Quadcopter Cameraman

Final Report

Team Number: SDMAY19-42
Client: Mir Aamid Ahbab (Descarga Latin Dance Club)
Advisers: Zhengdao Wang

Luke A. Rohl — Scribe, Drone Developer
Mir Aamid Ahbab— Chief Engineer, Hardware Engineer
Isaac Holtkamp — Software Developer, Software Tester
Nate Allen - Software Developer, Data Analyst, Software Tester
Alexander Nicklaus — Embedded System Developer, Real Time Systems Engineer
Team Email: sdmay19-42@iastate.edu
Team Website: https://sdmay19-42.sd.ece.iastate.edu
Revised: 04/30/2019

Table of Contents

List of Figures

List of Tables

List of Definitions

Quadcopter - A robotic unit that utilizes four motorized propellers to move in 3 dimensional space
Drone - Synonymous with ‘Quadcopter’
ESC - Electronic Speed Controller
Quadcopter Cameraman - The name of the project and the generalized name of the robot being developed
Module - An electronic hardware device such as GPS, Barometer, Accelerometer, ect. Installed to the Quadcopter platform
Target - In context related to the project, a target is a human being recognized by the Quadcopter and image recognition software. Typically, the target is one or both of the performing dancers
User - A person intended to interact with the Quadcopter Cameraman deliverables including the performing dancer(s), android app handlers, or quadcopter technicians
Track - Used as a verb in context with the project. Means to know one or both dancers’ location(s) within the frame of the Raspberry Pi camera on board the Quadcopter Cameraman
Frame - The Quadcopter Cameraman will record video of a dance performance. A frame refers to a still image which is one of many still images which compose the full video
GPS - Acronym for Global Positioning System
Barometer - A hardware module which calculates air pressure. This is used to determine the altitude at which the air pressure reading was taken
Accelerometer - A hardware module which calculates the module’s acceleration in the x,y, and z axis
On-platform - The physical quadcopter
Off-platform - Not the physical quadcopter
Onboard - synonymous with ‘On-platform’

Facial identification - The ability to identify the presence of a face in a picture frame
Facial recognition - The ability to identify the presence of a particular face with a certain amount of confidence rating.
Arm the drone - Allows the motors to turn
Disarm the drone - Disables the motors from turning
Lighthouse - Will check the surrounding area by turning in place

Executive Summary

Problem Statement

The problem that we are trying to solve is that the Descarga Latin Dance Club on campus is having difficulties recording themselves and other members during performances. The main issue is that a cameraman can be obtrusive on a dance floor and get in the way of the dancers themselves or other dancers that may or may not be on the floor at the same time. We wanted to provide a more dynamic approach to recording dancers. Most dances involve moving across the dance floor which makes static methods such as a stationary tripod poor alternatives. To solve this issue our client has hired us to design, build, and program an autonomous quadcopter. This quadcopter will be able to identify the target dancers and follow them at a preset distance.

Scope

The objective is to build a camera drone capable of maneuvering and keeping multiple people in frame. To minimize externalities, we have set the project in the context of a dance performance. The types of dances include swing, west coast, salsa, and bachata. Target tracking and following are our primary goals. Performance and stamina come secondary. The drone should be able to follow the lead dancer at all times. The second dancer should be in frame whenever possible. As for performance, the drone should react quickly enough to create a seamless and effective recording of the dance. The drone should be able to maintain flight for the extent of the dance: a maximum of 5 minutes.

1. Requirements Specification

1.1. Functional Requirements

Flight Control

The drone needs to be able to pitch forward and backwards, roll left and right, yaw clockwise and counterclockwise, increase and decrease throttle, and hover. It receives input in the form of a command or location object. A location object will have both a distance and an angle to the target. With this information the flight controller can cause the drone to physically move so that the target is in the ideal location within a camera frame.

Image Recognition and Tracking

The drone will need to track the dancers while they move. To accomplish this the camera will stream camera information to the onboard Pi. The Pi will process the images and determine the location of the retroreflective tape. Using this location, the drone will then use machine learning to determine the distance and angle to the target. It will send this data to the flight controller, thus allowing the flight controller to make the necessary changes.

Video Quality

The drone shall be able to record video at a minimum quality of 480p. The video footage acquired will have both dancers present in 80 percent of frames.

1.2. High-level Requirements

Flight Time

The flight time of the drone needs to be a minimum of 5 minutes.

Drone Flight Control

The drone needs to have stable control of its own flight.

Video Quality

The quality of the video needs to be of at least 480dp x 640dp.

1.3. Use-cases

User Configures Run Time Settings

The User Navigates the Android app to open a screen with buttons for configuring the drone's mask settings for the computer vision software. On this screen, after the process is initiated by the user, there are several buttons the user can use to calibrate the mask. The purpose of calibration is to tell computer vision what is and what is not the target.

User Arms/Disarms Drone

The user navigates the Android app to a screen for initiating flight. Here, the user can arm or disarm the drone. Arming the drone allows it to take flight. Disarming the drone restricts the drone from taking flight.

When the drone is armed, the flight controller thread should be started. When the drone is disarmed, flight controller thread is ended.

User Uses Drone to record Performance

If and only if the drone is armed, the user can tell the drone to begin the session. When this is communicated, the drone will ascend to the configured altitude and begin tracking the target. A video is recorded for the duration of that session.

User Retrieves Video Recorded During Flight

The user navigates the app to a screen to manage videos stored from flights. The video can be uploaded to a repository to be viewed from a personal computer, or the app itself.

1.4. Non-functional requirements

Responsiveness

The purpose of this drone is to track dancers and capture video of their performance. The drone's movement can range from a stationary, hovering position to sudden, quick movements depending on the dance. As the dancers move about, the drone will have to move to keep them in center frame. It is of high importance that we minimize the lag between dancers' movements and the drone's response.

Security

The project has no reliance on a database. There are no user accounts, passwords, and data to be stored. However, the Raspberry Pi will act as a server for the android app client device to connect to. This device will send commands to the Raspberry Pi which will control many aspects of the quadcopter's behavior. This behavior can range from autonomous protocol activation/deactivation, altitude adjustments, forward and backward movement, side to side movement, platform rotation, and turning off and on of motors. Thus it is of high importance that the Raspberry Pi can only be connected to be the intended device. All device communication should not be susceptible to Replay Attacks.

Reliability

Given the extent of control which the android application has over the Quadcopter, it is imperative that the Raspberry Pi maintain connection with the android application. Restrictions should be placed on the quadcopter's freedom of movement to keep it within range of the off-platform device which is connected. Furthermore, when connection is lost, the Raspberry Pi should halt autonomous protocol and attempt to re-establish connection with the off-platform device

The project's objective is to record a pair of dancers for the duration of the performance. It can be said without debate that the software's reliability for correctly identifying dancers throughout the performance is the most important requirement within the project. Without this, there is no product.

Useability

The setup and usage of the product must be simple enough for any user to complete all of the defined use cases with a minimal strain on the user's comfort.

2. System Design & Development

2.1. Design Plan

Build a quadcopter drone capable of autonomous flight that can track and follow dancers. First we built and tested the individual systems such as the frame, electric systems, and motor

systems. Then we incorporated the systems into another while testing for maintained individual capacity. We continued from there to integrate the drone systems into one another.

2.2. Design Objectives, System Constraints, Design Trade-offs

Design Objectives

System Constraints

- 5 minute or more flight time
- There are no obstacles that the drone has to maneuver
- The target is easily distinguishable from another non-target
- The floor is level
- In door flight only

Design Trade-offs

Benefits

The general problem is following a person with a camera. Our solution is to use an autonomous drone that uses image recognition to identify the individual(s), track where they are in relation to the drone, and adjust accordingly. A benefit of our proposed solution is that drone technology is well understood and documented so we will have plenty of sources to draw from throughout the different stages of the project. A drone can have equal or better response time than a human leading to smoother and more reactive video. Our solution allows the user to get dynamic shots without needing another person. Essentially they can move around a great deal more than they would with traditional solo filming methods such as a tripod shot (static, camera does not move) or a dolly shot (dynamic but limited and expensive). An autonomous drone offers the user a cost-effective solution to get shots they couldn't normally get without expensive, specialized equipment and trained staff; it's a cheap jack-of-all trades solution. For example, a crane rig for which is used for overhead shots cost around $4,700 on the low end without a camera. Stable panning shots require a dolly system or steady cam which runs around $2,400 and $1000 respectively.

Weaknesses

Using a drone does have drawbacks. Flight time will be limited to battery life, thus not making it an ideal solution for repetitive, long takes or long shoots.This can be mitigated with multiple battery packs. Implementing this solution will be difficult. Controlling a drone even in an indoor environment is complicated. There are many different physical factors that need to be considered and accounted for in code, on the platform, or both. For example, before flying the rotors need to be balanced as even a slight imbalance can cause additional vibrations throughout the platform which is bad for filming. An example of using code to compensate for physics is how drones bank and turn. To bank right the drone reduces power for the two props on the side it wants to bank towards. This reduces the downward thrust in two ways. First, there is that there is less overall lift, and second, the drone's orientation shifts so some of the

downward thrust is used to move horizontally so the drone loses altitude as it banks. Turning along the yaw or rotating in place has a less dramatic effect on altitude as it lowers power to two props diagonally opposite each other by having these props reduce their speed. This reduces the compensation they produce for another two props which causes the entire drone to rotate. Like when banking, the overall thrust will go down, but the drone does not suffer the additional effect of downward thrust lost due to horizontal thrust. Since we do not want the drone's altitude to fluctuate while maneuvering we have to increase overall thrust in both cases. But as we end the move the drone must smoothly decrease back to the original thrust to return to a static hover. The code will also have to compensate for the momentum built up during the bank or turn. In short getting the drone to move around in a responsive but smooth fashion will be a significant technical challenge. Keeping the drone stable for the purposes of filming will require a mix between coding nuanced commands and changes to the physical rig to dampen the motion on the camera.

Trade-offs

A drone is a jack of all trades solution. You can get any shot, but it's not specifically built for any one shot. You'll get a better overhead shot with a crane rig and a better pan with a dolly rig. However, you can get all of these shots with one rig without hiring additional staff. The current solution focuses exclusively on tracking shots, or rather tracking one object throughout the scene which is preferable for dancers but limits the scope of applicability for other fields. Since the job of tracking of the dancers has been automated, the user will save on not hiring a cameraman but lose out on the experience and flexibility a cameraman brings. There is also the (ethical) issue of automating a profession that people may grapple with.

Comparison of Similar Systems

Drones currently on the market either use GPS or recognition technology to achieve autonomous tracking. DJI happen to be leaders in this field with products like their Mavic series, Phantom series and Spark series [4]. The Spark Drones can use either a smartphone or a hand gesture to set operating modes. The Phantom 4's active track feature is even able to track nonhuman moving objects like cars or trains. The drones can use a combination of gps and active track to follow their targets as well. The biggest drawback to this is the cost of the drones. The cheapest in the line is the Spark at $400 while the Mavic and Phantom are $1500 for the base drone [2]. While the Spark is feasible, the gesture controls create a liability for dancers, as their movement may also trigger a different operating mode than intended. While the packages that these platforms offer are great, a custom built drone will give us much more control to meet our client's needs than a repurposed drone.

2.3. Architectural Diagram, Design Block Diagram

Hardware

Our project design uses an autonomous quadcopter equipped with target tracking software to follow dancers during a performance. Additionally, the quadcopter interfaces with the user through a phone app. Computations will be done on a Pi mounted on the drone which

will feed information to the Flight Controller. The Flight Controller is a component normally found on drones. It interprets commands from a wireless transceiver, which receives commands from the user/remote control, and translates them into a signal that is sent into the ESCs (Electronic Speed Controllers). The ESCs control the amount of power sent to the motors from the battery. Each motor has its own ESC, and each ESC has its own connection to the flight controller. Our design has four motors and four ESCs. By having separate connections, the flight controller can control each motor speed individually allowing the drone to execute complex maneuvers such as banks, turns, and flips. In between the battery and the ESCs is a power distribution board which acts as a node to all four ESCs. The drone will use a camera and the flight controller's on board accelerometer as sensors for flight control. **Figure 1** below shows a conceptual sketch with wiring.

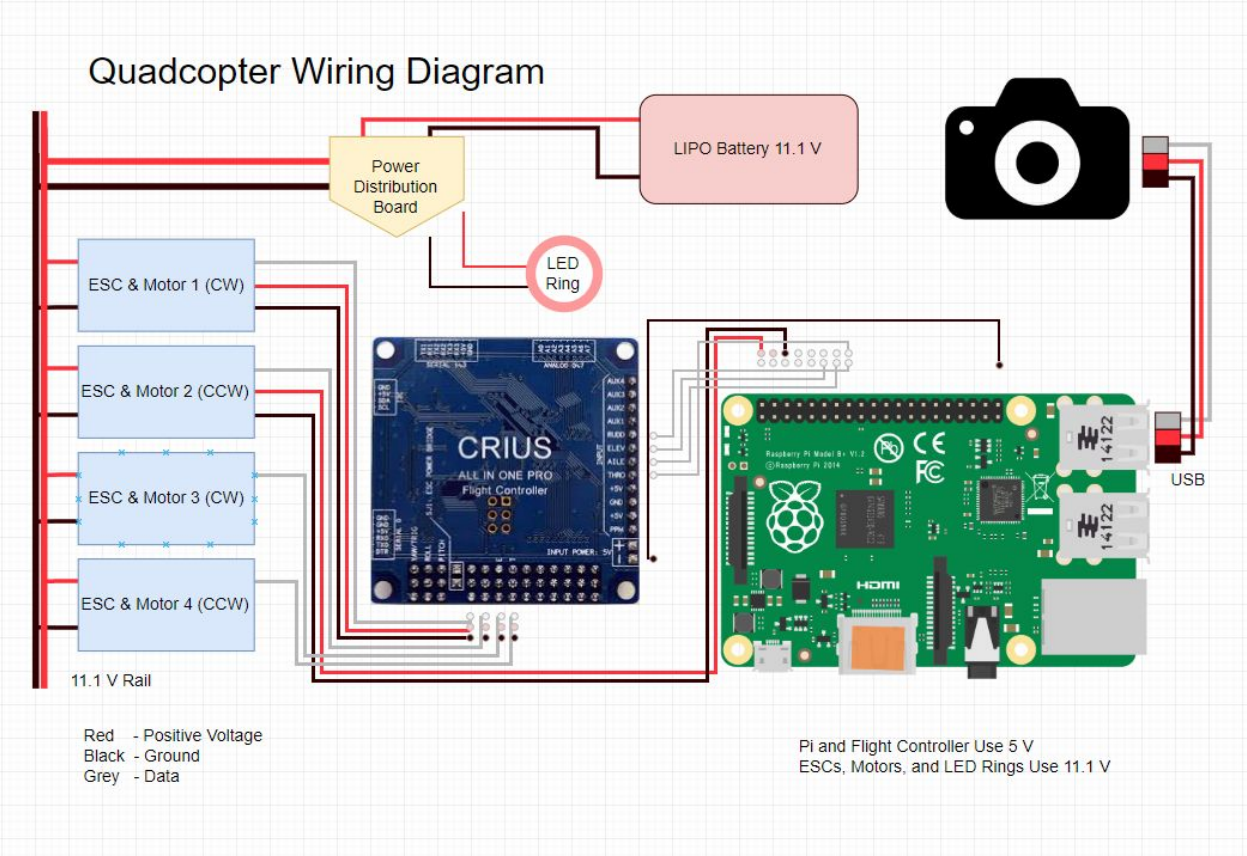


Figure 1: Hardware Wiring Diagram.

The above diagram shows how each component is linked to each other

Research and Alternatives

We considered using an off-the-shelf drone and programming it to track the dancers. However, it would pose a series of reverse engineering hurdles that made it a worse solution than building our own from scratch with open source components and software. First a drone that could do the work is already pretty expensive without the onboard AI, starting around $500 for the functionality we need. The software and possibly even the hardware would be proprietary and likely not have available documentation unless we could get source code from the company which is unlikely. Interfacing with the existing hardware might not be possible and ripping out the existing boards just to use the shell, power source, engines and camera is cost inefficient at best and still might not fit our needs.

Drone design is already a well understood and documented process with a plethora of both information resources from hobbyists and individual parts for purchase. Diving into the technical side of building a drone was more intensive and time consuming than using a pre-built drone, but the process of researching and understanding the physical system we will be useful. Going through the research also introduced us to concepts such as lift, torque, ducting the rotors to gain additional lift, how quads handle and maneuver, the difference between types of rotors, and how those differences affect performance. The value of this research can not be

understated. By researching and understanding the physical domain, we are better able to code autonomous controls, as we are aware of the implications the physics have rather than illicting those implications during coding which likely would have been a long and arduous task unlikely to produce a good final design. The price for our custom build is $393.72 which is cheaper than the off-the-shelf drone that would still require modifications. Additionally, we're able to customize the physical drone to the final goal rather than working around the physical design of a prebuilt. With all this in mind building a custom drone from scratch is the better solution for this project.

2.4. Description of Modules, Constraints, and Interfaces

Drone

For our custom build we used a 450mm frame with 10" rotors pitched at 4.5" and 920 kV motors. Research into drone parts indicates that a heavier drone is less likely to jitter during flight from interference and moves much smoother than a lighter one because of the weight. A larger drone has the space for more payload which our solution requires, as we will be adding on additional hardware. The 10" x 4.5" rotors and 920 kV motors provide the necessary lift to fly. The motors will generate 3200 grams of maximum thrust or 800 grams per motor. Research indicates that an ideal thrust to weight ratio for a quad is 2 : 1 meaning that the quad can hover at half throttle/power.

The Quadcopter Software required for the Drone will have four hardware systems: Command Systems, Motor Systems, Video Systems, and External Systems. Components are set into their systems based on functionality and compatibility with other required components. The drone will be classified as a 450mm quadcopter meaning the 4 motors are spaced evenly 225mm from the center of the drone.

Command systems will be the brains of the quadcopter and the central hub from which commands are given and processed. Initially this system included a pi and powersource. However, we were able to remove the separate power source, and run the pi using the main battery.

Motor Systems are the components that allow the drone to fly. The components involved are the battery, flight controller, power distribution board, electronic speed controllers, motors, and propellers. The flight controller will contain a gyroscope, accelerometer, and altimeter to feed position data to the Pi in the command system. The battery will provide 5 minutes of flight time.

Video Systems are what captures and records the video. As of now, we are utilizing a cheap Pi camera for testing purposes. The quality of the camera will be constrained by costs and weight. The quadcopter frame is also a part of this system, and is at a size of 450mm across.

External Systems are all components that are not mounted within the quadcopter. The only equipment in this system are chargers for the quadcopter lipo battery. The charger must be a balanced charger meaning it checks the voltages in each individual cell.

Software

The software must interface with all of the hardware modules, capture video, process images for target recognition, and compute an adaptable course of action through artificially intelligent algorithms. This will require a number of independent threads to monitor each of the separate modules. The AI will rely on the router module to handle all inter-thread communication.

The proposed software component diagram (See figure below) contains the main components for our drone software.

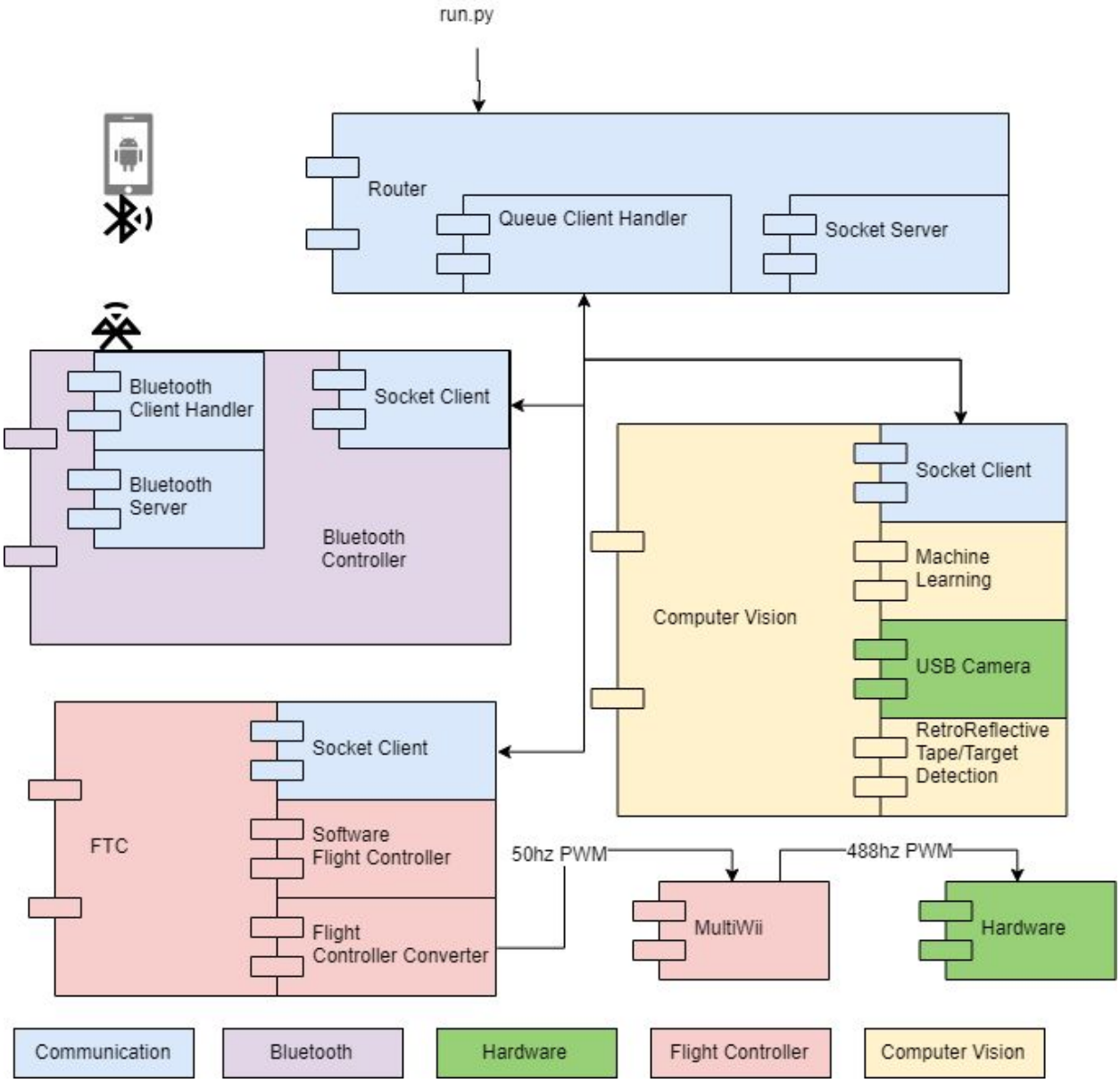


Figure 2 Software Component Diagram

This is an overview of the software components.

Router

Using a synchronous queue system the Router module allows for interprocess communication. When a communication is received, the to field is considered and either handled by the router itself or places the message in another threads queue. This process allows for a buffering like system. This is a scalable design because any number of threads may be joined to the router. This is also a module design because each module uses the same Client Socket class maintained by the Router allowing for interchangeability.

BTC - Bluetooth Controller

Software module that initializes the bluetooth server. After initialization the BTC waits for a bluetooth connection, after which it will connect to the router and allow the app to send commands to the rest of the system via the Router and vice versa.

My advice to future improvement would be to debug the reconnection system. If bluetooth connection is lost with the phone, then the whole flight must be restarted before connection can be established again. Additionally, security could be increases as there is no authentication other than to connect the phone must have been previously paired via bluetooth to the drone.

FTC - Flight Controller

Flight Controller module reacts from input received from either the android app or the computer vision module via the Router. When input has been received from the app the command is simply executed (i.e. Move Forward 5 causes the drone to move forward 5 ft). When the input has been received from the computer vision a small amount of logic needs to be performed to convert the given distance and angle into amount rotation or direction of movement.

Either way a command is sent to the MultiWii software. This command takes the shape of Pulse Width Modulation (PWM). The PWM is sent using a library called Pi-Blaster. Pi-Blaster generates a 50hz signal to the MultiWii Microprocessor. Once the signal has been received b the CRIUS AIO Pro board the commands are handled by the open source library MultiWii.

Computer Vision

Computer Vision module is the eyes for the drone. The target identification is reliant on 3 components: An LED Ring, A Camera, and Retroreflective Tape - not to be confused with Reflective Tape. The LED Ring is placed around the lens of the camera. This position allows the light from the ring to bounce off of the retroreflective tape and back to the camera. When the light is seen on the tape by the camera, the rest of this module's functionality can taken care of through the software. The image is processed into Hue Saturation Value (HSV) filter. This filter makes it easier to see reflective items. Here, a mask is applied to remove all parts of the image that are not the retroreflective tape.

Finally, the image is processed once more to find contours, these contours are bounding boxes for our targets. The contours are formatted into data columns and passed on to machine learning. Here, a cartesian decision tree is used to discern the target's distance from the drone. When the distance is known, the software can use that and the number of pixels the target is from the center of the frame to determine the target's angle to the drone. Now the Computer Vision module can pass the target's distance and angle on to the Flight Controller module, where the flight controller can decide what to do. Lastly, computer vision saves the image into a video file. At the end of the run, the video file can be uploaded to a repository to be viewed.

Drone Communication

The software module that will manage wireless communication with the Android App. This communication will handle incoming and outgoing commands with the android application. Once a command has been received it will be decrypted, and parsed for the relevant information. Example of commands that the drone can receive are: Select Target, Arm Drone, Disarm Drone and Upload New Target.

Android Activities

The Activities component act as the user interface. There are different activity pages for different uses; the developers activity page and the image alteration activity page. The activities are the users way to communicate to the drone and override the drones autonomous code in most situations.

Developers

The Developer section of the user application contains features that allow the user to control the drone and overrides most other autonomous code commands. Specifically, the developer section allows the user to issue commands to go: up/down, left/right, forwards/backwards, and rotate clockwise/counterclockwise. Additional commands include arming/disarming, land/takeoff, hover, and connect to the drone.

Mask Configuration

The Mask Configuration section of the user application contains features that allowed us to work/test our image recognition code on the Raspberry Pi. The application receives an image from the Pi and displays it on the screen. The user can then choose to change the contour size, and the lower and upper thresholds of the RGB values(Red, Green, Blue).

User Interface

Developer Interface

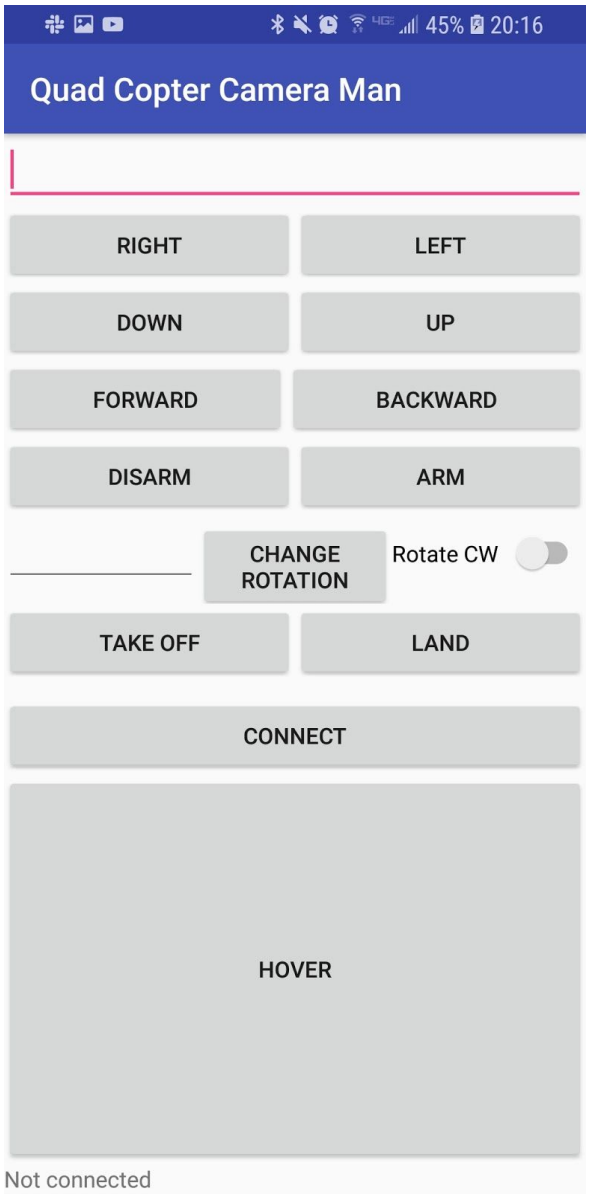
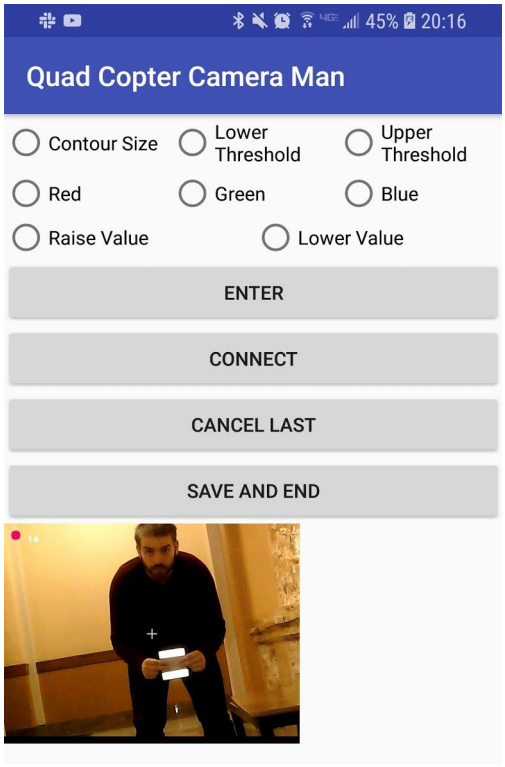
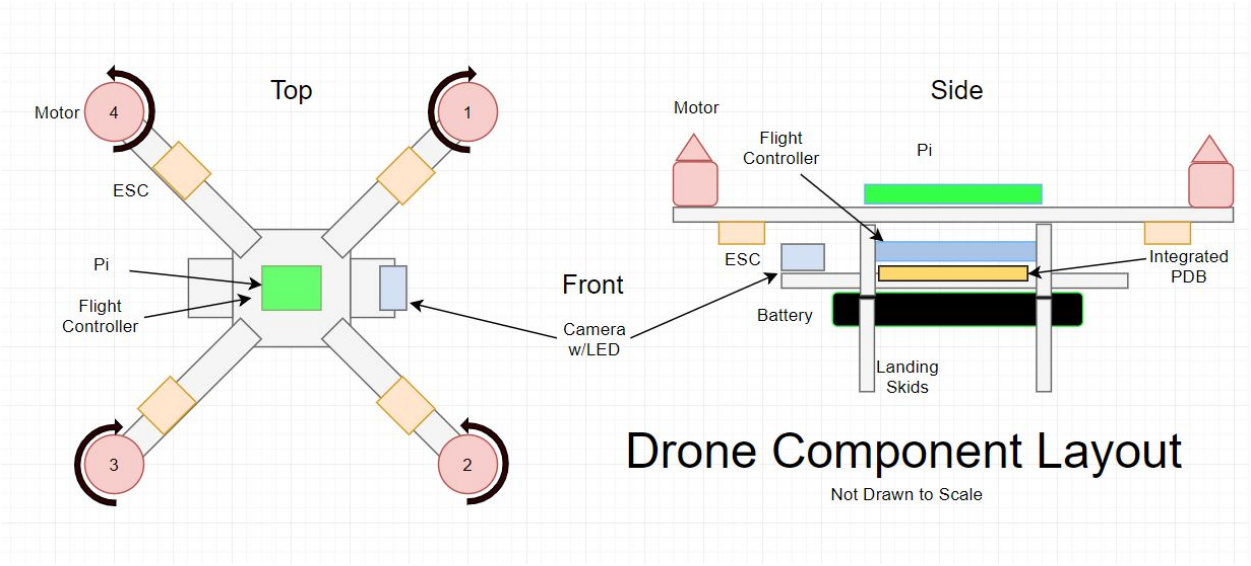


Image Alteration Interface



3. Implementation

3.1. Implementation Diagram



3.2. Technologies

Raspberry Pi 3 B+

The microprocessor used to receive commands from the android app, perform computer vision calculations, and control the flight of the drone. The majority of the drones logic takes place on this component, and it handles multithreaded communication between the drone's modules.

CRIUS AIO Pro Flight Controller

The microcontroller used to run the MultiWii software. The CRIUS AIO Pro Flight Controller received 50 Hz signals from the receiver (in this case the Raspberry Pi) and generates the post logic signals for the MultiWii software. The hardware itself has a gyro, a barometer, and a accelerometer onboard to provide feedback to the MultiWii software.

3.3. Software Used

OpenCV

Python3 library used to implement computer vision algorithm. This open source library allows us to use computer vision to determine target location on an image.

Sklearn & Pandas

Python3 libraries used to implement the machine learning algorithm. Specifically, we use a Cartesian Decision tree formulated through a statistical learning process. The algorithm produced helps us determine the distance of the target from the drone.

pyBluez

Python3 library used to handle bluetooth communication with the android app. This library allows us to create a bluetooth server on the Raspberry Pi and allows other bluetooth enabled devices to connect to it.

Pi-blaster

Python3 library used transmit PWM from Pi to MultiWii controller. This library allows for minute control over the PWM that are outputted from the Pi. By using this library we are able to send signals to the CRIUS AIO Pro board and have that input be given to the MultiWii software.

MultiWii

3rd party library used to control the flight controller output to the ESCs. The MultiWii library provides a host of benefits to the project. There are modes for flight stabilization, heading holding, and many other settings we weren't able to take advantage of due to our time constraint.

3.4. Rationale for Technology/Software Choices

Raspberry Pi 3 B+

Was chosen because it is small, lightweight option for on-drone processing to be performed that supports all of our software dependencies as well as integrate with the hardware for the drone.

CRIUS AIO Pro Flight Controller

Was chosen because it can run MultiiWii firmware, and for its ability to integrate with Pi-Blaster and the Electronic Speed Controllers.

OpenCV

Was chosen because it is a python library that performs computer vision tasks. A computer vision library was necessary to this project because without it we would have no way of determining where the target is. This provides important feedback about where the target is and how they are moving.

Sklearn & Pandas

Were chosen because they are a python library that performs machine learning tasks. The machine learning on this project is used to interpret image input and determine how far the target is from the drone.

pyBluez

Was chosen because it is a python library that allows the Pi to create a bluetooth server. This bluetooth server is also able to connect/pair with android devices, thus making this a good choice for bluetooth communication.

Pi-blaster

Was chosen because it can send PWM to the MultiWii Controller at the required frequency (50hz).

MultiWii

3rd party library used to control the flight controller output to the ESCs. The MultiWii library provides a host of benefits to the project. There are modes for flight stabilization, heading holding, and many other settings we weren't able to take advantage of due to our time constraint.

3.5. Applicable Standards and Best Practices

Bluetooth Standard - IEEE 802.15

This standard is applicable because we are using bluetooth technology and therefore need to make sure our project complies with the bluetooth standard set by IEEE.

Inter-board Communication - 50 Hz and 490 Hz PWM

This standard is applicable because it is the input signal used for the Crius AIOP Flight controller board and thus output signal of our Pi. The Flight Controller outputs at 490 Hz to the ESCs to control motor speed and maneuver the drone.

4. Testing, Validation, and Evaluation

4.1. Test Plan

Drone Flight

- Manual Flight
    - Manual Flight Tethered - partially successful began seeing cyclical oscillations in flight and drift. Drifting is to be expected in any drone build and can be easily remediated through trimming. Oscillations will require another solution
    - Manual Flight Untethered Trim - partially successful quadcopter's drift was able to be trimmed but cyclical oscillations still present shortly after takeoff
    - Manual Flight Untethered Angle Correction Set to LOW, MEDIUM, and HIGH - partially successful drone had a short stable flight however it rolled significantly right. Pilot was not able to correct in time and attempted to abort resulting in a crash landing and destruction of a prop halting further testing. Uncertain as to whether this was an issue with the accelerometer affecting controls or a trim issue.
- Target tracking of retro reflective tape successful in laboratory setting accurately identifying targets after manual masking
- Wireless control of the Pi
    - ssh to Pi successful: Commands sent to Pi via ssh successful: Pi generated signal sent to drone resulting in motor output matching the sent command.
    - App control successful - drone was tethered but able to spin up and respond to commands from the app appropriately
- We automated our software testing process by adding it as a flag while running the drone software "-test"

4.2. Unit Testing

- To run Unit test run "python3 run.py test"

4.3. Interface testing

- Interface testing was performed between the software components via unit tests.

- Interfacing between the software and hardware components was in the early stages of development and therefore were still being manually tested for correctness.

4.4. System Integration Testing

Connecting the Pi to the flight controller, and the flight controller to the ESCs required significant testing of each of the individual parts. We used a waveform generated to confirm each part acted as expected from the given input and we used an oscilloscope to confirm each part (one at a time) gave the correct output under given scenarios. When the sends and receives were confirmed working, we connected the parts (two at a time) and started running some of our code. When all the permutations of part pairings were verified, we finally connected all three parts and tested that ssh terminal could propagate through the Pi, to the flight controller, and finally out to the ESCs.

There were also several software components (ie ComputerVision, Flight Controller, Android App). These systems were integrated via a messaging network similar to a CAN bus implementation. A Router program would take all other components in as clients. Each client could write a message with a body and note the intended recipient. The messages were sent to Router and router then passed the messages on to the intended recipient. So testing the integration just meant confirming that everyone got the messages that were intended for them. We also implemented a mock client to impersonate components (ie receive messages intended for a software component without relying on that component) and to send messages for testing functionality.

4.5. Validation and Verification

Validation

We validated that the software was operating as expected through a set of unit tests.

Verification

We verified these controls manually while testing features that required these components to work together to accomplish.

4.6. Evaluation

Performance Metrics

Very few of the modules we created had a domain model that could be contained on a quantifiable scale. Our project required a more subjective look at each part and asking questions such as ‘In this video take, did the retro reflective tape detections work as well as we would like?’ It is difficult to build a test around such a module without knowing the expected bounds of each contour in every frame. Certainly, a test video could be constructed and paired with a datasheet of expected values, but risk of using subjective metrics was far outweighed by the cost of calculating test scenarios. So we chose to pursue the former.

Computer Vision

We decided it was fair to judge this based on the number of frames the target was in the frame vs. the number of frames the target was actually detected. This number should be 80% or higher. The angle the target can be from the camera would ideally be as high as 45 degrees. Lastly, the bounding boxes of the contours should have, at most, 2 pixels over or under bounded for the target. (ie 2 pixels too wide or 2 pixels too thin).

Retro Reflective Target Detection

We took video of a team member holding the target in front of the camera with light shining on the retro reflective tape. We used some built in libraries from openCV to draw boxes around our contours and output the video to be viewed by our ‘testers’. The original video was also kept (without any contours or alterations) to be fed into the computer vision module for subjective regression testing.

Evaluation Results

Retro Reflective Target Detection

With the videos taken and the contours bounded, we found that the target detection was highly consistent. The target was visible to the camera for angles up to and about 45 degrees. The number of frames the target was in frame and successfully detected was 100% (minus the frames where the angle of the target was unreasonably steep). And The contour bounds were always touching the actual target edges so long as the calibration was set correct.

5. Project and Risk Management

5.1. Roles and Responsibilities

Luke Rohl — Scribe, Drone Developer

Scribe

- Leads meetings
- Takes notes during the meetings
- Keeps track of due dates and tasks

Drone Developer

- Helped design drone’s software architecture
- Coded Router Module
- Coded Bluetooth Controller (BTC) Module
- Coded run.py
- Coded Flight Controller (FTC) Module

Mir Aamid Ahbab — Chief Engineer/Client, Hardware Engineer

Chief Engineer

- Manage budget and order items
- Communicate with Advisor
- Provide vision of project as Client

Hardware Engineer

- Determine physical components needed to meet requirements
- Assemble physical components in drone

Nate Allen — Software Developer, Data Analyst, Software Tester

Software Developer

- Computer Vision for drone
- Calibration setup program
- Camera interfacing and image processing

Data Analyst

- Research and Development for Machine Learning
- Collected and purposed data to train a distance classifier for computer vision

Software Tester

- Subjective verification of
    - Target detection successfulness
    - Target distance consistency and accuracy
    - Target angle consistency and accuracy

Isaac Holtkamp — Software Developer, Software Tester

Software Developer

- User Interface via android application
- Communication with drone
    - Computer vision messages to change image parameters
    - Drone communication to move Quadcopter

Software Tester

- Verification of Bluetooth communication

Alexander Nicklaus — Embedded System Developer, Technician

Technician

- Constructed drone and tackled problems that arose in the physical build
- Fabricated additional parts for the drone build

Embedded System Developer

- Worked on inter-board communication and signal standards
- Configured and worked with Multiwii code for drone flight

5.2. Task Decomposition

Luke Rohl

- **Drone Communications - Router**
    - Bluetooth Communication - BTC
        - Client
        - Server
    - Socket Communication
        - Client
        - Server
    - Router
        - Transferring communications to and from multithreaded modules
- **Drone Flight - FTC**
    - Software to Hardware Converter
    - Software Flight Logic
- **Init Script - Run.py**
    - Arg parser
    - Flags to turn modules off at startup

Aamid Ahbab

- **Component Documentation**
    - Component Price
    - Component parameters
        - Weight
        - Dimensions
        - Compatibility
- **Drone Design**
    - Frame Layout and Weight Distribution
- **Drone Flight**
    - Assembly and Maintenance
    - Safety of physical system and operators

Isaac Holtkamp

- Flight commands
    - Send commands to the quadcopter to move in a specific way
        - Roll/Pitch/Yaw
        - Up/Down
        - Rotate
- Computer Vision
    - Send commands to Pi to adjust picture settings
    - Display Image for users to see
    - String together messages from Pi to create image

Alexander Nicklaus

- **Multiwii**

        - Defining - a process in which the programmer configures third party code for their project by uncommenting defines within the code
        - Debugging
            - Arming/Disarming
            - Compiler issues with function prototypes
- **Interboard Setup**
    - Physical Wiring
    - Board Communication Standards
- **Drone Build**
    - Fabricated mounting jigs for Pi, camera, and sonar sensor
    - Assembly and Maintenance

Nate Allen

- **Drone Software**
    - Human object detection - deep learning (Deprecated)
    - Target tracking and predictive movement (Deprecated)
    - RetroReflective Target Detection
    - AI to determine the distance of target from drone - Statistical Learning
    - Interfacing with camera and image processing
- **Hardware**
    - Researched and selected necessary hardware for computer vision (minus the camera which was picked by Aamid)
    - Researched and developed a successful target shape that was both quickly processable and made it possible for camera to infer the distance
- **Drone communications**
    - Designed (but not personally implemented) a messaging system similar to CAN Bus design for communication between running processes.
    - Small hand in some networking functionality such as non-blocking receives by client
- **Communication protocols**
    - Sending Images to App in chunks to display during flight or calibration
    - Receiving incremental commands for values in calibration from App

5.3. Risk Management

1. Drone flight resulting in
    1.1. Bodily harm to person
    1.2. Irreparable damage to drone
    1.3. Repairable or replaceable damage to drone
2. Electrocution during R&D
    2.1. Minor
    2.2. Severe

Probability					
Severity		Unlikely	Rare	Likely	Frequent
	Severe	1.1, 2.2			
	Significant		1.2		
	Moderate				
	Minor	2.1		1.3	
	Minimal				

Table 1 Risks

This is an overview of risks, their maximum likelihoods, and maximum severity

Mitigations

1. We tethered the drone during initial flight testing until we were certain it safe enough to fly unrestrained.
2. Observed safe practices when handling and storing electrical components.

Notes on Propellers and Motors

Once the quadcopter was put together and ready for flight, we encountered multiple issues that increased the risk of injury and damage to the drone. The first issue was the propellers and motors. We first had to test the drone without the propellers on to see which way they spinned. We had to test the spin direction of the motors because they spun in different directions and if we attached the propellers to the wrong motor, they would generate “negative” lift and push the drone down. Additionally, if the propellers were on the wrong motors, there was a risk that the caps would shoot off of the drone.

Notes on Drone Flight

When the drone flies, it needs to trim its roll, pitch, and yaw to have stable flight.

5.4. Project Schedule

Gantt Chart

Figure 4 is a representation of the ideal timeline of our project. Integration of the MultiWii software and the ESCs was our largest roadblock this semester. Taking an unforeseen 3 weeks to complete. Our inexperience with Flight Controllers and ESCs might have been the largest part of this hurdle and requiring the whole team to complete this effort. After this hurdle was completed we were far enough behind in our schedule that only just allowed us to just begin the hardware testing phase.

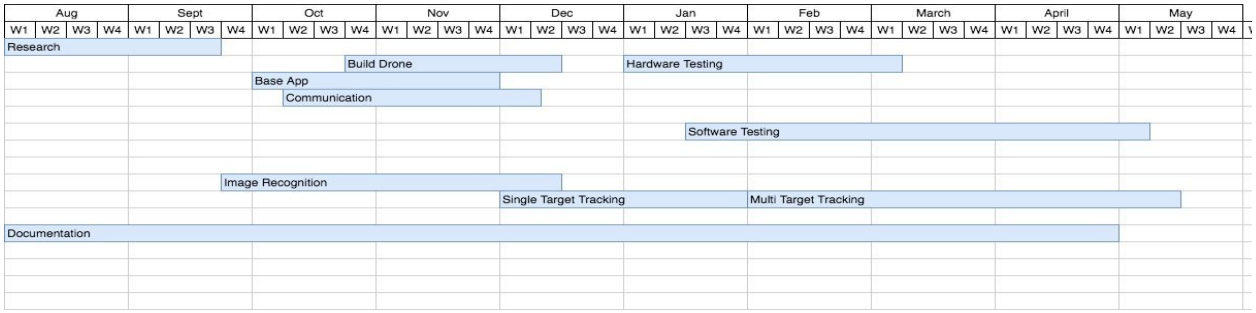


Figure 4 Project Timeline Estimation

Block-style scheduling table shows what tasks are planned and where they fall on a timeline for the project. The columns are broken down into months, and each month is broken down further into weeks. The blue boxes represent tasks.

The first semester primary goal was assembly of hardware, software modules, and the integration of the two. The secondary goal is to test the software modules and to calibrate the hardware components for our drone.

The second semester's primary goal was to test the hardware and software integration and to test our risk situations to ensure our product is safe for flight.

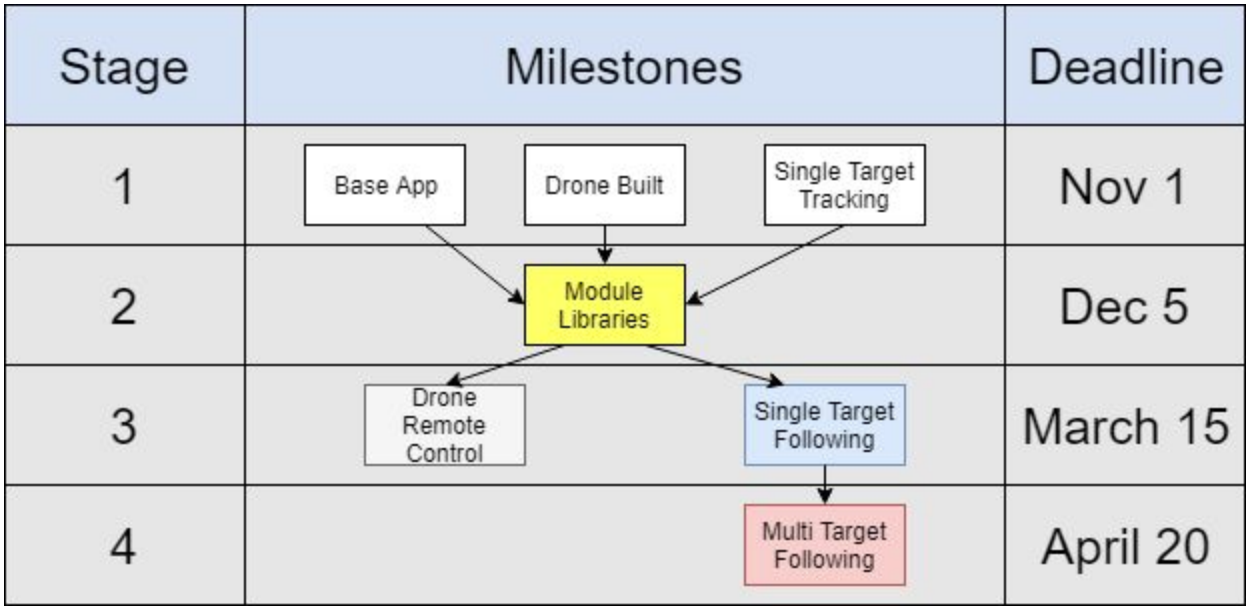


Figure 5 High Level Project Schedule

The table shows a high level schedule based on milestones' dependencies. Each colored rectangle is a milestone with an arrow showing the milestone which requires its completion before being started. The schedule is derived from that dependency graph.

Stage 1 - Completed

Stage one is the assembly phase. The foundation for all of the major focal points of are project are done at this stage. The drone, the app, and the technology for tracking targets have no dependencies on each other, so they can be done at the same time.

Stage 2 - Completed

The module libraries is the only milestone which will be completed at this time, since the next milestones are both dependant on this milestone. This stage will be all about the creation of the module libraries. These libraries will provide a layer of abstraction for the drone and the hardware. Since we have about 10 different hardware modules there will be a heavy workload fitting for an entire team to work on. Each of these hardware modules will need to initialize its respective hardware. It will also need to handle serialized communication with the hardware, and in some cases, a waveform generator will be needed to communicate with hardware devices.

Stage 3 - In Progress

This stage is almost complete, but is still missing those last few things. Computer vision was able to successfully identify a target, it's distance. The android app had been completed and the drone can be controlled (barely) by the app. This symbolizes round trip communication between the android app and the drone and allowed additionally trimming. Unfortunately, we were not able to finish trimming the drone's flight with the software before the end of the semester had arrived due to part damage.

Stage 4 - Incomplete

The final stage is only a small adjustment on the 3rd. By the end of the 3rd stage, we should be able to track and follow a single target. Also, most of our use cases will be finished in terms of app-drone communication. Now we only extend that functionality to follow multiple targets rather than a set single target.

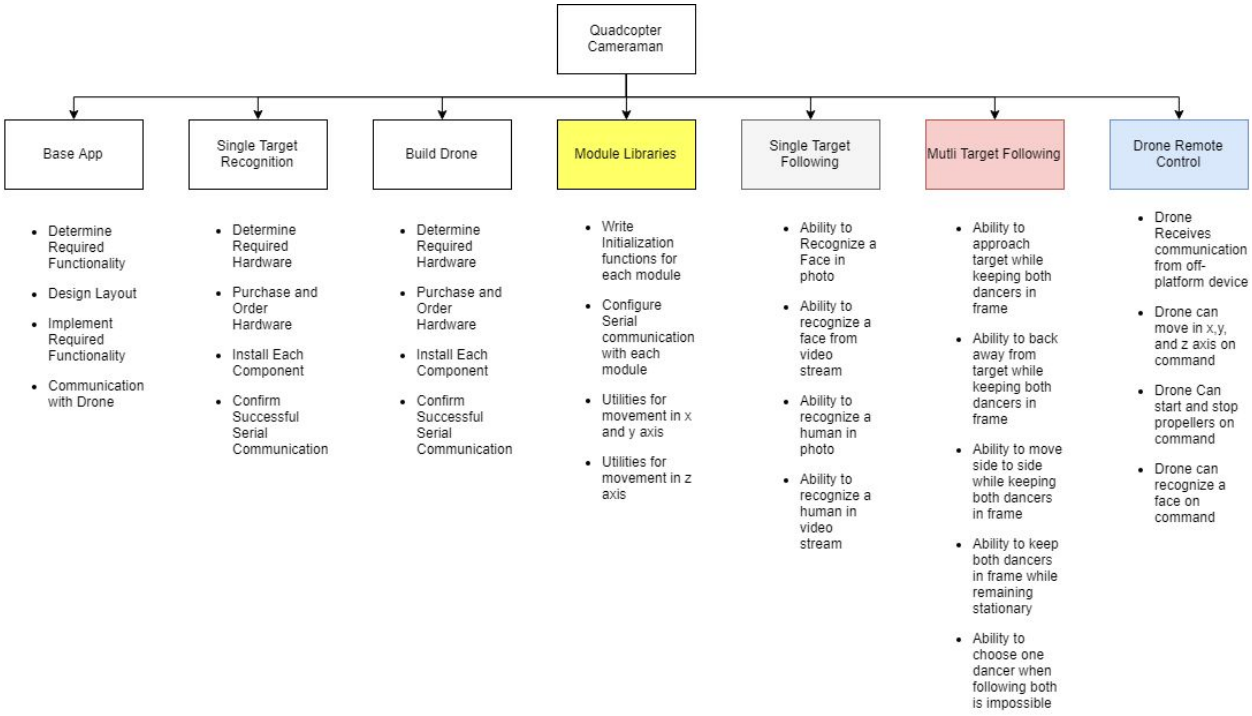


Figure 6 Work Breakdown Structure

The figure shows a tree-structured break down of the tasks required to be completed and deliver the product with all of its requirements by the end of the year. The rectangles are each of the milestones we identified earlier, the bullet points below detail the minor tasks within the corresponding milestone.

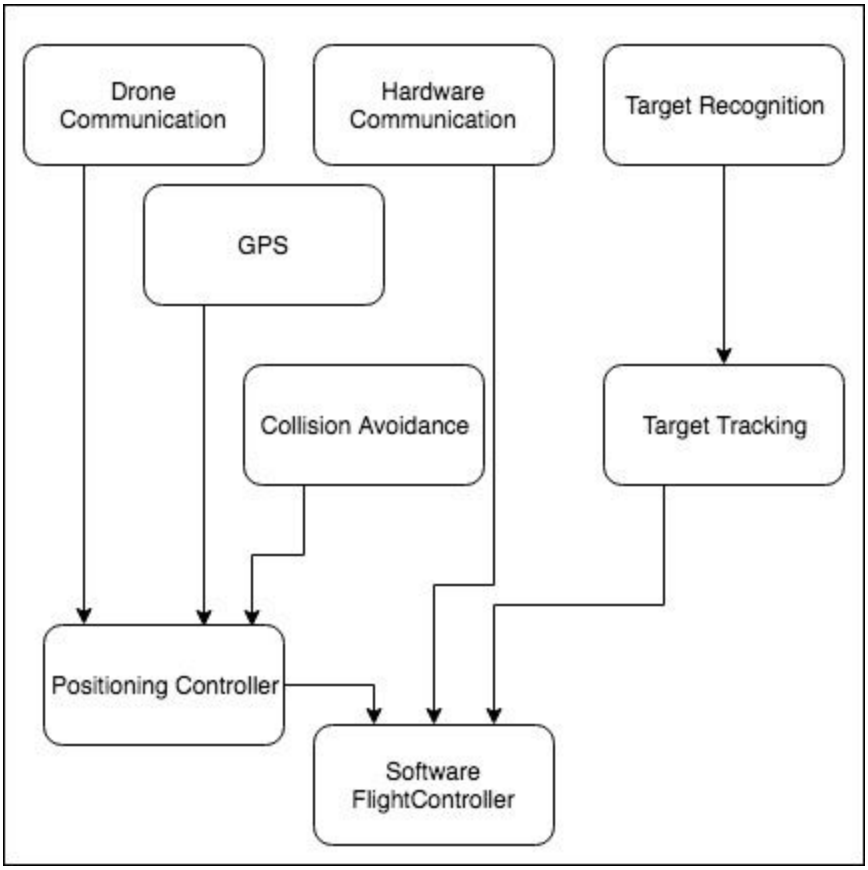


Figure 7 Software Component Dependency Graph

Is showing the dependency graph for the software components. The Software Flight Controller code is dependant upon the positioning Controller, Target Tracking and the Hardware communication. The Positioning controller is dependant on Drone communication and Collision Avoidance. The Target Tracking is dependant upon Target Recognition. Target Recognition, Drone Communication, Collision Avoidance and Hardware Communication are no dependant upon other software modules.

6. Conclusions

6.1. Closing remarks

Our goal for this project is to have an interesting and comprehensive way to record videos of dancers at dance competitions and performances. We would like to improve video recording and quality with a drone that would take out or reduce human error in recording. Our design includes multiple modes of recording and allow for a range of dances that can be recorded including swing, west coast, salsa, and bachata. We have the ability track dancers and follow seamlessly while recording the performance using a retroreflective tape implementation of target tracking.

The Quadcopter Cameraman team would like to kindly thank Iowa State University and the College of Electrical, Computer, and Software Engineering for promoting student professional experience and sanctioning this cross-disciplinary project. As students, the team

appreciates the university for prioritizing outstanding issues with the current small equipment checkout system from the Electronics and Technology Group (ETG). Project Quadcopter Cameraman's team would also like to thank ETG for their mentorship in developing the team's professional skills, for allocating human and financial resources, and for sharing their workspace with a handful of engineering students.

6.2. Future Work

There is a lot of future work that could take place on this project. Since this project was a strong balance between software and hardware, there is plenty of room for future work in both areas.

Software

On the software side of things there are always bugs to kill, security to increase, and efficiencies to officiate. The Android App itself could use more intuitive controls. Instead of typing in a particular value desired by the user the amount should automatically start at neutral and then be incremented up or down depending on the user input. The Drone software has a lot of room for improvements. The Bluetooth Controller (BTC) will only allow 1 client to connect. This isn't ideal and can be improved. The future work for the BTC could include starting the BTC again so that it waits for another connection. If it times out then it should land safely. The Drone Router could use several updates. Handling multithreaded anything isn't easy and the router is the same. There are bugs that need to be ironed out and much room for an increase in efficiency.

Hardware

Hardware is complete, but further upgrades could enhance the system offering greater flight stability, reliability, and potential new features. However all new upgrades are limited by funds available for future iterations.

The flight controller is the first area of improvement. A more advanced flight controller offers more reliability in its sensors. Advanced boards also have more features with a greater number and variety of sensors. Obtaining similar FCs to those of DJI will offer a greater advantage to both developers and end users due to their reliability and overall capabilities. Utilization of a new flight controller will also give developers the option to use alternate firmware for their boards. This has the potential to simplify coding for developers by using a friendlier firmware.

The camera we chose to use was simple and cheap, which made it great to develop with. However, the quality of the camera is not the best and future iterations of this project can definitely find a better camera to record with. A better camera will involve higher resolution and video quality. Decisions for future cameras will still be constrained by the weight of the device.

While the battery and motors on board are quite sufficient, upgrading the capabilities of each will allow more room to explore future functions for the drone involving heavier equipment like cameras. A battery that can operate at 14.8 V can supply more power to motors, thereby

increasing thrust. However, the motors will also need to be upgraded to be compatible with an increased voltage. This is a fairly costly upgrade and is also optional.

The current hardware that the drone utilizes is sufficient for the task at hand. Components are compatible and allows the drone to fly. However, upgrading of parts can lead to greater performance or better ease of use.

6.3. References

Batteries For UAV. [Online]. Available: http://dronesarefun.com/BatteriesForUAV.html. [Accessed: 27-Oct-2018].

"4 X GARTT ML 2212 920KV 230W Brushless Motor with Self Locking Adapter Quadcopter F450 Multirotor Drones-in Parts & Accessories from Toys & Hobbies on Aliexpress.com | Alibaba Group," *aliexpress.com*. [Online]. Available: https://www.aliexpress.com/item/4PCS-GARTT-ML-2212-920KV-230W-Brushless-Motor-For-DJI-Phantom-QuadCopter-F450X525-Multirotor-Drone/32561949556.html. [Accessed: 27-Oct-2018].

DJI Official. (2018). DJI - All Products. [online] Available: https://www.dji.com/products#drones-nav [Accessed 27 Oct. 2018].

F. Corrigan, "12 Best Follow Me Drones And Follow You Technology Reviewed," *DroneZon*, 06-Oct-2018. [Online]. Available: https://www.dronezon.com/drone-reviews/best-follow-me-gps-mode-drone-technology-reviewed/. [Accessed 27 Oct. 2018]

F. Corrigan, "How A Quadcopter Works With Propellers And Motors Explained," *DroneZon*, 06-Oct-2018. [Online]. Available:

https://www.dronezon.com/learn-about-drones-quadcopters/how-a-quadcopter-works-with-propellers-and-motors-direction-design-explained/. [Accessed: 27-Oct-2018].

F. Corrigan, “Drone Waypoint GPS Navigation Technology And Uses Explained,” *DroneZon*, 11-Sep-2018. [Online]. Available: https://www.dronezon.com/learn-about-drones-quadcopters/drone-waypoint-gps-navigation-technology-explained/. [Accessed: 27-Oct-2018].

Painless360, “How to select the right motor for your multi-rotor (all types - Tri, Quad, Hex etc.),” *YouTube*, 15-Sep-2013. [Online]. Available: https://www.youtube.com/watch?v=HSQGI6u2DIM. [Accessed: 27-Oct-2018].

“Multirotor Motor Guide,” *RotorDrone*, 30-Oct-2017. [Online]. Available: https://www.rotordronemag.com/guide-multirotor-motors/. [Accessed: 27-Oct-2018].

R. Riot, “Propellers,” *YouTube*, 18-Apr-2016. [Online]. Available: https://www.youtube.com/watch?v=TbAkcKPHqt8&feature=youtu.be. [Accessed: 27-Oct-2018].

“Understanding Kv Ratings,” *RotorDrone*, 27-Jan-2017. [Online]. Available: https://www.rotordronemag.com/understanding-kv-ratings/. [Accessed: 27-Oct-2018].

6.4. Team Information

Luke Rohl
Phone: 515-434-3099
Email: Luke.A.Rohl@gmail.com

Nate
Email: nkallen@iastate.edu

Alex

Phone: 949-677-8844
Email: nicklaus@iastate.edu

Isaac

Phone: 515-537-6445
Email: isaacholtkamp@gmail.com

Aamid

Phone: 309-453-5640
Email: aamid96@gmail.com