MicroCART 2016-2017

Project Plan

Group: May1716

Team Members

Brendan Bartels – *Controls Software Key Concept Holder*
Kristopher Burney – *Ground Station Key Concept Holder*
Joseph Bush – *Quadcopter Software Key Concept Holder*
Jacob Drahos – *Team Webmaster*
Eric Middleton – *Hardware Maintainer*
Tara Mina – *Team Communications Leader*
Andrew Snawerdt – *Controls Systems Key Concept Holder*
David Wehr – *Team Leader*

Advisors

Dr. Phillip Jones
Dr. Nicola Elia

Team Email: *may1716@iastate.edu*

Team Website: *http://may1716.sd.ece.iastate.edu*

Revision Date: *Monday, November 28th 2016*
Version: 3.0

Contents

Figures

Tables

1 Definition of Terms

Below are some of the terms we use regularly in this project plan document:

Term	Description
CLI	Command Line Interface, will be used to take in user input to get data from the quadcopter and give it flight commands
ESC	Electronic Speed Controller, a component between the Zybo board and the motor, which takes in a PWM signal from the board and converts this to an amplified current to control the speed of the motor
GUI	Graphical User Interface, will be used to take in user input to get data from the quadcopter and give it flight commands
LiPo	Lithium-ion Polymer battery, the type of batteries used on the quadcopter
LiDAR	Light Detection and Ranging, a laser used to detect and measure distances (ranging)
MicroCART	Microprocessor-Controlled Aerial Robotics Team, our senior design team
PCB	Printed Circuit Board, will be designed to include a battery voltage regulator and power distributor for improved power management on the quadcopter
PID	Proportional-Integral-Derivative, a commonly used type of feedback controller
PWM	Pulse-Width-Modulation, a digital signal with varying duty cycle but constant period, which is received by each of the ESCs to control the speed of each motor

Table 1: Definition of Terms

2 Introduction

The MicroCART (Micro-processor Controlled Aerial Robotics Team) senior design project has been passed down from team to team since 2006, developing a quadcopter for research purposes. Up to now, the MicroCART quadcopter (**Figure 1**) has been flying in the Distributed Sensing and Decision Making Laboratory, using the twelve-camera infrared tracking system for navigation. The quadcopter has been gradually improved by the work of each senior design team, and currently has much of the hardware necessary to accomplish our goals for this academic year.

Figure 1: MicroCART Quadcopter

2.1 Project Statement

We intend to create a modular platform for research in controls and embedded systems, building on the current system developed by previous teams. In addition to improving the modularity of the quadcopter system, we intend to advance its abilities and functions, including developing the ability to fly autonomously through a sequence of user-specified waypoints and to fly outdoors, without the help of the infrared tracking camera system in lab.

2.2 Purpose

By creating a modular platform, we wish for any controls student to be able to design their own controller, including new types of controllers, and test it with our system easily. In addition, we hope to develop a system that can execute more impressive abilities, including autonomous flight, flying outdoors using GPS navigation, and also potentially performing advanced flight maneuvers. Having these

more impressive capabilities to demonstrate will help excite new students, give them a friendly, hands-on platform to learn about controllers, and better represent the talents of our department to visitors, other interested students and faculty members. Furthermore, if we develop a platform that works well and is easy to use, this system will likely be used and added into the second semester course for controls systems, EE 476, where it can help students gain a more intuitive understanding of PID controllers and give them a platform to test their designs.

2.3 Goals

We plan to design several PID controllers for each direction of movement. Our plan will differ from approaches used in the past, deriving the controller from a systematically developed mathematical model of quadrotor dynamics, in accordance with the approach developed in a thesis written by one of the advising graduate students. Thus, we will initially conduct system identification of our quadcopter by taking measurements and performing data analysis, and afterwards utilize these parameters to characterize and implement the linear control systems necessary for controlling the movement of the quadcopter. We will also create a user interface that can be used to select waypoints which the quadcopter will follow while in autonomous flight. Eventually, we also want the quadcopter to be able to fly outdoors, without the infrared camera tracking system in the lab, by using GPS for navigation instead and a LIDAR sensor to detect the presence of obstacles located underneath the quadcopter.

Our advisors, Dr. Jones and Dr. Elia, had some high-level goals for what he wanted from our project, which can be summarized by the following points:

- Build a modular system for the quadcopter, so that each component can be easily removed and replaced by another of the same functionality, without breaking the operation of the entire system
- Design a control system for the quadcopter from a mathematical model representation of the quadcopter system, rather than from iterative testing procedures as done from previous teams
- Decrease the communication latency between the quadcopter and the ground station to be less than 10 milliseconds on average, to increase the stability of the system overall, which will most likely require WiFi communication rather than the current Bluetooth communication system
- Ensure that the communication system and controls system work together robustly, such that if any data packets are dropped, the quadcopter does not lose control and potentially damage itself
- Create a user-friendly GUI with nice multi-panel layout and click navigation features for the user to easily specify waypoints for the quadcopter to travel between
- Throughout the development process, create detailed and user-friendly documentation and tutorials for future MicroCART teams to follow and learn from, including video tutorials as well, and keeping the two-decade-long Wiki page up-to-date

3 Deliverables

The quadcopter system can be compartmentalized into three major subsystems or sub-projects: the development of the quadcopter software, the ground station, and the PID controls systems on the quadcopter. We will need to work on each of these components to transform it to a particular, useful state which will be essential to meet our objectives. In addition, there are overall, all-encompassing aspects of the project that we need to address, including improving on the modularity of the system.

3.1 Current Quadcopter Software

3.1.1 Autonomous Flight

The quadcopter has an autonomous mode in which it attempts to remain at a fixed point in space using the infrared camera tracking system in the lab. It is currently unstable and unreliable. Our team plans to:

- Update the controllers to improve the performance of this autonomous ability.
- Add the ability to receive waypoints from the ground station and fly towards the given point, regardless of the current location. (within the Coover 3050 IR tracking system)

Figure 2: Tracking System Camera

3.1.2 Improved Communication System

Communication between the ground station and quadcopter is currently done using bluetooth, which limits the range, bandwidth, and latency of our communication. On the quadcopter, we will:

- Replace the bluetooth receiver with a WiFi receiver, which will improve these three metrics [1]
- Design a communication system that will support both low-latency transfer of real-time data over UDP, as well as reliable data transfer for TCP

- This will require us to update the command structure on the quadcopter to differentiate between UDP and TCP

3.1.3 Support Flying without Ground Station

The current quadcopter software requires that the ground station be connected in order for any flight to start, which is unnecessary for flight modes that don't require data from the camera system. We plan to:

- Remove this requirement for any flight mode not requiring data from the camera system, such as manual and GPS-based autonomous flight modes
- Add a watchdog timer that will monitor the communication link in modes requiring camera data— such as indoor autonomous mode— that will force the system into manual mode if communication with the base station is lost

3.1.4 LiDAR Altitude Control

Once we have basic stabilized flight using the camera system, we will:

- Implement a LiDAR system which will allow autonomous control of altitude.
  - This system will involve a single LiDAR pointing downward so that we can measure the distance to the ground.
- The LiDAR distance reading will then be added into the PID control loop as the Z position.

Figure 3: LiDAR Sensor

3.1.5 Support GPS Navigation

The quadcopter currently receives position information for the camera system in 3050, which limits our autonomous capabilities to that specific area. Once we have basic stable flight using the camera system, and LiDAR control for altitude, we plan to:

- Integrate GPS into the system.
- Use the GPS data to add the ability to navigate autonomously outside of the camera system.
  - Using the GPS for the horizontal position and LIDAR for Z position the quad will be able to navigate outside.

3.1.6 Running Linux on second core

The Zynq-7000 SoC, as shown in **Figure 4** below, runs the software on the quadcopter, and it contains two ARM processor cores [7]. Currently, all of the quadcopter software runs on only one of the cores, while the other core is unused. To better support research, we will:

- Configure the unused core to run linux, which will expose higher level functionality to the programmer.
- Create a set of libraries that will allow programs to communicate with the real-time control program running on the other core.
    - These will allow researchers to use higher-level functionality, such as the computer vision library OpenCV, on the quadcopter without much effort.

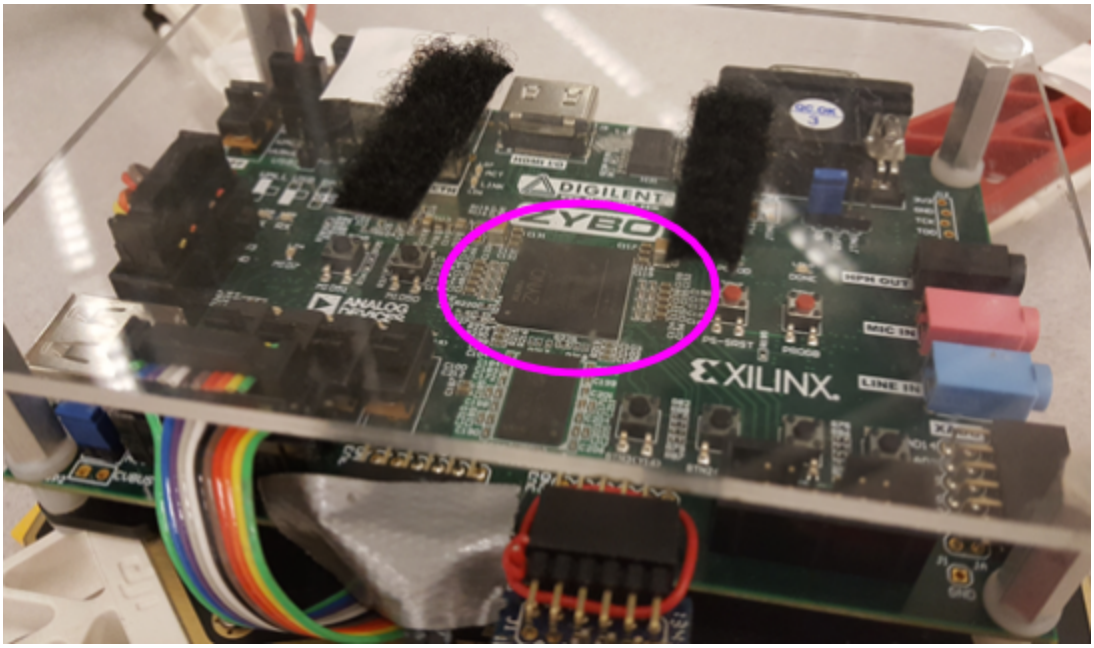

Figure 4: Zynq-7000 SoC on Zybo Board

3.2 Ground Station

3.2.1 Real Time Communication

The current system has a limited amount of communication between the quadcopter and the ground station. We want to have feedback from the quadcopter at all times in order to verify that we are performing the correct functions.

- Constant real time communication between quadcopter and ground station
- Ability to log data at all times from quadcopter, in both manual and autonomous mode

3.2.2 Back End

The back end will be designed to be a modular piece of the ground station. Any front end, whether it be a command line interface or a graphical user interface, will connect to the back end via a socket

connection. This backend will accept input through this socket and forward it through another socket connected to the quadcopter.

- Complete connection to quadcopter through a back end
- Server-Client relationship between back and and front end
- Command pass through from front end to quadcopter

3.2.3 Command Line Interface

A command line interface (CLI) will provide the lowest level of front end ground station control. Users will be able to set and request any relevant variables such as PID constants, pitch, roll and yaw set points.

- Command passing to backend through socket connection
- Multiple instances of CLI at any moment
- Wait for response from backend

3.2.4 Graphical User Interface

The current GUI system has a rather limiting set of options. If connection issues occur between the GUI and the quad, the only thing we know is that the connection was lost. While utilizing our new real time control, we will implement a more descriptive error detection and recognition. Continuing with the real time control, we will have constant updates of VRPN data as well as the current and desired position of the quad. We intend for the GUI to extend the usability of the CLI and provide easier control over high level aspects of the system.

- Visual command passing to backend through socket connection
- Graphical map waypoint selection
- Use the constant communication that the back end has with the quad to provide constant useful updates to the user

3.3 Hardware Improvements

Because this project has been passed down over many semesters with the focus of each team being to add functionality to the quadcopter, the actual system hardware is currently in a non-ideal state. Wire connections are not very secure, some wire connections are being held together with tape, and some plugs are very tight. Along with the rewiring of the quadcopter in general, we would like to design our own PCB which will regulate power on the quadcopter system. The old system utilizes 4 AA batteries which is unnecessary weight. These batteries also could not be fully charged as 4 AA batteries when fully charged is about 6 volts, when the board is powered off of 5 volts.

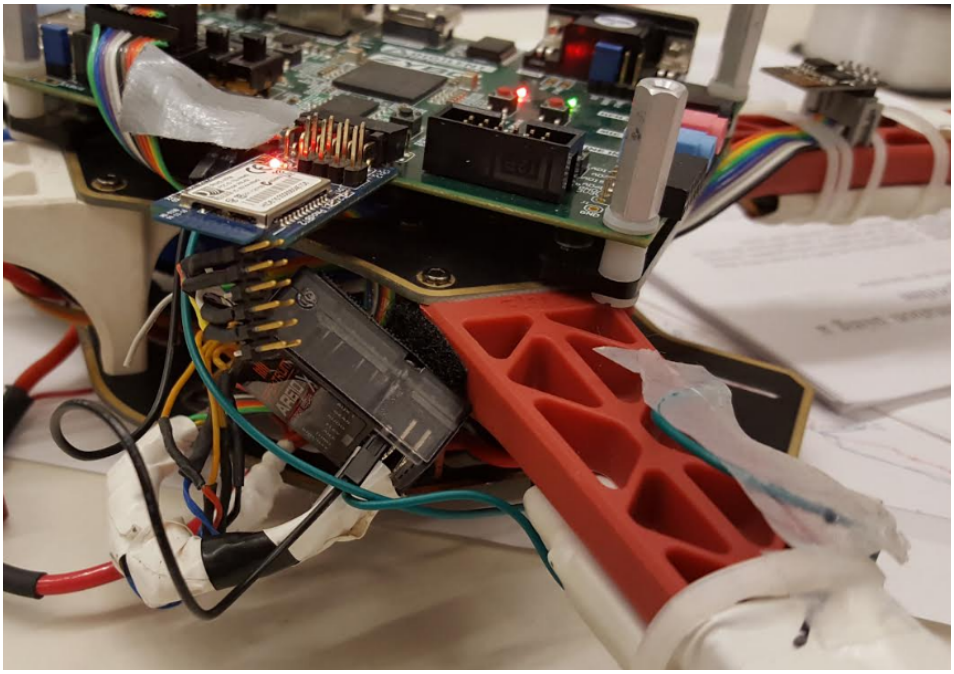

Figure 5: Current Quadcopter Hardware State

Creating our own board will also allow us to add additional quality of life components. We intend to explore improvements to these issues by doing the following:

- Replace all tape-maintained wire connections with secure wire connections
- Replace the plug connecting the battery with ones that are easier to open and close
- Design a PCB to have the following qualities to regulate power on the quadcopter:
    - Regulate the LiPO battery to power the Zybo board:
        - From 11.1 volts to 5 volts
        - At 3 amperes
    - Monitor the current and voltage of the LiPO battery
    - Include a motor cutoff switch
    - Allow for easy access to all of the signal lines

3.4 WiFi Communication

The system is currently configured to use Bluetooth as the communication method between the ground station and the quadcopter. Bluetooth has high latency, which presents difficulties for autonomous flight and running control algorithms on the base station. To improve this, we will:

- Replace Bluetooth with WiFi, using an ESP8266 microcontroller with integrated WiFi.
- Test different configuration parameters to identify the fastest way to send data.
- Write code to implement the communication.

Figure 6: WiFi Module

Our goal is to be able to send data between the ground station and the quadcopter with less than 5ms latency, which would result in 10ms round-trip, a small enough delay to allow the entire control algorithm to execute on the ground station.

3.5 Documentation

In order for subsequent teams to continue the project effectively, critical procedures need to be documented. In terms of developing good tutorials and instructions for future teams, we plan to:

- Write testing procedures for performing measurements for system identification including:
    - Taking moment of inertia measurements about the pitch, roll, and yaw axes
    - Measuring the thrust and drag constants
    - Determining the motor resistance
- Our documentation for test procedures will also include the following:
    - MATLAB scripts used to perform data analysis
    - Pictures of our setup for each test procedure, accompanying a written description and set of instructions for setting up
    - Video tutorials, also will be made available on our website, of our test procedures

3.6 PID Controller Design and Physical Model of the Quadcopter

The quadcopter is stabilized by multiple proportional-integral-derivative (PID) controllers. Historically, the coefficients of the PID controller have been determined through iterative optimization. Since this approach has a couple of downsides, which will be explained in more detail later, our team plans to characterize the quadcopter with a mathematical model and to then derive the PID coefficients from that model. More specifically, our team intends to:

- Create a mathematical model representation of the quadcopter system by following the thesis written by Matt Rich
- Simulate the quadcopter in Simulink from our mathematical model
- Design PID controllers from this mathematical model

4 Design

Below are the methods of approach our team came up with for our project. We plan to heavily reference a graduate thesis [6] written by Matthew Rich, who is currently one of the advising students for MicroCART, when designing the control system for the quadcopter. We have also included some systems-level block diagrams to help demonstrate where our plans lie with respect to the overall structure of the quadcopter system.

4.1 Previous Work and Literature Review

Our project this semester for MicroCART will include the design of a PID controller that is based on a mathematical model of the quadcopter, as derived and developed in Matt Rich's thesis. This thesis was specifically written for the purpose of developing a mathematical model that is simple enough that undergraduate students can easily understand it, but that is also rigorous enough that a robust control system can be designed to allow the quadcopter to maintain its current position and move to a new position. Indeed, one of the core objectives of Matt Rich's thesis was to "work cooperatively with the MicroCART project" [6].

Another quadcopter system also used for research platforms for controls applications is called Ascending Technologies, which is a well-known quadcopter company that is part of Intel. This company creates many different types of UAVs for research purposes. Allows for the user to implement his or her own C code with his or her design controls algorithms. Like our quadcopter, this system also uses two ARM microprocessors. Additionally, with this system there is a strong, robust controller that can be switched on if the experimental controller is making the quadcopter unstable [5].

Figure 7: Ascending Technology Hummingbird Quadcopter

Looking at this example and its capabilities helped us realize what we wanted to see from our quadcopter system, with our goals for modularity and the ability to have a robust controller. Additionally, for a strong research platform, we would be able to take over with our robust controller the same way that this system does.

4.2 System Design

The overall system is composed of four main components including the communication system, control system, actuation, and sensor system as shown below in **Figure 8**.

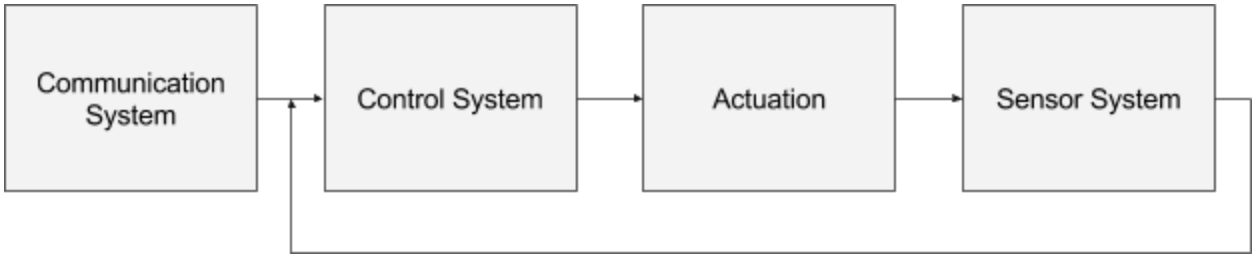

Figure 8: High-Level System Block Diagram

4.2.1 Communication System

The input to the control system from the communication system is dependant upon indoor or outdoor flight. When flying in the distributed autonomous and networked control lab, the x, y, z position of the quadcopter in space is determined from an OptiTrack camera system. This information must then be passed to the quadcopter through the ground station. However, during outside flight, the x, y position will be determined from an onboard GPS module, and the z position will be determined with a LiDAR solution. This GPS module and LiDAR solution will be a part of the sensor subsystem described in Section 4.2.4. Alongside this, the communication system will also provide any user input from the command line interface (CLI), or controller during manual flight. This overall communication process is represented below in **Figure 9**.

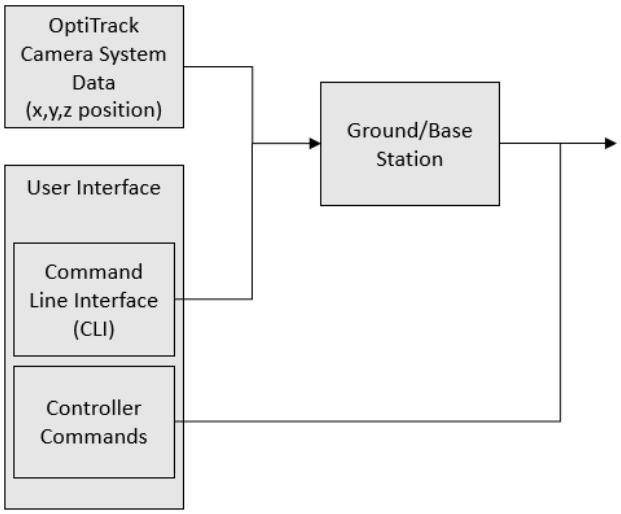

Figure 9: Communication System Block Diagram

4.2.2 Control System

The quadcopter is stabilized by multiple proportional-integral-derivative (PID) controllers. Historically, the coefficients of the PID controller have been determined through iterative optimization. However, this approach has a couple of downsides. Since an iterative approach is technically a guess-and-check method, it has little guarantee or assurance of yielding the most optimal system. It also only yields

appropriate parameters for the system on which the experimentation was performed. An alternative and preferable approach is to characterize the quadcopter with a mathematical model and to then derive the PID coefficients from that model. This approach has the advantage of having theoretical assurances of yielding an optimal controller design and also being general enough to apply to different quadcopters that fit the same generic model [6].

As described in Section 4.2.1 and Section 4.2.4, the control system inputs come from both the communication system and sensor system. The entire control system is composed of multiple nested PID controllers. The composition of a PID controller in a feedback loop is shown below in **Figure 10**, where r(t) is the set point value, and y(t) is the measured output. The primary components of the PID controller include the K_p , K_i , and K_d terms, which denote the coefficients for proportional, integral, and derivative terms, respectively [3]. These coefficients will be determined using a mathematical model of the quadcopter as described in the introduction of Section 4 and in Section 3.6.

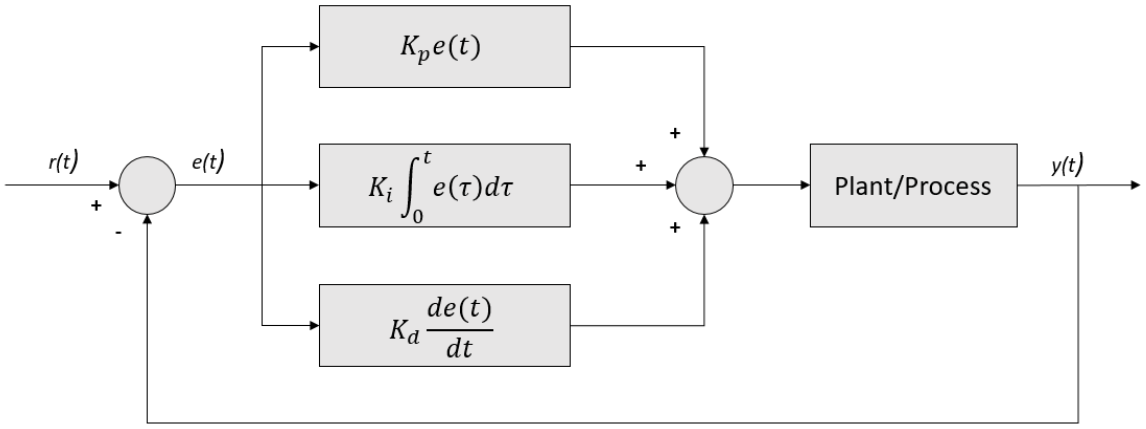

Figure 10: PID Controller Block Diagram

The overall control system is broken down into four major subsections including the height controller (z-axis), longitudinal controller (y-axis), lateral controller (x-axis), and a dedicated yaw controller [6]. Most of these controllers have nested controllers associated with them, as shown below in **Figure 11**.

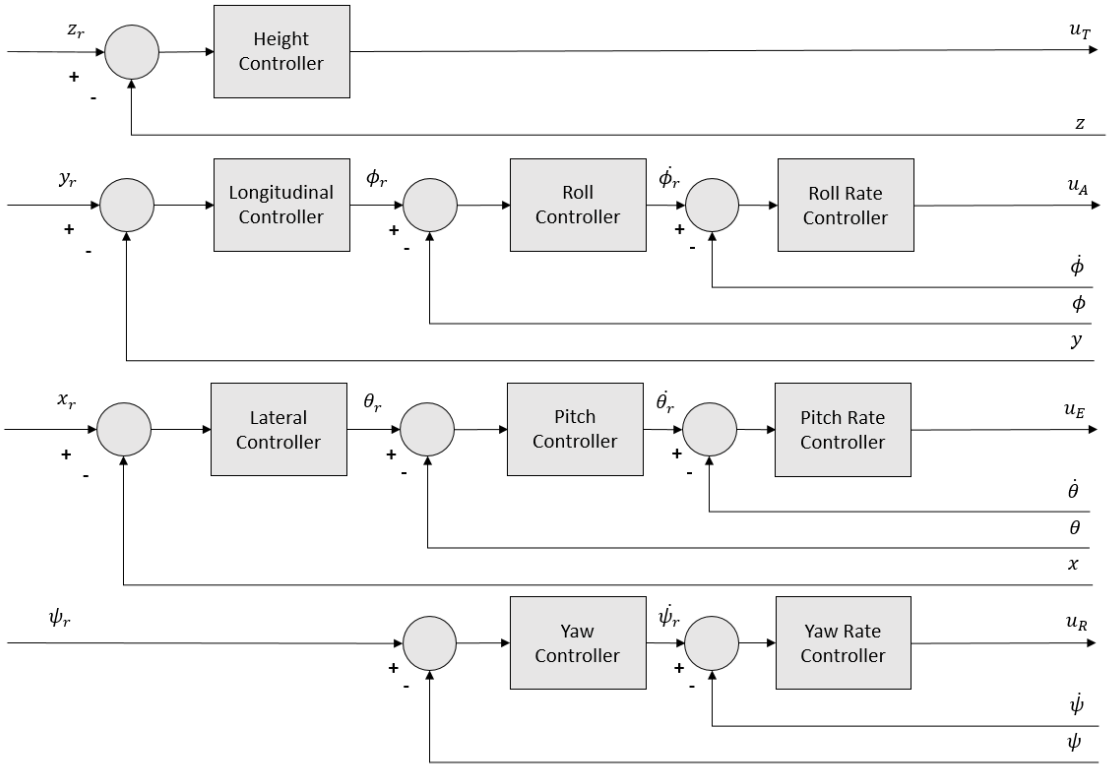

Figure 11: Nested Loop PID Architecture

This architecture allows us to not only control the body frame position of the quadcopter, but its velocity as well. When describing things in terms of body frame we are referring to the frame of reference of the body of the quadcopter, which assumes that the origin of this axis lies at the center of mass of the body [6]. The variables utilized in Figure 11 are defined in **Table 2** below. Note that in the above image variables denoted with the subscript "r" represent setpoint values.

Variable	Definition
x	Body frame x position
y	Body frame y position
z	Body frame z position
ϕ	Body frame roll angle
θ	Body frame pitch angle
ψ	Body frame yaw angle
$\dot{\phi}$	Body frame roll angular velocity
$\dot{\theta}$	Body frame pitch angular velocity

$\dot{\psi}$	Body frame yaw angular velocity
u_T	Throttle command
u_A	Aileron command
u_E	Elevator command
u_R	Rudder command

Table 2: PID Architecture Variable Definitions

The last portion of the control system is converting the actual output commands of the controllers to equivalent input commands for each of the four individual electronic speed controllers (ESCs). To do this we utilize a signal mixer [Matt's thesis], defined by the following matrix:

$$M_g = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

This addition is shown in **Figure 12** below:

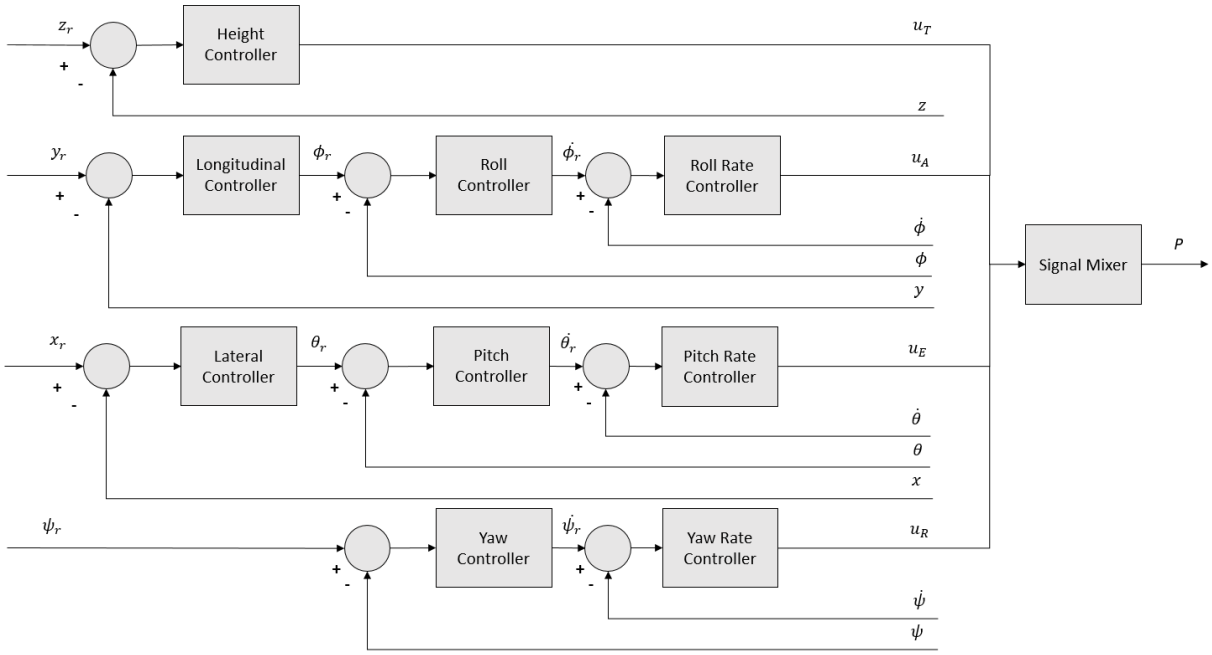

Figure 12: Nested Loop PID Architecture with Signal Mixer

Note that "P" in the above system diagram represents the vector of all four ESC duty cycle percentages.

4.2.3 Actuation

The actual actuation, or mechanical movement of the quadcopter occurs through the driving of each motor/rotor combination [6]. As stated previously the output from the control system provides ESC duty cycle percentages, these percentages are used by the ESC to coordinate what voltage to apply to each motor, represented as the vector V in **Figure 13**. The ESCs themselves are powered from a 11.1V LiPO battery (nominal voltage). From there the we can determine the actual angular velocity and acceleration defined below:

$$M = \begin{bmatrix} \omega \\ \alpha \end{bmatrix}$$

where ω and α represent the angular velocity and acceleration respectively [Matt's thesis]. From this we are able to derive the overall block diagram for the actuation of the quadcopter.

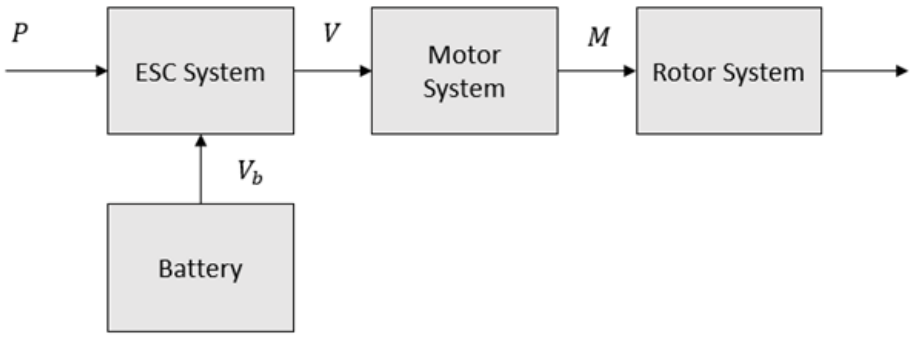

Figure 13: Actuation System Block Diagram

4.2.4 Sensor System

The sensor system is composed of the MPU-9150 IMU (Inertial Measurement Unit) which provides gyroscope and accelerometer data and is also an input to the control system. Alongside this, during outside flight the OptiTrack camera system will be replaced with a dedicated GPS module and LiDAR solution for determining the x, y, z position of the quadcopter. Data from the gyroscope and accelerometer can be used to find the yaw, pitch, roll angles, and angular velocities. With this and either the OptiTrack camera system or GPS module and LIDAR solution, we are able to provide all the required inputs to the control system.

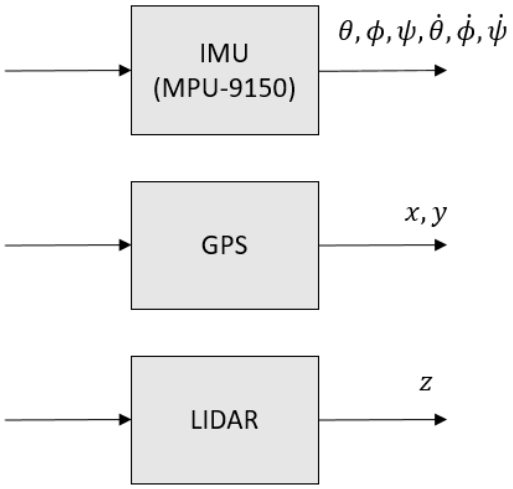

Figure 14: Sensor System Block Diagram

We have recently been looking into representing these systems in our Simulink model of our quadcopter in the best possible way. Currently our software code that takes the readings from the IMU to determine the pitch and roll angle of the position of the quadcopter is based on the a few equations. And though this makes intuitive sense, since we were able to rederive these equations using basic trigonometry, our graduate student advisors, Matt and Ian, have been explaining to us that some researchers may have developed a better way to represent these angles of the quadcopter [2]. We will look into this to see if this is indeed a better method that will be worth switching to depending on how much more accurate it is as well as how much more effort it requires and complication it adds to the project.

4.3 Quadcopter Software

4.3.1 Autonomous Flight

The stationary mode of the autonomous flight already has PID controllers configured. These just need to be tuned correctly. The waypoint mode can be designed in a few possible ways, which we will investigate. The most promising method is to incrementally adjust the <X,Y,Z> setpoint after each iteration of the control loop. This will allow the controller to continually move towards the waypoint without accumulating large errors, which would result in the quadcopter wildly overshooting the waypoint.

4.3.2 Improved Communication System

In order to decrease the communication latency between the ground station and the quadcopter, we will replace the bluetooth module with an Espressif ESP8266 microcontroller with integrated WiFi. To allow for guaranteed and non-guaranteed delivery types, we will use the built-in TCP and UDP, respectively, or build our own message verification built around UDP.

4.3.3 Support Flying without Ground Station

To liberate the quadcopter from its dependency on the ground station, we will need to refactor the code to allow the quadcopter program loop to continue without waiting for the ground station. We will first require that the quadcopter have this independence only when flying in manual mode, since autonomous flight currently requires camera data for stabilization, which is currently given to the quadcopter from the base station. Once the quadcopter can fly without the camera system, we will add autonomous flight back in when not connected to ground station.

4.3.4 LiDAR Altitude Control

LiDAR will be used as an altitude sensor when the tracking data from the infrared camera system is not available. The altitude provided by GPS is relative to sea level, which makes it impossible to reliably land in autonomous mode. To get accurate altitude measurements, we will add a LiDAR-Lite v3 from Sparkfun to the quadcopter. When the quadcopter is flying outside of the camera system, it will use the LiDAR sensor for z-axis localization. When the quadcopter reaches altitudes out of range for the LiDAR sensor, it will fall back on GPS altitude, which is sufficiently accurate at those heights.

4.3.5 Support GPS Navigation

To support GPS navigation, we will have to modify the quadcopter software to interpret locations relative to the starting location. The current software receives absolute coordinates from the infrared tracking system. To use relative locations, we will modify the software to obtain its location on startup, and transform any new coordinates to the relative coordinate system. For GPS, we convert coordinates to the local system, and then convert them to meters. We may also need to add a layer of sensor fusion to combine the GPS and accelerometer data to provide more accurate location data at a higher frequency than can be obtained from solely GPS.

4.3.6 Running Linux on second core

Running linux on the a single core of the Zync-7000 SoC is fully supported by the Xilinx toolchain, so actually loading the operating system won't require too much of our own design. The most complicated part of this deliverable is creating the libraries that will simplify communication between the linux core and the real-time control core.

At the lowest level, the IPC (inter-processor communication) system will memory map a shared region of memory on both processors that will be used for message passing. Each processor core will also have an interrupt that can be raised by the other processor. After this system is initialized, each processor can send a message to the other by encoding a message in the shared memory region and raising an interrupt on the other processor. The other processor will then jump to the interrupt handler, which will fetch the message from the shared memory region and add it to the message queue. The main software running on each core will be responsible for checking the queue and handling any messages that are received.

The low-level message passing interface will be somewhat tedious to work with directly, so we will be creating a higher-level API that will be used to set and get parameters from the other core. For example, an API function called int IPC_getLocation(Location_t*) would be used to request the current location of

the quadcopter from the real-time core using the lower-level message passing system. This higher-level API will be the one that researchers in control systems could use to design high level algorithms that run on the linux core of the processor.

4.3 WiFi Communication

The WiFi communication will be implemented with the ESP8266 microcontroller with integrated WiFi. The WiFi module will act as an access point, so the ground station can connect to the WiFi network hosted on the module without any extra configuration. To program the ESP8266, we will use features provided by the Arduino library for the ESP8266, as well as functionality provided by the Espressif SDK. The module will support both TCP and UDP for communication. To simplify the integration of WiFi, it will forward data from UART to WiFI, and forward the data received over WiFi to UART. This way, no changes will be required in the quadcopter software to use WiFi as the communication method.

4.4 Ground Station

4.4.1 Backend

The backend system will run on its own and handle the VRPN connection as well as the connection to the quad. It will expose its service via a socket to the front-end. Various front-ends (command line, control loops, GUI) can close the loop via that socket, without having to worry about details such as WiFi/Bluetooth, or VRPN vs GPS. The backend will handle parsing of quad data, as well as transmitting commands to the quad.

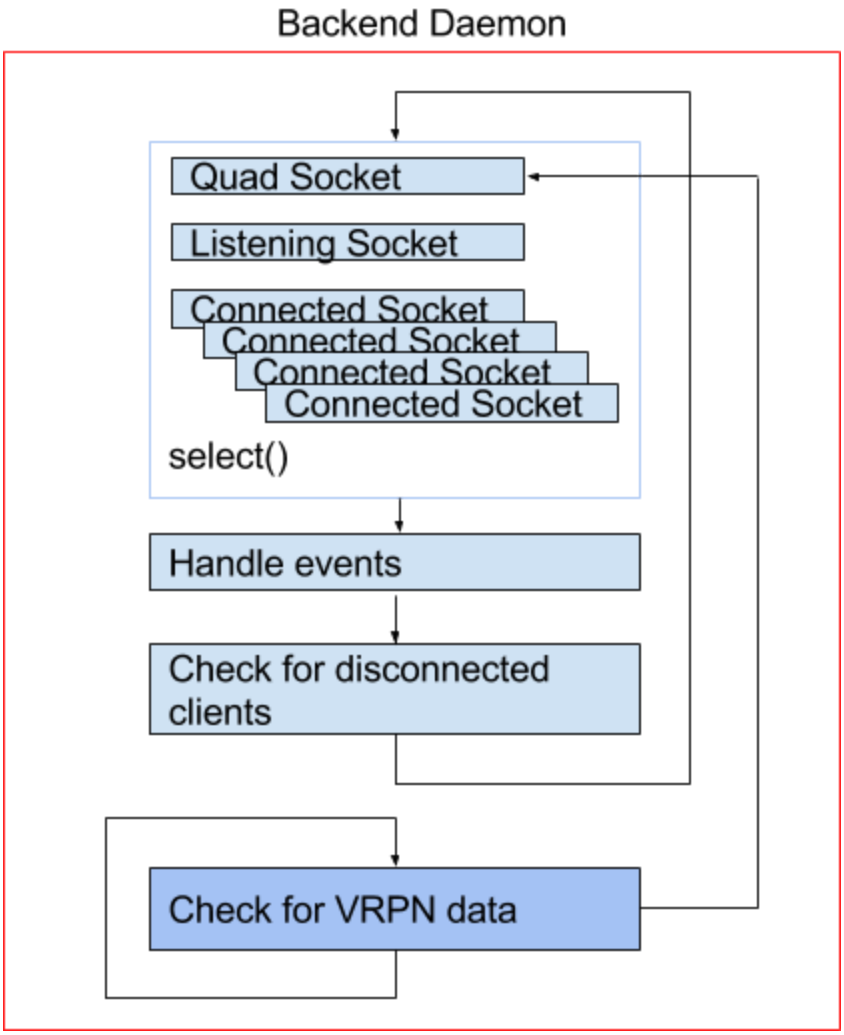

Figure 15: Backend Daemon

The backend daemon runs and manages connections with the quad. Communication with the quad uses a low-overhead binary protocol. It also connects to the tracking system and forwards tracking information to the quad. Frontends, CLI or GUI, connect to the backend and issue commands in an easy-to-parse ASCII protocol. These commands manipulate the state of the quadcopter.

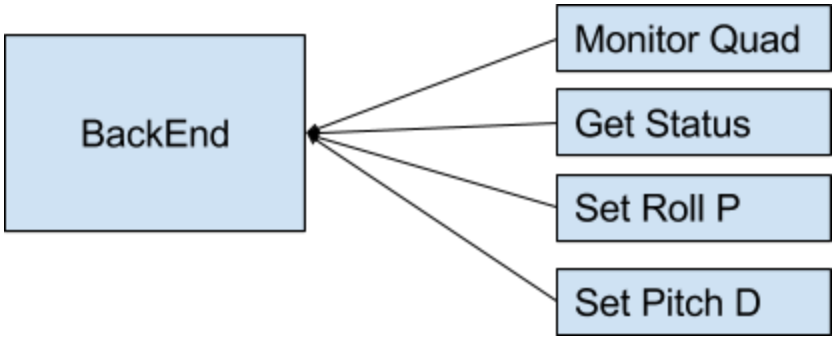

Figure 16: Command Line Interface Block Diagram

The design of the CLI is to create a Client-Server relationship with the BackEnd. The CLI will be split up into separate programs. These programs will either create a connection and perform the task, or create a connection and hold it until the user is done with the task. This will allow tab completion in a shell, bash scripting to allow test scripts as well as a history of commands with BASH_HISTORY.

4.4.2 Command Line Interface

The command line interface (CLI) will connect to the backend via a socket. The CLI will be listening for user input in the form of commands. These commands will be parsed and the correctly formatted packets will be sent to the backend. From there the backend will send to the quad. The CLI will be listening for responses from the quad at the same time, providing the data from those responses to the user.

4.4.3 GUI

The GUI will connect to the backend over the same socket interface as the CLI. The GUI will be able to support autonomy to various degrees, selectable on-the-fly. This will include full waypoint-autonomous mode, full manual control, and semi-autonomous mode. In semi-autonomous mode, one or more coordinates (X,Y,Z) or a subset of attitude DoF (likely just yaw) will be stabilized by a closed-loop control system, while the rest of the quad will fly with manual control. The GUI will also support much of the CLI functionality, such as downloading logs from the quadcopter.

5 Test Plan

Here we will explain how to evaluate the success of our methods for approaching the project statement. At the beginning of this section, we will assess our current proposed solutions, thus explaining why we believe the solutions planned out in section 2 and described in section 3 will be better than other possible solutions. Additionally, this section will also include a functional test plan for checking that our approach was successful and meets our core objectives.

5.1 Assessment of Proposed Solution

5.1.1 Current Quadcopter Software

The autonomous mode of the quadcopter is very unstable and limited, so our plan to improve upon this system will involve improving many facets of the current software. Currently, the quadcopter must be manually piloted into a stable flight state, then transitioned into autonomous mode. Once in autonomous mode, the quadcopter will start oscillating about its target position until the pilot takes control or it crashes. We will start by properly tuning the PID controllers that control the autonomous system. We will also improve the current system by incorporating a system for totally autonomous takeoff, flight, and landing.

5.1.2 Ground Station

The current ground station has many limitations that we plan on relieving. We will increase communication with the quad so that we always know the state of the quad and so that we can better manage the quad. The CLI and GUI will use a more streamlined command structure than is currently being used. These commands will send binary data instead of ascii values, which will reduce the amount of information needed to be sent back and forth. A GUI with much more information and input options will be created to replace the old GUI. We will create the ability to determine error flags from the quad and have constant numerical and visual representations of the quad's current state. In addition, the new GUI will provide input methods such as pointing and clicking on a map to make the quad go where you want it to. This will enable a far wider set of flight options than the current GUI provides.

5.1.3 Hardware Improvements

The wiring between the sensors, actuators, and control board are very disorganized, and the control system was powered by a set of four AA batteries that were not regulated. Our hardware improvements will primarily focus on solving those problems, plus adding extra functionality that could be useful for future use of the quadcopter. We will use a custom circuit board that will have a buck regulator and connectors for routing wiring. We will also add monitoring of voltage and current from the battery, as well as a high-side solid state switch for controlling power to the motors from software.

5.1.4 WiFi Communication

The system is currently configured to use Bluetooth as the communication method between the ground station and the quadcopter. Bluetooth has high latency, around 25ms one-way, which presents difficulties for autonomous flight and running control algorithms on the base station. WiFi can send 32KB of data from the ground station to the quadcopter in less than 2ms, which is much faster than Bluetooth.

5.1.5 Documentation

To allow for future MicroCART teams to be able to continue the progress of this ongoing project in an efficient manner, we believe some additional documents need to be created that will help guide future teams through some of the testing procedures and physical concepts we had to learn through a lot of trial and error and with the help of other graduate students present in the lab. Although it is beneficial to struggle through some of these issues and gain experience working through problems during the design and testing process, we would rather students focus on newer, harder challenges that come with enhancing the current system, not tackling the same issues that have already been solved multiple times before.

5.2 Validation and Acceptance Test

Most functional requirements will be verified through experimentation in Coover 3050, especially in preliminary testing, when our system will be using the infrared tracking system available in the lab. In the future, when we begin using GPS for navigation of the quadcopter, we will gradually move out of the lab environment for testing purposes.

5.2.1 Quadcopter Software

When making updates to the quad software, flying the quad will reveal the most information concerning adherence to our functional requirements, so we will eventually verify all changes with flight tests. Where plausible, we will first verify the functionality of the software by observing simple outputs available on the Zybo board. We plan to test all of the desired functions for the quadcopter, namely verifying the different autonomous modes including: basic autonomous stabilization, autonomous point to point navigation, obstacle avoidance, and GPS navigation.

5.2.2 Ground Station

Preliminary testing will involve downloading bulk logged data from the quad using CLI software with the base station. Further testing will involve testing semi- and full-autonomous modes. Testing semi-autonomous modes will test the ability of the ground station to forward commands to the quad. A thorough test of full-autonomous mode will confirm the ability of the ground station to continually update the control parameters sent to the quad. Eventually, coordinate-based control loops will be moved to the ground station. When this is done, testing waypoint-based fully-autonomous will test all ground station capabilities.

5.2.3 Hardware Improvements

After making some improvements to the hardware, before flying the quadcopter, we will ensure by simple inspection that all of the wire connections appear to be secure, that they will not interfere with the rotation of the blades, and that they will not move too much during flight. In addition, each time that we disconnect especially critical wire connections, we will test that the connection was properly re-made using the continuity test feature of the multimeter. In addition, before too many wires connections are updated, we will occasionally perform a simple flight test to make sure that all of the connections in the system are exactly the same.

5.2.4 WiFi Communication

The WiFi module should be able to send a 32-byte message from the ground station to the Zybo board and back to the ground station in less than 10ms. This will be tested by modifying the quadcopter software to echo the location data sent by the ground station, and calculating the time difference between sending and receiving the data on the ground station. The communication should be reliable at the sub-10-millisecond latency up to 100 feet without obstructions. This will be tested by pinging the module outdoors, with a gradually increasing distance between the base station and module. We will also test the range with the module at various orientations to ensure that the connection is not dropped when flying. Finally, we will test that the module and ground station will automatically reconnect if the connection is lost.

5.2.5 Documentation

We want ensure that our documentation that we write is readily usable by anyone who is not familiar with the quadcopter system and the sub-system for which that the document is written. One approach to testing that the documentation is clear and easy to follow is to have a member from the team who is unfamiliar with a particular sub-system to read the procedure and documentation written to explain that sub-system. For example, to verify the clarity of the data collection and analysis procedure for measuring and determining the thrust constant of the quadcopter, which is written by the controls sub-team, after completing documentation on this process, we will have a member from one of the other sub-teams, such as the quadcopter software sub-team, who is unfamiliar with the physics of the quadcopter and its utilization for the testing procedure.

5.2.6 PID Controller Design and Physical Model of the Quadcopter

The controls sub-team must also become very familiar with the quadcopter software, and must also be able to describe the high-level structure of the code as well as be able to vividly describe how data is brought in and processed from sensors on the Zybo board, which represent the angular position and velocity of the quadcopter, as well as its linear position and velocity. In addition to having this level of conceptual understanding of the quadcopter software and being able to explain the data processing steps being performed in the software, the team's PID controller design must also be able to maintain the position of the quadcopter and *it must be stable*, rather than experiencing uncontrolled oscillations that gradually get larger and eventually cause the quadcopter to yank its tether and fall to the ground.

6 Design Requirements and Specifications

6.1 Functional Requirements

Below is a brief explanation of the functional requirements of our quadcopter system, which includes all of the technical requirements describing the behavior we want the quadcopter to have in its final version. The enumerated functional requirements in this section will thus be gradually incorporated into our project over the course of these next two semesters.

6.1.1 Quadcopter Software Functional Requirements

- *The quadcopter will be able to receive waypoints from the ground station, and autonomously navigate to them, then remain there until further waypoints are given.*
- *The quadcopter shall be able to fly in manual mode without a connection to the ground station.*
- *The quadcopter will be able to send messages to the ground station with two methods:*
    - *Guaranteed delivery method, to ensure important data will be received*
    - *Non-guaranteed method*
- *The quadcopter shall be able to connect to the Sparkfun LiDAR Lite module, take measurements from it, and use those measurements as the altitude information for autonomous flight.*
- *The quadcopter shall be able to connect to a GPS module, take measurements from it, and use the GPS coordinates as information about the horizontal location of the quadcopter.*
- *The linux operating system shall be capable of communicating with the real-time processor with less than a 1 millisecond round-trip latency*

6.1.2 Ground Station Functional Requirements

- *The ground station will be able to connect to the quadcopter using WiFi, with less than 10 milliseconds round-trip latency, to enable safe and reliable autonomous flight*
- *The ground station will be able to send waypoints to the quadcopter in the form of relative coordinates*
- *The ground station will create event logs and save information about the quadcopter continuously, when the quadcopter is both in both autonomous and manual mode, at a rate of at least every 20 milliseconds*
- *The ground station will be able to send information to the quadcopter with two methods: A guaranteed delivery method, and a non-guaranteed method*

6.1.3 Voltage Regulator Requirements

- *The voltage regulator will output 5v +/- 2%*
- *The voltage regulator will provide up to 3A continuous and stay within voltage spec*
- *The current monitor will measure up to 30A at 12-bit accuracy*
- *The voltage monitor will measure up to 13v at 12-bit accuracy*
- *The solid state power switch will be capable of handling up to 30A continuous current*

6.1.4 Wifi Module Functional Requirements

- *The ground station will be able to send 32 bytes to the quadcopter in less than 5ms, and the quadcopter will be able to send 32 bytes to the ground station in less than 5ms.*

- *The quadcopter and ground station will be able to communicate for up to at least 100 ft without obstruction.*
- *The WiFi module will be able to send data over a protocol that guarantees delivery.*

6.1.5 Control System Functional Requirements

- *The quadcopter, while in autonomous mode and while in the Coover 3050 lab, shall be able to hover and remain at a fixed point, deviating less than 1 foot at maximum.*
- *The quadcopter, while in autonomous mode and while in the Coover 3050 lab, shall be able to travel from a point in space to another point in space autonomously in 7 seconds or less, when given a position input from the user in the GUI interface.*
- *The quadcopter, while in autonomous mode and while outdoors with adequate GPS reception, shall be able to hover and remain at a fixed point, deviating less than 4 feet at maximum.*

6.2 Non-functional Requirements

In this section, we list the non-functional requirements we have determ of the project, including requirements that our quadcopter project should have, that are not necessarily direct behaviors we want to implement in the system, but attributes we want our system to have. These non-functional requirements should include more detail about how we want to do our design work.

6.2.1 Current Quadcopter Software Non-functional Requirements

- *The autonomous flight mode should be stable and without oscillations increasing in magnitude*
- *Sending messages over the guaranteed and non-guaranteed methods should not require excessive burden on the programmer.*
- *The quadcopter should be able to fly without requiring a ground station or camera tracking system to be initialized.*
- *LiDAR should provide accurate enough altitude to enable autonomous takeoff and landing for future control systems.*
- *Communicating between Linux and the dedicated control algorithm should not require extensive work to perform the inter-processor communication.*

6.2.2 Ground Station Non-Functional Requirements

- *Written largely in C*
- *Maximize code reuse between CLI and GUI programs*
- *Leave readable code for future teams*
- *Design to allow modular replacement of components*
- *Code should be written to good, safe standards.*

6.2.3 Voltage Regulator Non-Functional Requirements

- *The custom PCB shall be small enough to fit in the spare space on the quadcopter frame*
- *All ESC signals shall be broken out to allow custom signal injection*
- *The WiFi module shall be placed on the board in an orientation that does not affect signal strength*
- *All wiring shall be neatly routed to and from the PCB*

6.2.4 Wifi Module Non-Functional Requirements

- *The WiFi module should be lightweight and low power*
- *The communication should be fast enough to enable running the control algorithm on the ground station*

6.2.5 Control System Non-functional Requirements

- *The control system shall be implemented with a PID controller*
- *The quadcopter shall be controlled with 9 PID controllers, consisting of the following:*
    - *3 PID controllers controlling the angular position of the quadcopter*
    - *3 PID controllers controlling the angular velocity of the quadcopter*
    - *3 PID controllers controlling the linear position of the quadcopter*
- *Each PID controller shall be implemented as a PID_controller object within the quadcopter software, having the following fields:*
    - *Ki constant, the integral constant of that particular PID controller object*
    - *Kp constant, the proportional constant of that particular PID controller object*
    - *Kd constant, the derivative constant of that particular PID controller object*

7 Challenges

Include any concerns or details that may slow or hinder your plan as it is now. These may include anything to do with costs, materials, equipment, knowledge of area, accuracy issues, etc. This should also include risks, an assessment of the feasibility of this project, as well as considerations of cost with a realistic estimate of project costs.

7.1 Risks

Some possible risks that our team may incur, which would slow our progress while working on this project, include possibly damaging the hardware or other equipment on the quadcopter. This would increase the amount of money spent, beyond what we specify in our estimated project costs in the upcoming section, and it would additionally pause our progress until we can purchase and implement replacement parts.

Another important risk that our team must try to prevent from occurring is having a lack of communication between sub-teams and with the advisers. Not having good communication poses a problem because when working on a project it is important to communicate when major obstacles come in the way, instead of quietly trying to resolve the issue personally, for it is very useful to discuss potential solutions and come up, as a team, with the best course of action to follow. In addition, when changes are made to the current system without good communication with other team members, whether they are small or large, this can cause great inefficiencies in development and may require an annoying, tedious process of undoing large amounts of work done to find the original issue between what different sub-teams were expecting in their design and rework what was done so that all teams have a consistent understanding of the system and have that understanding reflected into their own contribution of the current design.

7.2 Feasibility Assessment

Physical modeling of the quad requires an accurate way of determining moments of inertia. Historically, this task has been completed with experiments involving the ECP. However, after thorough experimentation, we have determined that the ECP gives inconsistent data, so we will need to develop a new method of accurately measuring the moment of inertia of the quadcopter. Fundamental obstacles like these will delay our progress, since we need certain functionality to be in place before we can move on to the next tasks in order.

So, if and when we run into these obstacles, it will cause us to be behind on our scheduled goals according to our timeline, which is in the next section, Section 8. And indeed, our timeline is quite tight, so it will be difficult to keep up with our goals once we get behind. This may prevent us from being able to accomplish some of our final goals at the end of next semester, like doing advanced flight maneuvers or incorporating a camera sensor onto the quadcopter for outdoor flight.

7.3 Cost Considerations

The motors used by the quadcopter are no longer manufactured or supported, so we will likely need to invest in a new set of motors, as well as backup motors, in case one of the current motors is broken. In addition, for some of the hardware improvements we had in mind, we will need supplies to create the power distributor and voltage regulator circuit, which will be on a PCB. These items have been approximated for cost[4] and included below, in **Table 1**:

Item	Associated Cost
6 new motors	$200
Supplies for power management circuit	$100
Total Cost	$300

Table 3: Cost Considerations

8 Timeline

To have a successful attempt at our senior design project, we will need to come up with an approximate schedule for when we want to have our tasks completed over the course of the next two semesters. Therefore, we followed the advice of our senior design professor about “working backwards”. So we discussed together as a group our goals for this semester, what our needs are for tasks we must have accomplished before the start of next semester, and thus, what we need to accomplish over the course of this semester to get to an ideal state for the overall project so that we have time to accomplish everything that we would like in Spring.

After deciding these “milestone” points, we went through the individual tasks we wanted to complete for each milestone, thought of about how long that task would take us to accomplish, and came up with a general sense of by what time we wanted to accomplish it. Thus, using this method of scheduling out our tasks over the next academic year, we came up with the timelines presented in the next two subsections. One representing, the time schedule for our tasks over the rest of this semester, Fall of 2016, in section 8.1, and the other representing our scheduled tasks over the course of the following semester, Spring of 2017.

8.1 First Semester

For the first semester, our team must first become very familiar with the current structure of the quadcopter software structure and code, as well as the hardware setup of the system, thus gaining a more textured understanding of how the system works on a high-level and at much lower levels as well. In addition to being a first step toward our senior design project, this is also somewhat of a continuous, ongoing process, for as we work with different parts of the system more closely, we should try to become even more familiar with that particular sub-system and gain a very good understanding of how it works and relates to the other systems within the quadcopter.

To achieve these goals, our team will spend time reading documentation created by last year’s team, including their final report, their systems level diagrams of the system overall, and the procedures they wrote on how they took measurements of the quadcopter. In addition, to gain a more intuitive sense of how the quadcopter system flies, responds to different manual inputs, and controls its position in space, we will all learn how to fly the quadcopter adequately well.

We will also need to work more closely with the MicroCART quadcopter and replicate what previous semesters have done to implement the current abilities and functionality of the quadcopter, thus getting us more familiar with not just what the previous team has done to program the system to maintain its current position with some manual inputs, but also how their programs work in detail, for the quadcopter to remain steady.

In addition, we will need to start taking measurements and characterizing the quadcopter. This includes measuring the moment of inertia of the quadcopter about the roll, yaw, and pitch axes, as well as measuring the thrust constant and drag constant. There are additionally many other measurements we will have to take, all of which are listed and in the graduate thesis.

Taking measurements is important in order to build on a mathematical model of our system and perform system identification to design an optimized control system by systematically deriving the PID constants for controlling the motion of the quadcopter about different axes of rotation and with respect to its

linear position axes, defined from its own point of reference. See our Fall semester timeline below in **Figure 16** starting from the beginning of October.

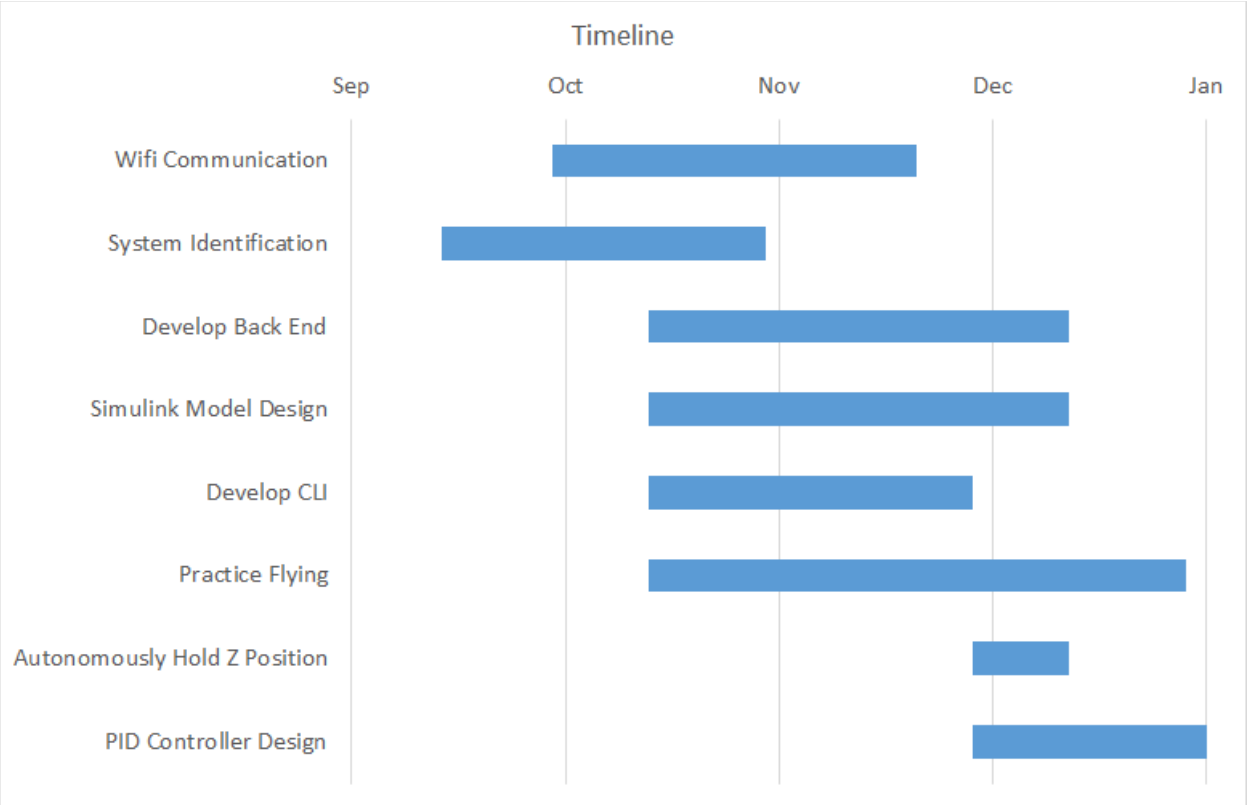

Figure 17: Fall Semester Timeline

8.2 Second Semester

For our second semester, we must continue our progress we made in the first semester, as needed, if we get behind on our first semester schedule. Additionally, we need to go beyond what we accomplished our first semester. This will include us building on the foundation we laid out during our first semester, which is characterizing the quadcopter, developing the 9 optimal PID controllers, and fine-tuning these constants so that we can enable the quadcopter to hover and hold its position in space autonomously, as well as to fly autonomously to different waypoints which we specify in space, using the GUI interface designed for this very purpose.

On top of this foundation of the capabilities of the quadcopter, and assuming we have completed everything on time, according to our timeline laid out for the rest of this semester, we can do many interesting things our second semester, and possibly make progress on our ideal goals listed at the end of the semester if we have time. Our goals for the second semester include being able to fly the quadcopter outdoors autonomously, using a mixture of GPS data and accelerometer data so that the quadcopter can determine a best approximation of its position in space, and fly quite accurately, while overcoming the 1 meter range of error that comes with GPS. If we can successfully get to this point, then we can start looking into some of the other functionalities we wanted to implement, as well, including performing

advanced flying maneuvers, such as rotating while moving in a line in a horizontal plane. Our scheduled time line for the second semester is recorded below in **Figure 17**:

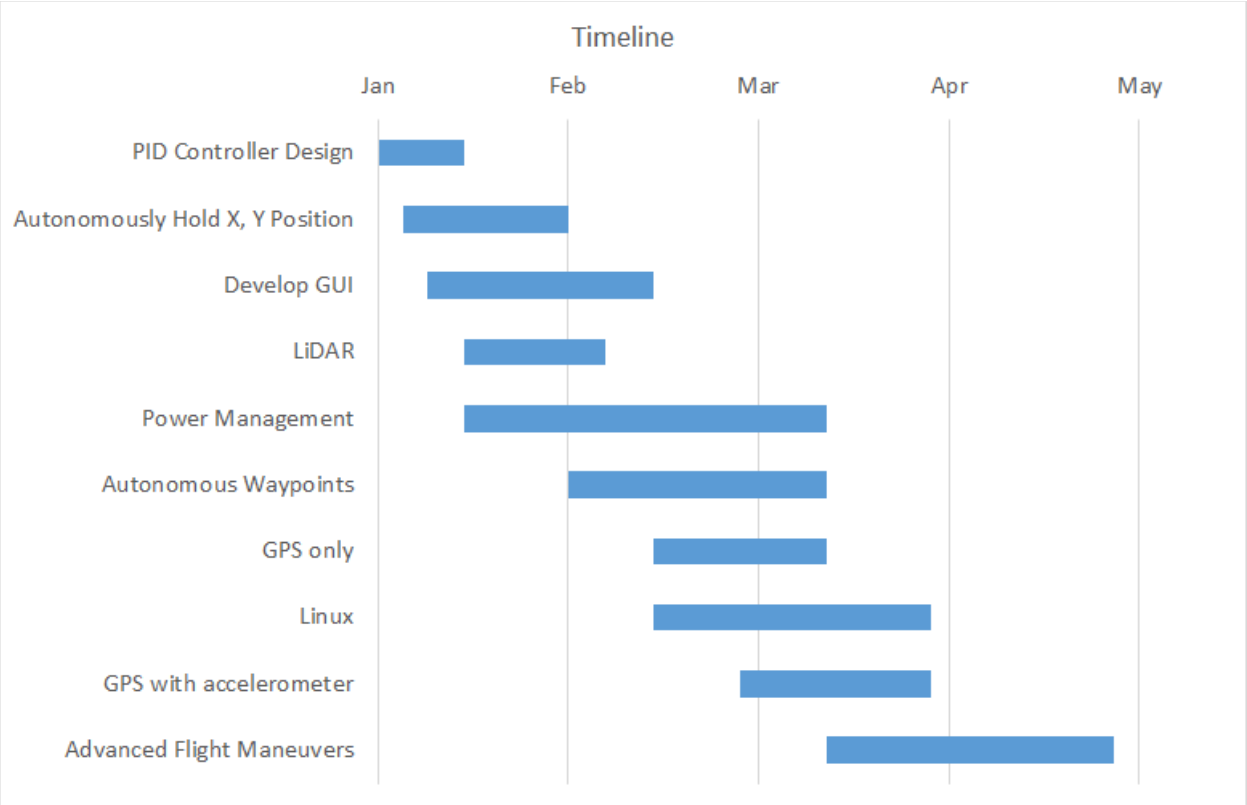

Figure 18: Spring Semester Timeline

9 Conclusions

This year (the 2016 to 2017 academic year), MicroCART will improve the current system by designing PID controllers to manage the movement of the quadcopter in each axis of rotation and linear motion. Eventually, starting next semester, we will begin the process of enabling the quadcopter to fly outdoors using GPS navigation, along with a LIDAR sensor attached underneath to detect its height above ground or obstacles located beneath it, which will be important for achieving successful landings. Finally, after accomplishing these main goals listed, we would potentially like to enable the quadcopter to perform advanced flying maneuvers, as well.

10 References

[1] *Bluetooth vs Wi-Fi. (n.d.). Retrieved November 19, 2016*, from http://www.diffen.com/difference/Bluetooth_vs_Wifi

[2] Cavallo, A., A. Cirillo, P. Cirillo, G. De Maria, P. Falco, C. Natale, and S. Pirozzi. *Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation*. Thesis. Second University of Nepales, 2014. Aversa: ScienceDirect, 2016. Print.

[3] Ogata, Katsuhiko. *Modern Control Engineering*. 5th ed. Englewood Cliffs, NJ: Prentice-Hall, 1970. Print.

[4] "Products." *DJI Store*. DJI, 2016. Web. 12 Oct. 2016. <http://store.dji.com/>.

[5] "Research UAV – Drones / UAS for Research & Development." *Ascending Technologies*. N.p., 5 Nov. 2016. Web. 04 Dec. 2016. <http://www.asctec.de/en/asctec-research-uav/>.

[6] Rich, Matthew. *Model Development, System Identification, and Control of a Quadcopter Helicopter*. Thesis. Iowa State University, 2012. Ames: Graduate Theses and Dissertations, 2012. Web.

[7] *Zynq-7000 All Programmable SoC Overview*. DS190 (v1.10). Xilinx. September 27, 2016