MicroCART

DESIGN DOCUMENT

Team Number: 50
Client: Dr. Phillip Jones
Advisers: Matt Cauwels, James Talbert
Team Members:
Evan Blough -- Technical Team Lead, Embedded Software Lead
Kynara Fernandes -- Ground Control Station Lead
Aaron Szeto -- Controls Lead
Joe Gamble -- Embedded Hardware Lead
Shubham Sharma -- Crazy Fly Implementation Lead,

Executive Summary

Development Standards & Practices Used

1. Follow the Principles of Programming by adhering to IEEE software development practices

- IEEE 1233-1996 IEEE Guide for Developing System Requirement Specifications
    - Used for determining system requirements in relation to client needs and functional/non-functional requirements
- IEEE 12207-2017 International Standard: Software life cycle processes
    - Used for involving our stakeholders in the development process and production of maintainable software.
- IEEE 1008-1987 Standard for Software Unit Testing
    - Used as a model for developing unit testing for our control software
- IEEE 802.1-2010 Standard for Port-based network controls
    - Used as a model for communicating with the drone

2. IEEE Code of Ethics provided by the IEEE organization

Summary of Requirements

List all requirements as bullet points in brief.

1) Integration of a **secondary** quadcopter into the high-speed camera system,
2) Extend modular design to enable testing advanced controllers and signal processing,
3) Design and develop capabilities to support the implementation of the real world application by improving the GCS functionalities
4) Design and implement a mock real-world demo for visitors.
5) Integrate Crazyflie platform into system

Applicable Courses from Iowa State University Curriculum

List of Iowa State University courses which were applicable towards the MircoCART project:

- CPRE 288:
    - This course:
        - Goes over elementary embedded design flow/methodology
        - It gives students a basic understanding into micro-controllers.
        - Applications laboratory exercises with embedded devices.
    - We work with microcontrollers a lot with the project as evident from the project title. Hence, this course is a great foundation to assist us in the project.
- EE 230:
    - This course:
        - Is an overview of circuitry design and analysis
        - Gives students experience working with laboratory instrumentation and measurements
    - This is an applicable course as the project consists of circuit design knowledge from 230.
- CPRE 381:
    - This course:
        - Is an Introduction to computer organization, evaluating the performance of computer systems
        - Datapath and control, scalar pipelines, introduction to memory and I/O systems
    - The project consists of ARM-processors we would need to work with. And since this course is centered heavily around processors it is helpful towards the project.
- EE 330:
    - This course:
        - Goes over semiconductor technology for integrated circuits.
        - Analysis and design of analog building blocks.
        - Laboratory exercises and design projects with CAD tools and standard cells.
    - The project would require an understanding of circuit issues such as unmatched impedance and circuit devices such as current mirrors. Which is why this course is applicable for the project.
- EE 224:

    - This course:
        - Introduction to using Matlab for EE work
        - Analysis of Signals and various signal systems like LTI
    - This class teaches how to use Matlab and quadcopter controls are entirely based in Matlab hence it is helpful for the project.
- EE 321:
    - This course:
        - Continuation of EE 224
        - Focuses more on modulation and data transmission
    - Focuses on data transmission which applies to how the quadcopter communicates with the controls and ground station, which is heavily used in the project.
- CPRE 488:
    - This course is about:
        - Embedded microprocessors and embedded memory
        - Component interfaces communicating to embedded software
        - Platform-based FPGA technology w/ hardware synthesis
        - And Real-time operating system concepts
    - This project requires knowledge with VHDL and C development on Xilinx platforms, multidisciplinary projects, embedded system design, and control systems on drones. Hence, why the course is applicable to the project.
- SE 339:
    - This course:
        - Software architecture.
        - Uses a raspberry pi to simulate a fleet tracking application.
        - Teached about how different components in a project should communicate and the best practices to do so.
    - This project requires knowledge of engineering and software architecture. Since various components of this project must communicate with each other like the camera system, the drones and the ground station. This course is applicable to this project in choosing the best architecture for our system

New Skills/Knowledge acquired that was not taught in courses

Qt software libraries - Qt is an open source GUI development framework built to run on C++ applications. Qt was used to create our graphical user interface to meet the needs of our research/development users.

RC communication - RC receivers and transmitters use defined method of communication. Our application interfaces with an RC receiver. The receiver puts out 6 channel PWM signals. These signals are used to control drone behavior.

PID Controls - PID Control in not generally taught in depth without taking specific controls classes. PID control uses feedback to account for present, past, and anticipated error. These errors are assigned different weights depending on gain values, and the overall output to the actuators is recalculated. PID is one of the control algorithms our drone platform uses to correct its position.

Table of Contents

List of figures/tables/symbols/definitions (This should be similar to the project plan)

Term	Definition
CLI	Command-line interface
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
FPGA	Field Programmable Gate Array; this is a board that can be programmed to simulate electrical hardware components. Used often in reconfigurable computing
GCS	Ground Control Station, the application that runs on a host computer that communicates with the quad via a Wi-Fi connection and sends its 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	A 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 the 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 (φ,θ,ψ) of the quad in the lab using a set of 12 stationary cameras and an IR transmitter on the quad
Shield Board/Breakout Board	PCB board designed by a previous MicroCART team that integrates with the Zybo board to interface with sensors and actuators used for autonomous flight.

1 Introduction

1.1 Acknowledgement

The development of this project has been aided by ECPE Faculty and graduate students at Iowa State. They offer valuable technical advice and insight. We would like to acknowledge their contributions to this project below.

- Matthew Cauwels
- Dr. Phillip Jones
- James Talbert
- May 2020 MicroCart Team

1.2 Problem and Project Statement

1.2.1 Problem Statement

MicroCART is a drone platform that will be used by graduate students to perform research on embedded systems and controls topics. The platform will also need to demonstrate its operation to professionals or future students to showcase the utility of skills obtained through the course of a EE/CPRE/SE degree. This platform will also need to be maintainable and modifiable by future senior design teams. Currently, the MicroCART platform is functional, and it will be extended to a copy of the current platform to implement swarm flight. In order to achieve these objectives, we will need to copy, refine, and modify the current platform to have more intuitive interface and more accurate flight performance.

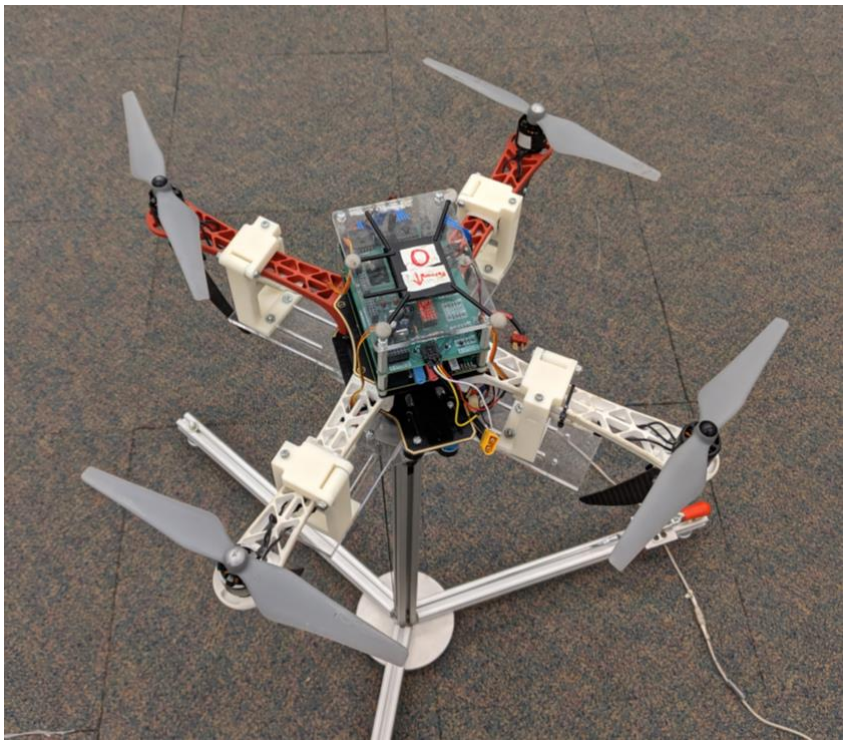


Figure 1.2.1: Current drone implementation

1.2.2 Solution Approach

We plan to further our knowledge of the previous development done on the project. We will need to read current documentation of the drone's subsystems and infer the functionality of portions that aren't documented. We also elected focus areas for each team member to specialize in areas of drone development. Once each of us are familiar with our respective subsystems, we can start building the second platform.

There is a smaller drone platform that we can use to familiarize ourselves with navigation. We hope to implement swarm flight with these smaller drones, and once we have experience with the smaller drones, we will be able to move on to navigating the larger drone. By the end of the first semester, we hope to have both large drones operating.

The second semester would then consist of us adding new features to the drone to fulfill the purposes outlined above. The controls and embedded software leads will work on making more intuitive source for swapping controls algorithms on the drone. The GUI leads will be working on creating an easy to use interface for the drone. The embedded hardware will generate fast feedback loops to make the control systems more efficient and regulate memory use for each control algorithm. The whole team will work on implementing swarm flight. As a team we might work to add Linux to the second core to promote development on our platform. This second Linux core could have high level C++ libraries like OpenCV that could be used for soft real time object tracking.

1.3 Operational Environment

The operational environment for the quad is the inside of Coover 3050, and this room has a VRPN camera system that can be used for drone navigation. For indoors operation there will not be any hazardous environmental factors like extreme heat or cold. The environment in the lab will have people in it, and the drone could collide with expensive equipment in the lab, so failsafe for flight faults are important considerations for our operating environment. Due to the fact that the operation environment is identical to its use environment, we expect little design difficulty caused by the operational environment

1.4 Requirements

The functional requirements for this project are building our quadcopter, get it flying at the same time as another quadcopter, and modifying the controls system to be compatible with other control systems as seen in [1]. Some economic/market requirements include last year's quadcopter assembly materials, quadcopter replacement parts, and spare Lithium Ion batteries. The environmental requirement is that the quadcopter should be able to fly in an indoor environment. UI requirements include making the control systems more user friendly. The non-functional requirements for our design are as follows: improving the usability of our GCS, improving drone flight responsiveness/latency, enhancing flight accuracy of the drone during autonomous navigation.

1.5 Intended Users and Uses

The platform will be used to test different control systems and their varying efficiencies for students. Students will require quick, simple interfaces to maximize learning and reduce time use for lab. The system will also be utilized in future senior design projects, requiring our team to practice good documentation and written communication techniques. Finally, the project will be utilized in college demonstrations as a means of showing off to our employers what our college can do as well as an attempt to recruit future students.

1.6 Assumptions and Limitations

1.6.1 Assumptions

- We won't have more than two large quadcopters in our operating environment
    - Further maybe hazardous/ too complicated
- The operating environment indoors doesn't contribute any significant hazards
    - Humans acting in a hazardous manner might complicate this
- We assume that standard components we buy from manufacturers come in working condition
    - This assumption may not be valid, but will save time for excessively testing every single electrical component

1.6.2 Limitations

- Drone flying environment limited to the area of the camera sensors within Coover 3050
    - The flight is dependent upon cameras tracking drone
- The computational power of the drone
    - Limited in processing time
    - Limited in memory access speed
- Feedback delays in control systems by the camera systems
    - Delay in ms expected
    - From Wi-fi communication
- Battery to payload ratio imposes limitations on overall runtime
    - Battery technology and weight characteristics of drone limit flight to appr. 5 minutes
    - Frequent replacement of battery systems necessary

1.7 Expected End Product and Deliverables

Our product is academic in nature, so we won't have commercialized descriptions. We will need to provide detailed documentation through gitlab documentation and documentation generators like doxygen.

1.7.1 Implementation of Second Drone

We are going to make an exact copy of the current drone platform from the previous year. Most of the work done on this portion will be in the form of research on the existing system. In order to make another platform we will need to research the existing documentation. We are making this deliverable, because we need to have one stable platform to demo our solution to

people and one for a senior design team to do development on. This deliverable should be completed by October 25th.

1.7.2 Swarm Flight of Crazy Flies

After we have a second drone, we will hopefully have access to several crazy flies. We can use these platforms to test out extending our GCS software to multiple platforms. This is an intermediate deliverable. We will use this deliverable to get experience with programming swarm flight with inexpensive platforms and move on to our permanent solution for swarm flight. We should have this portion completed by the end of the first semester or the beginning of the second semester.

1.7.3 Swarm Flight of large drones

After we have experience with swarm flight on the crazy flies and have the second drone built, we will work on having two drones operate simultaneously. This deliverable relates to our client need of demoing to students and faculty. Showing multiple vehicles operating simultaneously is more impressive than just a single vehicle. This showcases the usefulness of skills obtained with an ECPE degree. Controls students also will want to test control algorithms with multiple platforms interacting. This should be done by midway through the second semester.

1.7.4 Addition of Linux to Second Core

After we have both drones fully implemented and functioning properly, we will need to add new features to our board to showcase skills that we have learned from our degree. One of the features we plan on adding is Linux to the second core. The Zybo board has two ARM cores. We will install Linux on the second core and this will give researchers and future project teams greater flexibility to control our platform. We can also use it to view the value of a degree. We can install an image detection library like OpenCV and a camera on Linux and use image detection to give control commands to the drone. This will be done at the end of the second semester of this course.

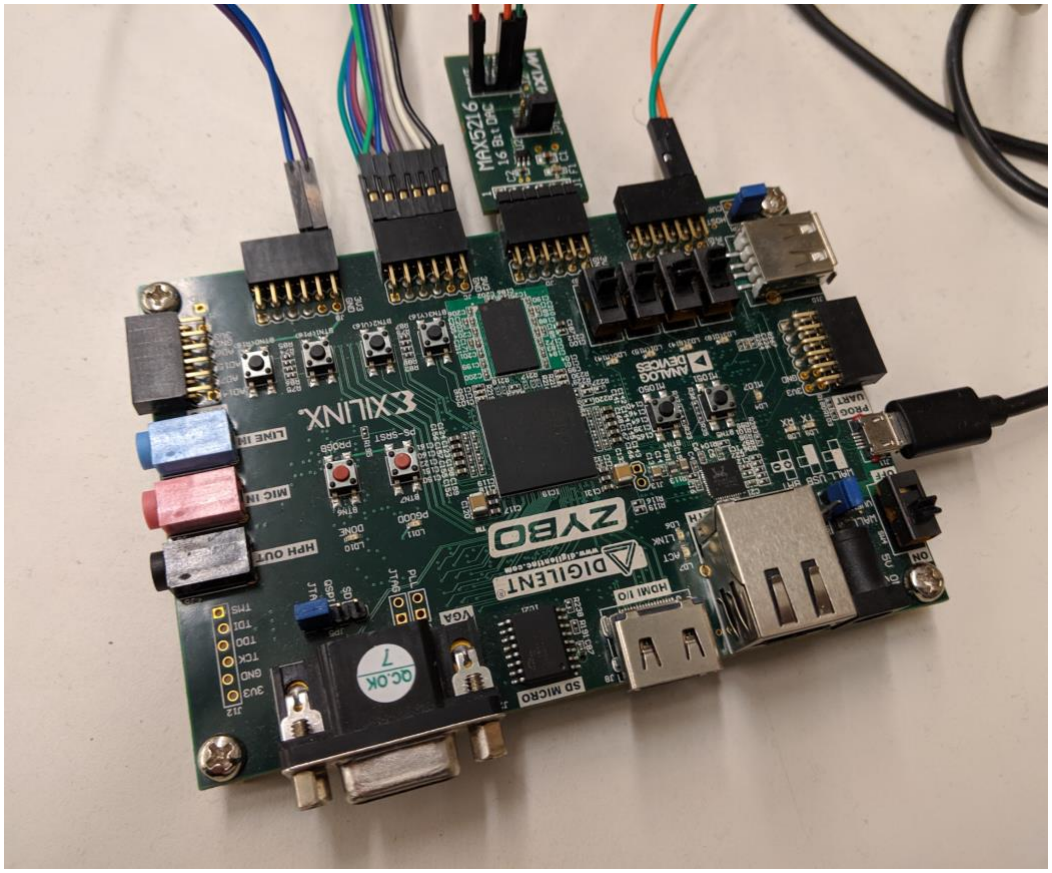


Figure 1.7.4: Zybo platform for drone development

1.7.5 Modular Control Algorithm Swapping on the Drone.

Another feature we plan on implementing is the modularity on control algorithms. Currently, the drone control program runs a PID based position correction algorithm. It measures the differences between the set points and the actual position of the drone that is captured.

1.7.6 More Interactive and Comprehensive UI

When talking with the current users of our project, we found that it would be helpful to create a more interactive element for the data from the quadcopter, camera system and ground station. We will do this by adding new CTEs, sliders and graphs to help visualize the input, output and calculated data.

2. Specifications and Analysis

2.1 Proposed Design

So far we've started looking at the documentation and getting familiar with previous years work. We've also talked with our client about what's expected of us and to help us plan out what we want to do with the quadcopter. To approach solving these problems we are using the Agile method

to coordinate our coding work and the discord application to coordinate in-person meetings to work on the hardware aspects of the quadcopter.

2.2 Design Analysis

During the beginning design portion of the project we have worked to understand the current implemented system. Each component of the system has been built over many years and passed through many teams. It takes time to understand each portion of the system to an extent at which it can be utilized; therefore, the majority of the time has been spent reviewing documents.

Our ability to process the current documents has progressed well. We have begun to run the drone independently and some initial work. Each person on the team has demoed the drone and has read documentation. We have begun to complete initial work including soldering, control system analysis, gui interfaces, and initial design document creation.

Strengths

- Team comradery and communication
- Good mix of disciplines w/ large knowledge base
- Contacts with previous MicroCART users to answer questions
- Large group to distribute work

Weakness

- Large number of systems with intricacies
- No control theory knowledge base
- Poor documentation
- Limited timeframe to implement

2.3 Development Process

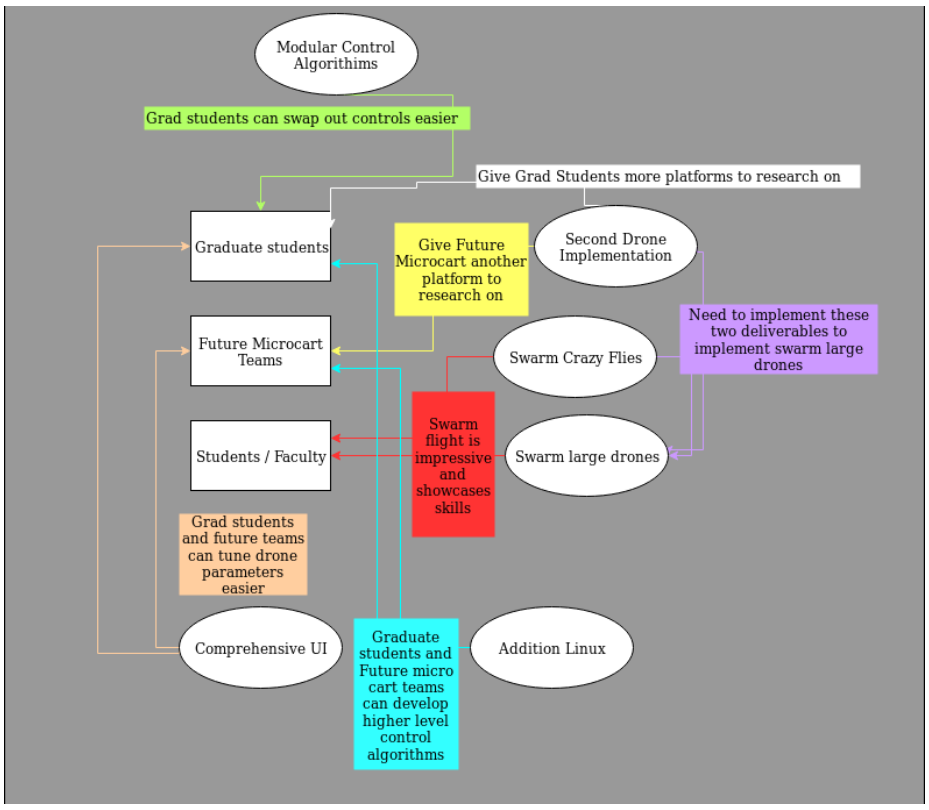
After being assigned our project, we met with our customer and advisor to discuss the requirements for our project. Since we're working on an embedded systems project, our system has multiple components that would have different requirements that would change as we move through the school year.

We are currently in the design phase of our project. We are researching materials like control theory etc. to help us understand our customer/users better and in turn, produce a better tailored product.

With these two phases, we are currently in the waterfall process. But since our requirements will change during the phase of the year, we are transitioning into an agile development process. The plan is as such:

- **Tickets:** We are currently using Trello to handle our tasks and keep track of the tasks that need to be done, the tasks that are in progress and our accomplished tasks.
- **Meeting with stakeholders:** We meet with our customers, users, and advisor once a week on Mondays to discuss our accomplished tasks and the concerns or goals they have for the coming sprint or sprints.
- **Sprints:** We have 1-week sprints in which we will go through one iteration of standup, grooming, retrospectives and stakeholder meetings.
- **Grooming:** After our stakeholder meeting, we regroup, and create and prioritize tickets on Trello for the new sprint.
- **Stand-up and Retrospective:** We start our weekly team meetings with scrum, where we talk about what we did, what we plan on doing and if we had any blockers for that sprint. Since we are full-time students, we compromised on doing this weekly instead of daily.

2.4 Design Plan



3. Statement of Work

3.1 Previous Work And Literature

The MicroCART Quadcopter is an ongoing project at ISU that has been worked on by generations of students. Previous teams have already built a quadcopters drone, and we will use their drone as starting point for how we want to build and modify our own version of the quadcopter. An advantage of having the previous year's drone with code available, is that it gives us a great idea of what we're trying to do and how to go about achieving it. A disadvantage is that we can't follow previous years example too closely because some of their code doesn't work. Our team has already encountered several examples of broken code that previous teams have uploaded to the github.

3.2 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

Strengths

- High performance processing
    - Complex control integration
    - Visual data computation
    - General quickening in data processing
- New batteries with improved battery life and energy densities

- High performance motors
- PCB and CAD designing tools
- Large software libraries
- Open source material
- IMU sensors with high accuracy positional data
- Camera system with high accuracy positional data

Weaknesses

- Cost of new material and parts
- Weight added to quadcopter resulting in lower battery life and risk of loss of flight
- Increased dependency on a large volume of different technologies results in a convoluted mashing of technologies that is not digestible by future teams/students

Trade-offs (Summary)

- Cost and complexity for more advanced solutions with return on drone abilities
- Usage of new technologies tends to increase the abilities, but increases weight, power, and generally reduces performance of the drone

3.3 Task Decomposition

3.3.1 Drone Construction

Leveraging of current drone hardware buildup and extra parts to create a new, fully functioning drone. The drone buildup will require the use of knowledge in electrical hardware skills and some mechanical understanding. The information on how to build-up the new drone will come from observations of the previous drone.

3.3.2 Controller Swapping

Current drone software performs calculations to determine motor output using a PID scheme. The desired outcome is to make the scheme swappable. Effort needs to be put forth into understanding how the current implementation works and adding commands that are usable across all control schemes. An understanding of how to construct these control methods must also be done.

3.3.3 Image Data Processing

A second linux core is being added to the device to process image data from cameras not currently on the system. We then want the libraries on the linux core to support a C++ image processing library like OpenCV. We will have to find an attachment point for the camera and develop and electrical interface, which could be I2C or USB.

3.3.4 Multi-Drone Flight

The new drone built from the drone assembly process will be flown in tandem with the current working drone. The crazyflies in development are also going to be flown with the large drones. This will all require code for the drones to recognize each other's position and ensure no collisions occur. We will also have to modify the current networking scheme. The GCS and drone

interact only with each other, so to add multiple drones we will need to reconfigure the networking scheme to an appropriate type [2].

3.3.5 Organizing UI and Git

Much of the work that has been done over the past few years has been somewhat organized by the many hands the work has passed through. We will work to organize the hardware components, software code, git, and help-documents.

3.4 Possible Risks And Risk Management

3.4.1 Equipment Risks

There are several things that may impede our progress on this solution. The equipment that we were given to support software development on our platform is very dated. We were supplied one 15-year-old computer that did not support c++ development, which is what our ground station was built with. This has already impeded development progress and may continue to slow our implementation time in the future. The other working development computer had several networking issues that were introduced by integration with the VPRN camera system. Some of the components that are supported by our previous teams are very old. The age of the sensors can introduce development problems.

3.4.2 Knowledge Area Risks

There are several development areas that could present risks because we have a lack of knowledge in these areas. The swarm flight requirements introduce some significant variation to the networking scheme of the current system. The current system has the ground station machine connect to a wifi network created by a small wifi microprocessor chip on our flight control board. This introduces a risk to our team because in the new configuration this configuration is not viable. We also have limited knowledge of methods for implementing a new networking scheme with the existing hardware. This will likely impede development progress at some point.

The implementing Linux on the second core feature has some knowledge area risks. We don't have knowledge of how trying to implement this feature will affect our latency in communications with the control station. We will have to thoroughly research and test to ensure this feature implements itself without compromising the safety-critical feedback systems.

3.4.3 Costs Risks

We have several dependent hardware components that we need to buy to support our drone applications. Buying these components impedes our progress because we have to go through a purchasing procedure and justify our need for these components. Some components may break or malfunction, so we will have to buy new ones. We have to buy custom circuit boards, RC controllers, Lithium Polymer batteries, etc. We have to wait for these things to be purchased in order to develop.

3.5 Project Proposed Milestones and Evaluation Criteria

Some key milestones for our project are

1. Having multiple autonomous Crazy Flies controlled with our GCS software
2. Build our own Quadcopter drone
3. Swarm flight with the Quadcopter
4. Implementing controls algorithm swapping
5. Implement Linux to second core
6. Improve UI and Github organization

For testing our first and second milestones we need to test how well they fly and then working on improving our controls system. The other milestones we will need to test using cod and CICD tests.

3.6 Project Tracking Procedures

Our team has multiple ways of tracking our project's progress. We use Trello to communicate what needs to be done, their deadlines, and by whom. Our client also requires us to write weekly reports to track our one-week sprint's progress and a meeting to discuss the sprint. We have a scrum-style stand-up every week to ensure that our team members are making significant progress, that no progress is being blocked and to plan for collaboration.

3.7 Expected Results and Validation

This project will be considered a success if the features we outlined earlier meet the needs of our clients. We have three client bases, future Microcart teams, Controls systems graduate students, and demonstration viewers. We need to support future Microcart teams by having detailed and organized documentation. We need to have maintainable software development practices. Meeting these requirements will give us a desired outcome for this client base.

We also need our application to support graduate student research development. To ensure this feature we need a comprehensive GUI tool that they can use to interface with the drone. The tool will need to interact with the quadcopter system to support real-time very precise indoor navigation. We will interact with our client frequently to ensure the tool implements his desired features. An example of one of these features is a slider bar to set PID gains on the control algorithm on the drone platform. One requirement that supports this outcome is the installation of Linux onto the second core. We will generate a successful outcome if this client finds our work satisfactory.

Finally, we will need to develop software and hardware features to showcase the value of an ECPE/SE degree to alumni, future students, anyone who may be interested, etc. Having features like swarm flight supports this outcome. These features showcase the value of skills gained because it is an incredibly difficult feature to implement.

We will confirm that all the desired outcomes are met at a high level by continuously asking for client feedback during and after the development process. We will generate various test cases based on our understanding of the user's needs, and we will make sure that these tests pass.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

First Semester Timeline

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Drone Assembly													
Board Assemble													
Chasis													
Testing boards													
Zybo refitting													
Interconnect assembly													
Crazyflie development													
Documentation of research													
Autonomous flight													
Integrating with GCS													
Multiple drone flight													
Ground Station Development													
Implement new PID features													
Documentation for future teams													
Reimplementing lost work													
Features for different controllers													
Project clean-up													

Second Semester Timeline

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Crazyflie development													
Develop GCS Adapter													
Implement Swarm flight													
GCS Development													
Graph of Drone Position Widget													
Draw Flight Pattern Feature													
Extending GUI Functionality to Multiple Drones													
Quad-copter Development													
Extend Networking to Allow Swarm Flight													
Control Algorithm Swapping Support													
Addition of Linux to Second Core													
Project Documentation for Future MicroCART Team													

4.2 Feasibility Assessment

Realistically, we should be able to complete most of our set milestones. We've already achieved two of them, getting the Crazyflies flying and getting a second drone built. Now we're working on the more coding intensive work and we've already run into some issues with previous years code not working. We're currently debugging them and are struggling due to inexperience with the new programs. However, we should still be able to complete most of our goals, since we are making progress and the deadline is about half a year from now.

4.3 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the projected effort required to perform the task correctly and not just "X" hours per week for the number of weeks that the task is active

Task Table

Task	Weeks to Implement
Drone Assembly	
Board Assembly	3
Chassis Assembly	2
Board Testing	1
Zybo Refitting	2
Interconnect Assembly	2
Crazyflie Development	
Documentation of Research	4
Autonomous Flight	3
Integration with GCS	3
Multiple Drone Flight	3
GCS Development	
Implement New PID Features	2
Documentation for Future Teams	3
Reimplementation of Lost GCS Features	3

Features for Different Flight Controllers	3
Project clean-up	2
Crazyflie Development	
Development of Crazyflie Adapter	6
Implement Swarm Flight	6
GCS Development	
Graph of Drone Position Widget	4
Draw Flight Pattern Feature	4
Extending GUI Functionality to Multiple Drones	5
Quad-copter Development	
Extending Networking to Allow Swarm Flight	5
Control Algorithm Swapping Support	5
Addition of Linux to Second Core	2
Project Documentation for Future MicroCART Team	4

Timeline Justification

Our timeline can be broken into two development cycles, there is one cycle for the first semester deliverables, and there is another cycle for the second semester. Each cycle has 3 sub-categories for development. The first cycle has Drone Assembly, Crazyflie development, and GCS development, and the second cycle has Crazyflie development, GCS Development, and Quad-Copter Development. The proceeding description will break down the tasks of each subcategory and justify their allotted completion times.

All of the work for drone assembly has already been completed this semester. The process of assembling the shield board with its components took 3 weeks. This was due to the fact that we had one team member that was comfortable with soldering PCB's. The assembly of the quadcopter chassis took 2 weeks. This task involved assembling all the drone parts in the first week, and attaching motors and soldering ESC's onto the frame of the drone during the second week. We had two team members that were able to implement these features. This gave another team member time to work on refitting the headers on the Zybo board to fit with the shield board. This was accomplished in two weeks. It took one week to perform some basic board tests on the shield board and Zybo Board to confirm their functionality.

Crazyflie development this semester will also lead into next semester because there are features that still need to be implemented. Our Crazy fly lead used four weeks to expand his knowledge of the Crazyflie development environment. He was able to get autonomous flight operational within 3 weeks. The integration with the GCS and multiple drone flight features are still ongoing and showed in the second semester's timeline. The adapter will take 6 weeks to implement during the next semester. This allows 4 weeks to finish implementation and 2 weeks to test and evaluate. During the testing phase of the GCS adapter, the crazyflie lead will begin to implement swarm flight.

The remaining subcategories of GCS and Quad-copter development have similar methodologies for assigning time for tasks with some variance. Each task will have 2 weeks for testing and evaluation at the end of its timeslot. While testing is being performed, implementation of new features should begin. We allocate the last four weeks of the second semester to project documentation. We aim to stop development and thoroughly document the existing system to create and extensive report for the future MicroCART team to pick up where we left off. This manual will prevent the transition issue we had at the beginning of our semester.

4.4 Other Resource Requirements

We have identified the parts and materials required for this project in this document. In addition to that we have identified and ordered switch parts that we need for the Crazyfly and Drone. We also need access to the Coover 3050 room since it holds the quadcopters and the programs needed to work on the drone.

4.5 Financial Requirements

We shouldn't need any financial resources for our project. For anything we need to buy we need to get approval from Dr. Jones. If he agrees he'll order them for us.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case
5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested

8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

Our project requires a wide array of interface specifications. One of the most important specifications we are working on is control algorithm switching for the drone controls. For example, the user should be able to remotely control the drone with a controller and then be swap mid-flight into a different control scheme like PID. Another specification we are working on is improving the GUI to be more user friendly for future Microcart teams. Some examples of our improvements are adding x,y,z axes viewing and implementing slider bars. Another specification we are working on is with our Matlab data analysis tool. It should be able to take test flight data from a txt file and graph using Matlab's graphing software.

5.2 Hardware and software

We have multiple software projects that we use for this project. This means we have multiple interfaces that we use to debug, interact, and develop on these projects. Software we are working on includes Matlab programs for modeling flight behavior, C/FPGA application, and C++ Qt Creator environment with GDB debug tools.

We interface with matlab through the matlab application. The Matlab console will output errors and expected values easily to the screen .Matlab interface specifies debug functions that can be used to trace computational errors in our simulation software.

We interface with the embedded C and Programmable logic on the Zybo 7020 board with the Vivado and XSDK design tools. Both Vivado and XSDK have comprehensive testing functionality. XSDK can use JTAG/USB interface to debug embedded software on the FPGA board.

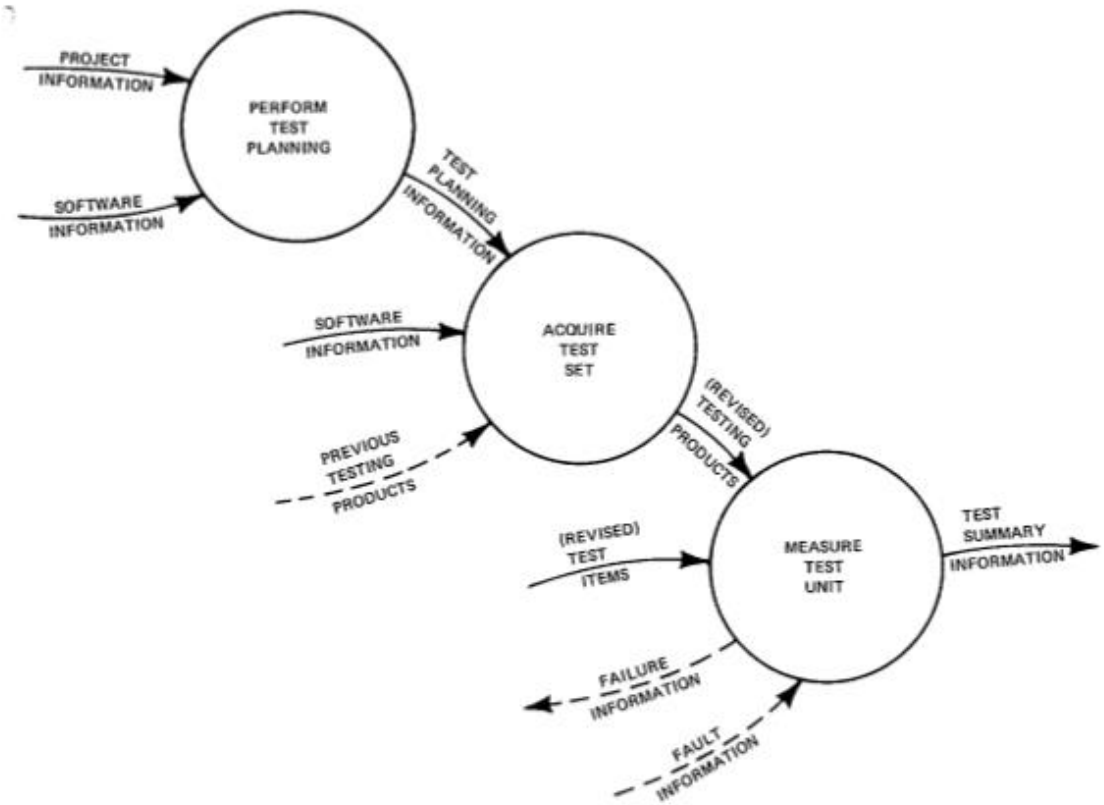
We interface with our C++ GUI application with the QTCreator IDE. This IDE has the Qt Test library which supports a testing interface similar to JUnit tests. This tool also allows you to set breakpoints in your code and use GDB to debug segfault errors.

We also have hardware on our project that we need testing/debug interfaces for. The printed circuit board we use has several copies of outputs. This will allow us to interface the hardware with oscilloscopes to take traces to resolve complex issues.

5.3 Functional Testing

5.3.1 Unit Testing

During our development and testing project any new and some existing software modules that we create will ideally be tested in accordance with IEEE software unit testing standard.We will follow the flow diagram below for developing comprehensive unit tests for modules both on the GCS software, embedded platform software, and Matlab simulation software.



This unit testing scheme can be revised somewhat to include unit testing for critical hardware components. Unit testing has already been done for many of the hardware components on the drone.

5.3.2 Integration Testing

After a software/hardware module makes it past successful unit tests, we assume that a module works, but the only way to verify its functionality after this point is to perform integration tests. Some integration tests have already been conducted for system submodules. We ran unit tests on our Breakout Circuit board and Zybo board, so after they were verified, we ran integration tests with the two programmed boards interfacing with one another. These tests confirmed functionality for the Lidar and IMU hardware modules within the board integration environment.

5.3.3 System Testing

When substantial progress has been made on the development of the project, we will run system testing to confirm that the drone solution has fulfilled our design requirements as stated earlier in this document. One of the future tests could be a test script that generates flight paths for multiple drones. If the drones navigate this path successfully, this would be a solution that met our functional requirement of swarm flight. Outlining general concepts for system testing is important, but the actual implementation of system tests cannot be planned until both unit and integration tests have been finished for the system.

5.3.3 Acceptance Testing

Acceptance testing will be the very last phase of our design evaluation. This will be done with our client and adviser Dr. Phillip Jones. This will determine whether the features we have developed for our drone system have adequately met our system requirements and our client's expectations.

5.4 Non-Functional Testing

The non-functional requirements for our design are as follows: improving the usability of our GCS, improving drone flight responsiveness/latency, enhancing flight accuracy of the drone during autonomous navigation. These requirements will each have separate tests to ensure that their performance metrics are met.

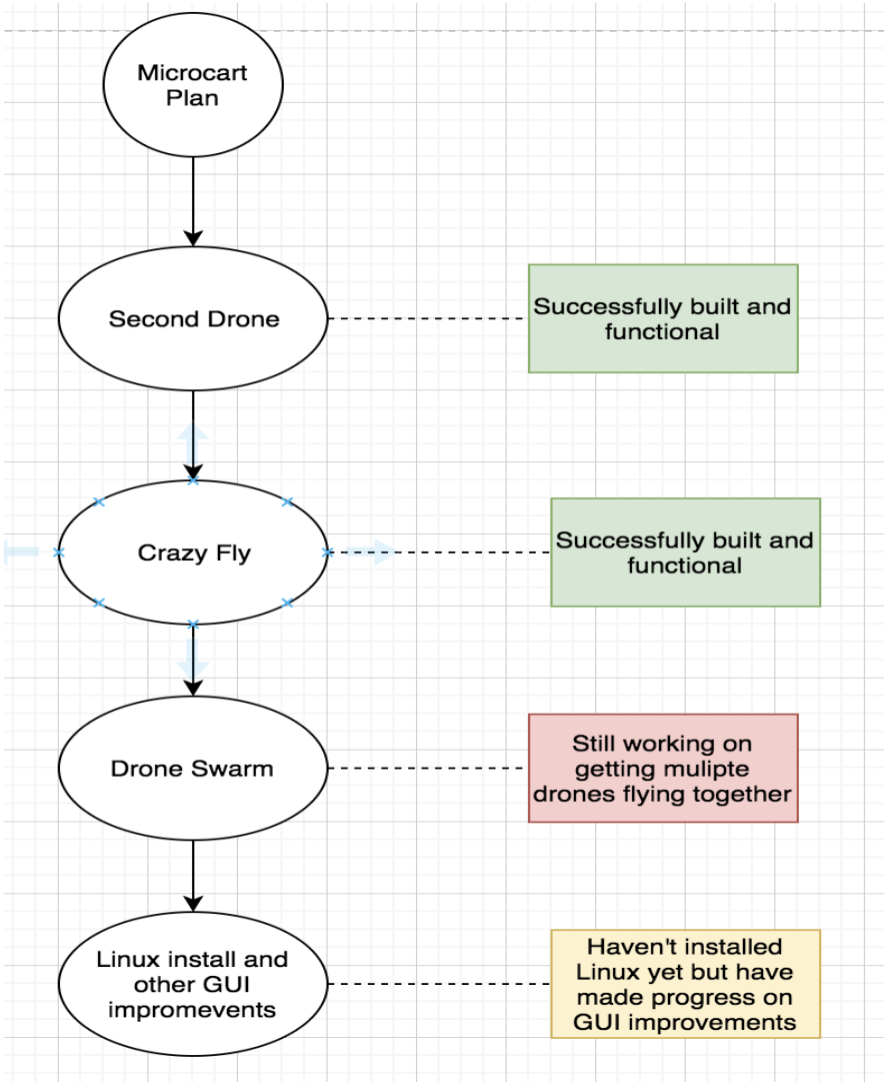
The utility of our GCS software will be evaluated by our primary user group. The GCS station has various functional requirements like the ability to send parameters to the drone, and the ability to receive measurements / data from the drone. We are in contact with a graduate student that is in charge of emulating the needs of our primary user base. He evaluates the non-functional requirements for our GCS software like being able to find a functionality easy.

In order to fulfill the nonfunctional requirement of latency, we have to specify exactly what type of latency is being examined. The drone platform currently has log files that record sensor data such as, actuator output and user inputs. If a setpoint is given to a drone platform like the quadcopter or crazy fly, the latency can be examined by observing the time between when a command was sent from the GCS and when the command was received by the quad platform. The latency of the current system with a networking configuration of one drone is on average 3-4 ms from the ground station and back. There will be changes made to the networking configuration, so we will need to examine the end to end latency after we implement multiple drones to ensure this non-functional requirement.

Flight accuracy metrics can be analyzed with the drone's data logging capabilities. The VPRN camera system will give an absolute reference to compare actual location to setpoint values. A drone navigating on an autonomous path shouldn't vary more than half a foot from its intended location on any axis. Any more variance in flight path could introduce danger during swarm flight, and it would make the project ineffective for autonomous navigation research.

5.5 Process

We used a variety of tests to check on our second drone. Some examples of this include testing the motors individually and all at once and testing if the drone worked with the given scripts. We ran similar tests on the Crazyflie but with manual flight instead of scripts. The other methods like drone swarm and GUI improvements are still a work in progress and we are still in the process of testing them.



5.6 Results

We've managed to make good progress on our Micro Cart project during this first semester. Our successful results so far include building a second working drone, soldering an FPGA, implementing some GUI improvements like sliders, building several Crazyflies and flying one of them, and fixing last year's data analysis code.

The results we are working on improving is getting two drones flying at the same time, getting a Crazyflie swarm working, implementing more GUI improvements, and implementing control algorithm switching.

Some implementation issues we are running into is the lack of documentation for previous years code. This is a problem because whenever we try to run their code, we get error messages so a good chunk of our workload this semester was trying to understand and fix the code. Similarly, we also have some issues on the hardware side of the project because there's no documentation or instruction on what we're supposed to do. We also had issues with our lab space since we need admin access on the computers to continue working and we're still waiting on that.

6. Closing Material

6.1 Conclusion

The work we have done so far is building several crazyflies and achieving flight with one of them and successfully building a second working drone. Some more abstract work we've also finished is GUI improvements to the software and fixing up last year's code. Our end goals include getting crazyflies swarm going, two drones flying at the same time, and maybe both the crazy and drones flying at the same time. Our current plan that has worked out well so far is to have several smaller work meetings throughout the week followed by a big work meeting with everyone on Sunday. Since it's been so successful so far we'll probably continue doing this for next semester.

6.2 References

Minimal technical references were made throughout the document. Little technical documents were referenced as the majority of knowledge was pulled from previous work done by previous teams.

[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, Link.

6.3 Appendices

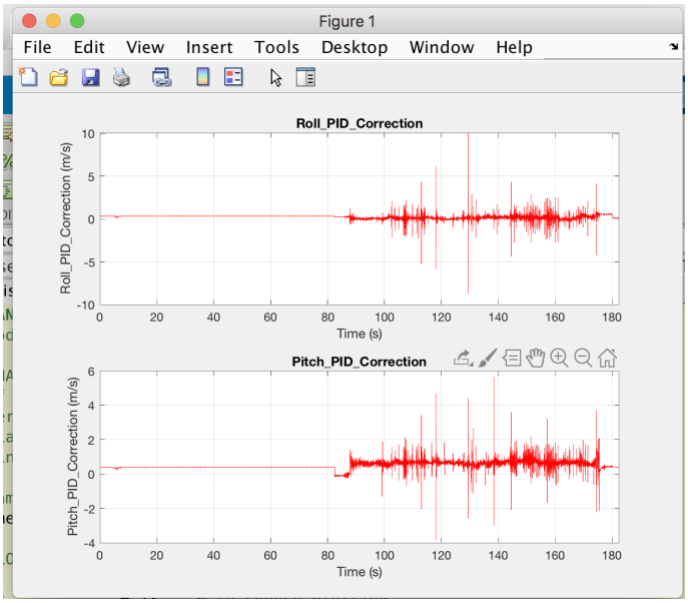


Fig 1. Data tool analysis sample

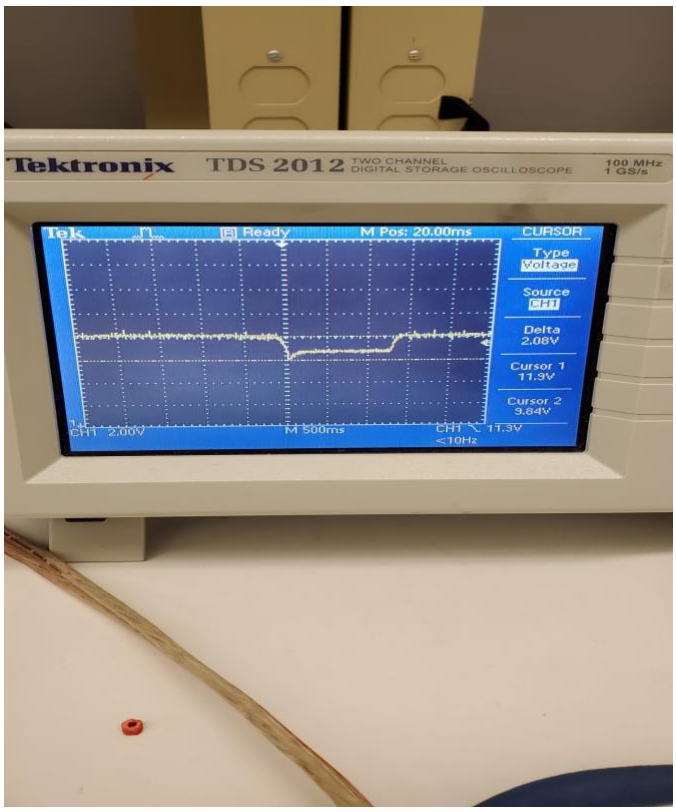


Fig 2. Successful Quad motor test reading