MicroCART

Final Design Document

Team Number: 43
Client: Dr. Phillip Jones
Advisers: Dr. Phillip Jones
Team Members/Roles
Brandon Cortez - Test Stand Lead
Reid Schneyer - Test Stand Sub-team
Colton Glick - Firmware Lead, Git Manager
Ellissa Peterson - Team Member
Ryan Hunt - System Architect
Carter Irlmeier - Web Manager, Lighthouse Sub-team
Zachary Eisele - Groundstation Lead, Co-System Architect
Team Email: sdmay22-43@iastate.edu
Team Website: https://sdmay22-43.sd.ece.iastate.edu
Revised: 4/28/2022

Table of contents

Executive Summary

Development Standards & Practices Used

- CI/CD pipeline in git
- 3D printing guidelines
  - IEEE P3030
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7308141
- Bluetooth/Radio communication standards
  - https://standards.ieee.org/standard/802_15_1-2002.html
- Crazy Real Time Protocol (CRTP)
  - Packet protocol used by Crazyflie
  - https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/crtp/

Summary of Requirements

- Test stand must record and transmit movement data to a host computer
- GUI must display all relevant data
- Drone firmware must be modular so that control logic can be removed and substituted
- Develop software to stabilize, and communicate with the mini-quadcopter.
- Develop testing rigs to allow the team and users (CPRE 488 students) to interact with the mini-quadcopter. For example, for tuning control algorithms to stabilize the mini-quadcopter
- Laboratory document instructions need to be clearly written to guide CPRE 488 students while working with the mini-quadcopter
- Demonstration of the mini-quadcopter's autonomous capabilities should be significant and show the team's technical abilities

Applicable Courses from Iowa State University Curriculum

- COM S 309, 319
- CPRE 288, 458, 488
- EE 333

New Skills/Knowledge acquired that was not taught in courses

- 3D Modeling Software
- Software Architecture of the Crazyflie Drone
- Operation of camera rig tracking system
- PID Controllers
- Socket Communication Protocols
- OpenVR API
- QT for GUI development
- Multi-Threaded, responsive graphical user interface

List of figures/tables/symbols/definitions

1. Team

1.1. Team Members

Brandon Cortez - Test Stand Lead
Reid Schneyer - Test Stand Sub-team
Colton Glick - Firmware Lead, Git Manager
Ellissa Peterson - Team Member
Ryan Hunt - System Architect
Carter Irlmeier - Web Manager, Lighthouse Sub-team
Zachary Eisele - Groundstation Lead, Co-System Architect

1.2. Required Skill Sets

- 3D Design & Printing
- GitLab code/issue management
- Software Architect
- Firmware Development
- GUI Frameworks
- PCB Design & Electronics Prototyping

1.3. Skill Sets covered by the Team

- 3D Design & Printing - Brandon, Reid
- GitLab code/issue management - Colton, Ellissa
- Software Architect - Colton, Zach
- Firmware Development - Ryan, Zach
- GUI Frameworks - Ellissa, Carter
- PCB Design & Electronics Prototyping - Reid, Brandon

2. Introduction

2.1. Problem Statement

CPRE 488 students need a functional drone system where they can write and test their own control logic for in-class labs to learn more about advanced embedded systems. We developed a platform to facilitate experimentation for these students. Additionally, we created impressive quadcopter demonstrations to show off to potential Iowa State students and industry contacts.

2.2. Requirements & Constraints

- Test stand must record and transmit rotational position data to a host computer
- GUI must display and graph drone position information
- Drone firmware must be modular so that control logic can be removed and substituted by student's code
- Must develop software to stabilize, and communicate with the mini-quadcopter.

- Must develop testing rigs to allow the team and users (CPRE 488 students) to interact with the mini-quadcopter. For example, tuning control algorithms to stabilize the mini-quadcopter

2.3. Engineering Standards

- Must be compatible with existing Crazyflie standards and systems
  - Low-level C on the quadcopter, and Python for client application
- CI/CD pipeline in git
- 3D printing guidelines
  - IEEE P3030
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7308141
- Bluetooth/Radio communication standards
  - https://standards.ieee.org/standard/802_15_1-2002.html
- Crazy Real Time Protocol (CRTP)
  - Packet protocol used by Crazyflie
  - https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/crtp/

2.4. Intended Users and Uses

- Students taking CPRE 488 in the Spring 2022 semester will benefit from our project as the lab uses quadcopters as a teaching tool.
- Iowa State University is also a potential beneficiary, a demonstration of the drones will attract potential students and corporate representatives

3. Project Plan

3.1. Project Management/Tracking Procedures

Our team used a waterfall+agile management style. The nature of our project was such that we could immediately begin adding features to the Crazyflie, allowing an agile approach to work well. We planned to track our progress in the project through a GitLab kanban board. Tasks/issues would be created on the board as we began development, which would then be assigned to different members of the team. Additionally, milestones with due dates were created to make sure we kept a good pace as the project progressed.

3.2. Task Decomposition

Tasks Completed:

1. Investigate Crazyflie firmware architecture
   a. Learned how to modify and flash a new firmware to the Crazyflie
   b. What is the current architecture structure?
   c. Can the control code be easily modified?
   d. How easy is the control code to understand for new users?
2. Modified Crazyflie firmware to be as modular as possible

    a. Abstract the control code to a standardized interface to allow other control algorithms to be easily implemented through an adapter architecture
    b. Rewrite the existing control code to utilize the new interface
3. Write a basic PID control loop to maintain a stable hover, using the new interface
4. Develop ground station software to communicate with and control Crazyflie
    a. Start with a command-line interface on Linux
    b. Build a GUI/frontend once the backend is mostly working
5. Develop test stand hardware
    a. Determine what electronics will be used to record & communicate data
    b. Integrate chosen electronics into test stand for data collection
    c. Design and print test stand model to mount Crazyflie
6. Develop test stand software to measure and log rotation of the Crazyflie while held in test stand
    a. Should collect and record all desired data from the Crazyflie in real-time
    b. Should communicate with the ground station to allow for easy saving of log data
7. Write lab instructions and documentation for interfacing and using the modified Crazyflie
    a. Basic quick start guide
    b. Detailed proposed lab activities
8. Develop an demonstration that will be performed by the Crazyflie
    a. Research possible routes to take for demonstration
    b. Implement chosen route

3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

Project Milestones;

1. Working test stand prototype
    a. 3D printed and assembled
    b. Electronics selected and assembled
    c. Firmware is written and reporting back
2. Custom firmware running on Crazyflie
    a. First modified firmware running on Crazyflie
    b. Control software abstracted with adapter interface
    c. Existing control code running through adapter
3. Custom ground station CLI
    a. Control commands can be sent to Crazyflie and an acknowledgment is sent back
4. Ground station GUI based on CLI
    a. Basic GUI that sends pre-configured commands through the CLI

   b. More advanced GUI that displays flight data and allows for gamepad controls
5. Crazyflie flight demonstration
   a. Interacting with lighthouse system
   b. Running basic autonomous control script
   c. Complex autonomous drone demonstration

3.4. Project Timeline/Schedule

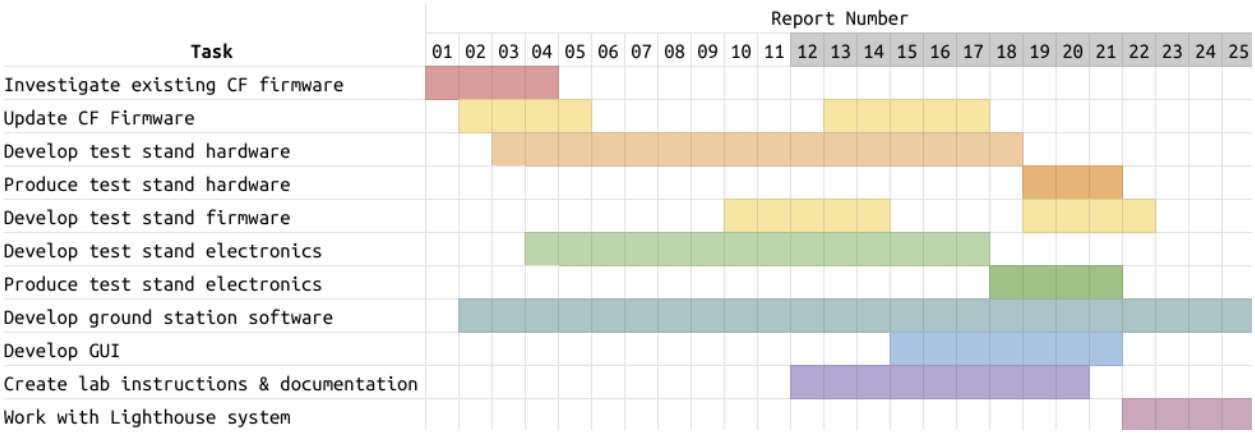

Table 1. Project plan Gantt chart of milestones and deliverables.

3.5. Risks And Risk Management/Mitigation

Risks:

1. Firmware is much harder to adapt to an adapter architecture, takes longer than expected, 60%
   a. Mitigation plan: Begin researching and modifying the Crazyflie firmware as soon as possible. May need to push back tasks that are dependent on this, the different control algorithms, and lab instructions.
2. Can't get voltage divider ratio to work for test stand electronics, 60%
   a. Use a 5v logic level microcontroller to read test stand data
3. CLI cannot communicate with Crazyflie, 30%
4. GUI takes too long to create, 40%

3.6. Personnel Effort Requirement

Task	Time Estimate (Hours)	Explanation
Investigate existing firmware	35	Depends on firmware complexity
Create control firmware	50	Depends on firmware complexity
Develop ground station software	200+	Create CLI & GUI for communicating with Crazyflie
Develop test stand hardware	32	Design & build test stand
Develop test stand software	30	Write firmware for test stand electronics
Write lab instructions	25	Create assignment for CPRE 488 lab
Create autonomous demonstration	40	Become familiar with system and implement

Table 2. Table of Personnel Effort Requirements by milestone

3.7. Other Resource Requirements

- Access to several Crazyflie drones to test and develop on
- Development computers running Linux to build and test the system on
- SIC access for 3D printing components of the test stand
- RC controllers
- Test Stand Control board components (see Bill Of Materials in appendix)

4. Design

4.1. Design Context

4.1.1. Broader Context

Our project has a fairly narrow context, all things considered. We're designing for Iowa State University students interested in learning more about complex embedded systems, specifically those taking CPRE 488 in the spring 2022 semester. The project addresses the societal need of needing skilled embedded programmers as more and more devices are created and manufactured.

Area	**Example**
Public health, safety, and welfare	The drone could cause minor injury if it collides with an individual
Global, cultural, and social	The drone will be used to teach engineering students more about advanced embedded systems and possibly inspire others to take interest in computer engineering
Environmental	The batteries the drones use are not great for the environment and can catch fire
Economic	An impressive drone demo could attract potential corporate or private donors, and potential future students

Table 3. Table of Broader Design Context

4.1.2. User Needs

CPRE 488 needs a functional drone system where they can write their own control logic for in-class labs to learn more about advanced embedded systems as well as a way to test and tune this control algorithm for table flight of the drone.

4.1.3. Prior Work/Solutions

We followed previous work that had been done for years, most recently by the 2020 MicroCART team. This is advantageous in that we have code and repositories (https://git.ece.iastate.edu/danc/MicroCART/-/tree/2020-team-final-state) to look at, but we definitely are operating a different project. We dealt with a much smaller version and different kind of drone than what has been used previously, which definitely differentiated our work from others.

4.1.4. Technical Complexity

The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles. These components/subsystems are:

- Groundstation
  - includes computer software skills
- Firmware flashed to the Crazyflie
  - includes computer engineering skills
  - Will utilize several popular software engineering principles such as Separation of Concerns, Modularity, Abstraction, and Incremental Development
- Test Stand
  - involves 3D modeling,
  - electrical engineering
  - embedded systems

4.2. Design Exploration

4.2.1. Design Decisions

- Wifi integration decision
- What exactly to do for the demonstration
- GUI layout
- What sensors will be embedded in the test stand

4.2.2. Ideation

We identified potential options for sensors needed through client and group requirement discussions and brainstorming:

- Sensors
  - Rotation
  - Vibration
  - Voltage
  - Gyroscope
  - IR

4.2.3. Decision-Making and Trade-Off

Our process for creating these pros and cons was through discussion with the client and group members to make sure we covered all areas of concern.

- Affordability (keeping in mind project budget),
- Availability (what is in stock and what the lab already has),
- What fits our design
- What works with our firmware

4.3. Proposed Design

- We have a working test stand that lets students measure roll, pitch, yaw and the rates for each and sends that information to the ground station
- We have a ground station that allows us to communicate with the drone to receive information from its sensors and send setpoints or parameters.
- We have a GUI for the ground station which allows students to graph the data from the sensors or from their own firmware.
- We have created our own PID controller for the Crazyflie and made a stripped down version for the students to write their own control algorithms
- We have written the lab for the students where we introduce them to our system then they tune their own PID values and write the firmware which includes the PID equations
- We have an autonomous demonstration working as well as a way to control the drone using a VIVE controller

4.3.1. Design Visual and Description

Test Stand Model:

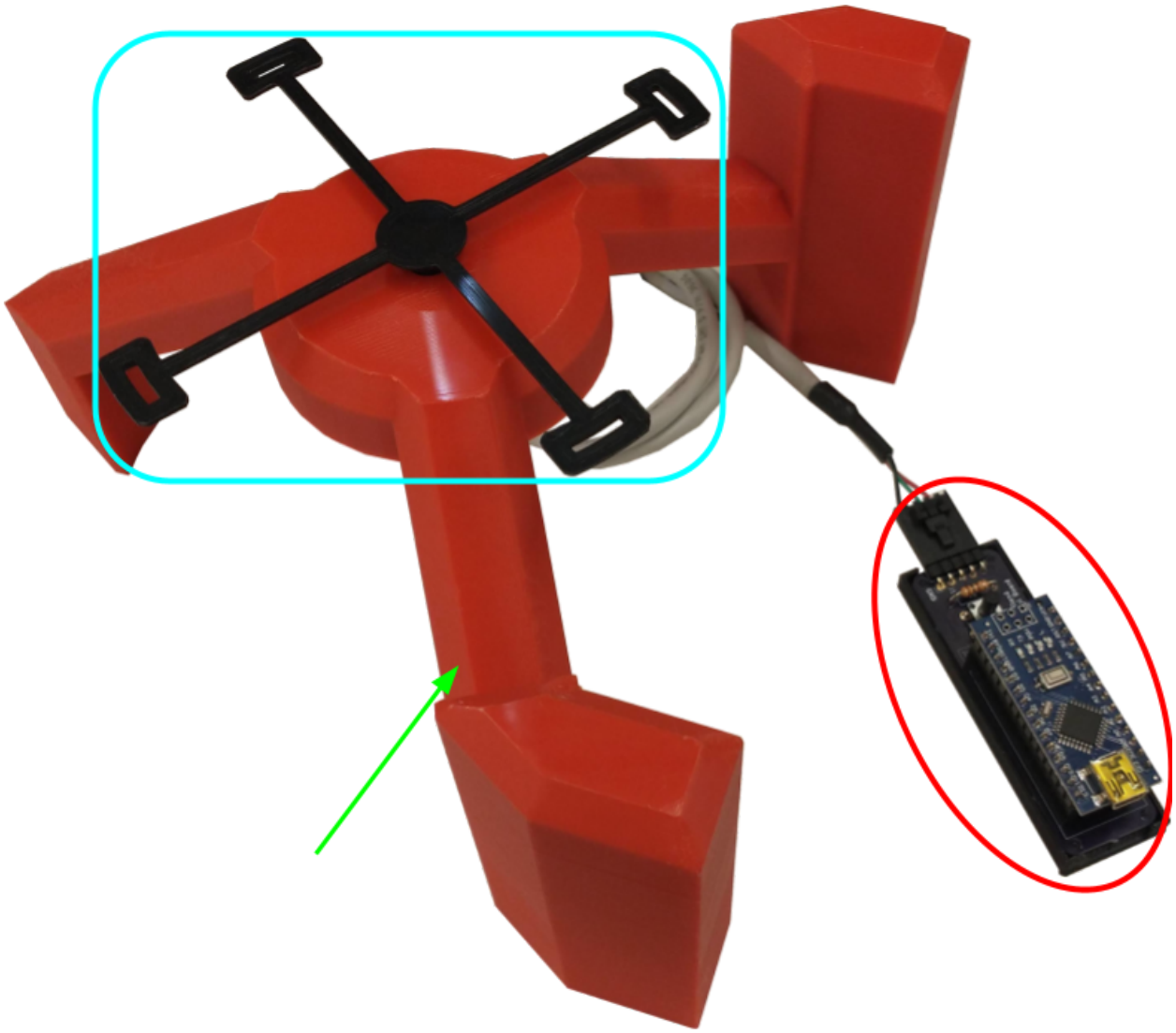

Figure 1. Crazyflie test stand and control board

The test stand assembly consists of 3 major components. Circled in red, the control board provides +5V and ground to the MA3 absolute rotary encoder mounted on the underside of the test stand. It reads the analog voltage value provided by the encoder, then translates that value into a rotational position, which is sent over USB serial to the ground station computer. The 6mm pushbutton on the control board serves a dual role. When held, it toggles the control board from reporting the rotational position to reporting the rotational rate, or vice versa. When in position mode, a short press of the button “zeros” the reading output, similar to the tare function on a digital scale.

The blue rounded rectangle shows the test stand quadcopter mount. This component has four holes that the plastic quadcopter legs fit into, and it holds the drone to the rotary encoder, preventing unintended liftoff. This component is fitted to the shaft of the MA3 absolute rotary encoder, and rotates the encoder shaft as the drone rotates.

The third major component of the test stand assembly is the test stand, indicated by the green arrow. The test stand houses the body of the MA3 absolute rotary encoder and allows the quadcopter to produce thrust and execute rotational maneuvers to test for yaw rate and position. The test stand is also designed to allow it to be rotated 90° and set on its side to allow the quadcopter to be tested on a different axis for pitch and roll values using a side-mounted quadcopter mount.

Figure 2. Crazyflie 2.0 mini quadcopter

The Crazyflie is a COTS open source development platform quadcopter. This is what the students tested their control algorithms on. We modified the firmware of the Crazyflie so that the PID controller was broken into modular parts and then stripped down for the students to fill out. We also created our own PID controllers from the modularized code to prove that it could be done.

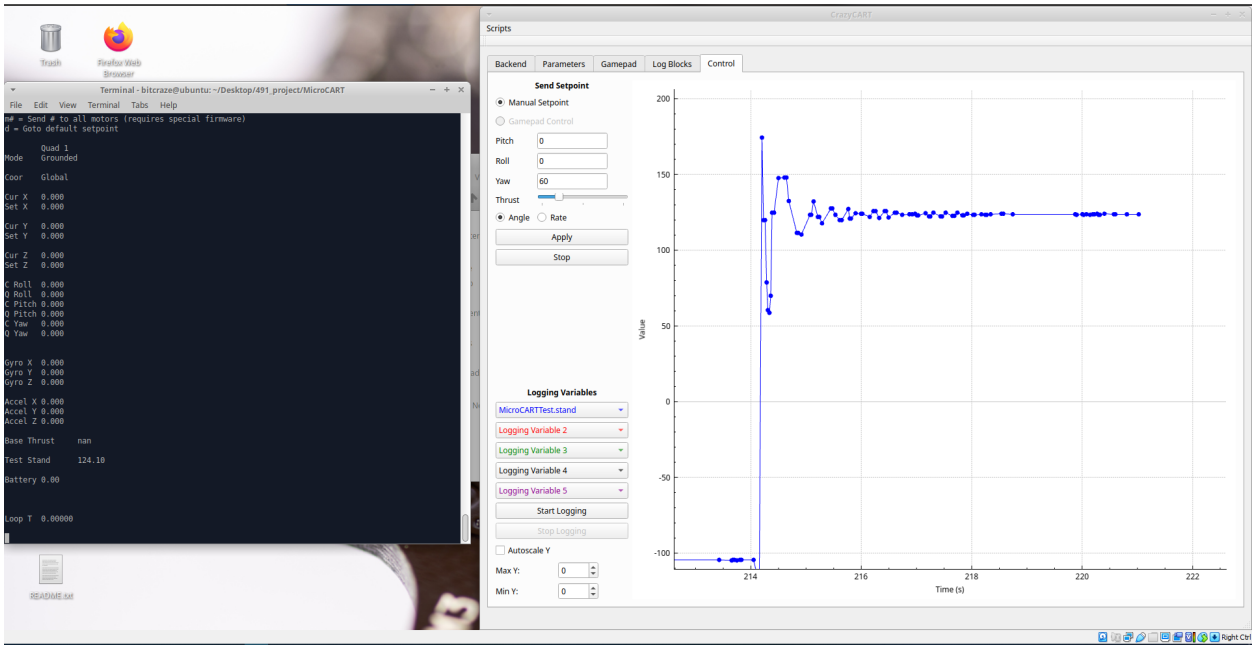

Figure 3. Groundstation and CrazyCart GUI

The groundstation backend was initially created by a MicroCart team a few years ago as a way to communicate with the larger drones they had via CLI commands, an adapter was made by a more recent team which utilizes the ground station but communicates with the Crazyflie. We made additions and updated outdated code to support the functionality that we wanted. We then created a GUI, that works with the backend, that would allow for easier means of communication with the drone and display the data that was being communicated.

4.3.2. Functionality

Our design lets students create and test their own control logic for the drone and gather data from the test stand as well as to show off when prospective students visit.

Our design fits the functional and non-functional requirements. Of course there could be improvements such as a few more features that would make students' lives easier when writing control algorithms and a few bugs that are documented. As this is a continuous project there will be ever changing and adding of requirements to make the best lab possible. As for this year's lab all of the requirements were met.

4.3.3. Areas of Concern and Development

Since we have completed our project and almost all of the students were able to complete the lab we met the needs and requirements there. Since

this is an ongoing project and there will be a team improving upon what we have created this year, the needs have shifted into needing documentation to integrate future teams into what we have done easily.

Our immediate plans are to document every part of our project in gitlab by giving instructions of how they were developed, how to continue development, and how to use what we have created. There are also a few students who have indicated that they want to work on the project next semester so we are in the process of showing them what we have made and giving them hands-on experience beforehand.

4.4. Technology Considerations

In working on our senior design project, we had to determine which type of protocol we would use to communicate between the ground station and the Crazyflie drone. The ground station that Bitcraze provides uses a radio antenna, but Professor Jones wanted us to look into using either Bluetooth or WiFi to communicate in our ground station software. While the Crazyflie does have an existing Bluetooth IC, we would have to use a separate board for WiFi communication, such as the ESP8266. Professor Jones' concern was that Bluetooth might have more latency than going with the Wifi option, since we could use socket communication over Wifi. We ended up sticking with radio communication as that was already implemented on the ground station and Crazyflie drone side.

4.5. Design Analysis

Yes, our proposed design worked as the students were able to complete the lab with all of the functionality that was necessary. We did get a lot of feedback from the students with what features could be added/improved upon. We have a list of these which we discussed with ourselves and with our client on which were the most important then implemented from there.

4.6. Design Plan

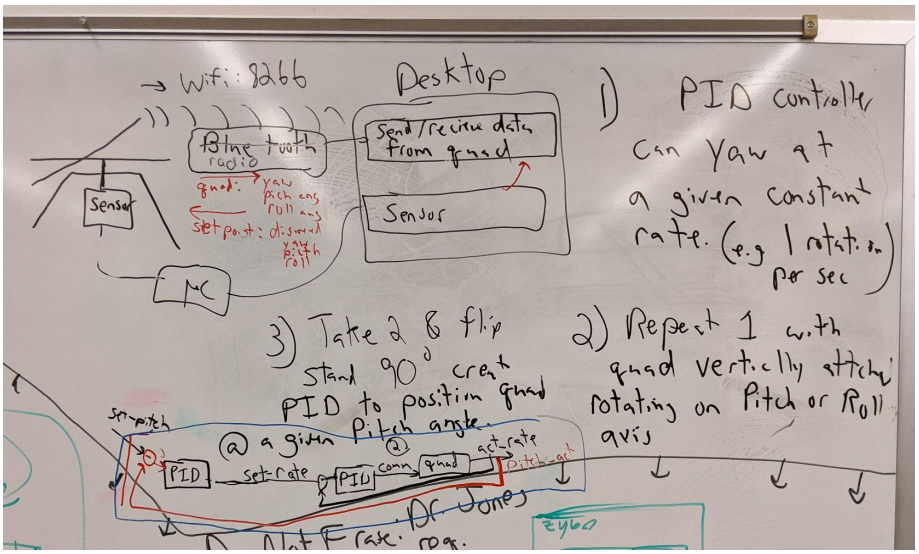

Figure 4. Module Interface Diagram

Our design plan is divided into three teams, the ground station, the test stand, and the firmware. I will first go over the individual requirements for each team then the requirements of all of the teams together. The plan for the ground station is to start with the previous year's MicroCART ground station as there was an adapter made for the Crazyflie last year, then improve upon it with any information that would be beneficial for students to use in the 488 lab. For the test stand the design plan is to first create a test stand similar to one created from a previous team then see what could be improved upon with it, make a new design with the previous flaws in mind. This will be continued till the design fits all of the needs. The next step would be adding sensors onto the stand so that students can get measurements from the Crazyflie quadcopter and create hardware that will display the sensor data. The design plan for firmware is to become familiar with the current firmware and figure out how the PID controls work, then implement our own PID controller so that it could then be stripped down into a template for 488 students. The plan for making these parts work together will be first setting up communication between the Crazyflie and ground station with data being sent and read both ways. The second part of communication would be the groundstation getting data from the test stand sensors and displaying that data. To begin, this data link will be through the Crazyflie radio. In future iterations we plan to switch to bluetooth or wifi or radio communication to standardize the system more.

5. Testing

5.1. Unit Testing

Quadcopter software:

- Any units inside of the quadcopter coding can be tested, this would include a variety of functions. An example would be a flying algorithm. This can be tested by making a testbench for the function and feeding it numbers that we would know the outcome for, then comparing to

expected output. This will eventually also be able to be tested on our own testing station where we can verify the actual response of the quadcopter from these equations. We also used the built in unit tests of the Crazyflie which test basic logging parameters verification as well as compilation error checking..

Test Stand:

- The unit testing for this part of the project would pertain to making sure we are getting correct data from the sensor. This can be done by measuring the values vs expected values of what the sensor is measuring.

GUI ground station:

- The unit testing for the GUI will be for it to display the proper information, this may include some calculations which would also be a part of the unit test. These would be tested by writing a testbench and making sure the values match the expected

5.2. Integration Testing

The integration in our design was between the GUI ground control and the test stand and between the quadcopter and the GUI ground control. We tested the interfacing with the GUI ground control and test stand by sending data from the test stand to the GUI and making sure it gets what was actually sent. We tested the communication between the quadcopter and the GUI by sending a command with the GUI and checking if the quadcopter responds accordingly.

5.3. System Testing

The system we are testing in our design is from the base station to the GUI then to the quadcopter. This was tested by first sending a command from the GUI to the quadcopter where then the test stand sensors recorded data from the movements of the quadcopter then sent that data to the GUI.

5.4. Regression Testing

When creating new additions we used version control in conjunction with testing to ensure that all bugs and issues are resolved before the master branch is updated.

5.5. Acceptance Testing

We demonstrated that our design met the requirements to our client by giving in-person demonstrations of our project progression. We involved our client, as well as the TAs, in this by guiding them through how our solution works first by individually showing them each part then how they all work together.

5.6. Results

The results of our testing showed that each part of the design works as stated in the requirements. The ultimate test of the students completing the lab was a success as almost every group was able to complete the lab. We also received

feedback of what students thought about the lab, which was relatively successful, this proves the usefulness of our project.

6. Implementation

6.1. Crazyflie Firmware

The Crazyflie provided an excellent existing base to work from. The code is published as an open source project by Bitcraze and is available on Github (see references at the end of the report). The firmware is written in low level C and allows for direct control over the Crazyflie's microcontroller. However, the firmware is also written in a modular manner that allowed our team to modify the control algorithms for the CPRE 488 students. Further work was done by our team to simplify the structure around the stabilization module to allow students to focus on the PID control algorithms. We simplified the stabilization by renaming variables and removing logic where it was no longer necessary. Additional work was done to implement firmware flashing over USB by automatically placing the quad into DFU mode when a special USB packet was sent to it. This feature was not fully utilized in the CPRE 488 lab but was contributed to the official Crazyflie open source project.

6.2. Test Stand Firmware

The test stand firmware is fairly simple, and is written using the Arduino IDE. It reads an analog value from the pin connected to the MA3 absolute rotary encoder, then scales the 0-1023 analog value to a -180 to 180 value corresponding to the degree position of the encoder. When the button on the PCB is pressed, it "zeros" the reading. When the button is long pressed, the firmware switches into rotation rate mode, which gets a moving average of the rate of the encoder rotation. Regardless of mode, the firmware then sends a single double precision value over serial every 100 milliseconds.

6.3. Ground station

The ground station consists of three parts: the MicroCART ground station, Crazyflie adapter, and the Crazyflie ground station. The MicroCART ground station was made by student's from previous years and was originally meant to communicate with custom quadcopter's they had made. This ground station needed to continue to be used so that in the future it could support multiple different types of quadcopters at the same time. In this project the main utility of the MicroCART ground station was command processing. The MicroCART ground station then sent commands to the adapter. The adapter was in charge of translating packets from the MicroCART configuration to the Crazyflie configuration. The Crazyflie ground station was the part of the system that communicated with the quadcopter itself. It would receive command packets from the adapter and send them to the Crazyflie, and receive logging and parameter information from the Crazyflie and store it for later use.

7. Professionalism

7.1. Project Specific Professional Responsibility Areas

Professional Responsibility	Project Application
Work Competence	**Why/why not?** - Our project will be used by future students thus it needs to be of high enough quality that it can be comprehended in the future to aid in education. **Performance(Medium):** We are continuously working to better our code but our documentation could be improved to help with future comprehension.
Financial Responsibility	**Why/why not?** - We are not buying or selling any products of significant value. Performance(N/A):
Communication Honesty	**Why/why not?** - We've had to communicate our progress to our contact on a weekly basis. **Performance(High):** We write up weekly status reports, as well as communicate frequently in discord and through email with our contact.
Health, Safety, Well-Being	**Why/why not?** - We do have potentially harmful equipment being used in our workroom. **Performance(High):** Safety protocols are in place and used during development and testing.
Property Ownership	**Why/why not?** - We use university-provided resources and technology in our development laboratory. **Performance(High):** We have been respectful of the lab & equipment, especially the sensitive drone electrical components.
Sustainability	**Why/why not?** -

	Sustainability is not a large part of our project as we use primarily small amounts of standard-grade plastic components. **Performance(N/A):** The scope of our project uses limited supplies and creates minimal waste.
Social Responsibility	**Why/why not? -** Our project does not have a high social responsibility because it is going to be used primarily only by future students. **Performance(High):** Although we do not have a high social responsibility, we are developing our project with the safety and education of the user in mind.

Table 4. Table of project-specific professional responsibilities

7.2. Most Applicable Professional Responsibility Area

Professional Responsibility:
Communication Honesty

Description, Demonstration, and Impacts:
For our project communication honesty consists of reporting our work to our client truthfully, transparently, and without deception. We demonstrated this responsibility by providing detailed reports to our client on a weekly basis that communicated our team's progress, what we achieved, what we are spending our time and resources on, and what our plans for the future are. Communicating clearly and honestly in this manner has allowed us to receive critical feedback from our client on our performance in order to improve our team's performance as the project progressed.

8. Closing Material

8.1. Conclusion

Our solution successfully fulfilled our client's product requirements. Our system allowed CPRE 488 students to write and test the controls of the Crazyflie drone from a ground station, while sensors in the test stand monitor the movement and behavior of the drone.
Some aspects of the project that we would have done differently include,

- Having a more in depth discussion with client about project requirements early on
- Allocating less time to research and more time to development
- Starting development on the ground station GUI earlier

- Using a web based GUI to improve performance and usability

8.2. Design Evolution since 491

At the end of last semester we had the following completed:

- Prototype of the test stand
- Prototyped control board on a breadboard
- Able to send rate setpoints via CLI from ground station
- Created a copy of existing controller as student controller in drone firmware

At the end of this semester we have completed:

- Test stand that allows for measurement of attitude and attitude rate
- Control board PCB that translate analog value to rotational position and rate
- Ability to get/set parameters, receive log data, send angle, rate and position setpoints from the ground station via CLI
- GUI that connects to the ground station and can perform the CLI commands as well as graph the logging data
- Created out own version of the firmware that was later stripped down and used as a template for students to write their own control logic
- Created multiple demonstrations with the drones ranging from autonomous flight to flight controlled by vive controller

8.3. References

US Digital, “MA3 Miniature Absolute Magnetic Shaft Encoder”, MA3 datasheet, Jun. 2021 [Accessed 06-Dec-2021].

“Bitcraze,” *GitHub*. [Online]. Available: https://github.com/bitcraze. [Accessed: 06-Dec-2021].

“Distributed Autonomous Networked Control Lab / microcart,” *GitLab*. [Online]. Available: https://git.ece.iastate.edu/danc/MicroCART. [Accessed: 06-Dec-2021].

“Start here,” *Bitcraze*. [Online]. Available: https://www.bitcraze.io/documentation/start/. [Accessed: 06-Dec-2021].

“CRTP - Communication with the Crazyflie,” *Bitcraze*. [Online]. Available: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/2021.06/functional-areas/crtp/. [Accessed: 06-Dec-2021].

9. Appendices

9.1.1. Operation Manual

You can find documentation for our project in our gitlab wiki here:

https://git.ece.iastate.edu/danc/MicroCART/-/wikis/home
We have also included the instructional lab document located at the end of this report.

9.1.2. Alternate Versions

- **WiFi communication**: During planning, our client initially wanted the Crazyflie to communicate over WiFi using TCP or UDP sockets. This would have standardized the communication protocol and allow the Crazyflie quadcopters to be operated without the need of the Crazyradio. This would require an additional board to allow for WiFi communication and was theoretically possible. However, we underestimated the amount of work that would be required in implementing WiFi communication and did not have enough time to include it. It was dropped in favor of more critical features for the CprE 488 lab.
- **Bluetooth communication**: Similarly to WiFi communication, our client wanted to use bluetooth communication to control the Crazyflie quadcopter. The Crazyflie already has the capability to communicate over bluetooth with a smartphone, However, again we underestimated the complexity of the system. The communication within the Crazyflie firmware was far more complex than initially anticipated. Thus we had to drop this in favor of more critical features for the CprE 488 lab. Our final design utilized the already tried and tested Crazyradio for communication.
- **OptiTrack System**: Coover 3050 contains a 12 camera, ceiling mounted OptiTrack system. This is an optical motion capture system that uses IR markers to detect objects within its range. The system in 3050 is outdated, as is the software that they use. “Tracking Tools”, from 2013, is what is available, while “Motive” is the currently supported application for tracking. This application is upwards of $1000 just for the license, while the alternative Lighthouse Positioning System by Bitcraze is around $500 for a full setup. Because Bitcraze makes the Crazyflies as well, the lighthouse system is extremely well integrated and documented. This newer system is also portable, meaning it can be moved and demonstrated almost anywhere, unlike our OptiTrack system.

9.1.3. Other Considerations

Test Stand Control Board Bill of Materials

Item	Quantity per board	Total Quantity
Arduino Nano	1	12
100 ohm resistor	1	12
6mm push-button	1	12
5 pin 2.54mm 90 deg header pin	1	12
15 socket 2.54mm header pin socket	2	24
Test Stand Control Board PCB	1	12
Miniature absolute magnetic shaft encoder	1	12

9.1.4. Code

The code for our project can be found on our Gitlab repository from the link below:
https://git.ece.iastate.edu/danc/MicroCART/-/tags A new tag will be created for the final state of our repository called `2022-team-final-state`, if this tag does not exist yet, the current state of the master branch is our project code.
The code consists of the three main sections, the Crazyflie firmware, the ground station software, and the test stand software. They are located in /crazyflie_software/crazyflie-firmware-2021.06, /groundStation/, and /test_stand_firmware respectively.