MicroCART

DESIGN DOCUMENT

Team Number: 27

Client: Dr. Phillip Jones
Advisor: Dr. Phillip Jones

Team Members:
Alex Bjerke –- Team Lead
Amith Kopparapu Venkata Boja –- Embedded Software Lead
Theodore Davis –- Embedded Hardware Lead
Grayson Goss –- Technical Lead | CAD Design Lead | Test Station Lead
Hannah Mohamad -— Team Webmaster
Russ Paulsen -— Test Station Assistant
Alfonso Raymundo -— Hardware Design Lead
Trent Woodhouse –- High-Level Software Lead

Team Email: sdmay21-27@iastate.edu
Team Website: https://sdmay21-27.sd.ece.iastate.edu

Revised: 4/25/21 - Version: Final

Executive Summary

Development Standards & Practices Used

Communication standards being used in this project include: WiFi, RF, PWM, and I2C. Industry standards being used in the project include: IPC, IEEE, UASSC, and ITU Radio Regulations. In order to maintain successful development, the team follows the Agile development style.

There are three kinds of standards that we follow in our project. They are:

- Project design and development standards:
  - IEEE 26515-2018: Followed agile practices while accomplishing major goals for the project and updating progress to the client
- Technical Standards:
  - IEEE 802.11-2020: Following the LAN standards when communicating information through WiFi
- IEEE code of ethics

Summary of Requirements

A brief summary of the provided requirements are as follows:

- Mini quadcopter with maximum size of 4.5 x 4.5 inches
- Drone is reprogrammable
- Flight Time of 5-10 minutes
- Communication through radio frequency and WiFi connection
- A ground testing station to secure the drone and measure its rotational data
- Total cost per drone and test station needs to be less than $50 each

Engineering Constraints

The constraints of this project come from the project's requirements and the inherent limitations of the project itself. They are as follows:

- Need a battery that can support 5-10 minutes of flight time and is not too heavy
- Given the size of the drone, the weight needs to be minimized for successful flight.
- Signal transmission speed to and from drone hardware and computation of on-board algorithms must be minimized.
- The drone's microcontroller needs hardware support for floating point arithmetic to minimize computation time
- The test station must be adaptable for changes in drone design.
- Ground control communication with the drone must be reliable for accurate data interpretation

Applicable Courses from Iowa State University Curriculum

CprE 288 Embedded Systems I:

- Overview of embedded systems and embedded programming.
- Interrupts, I/O, Timers, peripherals, resource allocation and optimization.
- Applications of embedded devices.
- Served as a foundation course for us to understand the nature of the MicroCART project.

CprE 489 Computer Networking and Data Communications:

- Learned about wireless communication protocols
- Learned how to utilize and implement data communication through raw TCP and UDP sockets in C

ComS 309 Software Development Practices:

- Introduction to managing software development, process models, requirements analysis, object-oriented design, coding, testing, and maintenance.
- Gave us first hand experience with a project development cycle and project management.

ComS 319 Construction of User Interfaces:

- Overview of user interface design.
- Evaluation and testing of user interfaces.
- Review of principles of object orientation, object oriented design and analysis using UML.
- Design of windows, menus, and commands
- Developing web and windows-based user interfaces.

EE 333 Electronic System Design:

- Introduction to Arduino tutorials, Sensors, Switched-mode power supplies (SMPS), KiCad and Printed circuit boards.
- Gave us first hand experience with a project development cycle and project management. From building the prototype, making the Schematic, Having the PCB made and ordering the parts and, building the Final PCB how to solder the components on to the PCB.

CprE 488: Embedded Systems Design:

- Working with Embedded microprocessors, memory and I/O interfaces. Programming in Embedded software to design computing systems.
- Using concepts like Device Driver development, FPGA programming, PID and microcontroller programming to program user interfaces like camera, VGA and UAV.

New Skills/Knowledge acquired that was not taught in courses

- Project/Product Lifecycle Management - While Computer Science 309 teaches us Software Development practices, that is the closest we get to learning about a project's life cycle. This project has taught us all what goes into designing/implementing a project from beginning to end.
- Pin Assignments - Use datasheets from parts to wire them together. Even though it is easy to do, I don't think any of my EE classes taught me how to use datasheets to do pin & wiring Assignments.
- Github Version Control- Most EE Students had no needs for software like this. But it was nice to learn how to use it outside of class. We all learned how to use this software together & are using it for our project.
- Soldering - Some team members learned how to solder for the first time.
- Arduino - To capture sensor data for the team's test station, an Arduino is being used, so some team members got a chance to learn a bit about Arduinos.
- Computer Aided Design Software - Some of the team members took time during the project to become familiar with CAD modelling software like FreeCAD, Fusion360 and SCAD to properly model the needed changes to the test station and its components to allow for a proper measurement of the drone's movement.
- Drone Design - As a whole, we learned the overall components needed in building a drone. We learned the physics behind the flight of a drone. We also learned the external factors that play a role in drone design. No class at Iowa State teaches the functionality of a drone, so this project helped us understand the factors to consider while building a drone.

Table of Contents

1 Introduction **7**

1.1 Acknowledgement 7

1.2 Problem and Project Statement 7

1.3 Operational Environment 7

1.4 Requirements 7

1.4.1 Functional Requirements 7

1.4.2 Non-Functional Requirements 8

1.5 Intended Users and Uses 8

1.6 Assumptions and Limitations 8

1.7 Expected End Product and Deliverables 8

1.7.1 Drone 9

1.7.2 User-Friendly GUI 9

1.7.3 Testing Station 9

2 Project Plan **9**

2.1 Task Decomposition 9

2.1.1 Initial Tasks 10

2.1.2 Drone 10

2.1.3 Test Station 10

2.1.4 Ground Control 10

2.1.5 Task Decomposition 10

2.2 Project Proposed Milestones, Metrics, and Evaluation Criteria 11

2.3 Project Tracking Procedures 12

3 Design **12**

3.1 Previous Work And Literature 12

3.2 Current Design 13

3.2.1 Drone 13

3.2.1.1 Current Design 13

3.2.1.2 Specific Changes Since 491 (First Semester Design) 17
3.2.2 Ground Control 19
3.2.2.1 Current Design 19
3.2.2.2 Specific Changes Since 491 (First Semester Design) 21
3.2.3 Test Station 21
3.2.3.1 Current Design 21
3.2.3.2 Specific Changes Since 491 (First Semester Design) 23
3.3 Design Analysis 23
3.4 Development Process 24
3.5 Initial Design Plan 24
4 Implementation **25**
4.1 Drone 25
4.2 Ground Control 25
4.3 Test Station 27
4.4 Implementation Not Finished 27
5 Testing Process and Results **27**
5.1 Testing Process 27
5.2 Results 28
6 Closing Material **30**
6.1 Conclusion 30
6.2 References 30
7 Appendices **31**
7.1 Appendix 1 - Operational manual 31
7.1.1 - Drone Operation Manual 31
7.1.1.1 - Prerequisites 31
7.1.1.2 - Operation 31
7.1.2 - Test Station Operation Manual 31
7.1.2.1 - Prerequisites 31

7.1.2.2 - Setup 31

7.1.2.3 - Operation 31

7.1.3 - Ground Control Operation Manual 32

7.1.3.1 - Prerequisites 32

7.1.3.2 - Setup 32

7.1.3.3 - Operation 32

List of figures/tables

Fig 2.2.1 Task Decomposition Diagram

Fig 3.1.1 Crazyflie 2.1

Fig 3.2.1.1.1 Wiring Diagram

Fig 3.2.1.1.2 Pin Assignments for ATSAMD51

Fig 3.2.1.1.2 Wiring Diagram

Fig 3.2.1.1.3 Main Control Loop

Fig 3.2.1.1.4 ESP8266 Software Flowchart

Fig 3.2.1.2.1 Wiring Diagram

Fig 3.2.1.2.2 Cockroach Super-Durable Frame

Fig 3.2.1.2.2 Sensor Data Processing Flowchart

Fig 3.2.2.1.1 Supported Commands

Fig 3.2.2.1.2 Data Packet Format

Fig 3.2.2.1.3 GUI Main Page

Fig 3.2.2.1.4 GUI Connect to Device

Fig 3.2.3.1.1 Test Station Sensor Circuit

Fig 3.2.3.1.2 Image of Test Station

1 Introduction

1.1 Acknowledgement

We would like to express our gratitude to Dr. Phillip Jones who gave us the opportunity to be a part of the team that can contribute to the Electrical and Computer Engineering Department. He has been a wonderful advisor for the continuous support and his guidance. We would also like to acknowledge the May 2021 MicroCart team's contributions to this project.

1.2 Problem and Project Statement

The Electrical and Computer Engineering department wants to create a mini quadcopter drone, test station, and ground station that is suitable for lab experiments in CprE 488. To achieve this, the drone will be capable of reading accelerometer, gyroscope, and magnetometer data; communicate via WiFi; communicate via 2.4 GHz RF; and be reprogrammable. The test station will need to also be able to sense angular velocity in all the axises. Finally, the ground station will be able to log and display information about the drone's motion with data from both the drone and test station.

As a starting point, we will have access to the work done by previous MicroCART teams. Their test station, which is built to measure yaw, will be modified so that with simple adjustments it can measure each axis of rotation. Similarly we will be modifying skeleton code for PID to adapt it to our chosen microcontroller.

1.3 Operational Environment

This drone will be used exclusively indoors, so it will deal with few to no environmental obstacles. The main operational area will be in the lab the class takes place in. This means the operational environment is very controlled and unchanging. The biggest obstacle for this operational area will be flight time of the drone.

1.4 Requirements

The requirements for this project were given to us by our client, Dr. Phillip Jones. This section will cover functional and nonfunctional requirements.

1.4.1 Functional Requirements

The functional requirements for this project include:

- The drone can receive commands through a WiFi and RF signal.
- The drone can interpret user input and respond accordingly.
- The drone has a PID controller that can maintain stable flight.
- The test station can capture rotational data that can be interpreted by the user.
- The ground control can interpret user input, then communicate with the drone or test station accordingly.

1.4.2 Non-Functional Requirements

The non-functional requirements for this project include:

- The drone's size must be no larger than 4.5 x 4.5 inches
- The drone's flight time should be at least 10 minutes.
- The drone's internal program loop execution time should be minimized to improve response time
    - The onboard microcontroller should have hardware support for floating point arithmetic to shorten computation time
- The test station platform can support various mini quadcopter designs
- The final drone and test station should have a cost of under $50 each.

1.5 Intended Users and Uses

MicroCART is designed specifically for students of CprE 488. It's intended use is to provide students an opportunity to work with embedded systems, data collection, and testing. Using Arduino IDE, students can directly program functionality into the drone. Then, they can control the drone with a radio controller, or alternatively, use the provided desktop application to control the drone via wifi. Students can then collect the data from the drone and the testing station and view it within the provided desktop application, and can make adjustments to the code accordingly.

1.6 Assumptions and Limitations

Assumptions

- Users of the drone will be knowledgeable in C
- All applications and uses for the drone are conducted indoors
- Users are given a primer on aspects of the drone before use
- Adjustments for additional payloads will be on the user to code
- This drone will likely be broken due to erroneous code or improper flight
- System will be rechargeable (Li-Po or Li-Ion batteries)

Limitations

- Cost per drone must not exceed $50 USD (client requirement)
- Working around COVID-19 restrictions

1.7 Expected End Product and Deliverables

The overall goal of this project is to build a drone design that will help students in CprE 488 class use it for learning purposes. Since this project is not commercialized, it does not need to be described from a commercial perspective. The following sections describe the end product deliverables for each project component.

1.7.1 Drone

Drone Hardware:

- It will contain the following components successfully wired together:
    - WiFi chip/module
    - Programmable MCU
    - IMU
    - Four brushless motors
    - RF transmitter/receiver module
- It should be powered by a rechargeable battery and programmable by USB.
- The drone's brushless motors should be controlled by the IMU and be regulated by Mosfets and Capacitors.

Drone Software:

- This drone needs to read sensor data and determine the orientation of the drone.
- It needs to send the orientation information to ground control for further analysis.
- It should be able to stably perform actions on the drone based on the user input.
- The software setup should let students to easily program the drone for flight automation.
- The software should follow a modular design for an easy-to-read design

1.7.2 User-Friendly GUI

This is the next phase of a user- friendly experience. One of the best ways to truly understand a drone's functionality is to read its data. Having a user-friendly graphical user interface will help students constantly get data from the drone and the ground station for future analysis. There will also be a command-line interface option for communicating with the drone.

1.7.3 Testing Station

The test station is required to help tune and test the drone. The platform helps tune the drone by allowing for a stable position for the drone to rotate without the chance of flight-based damages. In this way, the drone's code and algorithms can be tuned without any chance of damage. The test station will have the capability to record rotational data in three dimensions (Roll, Pitch and Yaw), which will aid in the tuning of the drone.

2 Project Plan

2.1 Task Decomposition

This section describes the task composition created in the first semester of this project. It includes tasks and steps to complete.

2.1.1 Initial Tasks

The first major project-based tasks that were necessary were establishing requirements and doing preliminary research. The preliminary research was meant for the team to gain a sufficient understanding about the operations of quadcopters and what goes into designing a mini quadcopter.

2.1.2 Drone

- PCB/Hardware
    - The PCB/Hardware component of the drone is perhaps the biggest subgroup. The tasks here include determining parts and sensors to use, designing the PCB, ensuring size requirements are met, prototyping, and testing.
- Embedded Software
    - The embedded software component of the drone reflects any software "living" on the quadcopter. This subgroup will need to research libraries supported by the chosen microcontroller, research other open source software used in programmable drones, and be responsible for the communication between the drone and external sources (ground station and RF controller).

2.1.3 Test Station

- Sensors
    - There will be a time when the test station's sensors and microcontroller need to be selected.
- CAD
    - There are previous teams' CAD files that will need to be reviewed. Also, once the team determines the sensors and microcontroller to use, someone will need to prototype/design the final product.
- Embedded Software
    - There will need to be software embedded in the test station responsible for sending data to the ground control.

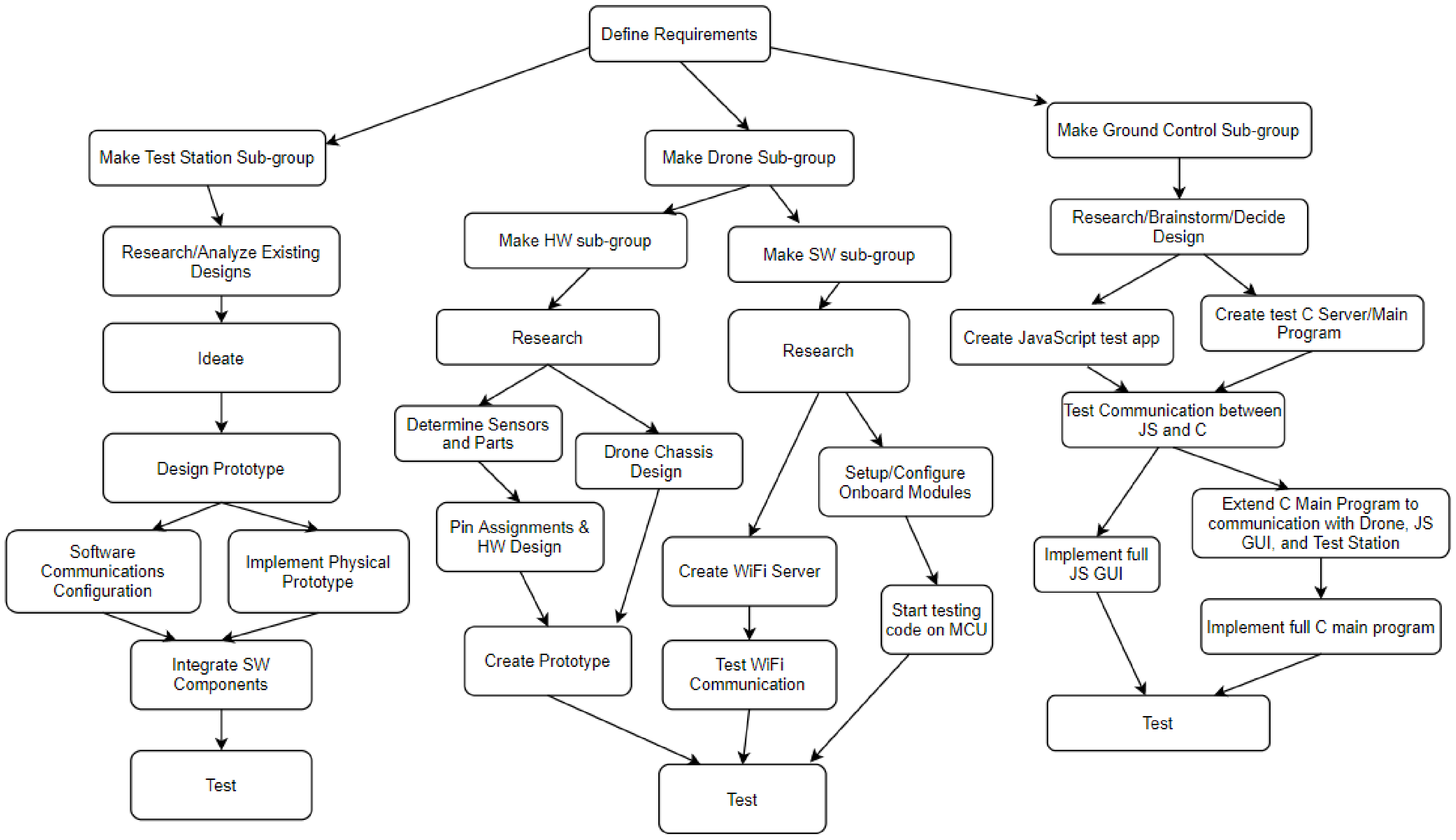
2.1.4 Ground Control

As there are no hard-set requirements given for the ground control program, the first major task is determining what is needed, possible, and feasible. After making these decisions, the next step is to begin making a skeleton application. During this process, documentation should be made.

2.1.5 Task Decomposition

Below is a high level task decomposition diagram. It covers all major and necessary tasks for the successful completion of the project. For most tasks that involve any implementation, there will be testing that takes place, but not all of it is shown in Fig 2.1.1.

Fig. 2.1.1 Task Decomposition Diagram



Define Requirements
Make Test Station Sub-group
Make Drone Sub-group
Make Ground Control Sub-group
Research/Analyze Existing Designs
Ideate
Design Prototype
Software Communications Configuration
Implement Physical Prototype
Integrate SW Components
Test
Make HW sub-group
Make SW sub-group
Research
Research
Determine Sensors and Parts
Drone Chassis Design
Pin Assignments & HW Design
Create Prototype
Setup/Configure Onboard Modules
Create WiFi Server
Test WiFi Communication
Start testing code on MCU
Test
Research/Brainstorm/Decide Design
Create JavaScript test app
Create test C Server/Main Program
Test Communication between JS and C
Implement full JS GUI
Extend C Main Program to communication with Drone, JS GUI, and Test Station
Implement full C main program
Test

2.2 Project Proposed Milestones, Metrics, and Evaluation Criteria

The biggest milestones of this project are:

- Gain sufficient understanding in the functionality of a drone
- Create a control algorithm to stabilize the roll and pitch axis of the drone.
- Forward sensor data from the drone to the ground control program with at least 90% accuracy.
    - Communicating with the drone motors for stable flight
    - Constant communication with all the sensors on the IMU and the drone
    - Receiving on instructions from the RC controller to control the flight of the drone
    - Using an ESP8266, having all the necessary data sent to and from the MCU through WiFi
- Build an effective prototype of the drone that incorporates all the sensors and motors while meeting the prior requirements.
- Design and implement a test station that can successfully read rotational data of the drone about a given axis.

- Send the captured data to the ground control program.

- Build an user-friendly application that acts as the ground control by receiving all the data from the drone and the test station and displaying it on a GUI.
    - Simplicity in the usage of the application
    - Ability to read data from the drone and test station based on user requirements.

2.3 Project Tracking Procedures

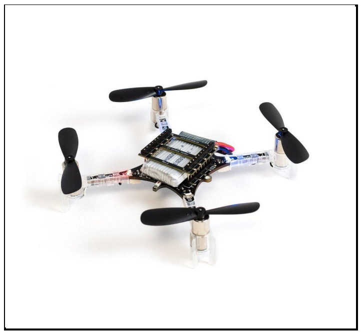
Our team will be following an Agile workflow with progress reports every week. To track individual tasks and milestones we will be using GitLab's Issues and Issue Board feature. This is a way we can assign tasks, give them dates, and track their status. Additionally, we also hold weekly meetings with our client to update them and gather their input. We also fill out a weekly progress report for our client which is also used to create the bi-weekly report for the class.

3 Design

3.1 Previous Work And Literature

For this project we had ample sources to look to for inspiration. We had last year's MircoCART's git repo we could browse for ideas as well as some open source materials such as the Silverware Wiki [1] for drone basics and how to flash commercial drones. Our model for success is to accomplish similar functionality to the Crazyflie 2.1 [2] seen in [Fig 3.1.1], so we were also using it as a reference for hardware specifications and similar details.

Fig 3.1.1 Crazyflie 2.1 [2]



The Crazyflie is a reprogrammable quadcopter of the size of which we are aiming for. The reason why our client chose to not purchase these is the price tag of $195.00. This design is built for indoor/outdoor high altitude flight. Our sole purpose for the MicroCART is a reprogrammable quadcopter capable of indoor, low altitude flight.

3.2 Current Design

This section covers the design aspects of the drone, ground station, and test station.

3.2.1 Drone

3.2.1.1 Current Design

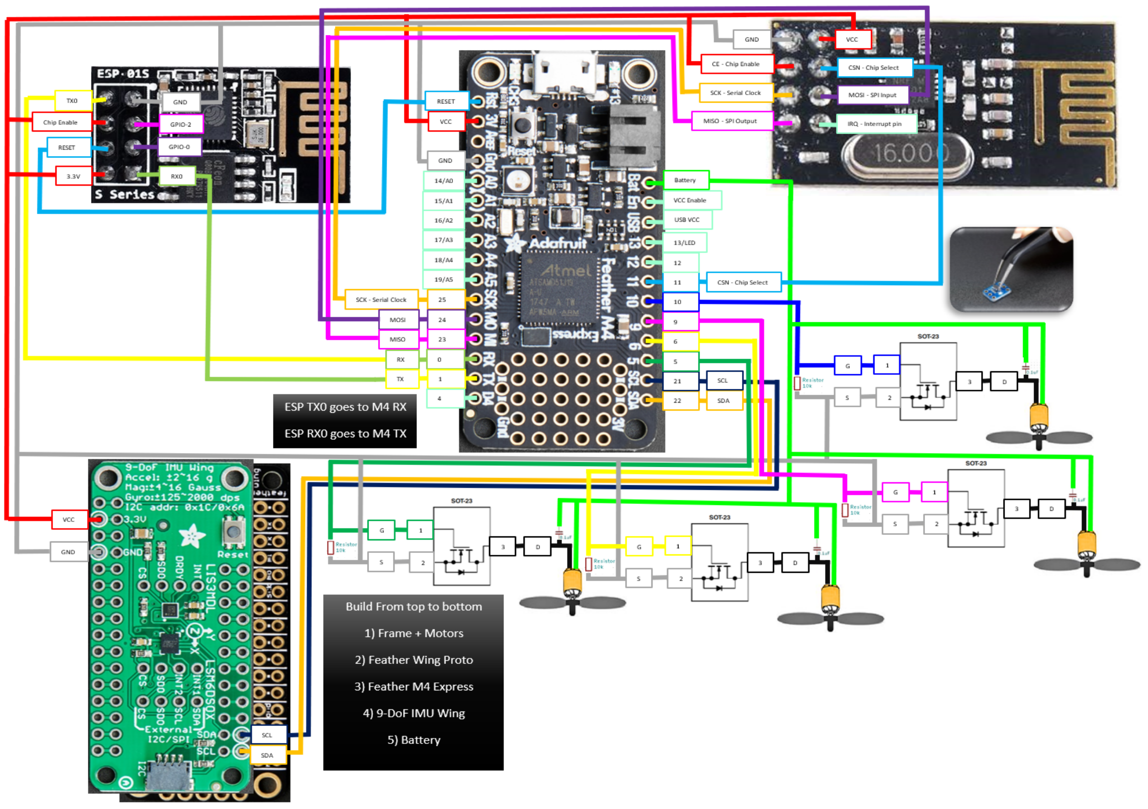
- Drone Hardware Design:

  Bill Of Materials:

  - Microcontroller (Adafruit Feather M4 Express)
  - Accelerometer + Gyro (9-DoF IMU Wing)
  - WiFi Transceiver & Receiver (ESP8266 ESP-01)
  - RF Transceiver & Receiver (NRF24L01+ 2.4GHz transceiver module)
  - Feather Wing Proto
  - Four 30 mm Propellers
  - Four 6mm Brush Motor @KV: 19,700
  - Four SOT-23 with Mosfets
  - Four Mosfets N-CH 8V 6A
  - Four 0.1 uf Capacitors
  - Four 10k Resistors
  - Four Schottky Diodes
  - Two Frames (NewBeeDrone Cockroach Brushed Super-Durable Frame 65mm)
  - 36 30 mm Long Connector Pins
  - Lithium Battery at 3.7V and 2.5 AH
  - Straight forward programing for students
  - Ground control station

Fig 3.2.1.1 is a non-schematic wiring diagram that we used to connect all the components.

Fig 3.2.1.1.1 Wiring Diagram



ESP TX0 goes to M4 RX

ESP RX0 goes to M4 TX

Build From top to bottom

1) Frame + Motors

2) Feather Wing Proto

3) Feather M4 Express

4) 9-DoF IMU Wing

5) Battery

- Drone Software Design:

For the Feather Express m4 we are using the following pin assignments shown in Fig 3.2.1.2.

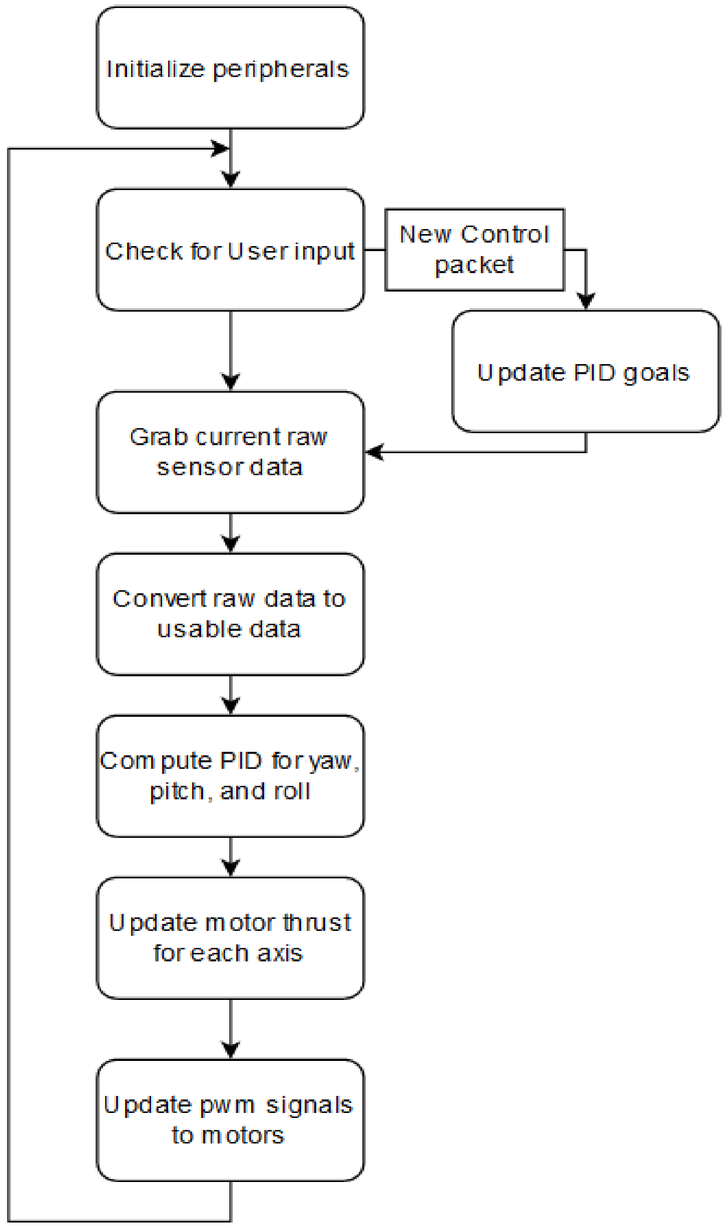
Fig 3.2.1.1.2 Pin Assignments for ATSAMD51

Pin Number	Port	Periphals	Use
Pin 0	PB17	UART RX	Connect to ESP8266
Pin 1	PB16	UART TX	Connect to ESP8266
Pin 5	PA16	TCC1:W[0]	Signal for Motor
Pin 6	PA18	TCC1:W[2]	Signal for Motor
Pin 8	PB03	GPIO / LED	On board Neo Pixel
Pin 9	PA19	TCC1:W[3]	Signal for Motor
Pin 10	PA20	TCC0:W[0]	Signal for Motor
Pin 13	PA23	GPIO / LED	On board LED
Pin 21	PA13	I2C SCL	Connect to IMU
Pin 22	PA12	I2C SDA	Connect to IMU
Pin 25	PA17	SPI SCK	Connect to NRF
Pin 24	PB23	SPI MOSI	Connect to NRF
Pin 23	PB22	SPI MOSO	Connect to NRF

- Feather M4 Express

  Fig 3.2.1.3 shows the general flow that our code will follow while operating the drone. Each iteration of the loop should take less than 10ms to complete in order to have stable and reactive output from the PID controller.

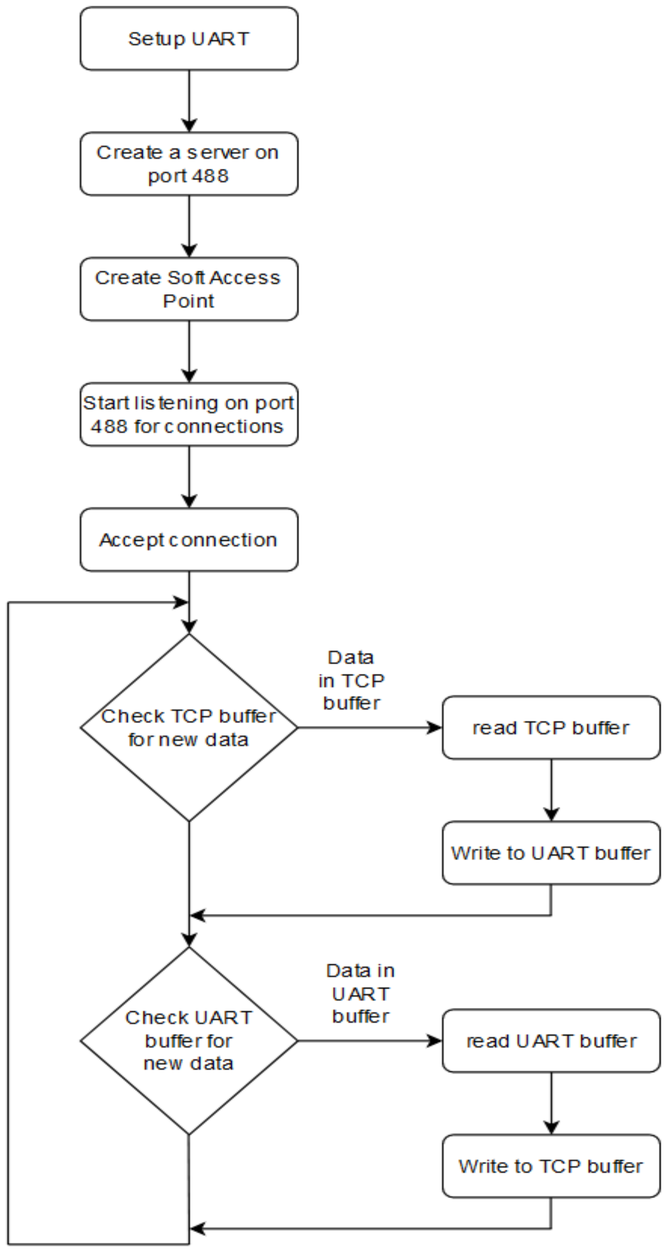
Fig. 3.2.1.1.3 Main Control Loop


Initialize peripherals
Check for User input
New Control packet
Update PID goals
Grab current raw sensor data
Convert raw data to usable data
Compute PID for yaw, pitch, and roll
Update motor thrust for each axis
Update pwm signals to motors

- ESP8266:

  The ESP8266 software follows the general flow of Fig. 3.2.1.4

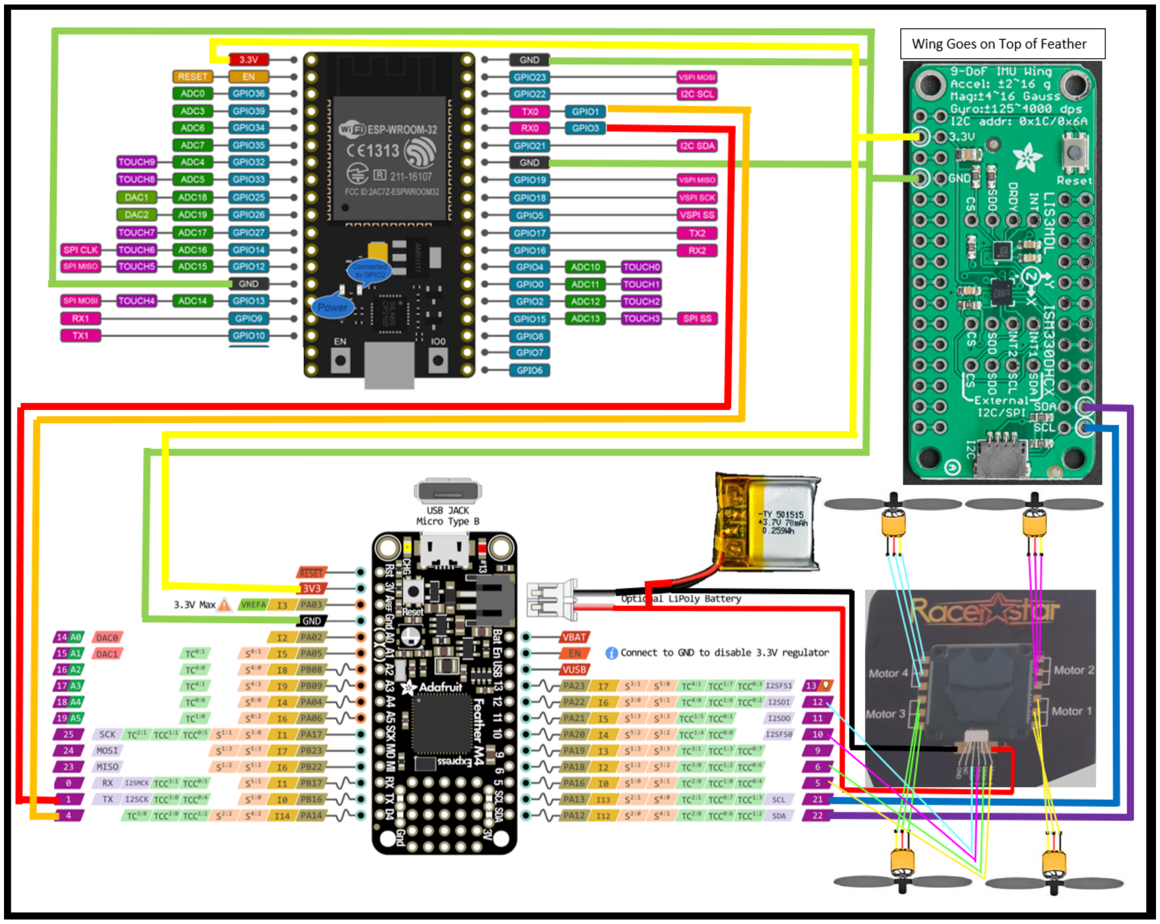
Fig. 3.2.1.1.4 ESP8266 Software Flowchart



3.2.1.2 Specific Changes Since 491 (First Semester Design)

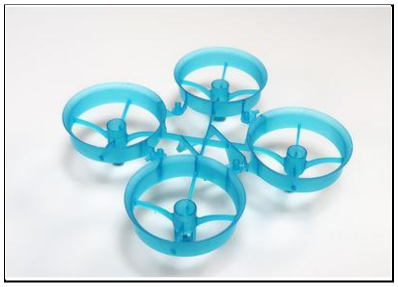
We removed the Esp-Wroom-32 for the ESP8266 (ESP-01) and changed the wiring from this Fig 3.2.1.2 Wiring Diagram to Fig 3.2.1.1 Wiring Diagram. Overall, the design of the drone from 491 is not the same as the current design for 492.

Fig 3.2.1.2.1 Wiring Diagram



- Drone Chassis:
    - Instead of designing our own chassis, we ended up using the Cockroach Super-Durable Frame as our chassis depicted in Fig 3.2.1.3.

Fig 3.2.1.2.1 Cockroach Super-Durable Frame [3]



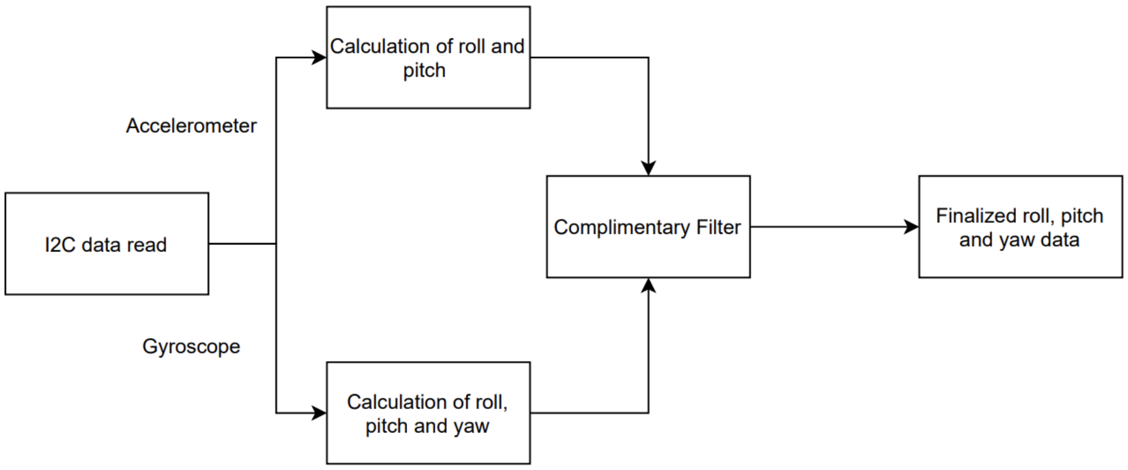
- Drone Software:
    - ATSAMD51:
        - Reading Sensor Data:

          Converting the raw accelerometer and gyroscope data collected from IMU into orientation angles of the drone.[Fig. 3.2.1.3]

        - PID Control

          Making motor control decisions based on the sensor data received and desired trajectory by the user through the PID algorithm.

Fig. 3.2.1.2.2 Sensor Data Processing Flowchart


I2C data read
Accelerometer
Gyroscope
Calculation of roll and pitch
Calculation of roll, pitch and yaw
Complimentary Filter
Finalized roll, pitch and yaw data

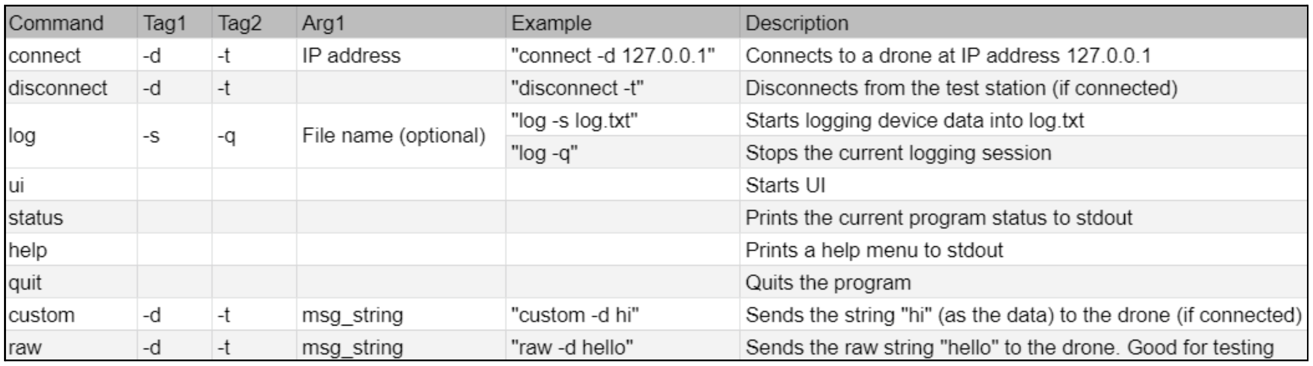
3.2.2 Ground Control

3.2.2.1 Current Design

The internal, main program was written in C:

- Upon start of the program, the user will be using a command-line interface. They will be prompted with “>>>”.
- The list of supported commands can be found by entering “help” and pressing enter, and also in the following table (Fig. 3.2.2.1.1 Supported Commands):

Fig. 3.2.2.1.1 Supported Commands



Command	Tag1	Tag2	Arg1	Example	Description
connect	-d	-t	IP address	"connect -d 127.0.0.1"	Connects to a drone at IP address 127.0.0.1
disconnect	-d	-t		"disconnect -t"	Disconnects from the test station (if connected)
log	-s	-q	File name (optional)	"log -s log.txt"	Starts logging device data into log.txt
				"log -q"	Stops the current logging session
ui					Starts UI
status					Prints the current program status to stdout
help					Prints a help menu to stdout
quit					Quits the program
custom	-d	-t	msg_string	"custom -d hi"	Sends the string "hi" (as the data) to the drone (if connected)
raw	-d	-t	msg_string	"raw -d hello"	Sends the raw string "hello" to the drone. Good for testing

- Any device communication (except for the “raw <-device> <msg_string>” command) will pack data into data packets. The following figure shows the data packet format (Fig. 3.2.2.1.2 Data Packet Format).

Fig. 3.2.2.1.2 Data Packet Format

Field:	Start Char	Msg Type	Msg ID	Data Length	Data	Checksum
Bytes:	1	2	2	2	x	1

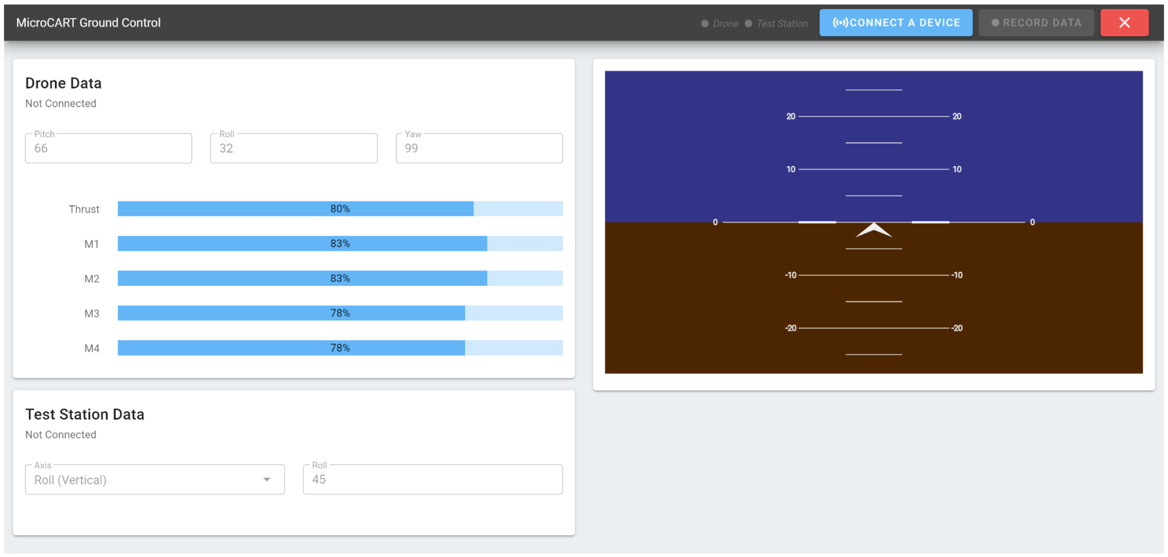
- The main loop of the program does the following:
  - Prompt the user if needed
  - Zero all buffers used
  - Using select(..), determine what socket(s) are ready to be read from, if any. These include: drone socket, test station socket, ui socket, stdin.
  - Process anything from stdin or the UI as user input. If the input corresponds to a valid command, execute the command entered.
  - Process anything received by the drone or test station by:
    - Unpacking the data

        - Determining what was received
        - Perform necessary actions for the data packet. For example, logging any sensor data received from the drone
    - Repeat
- The file packet_info.h serves a purpose that is not reflective of its name. It contains miscellaneous definitions that can be changed by the user. It contains the drone's port number, test station's port number, max packet size allowed, and the internal command used for starting the UI (this should not be changed unless necessary).

UI written in JavaScript:

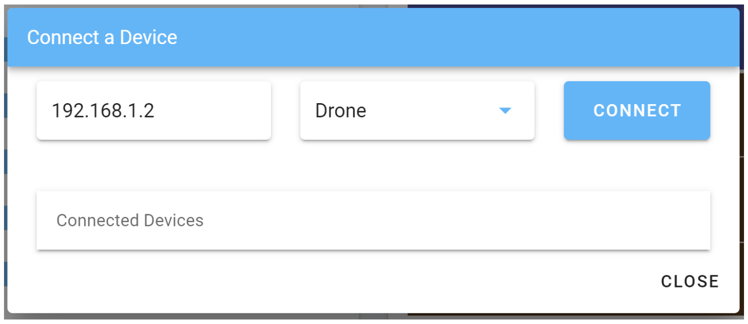
- Started by running the "ui" command in the command-line interface
- A window will pop up in the user's browser, as shown below in Figure 3.2.2.1.3.

Fig. 3.2.2.1.3 GUI Main Page



- From there, the user can click "Connect a Device" and connect to either a drone, testing station, or both.

Fig. 3.2.2.1.4 GUI Connect to Device



- Connected devices will display current data readings from the respective devices.
- If a drone is connected, clicking the "Record Data" button will begin logging data from the drone (and testing station, if connected)
- Clicking on "Log files" will pull up all logs of recorded flights

3.2.2.2 Specific Changes Since 491 (First Semester Design)

Main C Program:

- No longer using threads for communications. Instead, using the C function select() for determining all sockets that have new data to be read
- No longer using websockets for UI communication (just TCP)

UI:

- Added a node.js server to act as a communication server between the main C program and the GUI
- Updated GUI to make drone and test station data act as the main dashboard
- Added ability to establish connections to devices from GUI (acting through C Program)
- Added file reading to list log files and display their contents

3.2.3 Test Station

3.2.3.1 Current Design

The test station uses the magnetic shaft encoding sensor to measure angular data.

The current method of interfacing with the test station sensor is with an Arduino Nano plugged into a PC via USB and contains the circuit shown in Fig 3.2.3.1.1.

The following is a description of Figure 3.2.3.1.1:

- Red wire - Sensor's input voltage
- Blue wire - Sensor's ground
- Green wire - Senor's analog output
- Resistor - 4.7k ohms
- Capacitor - 100 picofarads

Fig 3.2.3.1.1 Test Station Sensor Circuit

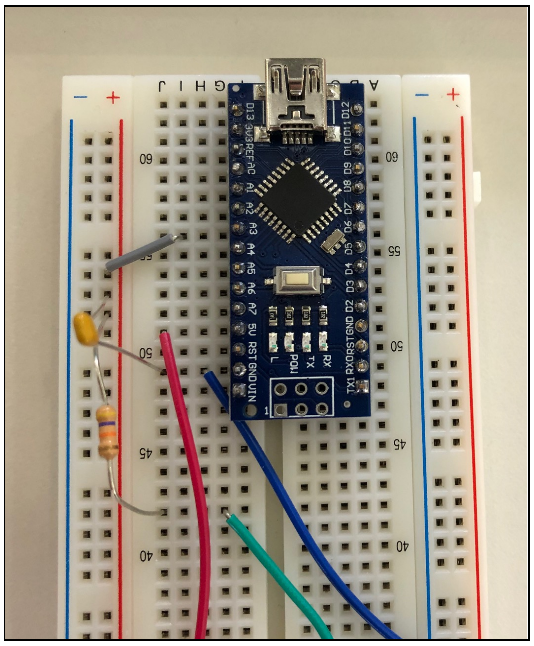
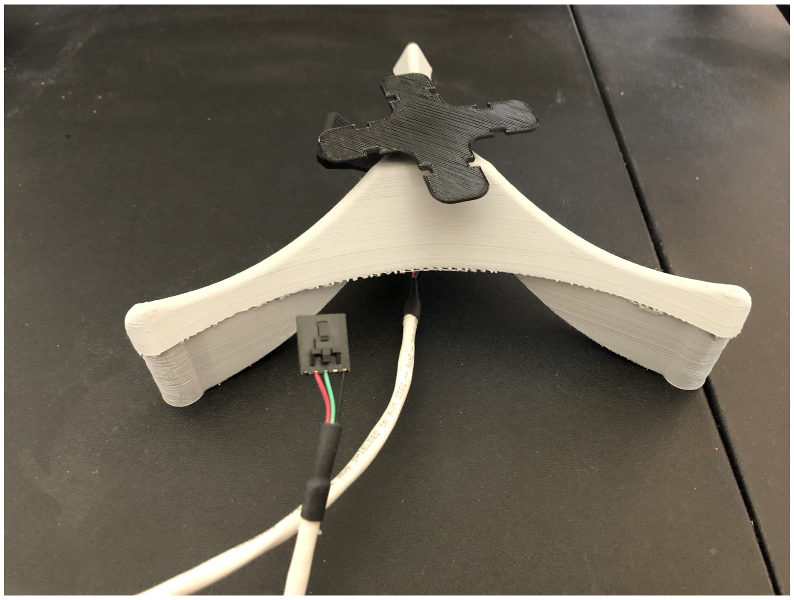


Fig 3.2.3.1.2 Image of Test Station



This design also includes the base station which secures a testing platform. The testing platform interacts with the magnetic sensor to record rotational data. There are two platforms so the full range of motion (yaw, roll, pitch) can be measured.

3.2.3.2 Specific Changes Since 491 (First Semester Design)

Below are the changes from the first semester design:

- Added legs to the base station. This allows for a better positioning of the magnetic sensor and provides some stability for both station orientations
- Developed two test platforms. The drone is secured to these platforms, which when inserted and locked into the base station, interface with the magnetic sensor. This allows us to get rotational data of the platform's respective axis whenever the drone moves.
- Developed positional capture code. Using an Arduino and the magnetic sensor, rotational data is sent in 10-bit packets to the ground station where the data can be interpreted on a graph showing the change in position from 0-360 degrees. We are still working on exporting this time-series data into an excel file for proper analysis.

3.3 Design Analysis

This section will analyze strengths and weaknesses of the design specified in section 3.2 of this document.

Strengths:

- Adaptability: By using open source software and hardware, changes to the entire project can be made with relative quickness.
- Test Station Precision: The accuracy of 3D printing has allowed for the drone's test platform to interface with it's sensor module with precision.
- Hardware Modularity: The drone is designed from several different modules which are straightforward to replace or add to with newer modules.
- Software Modularity: The code is designed to have drop in modules so that the user doesn't need every code file to program the drone.

Weaknesses:

- Weight: The drone is too heavy for successful flight, and the center of gravity is not centered.
- Durability: The drone is not protected from any form of rough landing.
- Height: The height of the first prototype prevents the drone from being properly attached to the test station.
- Wiring Organization: The drone's wiring is messy and hard to follow if changes need to be made.
- Software Drivers: Have to recompile code in Atmel Start to update driver definitions.

3.4 Development Process

Our team is using the Agile development process. Given the nature of the project, we are able to split the project into multiple sub-parts, then work in an agile style from there. The group is split into four main sub-groups, listed below. While the people listed for each group are the main contributors, some people also have contributions to different groups.

- Ground Control (Trent, Alex)
- Test Station (Russ, Grayson)
- Drone Software (Amith, Theodore)
- Drone Hardware/Design (Theodore, Alfonso, Hannah, Grayson)

Since these subgroups are able to work in parallel, it is natural to follow the Agile process. Additionally, we used Git Issues to assign, track, and manage tasks.

3.5 Initial Design Plan

This section describes the initial design plan from the end of the first semester of senior design.

Due to the novel nature of this drone design, we will require an iterative design plan to deal with setbacks in any part of the project. We will start out with just a simple prototype. No chassis, just the electronic components wired together to see if they all work together as planned. Then, after this is shown, we will construct a full drone with chassis and wiring harness. This will be the basis of our work with both the ground station and the testing station. We will start with testing how the drone flies and how all the components (sensors, motors, computation units, communication units) work together. After initial errors are ironed out, we will start on the ground station, which is the core controller of the drone. The ground station will connect over several wireless protocols to the drone and allow for us to test remote flight of the drone as well as begin crafting a UI for the intended users. After the ground station is completed, we will work on the test station. This hardware integrates with the ground station as a means of communicating data from the test station to the ground station, where data can be read and used to tweak the drone to a proper configuration. After all the modules are constructed, a full validation test will be conducted on the entire system to ensure that there are no bugs or faults. After a clean bill of health, the project is deemed complete.

4 Implementation

The following sections indicate the steps taken to implement the design described in Section 3.2 of this document.

4.1 Drone

Drone Hardware

- Research microcontrollers and modules to determine which should be on the drone.
- Develop a pin assignment for the MCU.
- Create a wiring schematic based on the pin assignment.
    - Develop Bill Of Materials.
- Build a prototype on breadboards to verify the wiring diagram.
    - Verify data connections and power distribution.
- Build a prototype drone using protoboards based on the wiring diagram.
    - Verify connections on prototype.
- Design PCB to replace protoboard (Incomplete).

Drone Software

- Interpret all the accelerometer, gyroscope and magnetometer raw data from the sensors.
- Convert the interpreted raw data into orientation angles for interpretation of the drone's state.
- Show communication between the Feather and ESP8266.
- Establish PWM communication to control the motors.
- Send the converted data through ESP8266 to the ground station.
- Develop a PID algorithm to process the data and user input and control the motors on the drone.

4.2 Ground Control

Main program written in C:

- Create the main C program that can make two TCP connections (test station and drone)
    - Test this by writing locally-running c programs to simulate the devices
- Convert the working connection code into generic functions (where applicable) and put these in *connection.h* and *connection.c*
    - Test again to make sure nothing was broken
- Expand the program to support three connections: UI, drone, test station.
    - Test to make sure the three connections can be connected and communicated with all in the same main program execution
- Transition to using the C function select() as the source for determining which file descriptors are ready to be read from (stdin, ui, drone, test station)
    - Test to make sure all of the mentioned data sources can be managed simultaneously using select()
- Investigate packet format options

- Implement the ability to pack data to send and upack received data based on a defined packet format (see Fig. 3.2.2.1.2)
- Create generic functions to handle data packets and put them into the files *commands.c* and *commands.h*
    - Test to validate the packets are correctly created and parsed
- Determine the base user-input commands to be supported by the main program (see Fig. 3.2.2.1.1).
- Modularize the main program to take input from the user (via stdin or the UI), then support the processing of each command
    - As needed, create new message types in *commands.h* to support commands
    - Test to make sure the all user input is taken into account and all supported commands are recognized
- Modularize the main program to receive data from devices, then process the data based on the indicated message type
- Implement logging via the "log" command. Put all logging functionality into the files *logger.c* and *logger.h*
    - Test to make sure logging is initiated correctly, handled correctly during the logging session, and terminates correctly.
- Run a full test via local simulations of the ui, drone, and test station, then make changes as bugs are found. Repeat this step until satisfied with the results.
- Demo to the team for criticisms, then make adjustments as necessary.
- Demo to the client/advisor, then make adjustments as necessary.

UI written in JavaScript:

- Design the main dashboard of the program
    - Attitude Indicator for drone roll and pitch
    - Motor thrust values
    - Test station rotation value
- Create a communication server with nodejs, that communicates with the C server via TCP packets, and with the GUI via websockets
    - Automatically opens GUI on start
    - Automatically connects with running C server on start
- Add form to connect to new devices
    - Test with invalide/already connected to IP addresses to make sure error reporting works
    - Converts data into command, and is passed along the communication server to the C server.
- Design UI to view and open log files that have been generated by the C server
- Implement "logging", which just sends the logging start/stop command to the C server

4.3 Test Station

- Began by furthering the design toward a working solution
- Create a CAD design for a prototype the platforms, base station
- Integrate UART communication with ground control
- Developed sensor capture code to record rotational data.
- Test the prototype and adjust software for data recording as needed

4.4 Implementation Not Finished

- The PCB for Prototype 2 was not done in time. Prototype aims to be much lighter and have a smaller profile that accommodates the test station's platforms.
- The ground control program is created for WiFi communication with the test station. Either the test station needs to support WiFi communication of its data, or the ground control needs to be updated to support serial communication with the test station.
- The orientation data calculated on the drone is not fully sent to the ground station.
- The drone's PID algorithm is written, but the tuning of the algorithm still needs to happen.
- GUI platform without the need for NPM (Node Package Manager).

5 Testing Process and Results

5.1 Testing Process

This section describes tests that were/are needed for successful completion of the project and its subcomponents. This section focuses on major parts of the project that needed testing. Therefore, it leaves out the minor components.

- Ground Control - Full test of the C-written code
    - Without fully testing the main portion of ground control (written in C), the following cannot successfully happen:
        - Successful integration with the JavaScript UI
        - Communications with the drone and test station
    - This was tested by creating programs to run locally that simulate a UI, drone, and test station. These were used to replicate the functionality of the corresponding device
    - See Section 5.2 of this document for results.
- Ground Control - Full test of Javascript code
    - Controls many modes of communication, successful implementation means the GUI is able to interface well with the C server and do nearly everything that the C server can do
    - Tested by running communication tests, e.g. Sending the command "connect -d 192.168.1.1" and making sure the C server connects with the drone at ip address 192.168.1.1.
    - See Section 5.2 of this document for results.

- Test Station - Capturing Sensor Data, Measuring Physical Dimensions
    - In order to successfully do PID tuning on for the drone, the test station's sensor data needs to be captured and interpreted correctly
        - Testing of the physical capacity of the test station also allows for any inefficiencies or improper tolerances to be discovered, reducing error in rotational calculations
    - To test this, the team used an Arduino Nano to capture the 10-bit analog signal and display it via the Arduino IDE serial plotter
    - See Section 5.2 of this document for results.

- Test Station - Structure and Support
    - The 3D printed structure of the test station needs to be able to:
        - Spin with minimal friction
        - Hold the sensor used for measurements
        - Support and secure the drone
    - This was tested by having multiple team members analyze and discuss the printed design
    - See Section 5.2 of this document for results.

- Drone Hardware - First Prototype
    - This is the first built prototype of the drone. Testing of this allows the team to detect wiring and hardware issues that need to be addressed.
    - This was tested by attempting to power it, making sure all components are correctly powered, and running code
    - See Section 5.2 of this document for results.
- Drone Software - Sensor Data
    - Testing this section is important, because it determines the correctness of the information displayed by the ground station, and the correctness of the calculations done by PID. The sensor orientation data calculations need to be tested to see if the IMU is detecting the correct position of the drone.
    - This was tested by setting the microcontroller at different positions, and compared with the angular position returned by the drone.
    - See Section 5.2 of this document for results.

5.2 Results

This section describes and analyzes results obtained from testing indicated in Section 5.1 of this document.

- Ground Control - Full test of the C-written code
    - Logging headers need to be updated to correctly match expected MATLAB format
    - The timing of the function waittoread() from *ground.c* takes longer than desired (multiple milliseconds)
    - All else is acceptable

- Ground Control - Full test of Javascript code
    - Result is acceptable
    - Drone and testing station connects successfully.

- Test Station - Capturing Sensor Data, Measuring Physical Dimensions
    - The data has some inaccurate spikes. We believe this will be fixed by having the sensor secured on the completed test station. Otherwise, it is accurate for measuring rotational data and analyzing movements of the drone.
    - Dimensions and tolerances of the physical test station were measured with calipers and any inefficiencies or bad sectors of the printed models were sanded down to allow for a smooth rotation of each test station platform.

- Test Station - Structure and Support
    - The shafts that holds the magnet is too long
    - The roll/pitch rotational platform has too much friction
    - Untested: the support and securement of the drone

- Drone Hardware - First Prototype
    - The team forgot to add the Schottky Diodes
    - Drone did not fly because it was too heavy
    - A couple motors were wired incorrectly
    - Did not have sufficient batteries for the design
    - We ran out of time in the semester for creating a second prototype.

- Drone Software - Sensor Data
    - The angular location determined by the sensors and algorithms was within 1 degree of the true orientation of the microcontroller. The data also had a response time of about 10ms based on the change of the microcontroller orientation.

6 Closing Material

6.1 Conclusion

After a slow start to the first semester of design, the team had a high workload coming into this semester. We started out by getting tasks created, divided, and organized. The big milestone of the project was getting to a point of PID tuning the drone. This requires a working prototype of the drone, a functional test station, and a ground control that can log measurements. Each sub team was responsible for their respective tasks while working to reach this goal. In the end, the team ended with a test station that can successfully measure rotational data but missing the link to ground control. The team has a first prototype of a drone, but it is too tall and heavy to maintain flight. The drone software is at the point of being ready to test the PID algorithm, but that is blocked by the physical drone's limitations. The ground control C server and command-line interface are complete, but the main loop's timing is a little slow. The ground control UI is mostly complete, but not fully tested. The biggest challenge was working around the University's COVID restrictions because we did not have a dedicated lab for use at nights and on the weekends. Overall, the team faced various challenges and learning experiences throughout the project. We hope that the work done can be useful for a future MicroCART team seeking to perfect it.

6.2 References

[1] "Start [Silverware Wiki]". Sirdomsen.Diskstation.Me, 2020, http://sirdomsen.diskstation.me/dokuwiki/doku.php?id=start. (Accessed 15 Nov 2020).

[2] Bitcraze. "Crazyflie 2.1." https://store.bitcraze.io/products/crazyflie-2-1. (Accessed 28 Aug 2020).

[3] NewBeeDrone. "NewBeeDrone Store". https://newbeedrone.com/products/colored-cockroach-super-durable-frame (Accessed 24 April 2021)

7 Appendices

7.1 Appendix 1 - Operational manual

7.1.1 - Drone Operation Manual

This section will be a walkthrough of the setup and operation of thedrone.

7.1.1.1 - Prerequisites

Please reference embedded/software/doc/readme.md in our GitLab for instructions including required software and instructions for uploading code to the drone.

7.1.1.2 - Operation

To upload code to the drone double tap the reset button. If done correctly the LED on the drone should turn green. Once the code is uploaded the LED should turn off (unless controlled by the code). Then it will boot to the user code. During operation make sure that the drone is tethered to the ground or testing station.

7.1.2 - Test Station Operation Manual

This section will be a walkthrough of the setup and operation of the test station.

7.1.2.1 - Prerequisites

To program the Arduino, the user should have the Arduino IDE installed.

7.1.2.2 - Setup

Below is the setup required for the test station.

1. Setup the circuit provided and described in Section 3.2.3.1 of this document.
2. Ensure that the test station's Arduino Nano has a good USB connection to the ground station computer.
3. Select the desired platform for drone measurements.
4. Attach the drone to the platform
5. Insert the drone and platform combination into the top of the base station until the bottom of the platform reaches the sensor.
6. Insert the platform key into the hole on the side of the base station. This restricts movement of the platform.

7.1.2.3 - Operation

Operation of the test station is simple. Sending commands from the ground station moves the drone attached to the test station. The test station in turn sends positional data to the ground station application which can be interpreted by the operator.

7.1.3 - Ground Control Operation Manual

This section will be a walkthrough of the setup of the ground control program and how to use it.

7.1.3.1 - Prerequisites

The following should be installed and usable on the machine:

- Git
- Make
- GCC
- Node

7.1.3.2 - Setup

1. Clone the team's Git repository: https://git.ece.iastate.edu/sd/sdmay21-27
2. Navigate to the folder: .\Ground_Station\ground-control-c\ground_main\
3. Run 'make' -- this should create 4 executable files:
    a. 'ground' - main program
    b. 'ui' - test program to simulate the UI (only used for testing purposes)
    c. 'drone' - test program to simulate the drone (only used for testing purposes)
    d. 'test_station' - test program to simulate the test station (only used for testing purposes)
4. Navigate to the folder: .\Ground_Station\ground_control_mid
5. Run 'npm install' to install package dependencies
6. Navigate to the folder: .\Ground_Station\ground-control-ui
7. Run 'npm install' to install package dependencies

7.1.3.3 - Operation

1. Navigate to the folder: .\Ground_Station\ground-control-c\ground_main\
2. In a terminal/command line, execute the file: 'ground'. You should now be running the main ground control program. The first thing to expect is a ">>>" as a user input prompt.
3. To see the supported commands, see Section 3.2.2 of this document, or enter "help" in the command-line interface
4. *(Optional/Testing) If not using a drone and/or test station for testing, also run the following executable files separately (in different terminals): 'drone', 'test_station'. These will act like devices to connect to on local IP address 127.0.0.1.*