MicroCART

Design Document

Team Number: 45
Client: Dr. Phillip Jones
Advisor: Dr. Phillip Jones

Team Members:
Tyler Johnson - Project Manager
Austin Beinder - Simulation/Controls Lead
Emily Anderson - Telemetry/Backend Lead
Grant Giansanti - Client Interactions/Testing
Connor Ryan - Physical Systems Lead
Cole Hunt - Git Master/Device OS
Gautham Ajith - Lead Youtuber/GUI

Team Email: sdmay23-45@iastate.edu
Team website: https://sdmay23-45.sd.ece.iastate.edu/#

Revised: 04/29/2023 - Version Final

Executive Summary

Summary of Requirements

Our project requires improvement and expansion of the quadcopter resources developed by prior MicroCART teams. The primary avenues of improvement are the 'Ground Station" PC software for interaction with quadcopters, lab materials for CPRE 488's UAV Control lab, and developing a small quadcopter with multicore computing capabilities.

Applicable Courses from Iowa State University Curriculum

- CPRE 288 - Embedded Systems
- CPRE 308 - Operating Systems: Principles and Practice
- CPRE 458 - Real Time Systems
- CPRE 488 - Embedded Systems Design
- CPRE 489 - Computer Networking and Data Communications
- COMS 309 - Software Development Practices
- EE 333 - Electronic Systems Design
- EE 475 - Automatic Control Systems

New Skills/Knowledge acquired that was not taught in courses

- CAD 3D Modeling
- PCB design
- Raspberry Pi development
- Ubuntu imaging
- QT GUI programming
- GitLab Documentation

Development Standards & Practices Used

- IEEE 1936.1-2021: IEEE Standard for Drone Applications Framework
- IEEE 1625-2008: IEEE Standard for Rechargeable Batteries for Multi-Cell Mobile Computing Devices
- Bluetooth Low Energy (BLE)
- Utilize a wifi standard within the 802.11 specification for wifi communication
- Waterfall Methodology

Table of Contents

List of Acronyms/definitions

MicroCART- Microprocessor Controlled Aerial Robotics Team

Crazyflie- Refers to the Bitcraze Crazyflie which is an open source drone hardware and software which has been developed for prior MicroCart Projects

CPRE 488 - Computer Engineering 488 refers to the class Embedded Systems Design

EE 476 - Electrical Engineering 476 refers to the class Control System

PID - Proportional–Integral–Derivative which relates to the Proportional–Integral–Derivative controller

PCB - Printed Circuit Board

FPGA - Field-Programmable Gate Array

IMU - Inertial Measurement Unit

ESC - Electronic Speed Controller

PWM - Pulse Width Modulation

GPIO - General-Purpose Input/Output

GUI - Graphical User Interface

CLI - Command Line Interface

MP-4 - Machine Project 4 refers to the CPRE 488 Embedded systems class project that involves using the MicroCART project

SDMAY - Senior Design May

BLE - Bluetooth Low Energy

I2C - Inter-Integrated Circuit

XIAO SAMD21 - Low powered Arduino Controller

Seeeduino - Arduino Microcontroller that is used as the motor controller

pycrocart - new GUI developed for use with crazyflie

flypi- the name given to SDMAY 45 new drone

1 Team

1.1 Team Members

- Austin Beinder
- Cole Hunt
- Connor Ryan
- Emily Anderson
- Gautham Ajith
- Grant Giansanti
- Tyler Johnson

1.2 Required Skill Sets for Your Project

- Software
  - C/C++
  - Qt
  - Arduino
  - MATLAB
- Controls
- Simulation
- Video Editing
- CAD/3D Printing
- PCB Design
- Networking/Communication

1.3 Skill Sets covered by the Team

- Software
  - C/C++ (Cole Hunt, Grant Giansanti, Emily Anderson, Gautham Ajith)
  - Qt (Emily Anderson, Gautham Ajith)
  - Arduino (Grant Giansanti, Connor Ryan)
  - MATLAB (Austin Beinder, Gautham Ajith)
  - Python (Austin Beinder)
- Controls (Austin Beinder, Tyler Johnson)
- Simulation (Austin Beinder, Gautham Ajith)
- Video Editing (Gautham Ajith)
- CAD/3D Printing (Austin Beinder, Tyler Johnson)
- PCB Design (Connor Ryan, Tyler Johnson)
- Networking/Communication (Emily Anderson, Cole Hunt)

1.4 Project Management Style Adopted by the team

We have an assigned project manager who is to uphold with the help of the other team members a waterfall project management style. The design of the product requires each step to be done prior to beginning the next. Product development follows a linear sequence due to researching parts, ordering, assembling, and testing being a time/resource consuming process which limits the number of iterations allowed.

With all of the files being handled by git, we use it to track the progress of our project. Last team's year used git issues to track progress and using the same strategy would improve cohesiveness and readability for future teams. Progress is being documented through Git Issues and well written commit messages following a commit template with relation to the corresponding issues. Gantt chart tracks our top-level deadline for our project.

One major change we made for the Spring semester is working in 2 week sprints to increase frequent communication and keep each other more accountable. It was clear to us last semester that we did not make as much progress as we wanted to so therefore we mitigated this with our sprints.

1.5 Project Management Roles

- Project Manager: Tyler Johnson
- Git Master- Cole Hunt
- Physical Systems Lead: Connor Ryan
- GUI Lead: Gautham Ajith
- Backend/Telemetry Lead- Emily Anderson
- Device OS- Cole Hunt
- YouTube Lead: Gautham Ajith
- User Interaction/Testing: Grant Giansanti
- Simulation/Controls Lead: Austin Beinder

2 Introduction

2.1 Problem Statement

For many years, Professor Jones has been advising MicroCART teams to construct a variety of aerial robots for use in research on embedded control systems. Recent years have gravitated towards quadcopters for their controllability. Past quadcopters were massive with early iterations even being gas powered. As technology has advanced, smaller electrically powered quadcopters have become practical. Past teams have developed large FPGA controlled quadcopters and small drones controlled by microcontrollers. For future experimentation there is a desire to create a small to medium sized quadcopter backed by a multicore microprocessor. The new platform would allow the exploration of advanced control algorithms and the incorporation of operating systems. The quadcopter should maintain a small size for convenient indoor flight and experiments. The Ground Station PC software used to interface with MicroCART quadcopters should also be adjusted for improved performance and compatibility with the new quadcopter.

2.2 Intended Users and Uses

MicroCART strives to develop new quadcopter resources to be used throughout the Iowa State Community. The three primary groups the current team targets with new innovations include:

- User Group #1: CPRE 488 Students
    - SDMAY22-43 developed a UAV Control lab for the class using MicroCART resources
    - Improvements to Ground Station GUI increase the usability and enhance students' experiences in the lab
- User Group #2: Graduate Students
    - Graduate students researching quadcopter control need a variety of platforms to conduct experiments
    - Documentation is needed to quickly allow researchers to integrate and adapt the resources provided to them
- User Group #3: EE 476 Students
    - Students enrolled in control system courses will likely have a curiosity towards quadcopter flight controllers
    - The course provides a future opportunity to use MicroCART resources in lab experiments
    - Lab experiments already using similar quadcopters but are limited by using manufacturer software

2.3 Requirements & Constraints

- Functional Requirements:
  - The drone communication must have a maximum latency of 10 ms
  - Should be able to maintain hovering in place for 5 seconds
  - The quadcopter should perform as well or better than the Crazyflie currently used
- Resource Requirements:
  - The quadcopter should communicate over Bluetooth Low Energy or the wifi network cards on lab computers
  - The quadcopter must use brushless motors
  - The processor controlling the quadcopter should have multiple cores available
  - The flight control software should be useable on lab computers without the need for administrative privileges
- Physical Requirements:
  - The dimensions of the quadcopter will be less than 125mm x 125mm.
  - The quadcopter will have a balanced layout to allow for stable flight.
  - The quadcopter will weigh less than 300g.
- User Experiential Requirements:
  - The documentation for the hardware and software should be easily comprehensible by senior level engineering students interested in controls or embedded systems.
  - The GUI needs to be easy to navigate and simplicity of the interface
  - The GUI should provide reliable performance when communicating with the mini quadcopter, crashing less than 2.5 times per 3 hour work session.
- Environmental Requirements:
  - The quadcopter should be safe for indoor use.
  - A test stand capable of restraining the quadcopters motion to only roll, pitch, or yaw.
  - Student labs will often have multiple students present and the risk of physical harm to students should be low. No more than one incident should occur per semester.

2.4 Engineering Standards

- IEEE 1936.1-2021: IEEE Standard for Drone Applications Framework
  - Within this framework it describes the support of drones including the flight platform, the control systems,qualification of operators

and insurance. We will need this to make sure that we are using proper safety when operating our drone

- IEEE 1625-2008: IEEE Standard for Rechargeable Batteries for Multi-Cell Mobile Computing Devices
    - This describes the standards for using lithium ion batteries within a computing device. Since we are power both the PIE and the drone from lithion battery we need to make sure we are following proper procedures to make sure the drone is safe and does not exceed extreme temperatures
- Utilize a wifi standard within the 802.11 specification for wifi communication
    - This standard describes the protocols of wifi communication. We will be using the drone over wifi as well, therefore we will need to to follow these standards.

3 Project Plan

3.1 Project Management/Tracking Procedures

The project workflow follows the waterfall management style. The design of the product requires each step to be done prior to beginning the next. Product development follows a linear sequence as researching parts, ordering, assembling, and testing is a time/resource consuming process which limits the number of iterations allowed.

Progress throughout the project will be tracked using Git issues. This is a favorable option since past files are hosted on git. Former teams tracked issues in a similar fashion so using the same approach will result in material being readable and cohesive.

3.2 Task Decomposition

1. New Mini Quadcopter
    a. Hardware
        i. Research quadcopter hardware

ii. Select and order premade parts (frame,motors,battery)
iii. Design IMU/ESC PCB
iv. Order and assemble IMU/ESC PCB
v. Test basic operation of IMU/ESC Board
vi. Test mini quadcopter with Raspberry Pi flight controller firmware (correlates with 1.b.iii)
vii. Evaluate possible improvements to 1st hardware revision

b. Software
i. Convert Crazyflie flight controller firmware to Raspberry Pi
ii. Test I/O pins for proper operation
iii. Test firmware on mini quadcopter (correlates with 1.a.vi)

2. Ground Station
   a. Understand current CLI/GUI implementation
   b. Explore issues found while running through MP-4 (correlates with 3.b)
   c. Improve User Experience
   d. Minimize crashes and data loss
3. Improve Lab Material
   a. Complete MP-4 to gain familiarity with the lab
   b. Address issues encountered during MP-4
   c. Improve lab document for readability
   d. Add information to CPRE488 wiki to help students debug issues
4. Improve Simulation
   a. Understand current Simulink based simulation
   b. Move from being mostly Simulink based to being mostly Matlab based

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- New Quadcopter
  - Quadcopter will communicate with the ground station over WiFi with a latency less than 10ms.
  - After tuning PID values quadcopter will be able to maintain a controlled hover for more than five seconds.
  - The flight controllers control loop will execute at the same speed or faster than the current crazyflie flight controller.
- Ground Station GUI
  - A ‘Set Parameters From Json’ button will be added to streamline the process of setting PID values.

- The test stand software will be adjusted to provide data with less than a two second delay, compared to a prior delay of ten seconds.
- Lab groups will experience no more than 5 errors using the ground station when completing MP-4.

-

4 Design

4.1 Design Context

4.1.1 Broader Context

Research groups and quadcopter labs need an improved quadcopter from the current crazyflie model with additional computing power and flexibility. The current model is a limited platform with minimal wiggle room available for deploying new hardware or software. The interaction with the drone though the Ground Station GUI can be buggy, slow, and troublesome. Performance issues heavily impact classes including CPRE488, EE476, and research groups using the crazyflie. These groups could save time debugging the drone and its software if bugs were fixed and systems were better documented. Reducing the time for debugging can allow for the users to be able to quickly conduct the research or learning background information needed for using the quadcopter.

Our project must adhere to public safety standards as drones are commonly operated indoors. Failsafes will be put in place to mitigate potential risks to students and lab equipment. Safety systems include test stands for safely tuning flight control, kill switch for powering off uncontrollable quadcopters, and motor power being automatically cut when connection is lost.

The quadcopter labs have an economic impact because the current crazyflie drones cost around $225 each. Building our own quadcopter presents a chance to reduce cost and increase the durability of quadcopters. Economically impact can be further mitigated by designing the quadcopter in a way to be easily repairable with spare parts.

4.1.2 Prior Work/Solutions

The MicroCART team has been around for over 20 years working to continuously improve aerial robotic resources at Iowa State. There have been many different implementations over the years that we can take inspiration from. Each of these teams has a git repo and last year's team created a youtube channel with tutorials and demonstrations to reference. One past project, made by SDMAY 15-28, was a large quadcopter too big to be used in the classroom, shown in Figure 1.

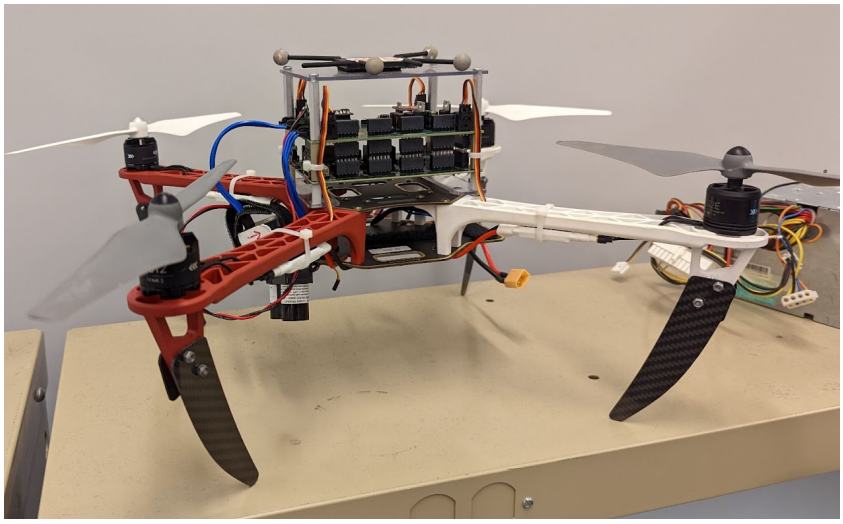


Figure 1.1: Team 15-28 microcart

The quadcopter had a range of capabilities with an onboard FPGA, but the size limited the ability to conduct experiments in the laboratory. The succeeding teams sought to develop a smaller quadcopter solution because reducing the size provided a safer environment and also reduced cost.

The previous team used a crazyflie as the drone of choice. The open-source design, shown in Figure 1.2, provided a slew of expansion decks and compatible software. The small size allowed the creation of lab experiments for undergraduate courses, but the quadcopter had greatly reduced computing power compared to prior generations.

Figure 1.2: Crazyflie

The crazyflie drone led to successful implementation of quadcopter labs, but student feedback collected by last year's team pointed to areas of improvement.

Our new quadcopter design seeks to find a balance between the convenient size of the crazyflie and the computing power of larger, research drones.

Lastly, we can also use other drone enthusiast work to help us on this project. There are many webinars that mathworks puts on that shows how to connect a drone to the simulations. These resources will help us to expedite our prototyping time.

4.1.3 Technical Complexity

Designing a new quadcopter and revising Ground Station software for improved performance presents a technically complex task.

The quadcopter design requires the integration of an ESC, IMU, and expansion decks with a Raspberry Pi Zero 2W. A power distribution system must be designed to provide sufficient power to the various modules. Software must be developed to interface the microcontroller with the motors and sensors. All of the modules and connections must be fixed on a compact PCB housed in the frame of the quadcopter. In addition, the Raspberry Pi needs to have an adapted image to provide wifi access through an Ubuntu operating system and reserve a core to run FreeRTOS on baremetal for the flight controller.

Making improvements to the Ground Station software requires a thorough knowledge of the existing architecture. The current system consists of a frontend, GUI, CLI, backend, and adapters for the specific quadcopter. An understanding of the operation of each module and how they interconnect is needed to effectively alter the software.

4.2 Design Exploration

4.2.1 Design Decisions

List key design decisions (at least three) that you have made or will need to make in relation to your proposed solution. These can include, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, features, etc. Describe why these decisions are important to project success.

Our project is to design a quadcopter. Quadcopters have a limited number of physical components and these include a frame, flight computer, battery, motors, motor controllers, and propellers. Our first design decision was to select the components we would use in supporting the quadcopter. Our client specified that

they would like us to either use a Raspberry Pi Zero or Zynqberry Zero as the main flight computer. We then also require an IMU to provide sensor data needed to support the flight computer in determining its position, and also a voltage regulator for stepping down the battery voltage. Knowing that we require these sets of components, we have spent some time in specifying at least three different options for each and then selecting what we believe to be the best combination for the hardware design of our drone. The selection of these components is important because they impact the total weight and size of the quadcopter. The correct combination of parts must be selected to provide a drone large enough to house the required hardware and enough motor power for efficient flight. A detailed description of the selection process for quadcopter parts is provided in Appendix 3.

The next design decision is based upon the microprocessor being used. The Raspberry Pi Zero 2W and Zynqberry Zero both have the same footprint, but they lack the basic sensor and motor capabilities needed for flight control. A breakout board to provide the microprocessor with the necessary abilities must be designed. Deciding between the microprocessors is important because they vary in computing capability and implementation complexity. Ultimately the Raspberry Pi Zero 2W was chosen due to a reduced implementation complexity and greater availability in the marketplace.

Our third design decision was to select problems with the Ground Station GUI to address and fix. Focus on improving the key functionality of the user interface is important to efficiently allocate resources to improve user experience. The decision to focus on fixing these issues will increase the amount of success that students enjoy during the lab.

4.2.2 Ideation

Over the course of the semester many different designs within our many parts of this project were considered. Back in 491 we had to decide the hardware components for our drone and that thought process is provided in appendix 3.

However one major design that required much ideation was the test stand to hold down the drone when testing and to be able to isolate one axis for PID tuning. This took multiple iterations and ultimately some redesigns.

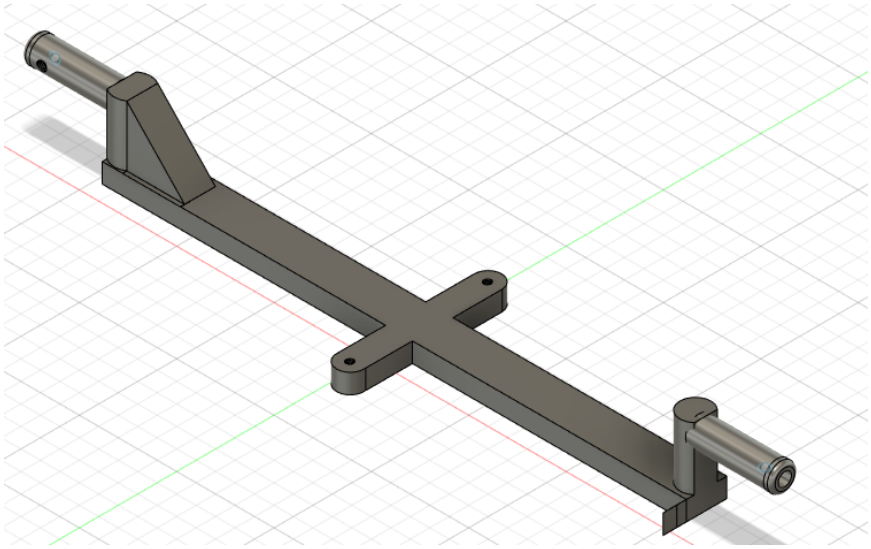


Figure 4.1: Roll Test Stand Design 1

In the above figure this was my idea for my roll test stand and this would attach to two holes directly onto our frame of the drone. However while this worked the placement of the battery in the design which was currently on top of the drone was making the drone unstable. Therefore we had to redesign a stand to hold the battery then be able to connect onto the test stand.

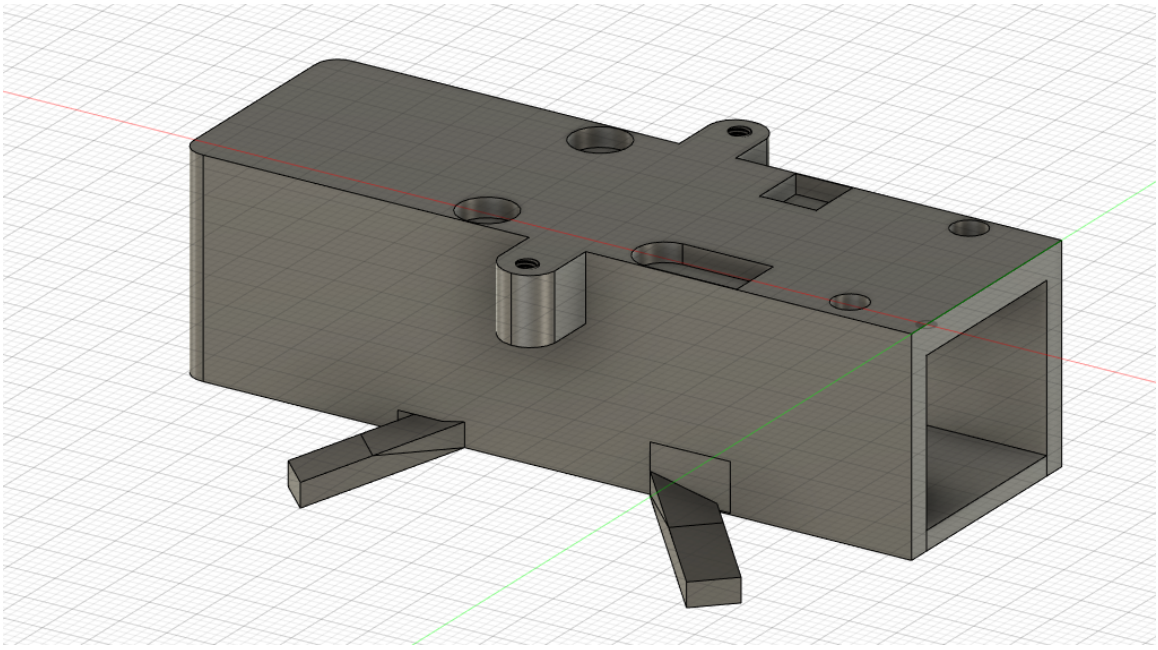


Figure 4.2: Battery pack design 1

This was our first design of the battery pack: the two holes should connect to the same two holds the test stand was supposed to connect with and the battery would just slide into the open end. Additionally there were a couple of legs that would be able to satellite the drone when it was sitting. The problems with this design were that the threads in the holes stripped immediately and thus had to be redesigned to allow for a not and bolt solution for future designs. also because the battery produced so much power there was a need for the battery to not be completely enclosed to allow for it to dissipate heat. Therefore a webbed design was used in future iterations. Finally the legs were way too thin and 3d printers had a hard time printing them therefore we switched to attaching legs to the motors.

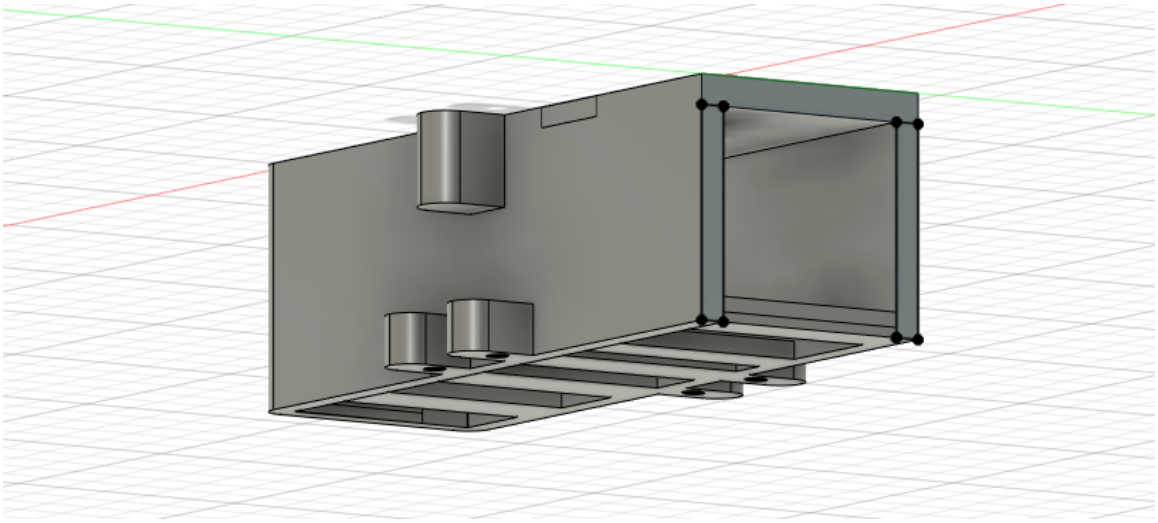


Figure 4.3: Battery pack final design

The above design is what we are currently using. It fixes all the above issues and also allows us to be able to attach a test stand to the bottom for testing.

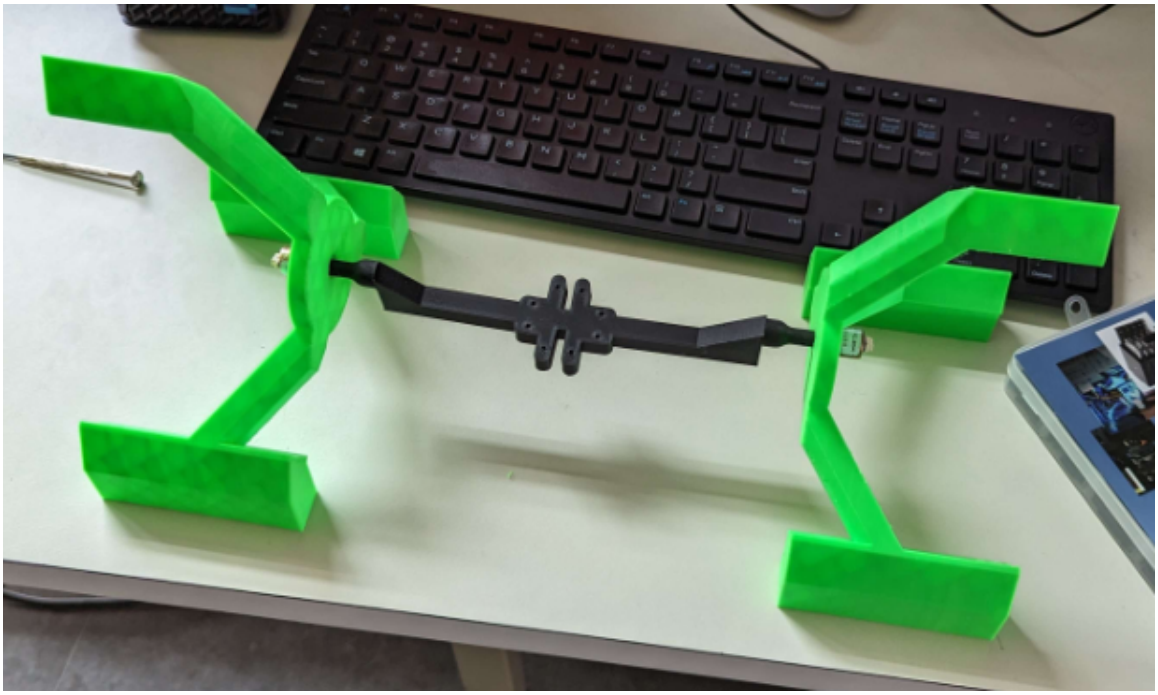


Figure 4.4: roll and pitch test stand design

Finally this was the final result of the test stand that connects to the battery back and the drone. This seemed to work the best but due to the massive power of the drone a need for clamps to clamp the plastic down was still needed.

If this design was used in the future a need for a metal test stand would be needed for structural integrity and weight.

4.3 Proposed Design

4.3.1 Overview

An improved quadcopter is needed from the previous models to balance size and computing power. The current model lacks multicore processing ability and has limited customizability for deploying new hardware or software.

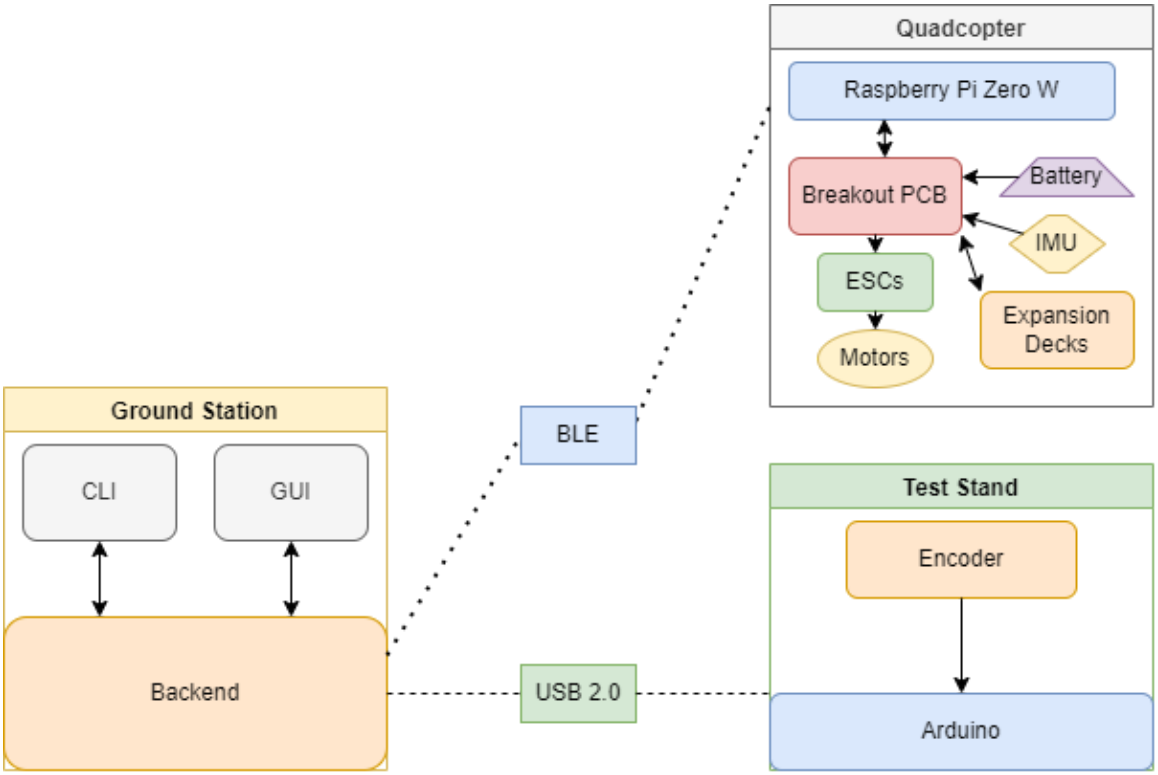


Figure 4.5: Proposed Design

The diagram, in Figure 4.5 above, shows a high-level overview of the system we are implementing. The PC runs the ground station software using either a CLI or GUI to send commands. The commands are sent over bluetooth to the Raspberry Pi Zero 2W mounted on the quadcopter. The microprocessor is running the flight controller using inputs and outputs from a circuit board our team designed. When calibrating the quadcopter it is mounted to a test stand which uses an encoder and Arduino nano to log position and rotation data back to the PC. This data is useful for tuning different control parameters in the quadcopters firmware.

A new design can be used to solve these problems, as well as enhance the drone's capabilities. Our new design will make the following improvements while keeping the capabilities of the previous model. We will make a small form factor drone with a carbon fiber frame making it robust to falling or crashing into things. This drone will also be capable of more complex computation by using a full processor like that of a Raspberry Pi Zero or Zynqberry Zero. To give the microprocessor the ability to fly the microcontroller will be attached to a custom PCB board to breakout its connections and interface it with flight systems. The first flight system is an IMU sensor which provides the flight controller with information

about the quadcopters orientation and rate of rotation. The second part of the flight system is the ESC, or electronic speed controller, which is how the quadcopter sets the speed of the motors. In order to power the flight systems a 5V regulator will be placed on the main board. The 5V regulator will be attached to a 3S battery which powers the ESC directly and all other systems through the 5V regulator. These quadcore solutions will enable researchers and students to run their software on a more sophisticated operating system. The motors will be brushless dc motors rather than brushed dc motors to improve the dynamic performance of the drone as it flies.

Beyond just a new quadcopter design, we will also be enhancing the existing software and simulation software. For the main PC side software, we will be finding various GUI, reliability and speed improvements that we can make as time goes on. Changes to the GUI include user experience and user interface enhancements made to improve the overall lab experience for students. These changes range from adding more convenient buttons to perform certain tasks to adding functionality for increased speed. For the drone side, we will be trying to run the drone using three cores to host Ubuntu and a single core running the flight controller to interface with the flight systems.

4.3.2 Detailed Design and Visual(s)

As mentioned above this project is mainly about improving the quadcopter resources that are currently available to students and researchers at Iowa State.

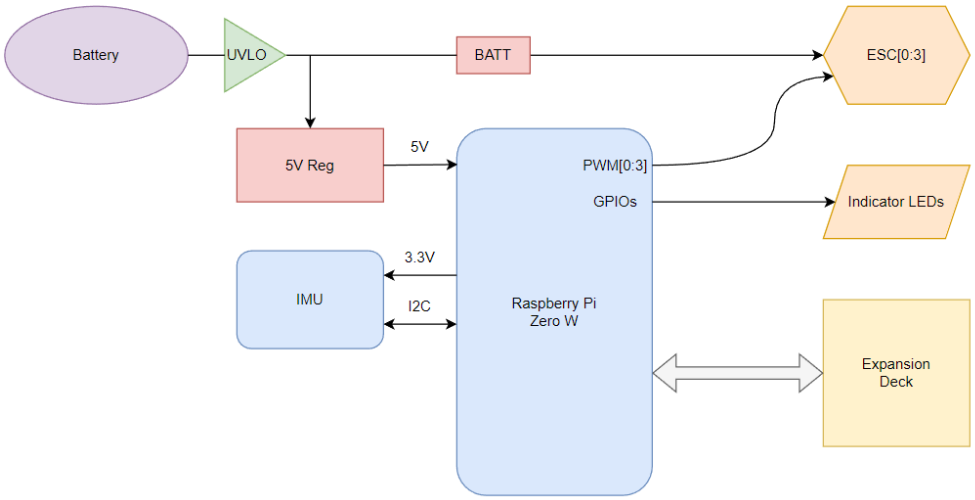


Figure 4.6: High Level PCB design

As you can see from above in Figure XX, our proposed quadcopter design uses a Raspberry Pi Zero 2W as the flight controller. The Raspberry Pi does not have the capabilities to operate a quadcopter on its own so we will design a printed circuit board to allow it to interface with an Inertial Measurement Unit (IMU), electronic speed controllers (ESC), and various expansion decks.

The quadcopter is composed of several manufactured quadcopter parts. The battery used is a 3S LiPo battery, for racing drones, capable of high current output. The propulsion is 3" propellers mounted to brushless DC motors for reduced noise and increased controllability. The ESC used to drive the motors is the Mamba Mini 4-in-1 F30 ESC and the phase is shifted on the front right and back left motors to swap their direction of rotation. The LiPo battery is connected directly to the ESC through a XT30 connector and a 47uF high-speed capacitor is placed across the terminals to reduce variable current draw.

The first part of the custom system to consider is power. The ESC connection to the main board provides a PWM channel for each of the motors and provides the power input to the PCB. The power connection is toggled by a switch extruding from the front of the quadcopter for easy access to power down the drone. The switch must be easy to reach in case the quadcopter becomes uncontrollable. The battery power is then filtered with a power line ferrite bead before reaching the 5V regulator. A switching power supply was chosen due to the high power efficiency requirements of a battery powered quadcopter. The power supply is rated for 3A to provide adequate power to the Raspberry Pi, PWM controller, and any expansion decks. The Raspberry Pi includes an onboard 3.3V supply which is used to power the IMU. The PWM controller and the IMU are connected back to the Raspberry Pi over I2C. The IMU has its own structure of reading and writing various registers to configure the accelerometer and gyroscope which is described in the parts datasheet. The PWM controller is a Seeduino supported by the Arduino platform. The Seeduino firmware was designed to output hardware generated PWM signals to the ESC since the Raspberry Pi is limited to two PWM channels. The Seeduino is connected to the Raspberry Pi over I2C and takes four 16 bit unsigned integers as input and assigns a value to each motor. Four LEDs are included on the board, one in each corner, with red LEDs in the front and blue in the back. The LEDs can be flashed by the microprocessor to display various error messages.

The flight controller software operates by periodically reading the sensor data from the IMU over I2C. The sponsor data is used for state estimation of the quadcopters attitude. The data is used in a PID controller where the recorded data is compared to a setpoint from the Ground Station PC software and the error is calculated. The error value is scaled based on PID values discovered through tuning on our 3D printed test stands. The test stands limit the quadcopter's range

of motion to only yaw, pitch, or roll depending on the configuration. A properly tuned PID controller will calculate the motor outputs to achieve the designated setpoint and the values are sent to the PWM controller over I2C. The PWM controller adjusts the duty cycle of the PWM wave for a motor channel based on the motor's specified output and the ESC will adjust the RPM accordingly.

The setpoints used to control the quadcopter are sent from the Ground Station software on a local PC. The project is configured to run off the existing crazyfly firmware so the communication pipeline was intended to be replicated. The crazyflie python library allows for communication between a Ground Station PC and another device through a TCP socket. The setpoints and other information to configure and operate the drone are sent through cflib (crazyflie python library). A python script running on the Ubuntu cores of the Raspberry Pi Zero 2 W capture this data and pass it to the FreeRTOS baremetal core where it is piped into the program as if it was directly communicating with the crazyflie as normal. The data being sent back to the Ground Station would follow a similar pipeline where it is passed to the Ubuntu cores through shared memory and is transmitted back to the Ground Station as if it were receiving information directly from a crazyflie.

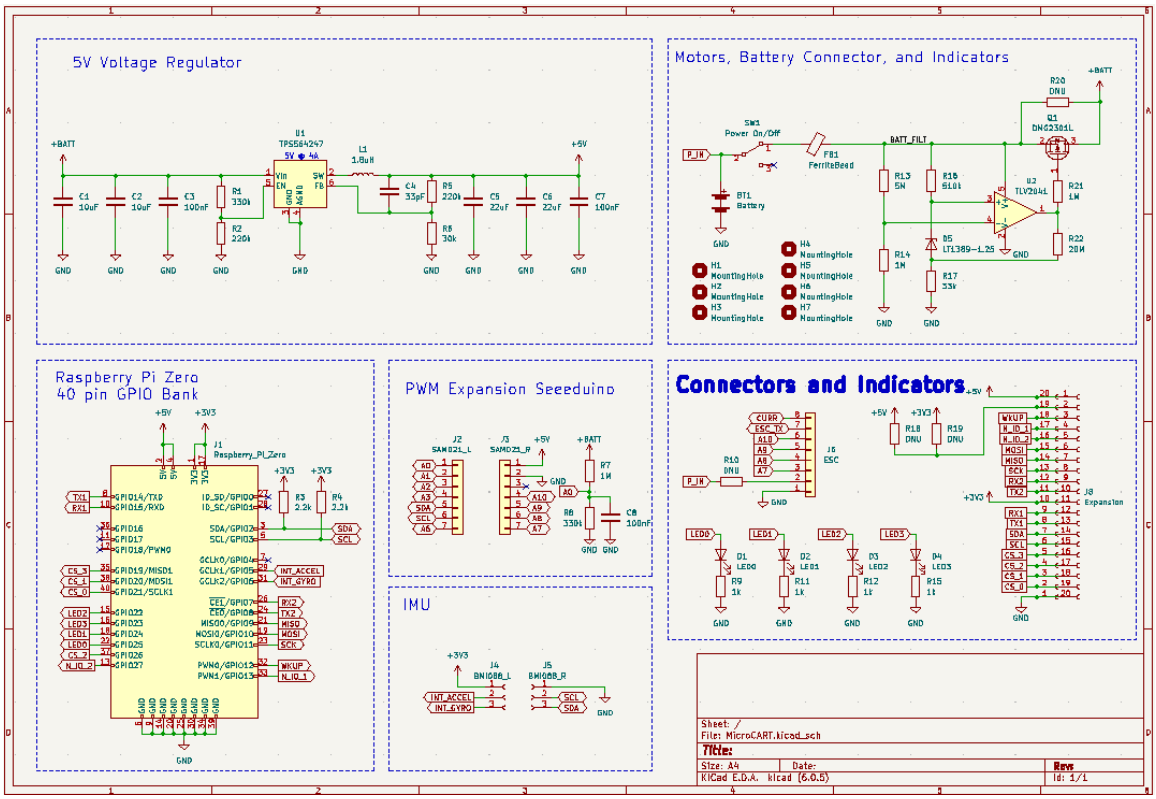


Figure 4.7: Preliminary Schematic

Shown above is the final schematic of the breakout PCB including the power management, Pi connections, external connections, and an inertial measurement unit.

4.3.3 Functionality

As mentioned above, our design is specifically targeting researchers, CPRE 488 Students, and EE476 students. It is intended as an opportunity for students to work on a real world embedded systems application while combining their hardware and software skills. Students will have the opportunity to put programs on the drone to directly control the functionality of the drone. They can control the drone with a radio and test it on the test stand. Additionally they can simulate it on MATLAB and test out their program before applying it to the drone. This can all be done on an easily accessible GUI that we will develop.

A student in CPRE 488 might first start with our simulation to simulate different PID values of the quad copter and to see what happens when you adjust these values. After mastering this part they will then move on to the drone, booting their software onto the drone using our intuitive GUI. This process will hopefully be very easy to follow for the students and make a lot of sense.Then the students can test their PID values or other research on the test stand to make sure that the drone is working properly. Finally they are able to control the quad copter using a joystick through the air. There are a couple of other functionalities however we are trying to make this as functional and as seamless as possible for students.

4.3.4 Areas of Concern and Development

We believe that based on the background our client has given us, and our continued meetings with them that our proposed design will satisfy the vast majority of user needs. Our understanding of user needs includes a better functioning 488 lab, a drone that offers more fine grained control, as well as greater computational power, and a simulation environment that would enable students and researchers to test their control algorithms in a low stakes environment.

To meet project expectations, specific areas of development from the GUI that students interact with, to the hardware that will enable greater computation and control, to the Simulation environment have been delegated among team members.

Our biggest concern when it comes to this is to stay on our timelines to ensure these tasks actually get done. We tend to be a pretty busy group, and so making sure that everyone actually finds time to work on the project has been a bit of a challenge. We do also have some pretty regular questions for our clients mainly focusing on what types of components they wish for us to use. We believe that by regularly meeting, asking questions and by continuing to address issues as we see them, we can surpass any and all of these issues and build a successful project.

4.4 Technology Considerations

The design will use a Raspberry Pi Zero 2W for the main computing unit on the quadcopter. This is a lightweight computer with a Quad Core processor and easy GPIO breakout and control. The small scale of the board allows for mounting on the quadcopter frame. The Pi Zero only has 2 hardware PWM channels and 4 are needed to control the quadcopter, however the variable usage GPIO pins can be used to create software based PWM channels which would allow for full control of the motors. Currently there is a potential setback depending on the performance difference between the software and hardware PWM channels. If this is the case a PWM expander IC will be needed to control all 4 motors.

4.5 Design Analysis

The Ground Station GUI has been altered to improve user experience. Feedback detailing CPRE 488 students' experiences with the GUI were recorded by last year's MicroCART. Adjustments were made to adjust key issues like not automatically connecting to the backend and the painstaking process of resetting PID values. When the GUI starts up, the first thing users must do is navigate to the backend tab and press connect. Forgetting this step causes the GUI to crash so the software was changed to eliminate the step and reduce crashes experienced by students. The lab requires the setting of eighteen different parameters for flight control and upon a power cycle all parameters must be reset. The process was irritating to use so a JSON file was created to house the key parameters. A button allowed the automated setting of all eighteen values helping students save time and conveniently record their values. The lab was run during the spring semester and while additional feedback of ways to improve was recorded the overall response to the lab's enjoyment and ease of use was improved from the prior year.

A new medium sized quadcopter was constructed and tested for flight capability. The manufactured quadcopter parts were first assembled and connected to the

main board. The PCB was altered to allow the quadcopter to be controlled by a crazyflie equipped with the BigQuad deck. The assembly provided a test bench to check all hardware was working properly with working software. The quadcopter had the flight controller's PID values tuned using the different 3D printed test fixtures we designed. Once the flight controller was properly tuned, a controlled flight was conducted by suspending the quadcopter from the ceiling with an elastic strap tied to the ground. The hover thrust was established using the described test set up and proper movement in the XY directions was confirmed. After the operation of the flight controller was evaluated, we began an unrestricted flight test. The quadcopter hardware is capable of controlled indoor flights as shown in our demonstration.

The quadcopter firmware continues to progress. The Raspberry Pi Zero 2W was configured to communicate with a Ground Station over wifi and dummy packets were set to test the interface. This works through the existing crazyflie python library interfacing methods. A python program on the Raspberry Pi replicates the "crazyflie" and passes on information it gets from the Ground Station to the Drone Firmware and vice versa. For best performance, the flight controller needs to be hosted on a single core of the Raspberry Pi with the other three running Ubuntu to handle wifi connection. This allows for the easy network interfacing options provided within the Ubuntu OS instead of having to implement a network connection on the baremetal level. The firmware team continues to work on compiling the FreeRTOS flight controller and deploying the implementation to the Raspberry Pi. They have followed several example repositories which are included below for further reference.

https://github.com/TImada/raspi4_freertos

https://github.com/eggman/FreeRTOS-raspi3

The first example provides insight on how to implement the Ubuntu and FreeRTOS system with in depth detail however the platform the program is intended for in a Raspberry Pi 4. These run on a different processor than the Raspberry Pi Zero 2 W. The next example is for a Raspberry Pi 3, which runs on the same processor as the Zero 2 W which provides insight on how to adapt the first program to meet our processor needs. The system works by loading a compiled FreeRTOS program into an Executable and Linkable Format (.elf) file that is loaded into memory and initialized to run on boot of the board. This will run on one core as stated above for optimal computing resource allocation. The FreeRTOS program is cross compiled from a development machine and loaded into the bootable partition of the Pi Zero 2 W micro sd-card.

U-Boot is installed and built from source here https://github.com/u-boot/u-boot using the same cross compiler. U-Boot is a customizable bootloader program that

is commonly used in embedded devices to allow packaging boot instructions to configure the device's operating system kernel for desired runtime operation. This will allow for the desired boot configuration with one core running FreeRTOS baremetal and the other three running a lightweight Ubuntu environment.

Current development status of the firmware is loading the compiled FreeRTOS program into the system memory so it can be configured by Linux on boot. The team has struggled mapping the digital file to its corresponding physical memory location. The firmware has been altered to interact with the new hardware drives such as the PWM controllers and is now awaiting testing upon completion of the setting up the boot configuration. These interfaces with the microprocessor have been tested and the quadcopter should work properly after these final steps are completed. This can be seen and confirmed with the usage of the crazyflie Big Quad Deck which allows the usage of the crazyflie microcontroller instead of the Raspberry Pi. This attaches a crazyflie to the MicroCART drone and uses the crazyflie hardware to send, receive, and process data, but uses the MicroCART drone hardware to physically move in 3D space.

5 Testing

5.1 Unit Testing

- Each component used on the boards will have both their inputs and outputs tested. This ensures that on the component level everything will work. Then we can test the microcart on a system level. Because our project is very broad and contains many different subsystems we have to test many different components to make sure the whole system works.
- For the PWM expansion deck, we sent a “fake” packet to then see if it gets correctly parsed to each pin and value. We can also see if the correct PWM outputs are set. We will be using a usb with it connected to ubuntu to be sending the packets.
- IMU was tested by reading the outputs based on the positioning of the imu. This can be done through the terminal on the pi or through a picoscope.
- Telemetry can be tested by logging the values the device experiences within the device itself, and also logging the values received over telemetry and comparing them. We can also use timestamps to test its frequency. This can also be tested on the pi.
- Main PCB is tested by first checking the resistance across the power rails to ensure the power supply is not shorted. A 12V power supply is then attached to the board and the 5V supply measured using a multimeter. The voltage level should be 5±0.05V. Once the 5V supply has been verified, the

5V supply is attached to a load test drawing two watts. After the operation of the 5V supply is validated the Raspberry Pi is attached. The 3.3V supply can then be tested using Pin 1 of the IMU slot.

- Our overall drone hardware and ESC can be tested by implementing a crazyflie BigQuad board with the drone frame and motors to test that the software and external hardware works.
- GUI can be tested by doing extensive usability testing.
- The simulation will be tested by creating unit test demo files for each individual component, from low level through system integration. We will then also test many individual parameters that define the drone. We will plug these parameters into the simulation and compare the simulated values to telemetry values to determine the effectiveness of the match. We will use the same control algorithm for the drone in the simulation as we do in real life.

5.2 Interface Testing

I2C is used between the IMU and the Pi and the Pi and the PWM expansion. To test this we will run a program between the IMU and the PI to make sure that they interfacing correctly with one another. These tests will confirm functionality of these two modules within the entire system.

The ground station software and bluetooth are being used to connect the GUI to the drone itself. This will be tested by making sure data is being received from the drone and it constraints the correct information. Some of the data that we receive is the sensor data, the gyro and the accelerometer on the IMU.

Additionally there are many modules that communicate with each other. All of these will need to be thoroughly tested.

- Commands will be sent to the Raspberry PI Zero II W, over BLE, to blink test LEDs in order. The test will be repeated over WiFi to confirm both BLE and WiFi communication.
- The drone will have its gyroscope data logged over bluetooth. The drone will be placed on the test stand and rotated to confirm valid values are being read. The test stand has different configurations allowing the reading of the x,y,z axis of rotation.

Figure 5.1: Test stand for roll

- IMU data to the breakout board then the PWM breakout board and then the PI
- The interface between the Raspberry PI Zero II W and the ESC is the PWM board. The XIAO SAMD21 acts as the interface and reads I2C data to set the output of 4 PWM channels. The interface will be tested by increasing the duty cycle to rev the motors.

5.3 Integration Testing

- Integration of the PI and the PWM expansion will need to be tested. We will send a command over I2C to set PWM0 to 50% duty cycle. The duty cycle of the waveform will then be measured using a Picoscope. The duty cycle will then be set to 75% and validated with the Picoscope, then repeated at 25% duty cycle. The duty cycle will then be swept from 0-100% duty cycle. The test will be repeated for the remaining three PWM channels.
- Integration of the IMU and the PI will need to be tested. The PI will be configured with a test program streaming gyroscope and accelerometer data back to the PC. The IMU will be placed on a flat surface so it can calibrate and the gyroscope and accelerometer should read near zero values for all data besides the accelerometers z value representing the force of gravity. The IMU will be placed on its side so gravity is completely in the x direction, then placed on the other side for an acceleration completely in the y direction. The gyroscope will then be tested by rotating the IMU around the board's z-axis.

The data will be checked to validate a positive value for counterclockwise rotation and a negative value for clockwise rotation. The test will be replicated for the x- and y-axis.

- Integration of the ground station software and the drone will need to be tested. The compatibility of the ground station software and the drone will be tested by completing the MP-4 lab experiment using the new drone. The lab experiment requires setting parameters and setpoints on the drone and will thoroughly test the combination of the two systems.

5.4 System Testing

- The drone must be able to hover in place for 10 seconds with a minimal acceptable amount of moving. This will be used to test the control loop, the state estimation algorithm, and the motor control. Additionally this will test how easily the drone can be controlled through the GUI. We want to make sure there is no lag from our commands to the drone movement.
- The drone must be able to maneuver correctly through a more complicated specified series of setpoints repeatedly when configured with the correct PID values. (10 times in a row without issues). This can test how easily our simulation can be transferred to the drone as well as testing the autonomy of the drone without user input.
- The ground station must be able to successfully and consistently communicate with the drone. This shall be specified by experiencing no issues for 5 battery life cycles in a row, on multiple drones.
- The drone should be able to be programmed with PID values on the ground station with minimal issues. It should also be able to be commanded from the ground station with minimal issues during this time. This shall be tested by tuning a few PID values, getting and setting each parameter, and applying setpoints at each iteration. This test shall be run on multiple drones, on multiple laptops.
- We can test our safety features by hovering, then hitting the kill switch, removing the battery while the drone propellers are still moving and testing the kill switch on the PCB. During an autonomous flight there will also be checks to make sure if a waypoint is outside of a safe range it will not allow the drone to go that far.

5.5 Regression Testing

With all of these testings we will be using GIT for all the software to make sure that we can go back to older versions if we end up changing something. Additionally for the hardware for final l PCB we have many detachable modules and test points that will make debugging and fixing and functionality within these components.

The requirements that are critical to our project are:

- The communication latency must be under 100ms for 10,000 telemetry messages in a row.
- The drone must be able to hover in place for 10 seconds.
- The control loop must respond in an expected manner to the inputs given.
- The state estimation algorithm must determine the state of the drone within an acceptable range of error.
- We set up a series of automated unit tests that tests much of the functionality outlined above for software changes. We can run these tests as time goes on to confirm continued functionality.

Without these features our drone will not meet the specifications defined by our client as well as the drone will not be able to fly correctly.

When performing these tests we have met each of these requirements using the big quad deck. We were able to fly and with the correct tunings of the PID were able to fly deftly and with a latency of less than the required amount. This is mainly because we redid the GUI and backend to make it much faster.

5.6 Acceptance Testing

Through our project acceptance testing will be conducted as our last step of our design process. This will be conducted on a large scale to see if our drone's system will meet the requirements of our client, Dr. Jones. Dr. Jones can evaluate our drone by flying it, Using the GUI, using the simulation tools and talking with students who will be using our drones within 488. After Dr. Jones

surveys students, we get feedback on the problems they experienced in the lab and how our solution can be improved. Finally we can evaluate how well we increase the documentation for drone technology at Iowa state by the amount of people who find our tutorial videos helpful. These videos will be demonstrating an overview of our implementation and will communicate with our users on the structure of our design and its performance.

As of this moment we are still in the process of acceptance testing. We have been evaluated on our 488 progress and there were alot more changes that were needed then what we had provided. As far as the tutorial videos we will have to see how the next year's team finds these helpful.

5.8 Results

We have effectively tested the new drone and with the crazyflie Big Quad Deck we can effectively have an extremely good quad that can hover in place for a lot more than just 5 seconds. These flight tests confirm the hardware operation of the drone as well as provide insight on the usability and functionality of the Ground Station communicating with the drone. The Big Quad Deck allows the usage of the crazyflie processor and firmware to control an external piece of hardware. It is configured to send, receive and process data as if it's a crazyflie, but interface with the MicroCART drone hardware moving it in 3D space.

The communication pipeline has been tested using the crazyflie python library and the Raspberry Pi Zero 2W. We are able to successfully send data quickly through an established TCP socket over the wireless IASTATE network. This allows for fast and reliable communication between the Ground Station and the MicroCART drone while on the Iowa State campus. The Big Quad Deck was also used to confirm device connectivity as it allowed pycrocart and the MicoCART Ground Station to communicate with the hardware of the MicroCART drone which data transfer speeds within the tolerable range for gamepad controlled flight. The performance of the drone reflects that of the crazyflie, but with more room for improvement and an upgrade in system integrability.

The MicroCART Ground Station improvements have increased usability and reduced student frustration with known issues in the 488 lab. The MicroCART team worked to integrate the new changes and worked to troubleshoot issues as they appeared in the lab. Resources were also created that walked students through the technical components of the crazyflie drone which help increase understanding in turn helping with lab completion. The full results from the lab

documentation and GUI interaction usability study can be found on the MicroCART GitLab page.

6 Evolution of design since 491

There have been many major changes in the senior design project since 491. One of the earlier and biggest changes has been with the GUI for the CPRE 488 lab. New features have been added such as the JSON parameter file upload options which allows the user to set parameters all at once by editing a JSON file. With this, came a progress bar and an auto connect to backend feature. Outside of this, the lab documentation for the CPRE 488 lab was updated to clarify missing details from the previous semester. Additionally we created a new test stand to adapt the lab to allow students to create a potential controller instead of the basic roll, pitch yaw. Along with the test stand, new improved edits to the code that connected with the test stand allowed students to tune their PID values with a higher degree of fidelity then what was previously available.

A lot of focus was put into the new drone. On the hardware side, we worked on wrapping up our breakout board. The initial design focused on allowing the Raspberry Pi to interface with the sensors and motors needed for flight. Crazyflie expansion decks provide the capabilities to upgrade quadcopters by choosing from a variety of decks for position tracking and object avoidance. A range of UART, SPI, I2C, and GPIO connections are required to support all decks. The board was upgraded to four layers to accommodate the addition of numerous data lines while maintaining routability and signal integrity. The decks must be mounted on the top or bottom of the quadcopter so an expansion board was required for socketing the decks and connecting signals back to the main board.

To integrate the hardware with the software, the protocols to communicate between the ESCs and the motors were figured out through the I2C protocol. The exact duty cycles and PWMs were figured out to control the motors with precision. In order to have software to interact with it, we attached a crazyflie board to do preliminary testing with the drone. This tuning would be pivotal to being able to fly the drone. A test stand was updated to handle the new drone's power and fit the new chassis. While this was happening, changes to the communication protocol and hardware were changing. Moving from a crazyflie 2.1 board to Raspberry Pi Zero 2 W brought challenges. Eventually communication between the pi and computer were made. The quad core computer had 3 cores running Ubuntu and a single core for the FreeRTOS firmware running the crazyflie firmware.

7 Professional Responsibility

7.1 Areas of Responsibility

For the most part the IEEE code of ethics does not differ from the NSPE version for each area; however, for financial responsibility, it focuses more on motives behind work rather than the waste of money. Also, for property ownership, IEEE wants people to understand that there is new work emerging rather than focusing on giving credit to the individual's work.

7.2 Project Specific Professional Responsibilities Areas

Work Competence is high for our project because we are producing a product that is buggy at the moment, but the goal is for it to be used for other classes and researchers, so our work should be at the highest level since others will be using our product. Also, we are representing Iowa State, so we should show the quality of work produced out of Iowa State. Financial Responsibility is medium because we do not want the drones to be very costly since there will be more than one drone being produced; however, it is not a huge focus in our project.

Communication Honesty is high because we are working with an advisor who has been on this team for many years, so he has a lot of good advice to give us. Also, it is important to keep other team members informed as well as our advisor/client. Health, safety, and well-being has a high importance because we are working with drones that are considered a flying hazard. We do not want to have a drone that may fly into someone and injure them. Property Ownership is medium because we want to give credit to open source code, but we are also working on moving away from open source to using our own code. Sustainability is low because the drones we are building will be used in a classroom setting. Social Responsibility is of medium importance because we are building something that will benefit the Iowa state ICpE community, but other than that, there are no current plans for this to be used outside of Iowa State Classrooms.

Currently, our team's work competence is performing at a medium level, mainly because we were slow to start working because we had to wait for parts to come. We are slowly getting more involved with our work so the level is getting close to being considered high. We are not very conscious about our financial responsibilities so it is low. Our communication honesty is high because we meet with our team a lot and with our advisor weekly. Also, we have a discord set up to allow for virtual communication. We understand that drones are dangerous so the level is medium. Our property ownership is medium just because we have not started developing any code yet, but we have reviewed all of the previous developed code from years past. Sustainability is N/A since this has not been a

concern for our team. Lastly, our social responsibility is at a medium mainly because we know the purpose of our project, but some of us have not taken the class that these drones are used for so we do not know the current challenges in the class.

7.3 Most Applicable Professional Responsibilities Area

We think that our communication honesty has been critical to our project, and our team has been doing a relatively good job with it. This is important because this project has been ongoing for many years, so there is a lot to learn about it, and there is a lot to communicate about with our advisor who is also our client. Our communication has been really good with our advisor as well as with our team which has led to being able to solve problems quickly while also being able to move forward more quickly. Additionally with communication comes documenting the steps. As Microcarts work continues on for many more years our group needs to leave this project in a place where it can be easily adapted. This is done by creating many transition documents that we made including, MP-4 boot camp, pyrocart readMEs, GUI documentation and many youtube videos all for the future teams to watch and get up to speed on what we have accomplished.

8 Closing Material

8.1 Discussion

As of right now, the results of our project are still a bit underwhelming. We have successfully flown our new drone however we have yet to have the multi core functionality that we discuss above. We wanted to get to a point where we could fly the drone via the pi but due to unexpected issues we didn't get there. Additionally we didn't leave enough time to order another revision on the PCB to make a better expansion boards. On a positive note the drone flies very stable and can be controlled using a gamepad. We also developed new test stands to test the PID values and are able to isolate a single axis for tuning. Finally we were able to help with the 488 lab and give ideas for next year on how this can be improved.

8.2 Conclusion

While we know we would have loved to do more with this project, we have put a total of over 1200 hours into this project and are extremely happy with the level of progress we completed while we didn't accomplish all of our goals we think we

established a good starting place for the future teams to be able to pick up where we left off and continue on with our work.

The goals for our project are to improve the 488 lab experience, and to build a platform that would be attractive to researchers seeking to conduct studies on quadcopters. To improve the experience, we will improve the physical design of the drone, update the user-interfaces of the GUI, ground station and drone software. We also seeked to improve the current simulation support to better match both our drones, and researchers' needs. We accomplished this by creating more resources for the simulation and a demo of which we showed to future students.

One of our biggest challenges is time management and procrastination. When you look at the total number of work hours comparatively of this semester then last semester you will see that we put significantly more time this semester into the project. You will also see that most of our major accomplishments and achievements happened within the spring semester as we found our footing and really put in a big push to accomplish our goal. However this was done too late and just didn't allow us enough time to really get to where we wanted.

The hardware on the flypi was one of our great successes as it matched all of our goals and was able to do everything we wanted. We were able to tune the PID values with a new test stand and with the PCB create a way that research could be done on it.

Overall we think we left the project in a place that can be continued in the future and we are all very proud of what we accomplished. We hope that in some way we lived up to the expectations of the microcart team and will be able to look back at the work we have done and see the revisions and additions that were made.

8.3 Appendices

Appendix 1: Demo

This appendix is going to show how to demo our new drone and how to use it in its current state. As mentioned above the idea was to have it run with the PI however this was unable to be accomplished. Therefore our final demo is running the crazyflie software with our hardware, independent of the pie. Using the crazyflie big quad deck allows the bitcraze software to interface with our drone

thinking it's interfacing with a crazyflie when in fact it is interfacing with a much bigger drone. As this drone will hopefully be used in a future senior design teams project all of this is documented on our gitlab page for MICROcart and on our website.

1. To start the demo you must have a full 4s battery. Charging these batteries can be done by watching this (https://www.youtube.com/watch?app=desktop&v=p8OzEviW29A&ab_channel=MaxAmps.com)

   video and using the charger within the Coover 3050 lab. These batteries usually last only 10 minutes of flying and take about 30 minutes to recharge.

   Then you will need to connect the battery to the the drone

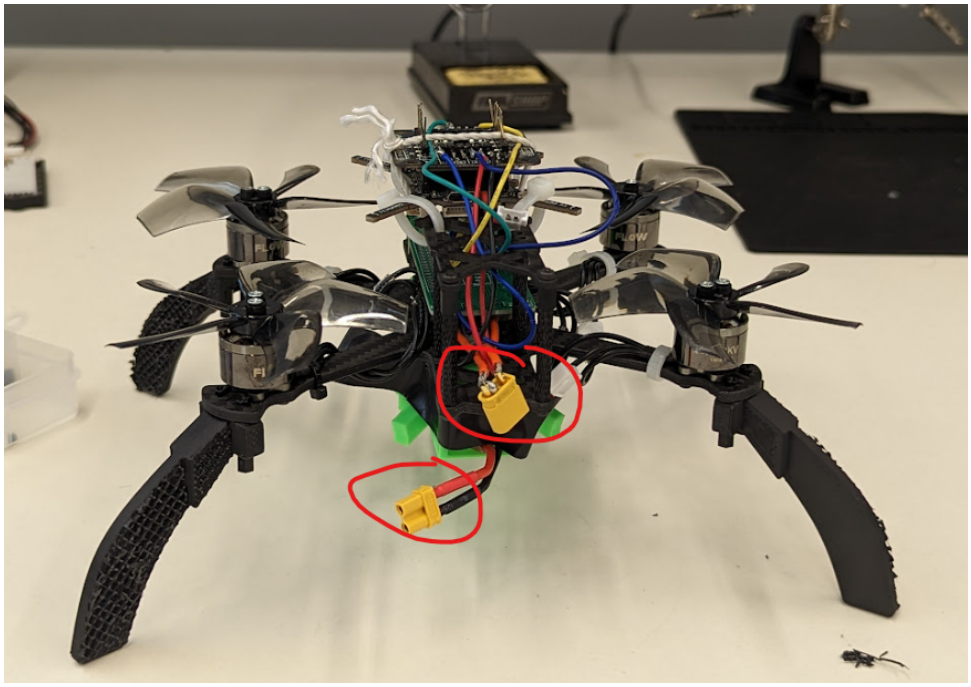


Figure A1.1: Battery connection

3. Now you need to interface with the drone this can be done with a couple of ways. The team created a new GUI outside of the MicroCART infrastructure that works really well when interfacing with the drone.

Pycrocart is a python implemented GUI that rather than interfacing with the rich MicroCART infrastructure of the backend and the rest of the quads, is connected

only to cflib which can be used to connect to crazyflie based quadcopters. While it is intended to eventually be slotted into the Microcart infrastructure, it is also able to function without it. This allows it to be used both as a debugging tool to define whether a bug exists in the backend, or the GUI. It also eliminates a many year buildup of technical debt and poor documentation in the GUI. Furthermore, in case the backend still has lots of issues like it does for our team (sdmay23), but you still need to run MP4 with crazyflies, you can use this GUI to act as a more stable version which will hopefully allow students to complain less about random crashes and more about how they can't get their PID to work the way they think it should.

Pycrocart is implemented using PyQT, and it uses cflib in order to integrate with a crazyflie. The intention is that we will be able to use the TCP connection example that bitcraze offers in order to connect to the FlyPi as well. This will be easier to implement than connecting to the MicroCART backend, which should be the ultimate end goal.

Furthermore, while the bitcraze vm is nice, this GUI runs 100 million times faster and silky smooth when not on a VM. Whether this is a linux issue or a VM issue I'm not sure yet, but I'd highly recommend not running it on the VM.

Requirements

Developed with Python 3.10 64-bit on Windows 10 in Pycharm 2022.

Library requirements can be installed in a venv via requirements.txt.

pip install -r requirements.txt

Or you can use Pycharm to install the requirements by browsing into any .py file. This is what is probably easiest.

How to run

Pycrocart was developed in Pycharm and will work best when used in pycharm. As of right now, in order to use it you must have a crazyradio plugged into your computer, and a drone turned on and on the same uri as said in the __ main__ function. The default right now is 120 which is what the fly pi runs at. Hopefully we will have a connect button soon so that you can start the GUI up and connect to a crazyradio dongle and drone during runtime.

In order to run the program from the command line enter:

python ./Pycrocart.py

This will execute the program and if all went well you should see the GUI appear.

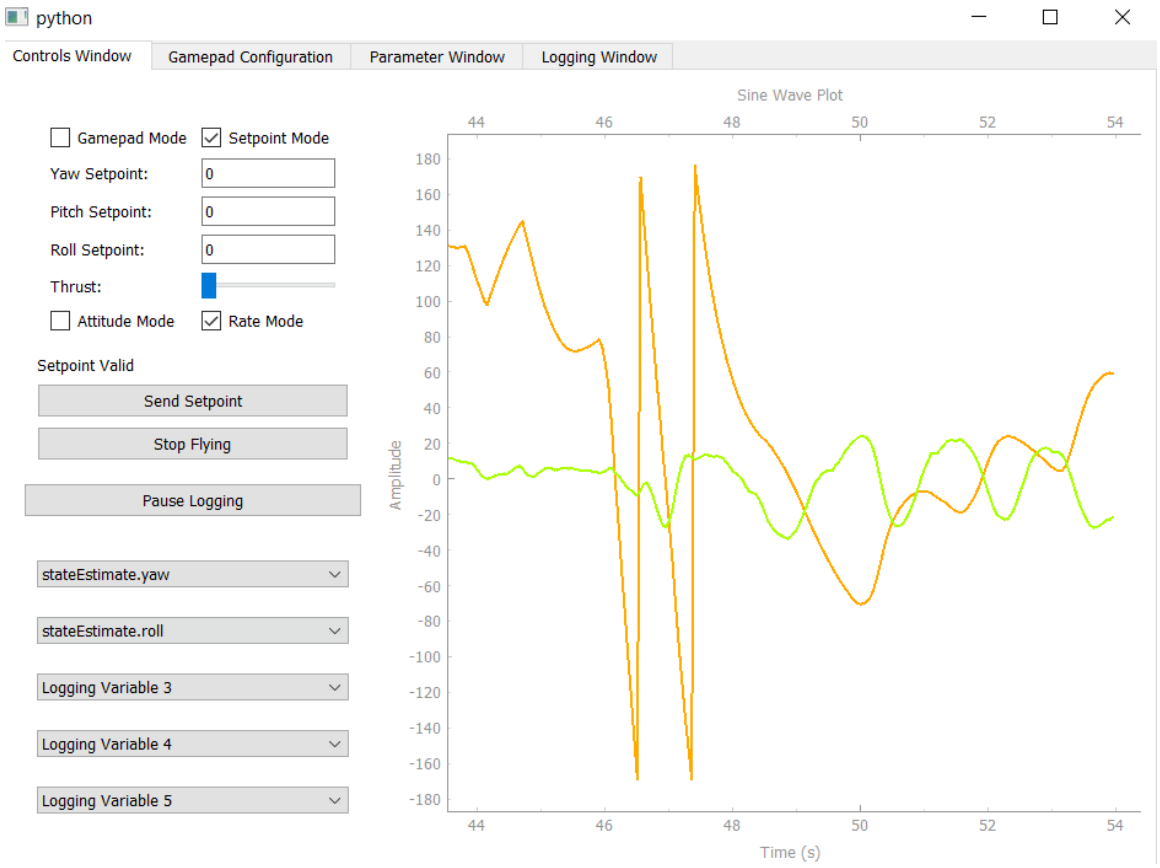


Figure A1.2: Pycrocart GUI

You can also run it from Pycharm by clicking the run button near the __ main__ function inside the Pycrocart.py script.

How to use

The pycrocart GUI has several windows: the controls window, the gamepad configuration window, the logging configuration window, and the parameter window.

Controls Tab

Seen above, the controls tab is the first page shown when entering the GUI. The controls tab features a setpoint menu which allows the user to send setpoints of thrust, yaw, pitch, and roll. It can do this in rate mode (yaw rate, pitch rate, roll rate) or attitude mode which controls the raw angles themselves. When gamepad mode is enabled the setpoints enter a "mixed" control mode. This lets the pitch and roll angles be set directly but the yaw is not. Instead, the user controls the yaw rate, and the controller will hold the yaw angle it accumulates to.

Also in the controls tab is a logging selection menu. This allows the user to select up to 5 logging signals to plot in the plotting window which covers the entire right-hand portion of the controls tab.

Gamepad Configuration Tab

Seen below, the gamepad tab is used in order to configure the gamepad to detect each axis of control correctly. The detect button is used in order to detect when a joystick or button is pressed. This is a copy and paste of the gamepad configuration tab in cfclient. If you would like more documentation of how to use it please see the bitcraze wiki. As of right now, in order to use a gamepad, it must be connected to the computer at launch of pycrocart. The logitech controller has been saved, at least on my computer as logitech. Theoretically, other controllers can be connected as well but this has not been tested by me.

To connect a controller once the program is running, click configure, select logitech from the combo box at the bottom of the page and click load to load the logitech controller configuration. You can also save a configuration. If you do not click load, a different mapping of inputs will be used. If you do not click configure, no input from the gamepad will be used.

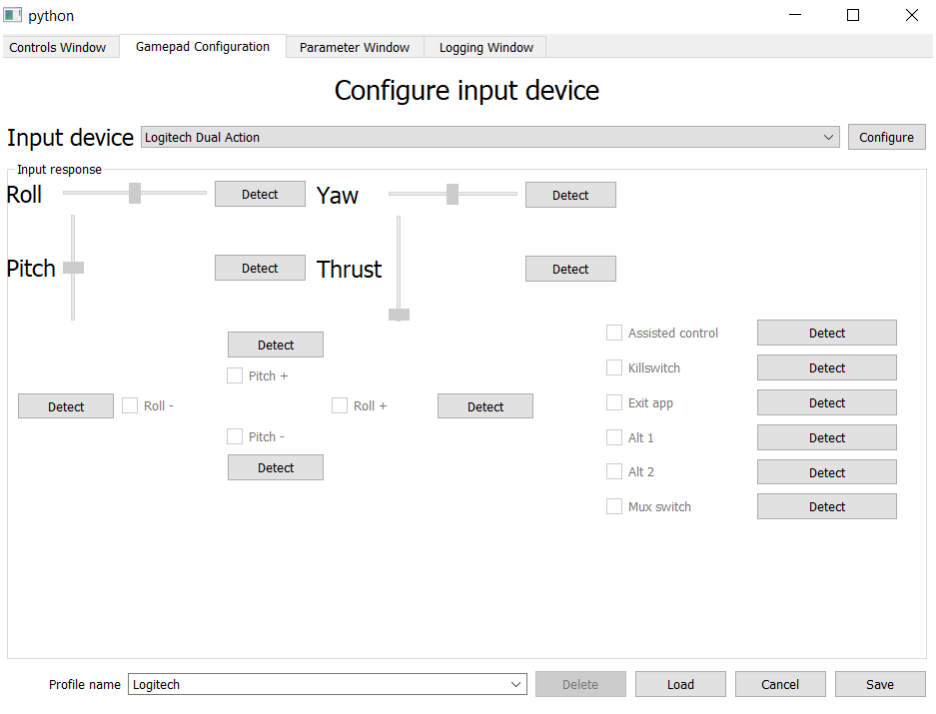


Figure A1.3: Pycrocart GUI

4. After connecting the gamepad you must upload the PID parameter values to the drone by using the .json file

Parameter Tab

Seen below, the parameter tab is able to be used in order to either set individual parameter values, or whole groups of parameters. There is also a parameter json file which can be opened by clicking edit from the UI. These parameters will be sent to the crazyflie whenever the set params from the json file is pressed. Due to the amount of data being sent, a slight delay has been added to ensure that parameter is set successfully. While there are default groups within the file for attitude rate and attitude PID's, the file is also capable of setting other parameters and groups. Anything that is able to be set from the set parameter interaction is able to be set from the file.

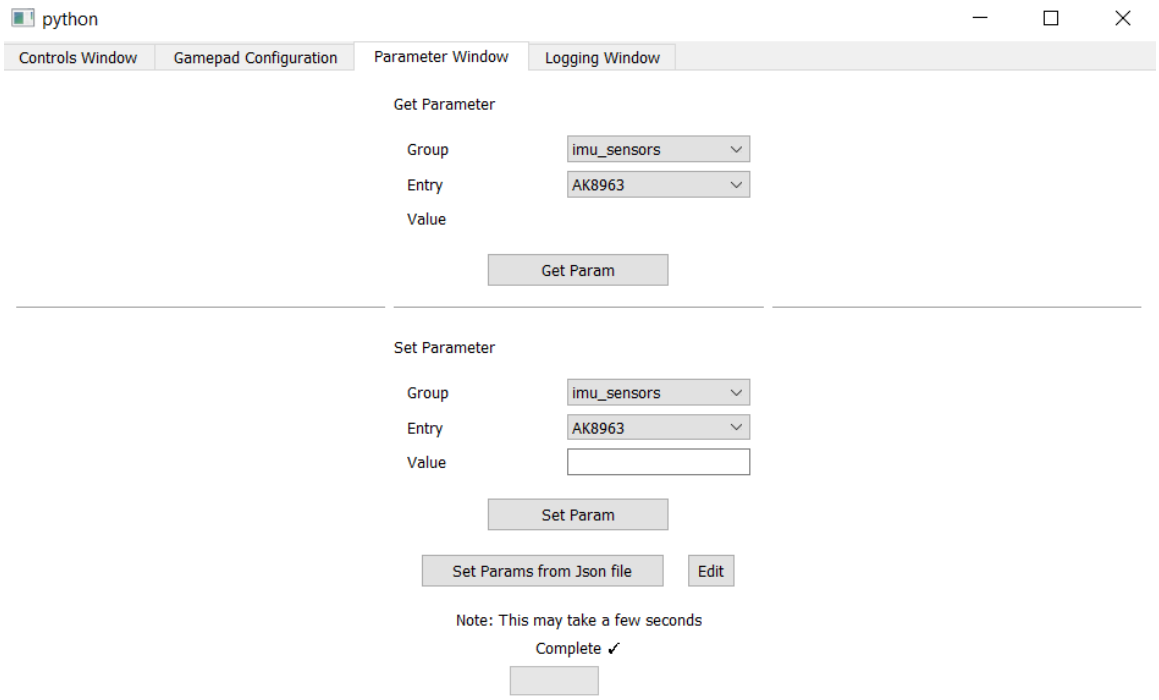


Figure A1.4: Pycrocart GUI, parameter tab

The correct .json file to be uploaded for the flypi can be found within the pifly folder under PID parameters

4. After uploading you are ready to fly! Make sure to have the necessary precautions in place I.E. netting around the flight area and a good pilot. This drone can cut a finger off so safety is #1 priority.

Appendix 2: PCB Design

Rev 1

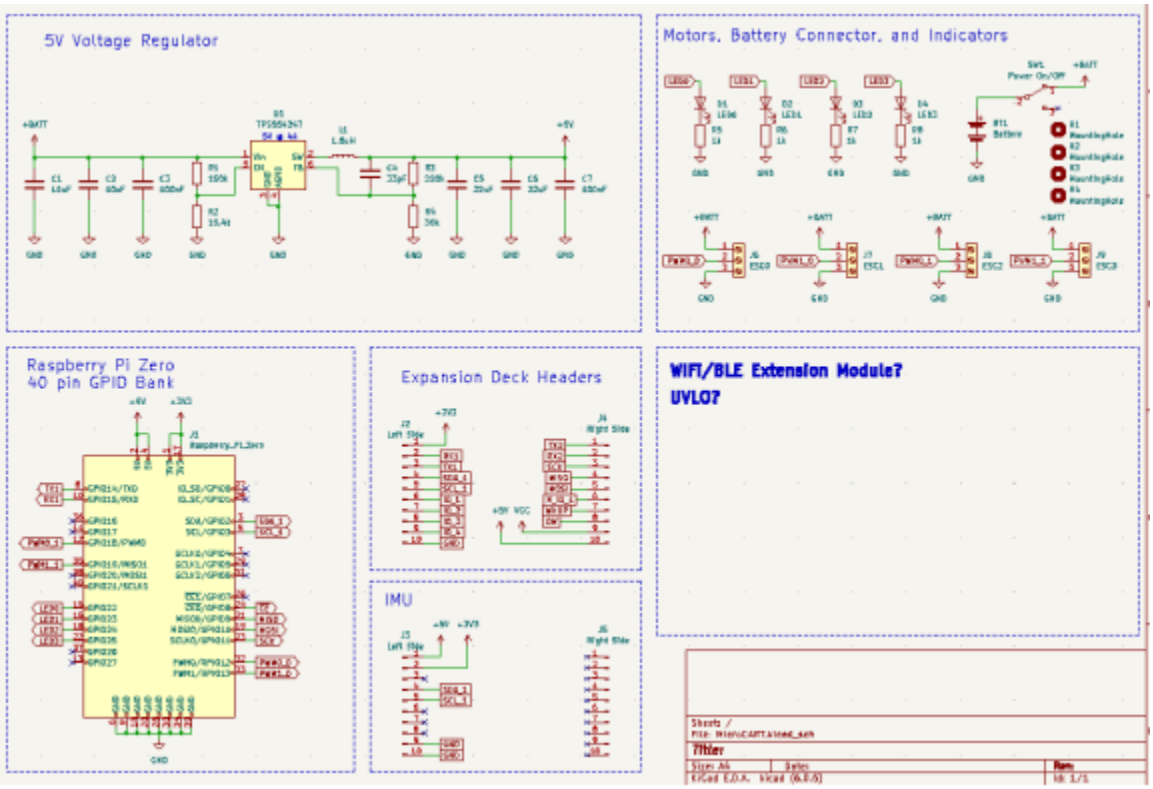


Figure A2.1: Breakout Board Schematic, Rev 1

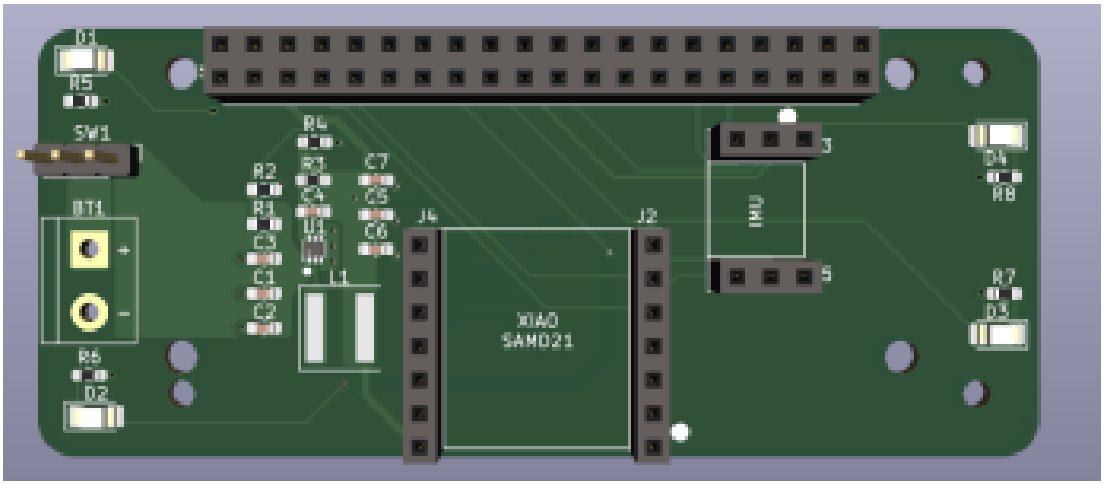


Figure A2.2: Breakout Board 3D View, Rev 1

Rev 2

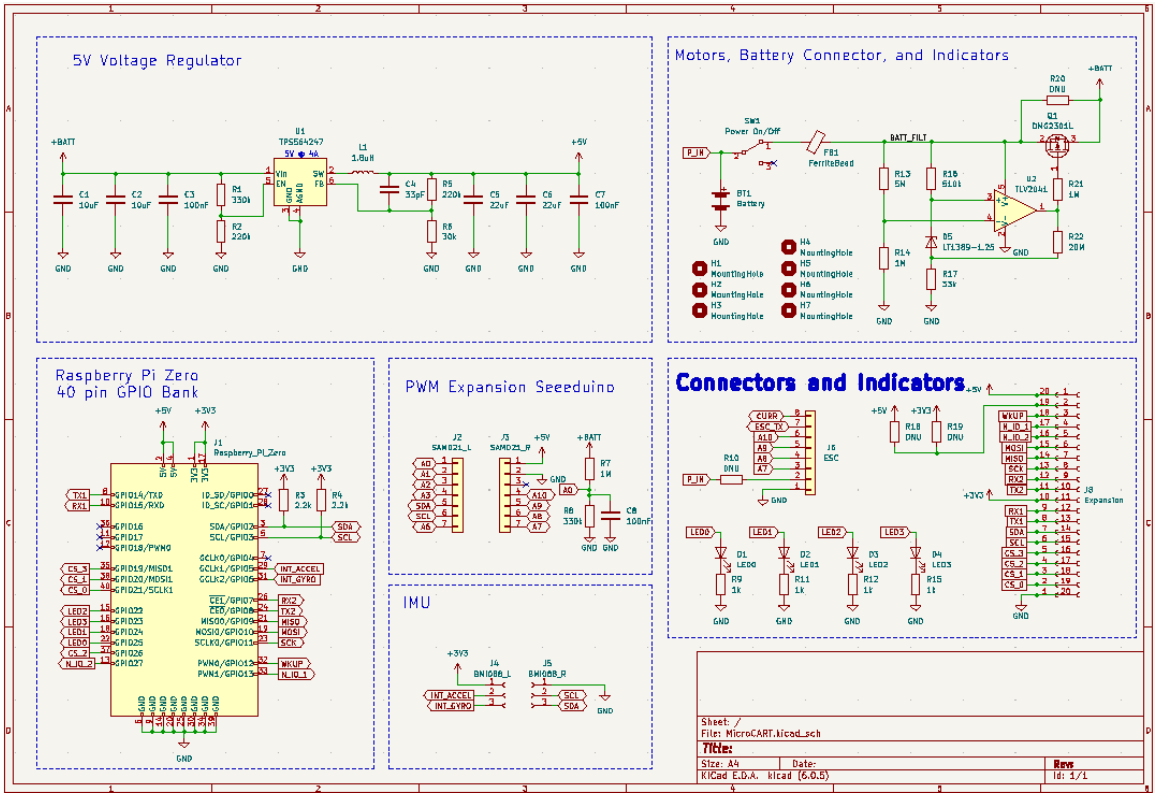


Figure A2. 3: Breakout Board Schematic, Rev 2

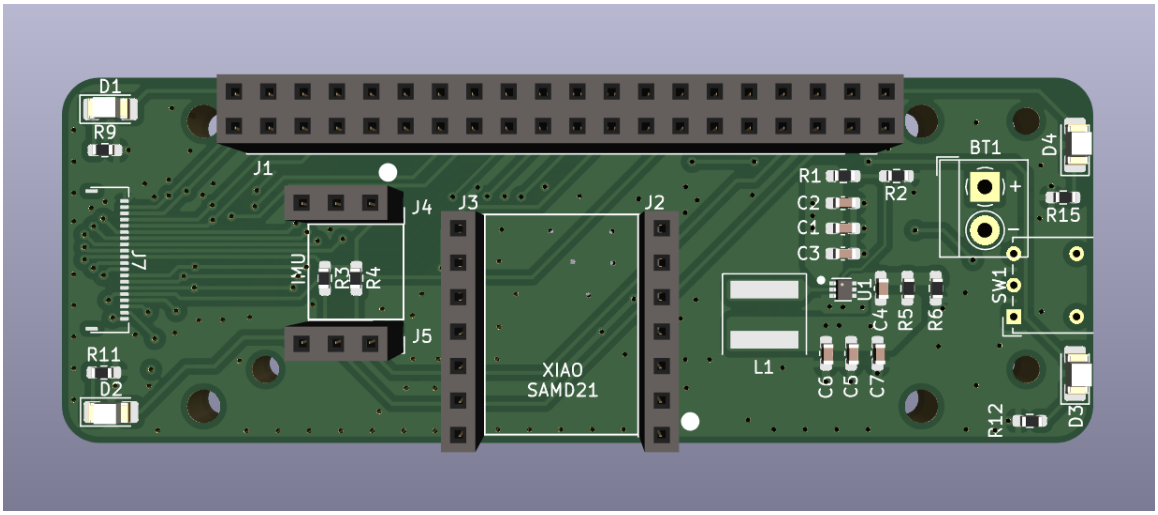


Figure A2.4: Breakout Board 3D View, Rev 2

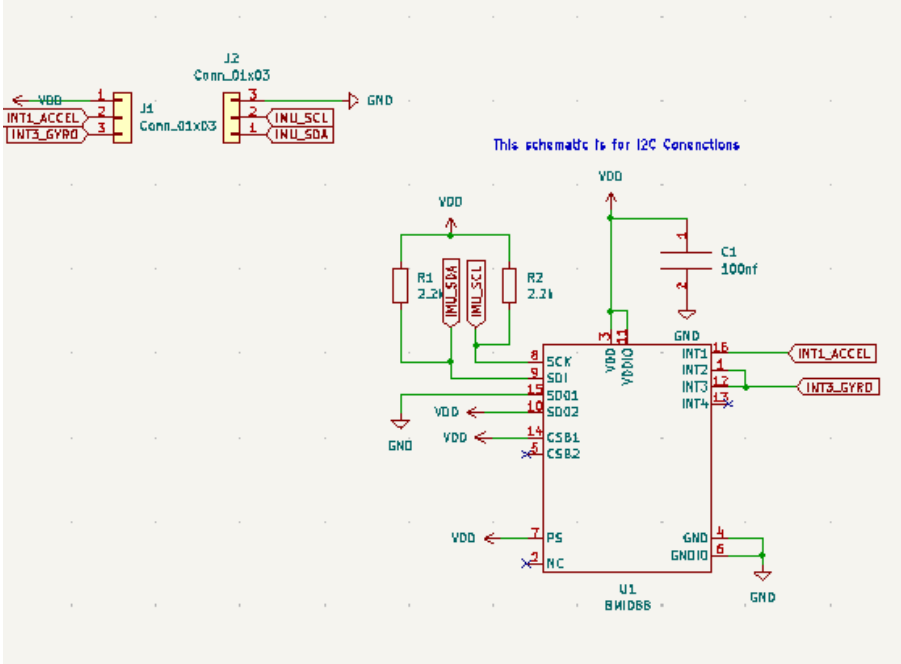


Figure A2.5: IMU Breakout Board Schematic

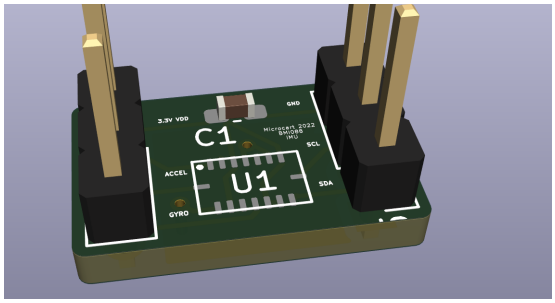


Figure A2.6: IMU Breakout 3D View

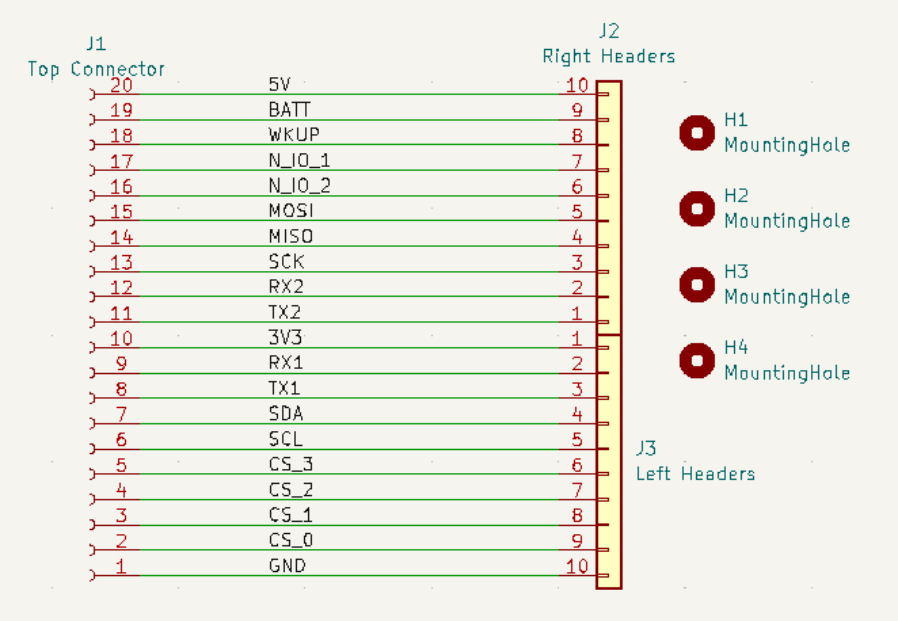


Figure A2.7: Expansion Board Schematic

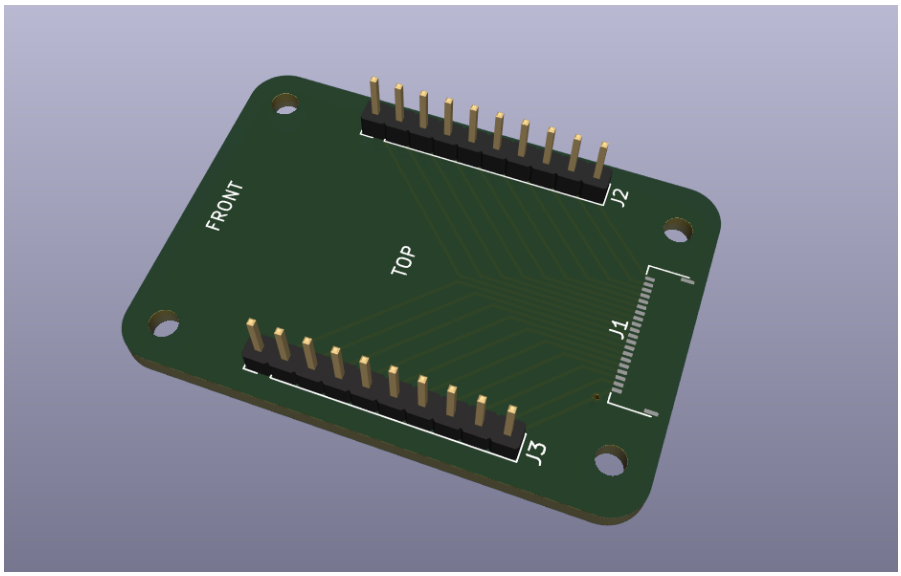


Figure A2.8: Expansion Board 3D View

Appendix 3: List of Components

To determine our different components, we found three options for the majority of sub components on the drone. We tried to find components that would be mostly compatible with each other. We did this to determine the range of options available. Below we have provided the information gathered about our different options. This process allowed us to learn in depth about the different components

of a drone such that we could explore different ideas simply through finding the options available.

Frame

Specification	CinemAh	Cockroach	GEPRC CP 2"
Weight	68g	6.98g	50g
Size	150mm	75mm	115mm
Motor Size	14xx/20xx	0802	1104~1207
Propeller Size	3in/76mm	40mm	2in
Price	$89.99	$7.99	$34.99

Motors

Specs	**Option 1**	**Option 2**	**Option 3**
Weight	4.7g	7.2g	5.25g
Size	F1203	M1106	F1203
9x9 Mounting?	Says “Standard mounting pattern”	Yes	Yes
KV	3500KV, 5500KV	6000KV	7000KV
Configuration	9N12P	-	9N12P
Diameter	16.2mm	16mm	15.5mm
Shaft Diameter	1.5mm	1mm	1.5mm
Internal Resistance	-	-	198mOhm
Current Draw	1.2A @10V Idle	-	0.64A @10V
Power Draw	-	-	192W Max
Battery	3-4S	3-4S	2-3S
Recommended Propeller Size	Recommended for 2”-3” racing builds	“Toothpick builds”	-
Cost Per Motor	$13.99	$13.99	$14.99

Battery

Spec	Option 1	Option 2	Option 3
Style	3S	3S	4S
Nominal Voltage	11.1V	11.1V	14.8V
Capacity	450mAh	550mAh	650mAh
C rate	80C	70C	80C
Dimensions	14.5x32x57mm	17x31x56mm	26x28x59mm
Weight	47g	55g	80g
Price per battery	$8.99	$9.99	$14.17

Propellers

Option #1: 40 mm
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/hqprop-40mm-2-blade-micro-whoop-propeller-1-5mm-shaft-set-of-4-gray.html

- Cost: $2.25

Option #2: 2in
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/betafpv-gemfan-2020-4-blade-propeller-1-5mm-shaft-blue.html

- Cost: $2.50

Option #3: 3in
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/emax-avia-th1609-3-2-blade-propeller-set-of-4.html

- Cost: $2.99

Regulator

Value	Regulator 1	Regulator 2	Regulator 3
Dual Voltage Output?	Yes	Yes	Yes
Max Current Output	3A/2A	3A/2A	3A/2A
Input Voltage Range	4.5~18V	4.5~18V	4.5V~18V
Output Voltage Range	0.76~7V	0.8~15V	0.76~7V
Dimensions	4.4x5mm	4x4mm	4x4mm
Adjustable?	Yes	Yes	Yes
Cost	$2.85	$3.03	$2.99

IMU

Field	IMU 1	IMU 2	IMU 3
Type	Accelerometer, Gyroscope, Temperature, 6 Axis	Accelerometer, Gyroscope, 6 Axis	Accelerometer, Gyroscope, Magnetometer, 9 Axis
Comm Busses	I²C, SPI	I²C, SPI	I²C, SPI, UART
Accelerometer Refresh Rate	12.5-1600Hz	1.6kHz	500 Hz
Accel Sensitivity	±2/±4/±8/±16 g full scale	+- 24 g	+- 8 g
Gyro Refresh Rate	12.5-1600Hz	1.6kHz	1 kHz
Angular Sensitivity	±125/±250/±500/±1000/ ±2000 dps full scale	±125/±250/±500/±1000/ ±2000 dps full scale	+- 2000 dps
ADC Bits	12?	16?	12-bit accel, 16-bit gyro
Supply Voltage	1.71~3.6V and 1.62V	1.2~3.6V	2.4V~3.6V
Dimensions	2.5x3x0.83mm	4.5x3mm	3.8x5.2x1.1mm
Current Consumption	1.25 mA	5mA	10 mA
Cost	$2.78	$10.70	$18.57 - 8 in stock

4.2.3 Decision-Making and Trade-Off

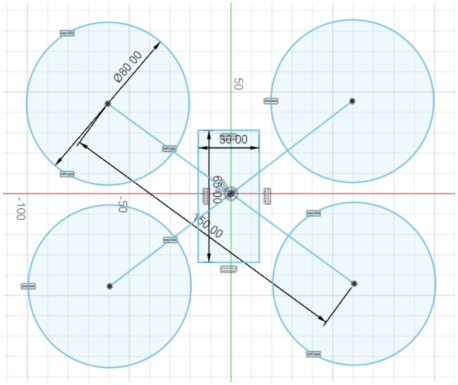
The way we determined the best parts to use was by laying the options out next to each other as seen in the section above, along with listing the pros and cons of each item. We then chose the parts that best conformed to what we thought was beneficial, and conformed with the clients desires. We confirmed that they met our clients desires by showing them the options and picking items we all agreed on. Below is the full document showing the pros and cons, along with the options we used. The highlighted options are the choices we ended up deciding on.

Frames

Option #1
https://newbeedrone.com/products/newbeedrone-cinemah

Pros
- Should have sufficient space

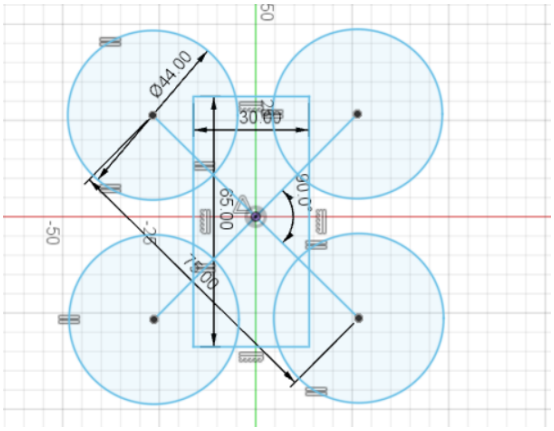
Cons
- Little too big
- Reviews claim it is fragile



Option #2
https://newbeedrone.com/collections/whoop-fpv-drone-frames/products/cockroach75

Pros
- Ideal size
- Different Color Options
- Infinite Warranty

Cons
- Does not have enough space



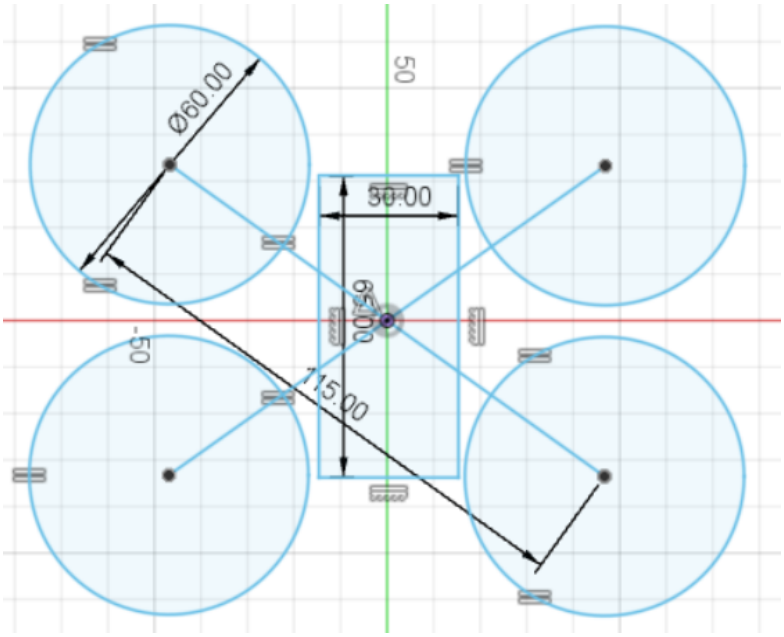
Option #3
https://www.getfpv.com/geprc-cp-2-frame-kit.html?gclid=CjwKCAjwhNWZBhB_EiwAPzlhNqbRsqpoqXHjO5L130LDAjFPiIHzGB3xWbvz65zwj-5GQwNiQD_opBoCokoQAvD_BwE

Pros

- “Minimal” size to fit pi zero
- Looks like butterfly

Cons

- Lots of pieces, requires assembly
- Too heavy?
- Recommends 4S battery
- H-frame



Specification	CinemAh	Cockroach	GEPRC CP 2"
Weight	68g	6.98g	50g
Size	150mm	75mm	115mm
Motor Size	14xx/20xx	0802	1104~1207
Propeller Size	3in/76mm	40mm	2in
Price	$89.99	$7.99	$34.99

We ultimately chose option 3 because we believe it represented the smallest yet most robust drone frame, at the best affordable price. Furthermore, it was the only design we really believed that would actually fit the footprint of the Pi Zero without interfering with the propellers.

Motors

Motor selection will be dependent on frame selection. These motors assume the selection of the butterfly frame.

Option #1:

Link:
https://www.getfpv.com/motors/micro-quad-motors/flywoo-nin-v2-1203-pro-motor-3400kv-4850kv-5500kv-11500kv.html

Specs	**Values**
Weight	4.7g
Size	F1203
9x9 Mounting?	Says “Standard mounting pattern”
KV	3500KV, 5500KV
Configuration	9N12P
Diameter	16.2mm
Shaft Diameter	1.5mm
Internal Resistance	-
Torque Constant	-
Current Draw	1.2A @10V Idle
Power Draw	-
Battery	3-4S
Recommended Propeller Size	Recommended for 2”-3” racing builds
Cost Per Motor	$13.99

Pros

- Customers seem happy
- Max battery power
- More power in small frame
- Light

Cons

- Slowest option
- High Idle power consumption

Option #2
Link:
https://www.getfpv.com/motors/micro-quad-motors/t-motor-m1106-micro-motor-6000kv.html

Specs	**Values**
Weight	7.2g
Size	M1106
9x9 Mounting?	Yes
KV	6000KV
Configuration	-
Diameter	16mm
Shaft Diameter	1mm
Internal Resistance	-
Torque Constant	-
Current Draw	-
Power Draw	-
Battery	3-4S
Recommended Propeller Size	“Toothpick builds”
Cost Per Motor	$13.99

Pros

- Allows for 4S
- Fastest option

Cons

- Heaviest option

Option #3
Link:
https://www.getfpv.com/motors/micro-quad-motors/t-motor-f1203-7000kv-motor.html

Specs	**Values**
Weight	5.25g
Size	F1203
9x9 Mounting?	Yes
KV	7000KV
Configuration	9N12P
Diameter	15.5mm
Shaft Diameter	1.5mm
Internal Resistance	198mOhm
Torque Constant	-
Current Draw	0.64A @10V
Power Draw	192W Max
Battery	2-3S
Recommended Propeller Size	-
Cost Per Motor	$14.99

Pros

- More specs are given
- Smallest option

Cons

- Only 2-3S

Options Laid Out

Specs	**Option 1**	**Option 2**	**Option 3**
Weight	4.7g	7.2g	5.25g
Size	F1203	M1106	F1203
9x9 Mounting?	Says “Standard mounting pattern”	Yes	Yes
KV	3500KV, 5500KV	6000KV	7000KV
Configuration	9N12P	-	9N12P
Diameter	16.2mm	16mm	15.5mm
Shaft Diameter	1.5mm	1mm	1.5mm
Internal Resistance	-	-	198mOhm
Current Draw	1.2A @10V Idle	-	0.64A @10V
Power Draw	-	-	192W Max
Battery	3-4S	3-4S	2-3S
Recommended Propeller Size	Recommended for 2”-3” racing builds	“Toothpick builds”	-
Cost Per Motor	$13.99	$13.99	$14.99

We chose option 1 because it was affordable, and because it was the lightest option.

Batteries

Our frame is capable of holding a 3S-4S battery. The boards require 5V DC. It seems to be generally recommended for drones to use LiPo batteries due to their lighter weight and higher maximum current over Li-Ion batteries.

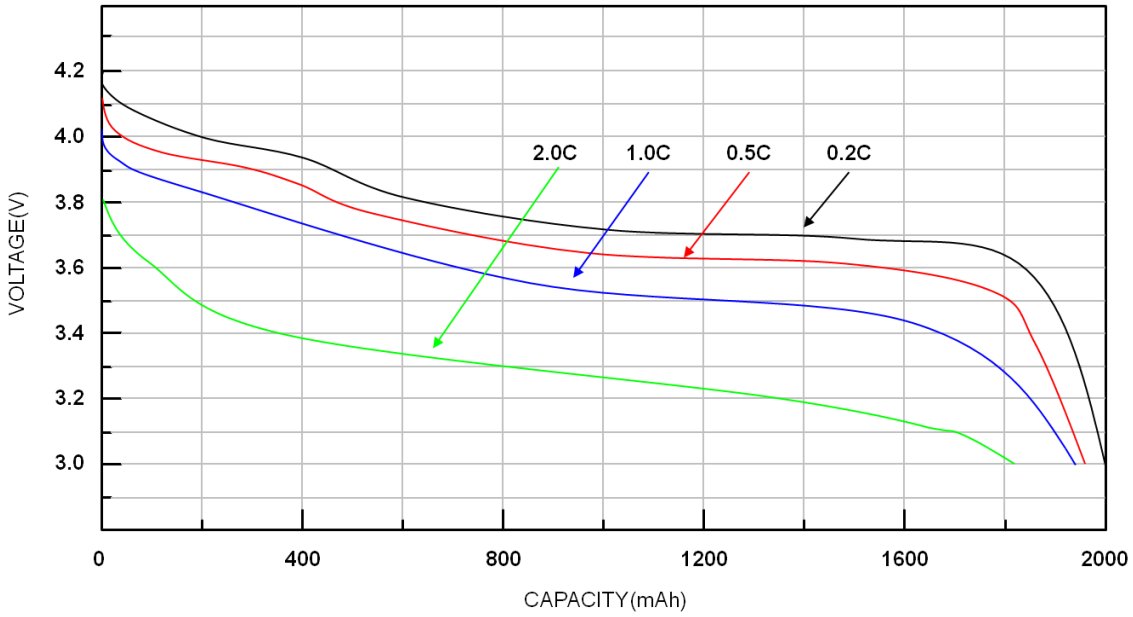
If our battery were 3S-4S our input voltage range would be:

Battery Type	Undervolt	Nominal Voltage (V)	Overvolt
3S	9.6	11.1	12.6
4S	12.8	14.8	16.8

The exact undervoltage and overvoltage specifications are usually not listed, but it seems that between 3.2 volts and 4.2 volts is commonly held as the usable voltage range, with many listing down to 3.0 volts.

LiPo batteries follow the below discharge curve, with varying voltages under different loads due to internal resistance and RC characteristics.

Battery selection will be dependent on the frame and motor selection. These batteries assume the selection of the butterfly frame.



As both the Pi Zero and Zynqberry parts depend on 5V USB power, it will probably be necessary to regulate the voltage supply to the boards.

Option #1
Link:
https://www.racedayquads.com/products/rdq-450mah-3s-fpv-battery-for-2-and-3-quads-80c

Spec	**Value**
Style	3S
Nominal Voltage	11.1V
Capacity	450mAh
C rate	80C
Dimensions	14.5x32x57mm (Fits butterfly well)
Weight	47g
Price per battery	$8.99

Pros
- Smallest dimensions
- Lightest option
- Cheapest option

Cons
- Smallest Capacity
- The butterfly might be able to hold more capacity
- Wide

Option #2
Link:
https://www.racedayquads.com/products/rdq-550mah-3s-fpv-battery-for-2-and-3-quads-80c-xt30

Spec	**Value**

Style	3S
Nominal Voltage	11.1V
Capacity	550mAh
C rate	70C
Dimensions	17x31x56mm
Weight	55g
Price per battery	$9.99

Pros

- Larger Capacity than option 1
- Less long than option 3

Cons

- Wide
- Lower C rating than options 1 and 3

Option #3
Link:
https://www.racedayquads.com/products/3-pack-of-rdq-series-14-8v-4s-650mah-80c-lipo-micro-battery-xt30

Spec	**Value**
Style	4S
Nominal Voltage	14.8V
Capacity	650mAh
C rate	80C
Dimensions	26x28x59mm (Fits butterfly)
Weight	80g
Price per battery	$14.17

Pros

- Large Capacity

Cons

- Heavy
- We don't know the exact dimensions of the butterfly battery area, and this might be a little tight.

Spec	**Option 1**	**Option 2**	**Option 3**
Style	3S	3S	4S
Nominal Voltage	11.1V	11.1V	14.8V
Capacity	450mAh	550mAh	650mAh
C rate	80C	70C	80C
Dimensions	14.5x32x57mm	17x31x56mm	26x28x59mm
Weight	47g	55g	80g
Price per battery	$8.99	$9.99	$14.17

We chose option 2 because it wasn't considerably more expensive or heavier than option 1, and the increases in battery capacity represent a larger gain than the increase in weight. It also has a less long and wide geometry than option 1 which allows us some more confidence when placing it in the butterfly frame.

Propellers

Every propeller was specced with a 1.5mm shaft. Should we choose the motor with a 1mm shaft we will need to find different props. Each propeller was specced to the size of the different frames outlined above. The propeller will be specific to the frame we choose.

Option #1: 40 mm
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/hqprop-40mm-2-blade-micro-whoop-propeller-1-5mm-shaft-set-of-4-gray.html

- Cost: $2.25

Option #2: 2in
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/betafpv-gemfan-2020-4-blade-propeller-1-5mm-shaft-blue.html

- Cost: $2.50

Option #3: 3in
Link:
https://www.getfpv.com/propellers/micro-quad-propellers/emax-avia-th1609-3-2-blade-propeller-set-of-4.html

- Cost: $2.99

We chose option two because they were a compatible size with the butterfly frame, and a compatible shaft size with the motor we selected.

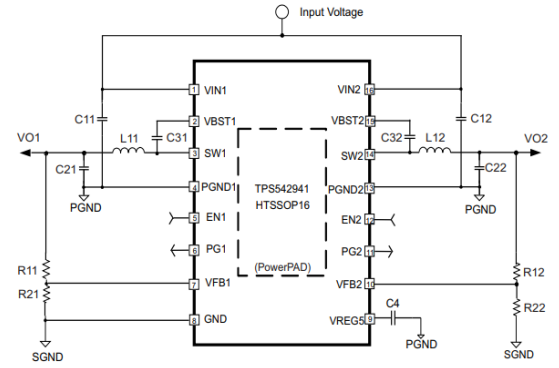
Voltage Regulator

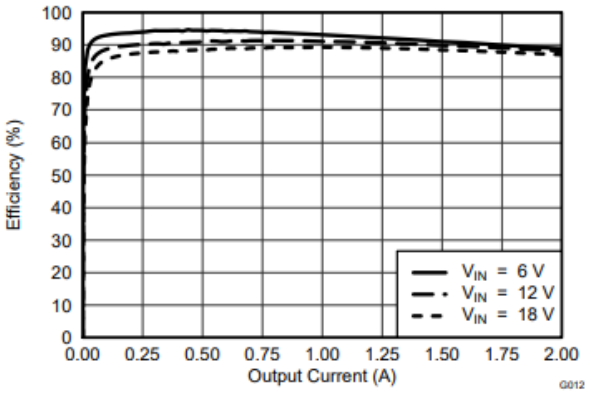
The raspberry pi zero and the zynqberry both require 5V DC. We can design our motor control and IMU board to run on a mixture of battery voltage and 5V DC. We should assume the motors will be running on the battery voltage and require no regulation, while the flight control will require 5V. As such a voltage regulator needs to be able to supply the power requirements of both such boards. It seems reasonable to take the power requirements of the pi zero, the zynqberry and the IMU and multiply by two in order to derate the regulator and provide room for expansion.

Part	**Estimated Maximum Current Consumption**
Pi Zero W	1 A (this is overkill)
Pi Zero 2W	2.5 A (recommended by datasheet)
Zynqberry	500 mA (per USB 2.0 spec)
IMU Option 1	1.25 mA
IMU Option 2	5 mA
IMU Option 3	10 mA

This means the regulator at most needs to power 3A, and because the IMU's operate on between 1.2 and 3.6 volts, we should probably try to find a regulator that supplies both 5 Volt and 3 Volt lines, preferably one intended for powering the USB 2.0 standard.

Option #1
Link:
https://www.digikey.com/en/products/detail/texas-instruments/TPS542941PWPR/3548336





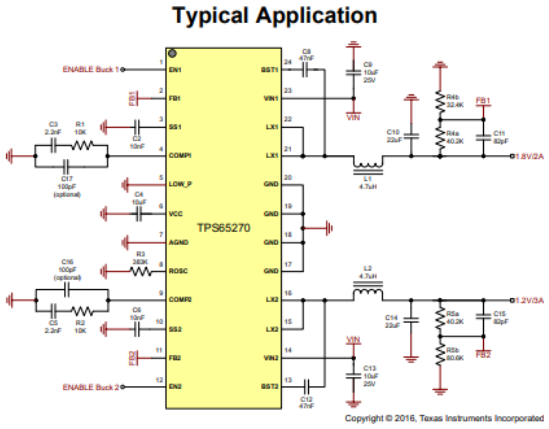
Pros

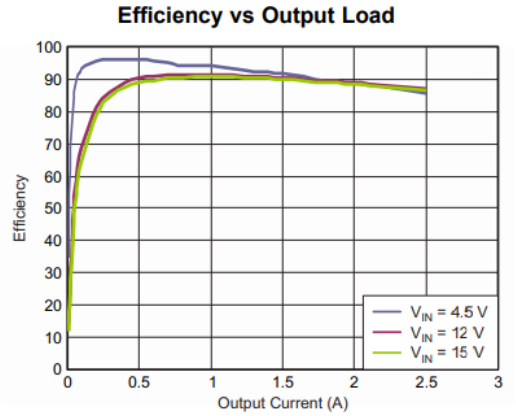
- Most affordable
- Requires few passive components.

Cons

- Pad on bottom requires reflow... but so do the other two chips

Option #2
Link:
https://www.digikey.com/en/products/detail/texas-instruments/TPS65270RGER/2798812





Pros

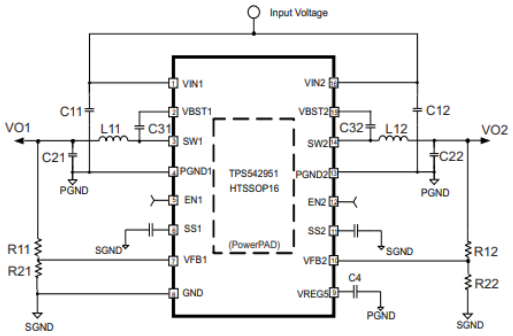
- Largest output voltage range
- Smallest footprint

Cons

- Most expensive
- Requires the most external passive components

Option #3
Link:
https://www.digikey.com/en/products/detail/texas-instruments/TPS542951RSAR/4288723

https://www.ti.com/lit/ds/symlink/tps542951.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1665119468248&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftps542951



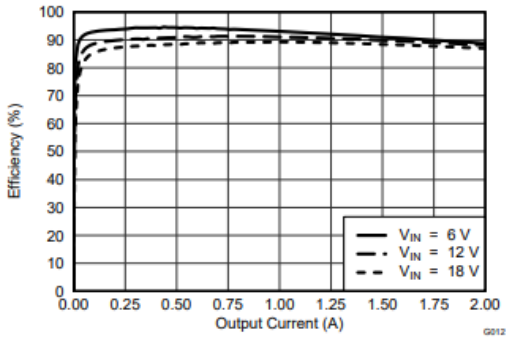


Figure 12. VO1 = 3.3V, Efficiency vs Output Current

Pros

- Nothing different than option 1, same chip smaller package
- Requires few passive components.

Cons

- Nearly identical to option 1 but more expensive

Value	**Regulator 1**	**Regulator 2**	**Regulator 3**
Dual Voltage Output?	Yes	Yes	Yes
Max Current Output	3A/2A	3A/2A	3A/2A
Input Voltage Range	4.5~18V	4.5~18V	4.5V~18V
Output Voltage Range	0.76~7V	0.8~15V	0.76~7V
Dimensions	4.4x5mm	4x4mm	4x4mm
Adjustable?	Yes	Yes	Yes
Cost	$2.85	$3.03	$2.99

We chose option one because it did basically everything the other options did but at a lower price.

IMU

According to hobbyists, a 9DOF Accelerometer that includes a 6-axis accelerometer, gyroscope, and a magnetometer is best.

Option #1
Link:
https://www.digikey.com/en/products/detail/stmicroelectronics/LSM6DS3TR/5180552

Pros

- Cheapest
- Seems to be comparable with other IMU's
- Smallest form factor

Cons

- Cheapest may mean worst
- Not a 9-axis

Option #2
Link:
https://www.digikey.com/en/products/detail/bosch-sensortec/BMI088/8634936

Pros

- Highest gravitational sensitivity
- Used on the previous drone so the software would still be compatible

Cons

- Not a 9-axis

Option #3
Link:
https://www.ceva-dsp.com/wp-content/uploads/2019/10/BNO080_085-Datasheet.pdf
https://www.digikey.com/en/products/detail/ceva-technologies-inc/BNO085/9445940

Pros

- 9 axis
- Seems to be the most intended for a drone
- Arm processor on chip

Cons

- Only 8 in stock
- Price

Field	IMU 1	IMU 2	IMU 3
Type	Accelerometer, Gyroscope, Temperature, 6 Axis	Accelerometer, Gyroscope, 6 Axis	Accelerometer, Gyroscope, Magnetometer, 9 Axis
Comm Busses	I²C, SPI	I²C, SPI	I²C, SPI, UART
Accelerometer Refresh Rate	12.5-1600Hz	1.6kHz	500 Hz
Accel Sensitivity	±2/±4/±8/±16 g full scale	+- 24 g	+- 8 g
Gyro Refresh Rate	12.5-1600Hz	1.6kHz	1 kHz
Angular Sensitivity	±125/±250/±500/±1000/±2000 dps full scale	±125/±250/±500/±1000/±2000 dps full scale	+- 2000 dps
ADC Bits	12?	16?	12-bit accel, 16-bit gyro
Supply Voltage	1.71~3.6V and 1.62V	1.2~3.6V	2.4V~3.6V
Dimensions	2.5x3x0.83mm	4.5x3mm	3.8x5.2x1.1mm
Current Consumption	1.25 mA	5mA	10 mA
Cost	$2.78	$10.70	$18.57 - 8 in stock

We chose option two because it is the same as the IMU on the previous drone which allows us to not change that portion of the software. We believe that by following this process, we were able to come to the best possible solution.