three jumper wire fixes. Future constructions of this board should fix these mistakes on the PCB design before sending the board to be manufactured.

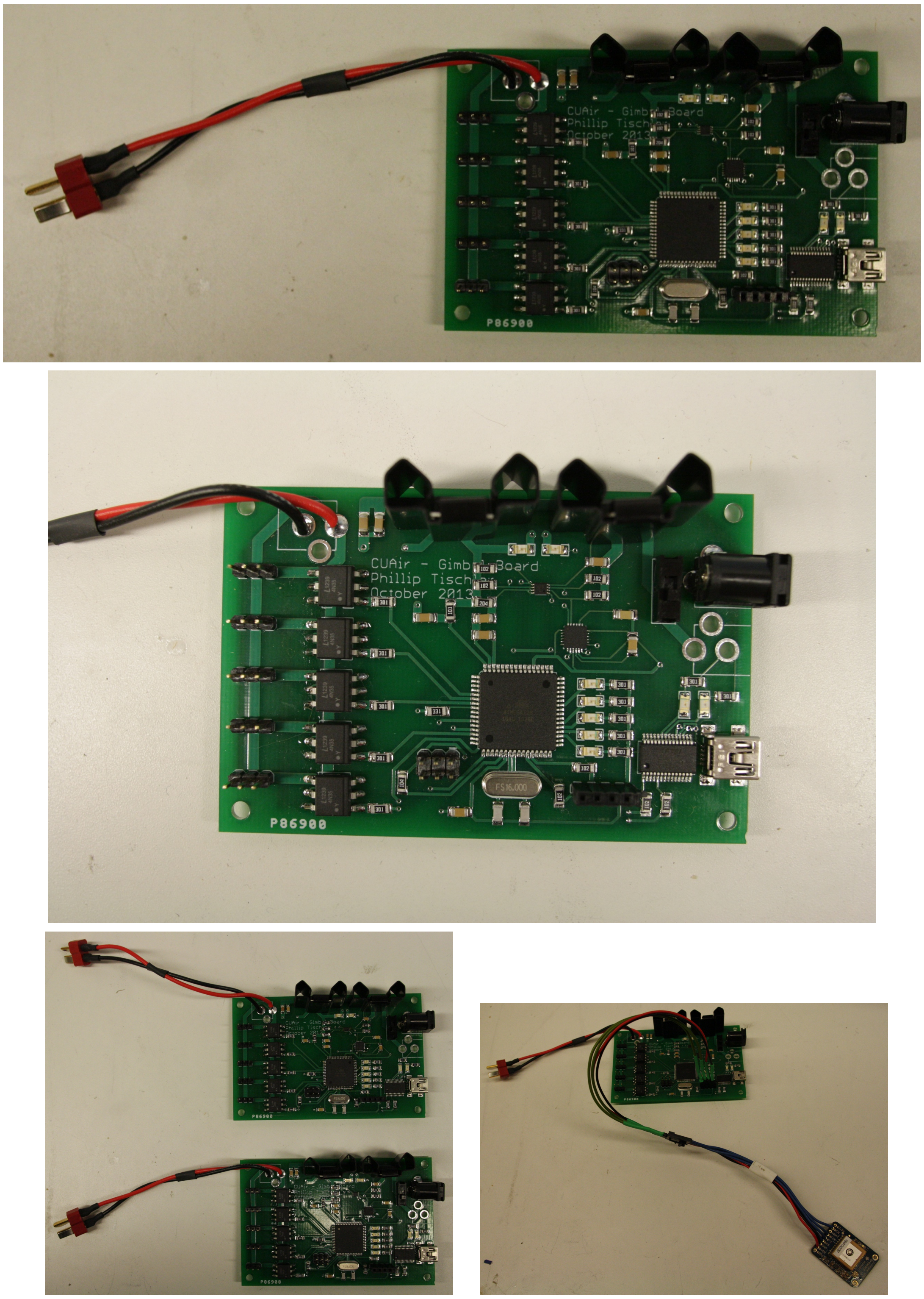


Figure 22: The Gimbal Control Board PCB Manufactured Boards and Reflow Assembly Images. The top image shows the board after the initial reflow and hand soldering process. The second image is a close-up of the board after it was completed. The third image shows how two boards were fully assembled. The final image shows the board with the GPS module connected.

4 Software Design

This section describes the design of the software for the Stabilized Gimbal System. The first subsection describes the software that executes on the micro-controller. The second subsection describes the software that executes on the host computer. In production this is the aircraft node on the payload computer. For the purposes of this project, the host computer executes the test client application.

Figure 23 shows the overall software stack for the Stabilized Gimbal System. The micro-controller executable has a set of software modules that handle specific portions of the micro-controller's logic. The main application function connects these modules together to get the full gimbal control system. The message communication module is responsible for sending and receiving logical messages from the host computer. The GPS control module is responsible for receiving GPS NMEA strings, parsing them, and storing the parsed GPS data. The message communication module and the GPS control module both use the serial communication module to communicate with their underlying interfaces. The UART filestream module uses the serial communication module to access the host computer serial interface only when in debug mode. The UART filestream module provides *printf* functionality whereas the message communication module sends logical messages as raw binary using a custom communication protocol. The IMU module uses Peter Fluery's I2C code [11] and the InvenSense Motion Processing Library [14]. The servo control module uses the waveform generation of the modules to convert desired servo angles into pulse signals. The gimbal control module is used to implement the higher level logic which controls servos based on a desired behavior. The servo LED module turns on debug LEDs to help debug the system, and to indicate the current state in production. Finally, the ring buffer module is the module that ties all the components together in a highly efficient and concurrent way.

The host computer and the micro-controller communicate with one another through the MCU interface library. This library defines the messages that can be passed between the two systems, and how the messages are serialized and de-serialized. On the host computer side, the Gimbal serial communication library is the main library for controlling the Stabilized Gimbal System. This library gives the user a Gimbal Communication Manager which can send and receive logical messages to and from the micro-controller. This can be used to send control messages and receive state messages. The Serial Communication Manager abstracts the serial interface so it is not dependent on any specific message types. The Gimbal serial communication library is then integrated into a test application, which can be used to visualize state and send command messages. This application was demonstrated as part of the ECE 4760 final project demo. The CUAir aircraft node is the final integration step, a step which will be completed over winter break.

4.1 Micro-Controller Software

This sections describes the software that executes on the micro-controller. It does not explain the code or the interfaces, the comments and API documentation explain that portion of the software. This section describes the underlying rationale for the design and the code. The first subsection describes Q number format, a method to make arithmetic operations faster on the micro-controller. The second subsection describes the code for asynchronous serial communication. The third subsection describes how GPS packets are processed. The fourth subsection describes how servos are controlled using waveform

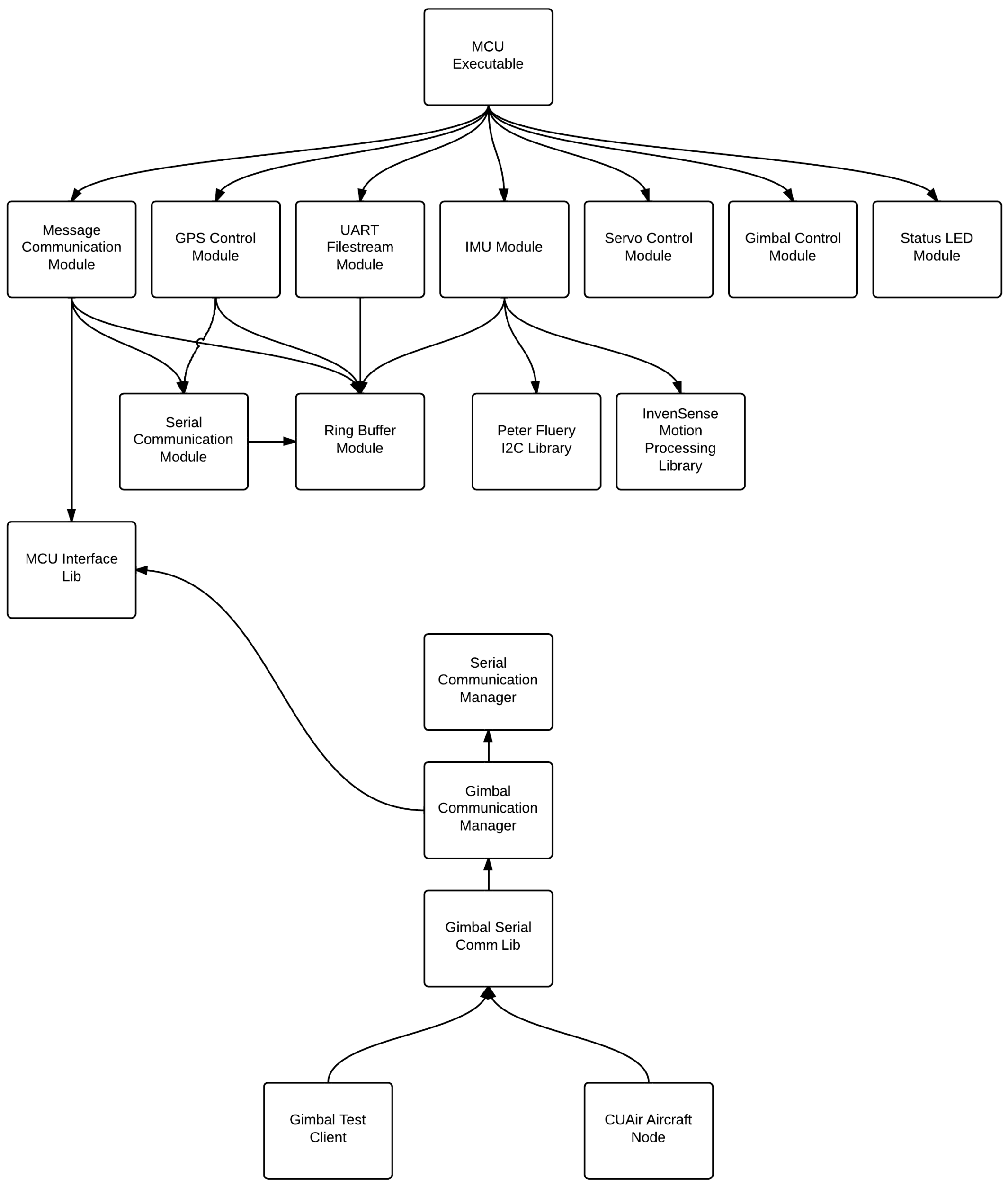


Figure 23: The software stack for the Stabilized Gimbal System. The top shows the components leading to the micro-controller executable that controls the physical gimbal. The bottom shows the C++ components that create the interface between the gimbal and the payload computer, and its relation to the test client and CUAir aircraft node.

generation. The fifth subsection explains the IMU software library and the work that was necessary to integrate the device. The sixth subsection explains Quaternion representations for rotations and the math that is used with it. The seventh subsection explains the algorithm for estimating state from the IMU output. The final two subsections describe the algorithms for stabilization and point at GPS.

4.1.1 Q Number Format Representation & Math

Micro-controllers do not have floating point hardware. Floating point math is simulated by the micro-controller using integer math. The result is that floating point operations take 100 to 300 cycles to complete. Many floating point operations can slow down the micro-controller drastically. The Stabilized Gimbal System needs to perform many calculations, must store various forms of data, and must be able to serialize that data. Q Notation math can provide 2 to 4 times speedup compared to floating point math [18, 26]. It also makes serialization the task of serializing an integer, not a floating point number.

Consider the math required to convert a quaternion to euler angles [25]. There are 15 multiplications, 4 additions, 3 subtractions, and 3 trigonometry calculations. Lets assume that each arithmetic operation takes 200 cycles and each trigonometry calculation uses about 1000 cycles. This means each calculation requires $22 * 200 + 3 * 1000 = 7400$ cycles needed per calculation. This calculation is done at 200Hz, which means it takes up about 1.48MHz of the processor. In reality the calculation will not be perfectly efficient and more cycles will be used that this lower bound. Approximately 1/8 of the micro-controller's CPU will get used to simply convert the representation of rotation. Q number format can cause the same calculation to only take 1/32 of the CPU.

Q Number format represents a decimal number as an integer number and a fractional numerator. The Q number is usually expressed as $Qm.n$ where m is the number of bits used for the integer number and n is the number of bits used for the fractional numerator. The integer number occupies the highest m bits of the number and the fractional numerator occupies the lower n bits of the number. Lets say the value of the m bits is a and the value of the n bits is b, then the value of the number is $a + \frac{b}{2^n}$ and the number is represented in memory as $(a << n)|b$. Converting a floating point number to Q number format means multiplying by the number by 2^n and rounding the value to the nearest integer. Converting from Q number format to floating point means converting the integer value to a floating point value and multiplying by 2^{-n}. The value of n is chosen prior to execution, so the values of 2^n and 2^{-n} can be computed and stored prior to execution. [26].

The numbers are represented as fractions, so additions and subtractions are equivalent in the Q number format. For multiplications and divisions, the answer must be multiplied or divided by the denominator respectively. This can be accomplished by left shifting or right shifting respectively. [26].

It should be noted that Q number format requires the user to select values of m and n prior to writing the code. Typically the values of m and n are chosen based on the domain of the problem being computed. If too small of values are chosen for m or n then the computation can suffer from overflow or lack of precision. If too large of values are chosen then the Q number format calculations will run slower than desired.

4.1.2 Asynchronous Serial Communication

The asynchronous serial communication module uses a concurrent (single reader, single writer) ring buffer to achieve concurrency. Receive interrupts signal that a byte has been received, and they insert data into the ring buffer. Send interrupts signal that a byte can be transferred, and they pull data from the ring buffer. The user can send and receive data by storing and pulling data from these ring buffers. The result is asynchrony between the user who sends and receives data and the underlying interface which does the sending and receiving. This does mean, however, that if data is not pulled from the buffer fast enough then data will be dropped. It also means that writes to the buffer may fail if there is not enough space in the buffer, which occurs when data is being written to the buffer faster than the underlying interface can send the data. This implementation also requires the extra memory to buffer data in the ring buffers.

The benefit of the asynchronous serial interface is the extreme improvement in CPU utilization. The ECE 4760 course code blocks until each byte can be sent. Lets say the UART is configured to the standard 9600 baud rate. This means 9600 bits can be transmitted per second. Lets say the user wants to send 100 bytes. For the asynchronous serial communication system, this would mean 100 store operations and 100 add operations to shift the data into the buffer for a total of 200 cycles, and about $70 * 100 = 7000$ cycles for the interrupts to send data. This makes the asynchronous system take 7200 cycles total to send 100 bytes of data. The course code, which implements the block until write operation, takes more cycles. At 9600 bits per second it takes $16\text{MHz} * 100\text{bytes}/(9600\text{bitsPerSecond}/8\text{bitsPerByte}) = 1333334$ cycles to send the data. This time spent waiting for the data to be sent could be used for other processing. In this example, the asynchronous communication implementation is 185 times faster than the block until written implementation.

The next piece of the asynchronous serial communication system is the method for which messages are sent, which is implemented in the message communication module. Messages are logical groupings of data that have meaning as a group. The asynchronous serial communication system uses a fixed size raw message header with a variable size raw message payload. The ECE 4760 course code uses printable characters to encode and transmit data between endpoints like the micro-controller and host computer. Lets say the system wants to transmit a 16 bit signed number. In the worst case it could be the number -32768. This would require 6 characters in order to transmit the information as plain text, whereas sending it as a raw 16 bit number would only require 2 bytes. This is a factor of 3 performance gain in amount of data that can be transmitted. There is additional performance gains when multiple pieces of data are sent in a single message because printable characters require a either a separating character or for the data to appear with fixed length and require processing. As the amount of data is sent in a single message increases, the greater improvement the fixed size header has over character separators.

The final piece of the communication system is how data is sent in a raw form. Data must be converted from the architecture dependent form that is represented on the micro-controller to a interface neutral form. This is designed to fix difference in architecture endianness, and differences in word size that causes byte padding. The conversion method is usually called byte packing, serialization, or marshaling. Data is converted from the micro-controller endianess to a specific endianness before being stored. The data is also stored as characters to prevent word alignment that is done by the compiler. This creates two data structures, the raw version which has the standard representation (e.g. integers) and the packed version which has the neutral format (e.g. array of characters). The workflow is as follows: endpoint A packs a message structure using standard assignment of variables, endpoint A packs the data into the packed version of the message structure by accounting for endianness and data padding, endpoint A sends the raw message buffer through the underlying interface (serial in this case), endpoint B receives the raw message buffer and uses the message header to determine the message type, endpoint B unpacks the raw message buffer by converting from the neutral endianness to the internal endianness, endpoint B uses the message data using standard variable assignment. This method of sending messages provides a higher performance means of sending data.

The final part of the communication protocol are the details regarding the message header and error handling. The message header contains the size of the data being sent and the message type, components which are used to unpack the message and process it afterwards. The message header also contains a sequence of known bytes which can be used to test if the current byte sequence is the start of a message. The micro-controller code scans the byte stream until it detects the start sequence. It then loads the message header and message payload. It then checks for the start sequence for the next message and so on. This helps detect corrupted streams and helps recover from errors. It should be noted that message error detection, using techniques like cyclic redundancy checks (CRC), are not performed to save on CPU overhead and complexity. The system assumes that message corruption is extremely unlikely to occur once the system is fully setup. If guarantees are desired, CRC can be added or the messages can be sent multiple times.

The end result is that data can be sent in raw form extremely efficiently in terms of bandwidth utilization and CPU utilization. This method also makes updating message types and introducing new message types an easy task. The message types and definitions are shared between the micro-controller and serial communication library. The serial communication library implements the same protocol in C++ using the Boost Library to manage serial communication ports.

4.1.3 GPS Packet Processing

GPS data is sent from the GPS module to the micro-controller through a serial interface. The serial communication module is used to receive this data. Once received the character stream is scanned until a new line is detected. This new line signals the end of a NMEA string. Once a full new line is received it is then split by the comma character into an array of strings. The first string indicates the message type. The message type indicates how the remainder of the strings should be parsed. The gimbal control module uses the RMC and GGA strings to obtain latitude, longitude, altitude (MSL and AGL), velocity (speed and angle), and time. [1, 12] Once both sentences have been received the data is stored in a GPS data structure and enqueued in a ring buffer to be received by higher level logic. The higher level logic then requests data to be received and it is pulled from the ring buffer and returned. It should be noted that all floating point variables are represented as fixed point Q number format variables. This makes math operations quicker and serialization simpler.

This modules does not convert all of the data fields of the GPS packet. Only the fields that are currently being converted were deemed useful for the current Stabilized Gimbal System. The code was made fairly modular so that it can be expanded to process additional types of NMEA strings.

4.1.4 Servo Control Using Waveform Generation

Servo control is quite difficult because it is a very undocumented interface. General research resulted in the following control method: send a control pulse that ranges between 1 to 2 milliseconds every 20 milliseconds. The 1 millisecond pulse corresponds to a full left deflection whereas a 2 millisecond pulse corresponds to a full right deflection. This waveform has been successful and control most servos. It has failed to control some digital servos, which is mostly believed to be caused by differences in voltage levels. This project does not require digital servos, so the waveform generated is sufficient.

Servo waveform generation is best done in hardware because it makes a more accurate signal and offloads the work from the CPU. This can be accomplished with waveform generation. The micro-controller is configured to use a timer which counts from 1 to $40,000$ every 20 milliseconds. This is done by pre-scaling the clock by a factor of 8 (so it increments once every 8 clock cycles) and setting the top value of the timer's counter to $40,000$ (so it resets to 0 every time it reaches $40,000$. The output compare value is set between $2,000$ and $4,000$ to cause the timer to output a high signal between 1 to 2 milliseconds every 20 milliseconds.

The servo control module uses Q number format to accept an angle between -90 and 90 degrees. It then converts this number to map between the values of $2,000$ and $4,000$ so that it can assign it to the output compare register for the waveform. Form there the waveform generation outputs the control signal continuously to command the servo to the desired angular value.

4.1.5 IMU Software Integration & Processing

The InvenSense 6 axis IMU uses I2C communication to set and read it's registers. These registers are used to set the modes of operation for the IMU and the read the sensor's state. The IMU also has external interrupts to notify other chips when data is available to be read. The IMU also has a FIFO buffer to queue the data so that data generated by the accelerometer and gyroscope can be synchronized before it is read. The IMU also has a Digital Motion Processor (DMP) that can calculate the state as a quaternion at a rate of 200Hz. [15]

InvenSense releases a software library through their developer portal which can be used to control the IMU. It is configured to work with the MSP430 but is designed to be integrable into other chips. [15] The first step to using this library was to replace the interrupt, clock, and I2C portions of the software library with that of the AVR. For the I2C portion I used Peter Fleury's I2C library that can be downloaded from his website. [11] Once the software was configured the IMU's interrupts could be configured, the FIFO buffer could be setup and enabled, and the accelerometer and gyroscope data could be obtained. To setup the DMP the IMU's firmware needs to be loaded at runtime. The software library has routines for this, but it could not be stored in the data segment of the Atmega128's memory. The firmware storage and loading routines were modified so that the data was stored in program memory and so that the firmware loading routine loaded it properly. The micro-controller receives an interrupt when data is ready, reads the FIFO buffer, parses the data into an IMU data structure, and then stores it in an IMU packet ring buffer. The higher level logic then reads that ring buffer to receive data packets.

4.1.6 Quaternion Representation & Math

A quaternion is another way to represent a rotation. A unit quaternion with rotation around axis v by angle θ can be seen as [27]:

$$q = (cos(\theta/2), v * sin(\theta/2))$$

The quaternion representation is useful for operations with rotations. A quaternion can be converted to Euler angles with the following formula [25]:

$$\begin{aligned} Roll = \phi &= \arctan\left(\frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)}\right) \\ Pitch = \theta &= \arcsin\left(2(q_0q_2 - q_3q_1)\right) \\ Yaw = \psi &= \text{atan}\left(\frac{2(q_0q_3 + q_1 + q_2)}{1 - 2(q_2^2 + q_3^2)}\right) \end{aligned}$$

4.1.7 Stabilization Algorithm

State estimation was accomplished using the Digital Motion Processor (DMP) on the MPU-6000. The DMP on the Inertial Measurement Unit (IMU) chosen outputs a quaternion at 200Hz that can be used directly as the state for stabilization and point at GPS. The quaternion values received from the DMP are converted to euler angles and used directly for stabilization. Point at GPS uses the same state estimate combined with the GPS data to get position and heading.

The euler angles represent the roll and pitch of the gimbal that must be compensated for. To calculate the servo angles needed to do so, the system subtracts the roll and pitch of the gimbal from the desired roll and pitch pointing direction.

4.1.8 Point at GPS Algorithm

The Point at GPS algorithm used is very simple. Given the state which consists of attitude, GPS position, and heading, and the desired GPS position to point at, you calculate a pointing vector, the quaternion needed to achieve that rotation, and then the conversion from quaternion to euler angles. The pointing vector can be calculated by computing the different in GPS position for the X-Y plane component, which can be rotated by the heading to account for reference frame, and then using the difference in altitude (assumed to by AGL of the gimbal) for the Z component.

4.2 Host-Computer Software

This section describes the host computer software library that was built to communicate with the Stabilized Gimbal System, and the test client application that was built to demonstrate the functionality of the system. The serial communication library will be used to integrate the Stabilized Gimbal System into the rest of the CUAir system. The test client application will be used to debug the gimbal system, demonstrate it at the ECE 4760 final project demonstration, and to demonstrate it at various other events like the Cornell BOOM competition.

4.2.1 CMake Build System

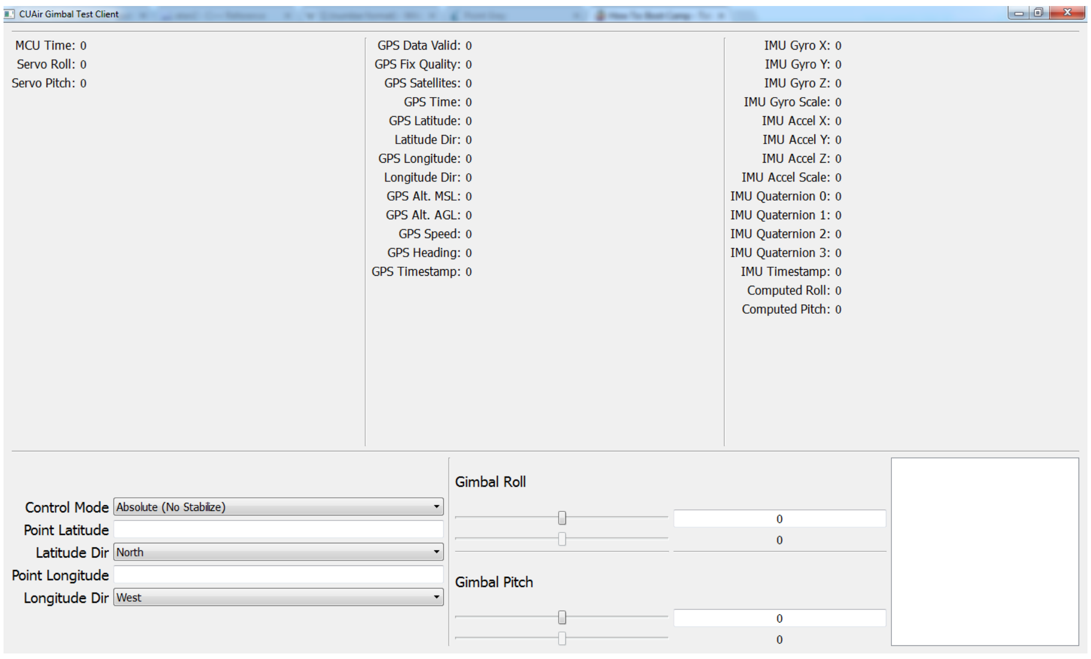
CMake is a cross-platform build system generator that can be used to decouple the build system from the operating system, the compiler, the IDE, or any other such systems. The CMake system is well documented on their website, and it has become a standard for open source project development. The Stabilized Gimbal System's C++ software systems that execute on a host computer were setup to be built using CMake to demonstrate its power, its flexibility, and its simplicity. Once the CMake build system was fully understood, it made developing the C++ side of the project much simpler. This document will not serve as documentation for the CMake tool, but simply to motivate its use.

4.2.2 Asynchronous Serial Communication Library

The asynchronous serial communication library implements the same protocol described in the micro-controller software section. This library uses the Boost asio library to manage the serial communication interface. The Serial Communication Manager class is responsible for managing the interface and implementing the serial protocol. The Gimbal Communication Manager class uses this serial communication interface to send and receive messages defined in the shared interface library. The Gimbal Communication Manager can be used to send message objects, or it can be used to register callback functions that will be executed when messages are received by the gimbal.

4.2.3 Test Client Application

The test application is a GUI which allows a user to view the gimbal messages received which contain the gimbal's current state, and also set the settings for the gimbal which will be sent to the gimbal control board via the Asynchronous Serial Communication Library. This application is designed to demonstrate functionality as well as debug problems with the gimbal. This application can also serve as a reference for the eventual integration work that will be done to integrate the Stabilized Gimbal System with the main CUAir system. Figure 24 shows the test client application running.



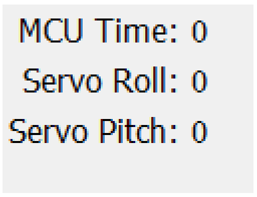
CUAir Gimbal Test Client

```
MCU Time: 0        GPS Data Valid: 0      IMU Gyro X: 0
Servo Roll: 0      GPS Fix Quality: 0     IMU Gyro Y: 0
Servo Pitch: 0     GPS Satellites: 0      IMU Gyro Z: 0
                   GPS Time: 0            IMU Gyro Scale: 0
                   GPS Latitude: 0        IMU Accel X: 0
                   Latitude Dir: 0        IMU Accel Y: 0
                   GPS Longitude: 0       IMU Accel Z: 0
                   Longitude Dir: 0       IMU Accel Scale: 0
                   GPS Alt. MSL: 0        IMU Quaternion 0: 0
                   GPS Alt. AGL: 0        IMU Quaternion 1: 0
                   GPS Speed: 0           IMU Quaternion 2: 0
                   GPS Heading: 0         IMU Quaternion 3: 0
                   GPS Timestamp: 0       IMU Timestamp: 0
                                          Computed Roll: 0
                                          Computed Pitch: 0
```

Control Mode: Absolute (No Stabilize)
Point Latitude:
Latitude Dir: North
Point Longitude:
Longitude Dir: West

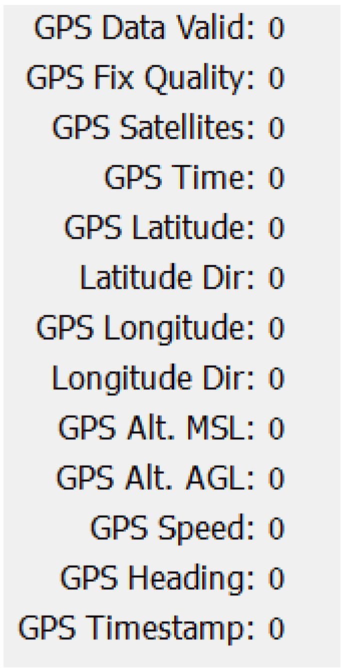
Gimbal Roll: 0 / 0
Gimbal Pitch: 0 / 0

Figure 24: Test Client Application GUI. This application is used to view the state of the Gimbal Control Board and to change settings. The application is used to verify operation of the Stabilized Gimbal System.



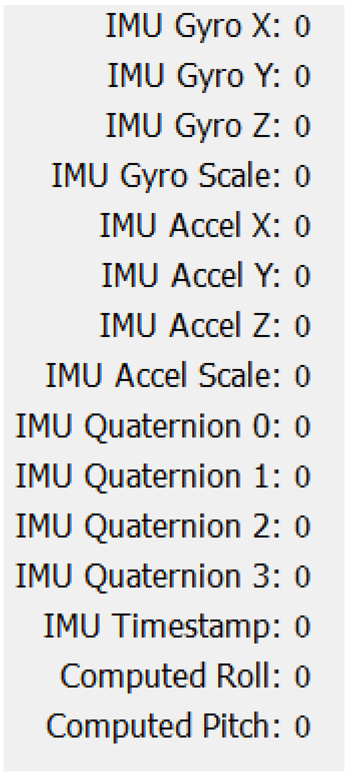
```
MCU Time: 0
Servo Roll: 0
Servo Pitch: 0
```

Figure 25: Test Client Application GUI: MCU Time and Servo Values. This portion of the Test Client Application shows the onboard MCU time (milliseconds since power-on) and the current output values for the roll and pitch servos.



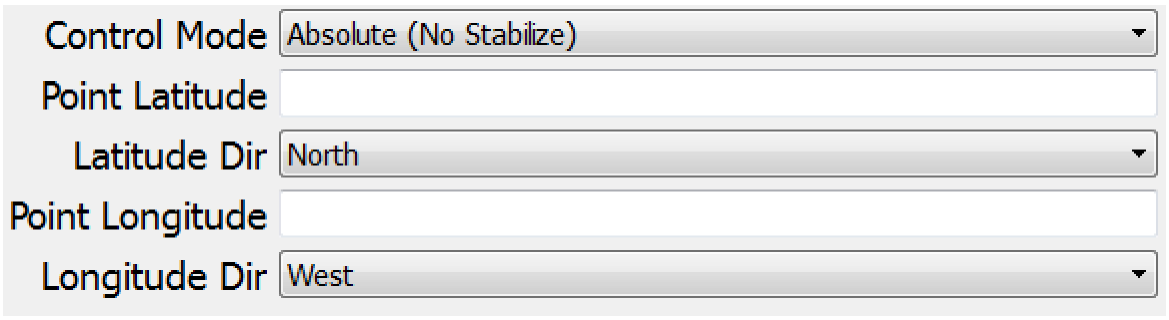
```
 GPS Data Valid: 0
GPS Fix Quality: 0
 GPS Satellites: 0
       GPS Time: 0
   GPS Latitude: 0
   Latitude Dir: 0
  GPS Longitude: 0
  Longitude Dir: 0
   GPS Alt. MSL: 0
   GPS Alt. AGL: 0
      GPS Speed: 0
    GPS Heading: 0
  GPS Timestamp: 0
```

Figure 26: Test Client Application GUI: GPS Data. This portion of the Test Client Application shows the GPS data received from the GPS module connected to the Gimbal Control Board. The GPS data is first parsed, converted to an internal representation, and then sent through serialization to the host computer that is executing the Test Client Application. This view shows proper GPS data is being received.



IMU Gyro X:	0
IMU Gyro Y:	0
IMU Gyro Z:	0
IMU Gyro Scale:	0
IMU Accel X:	0
IMU Accel Y:	0
IMU Accel Z:	0
IMU Accel Scale:	0
IMU Quaternion 0:	0
IMU Quaternion 1:	0
IMU Quaternion 2:	0
IMU Quaternion 3:	0
IMU Timestamp:	0
Computed Roll:	0
Computed Pitch:	0

Figure 27: Test Client Application GUI: Inertial Measurement Unit (IMU) Data. This portion of the Test Client Application shows the IMU data received from the MPU-6000 chip. The view shows the raw gyroscope and accelerometer data, the Digital Motion Processor (DMP) Quaternion output, and the roll and pitch angles computed from the output quaternion. This view is used to show both IMU data is being received, and to verify the computed roll and pitch values.



Control Mode	Absolute (No Stabilize)
Point Latitude	
Latitude Dir	North
Point Longitude	
Longitude Dir	West

Figure 28: Test Client Application GUI: Mode Control. This portion of the Test Client Application controls both the mode of the Stabilized Gimbal Control System, and the Point at GPS location to use when in that mode. The different modes available are absolute (point without stabilization), relative (point with stabilization), and point at GPS (with stabilization).

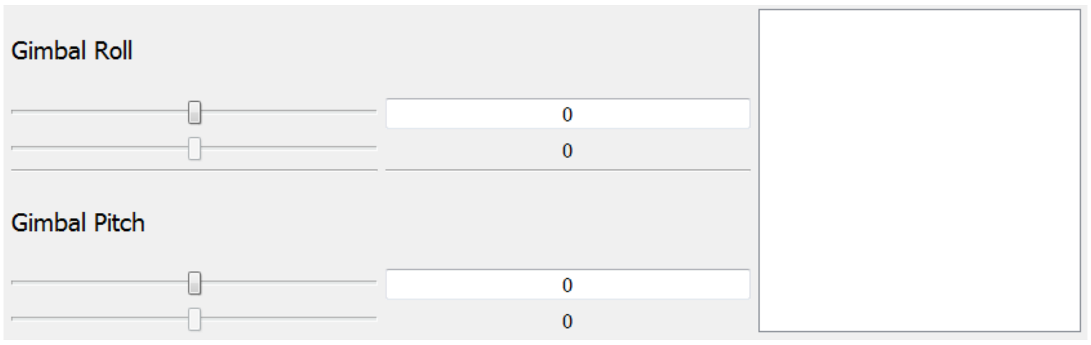


Figure 29: Test Client Application GUI: Point Control & Servo Visualization. This portion of the Test Client Application is used to control the angle at which the Stabilized Gimbal Control System points at. The sliders and text boxes can be used to specify values for one axis individually. The slider underneath the selection slider is used to visualize the output servo values on that axis. The box can be used to convert mouse coordinates to point angles for both axis simultaneously.

5 Project Results

This section describes the results of the Stabilized Gimbal Project. The first subsection describes results for the core functionality, components that needed to be completed for the project to be considered successful. The second subsection describes the results for internal stabilization and point at GPS, components which are necessary for the project to be considered optimal.

5.1 Core Functionality

Core functionality components include the asynchronous communication system, the servo actuation, the GPS data processing, and the IMU data processing. These components represent the core infrastructure that doesn't need to change between iterations of the Gimbal Control System. All of the core components are completed, tested, and performing optimally. The test client executable can be used to validate each of these components.

Asynchronous Communication System. The asynchronous communication system is used to communicate during normal gimbal operation, and during test operation. Normal operation involves communicating using the established message protocol and message data structures. The test client uses the Serial Communication Library to display the state of the gimbal and control the gimbal using command messages. This application was demonstrated during the ECE 4760 final demo and shown complete. Test operation is where various components of the board can be validated by printing the state of the micro-controller to the UART as plain text, a state representation which can be viewed with a program like Putty. This test mode was also demonstrated during the final demo, which indicates the asynchronous communication system is operating correctly. The theory of operation, which was previously discussed, shows that the system operates optimally. The team tested the communication rate and determined that the system can communicate the full state and settings at a minimum of 10Hz, which is faster than the current update rates of other CUAir systems which are typically 1Hz. The only method to improve this component would be to introduce a micro-controller which has Direct Memory Access (DMA) or to use a faster communication interface (e.g. not UART). The ECE 4760 course required an Atmega architecture micro-controller, and the interface requirements required UART, so for now this system is optimal. Although different architectures may improve the system performance, it may come at a cost of complexity and familiarity.

Servo Actuation. As previously shown in the design section, the micro-controller sends sub-degree commands to the servo. The servos and the mechanical gimbal themselves have less accuracy than the servo control signal. The final demonstration showed the servos can be controlled accurately and reliably. This system uses the PWM waveform generation, so the micro-controller CPU load is optimal (no load except to change signals). Thus, the current servo actuation mechanism is optimal. There are improvements that could be made. First, the gimbal board was unable to control certain digital servos. The team used an oscilloscope to test different signals, ones that worked and the gimbal control board which didn't, and could not find a difference other than the voltage level. The gimbal control board uses a 5V signal whereas the other controllers use a 3.3V signal. If digital servos become necessary, even though they currently are not, then a voltage divider can be integrated either off-board or in a future design revision in order to drop the control signal to 3.3V. Otherwise, the servo actuation is working perfectly.

GPS Data Processing. The GPS data is successfully being received and parsed by the Gimbal Control Board. The data is then correctly being sent to the host computer and visualized in the Test Client Application. Google Maps was used to verify that the data received from the Gimbal Control Board represents accurate position information. The GPS data is used for the point at GPS calculation, which is attempting to get a known GPS position to appear within the field of view of the camera, which currently has a field of view of 60° horizontally by 40° vertically. Thus, positional accuracy required for this project is no greater than that demonstrable by visual inspection and this was demonstrated during the final demonstration.

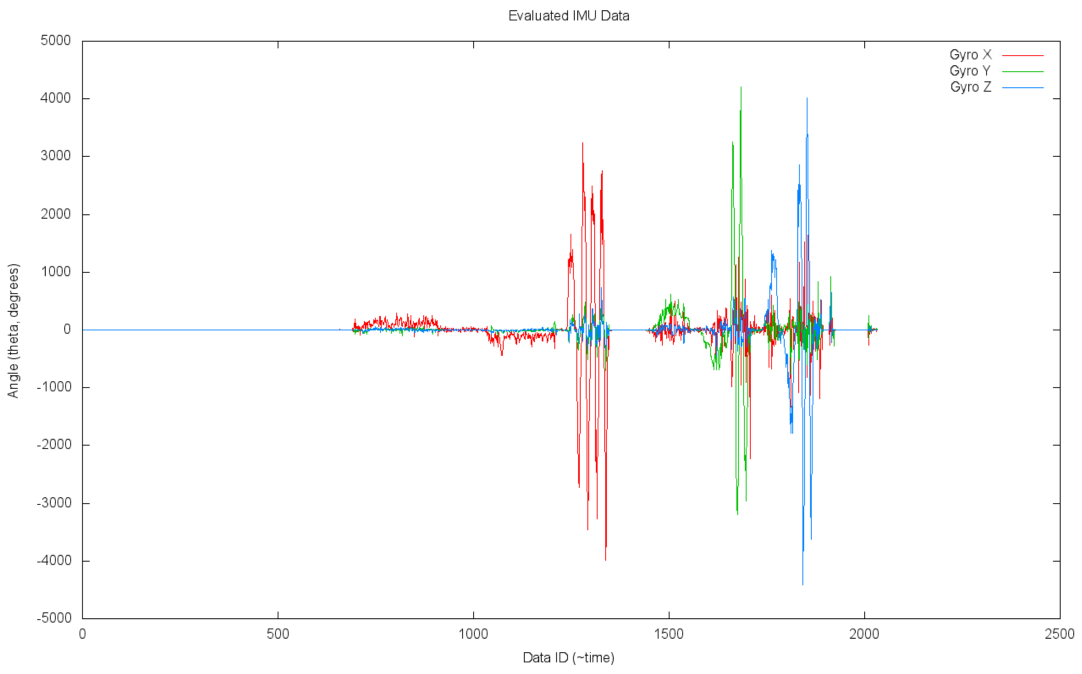
IMU Data Processing. The IMU data is successfully being received and processed as well. An offline test program was written in Python to visualize the data being generated by the IMU. Example data obtained from the IMU and then visualized by the program is shown in the graphs in Figure 30. The quaternion output and computed roll and pitch angle conversion are discussed in the next section.

5.2 Internal Stabilization & Point at GPS

This section describes the results of the internal stabilization and point at GPS algorithms. These features are non-core functionality, and were desired but not necessary. The internal stabilization was shown to work during the final demonstration, and the point at GPS system is code complete. However, the point at GPS algorithm is not tested at this time due to insufficient time remaining in the semester.

Internal Stabilization. The internal stabilization algorithm was shown to work during the ECE 4760 final project demonstration. The physical gimbal moves in response to changes in the attitude of the Gimbal Control Board. Figure 31 shows a test camera, the Point Grey Flea3, which was mounted to the gimbal in order to test stabilization. Figure 32 shows a top-down view of the full stabilization test. Figures 33 and 34 show screen-shots from the video demonstration. Figures 35 and 36 shows imagery from the camera that were used to verify the stabilization. As shown in these images and the stabilization test video, the stabilization mode of the Stabilized Gimbal System performs as needed by the CUAir team.

Point at GPS. At this time the Point at GPS algorithm is code-complete but untested because the team ran out of time before the end of the semester. The team plans to fully test point at GPS over winter break, and then test-fly the entire system using one of CUAir's aircrafts during spring semester.



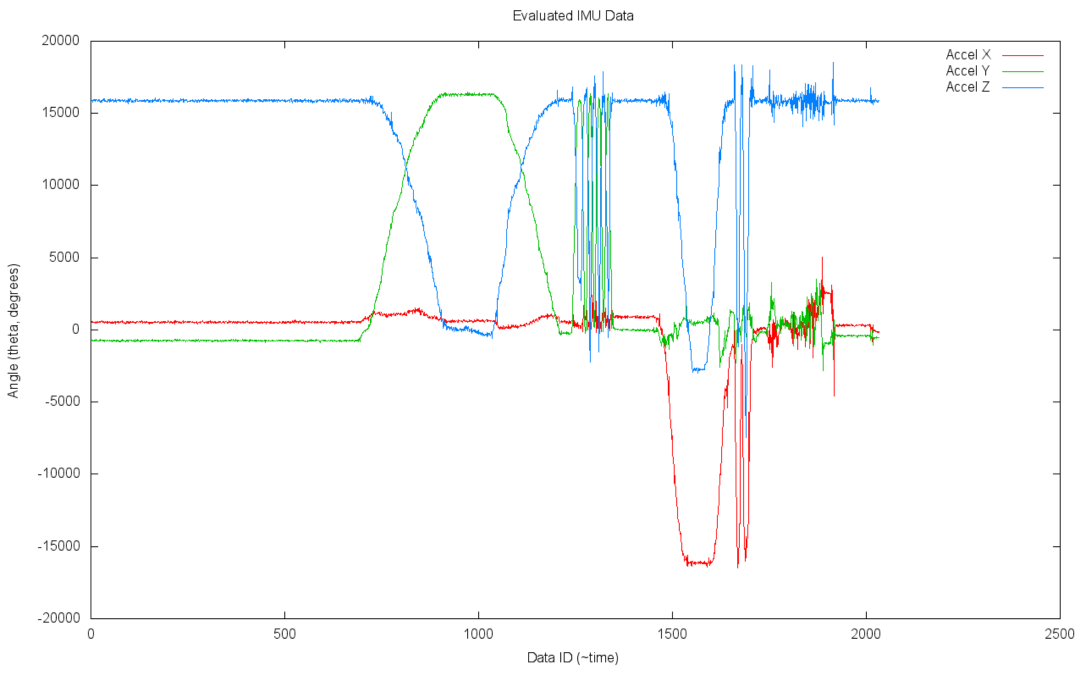


Figure 30: IMU Raw Data Output. The top graph shows the gyroscope readings and the bottom graph shows the accelerometer readings. Note that the angular degree measurements (y axis) show the output as received from the micro-controller, which is a 16 bit signed integer value. Note that the time scale is the ID of the data. The data was sampled at a high rate, between 40Hz and 200Hz. The graphs represent a few minutes of data. The data shows how it was rotated to align the board with the various axis. You can see this in the gyroscope output, where one axis dominates the magnitude of the rotation during different time periods. This can also be seen in the accelerometer output, which indicates one specific axis has approximately the entire gravity component at one time.

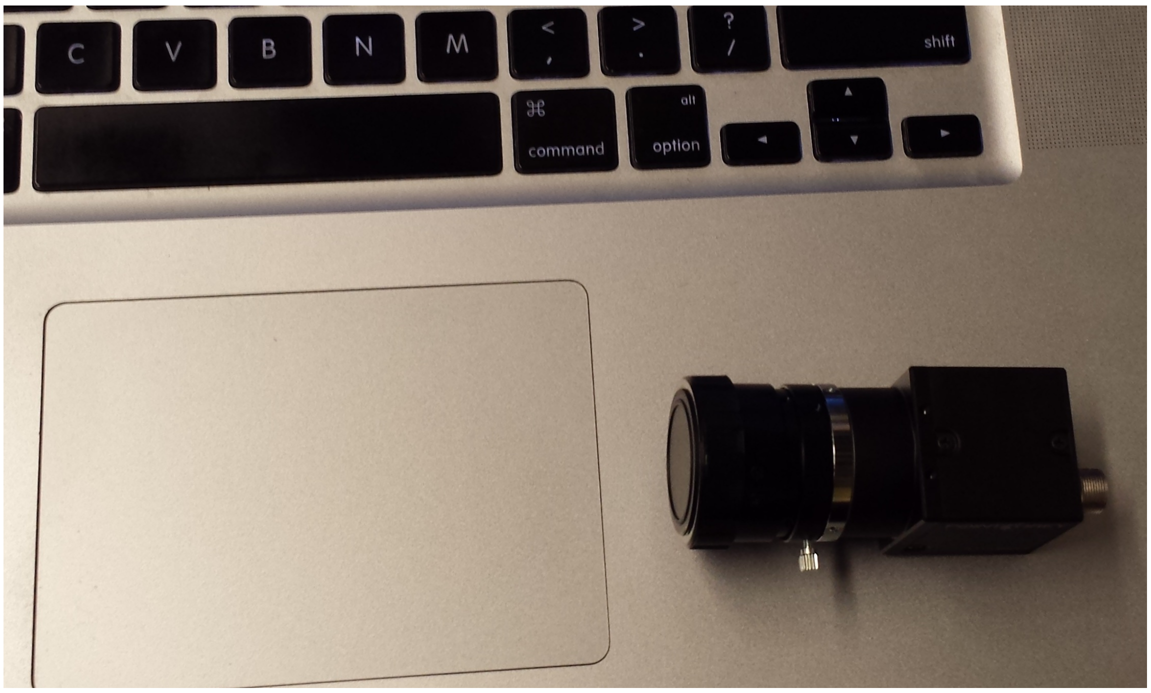


Figure 31: Point Grey Flea3 Camera which was used to test stabilization mode of the Stabilized Gimbal System.
.

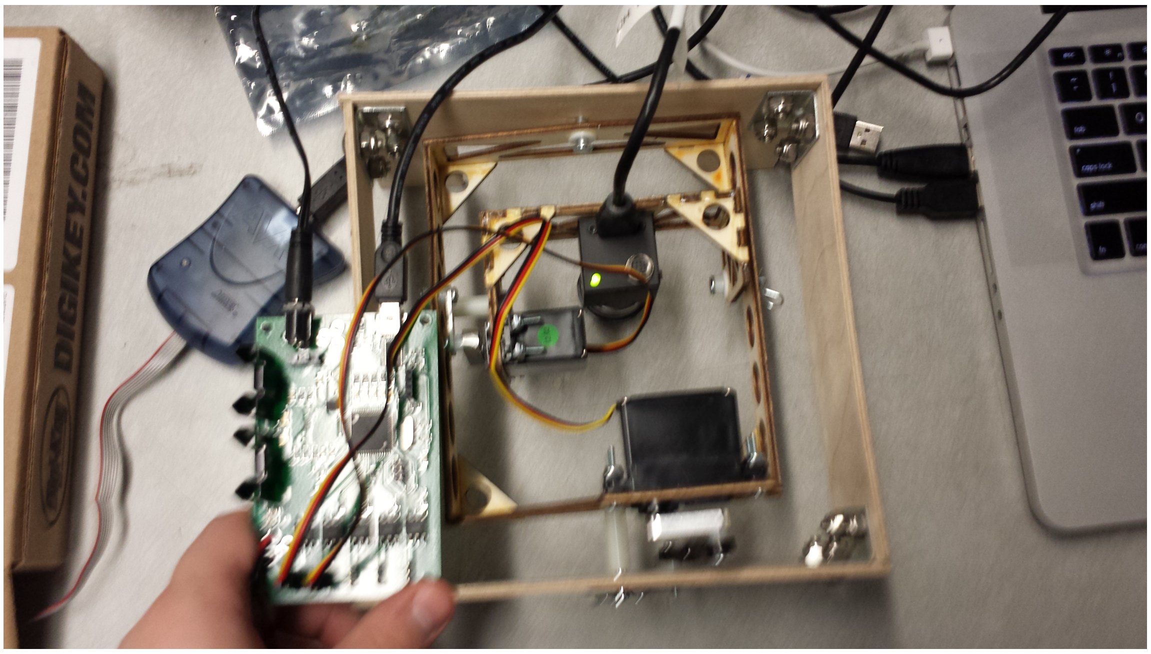


Figure 32: Point Grey Flea3 Camera which was used to test stabilization mode of the Stabilized Gimbal System.
.

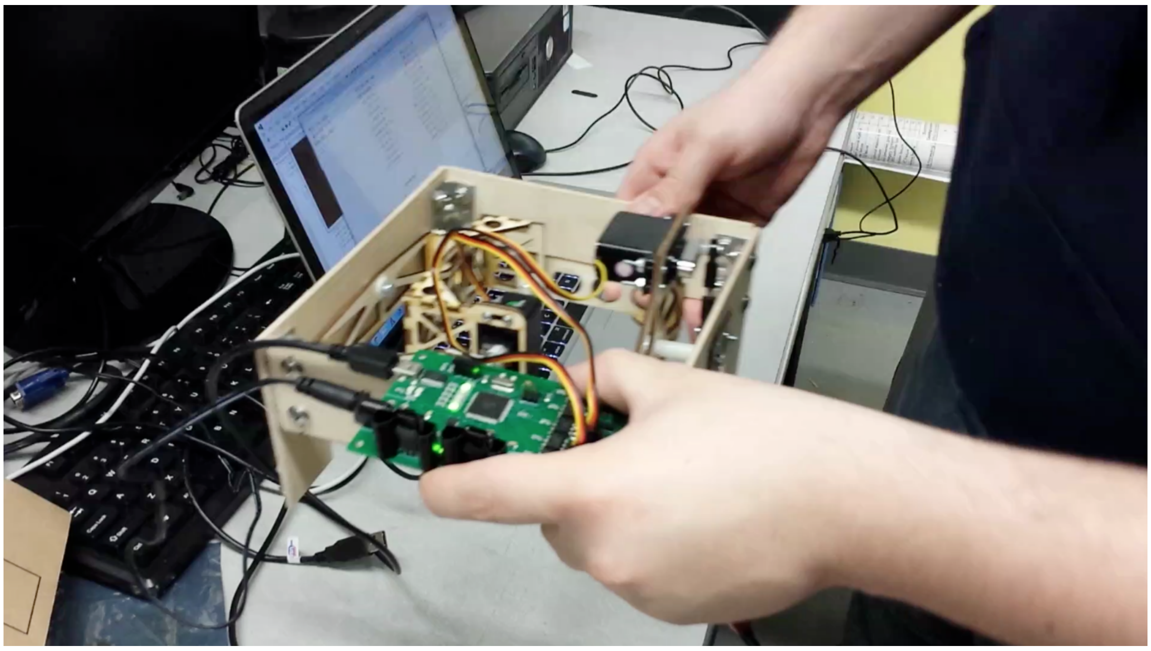


Figure 33: Screenshot from the stabilization demonstration video.

.

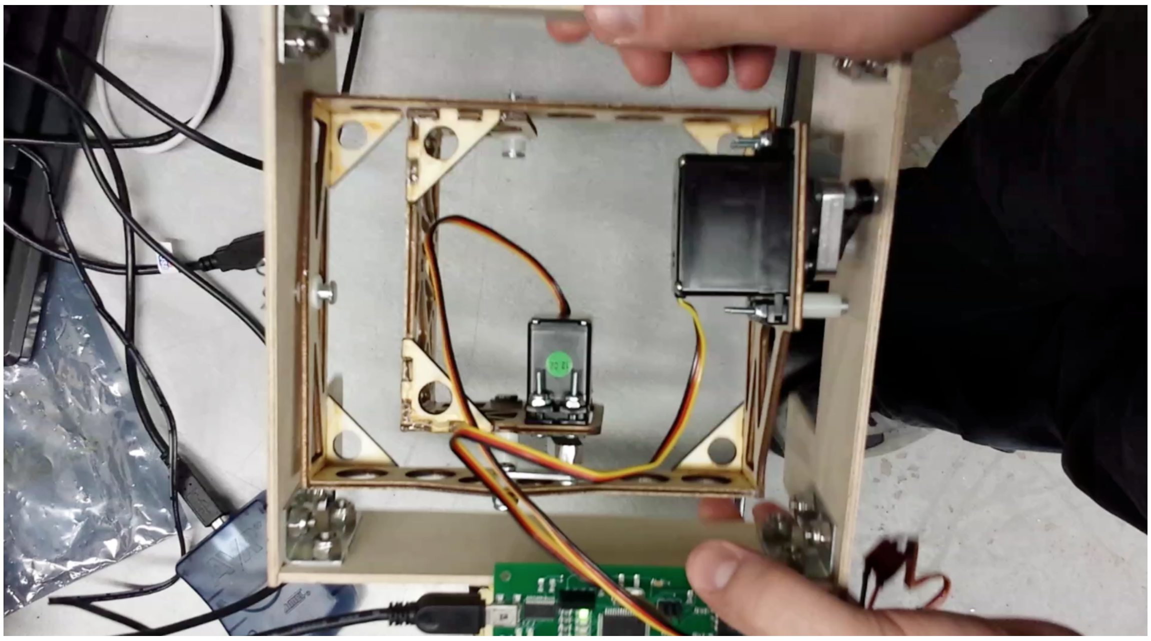


Figure 34: Screenshot from the stabilization demonstration video.

.

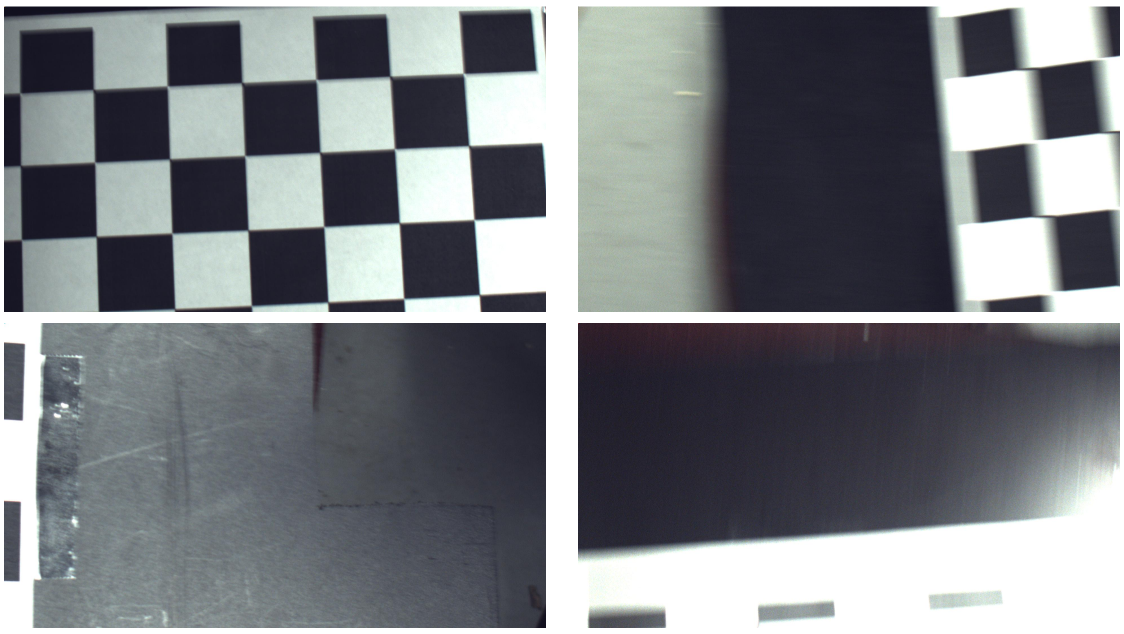


Figure 35: Pictures taken from the test camera to demonstration with and without stabilization. The Stabilized Gimbal System with the test camera mounted was placed above a checkerboard background. The images shown above are without stabilization. Notice that the background warps to demonstrate that pictures are taken from an angle. The images also show that the camera frame goes off the checkerboard background entirely in some frames.

.

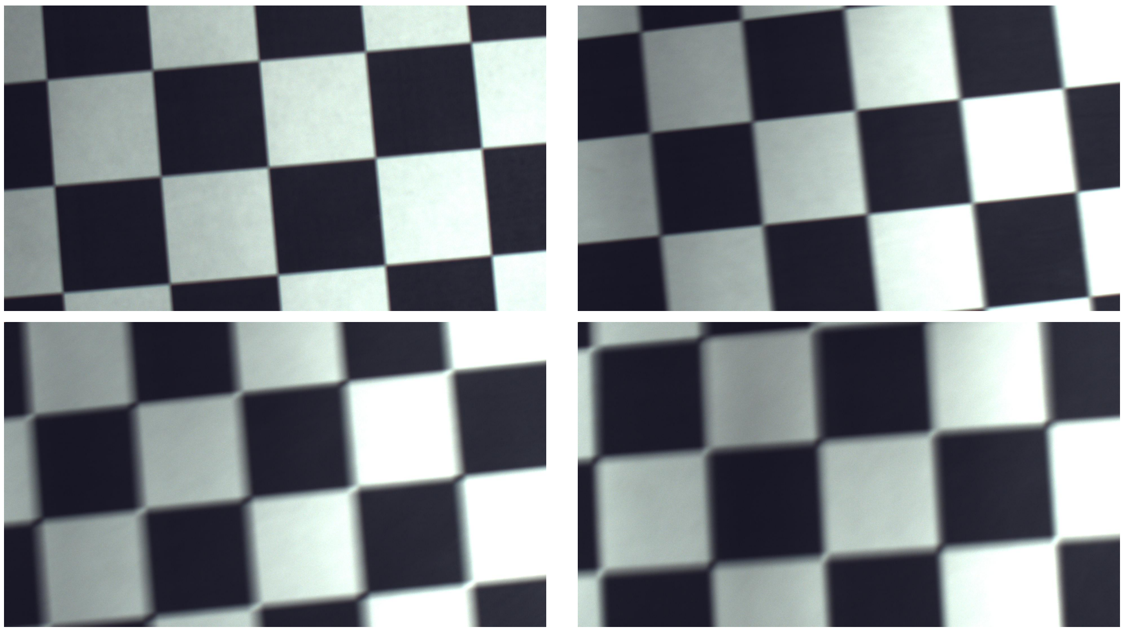


Figure 36: Pictures taken from the test camera to demonstration with and without stabilization. The Stabilized Gimbal System with the test camera mounted was placed above a checkerboard background. The images shown above are with stabilization. Notice that the background does not warp nearly as much as the un-stabilized version. The camera also does not go off the checkerboard background, and rather stays centered with the same squares in the video feed.
.

References

[1] aprs.gids.nl. Gps - nmea sentence information. http://aprs.gids.nl/nmea/, December 2013.

[2] AUVSI. Auvsi seafarer chapter. http://www.auvsi-seafarer.org/, December 2013.

[3] Advanced Circuits. Printed circuit board full spec 2-layer pcb design specials. http://www.4pcb.com/33-each-pcbs/index.html, December 2013.

[4] CMake.org. Cmake - cross platform make. http://www.cmake.org/, December 2013.

[5] CTS Electronic Components. Mp series quartz crystal datasheet. http://www.ctscorp.com/components/Datasheets/008-0308-0.pdf, December 2013.

[6] FCI Connect. Mini usb b-type smt receptacle datasheet. http://portal.fciconnect.com/Comergent//fci/drawing/10033526.pdf, December 2013.

[7] Atmel Corporation. Atmega128 atmega 128 datasheet - doc 2467. http://www.atmel.com/Images/doc2467.pdf, December 2013.

[8] LITE-ON Technology Corporation. 4n35 datasheet. http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/4N35_37.pdf, December 2013.

[9] LITE-ON Technology Corporation. Ltst-c150gkt datasheet - green led. http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTST-C150GKT.pdf, December 2013.

[10] CUAir. Cuair: Cornell university unmanned air systems. http://cuair.engineering.cornell.edu/, December 2013.

[11] Peter Fleury. Peter fleury online: Avr software. http://homepage.hispeed.ch/peterfleury/avr-software.html, December 2013.

[12] gpsinformation.org. Nmea data. http://www.gpsinformation.org/dale/nmea.htm, December 2013.

[13] Adafruit Industries. Adafruit ultimate gps breakout - 66 channel w/10 hz updates - version 3. http://www.adafruit.com/products/746#Description, December 2013.

[14] InvenSense. 6-axis platform independent solution based on the embedded motionapps 5.1 architecture. http://www.invensense.com/developers/index.php?_r=downloads&ajax=dlfile&file=Embedded_MotionDriver_v5.1.zip, December 2013.

[15] InvenSense. Mpu-6000 and mpu-6050 product specification. http://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf, December 2013.

[16] InvenSense. Mpu-6000 and mpu-6050 register map and descriptions revision 4.0. http://invensense.com/mems/gyro/documents/RM-MPU-6000A.pdf, December 2013.

[17] InvenSense. Mpu-6000/6050 six-axis (gyro + accelerometer) mems motiontracking devices. `http://www.invensense.com/mems/gyro/mpu6050.html`, December 2013.

[18] Professor Bruce Land. Cornell university ece 4760 course webpage. `http://people.ece.cornell.edu/land/courses/ece4760/`, December 2013.

[19] Future Technology Devices International Ltd. Ft232r usb uart ic ft232r usb uart ic datasheet. `http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf`, December 2013.

[20] Fairchild Semiconductor. Lm78xx / lm78xxa 3-terminal 1 a positive voltage regulator datasheet. `http://www.fairchildsemi.com/ds/LM/LM7805.pdf`, December 2013.

[21] NXP Semiconductors. Pca9306 - dual bidirectional i2c-bus and smbus voltage-level translator. `http://www.nxp.com/documents/data_sheet/PCA9306.pdf`, December 2013.

[22] Sparkfun. Dc barrel power jack/connector. `https://www.sparkfun.com/products/119`, December 2013.

[23] Sparkfun. Spdt mini power switch. `https://www.sparkfun.com/products/102`, December 2013.

[24] STMicroelectronics. Ld1117 adjustable and fixed low drop positive voltage regulator. `http://www.st.com/web/en/resource/technical/document/datasheet/CD00000544.pdf`, December 2013.

[25] Wikipedia. Conversion between quaternions and euler angles. `http://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles`, December 2013.

[26] Wikipedia. Q (number format). `http://en.wikipedia.org/wiki/Q_(number_format)`, December 2013.

[27] Wikipedia. Quaternion. `http://en.wikipedia.org/wiki/Quaternion`, December 2013.

Appendix A: Printed Circuit Board Schematics

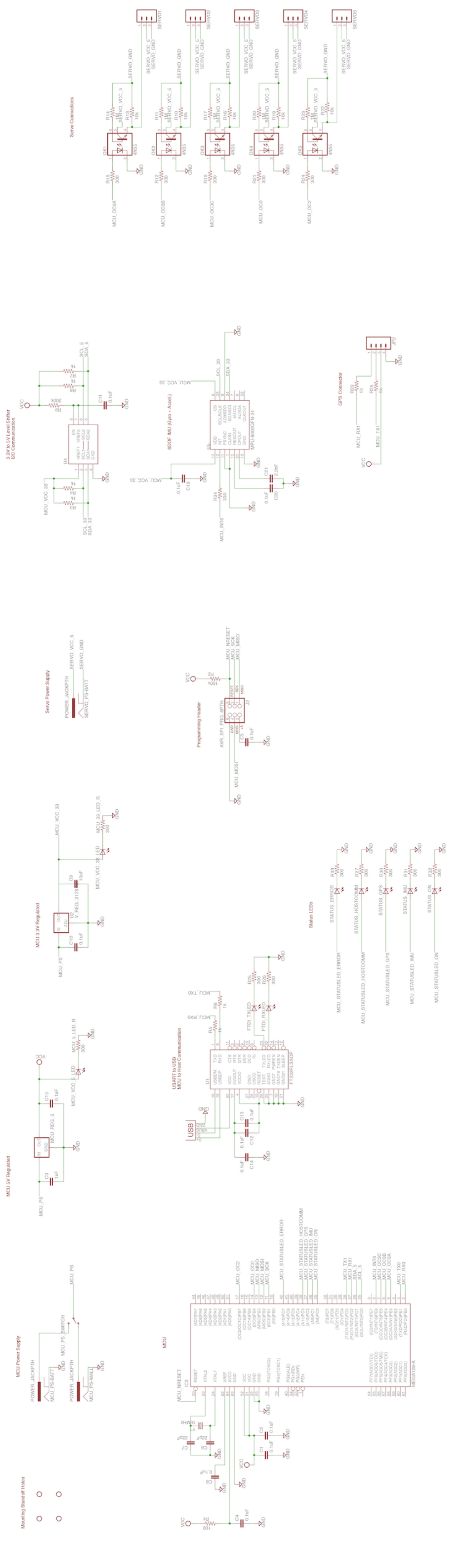


Figure 37: Gimbal Board Schematic. Original design was done in Eagle CAD.

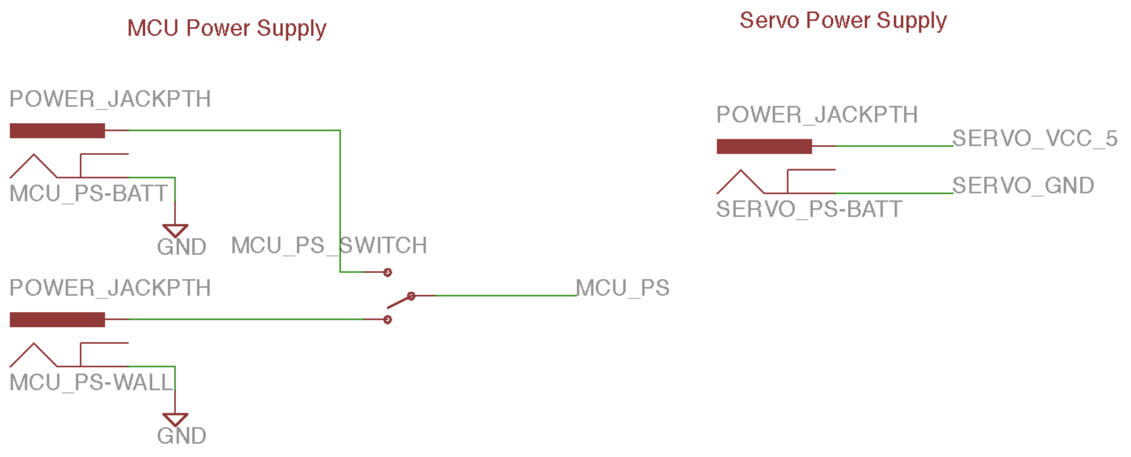


Figure 38: Gimbal Board Schematic. Power connections to the board.

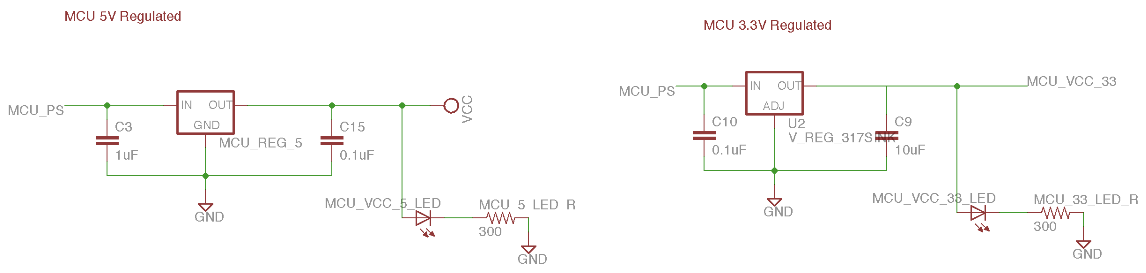


Figure 39: Gimbal Board Schematic. Voltage regulation on the board.

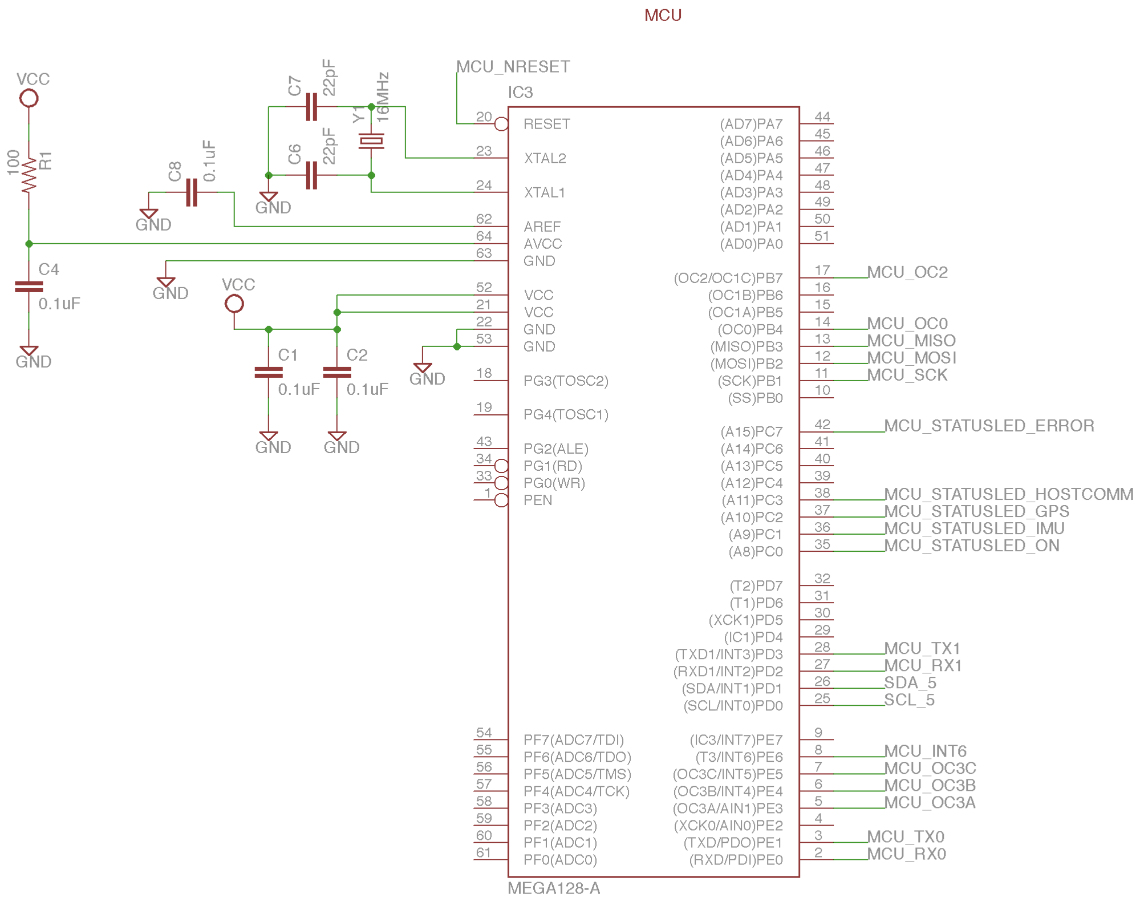


Figure 40: Gimbal Board Schematic. Micro-controller connections.

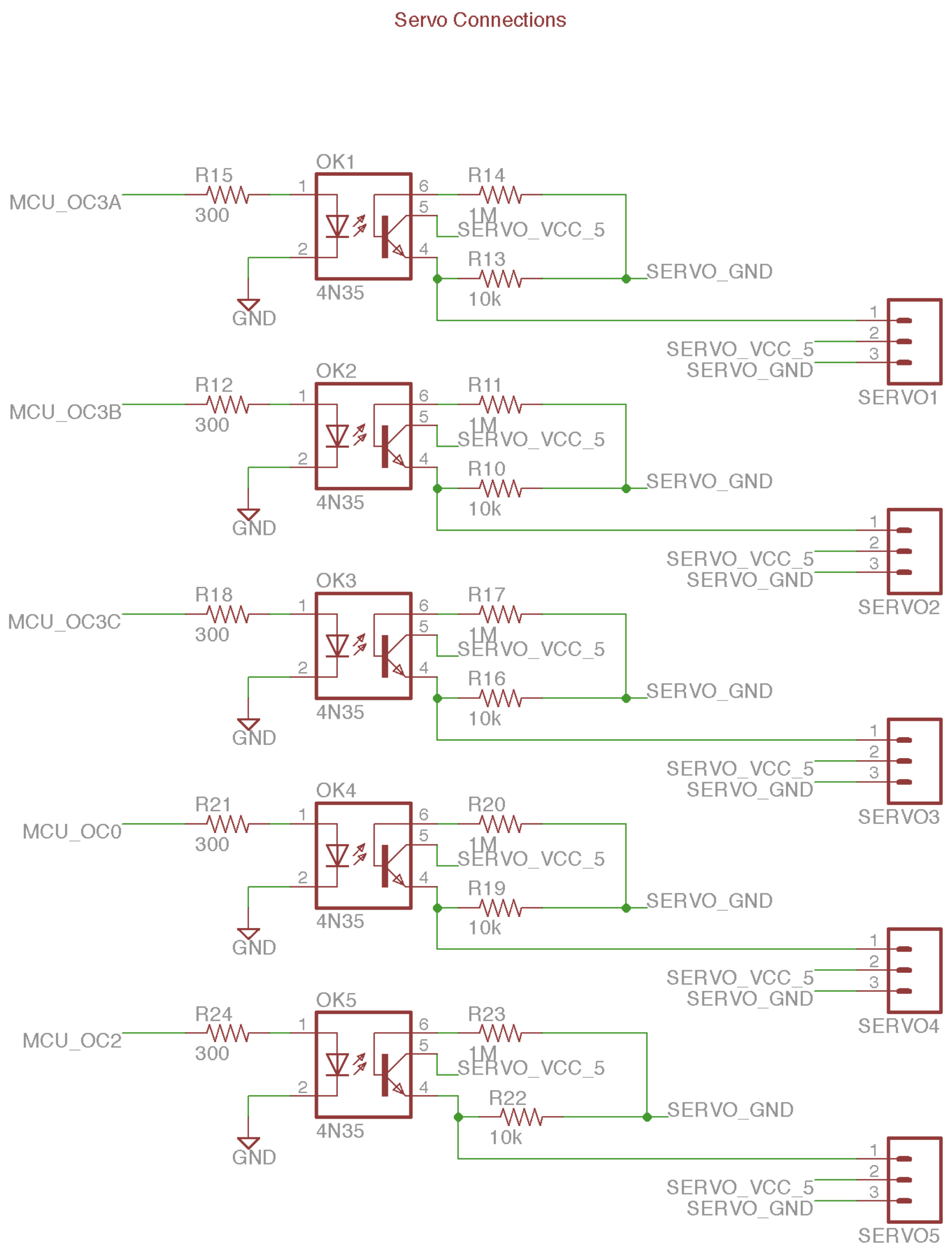


Figure 41: Gimbal Board Schematic. Servo connections on the board.

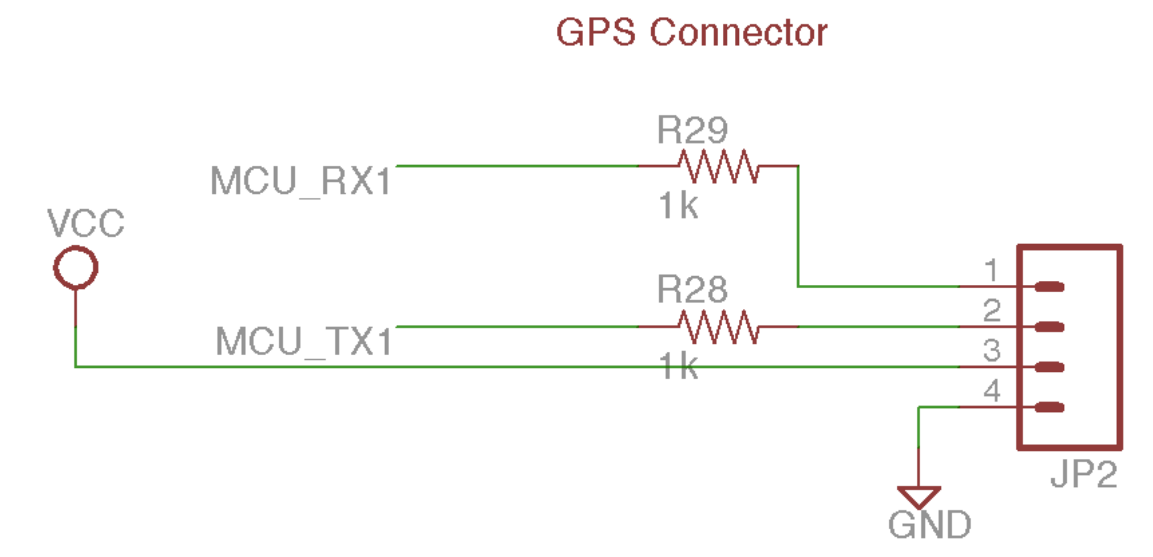


Figure 42: Gimbal Board Schematic. GPS Connections to the board.

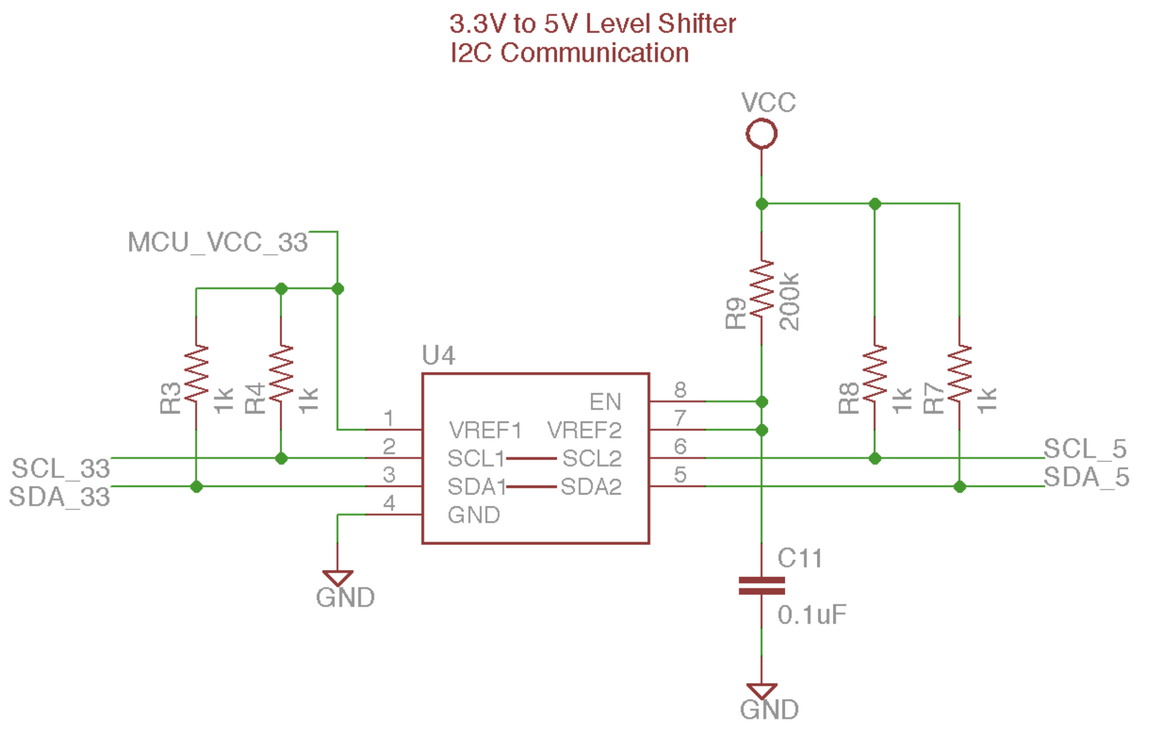


Figure 43: Gimbal Board Schematic. Logic Level Translation on the Board

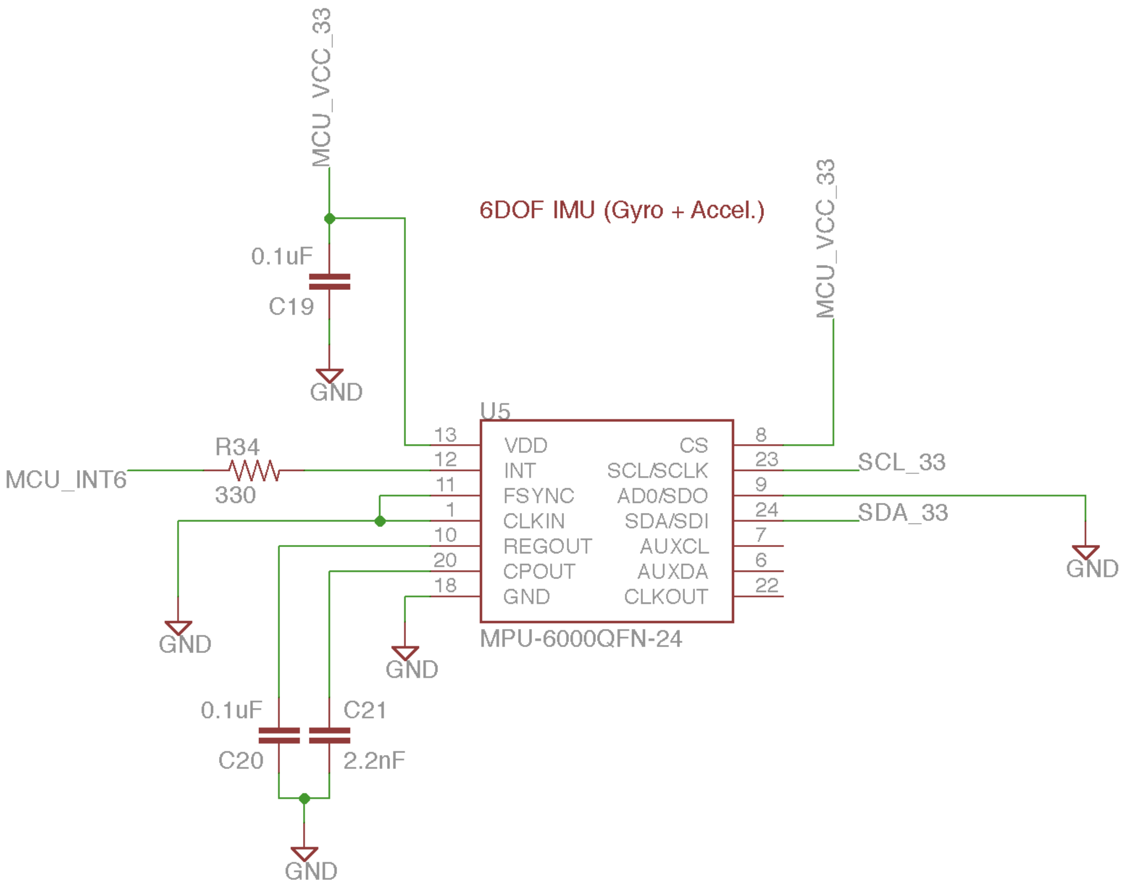


Figure 44: Gimbal Board Schematic. The Inertial Measurement Unit.

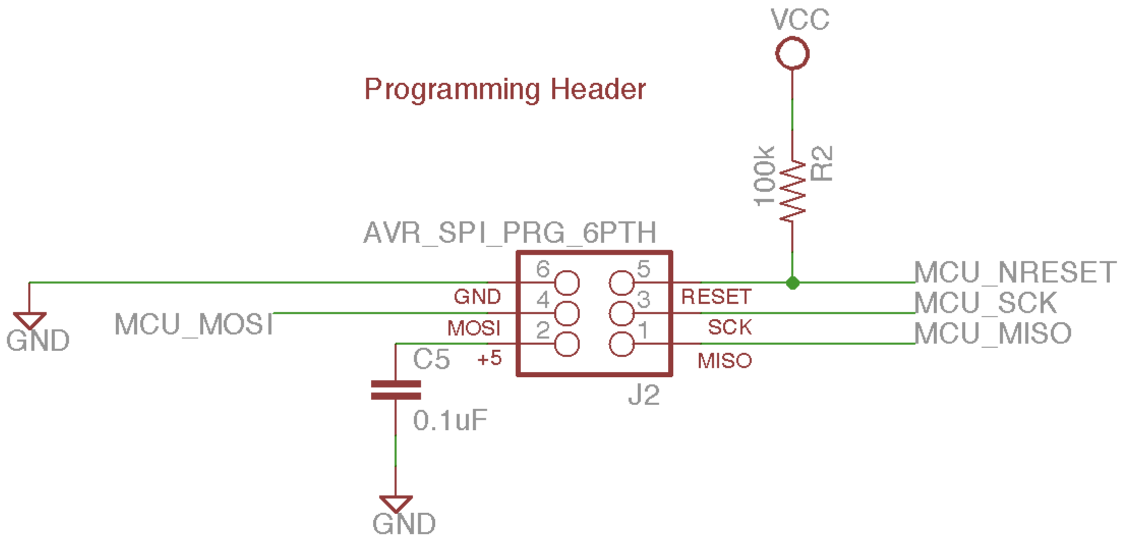


Figure 45: Gimbal Board Schematic. The Programming Header.

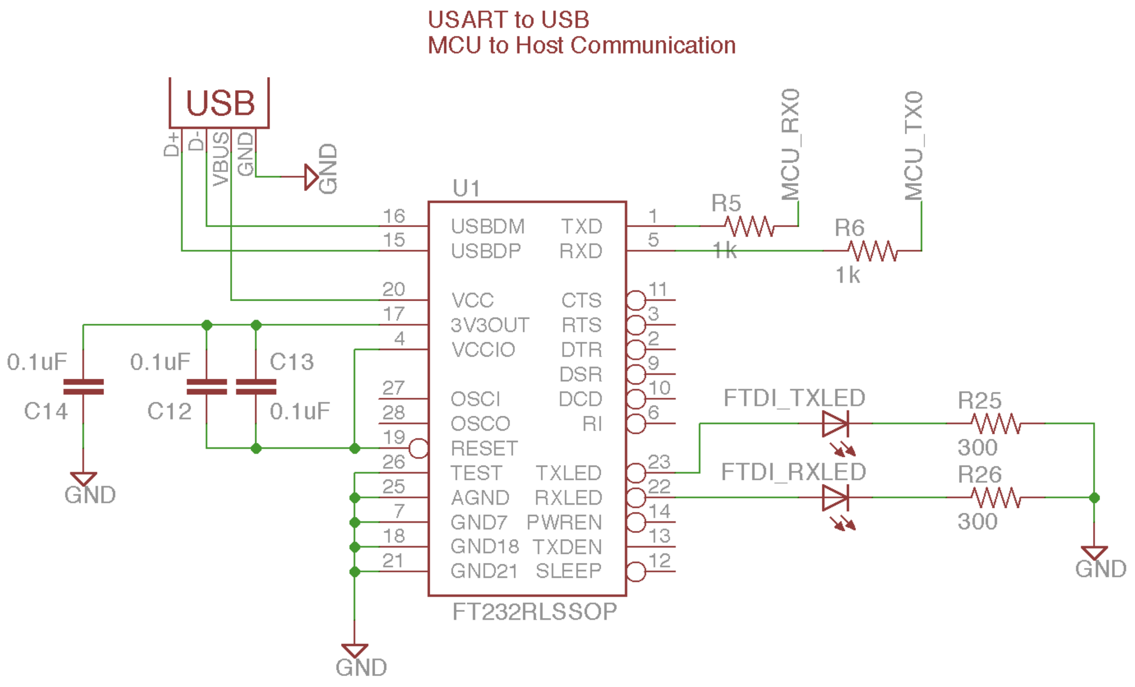


Figure 46: Gimbal Board Schematic. The FTDI Chip and USB Connection.

Appendix B: Printed Circuit Board Layout

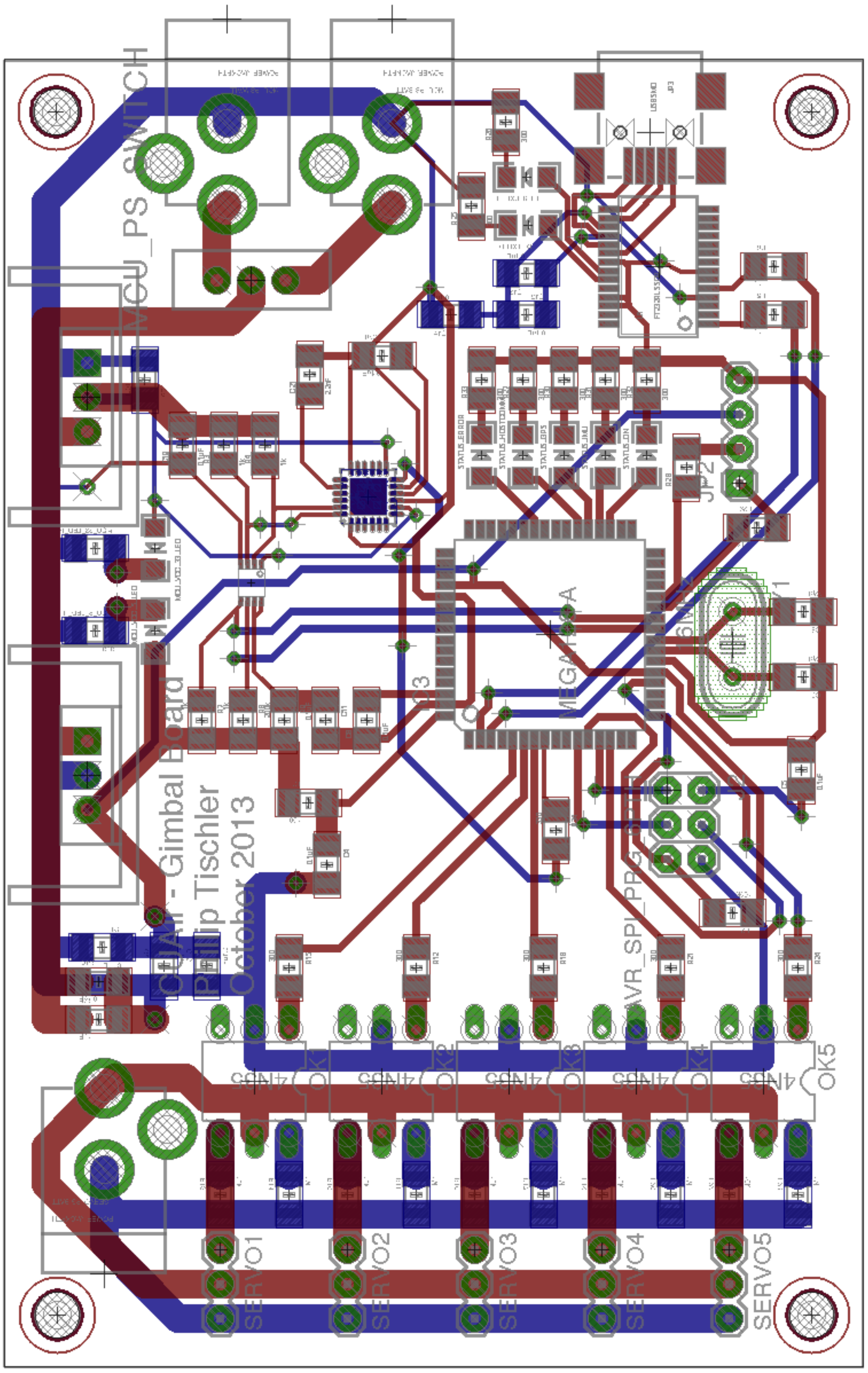


Figure 47: Gimbal Board Layout. Original design was done in Eagle CAD.

Appendix C: Assembled Gimbal Board

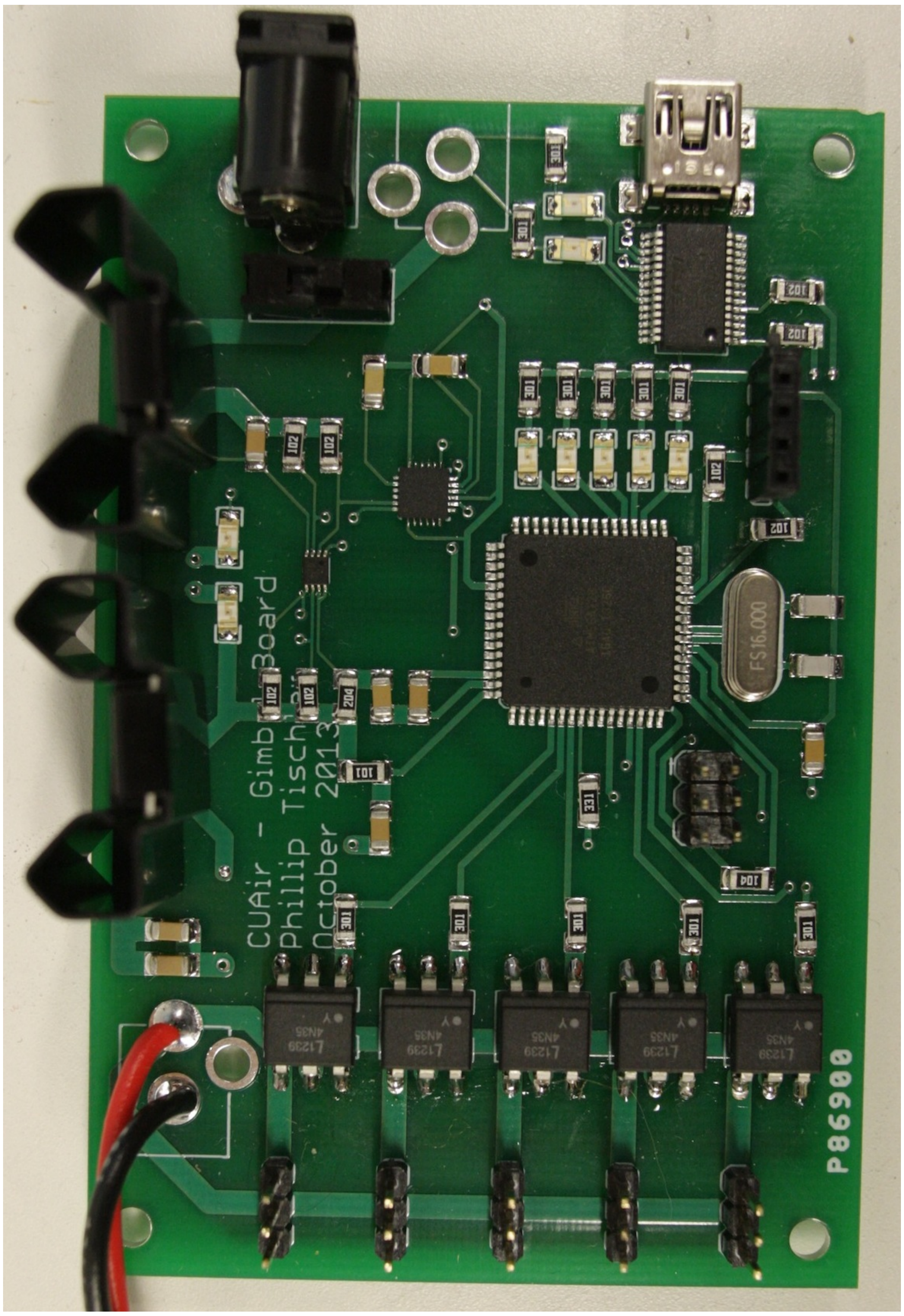


Figure 48: Assembled Gimbal Board.

Appendix D: Cost and Purchasing Details

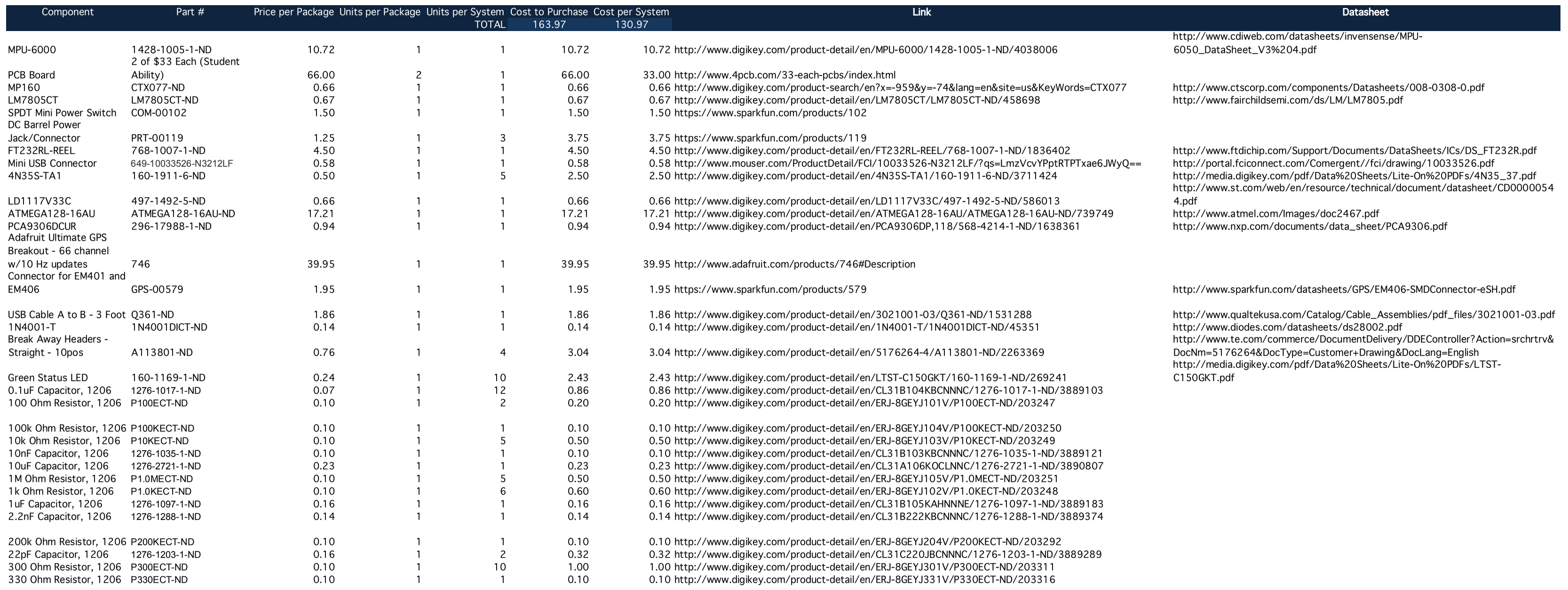
The following shows a general overview of the cost for the Gimbal Control Board. Note that the budget was waived for the ECE 4760 final project as it was for the CUAir team.

Component	Part #	Price per Package	Units per Package	Units per System	Cost to Purchase	Cost per System
				TOTAL	163.97	130.97
MPU-6000	1428-1005-1-ND	10.72	1	1	10.72	10.72
PCB Board	2 of $33 Each (Student Ability)	66.00	2	1	66.00	33.00
MP160	CTX077-ND	0.66	1	1	0.66	0.66
LM7805CT	LM7805CT-ND	0.67	1	1	0.67	0.67
SPDT Mini Power Switch	COM-00102	1.50	1	1	1.50	1.50
DC Barrel Power Jack/Connector	PRT-00119	1.25	1	3	3.75	3.75
FT232RL-REEL	768-1007-1-ND	4.50	1	1	4.50	4.50
Mini USB Connector	649-10033526-N3212LF	0.58	1	1	0.58	0.58
4N35S-TA1	160-1911-6-ND	0.50	1	5	2.50	2.50
LD1117V33C	497-1492-5-ND	0.66	1	1	0.66	0.66
ATMEGA128-16AU	ATMEGA128-16AU-ND	17.21	1	1	17.21	17.21
PCA9306DCUR	296-17988-1-ND	0.94	1	1	0.94	0.94
Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates	746	39.95	1	1	39.95	39.95
Connector for EM401 and EM406	GPS-00579	1.95	1	1	1.95	1.95
USB Cable A to B - 3 Foot	Q361-ND	1.86	1	1	1.86	1.86
1N4001-T	1N4001DICT-ND	0.14	1	1	0.14	0.14
Break Away Headers - Straight - 10pos	A113801-ND	0.76	1	4	3.04	3.04
Green Status LED	160-1169-1-ND	0.24	1	10	2.43	2.43

Table 1: Cost Spreadsheet. Part 1 of 2

Component	Part #	Price per Package	Units per Package	Units per System	Cost to Purchase	Cost per System
0.1uF Capacitor, 1206	1276-1017-1-ND	0.07	1	12	0.86	0.86
100 Ohm Resistor, 1206	P100ECT-ND	0.10	1	2	0.20	0.20
100k Ohm Resistor, 1206	P100KECT-ND	0.10	1	1	0.10	0.10
10k Ohm Resistor, 1206	P10KECT-ND	0.10	1	5	0.50	0.50
10nF Capacitor, 1206	1276-1035-1-ND	0.10	1	1	0.10	0.10
10uF Capacitor, 1206	1276-2721-1-ND	0.23	1	1	0.23	0.23
1M Ohm Resistor, 1206	P1.0MECT-ND	0.10	1	5	0.50	0.50
1k Ohm Resistor, 1206	P1.0KECT-ND	0.10	1	6	0.60	0.60
1uF Capacitor, 1206	1276-1097-1-ND	0.16	1	1	0.16	0.16
2.2nF Capacitor, 1206	1276-1288-1-ND	0.14	1	1	0.14	0.14
200k Ohm Resistor, 1206	P200KECT-ND	0.10	1	1	0.10	0.10
22pF Capacitor, 1206	1276-1203-1-ND	0.16	1	2	0.32	0.32
300 Ohm Resistor, 1206	P300ECT-ND	0.10	1	10	1.00	1.00
330 Ohm Resistor, 1206	P330ECT-ND	0.10	1	1	0.10	0.10

Table 2: Cost Spreadsheet. Part 2 of 2



Component	Part #	Price per Package	Units per Package	Units per System	Cost to Purchase	Cost per System	Link	Datasheet
				TOTAL	163.97	130.97		
MPU-6000	1428-1005-1-ND	10.72	1	1	10.72	10.72	http://www.digikey.com/product-detail/en/MPU-6000/1428-1005-1-ND/4038006	http://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf
PCB Board	2 of $33 Each (Student Ability)	66.00	2	1	66.00	33.00	http://www.4pcb.com/33-each-pcbs/index.html	
MP160	CTX077-ND	0.66	1	1	0.66	0.66	http://www.digikey.com/product-search/en?x=-959&y=-74&lang=en&site=us&KeyWords=CTX077	http://www.ctscorp.com/components/Datasheets/008-0308-0.pdf
LM7805CT	LM7805CT-ND	0.67	1	1	0.67	0.67	http://www.digikey.com/product-detail/en/LM7805CT/LM7805CT-ND/458698	http://www.fairchildsemi.com/ds/LM/LM7805.pdf
SPDT Mini Power Switch	COM-00102	1.50	1	1	1.50	1.50	https://www.sparkfun.com/products/102	
DC Barrel Power Jack/Connector	PRT-00119	1.25	1	3	3.75	3.75	https://www.sparkfun.com/products/119	
FT232RL-REEL	768-1007-1-ND	4.50	1	1	4.50	4.50	http://www.digikey.com/product-detail/en/FT232RL-REEL/768-1007-1-ND/1836402	http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
Mini USB Connector	649-10033526-N3212LF	0.58	1	1	0.58	0.58	http://www.mouser.com/ProductDetail/FCI/10033526-N3212LF/?qs=LmzVcvYPptRTPTxae6JWyQ==	http://portal.fciconnect.com/Comergent//fci/drawing/10033526.pdf
4N35S-TA1	160-1911-6-ND	0.50	1	5	2.50	2.50	http://www.digikey.com/product-detail/en/4N35S-TA1/160-1911-6-ND/3711424	http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/4N35_37.pdf
LD1117V33C	497-1492-5-ND	0.66	1	1	0.66	0.66	http://www.digikey.com/product-detail/en/LD1117V33C/497-1492-5-ND/586013	http://www.st.com/web/en/resource/technical/document/datasheet/CD00000544.pdf
ATMEGA128-16AU	ATMEGA128-16AU-ND	17.21	1	1	17.21	17.21	http://www.digikey.com/product-detail/en/ATMEGA128-16AU/ATMEGA128-16AU-ND/739749	http://www.atmel.com/Images/doc2467.pdf
PCA9306DCUR	296-17988-1-ND	0.94	1	1	0.94	0.94	http://www.digikey.com/product-detail/en/PCA9306DP,118/568-4214-1-ND/1638361	http://www.nxp.com/documents/data_sheet/PCA9306.pdf
Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates	746	39.95	1	1	39.95	39.95	http://www.adafruit.com/products/746#Description	
Connector for EM401 and EM406	GPS-00579	1.95	1	1	1.95	1.95	https://www.sparkfun.com/products/579	http://www.sparkfun.com/datasheets/GPS/EM406-SMDConnector-eSH.pdf
USB Cable A to B - 3 Foot	Q361-ND	1.86	1	1	1.86	1.86	http://www.digikey.com/product-detail/en/3021001-03/Q361-ND/1531288	http://www.qualtekusa.com/Catalog/Cable_Assemblies/pdf_files/3021001-03.pdf
1N4001-T	1N4001DICT-ND	0.14	1	1	0.14	0.14	http://www.digikey.com/product-detail/en/1N4001-T/1N4001DICT-ND/45351	http://www.diodes.com/datasheets/ds28002.pdf
Break Away Headers - Straight - 10pos	A113801-ND	0.76	1	4	3.04	3.04	http://www.digikey.com/product-detail/en/5176264-4/A113801-ND/2263369	http://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=5176264&DocType=Customer+Drawing&DocLang=English
Green Status LED	160-1169-1-ND	0.24	1	10	2.43	2.43	http://www.digikey.com/product-detail/en/LTST-C150GKT/160-1169-1-ND/269241	http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTST-C150GKT.pdf
0.1uF Capacitor, 1206	1276-1017-1-ND	0.07	1	12	0.86	0.86	http://www.digikey.com/product-detail/en/CL31B104KBCNNNC/1276-1017-1-ND/3889103	
100 Ohm Resistor, 1206	P100ECT-ND	0.10	1	2	0.20	0.20	http://www.digikey.com/product-detail/en/ERJ-8GEYJ101V/P100ECT-ND/203247	
100k Ohm Resistor, 1206	P100KECT-ND	0.10	1	1	0.10	0.10	http://www.digikey.com/product-detail/en/ERJ-8GEYJ104V/P100KECT-ND/203250	
10k Ohm Resistor, 1206	P10KECT-ND	0.10	1	5	0.50	0.50	http://www.digikey.com/product-detail/en/ERJ-8GEYJ103V/P10KECT-ND/203249	
10nF Capacitor, 1206	1276-1035-1-ND	0.10	1	1	0.10	0.10	http://www.digikey.com/product-detail/en/CL31B103KBCNNNC/1276-1035-1-ND/3889121	
10uF Capacitor, 1206	1276-2721-1-ND	0.23	1	1	0.23	0.23	http://www.digikey.com/product-detail/en/CL31A106KOCLNNC/1276-2721-1-ND/3890807	
1M Ohm Resistor, 1206	P1.0MECT-ND	0.10	1	5	0.50	0.50	http://www.digikey.com/product-detail/en/ERJ-8GEYJ105V/P1.0MECT-ND/203251	
1k Ohm Resistor, 1206	P1.0KECT-ND	0.10	1	6	0.60	0.60	http://www.digikey.com/product-detail/en/ERJ-8GEYJ102V/P1.0KECT-ND/203248	
1uF Capacitor, 1206	1276-1097-1-ND	0.16	1	1	0.16	0.16	http://www.digikey.com/product-detail/en/CL31B105KAHNNNE/1276-1097-1-ND/3889183	
2.2nF Capacitor, 1206	1276-1288-1-ND	0.14	1	1	0.14	0.14	http://www.digikey.com/product-detail/en/CL31B222KBCNNNC/1276-1288-1-ND/3889374	
200k Ohm Resistor, 1206	P200KECT-ND	0.10	1	1	0.10	0.10	http://www.digikey.com/product-detail/en/ERJ-8GEYJ204V/P200KECT-ND/203292	
22pF Capacitor, 1206	1276-1203-1-ND	0.16	1	2	0.32	0.32	http://www.digikey.com/product-detail/en/CL31C220JBCNNNC/1276-1203-1-ND/3889289	
300 Ohm Resistor, 1206	P300ECT-ND	0.10	1	10	1.00	1.00	http://www.digikey.com/product-detail/en/ERJ-8GEYJ301V/P300ECT-ND/203311	
330 Ohm Resistor, 1206	P330ECT-ND	0.10	1	1	0.10	0.10	http://www.digikey.com/product-detail/en/ERJ-8GEYJ331V/P330ECT-ND/203316	

Figure 49: Original Spreadsheet Used for Purchasing

Appendix E: Code Statistics

Language	Files	Comments	Code
C	17	1679	4391
C/C++ Header	26	818	1253
Python	1	23	133
C++	9	252	851
CMake	3	36	64
TOTAL	56	2808	6692

Table 3: Code statistics for the code that was written by Phillip Tischler during the course of the ECE 4760 final project

Language	Files	Comments	Code
C	41	4591	13938
C/C++ Header	54	2372	3433
Python	4	146	2101
C++	9	252	851
CMake	3	36	64
TOTAL	111	7397	20387

Table 4: Code statistics for the entire Stabilized Gimbal System. This includes the libraries and components that were integrated into the system.

Appendix F: Tasks Performed By Members of the Team

Table 5 shows all tasks that were necessary to complete the project and who completed the tasks. Phillip Tischler and Kelly Glynn are the project members for the ECE 4760 Final Project. Joel Heck and Derek Faust are members of CUAir who were also on the project but not in the ECE 4760 Final Project.

The project was for the CUAir team. Joel Heck is the Team Lead of CUAir, and was thus responsible for funding the project and making any purchases. Subsequently he was the person who actually performed any purchasing of components. Derek Faust is another member of CUAir. He was responsible for building the physical gimbal. Construction of the physical gimbal was not part of the ECE 4760 Final Project. The final project's scope was limited to the design of the electrical hardware and the development of the micro-controller and host computer code.

Task	Person Who Completed Task
Project Idea	Phillip Tischler
Project Proposal	Phillip Tischler
System Design	Phillip Tischler
Electronic Component Selection	Phillip Tischler
Printed Circuit Board Design	Phillip Tischler
Creation of Excel Sheet for Purchasing	Phillip Tischler
Purchasing of Components	Joel Heck
Assembly of PCB Board 1	Phillip Tischler & Joel Heck
Assembly of PCB Board 2	Phillip Tischler
Physical Gimbal Design and Construction	Derek Faust
Custom Micro-Controller Software	Phillip Tischler
Custom Host-Computer Software	Phillip Tischler
Identification of Peter Fleury's I2C Code	Kelly Glynn
Integration of Peter Fleury's I2C Code	Phillip Tischler
Integration of InvenSense IMU Code	Phillip Tischler
Identification of Wikipedia's Math for Converting Quaternion to Euler Angles	Kelly Glynn
Code for Quaternion Conversion	Phillip Tischler
Stabilization Algorithm	Phillip Tischler
Point at GPS Algorithm	Phillip Tischler
Gimbal Control System Report (This Document)	Phillip Tischler
ECE 4760 Final Report Website	Phillip Tischler

Table 5: Tasks and Person Responsible for the Stabilized Gimbal System Project