# MicroCART

## Design Document

Team Number: 27

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

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

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

Revised: 11/15/20 - 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.

## Summary of Requirements

The requirements for this project were given to us by Dr. Phillip Jones who wants us to develop a Mini 4.5 x 4.5 in, programmable quadcopter. The drone should be simple to reprogram and can be controlled by radio frequency and wifi. The drone will have a ground testing station where the students can perform testing on it, record its rotational data, program it, and recharge its battery. Additionally, there will be a ground control program that consists of a C-written command-line interface program for communicating with the drone and test station over WiFi. The total cost per drone and test station needs to be less than $50 each. Here is a list of the parts we plan to use:

- Microcontroller (Adafruit Feather M4 Express )
- Accelerometer + Gyro (Adafruit LSM6DSOX + LIS3MDL FeatherWing)
- WiFi + Bluetooth Microcontroller (NodeMCU-32S ESP32)
- Four Propellers in 40, 60, & 75 mm
- Four Brush Motors at 15000KV
- Expansion connector pins
- Lithium Battery at 3.7V and 2.5 AH
- Straight forward programing for students
- Ground control station

## 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.
- RC Communication - RC Communication is not something that is taught in our classes at Iowa State. In order to control the drone through RF, this was something that we had to learn.
- 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 Software - 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.

# Table of Contents

## List of figures/tables

# 1 Introduction

## 1.1 Acknowledgement

We would like to express our greatest 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.

This project could not be possible without the team members of the project: Alex Bjerke, Theodore Davis, Alfonso Raymundo, Amith Kopparapu Venkata Boja, Russ Paulsen, Trent Woodhouse, Grayson Goss, Hannah Aisya Mohamad. We will continue to give our best to execute the project successfully.

## 1.2 Problem and Project Statement

The Electrical and Computer Engineering department wants to create a mini quadcopter drone that is suitable for lab experiments in CprE 488. This class is focused on teaching students about embedded system design. The drone must be at most 4.5 by 4.5 inches in size and must be easy for students to use their own C code to control the drone. In addition to designing this drone, there must be a ground control program for students to connect to, and communicate with the drone. There must also be a test station where the drone's movement can be observed about a particular axis.

In order to solve this problem, the team of eight students will break down into sub groups and work on specific components that make sense for their skill sets. There are four main sub groups to consider: test station design, ground control program, drone software, drone hardware. We will use open-source and previous drone projects to guide our understanding and execution of the project.

## 1.3 Operational Environment

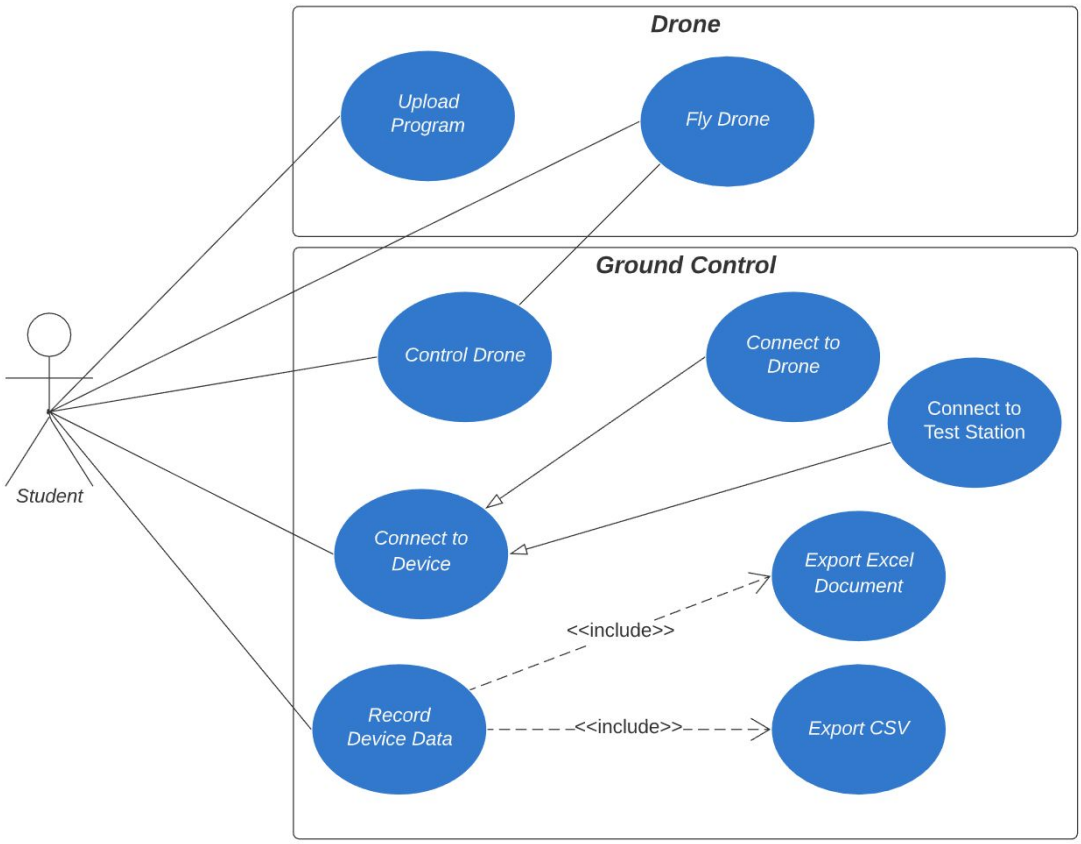
This drone will be used exclusively indoors, so it will deal with few to no environmental obstacles. The main operational area will be in the lab the class takes place in. This means the operational environment is very controlled and unchanging. The biggest obstacle for this operational area will be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

## 1.4 Requirements

The requirements for this project were given to us by Dr. Phillip Jones who wants us to develop a Mini 4.5 x 4.5 inch programmable drone. The drone should be easy to code and can be controlled by radio frequency and or wifi. The drone will have a ground control program where the students can send commands to the drone and receive data back from the drone over WiFi. The total cost per drone needs to be less than $50. Additionally, the drone should have about 10 minutes of flight time.

In addition to the drone, there should be a test station that the drone can attach to. This test station should be able to gather rotational data upon at least one axis. This data should be sent to the ground control application for students to analyze their code's performance on the drone.

Fig 1.4.1 Use Case Diagram




## 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)
- Payloads must not exceed 1/5th of the drone's total weight (flight limitation)
- Drone's size shall not exceed 4.5"x4.5" (client requirement)
- Drone must be able to communicate on several protocols (client requirement)
- Drone will not be subjected to harsh winds ( > 5 mi/hr)

### 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.

#### 1.7.1 Building a design

The first major section of our project is to build a working design of a drone. It should have the basic functionality of flying with the help of an RF controller. It should also use all the sensors attached to the drone to stabilize the drone or send data out in a readable form. For example, the accelerometer will be used to stabilize the drone, and a temperature sensor could be used to send the current condition's information to the user. Sensors like the magnetometer could be used by the students to read specific data and build algorithms to automate the flight of the drone. The student will be able to charge the battery and program the hardware of the drone using a USB port. This segment as a whole will ensure that students will focus on embedded programming aspects of the drone rather than the functioning of the 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 commanding the drone.

#### 1.7.3 Testing Station

The best alternative way of testing the drone's performance is to see its performance from a field's perspective. This testing station will record data of the drone in a field to determine the accuracy of

its calculations. It will also help students read data from the user end to understand the drone better.

### 1.7.4 Additional Improvements

After the drone was successfully built with an interactive GUI application, the next phase is to build a simple and secure library for the microcontroller programming of the drone, so it is easier for students to use to call certain functions.

# 2 Project Plan

## 2.1 Task Decomposition

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.

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).
- CAD
    - Drone chassis and wiring harness need to be constructed for this custom drone.

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

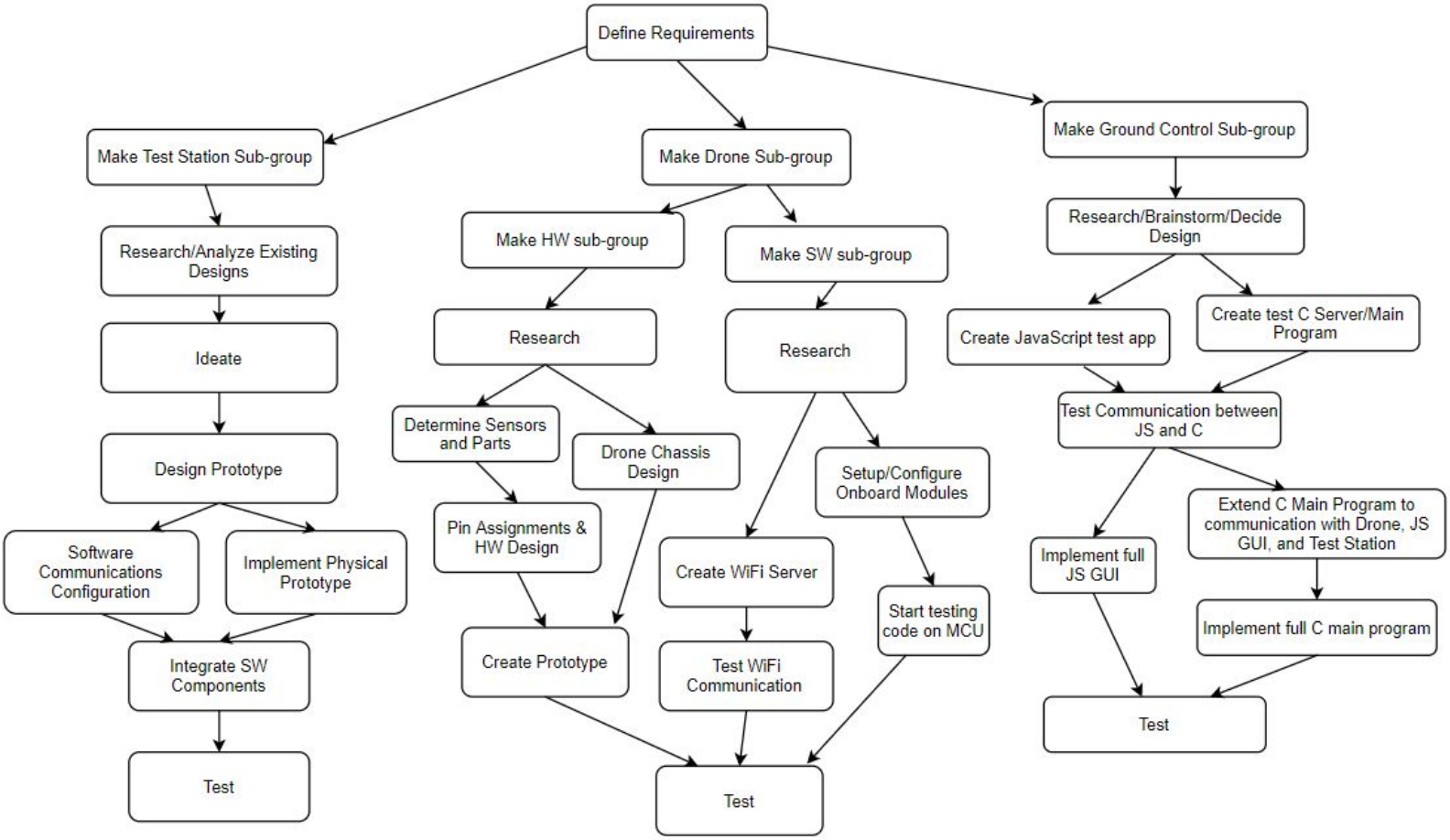
- There will need to be software embedded in the test station responsible for sending data to the ground control.

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. See section 3.3.2 for more details.

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 the figure.

Fig. 2.1.1 Task Decomposition Diagram



## 2.2 Risks And Risk Management/Mitigation

*Table 2.2.1 Risks and Risk Mitigation*

| Risk Factor | Risk Probability | Risk Mitigation |
|---|---|---|
| **Drone** | | |
| Connections may not be properly soldered. | 0.5 | Use practice boards or attend a workshop. |
| Erratic flight | 0.8 | Demo with open source code. |
| Unable to send data to the ground station. | 0.2 | |
| Drone is too heavy for take off. | 0.1 | |
| Battery may die while the drone is in flight. | 0.1 | charge the battery before every flight. |
| **Ground Station** | | |
| Does not connect to the drone. | 0.2 | |
| Displays sensor information incorrectly. | 0.2 | |
| Ground station loses power. | 0.02 | |
| Ground Station disconnects from the drone. | 0.1 | |
| **Testing Station** | | |
| Sensors do not record data. | 0.05 | |
| Is unable to send data to the ground station | 0.2 | |
| Drone is unable to move with negligible friction | 0.3 | |

## 2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The biggest milestones of our project are:

- Gain sufficient understanding in the functionality of a drone
- Building the smallest PCB possible ( i.e. A PCB that can fit on a 4.5" X 4.5" drone) while trying to keep the layer size of the PCB to 2 layers
    - Build protoboard to verify the schematics are good
    - Comprehensive schematic for PCB (includes pin assignments)
- Make all the actuators on the drone to communicate with the ground control program and also send data 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
    - With the help of ESP32, having all the necessary data sent to and from the MCU through WiFi
- Build an effective design prototype of the drone that incorporates all the sensors and motors while meeting the requirements. Minimize the size and weight of the drone by using special designed parts through CAD.
- Build an effective testing station that reads all the necessary information from the drone. It also determines the location of the drone in the field with 80% accuracy.
    - Having the ability to test every aspect of the drone
    - Constantly reads all data from the sensors and decides the performance and precision of the drone's output
    - Capability to communicate with the ground control station to deliver and receive necessary information
- 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.

We will evaluate the effectiveness of the contents on the drone and the testing station by comparing the expected results to the actual results of the sensors.

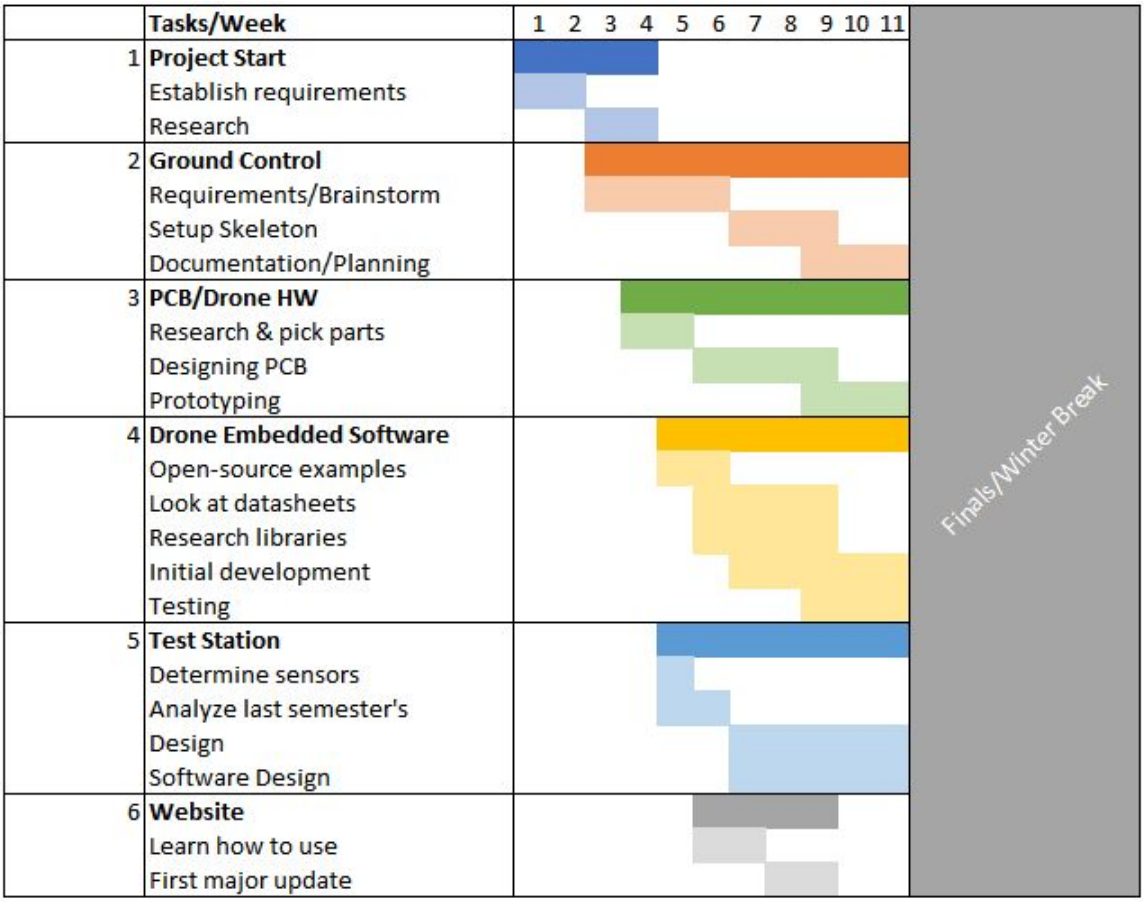
## 2.4 Project Timeline/Schedule

The figure below represents the first semester's timeline. This is using a start date of September 6, 2020. The first semester will be focused on the setup for a successful second semester. The plans for

the second semester are very tentative at the moment. The second semester will begin in a spot where prototyping has begun, and there is a clear path to the end of the project. Integration of different parts will take place, and there will be lots of testing needed.

At the end of the first semester, the project is roughly 2-4 weeks behind schedule.

Fig. 2.4.1 Task decomposition and schedule




## 2.5 Project Tracking Procedures

Our team will be using GitLab's Issues and Issue Board to track progress. This is a way to create tasks, assign them, and track their status. Additionally, we hold weekly meetings to sync with the team, with an additional weekly report of progress for our client, and a bi-weekly report for the class.

## 2.6 Personnel Effort Requirements

| Task | Time and Description |
|---|---|

| | |
|---|---|
| Prototyping | (2-4 weeks) This stage primarily consists of testing individual parts of the project then combining them together to form a complete prototype. |
| Embedded System Design (HW/SW) | (5-7 weeks) Embedded software and hardware will probably be one of the most intensive portions of the drone's construction since there are several modules that require programming for correct function. |
| Test Station Design and Construction | (1-2 Weeks) This stage is composed of the design behind the testing station for use in drone calibration. While we do already have existing files on hand, we have yet to determine the sensors to be used in this station. |
| Chassis Design Construction | (2-3 weeks) Due to this being a custom drone, a custom chassis will need to be modelled and simulated to determine weak points before production. Having this chassis be 3D printed will allow for a lower cost construction. This extended time accounts for potential failures in 3D printing |
| Ground Station Design | (4-6 weeks) Depending on the amount of open source code is available, the Ground Station (which will be used to communicate with the drone and the test station) will take between 4-6 weeks |
| PCB Design | (< 1 week) With a fully prototyped design established, the design of a PCB should be relatively easy considering this step consists of organizing parts and routing connections |

## 2.7 Other Resource Requirements

We have finalized all the components that we will be using to build the drones. Here is the list of all the finalized components that we will be dealing with:

- Microcontroller (Adafruit Feather M4 Express )
- Accelerometer + Gyro (Adafruit LSM6DSOX + LIS3MDL FeatherWing)
- WiFi + Bluetooth Microcontroller (NodeMCU-32S ESP32)
- RF receiver module
- RF controller
- Brush Motors at 15000KV
- Lithium Battery at 3.7V and 2.5 AH

- PCB manufacturer - Oshpark.com
- Propellers in 40, 60, & 75 mm

### 2.8 Financial Requirements

The final drone should be built for less than 50 dollars, and the test station should also be less than 50 dollars. The financial part of this project is being handled by Dr. Phillip Jones. When the team needs something ordered, we are to email the list of items to Dr. Jones, then if he approves, he will forward the email to ETG to have the items ordered.

# 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

The Crazyflie is a reprogrammable quadcopter of the size of which we are aiming for. The reason why our client hasn't chosen to 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 Design Thinking

This section covers our design and ideate phases of design thinking for this project.

### 3.2.1 Define

The problem at hand is that students' access to programming microcontroller-controlled devices are limited at Iowa State. There currently exists ComS 288, where students are able to program roombas to move around in an environment, but that is about the extent of it. So to solve this problem, we need to design and build a working drone, along with testing apparatus and an application to view incoming data from the drone. This will give students the hands-on experience working with microcontrollers.

### 3.2.2 Ideate

During our ideate phase, we clarified the devices and technologies we would be using, and designed diagrams to represent the different components of software and hardware. Additionally, we clarified what forms of communication we would be using between devices.

## 3.3 Proposed Design

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

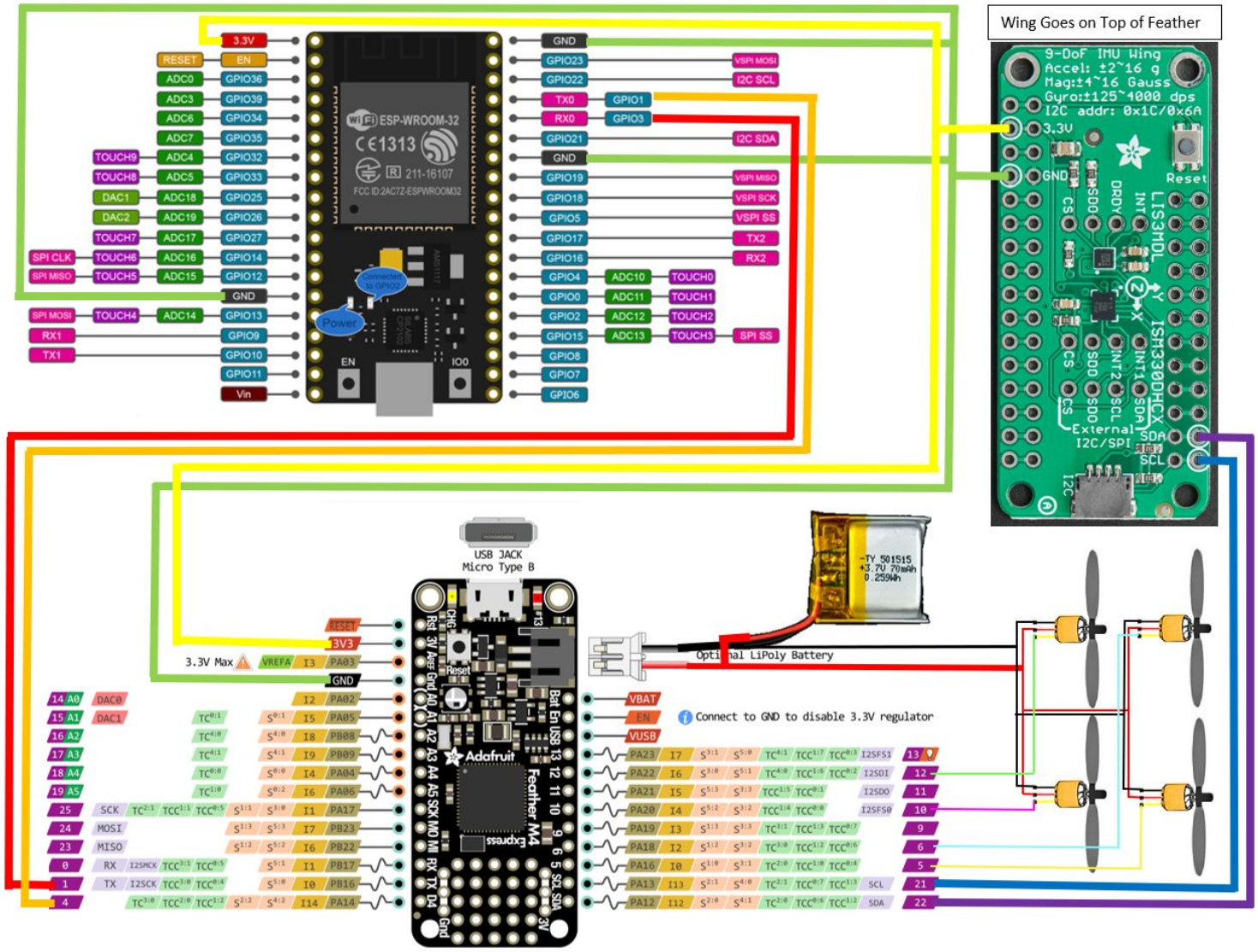
### 3.3.1 Drone

- Drone PCB Design:
    - The following components will be included in the pcb design:
        - Adafruit Feather M4 Express (ATSAMD51 Cortex M4)
        - Adafruit LSM6DSOX + LIS3MDL FeatherWing
        - NodeMCU-32S ESP32
    - We will be following [fig 3.3.1.1] for pin assignments.

Fig 3.3.1.1 Pin Assignments for ATSAMD51 and ESP32

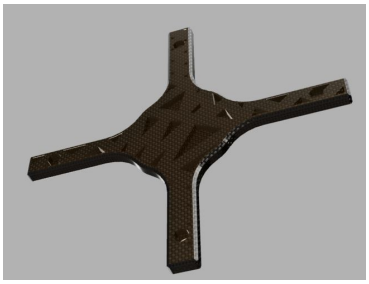
| Pin Number | Port | Periphals | Use |
|---|---|---|---|
| Pin 0 | PB17 | UART RX | Communicate with ESP32 |
| Pin 1 | PB16 | UART TX | Communicate with ESP32 |
| Pin 5 | PA16 | TC2 / PWM | Signal for Motor |
| Pin 6 | PA18 | TC3 / PWM | Signal for Motor |
| Pin 8 | PB03 | GPIO / LED | On board Neo Pixel |
| Pin 10 | PA20 | TC7 / PWM | Signal for Motor |
| Pin 12 | PA22 | TC4 / PWM | Signal for Motor |
| Pin 13 | PA23 | GPIO / LED | On board LED |
| Pin 21 | PA13 | I2C SDA | Communicate with Sensors |
| Pin 22 | PA12 | I2C SCL | Communicate with Sensors |
| Pin 19/A5 | PA06 | TC1 / PWM | Recieve pwm signal from RC |
| | | | |
| ESP 32 | | | |
| GPIO1 | TX0 | UART0 TX | Communicate with MCU |
| GPIO3 | RX0 | UART0 RX | Communicate with MCU |

Fig 3.3.1.2 Wiring Diagram



- Drone Chassis:
    - Using FreeCad, we have been able to create a CAD design for the drone's chassis.

Fig 3.3.1.3 Drone Chassis- FreeCAD



- Drone Software:
    - ATSAMD51:
        - Receiving instructions from ground control:

          Reading the instructions sent by ground control regarding the sensor data needed to be read. These instructions are read from UART coming from the ESP32.

        - Reading Sensor Data:

          This segment of the code is responsible for reading raw data from the Berry IMU, which contains the gyroscope, accelerometer and magnetometer, through the I2C module. This segment also converts the raw data into a readable format.

        - Transferring of the data:

          The formatted data is sent in bytes through UART to the ESP32, which is forwarded over WiFi to ground control.

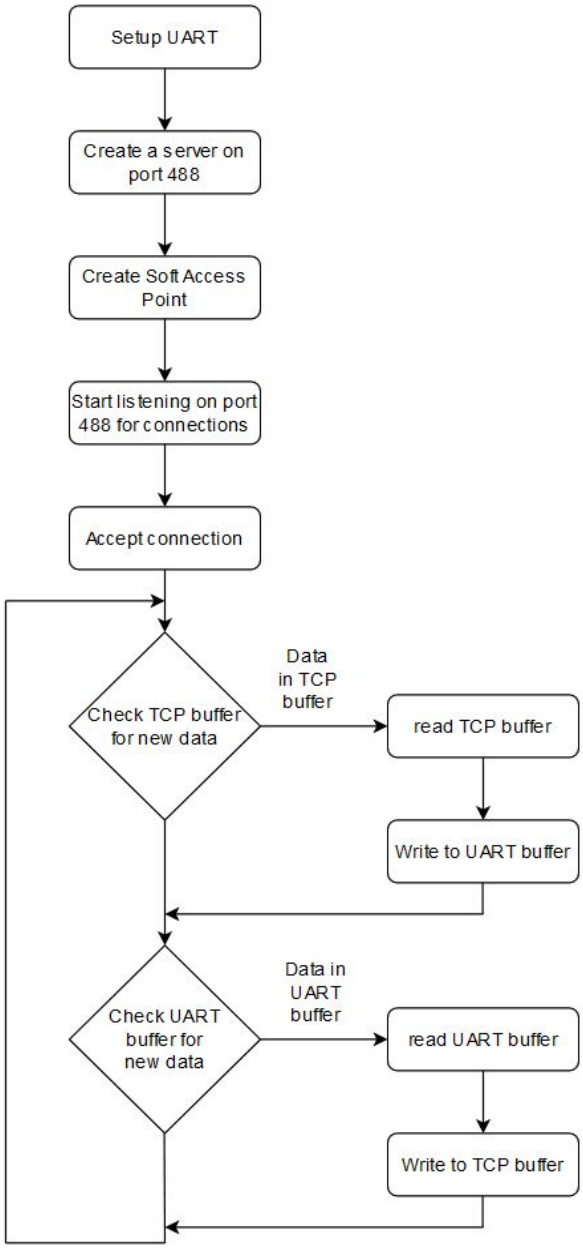
        - Reading and executing motion instructions:

          This segment is responsible for constantly reading and decoding the PWM signals from the RF controller. It is also responsible for using this using this information to run the propellers of the drone.

    - ESP32:

      The ESP32 software follows the general flow of [Fig. 3.3.1.3]

Fig. 3.3.1.3 ESP32 Software Flowchart



### 3.3.2 Ground Control

Below is the current description of the ground control:

- Main program written in C with the following behaviors:
    - Two threads responsible for communicating with the drone. One thread will be responsible for packing and sending data to the drone, then putting this data into a shared queue, called Q_TX_DRONE. The second thread will be responsible for receiving and unpacking data from the drone, then putting this data into a shared queue, called Q_RX_DRONE.
    - The main portion of the program will have two different modes, GUI mode and terminal mode
        - GUI Mode - Spawning a local JavaScript GUI that will then connect to the main C program via a local socket. There will be two more threads here for the JavaScript communication. One of these threads will be for receiving and unpacking data sent from the GUI. This data will then be put into a queue and treated as input to the main C program. The second thread will be responsible for taking data from Q_RX_DRONE, then packing and sending the response back to the GUI.
        - Terminal Mode - Taking input from stdin, from the terminal, then output to stdout. This will be a basic, command-line interface for sending commands to the drone, then seeing the drone's response (if any).
    - Test Station Communication - WiFi with similar methods used for communication with the drone. This socket will need to be bound to a different port number than the drone's socket.
- JavaScript GUI
    - When this is spawned from the C program, a websocket connection is made between this and the C program and allows data to pass back and forth.
    - The Javascript application is split into three pages:
        - Home - Students can get a basic overview of the application and/or the class, and other helpful information.
        - Connections - Students can connect to drones and testing stations, and can control the drones from this page.
        - Data - All data provided by the drone and testing station are sent to this application via websockets, and is presented in the form of graphs. Recording data and exporting it is also an option.

### 3.3.3 Test Station

- Design of the test station allows for analyzing different cardinal movements of the drone through the use of different drone orientations and fasteners for the drone
- The test station analyzes and helps calibration of the drone's function by providing data feedback to the users of the drone's performance at runtime.
- The test station design consists of a baseplate (to secure the overall apparatus to a table, floor, etc.), a sensor rig along the axis (to measure the various position and velocity related datas), and testing plate (to which the drone is attached in various orientations such that we can measure the lift from roll, pitch or yaw).

- Sensor being used on the test station is the MA3 Miniature Absolute Magnetic Shaft Encoder which measures rotation.
- The test station will utilize the Adafruit Feather M4 Express (ATSAMD51 Cortex M4) aswell to make parts orders easier and the chip has all needed functions that the test station needs.
- Our test station is a modification to last year's design team's design that has more ways to get information from the drone.

### 3.4 Technology Considerations

Strengths:

- Ability to refactor and change easily: New components can be added and existing components can be easily removed from the drone because of the control we have on every aspect of the drone.
- Simplicity in finding helpful resources: Every software and hardware component used in this project is commonly used for engineering projects, so if in the future, there were any unexpected issues, there are a lot of resources that can help solve the problem.

  The availability of resources will also give us an opportunity to get new ideas for improvising our product output.

- Simplified IMU: The major sensing system of the drone comes down to a simple board, helping us simplify the model of the drone and reduce weight and clumsiness of wires on the drone.
- User-friendly application: Ability to read important sensor data with the click of a button or a simple command, which would help people using the drone draw accurate and precise assumptions about the drone.

Weaknesses:

- Cost: Some components in the drone have components that are left unused, so money can be used for components that might not be necessary.
- Size simplicity of the parts: The parts used have potential to be reduced to meet just our requirements, but require special tools, and so there is more space on the drone needs.

Trade Offs:

- Even though the cost increases, it would give an opportunity to add additional components to the drone in the future. For example, a camera module can be added to the drone by connecting to the open microcontroller module.
- Even though size increases, we are only overestimating the drones requirements and not underestimating, which leads to malfunctioning drones. It might take more cost, energy and time to fix some of the new malfunctions.

## 3.5 Design Analysis

Given the current status of the project, we cannot say whether the full design proposed in section 3.3 works or does not work. The specific components chosen, and mentioned, in section 3.3 will function as expected. We know this because of the research that has been conducted (see section 3.1). For example, Figure 3.3.1.1 shows the chosen pin assignments. We know this will work because it is designed to compliment the provided peripherals on the MCU.

While the individual hardware components will work on their own, we will be responsible for their configurations and interfacing between components. Section 4.2 can be referenced to see the interfacing we will be responsible for. We do expect all components to be successfully integrated with each other in the end (again, this is based on the research mentioned in section 3.1).

When prototyping begins, we will be implementing the three sub-parts of the project in parallel (drone, test station, ground control program). The ground control can be tested as development occurs. To see the implementation tasks, see Section 2.1 and Figure 2.1.1. Regarding the drone, each interfacing component on the drone can be tested independently as the full prototype comes together (see Section 4 for more testing details). During testing and experimentation, each component will be assessed and deemed acceptable or not. If something does not function as expected, we will need to backtrack and find a different solution or modify the approach. When all interfacing components are tested, the full drone prototype will be built and tested. Once the full drone passes acceptance testing, the full functionality of ground control and the test station can also be tested.

## 3.6 Development Process

Our team is using the Agile development process. Given the nature of the project, we are easily 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 may have some less major contributions to different groups.

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

Since these subgroups are easily able to work in parallel, it is natural to follow the Agile process. Additionally, we are using Git Issues to track/assign tasks and enhance the Agile experience (see section 2.5 for more details).

## 3.7 Design Plan

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 Testing

## 4.1 Unit Testing

Individual components that will need to be tested independently are:

- The ground control program's communication via WiFi
- WiFi Communication through TCP or UDP sockets using the ESP32
- The sensors on the test station
- The drone's sensors (accelerometer, gyroscope, magnetometer)
- The drone's RF communication
- Powering the motors

## 4.2 Interface Testing

– After a software/hardware module successfully passes the 1st round of unit tests, we will move on to interface testing.

– We will perform the interfacing tests on the drone modules to verify that the Circuit board is working with the ESP 32, Feather M4 & Wing.

– After they are verified, we will run a program on the Feather M4 board to see if they were interfacing correctly with one another. These tests will confirm functionality of the Feather M4 and modules within the prototype circuit board of the quadcopter.

– Below is a list of hardware modules that communicate with each other. The Feather M4 is the main module. The Feather M4 will communicate with ESP 32, Wing, Motors, radio-frequency module, ground station, and test station sensor data. Each of these module interfaces listed below will need to be tested:

1) The Feather M4 will utilize the ESP 32 to communicate with ground control over WiFi.
2) The Feather M4 will receive the sensor data on the test station.
3) The Feather M4 will control/tell the wings to share the data/info it gets from its Accelerometer+Gyroscope sensors with the Feather M4 for its own use.
4) The Feather M4 will control the speed that the motors turn using the supply voltage from the battery.

5) The Feather M4 will work together with the radio-frequency module to take the input from the students and change the output to the motors

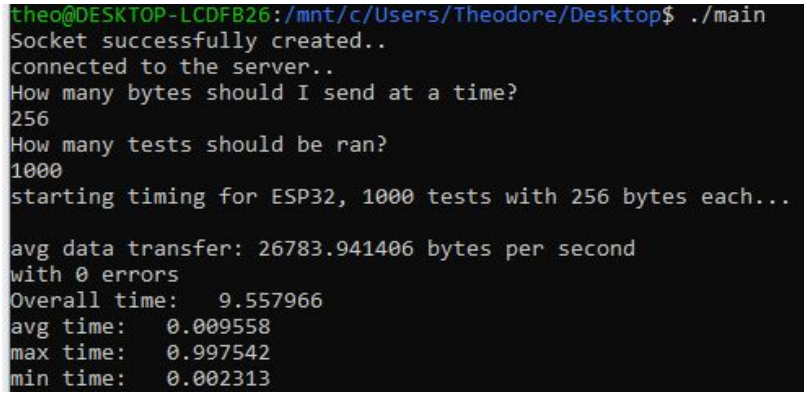
### 4.3 Acceptance Testing

We will be conducting the acceptance test as the last procedure of evaluation to see whether our development on our drone's system will meet our client's requirements and expectations. We will be evaluating our drones with our client and advisor, Dr. Phillip Jones.

### 4.4 Results

One part of our project we've been able to implement is sending and receiving TCP packets to the ESP32 through WiFi. We were able to successfully send and receive a number of bytes and were able to time the round trip time of the data, see [Fig. 4.4.1].

Figure 4.4.1, Timing turnaround time (in seconds) of the ESP32




With these timings we learned that echoing the bytes received over wifi with UART is quite slow, whether that's due to the nature of UART or some hidden limitations of the current library is unknown at this time.

Some errors we ironed out with this aspect of our drone was the naive approach of sending all our data at once and trying to read all the data at once. Not only was this approach slower than making multiple calls but we would run into scenarios where the ESP32 ended up behind a cycle of our testing program which would complain about byte mismatches!

## 5 Implementation

The following sections indicate the main tasks that are to be done next semester. These will be driving the implementation plan.

### 5.1 Drone

- Design a prototype schematic for our drone's pcb
- Develop software for testing the drone components
- Show communication between the Feather and ESP32

- Revise/optimize current solution for ESP32 code
- Order all the necessary parts for the drone, so we can put a prototype together
- Minimize the PCB size after a working prototype is made
- Software components are to be configured as we need

### 5.2 Ground Control

- Create a main C program that can make a TCP/UDP connection to the test station and drone at the same time
- Create capability for websocket communications in C
- Create a JavaScript browser GUI that can make a connection to the main C program
- Design an effective UI design that users can easily navigate through
- Test data communications between all sockets created (test station, drone, GUI)

### 5.3 Test Station

- Begin by furthering the design toward a working solution
- Create a CAD design for a prototype
- Integrate WiFi communication with ground control
- Test prototype, then make adjustments as needed

# 6 Closing Material

### 6.1 Conclusion

In the first semester, our MicroCART team has been working progressively on the project by starting off with intensive team research in order to help us understand the fundamentals of quadcopters. The work we have done so far is design of the drone, specifically the PCB where we design a schematic of how the components connected to each other. Nevertheless, our subgroup teams have been working in their assigned areas. Our end goal for this project is to be able to design a drone that can be used by CprE 488 students for its learning purposes. We believe that having tasks can help us reach our goal. We have seen that having an assigned task to our members by our project manager is working so far. Additionally, we have created a weekly schedule where we have 2 working times (Wednesday and Thursday) that are dedicated to work on our project. We have a follow-up meeting with everyone after our advisor meeting with Dr. Jones on Monday, and we hold a big meeting on Sunday to check with everyone's progress. Since this team schedule has been successful so far, we will continue having scheduled meetings throughout the week next semester.

## 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).