MicroCART 2017-2018
Project Plan

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

Revised: December 3, 2017

Version 3.0

Contents

Definitions

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

LIDAR: Light Detection and Ranging; this is a system for determining the altitude (z) of the quad using the onboard sensor

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

GUI: Graphical user interface

CLI: Command line interface

IR: Infrared; wavelengths of light longer than visible light; used in the VRPN system to determine the position of the quad

VRPN: Virtual-Reality Peripheral Network; this is the system used to determine the position (x, y, z) of the quad in the lab using a set of 12 stationary cameras and an IR transmitter on the quad

Setpoint: in a control system, the target value for an essential variable

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

Continuous Integration: automated process of running tests on every commit to the repository

PID: Proportional-integral-derivative control system; standard control algorithm used on the quad

Stable Platform/System: Decrease or maintain low latency, decrease deviation and oscillation when holding set point, decrease deviation from path between setpoints during movement, and increase accuracy of measurements from sensors.

Quadcopter or Quad: The MicroCART project is based around the use of a Quad-rotor helicopter to test controls and software.

Figures

1 Introduction

1.1 Project statement

Microprocessor Controlled Aerial Robotics Team or MicroCART project is centered around the development of a quadcopter (see Figure 1 below) and tracking system. The project aims to create a stable and easy to use platform for researching control theory.

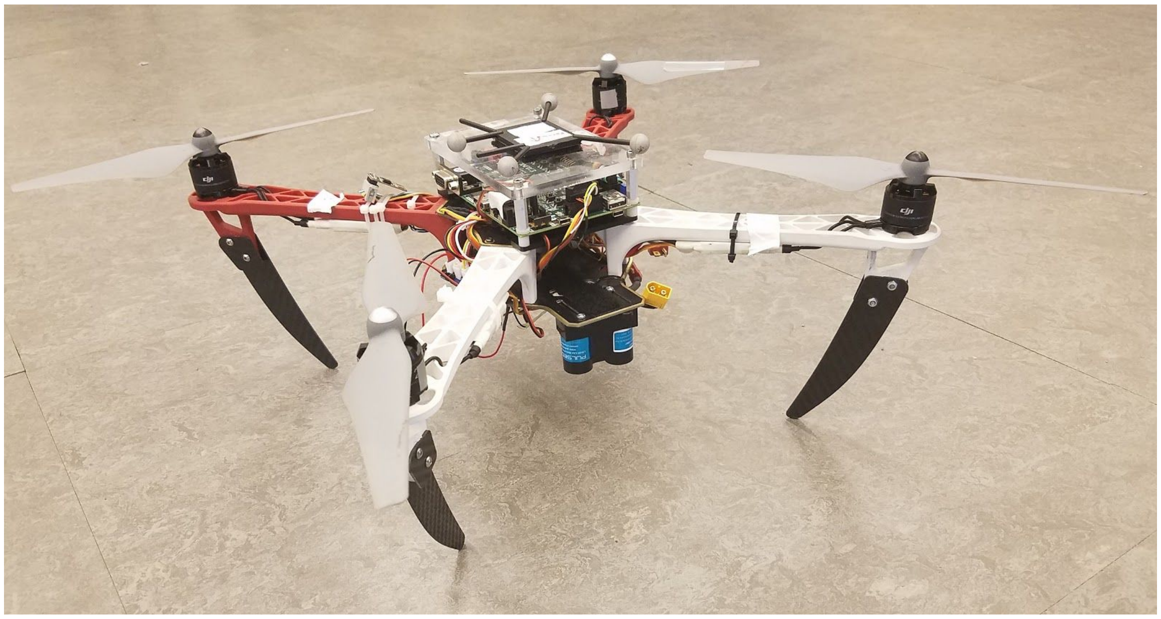


Figure 1: MicroCART Quadcopter

1.2 Purpose

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. Another purpose is to showcase the skills that a student in the ECPE department can gain throughout their time at Iowa State by creating demos 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 documentation and feel comfortable performing the demos.

1.3 Goals

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 platform is able to maintain position but experiences jitter during flight within the VRPN system. This jitter is more problematic while flying using the optical flow sensor for navigation and needs improvement to provide stable navigation outside of the VRPN. This will require understanding the current system and designing 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 and/or others. 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.

2 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.

2.1 Quadcopter Software

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.1.4 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.

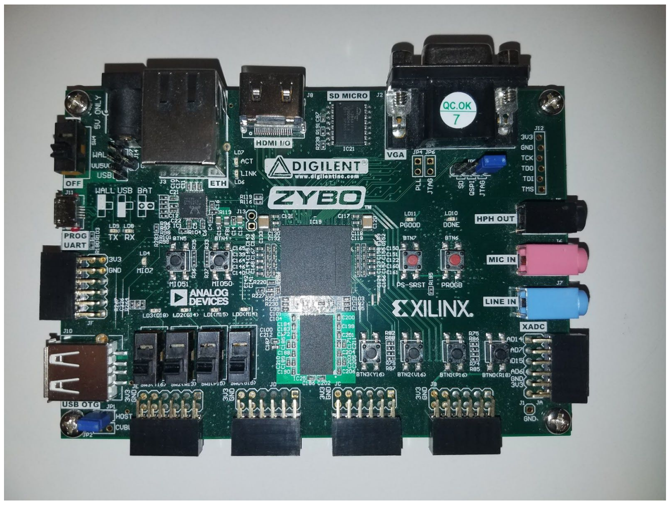


Figure 2: Zybo Board

2.1.5 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.

2.1.6 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.

2.1.7 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. A flight simulator will aid the testing of the control algorithms. We are planning on creating a robust flight simulator based on an existing flight dynamics engine which will be further integrated with the virtual quad software. On the server side of our git repository, the integration tests are run automatically by a git hook script once new changes are pushed to the repository.

2.2 Ground Station

2.2.1 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.

2.2.2 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 with the GUI.

2.2.3 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.

2.2.4 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 redesign of the current 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.

2.2.5 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.

2.3 Control Systems

2.3.1 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 of precision, rather than by the eye-test.

2.3.2 Advanced 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.

2.3.3 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.

2.4 Hardware

2.4.1 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.

2.4.2 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 and unplugged in 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.

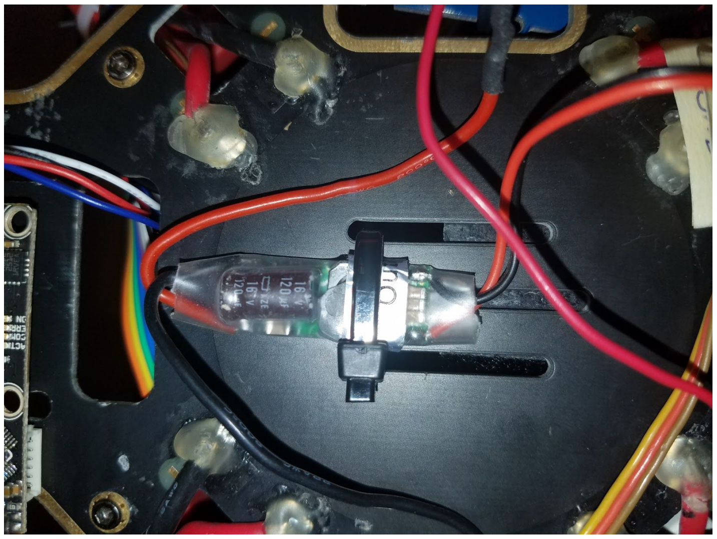


Figure 3: Current Voltage Divider Circuit

2.5 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.

2.6 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.

3 Design

3.1 Previous work/literature

The mathematical model used to control the quadcopter in its current form was given to us by the previous MicroCART team. This model was based off the work of Matt Rich in his thesis "Model Development, System Identification, and Control of a Quadrotor Helicopter". [1]

The previous team also mentioned that there is a similar commercial product being developed by Ascending Technologies. Ascending Technologies has a range of research UAVs, each with different prices and features. [2]

We reviewed the current platform and looked at the shortcomings of building off of the current state. The current platform is split into three areas of development being controls, quadcopter software, and ground station, with testing in all areas. This split between development areas is logical for this project. On the other hand the code base, while expansive, is very hard to understand and is difficult to change. There is use of symbolic links in the build process, and external references used in the tools for building quadcopter boot files cause many errors in the tools. We will not be fixing the workarounds used within this project, as this would require a major repository rework and is not within scope of our project. Additionally, the documentation for the project is severely lacking. Our team put a timeframe of eight weeks to gain understand of the existing platform. This is because there is a lack of upper level documentation, as well as in-code documentation. Another shortcoming is that our work will need to keep existing legacy functionality to maintain system usability. This leads to us needing to develop more complex and possibly less optimal solutions because we need it to fit within the system as it stands.

3.2 Proposed System Block diagram

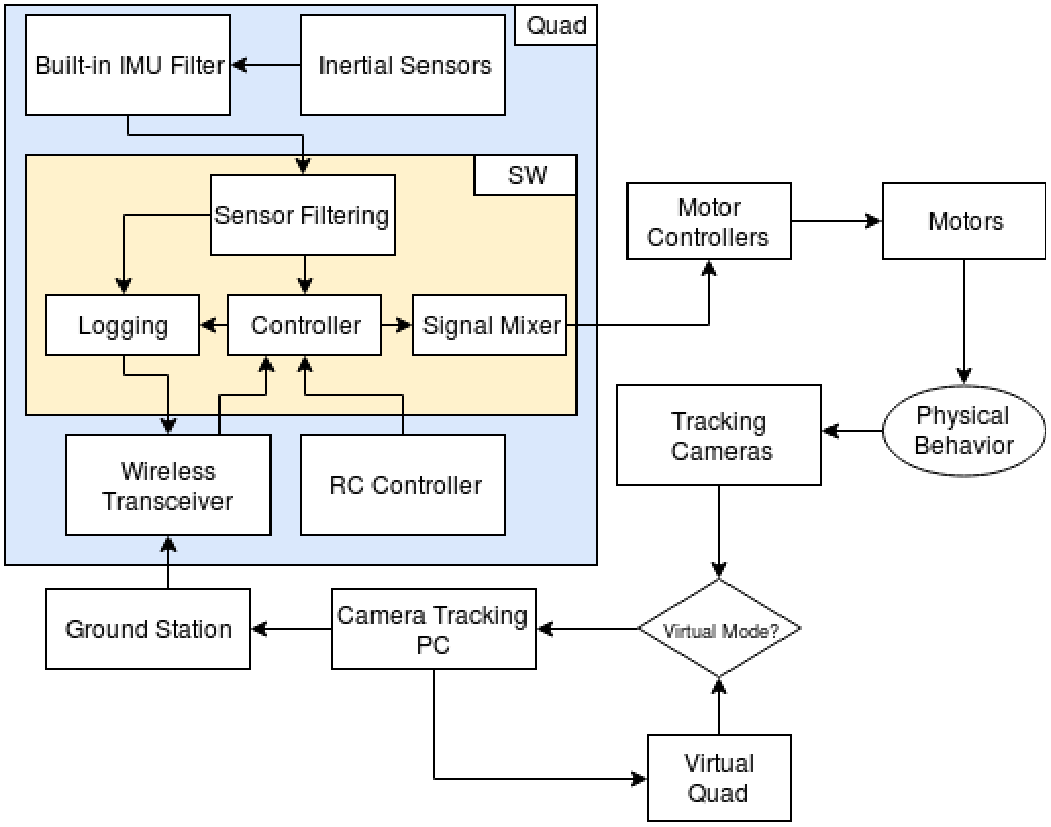


Figure 4: System Block Diagram

The figure above(Figure 4) represents the planned connections between different pieces of our project. The key areas that will require work will be the “Controller”, “Ground Station”, and

"Logging". The controller is the system that takes in the current orientation and the setpoint and adjust the quad accordingly. The Ground Station is the system that detects that current position of the quad using the cameras and communicates that and other key information to the quad using Wifi. Lastly the Logging portion is the system that takes in key metrics and sends them to the ground station for review. This step is key in debugging and evaluation.

The quad section of the diagram is focused on the actual hardware and software processes happening on the quad itself. These include the sensors, filters, and controllers on the hardware side and the controllers and logging on the software side. Outside of the quad are both the environmental factors we face and the system we use to track and communicate with the drone. The major component outside the quad that our group is developing is the "Ground Station". It keeps the quad updated on its current position (when flying in the VRPN system) and send setpoint information for the autonomous mode.

3.3 Technical Approach

3.3.1 Quadcopter Software

3.3.1.1 Autonomous Fight

We will update the quad software to allow the quad to receive object locations and set them as target waypoints.

3.3.1.2 Real-Time Flight Data Communication

Currently the quad only sends data back at the end of the flight. It sends back up to five minutes of flight data and the transmission of data takes about 70% of the flight time. We would like to change this, so that the data is sent back while the quad is in flight.

3.3.1.3 GPS Navigation

In order to improve the quads accuracy while flying we would like to add a GPS sensor to the quad. Unlike the VRPN system, the GPS sensor will not give an accurate location down to milimeters. It will however give the relative location of the quad when outside the camera system.

3.3.1.4 Linux on Second Core

Running linux on the second core would actually be quite trivial. Adding the appropriate files to support linux onto the SD card as well as adding the correct bootloaders when creating the boot-file. The real non-trivial approach here comes when implementing virtual quad. We are not at the point we need to further understand how the virtual quad works then reassess our options for approach.

3.3.1.5 Assisted Manual Flight ("Trainer Mode")

The approach we plan to use in reference to further assisting manual flight would be to further filter user input from the controller. Specifically in reference to throttle, in order to meet requirements, we would instead interpret the input on the throttle to a height setpoint rather than throttle. We would do so by creating a PID in order to get the quad to setpoint. The next step would then be to apply such a filter in all directions of motion. In order to do this we would need to first establish Autonomous Flight. (See approach in 3.3.1.1)

3.3.1.6 Continuous Integration

We will improve the testing infrastructure by adding additional test scripts to be executed after every commit, making existing tests more robust and thorough, and use standard test libraries. We will also look at porting existing tests from C to Ruby.

3.3.2 Ground Station

3.3.2.1 Real-Time Transmission to Backend

The backend already has the capability to communicate with the quadcopter, but is only able and setup to download the quadcopter log. We plan to open a socket with the quadcopter to have it incrementally send the data and log it on the Ground Station PC until the flight is finished. This will be done using new packets that will be sent out real-time asking for parameters that are being graphed.

3.3.2.2 Redesigned GUI

We will start the redesign by first adding all missing functionality to the exist GUI and testing that functionality. Next we will add safety measures to the CLI that is already in place to process commands. Finally, we will start the redesign starting with the controls graph that is currently a PNG and we will add more functionality to detect the PID values from information passed back and design an interface to select and change the values. The backend will be cleaned up and give more response to the user. And the Navigation screen will be added to with more options for demos, new options for navigation changes, and support for real-time graphing of quadcopter data.

3.3.2.3 Multiple Object Tracking Capabilities

The VRPN system already allows for tracking multiple objects and has a socket open to the backend to pass information to the Ground Station. We will have to make sure that information for objects is passed and parsed properly and setup the x,y,z information to be passed to the quadcopter. This would have to go in the already designed packets for reliable communication, but differentiated from the quadcopter coordinates.

3.3.2.4 Data Analysis Tool

We will create a stand alone tool to review data post flight logs. This will be a more complete solution than the current implementation within MATLAB. This tool will use the same logging format but will also be able to visualize data in real time with the ability to add different variables on and off.

3.3.2.5 Generic Object Integration to Backend Capabilities

We will create a new system to allow for integration of generic vehicles to our system. Essentially whenever future teams want to add a new type of vehicle to the ground station software they will follow a simple protocol to do this. The protocol calls for a translation layer that sits between our current backend and any of the new generic vehicles' software. The translation layer will take commands and translate them to whatever the new vehicle software needs to execute the functionality of the command being sent from ground station. This protocol will act essentially as a checklist for implementing new vehicle hardware.

3.3.3 Control Systems

3.3.3.1 Improved Stabilization

The existing controls are stable up to small oscillations of no more than 5 cm, so the first step to improving stabilization will be identifying the largest sources of error in the system. This could be in any of the controls model, the sensor input, or the physical variance of a motor or environment.

3.3.3.2 Advanced Control Maneuvers

The challenge of advanced control maneuvers is that they would put the quad at an extreme angle. In doing a flip, for example, the quad must past through at least two positions with singularities in its Euler angles (in the base inertial reference frame used). We will need to either take a different approach for handling orientation or manage when orientation values are updated such that we sample around the problematic points.

3.3.4 Hardware

3.3.4.1 Second Quadcopter

In building the second quad, we will improve on the current quadcopters design, both for improvement and out of necessity. We will be improving the mounting of components in order to produce a more consistent quad. We will also need to change the design due to the availability of components.

3.3.4.2 Power Regulation Board

The current power regulation board is an individual component used to convert 12.4 volts from the lipo battery to 5 volts which powers the Zybo board. The quad also uses a power distribution board. In creating a new power regulation board we would like to combine these two components of the quad into one board, simplifying the wiring and design of the quadcopter.

3.4 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.

3.5 Validation and Acceptance Testing

Most functional requirements will be validated by using the newly proposed virtual quadcopter simulator and by experimentation with the actual quadcopter in Coover 3050 using VRPN. In the later stages of the development, we might test the quadcopter outside of its VRPN area, especially for optical flow testing.

3.5.1 Quadcopter Software

Test that all of the quadcopter software components work as desired in all the different modes of flight (semi-autonomous, VRPN, VRPN+Lidar, OF) by receiving appropriate commands from the Ground Station. Virtual Quadcopter Simulator will be used to run sanity checks (e.g. quadcopter motors cannot run with the kill switch on) as well as simple virtual flight tests.

3.5.2 Ground Station

The ground station will primarily tested through use and performing test flights. In these flights the user will attempt to send coordinates that could harm the quadcopter to insure the Ground station is correctly handling the different situations. While doing these test flights navigation modes will be switched between and different demos will be run to insure correct operation from the Ground station side. Only if these tests can be passed will the Ground Station GUI and other components be fully rolled out for use by all the team.

3.5.3 Controls Model

The controls will need to be verified mathematically before being tested. Once it has passed that phase, it will go through the two forms of testing mentioned above: first through the quad simulation and then with testing on the physical quad. Continuous yawing will be tested by continuously yawing the quadcopter around the current critical region (+-180 degrees).

3.5.4 Testing Framework

The testing framework should provide an automated test environment with a final report that runs test scripts specially written to test different parts of the quadcopter software. Committing a file into a repository should trigger a git commit hook to run an appropriate test. We should be able to see if a test was ran in the Gitlab report section.

3.5.5 Documentation

The documentations is easily readable by someone who is not familiar with the project. Documentation follows our documentation standards set by Dr. Jones. All documentation should be reviewed by another member of the MicroCART team who is not affiliated with the group who wrote the documentation to assure high documentation quality.

4 Project Requirements/Specifications

4.1 Functional

Because this project is being built from an existing base, many basic functional requirements are met. The following requirements listed below will be focusing on new functionality added to the the existing base. This section again will be split into the three subsections referenced in the deliverables above.

4.1.1 Quadcopter Software

- Quadcopter shall be able to fly in stationary waypoint mode where the quadcopter stops at each waypoint before continuing to the next one.

- Quadcopter shall be able to fly in continuous waypoint mode where the quadcopter approaches adjustable threshold and without slowing down continues to the next waypoint, thus allowing for a smoother flight.
- Quadcopter shall send its flight data back in real time to the Ground Station.

4.1.2 Ground Station

- Does not allow the user to put in coordinates that can cause harm to the quadcopter.
- Allows the user to switch between different navigation modes.
- Allows the changing of PID values in the controls flow diagram.
- Supports graphing of the flight data in real time.
- Does not allow changing of PID values or navigation modes while hovering off the ground.
- Contains interfaces to display information relevant to new demos.
- Supports sending of coordinates of a trackable object to the quadcopter.
- Supports sending of commands to change the PID values and navigation modes to the quadcopter.
- Support real-time transmission of flight data to backend.

4.1.3 Controls Model

- The deviation from a hovering waypoint shall be less than 1 inch laterally and vertically in the static waypoint mode.
- The controls model shall allow for continuous yawing, that is more than 180 degrees in each direction.
- The controls model shall allow for wider range of operation to allow for non-standard autonomous flight regimes such as flips

4.1.4 Testing Framework

- Testing framework shall allow for running unit tests designed for testing different parts of the MicroCART platform.
- Each test shall have its own report of results
- The testing framework shall have an overall report after running all the continuous integration tests–“master run”
- Testing framework shall be integratable into the git hooks to run all the related tests for each repository commit.

4.2 Non-Functional

The non-functional requirements fall into three primary categories: documentation, performance, and testing.

The purpose of documentation, as already given above, is to improve maintainability of code, both for our immediate usage and for future semesters’ MicroCART team. The non-functional requirements are that the project documentation must be

- Complete
- Accurate
- Understandable

- Easy to Update

Performance is important to the purpose of the project: in order to provide a useful controls research platform, all other components of the system must be reliable and providing good data. For example, we want to improve the reliability of the optical flow position and velocity data. Doing so will provide any future controls algorithms with more accurate data, thus positively impacting the effectiveness of the research. In summary, the non-functional requirements for performance are that all components need to act with

- Robustness
- Speed (low latency)
- Accuracy (and precision)

Testing is largely described in a later section. To summarize, we want to have a continuous integration testing system in place to perform regression tests on each build of the project. These tests should be able to verify that new changes preserve all old functionality and do not introduce new bugs into previously completed areas. To list the non-functional requirements, the set of tests should be

- Complete
- Rigorous
- Modular
- Maintainable
- Scalable

4.3 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. We will also need to adhere to Iowa State's policies if we do any flight testing on campus outside of the lab. Currently, research is exempt from having to adhere the the Unmanned Aircraft Systems (UAS) policies [3]. If we do flight tests on campus and outside the research lab we will also have to adhere to all Federal Aviation Administration (FAA) regulations that pertain to unmanned aircraft systems [4].

5 Test Plan

5.1 Quadcopter Software

We will conduct three kinds of tests on the quadcopter software. The first kind are unit-level tests. The unit-level tests will be written by one of the quadcopter software group members and peer reviewed by another member. These unit-level tests will be run by our testing framework. Our virtual quadcopter simulator will run another series of tests to make sure that all of our automated flight scripts are performing nominally. Once confirmed in the simulator, we will proceed with the actual flight test in Coover 3050 lab.

5.2 Ground Station

The ground station testing will be also performed using virtual quadcopter simulator at first and then with an actual flight test. All the features of the ground station will be verified for nominal operation using a series of peer reviewed test scripts.

5.3 Controls Model

We will conduct experiment to verify the chosen controls model parameters using logged data flight analysis, as well as performing tests on a stationary zero friction rig for our quadcopter. The zero friction rig will allows us to verify step response of the quadcopter controller in the real world. The rig test should give us more accurate parameters to reach the one inch accuracy laterally and vertically.

5.3 Test Framework

We will use the Unity test framework which is already tested by its designers. For more information visit http://www.throwtheswitch.org/unity/. Our test scripts will be peer reviewed by at least one other person than the creator to ensure correctness.

5.4 Documentation

Documentation will also be peer reviewed to make sure it complies with Dr. Jones' standards for MicroCART. By following the standards, it should be possible for future MicroCART teams to understand the entire system faster and easier.

6 Assessment of Proposed Solutions

6.1 Quad Software

Improving autonomous flight will provide us with increased versatility while providing better overall functionality. with the tradeoff of abstracting user control and user input. Real-Time Flight Data Communication will allow us to better analyze in real time while providing additional room for data to be stored. But with the tradeoff of adding overhead to send data back to the ground station over wifi. GPS Navigation will enable a more realizable control algorithm, introducing more reliable positioning at the cost cpu utilization. Linux on Second Core will improve overall performance by enabling additional functionality such as interrupts and threading. Assisted Manual Flight ("Trainer Mode") will allow a person to fly the quad with little-to-no experience in flying quads to have an easy way to fly the quad without having to worry about damaging the quad. This assisted manual mode will provide confidence in someone's ability to fly.

6.2 Ground Station

Real-time transmission to backend will come at the cost of increased amount of throughput required by our WiFi system. This may also lead to an increased latency as a result. That being said the team that implemented the Wifi communication originally planned for a latency of under ten milliseconds and the system averages two milliseconds currently. We plan for this increase in latency to still remain under ten milliseconds. As such, we will have to thoroughly test and see what our limits are due to hardware. We may have to make some sacrifices in the control loop on the quad to make sure our quad can transmit in real time.

Redesigning the GUI involves building upon the existing QT project to add the new functionality. This makes the process less time consuming as we can look at their current methods of interfacing the frontend to the GUI to add our new features. Additionally, our new areas will make the system easier to use as the cost of added complexity into the GUI. With increased documentation this should make it easier to understand this now more complex area of development.

Both multiple object tracking capabilities and generic object integration into backend have many of the same issues. We plan to keep existing functionality at the cost of more work to track additional objects and integrate generic objects. This will lead to those that want to use our system to have to go through a longer development cycle to properly integrate their tools into our backend. As it stands the backend and ground station code as a whole is less complex than many of the other areas as it has a layered approach. Our solutions will increase the complexity, particularly within the backend files as supporting more than just our quadcopter will require more parsing, accepting additional commands, and expanding upon existing packets.

We plan on reusing previous teams data analysis tool in our new and improved tool. This will come at a cost of potentially more time if we have a hard time reimplementing their code for our purposes. This may also come at the cost of less customizable data analysis too.

6.3 Controls

The central component of the proposed controls solution is a linearized mathematical control model to replace the existing nested PID approach to in-flight control. The new model would rely significantly on a physics-based parameterization of all components and of the system as a whole. One disadvantage of this is that the system must be re-parameterized every time changes are made to the hardware. There is also the disadvantage that measuring all the different parameters is not trivial, and a different set of parameters will be required for each physical vehicle. On the plus side, this model has the potential to be far more accurate. The approach of a PID controller is to guess output values for a given input and iteratively correct for error, while a linearized mathematical model would provide a precise result with a single calculation, given that the input is within the supported region.

6.4 Hardware

Our plan to build more quads will be costly but will increase redundancy of the project. It will allow eventually have two or more of our current quads running at the same time without the fear of breaking the one and only quad.

For ease of use we plan on adding a power regulation board that will allow us to debug and test without our short lasting battery. this board will be only a minor financial cost and time sink to our team and will add only a minor amount to the overall weight of the quad.

6.5 Documentation

We are adding documentation to help next year's team get up to speed faster and rewriting existing documentation to make it more readable and easier to understand. This will add additional files to the repository which have the potential to make navigation more difficult if the documentation is not organized.

6.6 Demos

The demos that we plan to implement this semester are planned to both impress viewers as well as show existing functionality. Tracking objects and flying two drones together allows for more complex demos with objects such as balls or remote controlled cars, and doing much intricate demos with more than one object flying. This also accurately shows our new functionality as the additions to the backend will allowed increase controls, quadcopter software and controls will increase the safety of having two flying objects, and our additions to testing will allow us to further test before putting our quadcopters in harm.

7 Challenges

7.1 Risks

Throughout the year, as we work on improving the quadcopter's autonomous flight capabilities, we will face a multitude of risks. Most of these risks revolve around the potential destruction of the quad. The quad involves several moving parts, all of which are fragile. Every time we compile a change in the code and load the software onto the quad, we will face a risk that we changed something that was essential to having a stable flying quad. To minimize this risk we will be running tests on the virtual quad each time that we compile and merge our code onto the repository. We will also be subjecting our code to several static tests that will check modular sections of code based on the inputs and expected outputs.

Another risk that we will be taking is the risk of changing code that we are not completely familiar with. As we have divided up into teams that will work on different sections of software, it is important that we know the repercussions of changing any of the code. If we change one small piece of code, it could change some data that another team was expecting as an input, which could have catastrophic results. To minimize this risk we will all be sure to have at least a basic understanding of all aspects of the project, so that we know what specific parts of the project will be affected by each change we make.

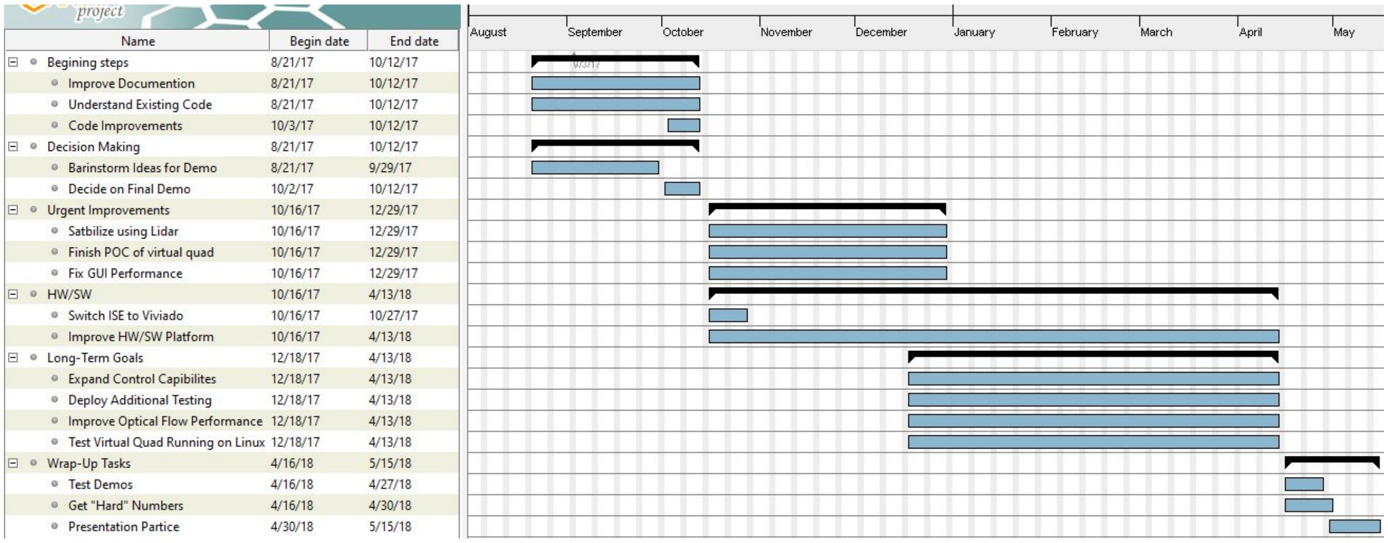
7.2 Feasibility

We already know that it is completely feasible to make an autonomous quadcopter using the VRPN camera system. As part of our project we would like to make the quadcopter autonomous outside of the camera system, using the optical flow sensor. We believe that this is feasible with the addition of some locating sensors, such as GPS, and with the incorporation of the accelerometer and gyroscope. Once we can accurately get the position and movement data from the quad, we should be able to achieve any simple tasks (demos) that we set out to achieve.

7.3 Cost Considerations

Part of our project consists of constructing one or two more quadcopters. This will consist of purchasing new parts to use in the construction of these quads. While we have been given approval by our advisor/client to order these parts, it will be vital that we do not blindly order parts without looking at the cost. If one part costs what is considered to be much more than normal, then we may have to look at building the additional quads in an alternative manner. For example, the motor speed controllers that are used on the current quad are hard to find and are expensive when they can be found. Therefore, we have chosen to use a different speed controller for the motors on the other two quads. We will also need to minimize the damage done to the quads so that we are not spending a lot of money on extra parts. The costs are looking to be a couple hundred dollars for the frame and the propulsion system. The Zybo board will cost about two hundred dollars. The other big costs will be the lidar and optical flow sensors. Both are around one hundred dollars each. The only other costs we should incur are the costs of batteries for the controller and for the quads themselves. We estimate the total cost for the construction of the quad to be about $1000.

8 Timeline



Name	Begin date	End date
Begining steps	8/21/17	10/12/17
Improve Documention	8/21/17	10/12/17
Understand Existing Code	8/21/17	10/12/17
Code Improvements	10/3/17	10/12/17
Decision Making	8/21/17	10/12/17
Barinstorm Ideas for Demo	8/21/17	9/29/17
Decide on Final Demo	10/2/17	10/12/17
Urgent Improvements	10/16/17	12/29/17
Satbilize using Lidar	10/16/17	12/29/17
Finish POC of virtual quad	10/16/17	12/29/17
Fix GUI Performance	10/16/17	12/29/17
HW/SW	10/16/17	4/13/18
Switch ISE to Viviado	10/16/17	10/27/17
Improve HW/SW Platform	10/16/17	4/13/18
Long-Term Goals	12/18/17	4/13/18
Expand Control Capibilites	12/18/17	4/13/18
Deploy Additional Testing	12/18/17	4/13/18
Improve Optical Flow Performance	12/18/17	4/13/18
Test Virtual Quad Running on Linux	12/18/17	4/13/18
Wrap-Up Tasks	4/16/18	5/15/18
Test Demos	4/16/18	4/27/18
Get "Hard" Numbers	4/16/18	4/30/18
Presentation Partice	4/30/18	5/15/18

Figure 5: Project Timeline

Task	Assignee	Man-hours
Improve documentation	All	80
Understand existing code	All	40
Code improvements	All	30
Brainstorm Ideas for demo	All	14
Decide on final demo	All	8
Stabilize using lidar	Austin & Blake	60
Finish POC of virtual quad	Peter & Jakub	80
Fix GUI performance	Dane & Matt	120
Switch ISE to Vivado	Kyle & Tyler	20
Expand controls capabilities	Austin & Blake	180
Deploy additional testing	Peter & Jakub	160
Improve optical flow performance	Kyle & Tyler	140
Improve Virtual quad performance on Linux	Dane & Matt	100
Test demos	All	40
Get “Hard” numbers on performance	All	20
Presentation practice	All	16

8.1 First Semester Breakdown

In the first semester our primary goal is to get the well acquainted with the current platform the quad is running on. To do this everyone must understand the core portions of the project. In order for this to be successful we must minimize the impact we have on the build and strictly focus on meeting specifications on documentation. Towards the end of the semester, once we already understand the quad, we can start to work towards our goal of providing an impressive demonstration of capability for the quad.

8.2 Second Semester Breakdown

In the second semester we will be better equipped to parallelize work in the different areas of the project. We will accomplish these goals by splitting into small groups and testing separately, while merging and testing the system as a whole as frequently as possible.

9 Conclusions

This year's MicroCART team will produce a more stable flying autonomous quad for both flying within VRPN camera system and when using the optical flow sensor. We will be building additional quads for graduates students and our own team members to use for research and improvement of the MicroCART software and hardware implementations. We will improve on the existing GUI for the ground station while adding new features to improve the ease of use. Documentation will be brought up to date as we work to understand the existing quad and will continue to stay up to date as we make improvements to the system. As requested by the client, we will be working towards a goal of having a demo for the end of the year that will show off what the capabilities of senior design students in the ECPE department.

10 References

1. Research UAV – Drones / UAS for Research & Development. (n.d.). Retrieved September 24, 2017, from http://www.asctec.de/en/asctec-research-uav/
2. Rich, Matt. *Model development, system identification, and control of a quadrotor helicopter*. Diss. Iowa State University, 2012.
3. Facilities and Grounds Use, Activities. Iowa State University. (n.d). Retrieved October 29, 2017. from https://www.policy.iastate.edu/policy/facilities/use#UAS
4. Unmanned Aircraft Systems. FAA. (n.d.). Retrieved October 29, 2017, from https://www.faa.gov/uas/