

# MicroCART 2017-2018

## Design Document

Team # 17

### Clients / Advisors:

Dr. Phillip Jones

Dr. Nicola Elia

### Team Members:

Jakub Hladik - Test Lead

Tyler Imboden - Quad Software Lead

Matthew Kelly - Webmaster and Documentation Lead

Dane Larson - Ground Station Lead

Blake Pries - Communications Lead

Austin Rohlfing - Controls Lead

Peter Thedens - Repository Lead

Kyle Trost - Team Lead

Team Website: <http://sdmay18-17.sd.ece.iastate.edu>

Team Email: [sdmay18-17@iastate.edu](mailto:sdmay18-17@iastate.edu)

Revised: December 4, 2017

Version 2.0

# Table of Contents

<b>1 Introduction</b>	<b>7</b>
1.1 Acknowledgement	7
1.2 Problem and Project Statement	7
1.2.1 Problem	8
1.2.2 Purpose and Goals	8
1.2.3 Solution Approach	9
1.3 Operational Environment	9
1.4 Intended Users and uses	9
1.5 Assumptions and Limitations	10
1.5.1 Assumptions	10
1.5.2 Limitations	10
1.6 Expected End Product and Deliverables	10
1.6.1 Quad Software	10
1.6.1.1 Autonomous Flight	10
1.6.1.2 Real-Time Flight Data Communication	10
1.6.1.3 GPS Navigation	10
1.6.1.4 Linux on Second Core	11
1.6.1.5 Assisted Manual Flight (“Trainer Mode”)	11
1.6.1.6 Virtual Flight Mode	11
1.6.1.7 Continuous Integration	11
1.6.2 Ground Station	12
1.6.2.1 Real-Time Transmission to Backend	12
1.6.2.2 Redesigned GUI	12
1.6.2.3 Multiple Object Tracking Capabilities	12
1.6.2.4 Data Analysis Tool	12
1.6.2.5 Generic Object Integration to Backend Capabilities	12
1.6.3 Controls Systems	12
1.6.3.1 Improved Stabilization	12

1.6.3.2 Advance Control Maneuvers	13
1.6.3.3 Linear Controls Model	13
1.6.4 Hardware	13
1.6.4.1 Second Quadcopter	13
1.6.4.2 Power Regulation Board	13
1.6.5 Documentation	14
1.6.6 Demos	14
1.6.7 Project Releases	14
<b>2 Specifications and Analysis</b>	<b>15</b>
2.1 Proposed Design	15
2.1.1 Quad Software:	15
2.1.2 Controls	15
2.1.3 Ground Station	15
2.1.4 Continuous Integration	16
2.2 Design Analysis	16
2.2.1 Quad Software	16
2.2.2 Controls	16
2.2.3 Ground Station	16
2.2.4 Continuous Integration	16
2.3 Implementation Issues and Challenges	17
2.3.1 Quad Software	17
2.3.2 Controls	17
2.3.3 Ground Station	17
2.3.4 Continuous Integration	17
2.4 Standards	17
<b>3 Testing and Implementation</b>	<b>18</b>
3.1 Interface Specifications	18
3.1.1 Ground Station Interface Specifications	18
3.1.2 Ground Station GUI Specifications	18

3.1.3 Ground Station CLI Specifications	18
3.1.4 Ground Station Access Point Specifications	19
3.2 Hardware and software	19
3.2.1 Automated Unit and Functional Testing	19
3.2.2 Flight Simulation	19
3.2.3 Flight Test	20
3.2.4 System Identification	21
3.3 Process Diagram	22
3.4 Modeling and Simulation	22
3.5 Results	23
<b>4 Closing Material</b>	<b>23</b>
4.1 Conclusion	23
4.2 References	23
4.3 Appendix	24

## List of Figures

<i>Figure 1-1: MicroCART quadcopter</i>	6
<i>Figure 1-2: Zybo Board</i>	10
<i>Figure 1-3: Present step-down converter circuit</i>	12
<i>Figure 3-1: Ground Station flow diagram</i>	17
<i>Figure 3-2: Diagram of automated testing</i>	18
<i>Figure 3-3: Simple quadcopter flight inside MCS</i>	18
<i>Figure 3-4: Flight Test with CLI</i>	19
<i>Figure 3-5: Design Process flow diagram</i>	20
<i>Figure 4-1: Block diagram of system hardware and software</i>	22

## Definitions

Term	Definition
CLI	Command line interface
Continuous Integration	automated process of running tests on every commit to the repository
Demo	Short for demonstration; this is one of the deliverables of the project: a demonstration of the quad's capabilities, for example, doing a backflip with the quad, finding an object and following it, communicating with a second quad to perform flight patterns
GPS	Global Positioning System; space-based radionavigation system using satellites to determine position; proposed to find position (x, y) when not in the lab using the VRPN system
Ground station	The application that runs on a host computer that communicates with the quad via a Wi-Fi connection and sends it coordinates to the quad
GUI	Graphical user interface
IR	Infrared wavelengths of light longer than visible light; used in the VRPN system to determine the position of the quad
LIDAR	Light Detection and Ranging; this is a system for determining the altitude (z) of the quad using the onboard sensor
Optical Flow	system using pattern of motion of objects, surfaces, and edges caused by the relative motion between the and the scene to determine position; used by the quad to calculate position (x, y) when not in the lab using the VRPN system
PID	Proportional-integral-derivative control system; standard control algorithm used on the quad
Quad	Short for quadcopter; this is the hardware platform we use in this project
Setpoint	in a control system, the target value for an essential variable
VRPN	Virtual-Reality Peripheral Network; this is the system used to determine the position (x, y, z) and orientation ( $\phi, \theta, \psi$ ) of the quad in the lab using a set of 12 stationary cameras and an IR transmitter on the quad

## 1 Introduction

Microprocessor Controlled Aerial Robotics Team or MicroCART project is centered around the development of a quadcopter (see Figure 1 below) and tracking system. This project has been in development since 1998 and the current system has been passed down since 2006. The project aims to create a stable and easy to use platform for researching control theory. The quadcopter flies primarily in the Distributed Sensing and Decision Making Laboratory within a twelve camera infrared tracking system.



*Figure 1-1: MicroCART quadcopter*

### 1.1 ACKNOWLEDGEMENT

MicroCART is a project that is assisted by graduate students working in controls under Dr. Nicola Elia and Dr. Phillip Jones. Additionally, as this is an ongoing project previous team members will also be providing help in understanding the current system. As such, we would like to acknowledge the assistance that will be provided by:

- Matthew Cauwels
- Robert Buckley
- Dr. Phillip Jones
- Dr. Nicola Elia
- May 17-16 MicroCART Team Members

### 1.2 PROBLEM AND PROJECT STATEMENT

We intend to create a modular platform that will primarily be used for research in controls and embedded systems and for department demos. This will entail building upon the current platform

designed by previous teams and adding/improving functionality for more advanced control maneuvers, more stable demos, and a more refined optical flow navigation system.

### 1.2.1 Problem

The MicroCART platform designed in previous years has many flaws that hinder its use for research and demo purposes. First, the platform fails to familiarize the user(s) of the system in a time horizon that would make it viable for research. This is because the current system does not have ample documentation available for the users of the system to learn about the platform and its uses. From the viewpoint of a user running demos the area within the VRPN system the platform as it stands is most stable when confined to flying within the VRPN system as the optical flow navigation can not hold position during flight. This means that the areas available to the quadcopter for demos can only utilize the small amount of space in the lab for a demo. During demos and for research use the quadcopter is limited to only flying to waypoints, it cannot perform flips or other control algorithms that could be used in research. Lastly, the quadcopter is controlled using a PID controller that requires logical guessing and checking to tune, we will be testing a new linear controller that can be computed faster and be tuned around multiple points on the non linear model.

### 1.2.2 Purpose and Goals

The MicroCART senior design team will serve several purposes. One purpose of the team is to improve on the existing quadcopter design in order to give graduate students a more stable platform to use for research and testing of control theory. Another purpose is to showcase the skills that a student in the ECpE department can gain throughout their time at Iowa State by creating an impressive demo that the quad can perform. The quad should also become more reliable so that anybody with little knowledge of the project should be able to read some documentation and feel comfortable performing the demo.

We plan to build upon the previous MicroCART team's platform by improving the stabilization, designing new demos, redesigning the ground station GUI, and building upon the virtual quadcopter software. The current system is relatively stable while the quad is within the VRPN system, but in order to use the optical flow for navigation, the system needs improvement. This will require us to fully understand the current system and design metrics that can be used to quantitatively show the results of changes. Additionally, we plan to implement the use of GPS to allow the drone to hold its position while navigating via the optical flow sensor. The demos that will be designed are meant to show the new functionality added to the quadcopter during our time working on the system. While working on the stabilization, the ground station GUI will be redesigned. The new GUI will allow for greater control of all functionality on the quadcopter and implement more safety measures to make sure that the user cannot cause unintentional harm to the drone. Next, the virtual quadcopter is a tool that allows testing of the controls and software prior to running the it on the quadcopter. This system is still in development and our goal is to have the ability to fully simulate movement and flights within the virtual quadcopter software, and to test this software on the quadcopter using Linux running on one of the cores of the ARM chip on the Zybo board. While working toward these goals, we want to make major improvement to the current state of documentation within the project that will allow next year's team to gain understanding within 4 weeks of gaining access to the files.



### 1.2.3 Solution Approach

Our approach to the various projects starts with becoming familiar with the current system. This includes updating documentation, reading existing documentation, and running previous demos. Upon gaining enough knowledge we will be building upon the existing platform to add demos and functionality that is currently capable. Along side of this we will develop testing hardware for parameterizing the motors for adding additional flight simulation capabilities for the virtual quad to improve testing as well as building a second quadcopter. By the end of the first semester we intend to have multiple object tracking working so that we can demo flight with two quadcopters. In the following semester each group will be working on adding new features. Ground Station will be updating the GUI to support real-time tracking and add support for the various demos. Controls will be developing a linear controls model to be tested against the current PID model. Quadcopter Software will be working on developing the new demos and work with hardware acceleration to support the new control algorithm. And lastly, the testing team will be finishing implementation of the virtual quadcopter to support full flight. During this whole period we will be demoing new features to show our clients and advisors progress and get feedback regarding functionality of the platform.

### 1.3 OPERATIONAL ENVIRONMENT

The end product (an autonomous quadcopter capable of the tasks described later in this document, henceforth referred to as “the quad”) will have two primary environments, one for each of its main data sources.

In order to fly using the VRPN software for position and orientation data, the quad must be inside of a small area (less than 10 m<sup>2</sup>) inside of Coover 3050. This lab is designed to cause very few environmental impacts on the quadcopter. Through the use of ventilation, window shades, and Coover’s heating and air conditioning, the lab has a nearly constant light and temperature with little to no accumulated dust to affect air quality.

Using optical flow to determine position and orientation, the quad could in theory be flown anywhere, but we will still avoid any circumstances that are significantly outside of normal conditions. Specifically, optical flow flights will not be held anywhere with temperature extremes, strong wind, or direct sunlight.

### 1.4 INTENDED USERS AND USES

The primary set of end users is composed of future MicroCART members and controls graduate students at Iowa State. For the goal of creating demonstrations for prospective students, someone from the two categories (the user) will be running the demo for them (the audience). This means that the users can be assumed to have competence in using multiple forms of programs (for example, either GUI or CLI) and in reading general technical documentation.

The other goal listed above regards the modular implementation of new control algorithms as a research opportunity for graduate students. These users have three primary needs from our product. The first is a robust and reliable system to decrease variation in test results. This includes having sturdy quad hardware, low communication latency, and a bug-free user interface. The second need is to have modular software with complete documentation to allow for them to achieve the implementation themselves, without the need for significant system rework or

intervention of the MicroCART design team. Finally, these students will need the data from the system identification in order to form their models. This includes information about mass, moments of inertia, motor resistances, rotor areas, and many other properties that determine the true actuation of the quadcopter.

## 1.5 ASSUMPTIONS AND LIMITATIONS

### 1.5.1 Assumptions

- No more than two quadcopters will be using the system simultaneously.
- Our VRPN camera system as it exists provides sufficiently accurate position data.
- The quad will be flying without significant external disturbances

### 1.5.2 Limitations

- Area limitation for multiple quadcopters within the VRPN camera system.
- Accuracy of onboard sensors (e.g. optical flow, LIDAR, IMU, GPS)
- Latency and range of the wireless link between the quadcopter and the ground station

## 1.6 EXPECTED END PRODUCT AND DELIVERABLES

The quadcopter system consists of three major subsections: the quadcopter software, the ground station, and the control systems. Each of the subsections is essential to meet the desired objectives and fulfill our requirements. Documentation and demos are also a major deliverable for our project and will be discussed.

### 1.6.1 Quad Software

#### 1.6.1.1 *Autonomous Flight*

The current platform allows for autonomous flight within the VRPN system or while using the optical flow sensor. This navigation is reliant on received coordinates from the ground station, we plan to allow the quadcopter to set its own waypoints to track objects within the VRPN system and with on board cameras. This will better support researchers, as only one person will be needed to fly the quadcopter safely, and allow for the creation of more advanced demos.

#### 1.6.1.2 *Real-Time Flight Data Communication*

During flight, the quadcopter saves 5 minutes of flight data to be transmitted to the ground station upon the flight ending. This action requires approximately 70% of the flight time to transmit the data back. This slows down the process of testing the quadcopter substantially and is not ideal if many test flights are planned to run. We plan to support the transmission of the data back to the ground station in real time. This will allow quicker analyzation of flight data, and reduce the risk of running out of battery after the flight is over to allow the data to be safely transmitted.

#### 1.6.1.3 *GPS Navigation*

The current optical flow navigation fails to hold position around a waypoint, as the optical flow system does not track the slight changes to the position as the quadcopter drifts. The use of the GPS will allow more precision while navigating with optical flow. Additionally, once this system is fully functioning, the quadcopter will no longer be confined to the area trackable by the VRPN system.

#### 1.6.1.4 Linux on Second Core

The Zybo board, shown in Figure 2, contains a Zync-7000 SoC, which runs all the software on the quadcopter, is only using one of its two ARM cores. The second ARM core could be used to run Linux to increase the usability of the quadcopter. This could potentially run the virtual quadcopter software to allow navigation, or be used by researchers for greater functionality as it could support libraries such as OpenCV for computer vision.

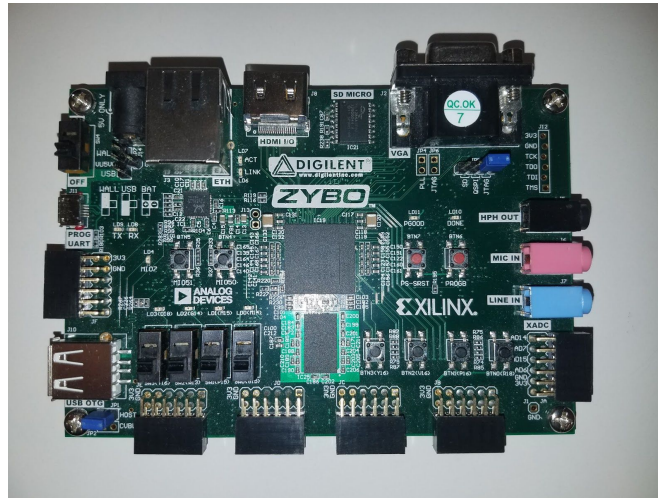


Figure 1-2: Zybo Board

#### 1.6.1.5 Assisted Manual Flight (“Trainer Mode”)

This platform, as it evolves, is also meant to be a learning platform where we can progressively turn on functionality as the user gets more experienced with controlling the quadcopter manually. As we can get the quadcopter to hold position within the VRPN system, we can combine this into a sequence of steps for a user to learn how to use the quadcopter. This will start with the user only being able to move in the x and y directions, then allowing z, and finally taking away all assistance but stabilization. This will give researchers and members of next year’s team the ability to learn how to fly the drone, before running different control algorithms, so that they can possibly save the drone if an error occurs by manually flying the drone to a safe landing.

#### 1.6.1.6 Virtual Flight Mode

Through the use of the MicroCART Simulator, the quadcopter can operate in a virtual flight mode, allowing the sensors and actuators to be emulated in software. This provides Software-In-Loop testing capability as well as Processor-In-Loop testing. Controls This will aid in accident prevention as the potential virtual crashes will have no impact on the physical quad.

#### 1.6.1.7 Continuous Integration

Continuous Integration is the system that tests changes to code using the virtual quadcopter software. We plan to expand the tests to allow for more thorough testing of code. This will require expanded functionality of the virtual quadcopter to allow for testing of all components of the software as well as control algorithms. The tests are ran automatically by a git hook script once new changes are committed into the repository.

## 1.6.2 Ground Station

### 1.6.2.1 *Real-Time Transmission to Backend*

The backend, as it stands, is setup to transmit VRPN x-position and y-position, as well as waypoints. There will have to be improvements to the backend to allow for the real-time transmission of data. This data can then be used to display important flight data in real time. This will give immediate results in the case of incorrect behavior and allow the GUI to display more information to the user.

### 1.6.2.2 *Redesigned GUI*

The GUI is not fully functional in its current state, and does not do checking for incorrect data entered by the user. The new GUI will add all missing functionality and allow users to switch modes of navigation during flight (if conditions are met). The GUI will also perform checks when sending coordinates to make sure that the user does not try to have the quadcopter accelerate into the ground or perform other tasks that may break a component on the quadcopter. Lastly, the controls graph generated currently is an image, we would like this to be capable of interaction so the user can change PID values from within the GUI.

### 1.6.2.3 *Multiple Object Tracking Capabilities*

To allow the quadcopter to track an object we first need to get the camera system to recognize a second object as trackable. The VRPN system has the capability to track more than one object and will send that information to the backend. The backend needs to send this new information to the quadcopter for the quadcopter to track the object.

### 1.6.2.4 *Data Analysis Tool*

As it stands all flight data is logged on the PC and there is a set of separate MATLAB scripts to perform analysis and visualize the data. We are proposing a new tool that can be used by graduate students to easily view current and past data and use a variety of analysis and visualization scripts. This tool will allow for users to add all their logged data and new scripts to the tool and have them automatically recognized and listed for use.

### 1.6.2.5 *Generic Object Integration to Backend Capabilities*

The current platform is only fit for use with our specific quadcopter that accepts our defined commands. We plan to expand the functionality of the backend to allow connection of other quadcopters or trackables into our system. We are proposing the use of an initialization file for the backend and new adapters that can connect to the backend that can be implemented by the user of the adapter.

## 1.6.3 Controls Systems

### 1.6.3.1 *Improved Stabilization*

The quadcopter is currently stable and works in demos, but it does still sway slightly in different situations. We would like to fine-tune the PID values to increase stabilization even further. This will benefit those doing controls research, but it will also allow the demos to run more smoothly. This will require new analysis metrics for flight data to show the actual increase, rather than by the eye-test.

### *1.6.3.2 Advance Control Maneuvers*

The controls will be further developed and this will allow the quad to do a backflip when commanded. This allows us to continue the controls research and further refine the capabilities of the quadcopter platform. This also gives us the opportunity to more fully understand the controls implemented by the previous team. When fully implemented, the backflip will be controlled completely by the quadcopter with the onboard sensors, and the command will come from the ground station.

### *1.6.3.3 Linear Controls Model*

Implementing a new controls algorithm in the form of a linearized model will further our goal of improved stabilization and precision. The model will be built with more direct numerical parameters that represent physical quantities on the quadcopter. This will give us a fully distinct controls algorithm to compare to the current PID; having two unique methods allows for isolation of other components of the system.

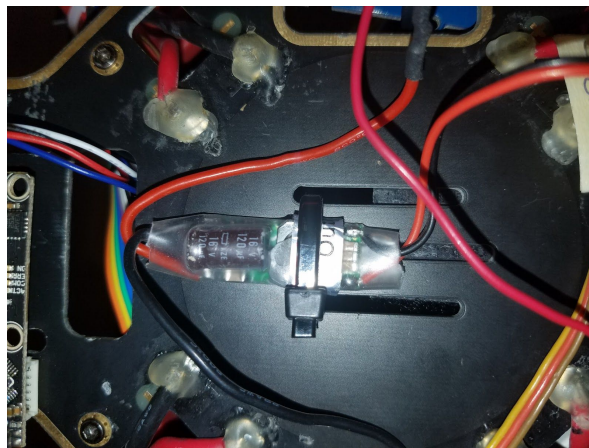
## *1.6.4 Hardware*

### *1.6.4.1 Second Quadcopter*

A second quadcopter will be developed as requested by our client and advisor, Dr. Jones. This provides a second testable quadcopter if one were to ever get damaged. It also allows for additional testing if there are multiple users running tests.

### *1.6.4.2 Power Regulation Board*

The Zybo board and the motors require different voltage levels to operate. The current method to control this and add safety measures is to have an additional cord that needs to be plugged in and unplugged before and after flights and to use a voltage regulator circuit, shown in Figure 3. A board that can control both voltage levels and provide a mechanism to turn off and on the motors, will be created for ease of use of the system. This board will regulate the LiPo battery from 11.1V down to 5V at 3A.



*Figure 1-3: Present step-down converter circuit*

### 1.6.5 Documentation

Many areas of the code are lacking documentation. This includes function and parameter explanations, especially in the quadcopter software related code. Additionally, the documentation is lacking in other areas such as demos. Our goal is to have documentation for all existing demos, documentation consistent in all code, and documentation for the research done during our time on the team.

### 1.6.6 Demos

As one purpose of this project is to showcase the talents within this department, new demos need to be developed to showcase yearly changes. These demos are performed to controls classes as well as to undergraduate students. We plan to develop new demos including: using the VRPN system to catch balls, using two drones to build a bridge in the air for a remote-control car, and a fully functional optical flow demo.

We plan to implement the following major demos:

1. Have a quad that tracks an object on the ground, or in air, and maintains a set distance away from it.
2. Have multiple quads running at the same time flying together.
3. Have multiple types of quads running at the same time flying together.

### 1.6.7 Project Releases

The current project version consists of the slight sway in VRPN system and a drifting quadcopter when using the optical flow for navigation. We plan for five releases, the first having the improved stabilization in all demos and a stable demo with LIDAR for height. Next, the transition times between waypoints will be reduced to start designing the ball catching demo. This version will also have a stable optical flow demo when around a given point (will use GPS to solve this issue). The following release will have the functionality to start testing our more advanced demos such as the sky bridge. After further testing the fourth version will have the demos fully functioning within the VRPN system and the final release will have the same demos but with either optical flow or VRPN to perform them. During the third and fourth releases we plan to demo to our clients and advisors to get feedback and further refine the created demos.

## 2 Specifications and Analysis

### 2.1 PROPOSED DESIGN

#### 2.1.1 Quad Software:

First of these two include implementing a system for inexperienced quad flyers to fly the quad with ease. Our current approach to this is to allow the PWM wave generated by the RC controller to control the height setpoint.

The second major goal of the quad software team is to modify the existing hardware and software running on the FPGA to support new demos. This includes but is not exclusive to the following: hardware acceleration to reduce utilization for drivers, adding new types of packets to increase the

capability of communication between the quad and ground station, and implementing software to enable object tracking.

### 2.1.2 Controls

The controls for the quad is currently implemented using nested proportional-integral-derivative (PID) controllers. There is a set of PIDs for each of the three Cartesian components of position (x, y, z) and one for yaw (rotation around the z-axis). These were chosen because they achieve a very configurable approach to quadcopter controls, as modifications to the quad can be accounted for by simply adjusting the various PID constants.

The future plan for the controller is to implement a nonlinear model with distinct linear segments that is capable of more precise control of the quad. It will achieve this through the use of more precise mathematical information about the quad and its dynamics.

### 2.1.3 Ground Station

For the ground station we have decided to design a more researcher oriented interface and features. We plan on creating a more robust software error reporting system, real time logging from quads, a built in data analysis tool, adding safety rules for different types of experiments, a more detailed documentation scheme for all files and be able to use other drones/trackables with our ground station software.

In doing this we believe our platform to be more ready for actual research tasks. Also with our update it will allow for teams who work on microcart in the coming years to be able to spend less time getting up to speed with the project and more time working it.

### 2.1.4 Continuous Integration

By performing Continuous Integration (CI), we will ensure that software committed into the Git repository passes a series of tests defined by test script files. These regression tests will initially be created to test all presently implemented functionality (insofar as they can be with individual unit tests). As new features are implemented, we will also add corresponding tests that test against their functional and nonfunctional requirements so that all future commits will be tested against the entire accumulated functionality of the software. MicroCART Simulator will aid in continuous integration of the controls testing.

## 2.2 DESIGN ANALYSIS

### 2.2.1 Quad Software

In terms of Quad software we have currently not made many modifications to the system from a functional perspective. We have looked into modifying the way our system boots to allow for multiple different types of sensors as feedback, but to no success yet. one thing I think we really need to implement is a better system of testing. When we attempt to test any changes to the system it can take several minutes and in turn slow development time significantly. One idea of making a wall plug to power the board and sensors but not the motors as a testing platform instead of the batteries. This would enable faster testing iterations and improve development speed significantly. Our solutions as of now seem to give us strengths in functionality but at the sacrifice



of future development time increasing. this is due to hardware acceleration being costly (in terms of time) to modify and test as opposed to a software solution.

### 2.2.2 Controls

As described in the Proposed Design section, the plan is to implement a nonlinear control in a finite number of linearized segments. This solution will have more precision than the existing PID controllers by computing control signals directly from the theoretical dynamics of the quad. This model will use a very precise representation of the quad obtained from planned work in system identification. To emphasize the point from above, this approach allows for higher precision - and thus speed - than a PID implementation at the cost of being more difficult to configure when the quad changes and having a smaller range of operation if not enough linear segments are included.

### 2.2.3 Ground Station

We currently have a robust framework and backend with a bare bones GUI implemented for controlling a single quadcopter. Moving forward we plan on using the backend only modifying what is needed to implement multiple quads and fix any bugs we find. However we will focus heavily on GUI development and making our platform one that is extremely easy to work with for demos and research. As defined in 1.6.2 we plan on adding real time transmission, redesigned GUI, a data analysis tool and multiple object tracking capabilities. Each of these parts will either make research easier to use, take less time to collect data, better review the data gathered, and allow for more complicated and impressive demos.

### 2.2.4 Continuous Integration

Integration of new features into the system is done through a series of tests ran automatically after every commit in the online Git repository. Tests are written in scripting programming languages such as Perl or Python. The merge request merge is unlocked upon successful run of the test scripts. MicroCART Simulator (MCS) will be a virtual environment for the current virtual quadcopter. Currently, the MCS is in the early stage of development and it is dependent on the successful completion of the quadcopter flight model description. Once completed, we will be able to simulate virtual flight and thus test the controls software along with our current simple software test.

## 2.3 IMPLEMENTATION ISSUES AND CHALLENGES

### 2.3.1 Quad Software

One of the main challenges of the Quad software design will be in transitioning platforms from XPS to vivado. This is due to Vivado not supporting the same IP's as XPS. Additionally the team will need to create, test and implement the custom IP's of the past teams. The next major challenge will be in maintaining a low control loop time while expanding the amount of data being sent to the ground station.

### 2.3.2 Controls

The primary challenge in the new controls implementation lies in the technical difficulty of the physics and mathematics. We will be collaborating with a controls-focused PhD student whose will be handling the theoretical model computation. This leaves the translation challenges to us:



accurately and efficiently implementing the precise software and simulation to reflect the specified controller.

### 2.3.3 Ground Station

There have been many issues when implementing new features. The ground station was developed for a single quad of a single type, this has caused some design issues with trying to implement multiquad support. Another big issue the ground station team has encountered has to deal with threading on the graphical user interface and blocking operations on the front end. Finally there are some design challenges working in real time when trying to send data to and from the quad.

### 2.3.4 Continuous Integration

One of the biggest challenges in continuous integration was designing an interface that would allow us to test the platform at the system level. The existing virtual quadcopter software did not simulate flight. We needed to create a reliable flight simulator to be able to test the controls and controls related quadcopter software.

## 2.4 STANDARDS

There is not a direct set of standards that is well suited for quadcopter drone software and hardware development. IEEE publishes some high-power electronics safety standards, but they are designed for systems significantly larger than ours. There are also pure software standards, but our project, as seen above, is not purely software. As such, the closest thing we have to a standard to follow is DO-178B, the aviation software standard created by the United States government. This still has its share of shortcomings in relation to our project, however. Given the experimental nature of MicroCART and its remarkably low risk of serious injury upon a significant software failure (compared to the manned aircraft that the standard was designed for), some of the requirements should be considerably loosened. For example, DO-178B gives an acceptable frequency of failure for each level of significance, and even the lowest level of risk is given an acceptable frequency of one failure per 1000 hours, which is unreasonably (and unnecessarily) strict given the scope and scale of the project at hand. Nonetheless, the standard sets forward a useful sequence of steps in which there is a process to work from requirements to code and then to fully test both for accuracy and completeness.

## 3 Testing and Implementation

### 3.1 INTERFACE SPECIFICATIONS

The major interfaces for the MicroCART project involve the ground station and the multiple areas including the Backend, Frontend, CLI, and GUI. Figure 3-1 shows the communication between the various areas and the sockets outside of the ground station computer.

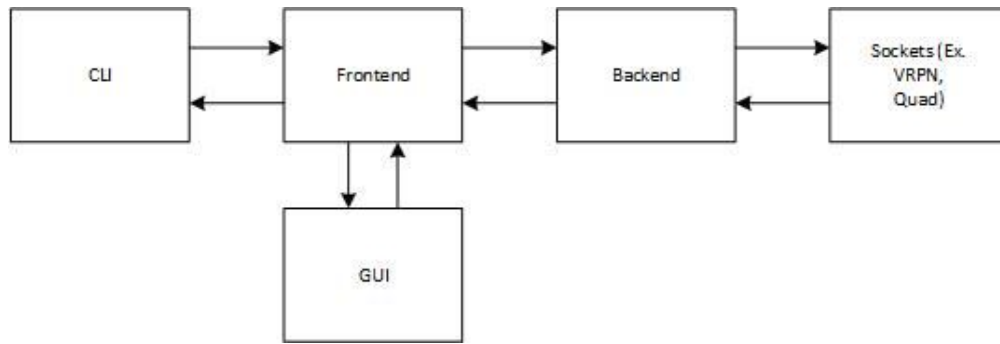


Figure 3-1: Ground Station flow diagram

### 3.1.1 Ground Station Interface Specifications

The ground station interface consists of two major components including the Backend and Frontend. The Backend provides a server that the VRPN system and user interfaces connect to via sockets for communication. This allows the obtaining of position information from the VRPN system as well as accepting commands from the frontend. Additionally, the backend connects to the quadcopter also via a socket. The frontend provides methods that handle the interfacing with the Backend for both the CLI and GUI.

### 3.1.2 Ground Station GUI Specifications

The Graphical User Interface will have four tabs that allow for starting of the backend, viewing the control graph, navigation, and real-time graphing. The backend tab both starts the backend and will also allow for the use of the Command Line Interface directly within the GUI. The controls tab allows the user to change the constant values within the controls to tune the PID values during flight. Navigation allows sending of coordinates to the quadcopter and the running of demos. Lastly, the real-time graphing tab will allow a configurable real-time data transmission between the quadcopter and GUI to graph during flight.

### 3.1.3 Ground Station CLI Specifications

The Command Line Interface provides direct access to the commands that the GUI sends for the user. It does not provide many of the extra features that the GUI provides such as automating setpoint sending, real-time transmission, and viewing the control graph. This is a more lightweight interface that still provides the use of all the same commands sent from the GUI.

### 3.1.4 Ground Station Access Point Specifications

To support multiple quadcopters we plan on creating a wireless access point on the ground station. This will require a server software such as hostapd and a reprogramming of the wifi on the current quad. Once implemented we will be able to host multiple quads simultaneously from a single ground station [\[2\]](#).

## 3.2 HARDWARE AND SOFTWARE

### 3.2.1 Automated Unit and Functional Testing

All commits to the Git repository are tested through a suite of continuous integration scripts. These scripts perform unit tests on the software that runs on the quad and higher level functional tests that run on the “virtual quad” which interfaces with a set of Unix drivers. The scripts are run

automatically using the GitLab pipeline integration. We will continue to improve the test coverage over the existing code, and as more features are added, tests will be added to cover them. The automated testing flow is shown below in Figure 3-2.

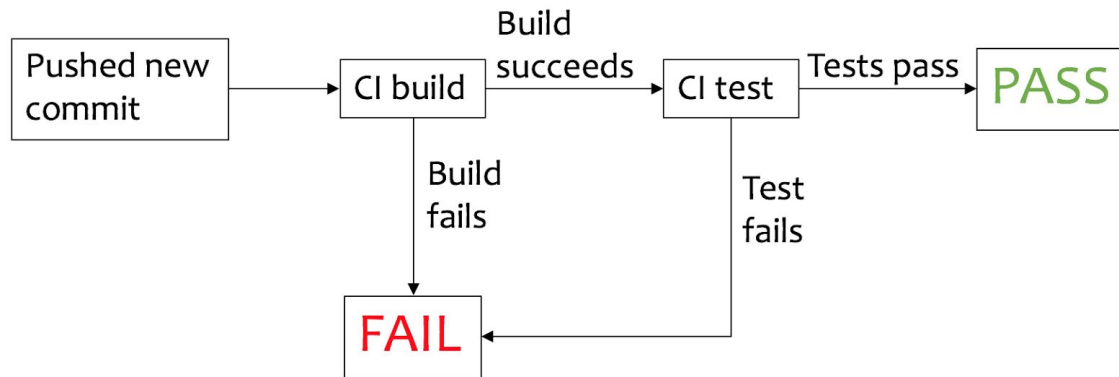


Figure 3-2: Diagram of automated testing

### 3.2.2 Flight Simulation

The correctness and requirements of the quadcopter software will be tested through the MicroCART Simulator (MCS) and the help of the simulator event test scripts. Different flight regimes will be tested and verified whether the quadcopter position and orientation are within a threshold. In case of an accident, ground contacts are detected and the unsuccessful test is terminated early. An example of the simulation output is provided in Figure 3-3.

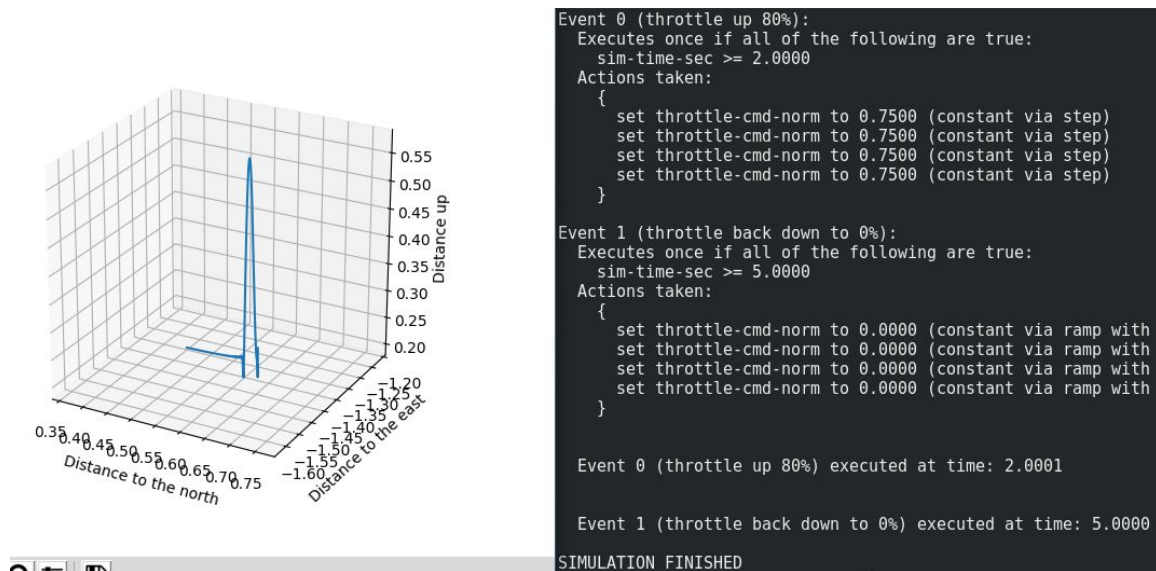


Figure 3-3: Simple quadcopter flight inside MCS

### 3.2.3 Flight Test

The current implementation of the ground station allows for a list of setpoints to be loaded into the GUI to allow for autonomous navigation. The quad will automatically relocate to the next point in the list once it gets within a defined range of the setpoint. We plan to create a list of setpoints that

will be used to test each new build of the quad software. This will enable us to test edge cases and to have a flight pattern that will test the more extreme patterns of movement. Another method of testing is to use the CLI to directly call commands that we are testing. In Figure 3-3 the CLI is shown in the top left, Backend in the bottom left and the flight is on the right.

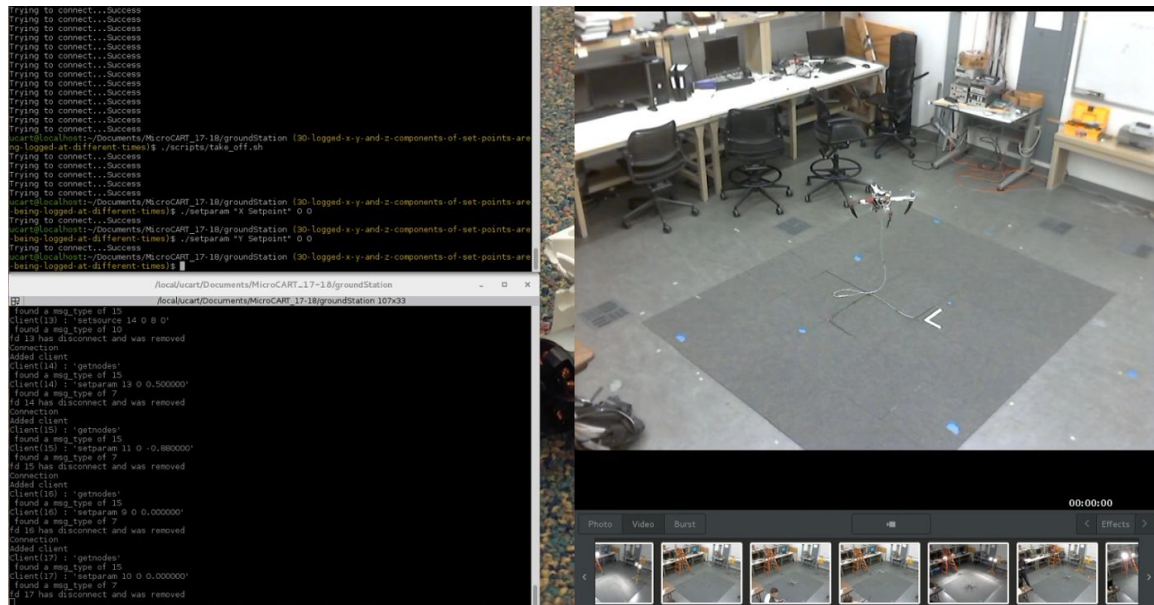


Figure 3-4: Flight Test with CLI

### 3.2.4 System Identification

Eventually, we plan to build custom testing fixtures for use in system identification. The first we will use is a stand to measure the thrust of a single motor with a prop. It will be connected to a scale to measure a difference between the gravitational force when stationary and when producing thrust. The main non-obvious physical feature of this component is its length; it is important to distance the rotor from the scale to minimize ground effect. Another piece of testing hardware will be similar, but for measuring torque. The physical setup will be different, but it will still use a custom-designed hardware connecting a single motor under test to a scale. This system identification will be used in forming models for both the the simulator (as described above) and the controls algorithm.

### 3.3 PROCESS DIAGRAM

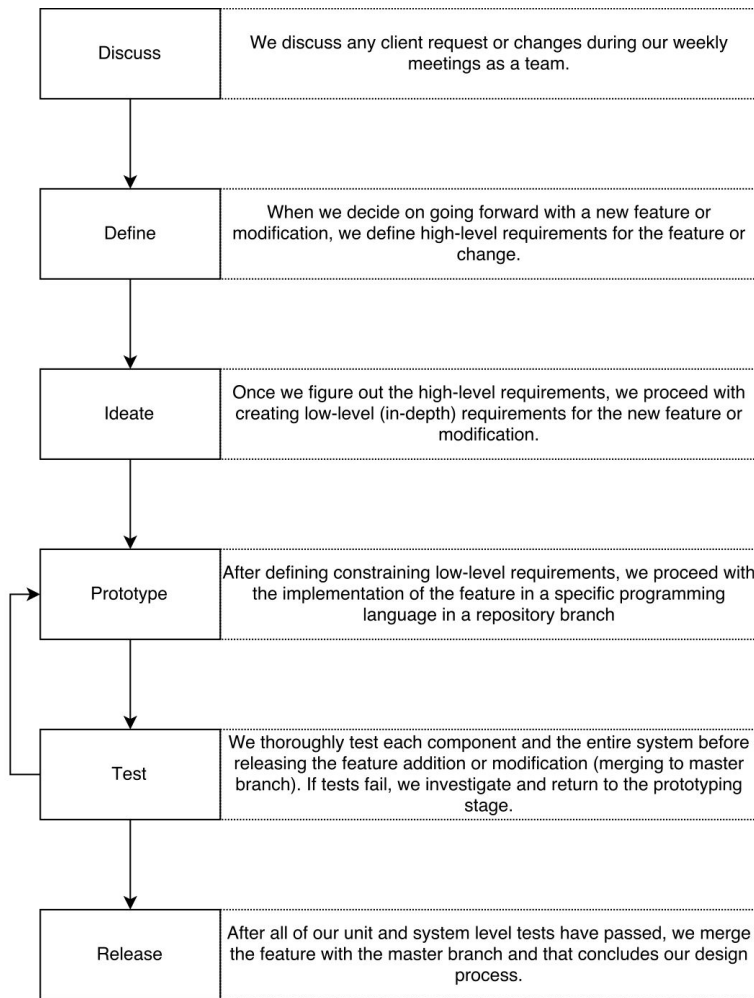


Figure 3-5: Design Process flow diagram, inspired by [\[3\]](#)

The above diagram explains the process we will use as a guide to development. To effectively produce and improve upon the existing system we must first discuss progress and determine what must be addressed. After determining the problems we must further define them and come up with solutions for those defined issues. After brainstorming prototyping solutions would be the next step. Moving forward, those solutions should be tested using the testing method described in section 3.2. After the correct solution is determined and confirmed that solution should be released into our master branch in the repository.

### 3.4 MODELING AND SIMULATION

To test the controls, we have a Simulink model that simulates three things: the quad control logic, the actuation of the quad given the output from the controls (i.e. physics), and the sensor and communication system. These three components tied together in a loop allow us to easily run

initial tests on modifications to (or new implementations of) the controls algorithm without the physical risks of running the untested model on the quadcopter itself. This model is based off of [\[1\]](#) and takes into account as many factors as possible (for example, the Sensors component passes the data and simulated noise through quantization and time delay), but it is still not a guaranteed proof of functionality.

Software-In-Loop and Processor-In-Loop simulation will be available through MicroCART Simulator (MCS). MCS backend runs on a flight dynamics simulator called JSBSim which provides environment simulation for aerial vehicles. The controls software is connected to MCS using a TCP socket and the output of the controls software is connected into the simulator to form a closed loop. By running different scripted scenarios, we can simulate different flight regimes and test our controls and the overall correctness of our quadcopter software.

### 3.5 RESULTS

Much of the semester to this point has been to the end of producing documentation, investigating the existing system implementation, planning future developments, and initializing test structures. As such, we have no results to report up to this point (October 15, 2017).

We will update this section of the document upon the completion of a meaningful amount of testing and collection of results.

## 4 Closing Material

### 4.1 CONCLUSION

Our MicroCART team has been steadily working to produce a more stable flying quadcopter that can be easily demoed to other students and faculty. We have vastly improved documentation for the entire MicroCART project that has been passed on to us from the previous team. With complete documentation for the current system, each sub team of MicroCART will be able to work towards their goals and allow the sub teams to better understand what how their portion interacts within the complete system. The quadcopter software sub team will continue to work towards their goals of improving the flight data communications, integrating GPS navigation for more precise flight, utilize the second ARM core on our board that is currently unused, and also add different flight modes to the quadcopter that will allow beginner and advanced users to operate the quadcopter. The ground station team is looking to improve the backend of the system to allow for the real-time flight data transmission. They will also redesign the GUI of the system and develop a new tool to analyze the flight data. Controls team is focused on tuning the PID values to further stabilize the quadcopter flight. Controls will also work on developing an advanced demo such as having the quadcopter do a flip. The continuous integration team will work on ensuring that all code committed to the Git repository is functional. They are also responsible for creating the tests that will determine the functional and nonfunctional requirements for new features. With all of the sub teams working on their portion of the project and focusing on their main goals, we should have no issues meeting the requests of our client.

## 4.2 REFERENCES

- [1] M. Rich, "Model Development, system identification, and control of a quadrotor helicopter" in *Iowa State University Digital Repository*, 2012
- [2] Wehr, David. "ESP8266 Wifi Latency Testing." 17 Sept. 2016,  
<https://docs.google.com/document/d/1VU99wMgkqK2EgbNLdqrDhvj9iikfk2gtUYQ367K5-Q/edit#heading=h.soog8emj18jx>
- [3] Plattner, Hasso. "An Introduction to Design Thinking PROCESS GUIDE." in *Institute of Design at Stanford*  
<https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/36873/attachments/74b3d/ModeGuideBOOTCAMP2010L.pdf>

## 4.3 APPENDIX

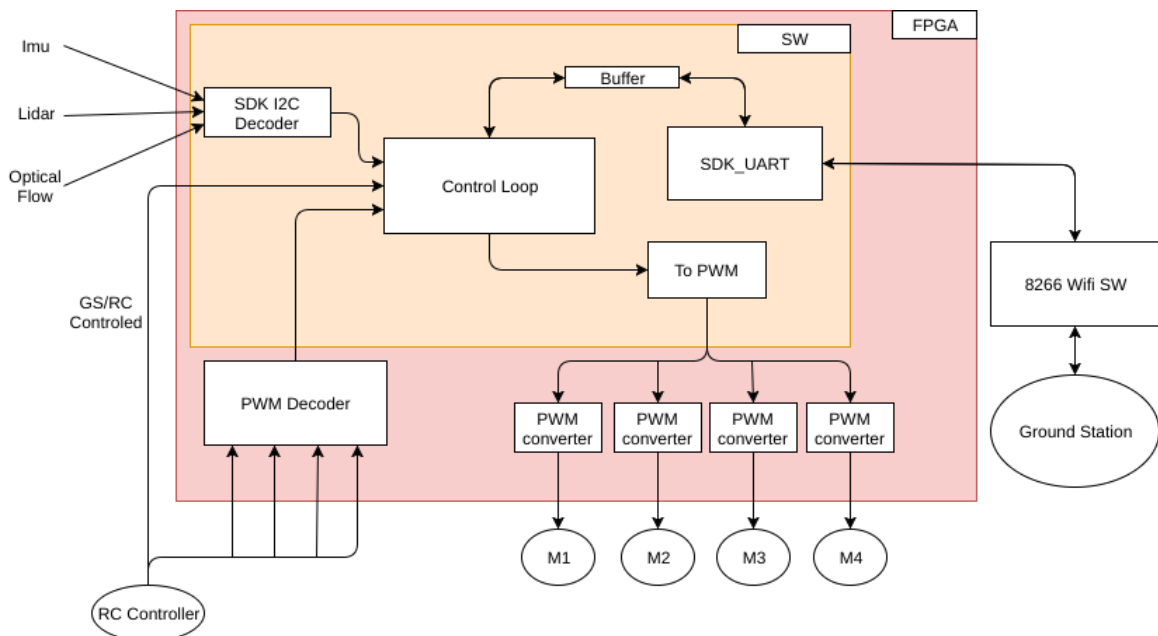


Figure 4-1: Block diagram of system hardware and software