MicroCART Mini

DESIGN DOCUMENT

Team Number: sdmay25-32

Client/Advisor: Dr. Philip Jones

Team Members & Roles:

Ryan Lowe: Technical Advisor

Daniel Zaucha: Client interaction, Communications Lead

Jonah Upah: Hardware Lead, Team Secretary

Yi Hang Ang: Software Lead

Team Email:

sdmay25-32@iastate.edu

Team Website:

https://sdmay25-32.sd.ece.iastate.edu

Revised 5 December 2024

Executive Summary

The name of our project is MicroCART which stands for Microprocessor Controlled Aerial Robotics Team. This project is divided into two parts. The first part is optimizing and improving a quadcopter called CrazyFlie which will be used to conduct Lab 4 of CPRE 488: Embedded Systems Design. This is important because we are providing a better and more convenient environment for CPRE 488 students to learn. The second part of this project is implementing and improving on the firmware of a quadcopter called FlyPi, which is a product of last year's MicroCART team that will be used in future live demonstrations during Scholar's Day at ISU to attract prospective students. The long term goal of this project is to learn from what was and will be implemented on the CrazyFlie and further improve on our heritage, the FlyPi.

Our key design requirements would be to improve the backend to frontend communication and the graph logger of the CrazyFlie to reduce latency by implementing a new packet in Python which will contain graph logging values. As of now, we will need to grasp a better understanding of the backend communication before implementation. If implemented perfectly, this design requirement will provide students with a smooth and clearer GUI graph logger, which will help students have a better understanding of the graph.

Besides that, we plan on implementing a new component called a test stand tracker for the CrazyFlie. This tracker is an arduino which will provide more accurate setpoint values of the PID controller. The tracker provides the students with more accurate setpoints which will assist them in deducing the parameters of the PID controller. What we are working on right now is connecting the test stand tracker to the backend.

For the FlyPi, we plan on implementing a global positioning tracking system that will keep track of the current position of the FlyPi while it is autonomously moving around the test field. This requirement will be implemented last, therefore we do not currently possess the knowledge regarding the details of this implementation, but the idea would be to calculate the distance the FlyPi has traveled in x,y, and z coordinates through the aid of the PID controller and the IMU (Inertial measurement unit).

Learning Summary

Development Standards & Practices Used

Since this project is mainly software focused embedded programming in C, C++, and Python, we would prioritize following standard software development standards.

- Variable assignments should not be made from within sub-expressions
- Class names should comply with a naming convention
- Local variable and function parameter names should comply with a naming convention
- Failed unit tests should be fixed
- Resulting code causing failed pipelines on Git should never be deployed

Engineering Practices

Engineering standards are essential because they ensure that the products produced by engineers meet users' expectations. This means that the product is safe for users, has a standard or consistent quality, and is environmentally friendly. Adhering to engineering standards increases work efficiency and speeds up the engineering process. Therefore, the existence of engineering standards is beneficial to both users and engineers, and these are the engineering standards that we think are applicable to our project.

- IEEE 802.11s-2011: IEEE Standard for Wireless LAN Medium Access Control (MAC)

    - This standard has relevance because it focuses on wireless communication. In our project, we have a backend that communicates the remote-controlled quadcopter with the ground station, which means wireless communication and data transmission.
- IEEE 1687-2014: IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device
    - This standard is about accessing instrumentation embedded within a semiconductor device, which is precisely what our project focuses on integrating and improving a quadcopter's embedded systems.
- IEEE 1936.1-2021: IEEE Standard for Drone Applications Framework
    - This standard is about frameworks for the support of drone applications. Our project emphasizes working with drones (quadcopters) and their flight control systems.

Summary of Requirements

1. Mini Quadcopter should be able to:
    - Fly smoothly
        - Flight stabilization
        - No sudden "random" movements
        - Quick reactions to directional inputs
    - Connect to remote equipment for data analysis
    - Be able to connect to remote sensors and utilize the information to fly

- Be utilized easily even by someone with no prior experience in controlling any remote control vehicle

2. Frontend/Backend should be able to:
    - Be accessible through the current method
    - Display data from flight information
    - Be able to enable an uncontrolled flight via sensor data
3. Documentation must:
    - Explain the steps throughout the project
        - Plan of action, task breakdown, time taken, changes made form the previous project, what we could have done better, what we did not get to and will leave for the following year's group to complete.
    - Document any problems that came up throughout the development process and record how we solved them for future project groups or, when applicable, by teachers, TA's and students.
    - How to solve issues that come up frequently (FAQ Sheet)
    - Catch users up-to-speed for the programming project depending on their role
        - Student, TA, Advisor, Teacher, Successor team, or the general public
    - Show and explain how our project connects to various other fields and draw interest from observers to look deeper into it

Applicable Courses from Iowa State University Curriculum

1. CPRE 2880: Embedded Systems 1: Introduction
2. CPRE 3080: Operating Systems: Principles and Practice
3. CPRE 4880: Embedded Systems Design
4. CPRE 4890: Computer Networking and Data Communications

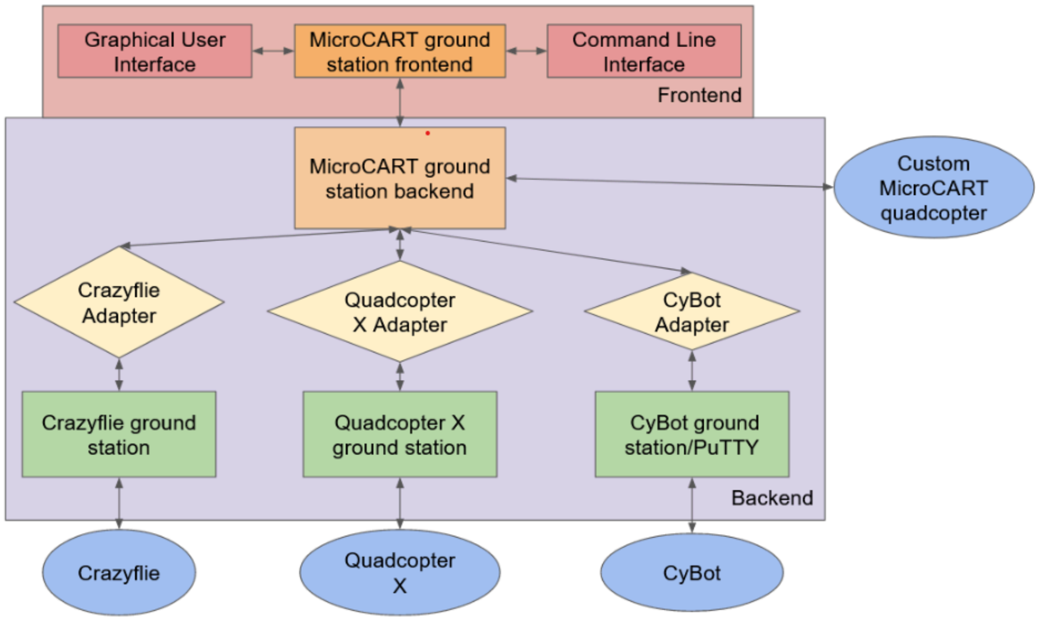
New Skills/Knowledge acquired that was not taught in courses

1. Control Systems Theory
2. Socket Programming

Table of Contents

List of figures/tables/symbols/definitions

1. MicroCART system overview:



2. Glossary of Terms:

   - CPRE 488 MP4/Lab 4 - The fourth lab of the CPRE 488: Embedded Systems Design course. Our team will be in charge of optimizing the equipment and software used to conduct this lab.
   - CrazyFlie - The small drone used for the CPRE 488 MP4 Lab. It is manufactured by Bitcraze, and has open source firmware which can be easily written.

- FlyPi - The larger custom quadcopter built by previous MicroCART teams. Currently in a complete state, but would require some optimizations.
- GUI - C++ based Graphical User Interface that is created with the application QT, which allows a user-friendly display of the frontend.
- CLI - Command Line Interface, an interface through the command line which allows the user to interact with the system using specific commands.
- Ground station - Name used for the group of software components that lie in between the quadcopters and the GUI: Backend, CrazyFlie Adapter, and CrazyFlie Ground station.
- Backend - Software module written in Python which handles incoming packets from the frontend and sends them to the necessary destination. Also handles data from cameras and other sources.
- Crazy radio/dongle - USB radio stick that sends packets to and from the CrazyFlies.
- Test stand - A device used to hold the CrazyFlie in place while fine-tuning its parameters, a port is located at the bottom of the test stand that allows an arduino to connect to it.
- Test stand tracker - An arduino connected to the test stand that will collect positional data from the CrazyFlie through the port and sends it to the PC directly via a USB cord.

1. Introduction

1.1. Problem Statement

Quadcopters, and drones in a broader sense, are seeing more day to to day usage across many fields such as agriculture, transportation of goods, the military-industrial complex and so many more! Our project is known as the MicroCART Mini project and we are designing/iterating new software for a mini-quadcopter that will be used as learning materials for our College's Computer Engineering students, in addition to actualizing a quadcopter into flight via designing hardware and software. As our project is focusing on creating small remote controlled devices, for both educational and non-specific usages, the focus of our design will primarily be on sustaining controlled flight. Uncontrolled flight is a hazard not only to the quadcopter but also to the environment around it, which means we will have to make the controls adaptable for the mini quadcopter to be used by untrained non-professionals and for an automated program to be able to utilize sensors to obtain information from around the quadcopter and through a program complete a flight in new terrain while minimizing damage from or outright preventing any crashes. Since our quadcopters are constrained by their small sizes, we will be connecting our quadcopter with remote sensors to absorb information such that our device, which lacks them, will be able to utilize it for flight navigation. Sensory navigation opens up the possibility for unnavigated routes to be flown, such as in a disaster scenario for search and rescue, to have new route information recorded, and for optimizing a flight path in new terrain safely.

1.2. Intended Users

1. **CPRE 488 Students**
    a. Senior/Graduate-level students taking CPRE 488 in Spring 2025.
    b. Must have completed CPRE 381 or COMS 321
    c. Must be able to perform with Mini-Quadcopters after 4 intro labs
    d. Limited amount of time they can dedicate solely to this class

Needs: CPRE 488 students need an operational and improved platform to work on Lab 4. Our Senior Design team can improve the prior hardware, systems and framework to provide students with a more convenient environment to work on Lab 4.

Benefits: Lab 4 for CPRE 488 students will be conducted smoother, increasing their productivity and making better progress. Students will also learn how to fine-tune control systems like the PID Control in Lab 4 and implement that into an RC quadcopter.

2. **Successor Project Team**
    a. Senior-level students working on the MicroCART Senior Design Project in the future.
    b. Senior-level knowledge base
    c. Multiple Disciplines (i.e., CPRE, EE, SE)
    d. Will be working off of what we left off

Needs: Successor Senior Design teams would need tutorials like a step-by-step guide or video tutorial on complicated parts of the project. Besides that, they would need proper and updated documentation on the project based on what we changed and improved from the past projects. They would also need code that is easy to understand and make changes to.

Benefits: With improved information “library”, successor project teams will be able to find the information that is associated with the different parts of the project they will be working with in a shorter period of time and be able to catch up or surpass the progress that our team has made in comparison. It will also give an outline of the order to go about the project when they are starting out to give themselves a longer period of time to optimize their own progress.

3. **CPRE 488 Teacher/Advisor/TAs**
    a. Course Instructors for CPRE 488
    b. High-level course knowledge
    c. May have seen previous projects done and performed
    d. Observing to see if our project and students’ work meet project requirements
    e. Have limited time, and more responsibilities

Needs: A high-level overview of what the project is and how we have organized the project. Separation of different presentations and the expectations that we were trying to meet for all. Note that detail where we found trouble and what helped us get past it. General instructions that detail the processes we went through, explaining why we chose certain methods.

Benefits: A reduced time period for reading necessary items and ignoring the details that they have seen before and can otherwise ignore. An enhanced ability to find where groups/individuals are struggling and to be able to quickly tell them possible solutions to issues that rise up. Able to take up less time than would otherwise be without the pre-recordings

4. **Potential Incoming College Students**
    a. High school tour groups
    b. Highschool level knowledge
    c. Want to attract them to be like us
    d. May have interests in other engineering fields
    e. Need to show them how this connects to other ISU disciplines

Needs: Potential incoming college students need to be interested and drawn into our project and be able to see what they could learn if they were to become students here. The incoming students need to have an explanation of the project that will make sense to them, given they will not be familiar with much of the material/technologies we use.

Benefits: Potential students may be able to base their decisions on colleges by seeing what engineering students at Iowa State can accomplish.

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

- <u>Mini Quadcopter should be able to</u>:
    - Fly smoothly
        - Flight stabilization
        - No sudden “random” movements
        - Quick reactions to directional inputs (For example, to stop turning when the turning button is no longer pushed)
    - Connect to remote equipment for data analysis
    - Be able to connect to remote sensors and utilize the information to fly
    - Be utilized easily even by someone with no prior experience in controlling any remote control vehicle

- <u>Frontend/Backend should be able to</u>:
    - Be accessible through the current method
    - Display data from flight information
    - Be able to enable an uncontrolled flight via sensor data

- <u>Documentation must</u>:
    - Explain the steps throughout the project

        - Plan of action, task breakdown, time taken, changes made from the previous project, what we could have done better, what we did not get to and will leave for the following year's group to complete.
    - Document any problems that came up throughout the development process and record how we solved them for future project groups or, when applicable, by teachers, TAs, and students.
    - How to solve issues that come up frequently (FAQ Sheet)
    - Catch users up-to-speed for the programming project depending on their role
        - Student, TA, Advisor, Teacher, Successor team, or the general public
    - Show and explain how our project connects to various other fields and draw interest from observers to look deeper into it

2.2. Engineering Standards

The sub-category that is appropriate for our project would be Computer Technology. The three IEEE standards that apply to our project are:

1. IEEE 802.11s-2011: IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment.
    - This standard has relevance because it focuses on wireless communication. In our project, we have a backend that communicates the

remote-controlled quadcopter with the ground station, which means wireless communication and data transmission.

2. IEEE 1687-2014: IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device
    - This standard is about accessing instrumentation embedded within a semiconductor device, which is precisely what our project focuses on integrating and improving a quadcopter's embedded systems.

3. IEEE 1936.1-2021: IEEE Standard for Drone Applications Framework
    - This standard is about frameworks for the support of drone applications. Our project emphasizes working with drones (quadcopters) and their flight control systems.

These standards for our device were chosen due to how our project, which is building and/or implementing a control system into a mini quadcopter, and this utilizes remote control through wireless devices, accessing the quadcopter itself for data, and reiterating the fact that this is a drone device that we are using.

Some of the other possible standards choices that we did not choose were battery standards. We did not use these standards due to them being rather broad and rather nonspecific to our project for the most part, due to them being a far more secondary aspect, in comparison to our chosen standards which are indubitably more so related.

3 Project Plan

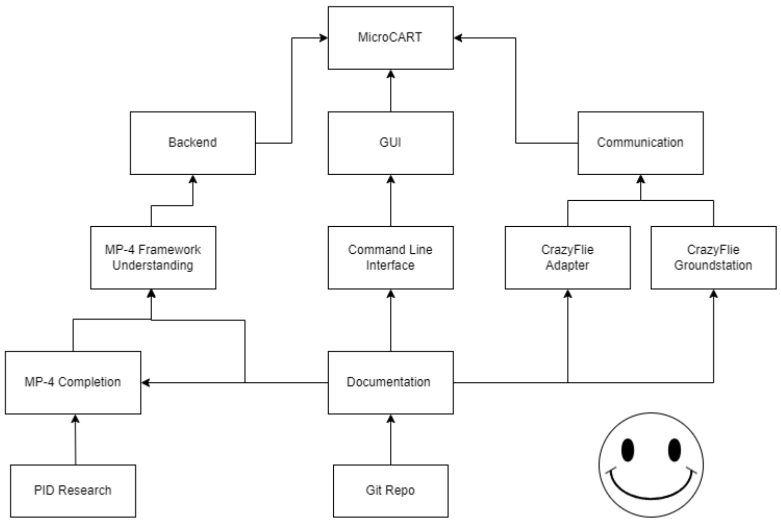
3.1 Project Management/Tracking Procedures

We are adopting a waterfall-agile hybrid methodology. We have broken down our project into different phases to guide the general path for the rest of the project. The different phases include MP4, Backend, and Frontend. To effectively distribute work and manage deadlines for these different phases, we adopt the Agile methodology to allocate tasks and issues.

Progress throughout the course of this project will be documented through the use of GitLab issues. GitLab issues will be used to designate tasks for each team member and provide a timeline for what we need to work on. This is how previous teams for this project have tracked progress, and we will follow suit. It is also helpful to track deadlines and motivate/keep track of team members to work on a specific issue before it is due.

3.2 Task Decomposition

- Documentation
  - Progressive throughout
- MicroCART
  - CPRE 488 - MP 4 (aka Lab 4)
    - PID Research
    - CrazyFlie



    - Backend
        - CPRE 488- Framework
    - Frontend
        - GUI & CLI
    - Communication
        - CrazyFlie Adapter
        - CrazyFlie Ground station
    - Global Positioning Control
- Semester End Presentation

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones:

MP-4 compilation:

- Measuring is easy, it's based on an existing 488 lab. The goal is 100% completion but not making a lab document that a student must submit.

Understanding the backend(Big picture):

- This involves understanding how the system works from a high-level perspective. It is quantifiable by the ability to to explain what each component is and its purpose.

Dive into sub-components of the communication pipeline:

- Expanding the last milestone, we then need to gain a deeper understanding of each subcomponent. This is quantifiable in a similar way.

Optimize issues with the GUI:

- There are issues within the GUI that affect the ease of use, while it is functional, there are prominent bugs that should be addressed. This is measurable by the amount of bugs encountered during a session. We aim to attack the most common ones to reduce the debugging time for students.

Add Global Positioning:

- This is a new feature we will be adding to the 488 lab. This would need some involvement from Dr. Jones to tie it to the 488 curriculum. To measure this would lay the groundwork for other teams to build upon. The best outcome would be to implement the feature and thoroughly add it to the lab.

Explore FlyPi:

- This is a stretch goal in the experimental portion of the project, the previous teams were doing some fairly complex stuff. A reasonable goal for us would be to organize better what already exists. The best outcome would be to expand upon what the last group left us.

Pick up where the last group left off (FlyPi):

- This is expanding on the last milestone. The actual contents of what is achievable are unknown at this point as we have not done the exploration yet.

3.4 Project Timeline/Schedule

Note:

- Subtasks are other colors of the same group
- Associated tasks are worked down over while working on the task itself

Tasks	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17
Documentation	⬛	⬛	⬛	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Backend CPRE 488- Framework				🟩	🟩	🟩	🟩	🟩							
CPRE 488 - MP 4				🟩	🟩	🟩	🟩	🟩							
PID Research					🟩	🟩	🟩	🟩							
CrazyFlie							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
GUI & CLI							🟦	🟦	🟦	🟦					
Communication (Adapter & groundsatation)									🟦	🟦	🟦	🟦	🟦		
CrazyFlie Adapter									🟦	🟦	🟦				
CrazyFlie Groundstation											🟦	🟦	🟦		
Global Positioning Control														🟦	🟦
Semester End Presentation													🟪	🟪	🟪

3.5 Risks and Risk Management/Mitigation

Risk Scale: 1 (Low) - 10 (High)

Backend CprE 488 framework/ CprE 488 MP-4:

(2) Low risk:

- We are working to optimize the existing solution. There are some bugs present that could hinder students' progress in lab 4. The risk is low because we can always revert to the previous version that contains minor bugs.

- Mitigation: do incremental solutions so that if we have trouble with one aspect, it doesn't affect others.

Ground Station and Adapter:

(1) Low risk:

- This was requested from the previous senior design team, They wrote the software to communicate with the crazy file through the crazy radio. In that pipeline, there is an intermediate component, the adapter. They mentioned that the ground station should absorb this. It works as is but might help speed up the communication pipeline.

- Mitigation: This is more just ensuring that we don't diminish performance and that the result is more readable to next year's team.

Global Positioning:

(7) Medium risk.

- This is a familiar feature therefore, we would be building it from the ground up. It would be a rewarding aspect of the project. But the risk is that we don't get it done in the time allotted and then would have to pass it to the next team which could lead to miscommunication or abandonment of the feature.

- Mitigation: makes sure that we document our intentions and our progress to a degree where, if we don't finish, the next group will be able to pick up where we left off quickly.

Deployment:

(9) High risk

- This is the final deployment of the VM containing our revisions. At the end of the year, we will want our changes to be deployed across all the 488 lab machines. There is a reasonable concern that if there is an issue, there would be little time to fix it. Therefore, none of our changes would take effect for the 488 lab.
- Mitigation: do a test deployment beforehand to identify potential issues.

3.6 Personnel Effort Requirements

4 Group Members - Expecting minimum of 6 hours per week from each person

{Please understand that for estimated hours we are using the value of the largest amount of time expected to be spent on a task, even though from the Gantt chart on 3.4.) we can see overlapping of multiple tasks on various weeks, so even though estimatedhours/week/

individual is 6 per say, then during the following weeks less time will be dedicated to the task.}

[Hours Total Formula]: Duration x Estimated Hours/Week x Number of individuals

Estimated hours Formula: [Hours Total] / 4 / 12

(Rounded up due to 3 weeks of solely research)

Tasks	Duration (Weeks)	Estimated (Hours/week) [Per individual]	Hours Total (Hours) [Sum of all team members]
Documentation & Research	14	2	96
Backend CPRE 488- Framework	5	–	–
CPRE 488 - MP 4	5	6	120
PID Research	4	2	32
CrazyFlie	9	–	–
GUI & CLI	4	3	48
Communication (Adapter & ground station)	5	-	-
CrazyFlie Adapter	3	4	48

CrazyFlie Ground station	3	4	48
Global Positioning Control	2	3	24
Semester End Presentation	3	3	48
Estimated Total Duration	12	~10	464 ~116 per individual

[While the above is a calculated estimate, a more likely approximation is 80 total hours and 8 hours/ week after accounting for time overlaps, division of work, and breaks]

3.7 Other Resource Requirements

Some non-financial resources that our project is utilizing include knowledge from our preceding groups and the code repository, virtual machine, a BitCraze CrazyFlie information sheet that will record the state of lab equipment. Quality assurance is checked by both us and our advisor to ensure that it works to expected specifications.

4 Design

4.1 Design Context

Our design focuses on improving and optimizing the infrastructure and equipment of CPRE 488 Lab 4 to improve the quality of learning of the students. Besides that, we aim to hold a live demonstration of our design of the FlyPi drone during Scholar's Day at ISU to attract prospective students.

4.1.1 Broader Context

This semester our project consisted of preparing the lab materials for future CPRE 488 classes, which includes preparing the lab documentation as well as fixing issues with the lab materials whether they be hardware or software related. This means the community we are designing for is the Iowa State University's engineering department of education. Our design will affect the senior level students that are both taking this course next semester, the teacher of CPR E 488 (Dr. Jones), and the senior design teams that take over this project next year. A broader summary of how our project covers the four principles of responsibility is detailed in the table below:

4.1.2 Prior Work/Solutions

The MicroCART project has been ongoing for many years, but the MicroCART mini aspect of the project has been going on for about 6 years, when the CPR E 488 class was revamped due to the progress of technology and started to move away from large and dangerous quadcopters. The teams from previous years have designed many of the class materials ranging from the classroom GUI that is used in their lab 4, the lab documents, and even the custom 3D-printed mini-quadcopter testing stands that are used constantly after getting into the later parts of the lab.

But how does our project compare to similar products (mini quad-copters)? To answer that, we must first explain what some of the mini-quadcopters available on the market do. The products that we are going to utilize for this explanation are the Kopis Freestyle 4-inch FPV Drone[5], the DJI Mini 4 Pro Drone[6], and an "Open source ESP32-based quadcopter made from scratch"[4], that was made from individually sourced pieces by a professional in the field of computers and robotics. The Kopis Freestyle 4-inch FPV Drone[5] is a mini-quadcopter with a first-person camera, but with a short flight-time of 5 to 6 minutes without a battery upgrade. The DJI Mini 4 Pro Drone[6] is a mini-quadcopter that is quite 'large', with a longer battery life of 34 minutes or 45 with the battery upgrade, a more crash-resistant frame, a camera with night vision, but at the cost of a much heavier price ($759 without upgrades) and a weight of 249 grams, one gram away from 250 grams at which point any drone must be registered to the FAA (Federal Aviation Administration)[3]. The most similar drone to our project is the "Open source ESP32-based quadcopter made from scratch"[4], which is a drone that is composed of individual parts that are all detailed with the locations of where to obtain them. This drone focused more on stability and was able to fly quite stably in the demonstrations shown.

Some of the things that make our MicroCART project unique from these mini quadcopters is how our GUI interacts with the CrazyFlie to transmit commands and receive data, while the also having the custom test stand that enables the testing and configurings of the drones flying capabilities while denying any accidents from occurring. Most drones on the market-place are connected to a limited piece of software, with only the open-source drone having the ability to make custom firmware for it in an easy manner. Unlike the other open-source drone however, our CrazyFlie is an open-source drone from BitCraze[1], who sells the drones and equipment needed to communicate with them, while also providing trouble-shooting software that is freely downloaded and can fix firmware should errors occur in the microprocessor during a crash.

4.1.3 Technical Complexity

The technical complexity of our project's design is slightly lower in regard to hardware compared to the rather large complexity when it comes to our software.

Our hardware is relatively new but also partially self-programmed and self-designed by past project groups since the hardware is intended for learning usage. The hardware consists of CrazyFlies (mini quadcopter(s) used for student practice), a test stand & its reading device, and controllers which all together are rather simplistic and require only a small amount of practice to get used to while lacking large difficulties when it comes to learning them.

The software complexity is akin to a high incline before a plateau in comparison to the hardware which is a small but linear hill. With the software having both a backend that works itself through the use of 3 different programming languages, and a frontend that has some issues reading from the backend that increase as the duration of the program increases but is otherwise straightforward when you use it along the given instructions. Our project scope includes making it simpler to utilize and measure the hardware while increasing ease of comprehension and decreasing issues and bugs that occur in the software. As our project utilizes open-source hardware and software, this means that there are likely other similar technologies, programs, or codes that utilize some of the hardware(s) and software(s) making it more well-established.

In conclusion, this leaves us with a project that has a medium-high internal technical complexity, while having a low external technical complexity.

4.2 Design Exploration

4.2.1 Design Decisions

The area that we are trying to optimize is the backend portion which is the backbone of our project. The backend connects to our test stand using USB, and it uses a TCP connection to connect to the CrazyFlie adapter and ground station.

A key design decision that we will need to make is implementing a test stand tracker and connecting it to the backend. A test stand tracker is a more convenient way to capture the yaw, roll and pitch rate values of the CrazyFlie quadcopter, because the test stand has a more accurate sensor, which is able to accurately calculate the rate values based on the movement of the quadcopter while bound to the test stand. As of now, students are using the on board sensors for logging variables in the CrazyFlie quadcopter to plot on the GUI graph. The built in logging variables may not be as accurate and as convenient because most of the time it fails to capture the specific state the quadcopter is currently in as well as its current yaw, roll and pitch rate values.

Besides that, another design decision that we made is combining the CrazyFlie adapter and CrazyFlie ground station. The CrazyFlie adapter decodes packets from the backend and sends data to the backend, and has some callback functions that do relatively little actual translating like set_param, and get_param etc for logging variables. So in reality, it appears that the adapter basically just lets things pass almost right through to the CrazyFlie ground station. Therefore, we can omit the CrazyFlie adapter entirely by combining the adapter and the ground station, with the benefit of this being the reduced communication overhead of sending data across less components.

Lastly, we plan on implementing a Global Positioning System (GPS) for the FlyPi quadcopter to be used in our demos during Iowa State school tours. In the past, MicroCART teams have found a workaround for this temporarily by comparing two different simulations of quadcopters, with one using a PID controller and the other using a LQR controller, which does not necessarily provide the touring students a comparison between simulation and reality. But with an implemented GPS, we are able to compare the aerial coordinates of a quadcopter simulated with a PID controller and the currently in-flight quadcopter using a GPS.

4.2.2 Ideation

Design decision: Fixing issues in the backend, such as inconvenience during graph logging that demonstrates out of order and dropped packets.

1. Start fresh
    a. Instead of working on the current iteration of our project code, we could create a whole new backend from scratch in an effort to solve any issues that we found.
2. Not changing the backend code, but changing the GUI code
    a. Deploy an artificial fix to the solution. When packet loss occurs, do not allow the setpoint/value to drop to 0, but instead retain its previous value. When out of order packets occur, never log and graph values that were in the past.
3. Polishing current backend
    a. Going through the lab that our project will be used in and then writing down and implementing fixes that we find along the way.
4. Combining languages for backend code
    a. Currently our backend code makes use of C, C++, and Python to work and if we were to unify it all, then it should be easier to find where an error occurs compared to having to check at least 3 code files.

5. Implementing a test stand tracker and connecting it to the backend component
    a. Implementing a test stand tracker will allow the GUI to log and graph more stable and accurate values compared to the built-in sensor values, allowing for a more convenient GUI and student interaction.

4.2.3 Decision-Making and Trade-Off

The following diagram is intended to enumerate how we felt about our solution in values from 1-5 with 1 as low (better) and 5 as high (worse) in terms of choice, while the most likely end result explains what we predicted the solution would turn out.

Solutions	**Time**	**Difficulty**	**Probably Outcome**
Start Fresh	5	5	● Maybe working ● Many bugs
GUI instead of Backend code fix	1	1	● Working, but still inherently flawed ● Later project groups may not see what we were looking at
Polish Current Backend	2	3	● Definitely working ● Fixes problems along the way
Combine Backend languages	4	4	● Maybe working ● Mixed results, not necessarily best ● Different languages have better results when used for specific purposes

Implement a Test Stand Tracker and connecting it to Backend component	3	2	● Working ● Not a cure-all for all of the problems that the backend is having ● May have other problems later down the line

We ended up choosing “Polish Current Backend”, while also looking at implementing the test stand tracker and keeping the GUI fix in mind. For our decision the probable outcome mattered the most in comparison to time and difficulty of the option itself. We chose this because it will address the backend problems and improve information intake, while also leaving us with a guaranteed working product with less flaws that a user would see, since we looked through and learned how to use it as users ourselves.

4.3 Proposed Design

4.3.1 Overview

Our main design will take place in the CrazyFlie ground station, a pre-implemented system, which has a high-level description as shown in Figure 1. The following are the descriptions of the key components of the ground station:

1. CLI / GUI

The CLI (Command Line Interface) and GUI (Graphical User Interface), are both interfaces which allows users to interact with the Crazyflie through the frontend. The GUI provides a more graphical way of representing data compared to the CLI which requires users to know specific commands to interface with the CrazyFlie. Both CLI and GUI transmit and receive data to and from the frontend component, allowing users to modify, receive parameters of the system, as well as receive data from the CrazyFlie to allow logging and graphing of the data in the GUI.

2. Frontend

The frontend of our ground station is what users see and interact with directly. Our frontend is composed of the CLI and GUI. It transmits and receives data from the backend, and presents that data in a more visually appealing or easier to understand way to the users.

3. Backend

The backend component is the backbone of our ground station. It receives and decodes packets of data from the sockets, as well as transmitting and encoding packets to be sent to the sockets. Overall, the backend component is in charge of translating packets into data or vice versa, while acting as a medium of communication between the user (frontend) and the CrazyFlie (sockets).

4. Sockets

The socket component here represents the way of how CrazyFlie communicates with our backend wirelessly. Through the wireless connection of TCP web sockets, our backend will form a two-way communication with the CrazyFlie, allowing us to modify the software currently present in the quadcopter.

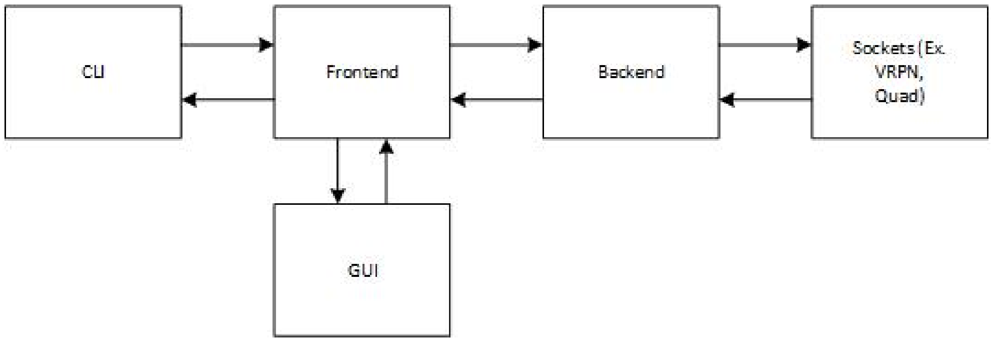


Figure 1: High-level description of the CrazyFlie ground station

4.3.2 Detailed Design and Visual(s)

A more specific description of the CrazyFlie ground station is shown in Figure 2, which includes details and more technical terms that are easier for engineers to understand. The following are the more detailed descriptions of the key components of the ground station:

1. CLI / GUI

Both the CLI (Command Line Interface) and the GUI (Graphical User Interface) receive data from the frontend and outputs it in a way that is easier for the users to understand. The CLI has multiple commands encoded, which requires users to know detailed commands in order for them to interact with the CrazyFlie. An example of commands include:

- sendSetPoints(), which allows users to set a specific setpoint for the CrazyFlie during PID controller tuning
- setParamValue(), which allows users to set parameter values such as the kp, ki, kd values of the PID controller

The CLI program is written in C but the GUI is written in C++, because our project utilizes an application development framework for creating GUIs called Qt, which generates C++ code for GUIs created. With the GUI, users are able to get and set parameters without the need to know the actual command to do so, with an example shown in Figure 2.

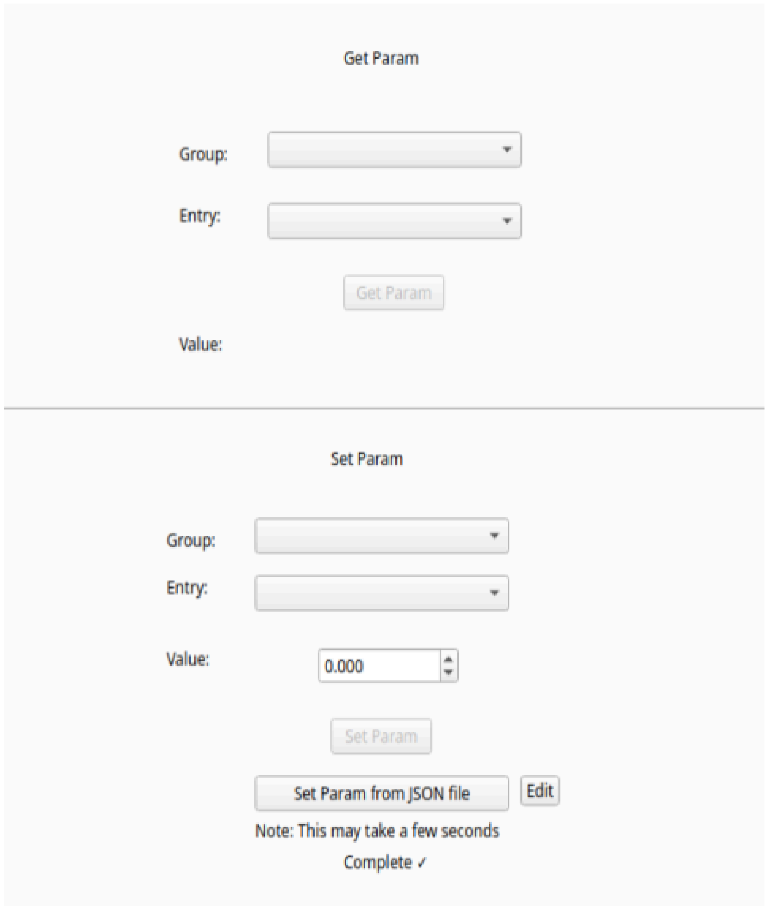


Figure 2: Using GUI to get and set parameters

2. Frontend

The frontend of our ground station is what users see and interact with directly. Our frontend is composed of the CLI and GUI. It transmits and receives data from the backend, with the importance of being able to create and store new logging variables to be displayed and graphed in the GUI graphing tool.

3. Backend

The backend component is the backbone of our ground station. The main function of the backend is decoding and encoding packets, it receives and decodes packets from sockets and transmits them to the frontend. It also encodes and transmits the data it receives from the frontend and transmits it to the sockets, it does so by combining metadata and data to form a wire-sendable packet, and sends it through TCP sockets, where the backend acts as a TCP socket server, listening for a client to connect.

4. Sockets

The sockets that we use in our project are TCP web sockets. Encoded packets are sent from the backend server socket to the CrazyFlie client socket and the CrazyFlie will modify its software settings based on the packets received. This allows users to fine-tune PID controllers by setting their parameters (kp, ki, kd) for either yaw, pitch and roll rates in the GUI and in the end it will get transmitted to the CrazyFlie.

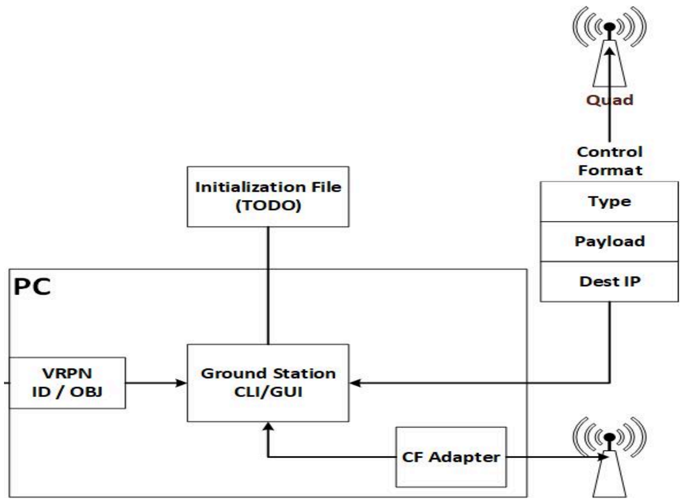


Figure 3: Detailed description of the CrazyFlie ground station

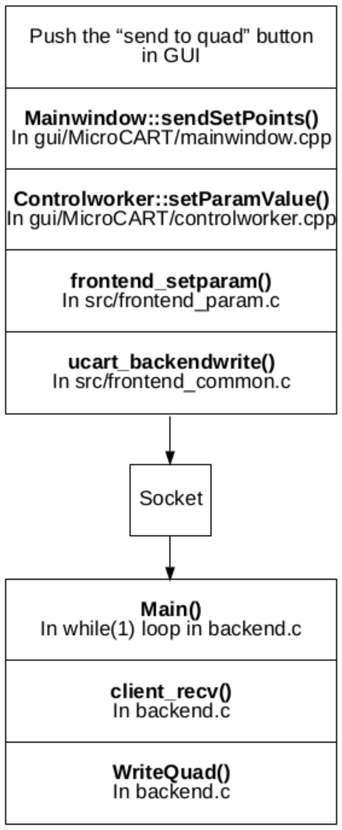

Push the "send to quad" button in GUI
Mainwindow::sendSetPoints() In gui/MicroCART/mainwindow.cpp
Controlworker::setParamValue() In gui/MicroCART/controlworker.cpp
frontend_setparam() In src/frontend_param.c
ucart_backendwrite() In src/frontend_common.c
Socket
Main() In while(1) loop in backend.c
client_recv() In backend.c
WriteQuad() In backend.c

Figure 4: CrazyFlie ground station call stack

4.3.3 Functionality

Our design is intended to be able to measure and then control the various directional movements of the quadcopter when utilized in the CPR E 488 lab 4. A user will connect a radio dongle (USB Radio) to the computer and after seeing it is selected when connected to the virtual machine that the lab computers will use, will then follow the steps in the lab to connect to the mini quadcopter and then be able to use the Lab 4 GUI to input and test values for, graph, and analyze the various axis of motion that the quadcopter can move. The system works by measuring how the quadcopter moves and then relaying it to a graphical display which has selectable variables that output the information itself. The input values can be adjusted on a separate 'window' of the GUI. Once this is done, the students will then work on programming the 'controller' screen of the GUI which is the rest of the lab.

4.3.4 Areas of Concern and Development

The current design can be used to complete the lab, but fails to meet the requirements of the teacher, as the backend has some bugs which interfere with data collection, and one of the important testing components of this lab, the test stand, is currently unable to output information that makes it to the GUI where values from it are used as variables.

As such, our primary concerns for this design are fixing some backend errors that we have noticed occurring as we work on them, while also re-enabling the test stand such that it can be used in the lab again. Finally, our last concern is making sure that the project when we leave it is easier for the next sd491 group members to learn about and then work through.

Our immediate plans involve going through all the code that we have and creating debug messages to help us find where, in the large amount of code files, the error is coming from. We do not have immediate questions when it comes to this solution or its implementation, but we do make sure that we are recording problems and solutions as they come up, and when we do have a question that we ask it.

4.4 Technology Considerations

In our project we are utilizing the following technologies:

- Virtual Machine (Software)
  - Project environment and means of deployment
  - GUI & remote connection
  - (-) Prone to connection issues even with simple usb inputs
- CrazyFlie - BitCraze's mini quadcopter model (Hardware)
  - Open source software and hardware for CrazyFlie
- Network technology / Radio Communications (Hardware)
  - The wireless communication method

- Microcontrollers/Low-level programming (Hardware/Software)
    - Drones onboard software
    - (-) Limited potential
- PCB fabrication (Solution)
    - Battery retention

The hardware is a set standard that we are unable to change in our project. Save for maybe the exception of the ‘battery retention racks’ or lack thereof, that can be remedied by adding PCB fabricated designs which can then be attached to the CrazyFlie. The virtual lab environment that is available on the CPR E 488 class website is instead what we will be changing. Specifically, we were editing the code that is part of the lab environment download for the CPR E 488 Lab 4 lab. Utilizing a custom GUI made by previous groups, we have to look through how they made it while we were editing it. The CrazyFlie mini quadcopter is a relatively inexpensive mini quadcopter and as such has small equipment with limited possibilities. In order to communicate with the lab drones, a USB radio is used to communicate with the CrazyFlie, though the radio can also communicate with other equipment that is available in the lab such as sensors which can then be fed back to the CrazyFlie and enable more advanced flying methods. Since it has been decided that we cannot change the hardware, the only changes we can make are through the software, unless we were to choose to start from scratch, but due to our limited timeline, it would be very unwise to choose to do so.

4.5 Design Analysis

Going into this project, while having knowledge that some of it worked, we also knew that there would be some issues and what a few of them are. We began by learning and testing the various features of the lab equipment, lab environment, and code that the lab will be using. We created many types of documents including organizational folders, a FAQ sheet, a project equipment health sheet, and quite a few others to make for better user satisfaction, since we found some issues when we were going through it ourselves. Our design from 4.3 is being utilized and helping to guide us while we tested the current state of the project itself and make notes of the various issues that we encountered with either the actual fix for them written by them, or descriptions of the issue and how we can replicate them. Although, we would only write down something under the fix area when we have actually implemented it.

For our future design, we will continue with bug-fixing and record keeping in order to ensure that future users will have an easier time learning about, setting up, testing out, and finally experimenting with this project.

5 Testing

5.1 Unit Testing

Our infrastructure has several software components, allowing us to perform unit tests on each of the parts individually to ensure that the system as a whole is working properly. These components include the front-end GUIs, the backend modules, and the groundstation. The GUIs can be tested by entering inputs and verifying that the correct outputs are there and the correct thing is sent to the terminal. The backend and ground station are more complex to unit test as one component, so this can be broken down into smaller components that can be tested manually through checking terminal output.

5.2 Interface Testing

Interface testing would be more critical to our design due to the level of complexity in communication between different components. Since we will not be rewriting the firmware code for the CrazyFlie, we would not need to test the interface between the open-source CrazyFlie and Crazyradio. Therefore, we would need to go through a thorough testing for the rest of the components, which are the backend and frontend communication via UNIX sockets. Our method of testing involves inserting print statements when each process sends and receives a message as well as outputting the outputs to a file for debugging purposes to ensure that data is correctly getting passed, which can be done easily with IDEs like Visual Studio Code.

5.3 Integration Testing

Integration testing will be similar to our interface testing such as running the components bit-by-bit/individually and not as an overall large component, and generating tests that will ensure the same output is generated when certain inputs are entered, which can be easily done through the GUI or through some effort through the CLI.

5.4 System Testing

After each component is tested individually, we will move on the test the entirety of the system altogether while running the tests with a CrazyFlie drone and conducting Lab 4 step by step. Through the use of our clearly provided documentation of Lab 4 as well as the GUI, we would be able to complete Lab 4 without encountering any errors or bugs, while improving any quality of life issues that arise.

5.5 Regression Testing

Our current project is based on updating and fixing up the work from teams as far back as 6 years ago. As the project is in an already working state, what we need to do is improve the project and make sure all pieces are working together just like, if not better, than before. We are ensuring that new features do not break the old functionality by utilizing an observe, test, check results, repeat first 3 as needed, and finally push the solutions to a development branch for further testing with the other group member's work. By utilizing this process to test for breakage along with utilizing *git* to observe changes that occur in the workspace, and backtrack when an update breaks something despite testing for otherwise, we are able to ensure stable testing and improvement.

5.6 Acceptance Testing

The first step of acceptance testing is to ensure client approval, which is accomplished by consulting our advisor, Dr. Jones before deploying the software for lab use. If we are given the green light by Dr. Jones, we would continue with the deployment and the actual delivery of Lab 4. We would then request feedback from students and compare it with last year's lab feedback to deduce if there was an improvement in the overall quality of the lab and their experience.

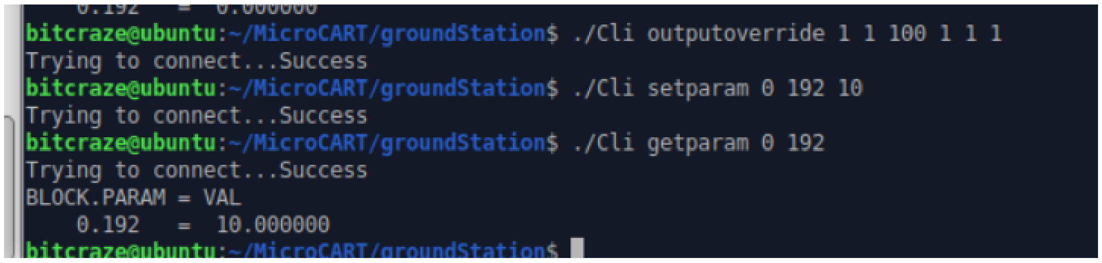
5.7 Security Testing

Not applicable for our project.

5.8 Results

As of now, the CrazyFlie runs perfectly well thanks to the previous MicroCART team, the changes that we made to the system were required changes but it does not result in any testing failures.

An example of the results of testing through the CLI is as shown in Figure 5 below.



```
    0.192   =  0.000000
bitcraze@ubuntu:~/MicroCART/groundStation$ ./Cli outputoverride 1 1 100 1 1 1
Trying to connect...Success
bitcraze@ubuntu:~/MicroCART/groundStation$ ./Cli setparam 0 192 10
Trying to connect...Success
bitcraze@ubuntu:~/MicroCART/groundStation$ ./Cli getparam 0 192
Trying to connect...Success
BLOCK.PARAM = VAL
    0.192   =  10.000000
bitcraze@ubuntu:~/MicroCART/groundStation$
```

Figure 5: CLI results of testing

6 Implementation

Since the main goal of our project is to improve and optimize the entire system, we would need more than to implement those improvements as we would need a thorough understanding and go through an immense debugging process to figure out where the problems lie. Our main plan for implementation of the project would be what was listed in Section 3: Project Plan of this document.

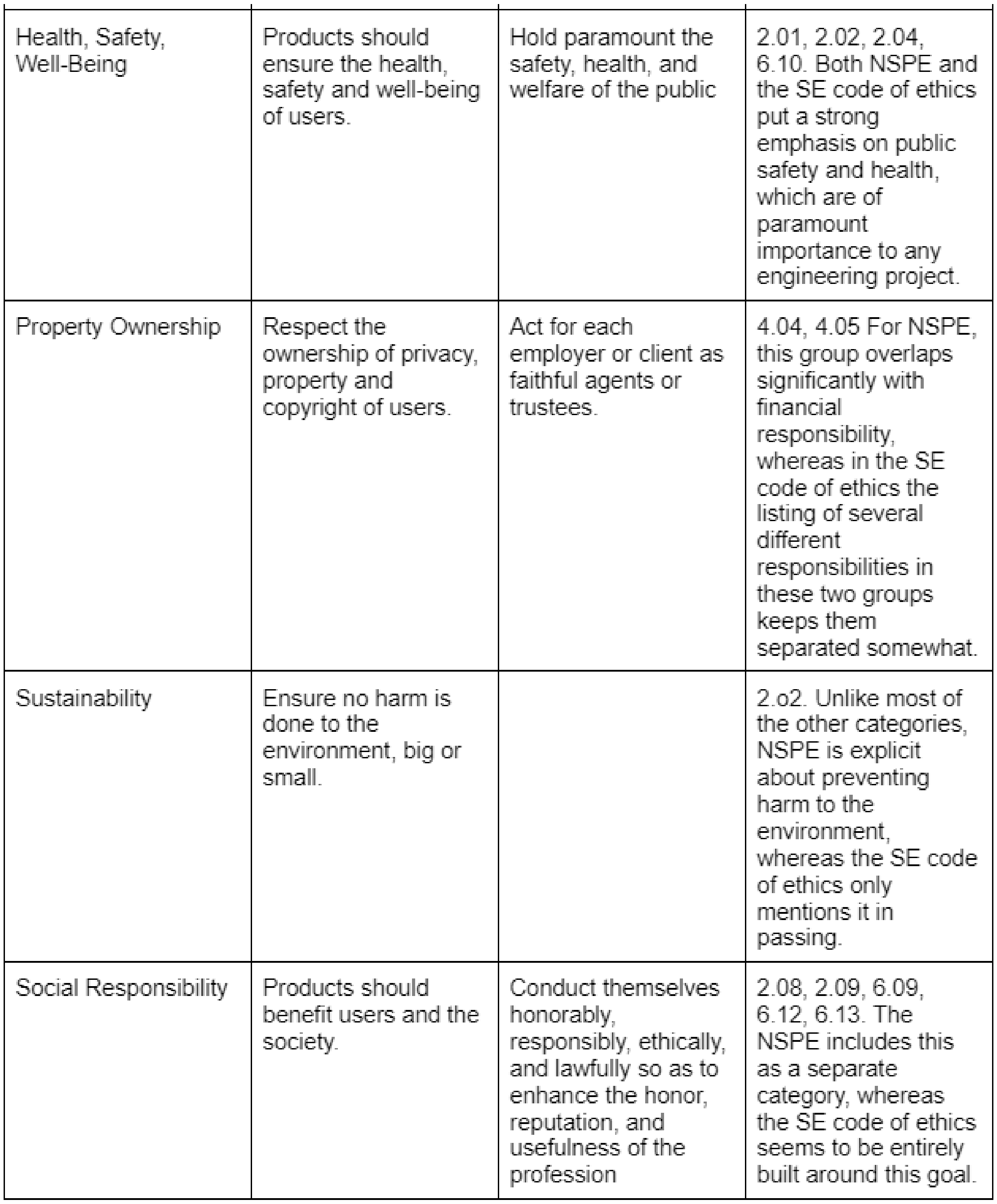
Overall, the progress that we made thus far through testing and debugging include:

- Figured out the main problem that results in "dropped packets" which relates to a log file that is used for graphing implemented by previous MicroCART teams.
- Solved the connection issue between the Arduino and the PC, so what is left now is determine methods to send and receive data via a USB cord.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	Definition	NSPE Canon	SE Code of Ethics
Work Competence	Producing a high quality work, while ensuring work ethics such as work integrity and timeliness.	Perform services only in areas of their competence. Avoid deceptive acts.	1.03, 1.07, 1.14, 1.15, 4.01. The SE code of ethics undertakes a similar effort to ensure those working on projects are well qualified and report timelines, costs and workload correctly and non-deceptively
Financial Responsibility	Producing products within the budget and marketing them at a reasonable cost.	Act for each employer or client as faithful agents or trustees.	1.07, 4.03, 4.04, 4.05, 4.06, 4.07, 4.09 The SE code of ethics covers similar bases as NSPE, but goes further to instruct professionals to avoid other outside influences of the project, such as outside work, illegal software, or other adverse interests. The NSPE does not mention these other possible factors.
Communication Honesty	Report work truthfully which is easily understandable to non-engineers such as stakeholders.	Issue public statements only in an objective and truthful manner. Avoid deceptive acts.	3.01, 3.03, 3.04, 3.05. Both NSPE and the SE code of ethics are explicit about objectivity when related to projects and public statements. The SE code of ethics enumerates the “deceptive acts” to be avoided such as bribery, payback, kickback, or other payment.



Health, Safety, Well-Being	Products should ensure the health, safety and well-being of users.	Hold paramount the safety, health, and welfare of the public	2.01, 2.02, 2.04, 6.10. Both NSPE and the SE code of ethics put a strong emphasis on public safety and health, which are of paramount importance to any engineering project.
Property Ownership	Respect the ownership of privacy, property and copyright of users.	Act for each employer or client as faithful agents or trustees.	4.04, 4.05 For NSPE, this group overlaps significantly with financial responsibility, whereas in the SE code of ethics the listing of several different responsibilities in these two groups keeps them separated somewhat.
Sustainability	Ensure no harm is done to the environment, big or small.		2.o2. Unlike most of the other categories, NSPE is explicit about preventing harm to the environment, whereas the SE code of ethics only mentions it in passing.
Social Responsibility	Products should benefit users and the society.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession	2.08, 2.09, 6.09, 6.12, 6.13. The NSPE includes this as a separate category, whereas the SE code of ethics seems to be entirely built around this goal.

One area that we are doing well is work competence. We are recording our workloads and reporting our individual contributions to the project honestly and non-deceptively. An area that we are working on is the health, safety and well-being of our users which are students because our product does not ensure that no users would be harmed while using our product. In order to improve this shortcoming, we would insert warning labels into the documentation for users to be aware of.

7.2 Four Principles

Four Principles	**Beneficence**	**Nonmaleficence**	**Respect for Autonomy**	**Justice**
Public health, safety, & welfare	Project helps improve the learning of all who are involved	Design promotes safe practices (ie: Test Stands)	Implementation provides a framework, that participants are expected to complete	Design allows for access to all parties
Global, cultural, & social	Brings different communities together to learn	Implementation harms no one indirectly	Design does not affect cultural practices	Benefits are shared equally amongst all parties
Environmental	Mini Quadcopters, are small, decreasing potential environmental impact	Rechargeable batteries and non-toxic, minimally processed materials ensure low environmental impact	Open-source design allows replacement parts to be sourced according to user's desires	Implementation does not harm the environment
Economic	Project teaches job-applicable skills	Project largely uses pre-existing open-source design parts	CrazyFlie is an open-source drone that can be found outside the school	Custom drone will not infringe upon any private sales; CrazyFlie software only affects our items

A broader area context-principle that we are utilizing positively in this project is how our open-source origin design allows for the replacement of parts. Since the CrazyFlie mini quadcopter is made use of within by numerous people for testing, ranging from students to teachers in our context specifically, accidents are bound to occur and can/have caused enough damage that mere tape and glue are no longer enough as solutions. Since there have been some issues that damaged a drone beyond repair already, we have had to look to other locations for replacement pieces, ranging from battery holding implements on top of many of the drones to full body replacement for a drone missing its corner, rotor and all.

One broader context row this project is largely missing or rather lacking is the Global, Cultural, and Social row. Inside of this row, many of the pairs involved in this row are largely due to the scope of the project, which is limited more-so to the college. Different communities are coming to the college to learn equally while our project instead is meant for the audience of a ‘student’ rather than a ‘student of CERTAIN CONTEXT’. Nonmaleficence, Respect for Autonomy, and Justice are all similarly passive with not much being put forward in this context. Thus, while we are certainly not performing in a fashion that hinders this broader context, our project finds it difficult to say that it is supporting it, hence it is a mostly neutral row.

7.3 Virtues

Order, moderation, and resolution[2] are three virtues that are important to our team. To us, order meant how we were to be utilizing the AGILE and Waterfall work methods. By working through items in a predefined manner, we were able to keep track of where we were as a group. Moderation is meant to keep us on track, but not go wild and either do too much or do too little. We each performed a moderate amount of work on our own, but when it came to working together we were rather lacking, and took more time to learn/check up on what each of us were individually working on when it would likely have been more efficient to work as a group for certain task items. Resolution, or more precisely resolve, meant to firmly decide on what we would do, before getting into a task and then finishing it, even if it meant having to ask questions or learn new but necessary information to complete the task. In the future we will be continuing this practice, but will ask our fellow team members when we come across something that we do not understand as this would have made many work halting issues be resolved much quicker.

- Ryan: A virtue that I feel I have demonstrated in this project is perseverance. Perseverance is important in projects such as this because there will be times when a goal may seem impossible or too hard to achieve. This project has brought plenty of challenges and I have had to get through these challenges to reach our goals. Another virtue that I find important that I have not demonstrated yet is collaboration and working with others. I have found it difficult to find time and opportunity to work with my team members as much as this project requires.

This is something that will need to be fixed next semester because it is impossible for engineering projects as large as this to be completed by yourself.

- Daniel: An important virtue to me that I have demonstrated is clear and thorough documentation. Throughout this project, I have been looking through, organizing, and writing documents in order to better guide future students and project members due to the quite bloated project base that has not really been pruned. In comparison, a virtue that I have not demonstrated well is attentiveness to the needs of others, which is largely in part due to not working deeply with all of my fellow team members, something that our sub-groups will have to work together to do as well.

- Jonah: A virtue that has been important to me has been the willingness to learn. This project focused on software development, an area that I was lacking in. The ability to recognize what I did not know and strive to research and gain an understanding of a subject on the fly has been a constant cycle in the development of our project. A virtue that I personally strive for but regarding this project ultimately fell short was time management. I did not spend enough time on this project throughout the semester. This will need to be fixed for next semester. It will mostlikey be caused by better personal planning.

- Yi: A virtue that I have demonstrated is inspiration and motivation. This is important because teams require motivation to work or they will end up with terrible work efficiency. What I did was create a Git Issues board and included tasks with a deadline that we will follow as a team. A virtue that I think is important but have not demonstrated is time management. Fall 2024 has been a very busy semester for me. I had bad task prioritization and scheduling that ultimately resulted in less work done for this project. In the future semester, I will be taking less classes, and manage my time more efficiently through a task scheduler.

8 Closing Material

8.1 Conclusion

In conclusion, our project so far has achieved a better organization of the communal MicroCART project repository, documented and improved the status/health of the CPRE 488 lab 4 materials, and updated the CPR E 488 software that is being utilized in this lab project. When we began this project, our goals were to evaluate and improve the CPR E 488 lab 4 experience, connect the CrazyFlie adapter and groundstation, and connect the test stand tracker to the backend.

Evaluating and improving the CPR E 488 lab 4 experience meant going through this lab ourselves and then doing an evaluation that ranged from how users interacted with the software, to finding, listing, and answering questions that would likely come up throughout the lab experience, and to fixing bugs/issues that would interfere with the student experience or were observed. Overall, we accomplished this goal; reducing the amount of issues that will come up in the next lab experience; improving the guide that students will look through in order to go through the project, now including access to resources and links that users will utilize when stuck on an issue; and for issues that we could not address, how to solve such an issue or avoid them when they could not be solved (ie: USB recognition and docking being an issue with the virtual machine itself and not the lab environment, hence out of our hands). We ran into difficulties when combining the CrazyFlie adapter and groundstation, with both the difficulties in finding the different dependencies that were effected by and affecting these different parts, since the different languages that were originally used to work on each one were different from the other, and will be completing it next semester.

We also worked to finish a tool for students, the current test stand for the crazy flie contains a sensor that could be used for monitoring the crazy flies movement. This sensor had not been integrated into the base station GUI. Attempts to get this tool up and running for next semesters students are looking promising. This will give the student in lab another tool to assist in their learning.

8.2 References

Websites:

[1] "Bitcraze Shop." *Bitcraze Store*, BitCraze, 2024, store.bitcraze.io/. Accessed 17 Nov. 2024.

[2] Franklin, Benjamin. "Benjamin Franklin - 13 Virtues | PDF | Virtue | Science." *Scribd*, 1790, www.scribd.com/document/235479275/Benjamin-Franklin-13-Virtues. Excerpt from Benjamin Franklin's Autobiography.

[3] "How to Register Your Drone | Federal Aviation Administration." *Faa.gov*, 18 Mar. 2024, www.faa.gov/uas/getting_started/register_drone. Accessed 7 Dec. 2024.

[4] Kalachev, Oleg. "Open Source ESP32-Based Quadcopter Made from Scratch." *Arduino Project Hub*, 6 Jan. 2024, projecthub.arduino.cc/okalachev/flix-58fe43. Accessed 26 Sept. 2024.

[5] Liang, Oscar. "Review: Holybro Kopis Freestyle 4-Inch FPV Drone." *Oscar Liang*, 27 July 2021, oscarliang.com/holybro-kopis-freestyle-4-inch/. Accessed 26 Sept. 2024.

[6] Nast, Condé. “The DJI Mini 4 pro Is a Small Drone with Huge Appeal.” *WIRED*, 23 Mar. 2024, www.wired.com/review/dji-mini-4-drone/. Accessed 26 Sept. 2024.

8.3 Appendices

Microcart Git Repository: https://git.ece.iastate.edu/danc/MicroCART

9 Team

9.1 Team Members

- Daniel Zaucha
- Jonah Upah
- Ryan Lowe
- Yi Hang Ang

9.2 Required Skill Sets for Your Project

- Coding Skills (At least one of: Python | C | C++)
- Networking knowledge
- Embedded systems
- Embedded programming
- Able to utilize Linux / Command Line Interface / Make

9.3 Skill Sets covered by the Team

Green = Have || Yellow = Some exposure || Red = Inexperienced

	Coding languages	Networking knowledge	Embedded Systems	Embedded Programming	Linux / CLI / Make
Daniel	🟩	🟨	🟩	🟨	🟩
Jonah	🟨	🟥	🟩	🟩	🟨
Ryan	🟩	🟨	🟩	🟩	🟨
Yi	🟩	🟨	🟩	🟩	🟩

9.4 Project Management Style Adopted by the team

Hybrid method of Waterfall method and Agile method for project management.

9.5 Initial Project Management Roles

Daniel Zaucha:	Client interaction, Communications Lead
Jonah Upah:	Hardware Lead, Team Secretary
Ryan Lowe:	Technical Advisor
Yi Hang Ang:	Software Lead

9.6 Team Contract

Team Members:

1) Daniel Zaucha 2) Ryan Lowe

3) Jonah Upah 4) Yi Hang Ang

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    - Advisor meetings: In-person: Wednesday: 10:00 - 11:00 AM
    - Work sessions: Thursday: 2:00 PM - 5:00 PM. Sunday 2:00 PM - 5:00 PM
    - Location: Coover Hall 3050

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    - Meetings are done in person, unless someone cannot come in person.
    - Discord server set up with all group members to communicate when not in person.

3. Decision-making policy (e.g., consensus, majority vote):
    - Majority votes are required for decision-making. If there is no majority or votes are split, we will discuss as a team and try to reach a consensus from group discussion over the issue (OR get Dr. Jones as a tie-breaker vote).

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    - Jonah will set up meeting notes and record action items during the meeting. Notes will be kept in a shared folder.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    - Everyone will be expected to attend scheduled meetings unless extenuating circumstances come up. If this happens, the team member will inform the group of this.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    - Everyone should contribute to completing team assignments, ideally before deadlines, but preferably a few days before. If someone is not going to get something done because of circumstances, be sure to contact the group for assistance.

3. Expected level of communication with other team members:
    - Discord is how we communicate with each other about meeting and tasks. During the working day responses are expected, subject to change.

4. Expected level of commitment to team decisions and tasks:
    - Everyone puts in an equal amount of time, and thought behind decision making.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
    - Jonah Upah: Team secretary, meeting organizer
    - Yi Hang Ang: Hardware design
    - Ryan Lowe: Software Lead
    - Daniel Zaucha: Client interaction

2. Strategies for supporting and guiding the work of all team members:
    - The report document will track tasks and time.
    - Meeting will be a place to discuss concerns about the time line.
    - Report doc will also have new issues.

3. Strategies for recognizing the contributions of all team members:
    - Contributions of team members will be noted on the weekly reports and discussed in meetings.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    - Jonah: EE guy, PCB design,
    - Yi: Embedded Systems datasheet guy, experience in coding
    - Ryan: Lots of experience with coding, linux experience
    - Daniel: S E into CPR E, Coding experience, communication skills

2. Strategies for encouraging and supporting contributions and ideas from all team members:
    - Allowing open discussion during meeting
    - Team discussion on any contribution.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
    - First off, bring up the issue to the group.
    - Vote or get Dr. Jones to act as a tiebreaker.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    - Have a working prototype/design by the end of this semester (subject to change depending what is expected from this project)
    - Complete all documentation before deadlines.

2. Strategies for planning and assigning individual and team work:
    - Work will be assigned in meetings. We will estimate how much time work should take and assign members with a similar work load so that everyone is working similar hours. Team work will also be assigned in the weekly meetings.

3. Strategies for keeping on task:
    - Weekly meetings will point out tasks that present an issue.
    - Report Doc will document what everyone is working on and needs to accomplish.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
    - Judiciously; as decided by the group at the time as necessary.
2. What will your team do if the infractions continue?
    - Notifying the advisor/instructors if there is no communication.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*

b) *I understand that I am obligated to abide by these terms and conditions.*

c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Jonah Upah	DATE 9/18/2024
2) Yi Hang Ang	DATE 9/18/2024
3) Daniel Zaucha	DATE 9/18/2024
4) Ryan Lowe	DATE 9/18/2024