QuadCopter CameraMan

Design Document

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

Luke A. Rohl — Scribe, Drone Developer
Mir Ahmed — 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: October 26th 2018 / Version 2.0

Table of Contents

List of Figures

List of Definitions

Quadcopter - A robotic unit that utilizes four motorized propellers to move in 3 dimensional space

Drone - Synonymous with ‘Quadcopter’

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 - a scanning procedure performed by the drone, where it will scan the surrounding area by rotating in place

1. Introduction

1.1. Acknowledgement

The Quadcopter Cameraman team would like to kindly thank Iowa State University and the College of Electrical, Computer Engineering, 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.

1.2. Problem and Project Statement

Problem Statement

Among dancers being able to showcase their talents through videography is an effective tactic that many dancers employ to capture a wide audience. In order to obtain the best videos, multiple cameramen need to be trained to work with their dancers to capture the best footage. This becomes an issue for dancers that are just starting out and don't have the resources to hire a trained cameraman. Descarga Latin Dance Club would like a simplified way for dancers to capture their own footage.

Project Statement

A quadcopter is an ideal solution for capturing footage because of the greater mobility over a human, as well as allowing the dancers to not rely on other people for recordings. Not only that, but drones have a smaller area of visual impact, meaning that audiences can watch the performance without a cameraman blocking their view as is common in many festival demonstrations. The QuadCopter Cameraman team aims to design our own drone to function much like a human cameraman, so as to track dancers in their performance. Our goal for this project is to be able to develop a drone for the Descarga Latin Dance Club, an on campus student dance organization, for use in their performances.

1.3. Operational Environment

The operational environment of the quadcopter will be indoors and with no overhead obstructions, because this drone will be used indoor there should be no weather conditions such as rain or wind to cause harm to the drone. There should also be enough overhead space to allow the drone to operate at it normal height (3 meters off the ground). The user should also be aware that the drone will try to maintain a distance of 3 meters from them and should therefore be aware that they don't position the drone so that it will run into a wall.

1.4. Intended Users and Uses

Intended Users

The intended audience for this project will be dancers with little to knowledge of the inner works of the drone. Therefore it will be required that the drone be user friendly and easily manageable through the android app user interface.

Use Cases

- User turns drone and pi on. User then opens app, arms drone, selects a pre-existing target to track, selects take off thus allowing the drone to take off and start tracking selected target.
- User opens app, selects add new target, and then uploads a photos of the target that needs to be tracked.
- User opens app, and disarms the drone. The drone will then perform a graceful disarming and will slowly land.
- Drone loses bluetooth connection. It will then try to connect over Wifi. Failing this the drone will then attempt a safe emergency landing.
- Drone loses sight of dancers, it will look at previous data to predict the location on the dancers. If it finds the dancers it returns to normal operations. If it cannot find the dancers then it will lighthouse to find them, if it still cannot find them then it will emergency land.

1.6. Assumptions and Limitations

Assumptions

- The dance is less than 5 minutes
- The dance has no audience
- Paired dancers are always easily distinguishable from one another
- The dance floor is level
- The auditorium has no obstacles
- The walls will not be so close as to pose a hazard to the drone
- All people involved in the use case will be easily distinguishable from one another (no twins or look alikes that could confuse facial recognition)

Limitations

- The drone will need constant wireless communication to ensure safety. If communication is lost the drone will perform a landing.
- The drone will store video in onboard memory, limiting the length and resolution of the video
- The drone will have a limited power supply, limiting the length of its flight, and thus the length of the performance
- A certain amount of time will be required for the drone to analyze the given photos of a specific target. Therefore the drone can't be used to track that target until this processing has been completed.
- Drone will not have the capability to respond to wind or drafts. Performances must take place in scenarios where the risks of such are minimized.

1.7. Expected End Product and Deliverables

Expected End Product

There will be two end products to deliver to the client the drone and the application which serves as a user control interface.

Drone

The drone will be a quadcopter capable of flight. This flight capability will be able to automatically handle stabilization, directional flight, taking off and landing. While in use the drone will attempt to follow the specified target using these built in flight systems. Product will be delivered by 05/01/2019

Android App

The android app will be a downloadable software able to be loaded onto **android devices only**. The android app will perform multiple tasks; allow the user to upload an image of the target, start/stop the drone, assign the mode of recording and tracking, command the drone to perform an emergency landing, and receive the recorded video for user viewing. Product will be delivered by 05/01/2019

Deliverables

Other deliverables include project reports and technical documentation.

2. Specifications and Analysis

2.1. Functional Requirements

Flight Control

The drone needs to be able to control each motor individually. The input for control will be sensors and processed image recognition which will output commands to the flight control which will in turn control the engines.

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 image and tracking data which it will further process to create flight commands that it will send to the flight controller.

PID Controllers

To control the drone the image processor will take frames from the video stream and output data which the PID controller can use as its current point, where it is in relation to the dancers. The PID controller set point, where it should be in relation from the dancers, should remain static i.e. we want the drone five feet from the dancers. Another PID controller will also be fed data from either a barometer on the flight controller or an additional sonar for the purposes of determining and maintaining a constant altitude. The Flight controller may be able to maintain altitude on its own depending on the hardware we purchase.

Video Quality

The drone shall be able to record video at a minimum quality of 720p. The video footage acquired will have both dancers present in 80 percent of frames.

2.2. Non-Functional Requirements

Security

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, movement forward and backward as well as side to

side, platform rotation, motors turn on/off. 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.

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 wild unpredictable 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 minimizing the lag between dancers movement and drone movement in response.

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.

Useability

The setup and usage of the product must be simple enough for any user to complete a majority of the use cases. Some use cases are naturally going to be more difficult and require a level of involvement regardless of technicality.

2.3. Proposed Design

Hardware

The purpose of hardware is to provide features to our quadcopter such as the ability move, obtain data, and process data. Each component that we use to assemble the drone will fall under one of the four following 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 there will be 4 motors spaced evenly 225mm from the center of the drone. The individual components and how they are linked are shown in Figure 2.3.1

Command systems will be the brains of the quadcopter and the central hub from which commands are given and processed. The command system will tell the motor systems how to maneuver the quadcopter so that the video systems can take footage of the target dancers. The two current components are a raspberry pi and a pi powersource.

Motor Systems will be 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. The motors will provide 3440 grams of maximum thrust or be able to generate 800 grams per motor.

Video Systems will be what captures and records the video. As of now, we are utilizing a cheap pi camera for testing purposes. The video system transmits data of where the dancers are back to the command systems. 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 but needed anyway. The only equipment in this system are charges for the quadcopter lipo battery. The charger must be a balanced charger meaning it checks the voltages in each individual cell.

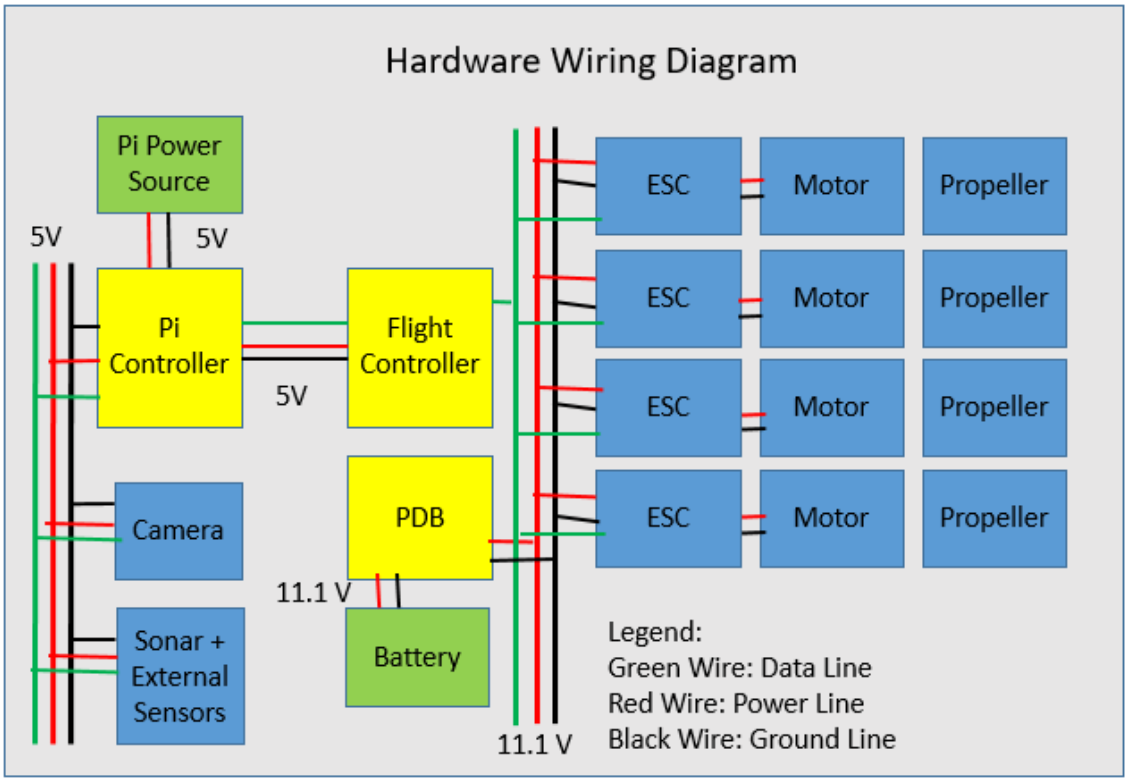

Figure 2.3.1. Hardware Wiring Diagram

Shows how each component is linked together whether through power or data wires.

Software

The software's purpose is to fulfill multiple difficult tasks. It 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 theatres. The AI will rely on real time systems to maintain the threads. This will help prevent confusion to the AI and prevent a

backlog of data which would potentially lead to a failure. Additionally, all threads will be synchronized to prevent complications. Figure 2.3.2 shows a conceptual sketch of the software.

The AI could potentially be confused if a thread which was created before another thread, finishes after a thread created later. This means that all threads must be synchronized to prevent such complications.

The software is currently in its early stages. So far it is capable of recognizing a human face (though not yet capable of distinguishing between a set of faces - ie recognizing or identifying a particular individual). The idea of multithreading is also being explored and there currently exists two threads. One is the main thread which captures video and stores it, the second is the processing thread which finds human faces in a given frame of the video. Even in its early stages, the method is working out quite well. The video stream maintains a steady fps and image processing can find a face roughly every ⅕ of a second. Further testing is required to determine if image processing is fast enough to fulfill the responsiveness requirement.

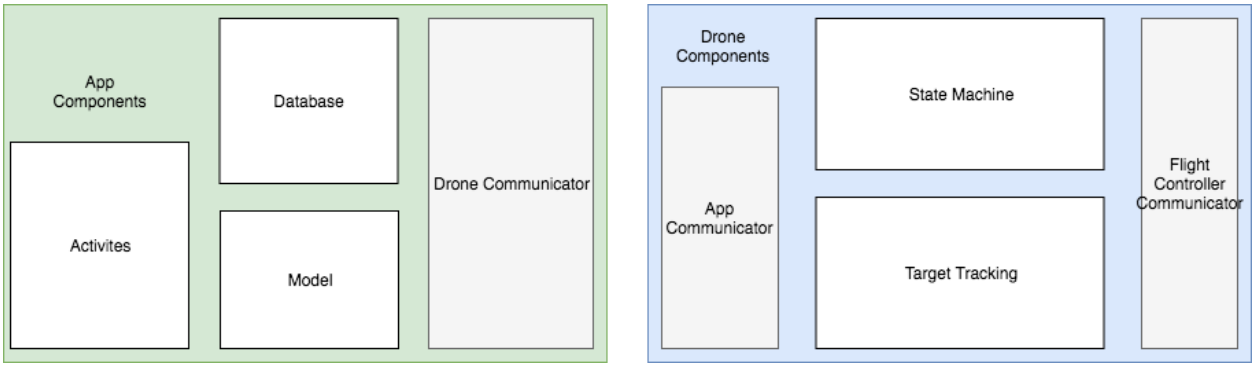

Figure 2.3.2. Software Component Diagram

This is an overview of the proposed software component diagram. The App components consists of the user Activities (UI), the models, databases, and drone communicator.

2.4. Design Analysis

Hardware

Command System: We decided a pi would be able to handle all the computing necessary for this project, and offer more options for software design than an arduino could. More members of our team have worked with pis over arduinos. We also chose to use a separate battery for the pi. This is because the primary battery produces an 11.1 V while the pi takes in only 5V. We also do not want the pi taking power that could be diverted to the motors for a longer flight time. A separate battery seemed the easiest option for us to implement.

Motor System: Compatibility of components in this section is incredibly important so as to not damage any hardware onboard. The simplest way to do this is to choose one part and develop the quadcopter around this. Since we are carrying a load, we decided the motors will be the primary component for which all else in this system will be compatible with. We chose a low kv (920) rated motor so that we can get more stable flight through the greater torque produced by the motor. The ESC choice was simplified by purchasing a bundle since that is more budget friendly, and we know that the ESCs works with the motors. The ESC rating was a maximum of

11.1V meaning we needed a battery rated at that voltage. The battery must be capable of powering the quadcopter for 5 minutes. The below equation calculates flight time:
Quadcopter flight times =(Battery Capacity * Battery Discharge /Average Amp Draw)*60

We chose our battery to have 5000 mAH and a discharge rating of 50C to reach 5 minutes with our motors. Lastly, our low kv motors are most efficient with large props so we used a prop type referenced with a datasheet for our motors. This caused us to also need a large frame.

External System: Our charger needed to be able to charge the battery safely. Thus we used a model that was referenced in a website. The charger is able to charge in about 2 hours and provide protection to the battery as well through an integrated circuit onboard the charger.

Video System: This last system was constrained by whatever remaining resources not utilized by the previous systems. In this case, it was cost and weight. At this point in the design, the quadcopter has about 80 grams remaining before being overburdened. We also only have $10 for this semester's budget. Thus we picked a cheap small component that we could upgrade in the future. Ideally, we'll upgrade the camera to be able to take higher quality video as well as attach hardware to allow smoother video capture. Finally an SSD card will be obtained later to store the video onboard the quadcopter.

Software

At this time, multithreading has been used to allow the software to stream video into a file stored in onboard memory at the same time snapshots are taken from the video stream and passed to a second thread where processing to find targets occurs. Further testing is required to determine what needs to be streamlined - however, the fundamentals of the idea are producing very satisfying results. I plan is to expand the idea to break the software down into a number of modules that will each be responsible for a separate key element of the software's functionality.

The software will be broken down into 6 modules. Each module is dedicated to its own thread. This will allow large computations to be made simultaneously in hopes to meet the responsiveness requirement.

- Main Thread - synchronizes the other threads which will all branch off of the main thread. The main thread will be responsible for initializing all variables and systems as well as shutting them down, failing nicely, etc.
- Video Streaming - This thread will capture video from the camera and either store it into a file or stream it to an off-platform device.
- Image Processing - Video streaming will pass an image to this thread. This thread will make the necessary pre-processing transformations (greyscale, resize) and then find the target in the image.
- AI Brain - This thread will be responsible for tracking a series of output from the processed images. This will know where the target was, where the target is, and where the target is likely going. Then the brain will 'think' about what its next action should be.
- Module Communication - This thread will communicate with the hardware modules. It will grab readings from them and store them in global variables to be accessed by the AI brain. This module will also take commands from the brain to communicate them to the hardware modules

- Client Communication - This thread will open a server port to accept the client app device and listen for commands. These commands will bypass the AI brain and go straight to module communication. The AI brain will be notified that external commands have been prioritized and the AI brain will rest while it waits to regain control of the device. Lastly, this thread will notify the client when a command has been received and again when the command is completed.

3. Testing and Implementation

Testing is an extremely important component of most projects, whether it involves a circuit, a process, or a software library
Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case
5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

3.1. Interface Specifications

User-App Interface

The primary source of user interface with our drone will be through the android application. This is primarily consist of the user clicking buttons, connecting to the drone's bluetooth, and taking pictures of the desired targets.

App-Drone Interface

The android app and the drone will communicate using bluetooth sockets using JSON. The reason we have chose this standard for our communication is because the JSON language integrates with all object oriented languages.This is useful because the Android App is written in Java and the drone code is written using Python. JSON makes coding between languages easy because it processes complex objects into strings.

A JSON object is enclosed by curly brackets '{}'. A JSON array is enclosed by square brackets '[]'. JSON Variables are denoted by the name of the variable and the value of the variable separated by a colon ':'and multiple variables are separated by commas. JSON objects

cannot contain methods. JSON objects are able to be nested inside one another but that is outside the need of this project and outside the scope of this explanation.

An **example** JSON object might look something like this:

```
{
Command: “arm drone”,
DronePassKey: “Ab12C1123Dd”
}
```

Before a JSON object can be sent over the socket it will be turned into a byte array. This is because a byte is a byte. This will also ensure that no data will be lost, nor misunderstood based on UTF-8 or conversions of plaintext characters.

Once the JSON object has been serialized into a byte array, the length of the byte array will be the first thing sent over the socket. This will allow the receiver to know the length of the byte array and to know when to quit attempting to read from the socket and to start processing the byte array into a JSON object.

Software-Hardware

For both the Sonar Sensor and the Flight controller we will be using python to integrate with these products. For the Sonar Sensor we will be using Scipy.Signal.Square to generate the square waves. For the Flight Controller we will be using pyMultiWii.

3.2. Hardware and Software

Cygwin

Linux-like terminal for Windows with some POSIX support. Used for unit testing of C code in a Windows environment.

Square Wave Generator

For testing we will may to generate square waves to test our Flight Controller. Upon receiving any electronic component we will measure them for adherence to advertised specifications.

Oscilloscope

We will need to read the square wave that the Pi generates to talk to the sonar for unit testing. Upon receiving any electronic component we will measure them for adherence to advertised specifications.

PuTTy

Used to read digital signals coming from Pi and Fight Controller. Upon receiving any electronic component we will measure them for adherence to advertised specifications.

Multimeter

Used to read voltage and amps that the flight controller and ESCs output to the motors. Will also be used to verify the exact voltages and amps the battery puts out. Upon receiving any electronic component we will measure them for adherence to advertised specifications.

3.3. Functional Testing

Unit Testing

As we develop hardware and software we will test them in isolation by developing unit tests. These unit tests will emulate expected inputs and test against expected outputs. We may have to work with other pieces of hardware or software to elicit what expect values should look like. For example the flight controller input is a digital signal but what do each of the pieces mean and how often can the board receive signals? All question that have to be answered before we attempt to develop software for it.

After we've elicited our test values and tested for expected values we will then test how the piece operates against unexpected values to verify that the program can continue to function normally or fail safely if it or another component fails during operation. After a piece has completed Unit Testing it can move on to Integration testing. There are 6 individual units that will be tested; they are the command systems, the motor systems, the video systems, the external systems, the pi software, and the android software. The command systems, the motor systems, the video systems and the external systems are all part of the hardware. The command systems will control the power supplies and must be able to supply power for up to 5 minutes. The motor systems controls the motors and must be able to provide enough lift to allow the drone to hover and move for up to 5 minutes.

The video systems control the video for the user and control supplying the pi with frames for body and facial recognition. It must be able to connect to the pi for the entire flight and record the entirety of the dance. Lastly for hardware testing is the external systems. These are the batteries and they must be able to last the entirety of the flight without falling below 80% battery life. This will be tested by running the drone for 6 minutes and 15 seconds. For Software, we will be testing to make sure that the app will connect to the pi for up to 10 meters and send commands to the pi. The pi will be tested to make sure that the commands are received and correctly implemented to the drone.

Integration Testing

Once a piece has passed all its unit tests it can move on to Integration testing which will test its performance once it has been implemented into the system where it will function in the final design. The first step in Integration Testing is to integrate the piece of software or

hardware. Then we will run the software's unit tests after integration to make sure it still functions properly. For example a piece of software runs really well in isolation on a laptop but once it gets to the Pi it runs slow or doesn't work as the Pi doesn't have all the libraries. If we run into problems here we'll remove the piece from the whole and go back and develop run the unit tests again and attempt to reintegrate. If the piece works well once integrated we'll further test its capabilities to function with necessary pieces of hardware and software.

We'll start by testing each piece one by one and then test them together. Say a piece of software processes the signals coming from the sonar and outputs a value that the flight controller uses to control the motors, then we'll test that it can process data coming from sonar and check the values it outputs then we'll test those output values on the flight controller. After we've tested each piece in isolation we test the entire subsystem. So in the example sensor sends data to program which sends signal to flight controller which sends power to engines. Once we've verified that the sub systems relevant to piece are functioning correctly we will move on to system tests to verify that the integration of the new piece has not screwed with any existing individual piece, functionality, or subsystem.

System Testing

What defines the system is will change over the course of the design. We will start with a few pieces of hardware and software and then build upon them. As we build we will generate tests for individual pieces of hardware and software as well as the sub systems they create together. We will maintain a running log of unit and sub system tests which will be used to verify that a new piece has not somehow affected other functionality and to test the whole system. As we add and verify pieces we will add that their unit and sub system tests to the log. With each new piece of hardware or software, we will create a new unit test which will then require a new integration test which includes the new addition.

Acceptance Testing

Acceptance testing will start with defining what functional requirements we need to produce. Then once we've developed and tested each functional requirement, the system is complete, and we have tested everything we can move onto acceptance test. We will take our list of functional requirements and run a set of acceptance tests verifying each of them, going back and fixing any that fail. Once that is complete for all functional requirements we can move on to full speed acceptance testing. During full speed testing we will test how the drone properly functions as a whole system in its environment. We will take necessary safety precautions to ensure safe operation of the drone for safety of the operator and testers as well as preventing serious damage to the drone. Full speed testing will test normal operation, its built in safety features, and how it operates under less than ideal but possible circumstances. Once it has passed the final stages of full speed testing the drone, its software, and subcomponents will be considered accepted and the project will be complete.

3.4. Non-Functional Testing

Security

In order to test security we will ensure that device communication is encrypted and the traffic should not be able to be sniffed, analyzed, or replayable. We will test to make sure that basic attacks cannot effect the drone or software and we will ensure that all of our components are made to be secure. In order to test this we will be using wireshark to watch the communication to ensure that it is indeed encrypted. If the traffic is encrypted then the test will be successful.

Responsiveness

There will be two way communication between the drone and the user's phone. We will first test that basic commands are sent and received by the drone and the app. These tests include arming/disarming, moving, rotating, etc. If we are able to make the drone move in the desired fashion then the first part of this test will be successful.

Reliability

Testing reliability will require us to test how well the quadcopter can maintain connection with the client app as well as how it handles failure and how well it recovers. This will require us to create scenarios that cause failures. Logs of the onboard process will be kept for review after the test is finished. The logs combined with visual evaluations of the protocols' effects in the real world will help us evaluate the success or failure of these tests.

Useability

Useability will be tested using the list of use cases in the above sections. We will test our useability by ensuring that the target users of our project will be able to perform all the cases listed above.

3.5. Process

At this stage in the project, the drone is being build, and an algorithm for tracking is being written.

Building the Drone

Building the drone started with finding the parts we needed. Several meetings occurred to talk over what parts would meet functional requirements. For example, we wanted to have wifi and bluetooth capabilities for the drone, so we select a Raspberry Pi 3b as our onboard computer. When all the parts were chosen and delivered, measurements were taken and the drone engineers played with different designs and setups until they had a place for each part and weight was evenly distributed across the drone. The weight of each fastener, screw, and

shelf was also measured to ensure that the drone maintained at least a 2:1 thrust to weight ratio.

Writing Tracking Software

The tracking algorithm was first given the ability to track the target based on facial recognition. Then an algorithm allowed for tracking based on euclidean distance to the target's last known location on frame. Some other tracking methods were implemented for tracking based on directional velocity and line of best fit over a series of known locations of target. Then each of these methods were combined into a parent algorithm which would "choose" the most effective tracking method or combination of tracking methods at a given time. The optimal tracking method is based off of where the target is located, where distractions are located, where objects are moving around, and when was the target's last known location.

Testing Tracking Software

A handful of animations containing people moving across frame were drone up and rendered to assist in testing the tracking software. Some scenarios contain only the target, others contain the target as well as distractions from the target. These animations are then fed into the tracking software in place of the drone's live feed camera. A test file is associated with each animation, containing a series of numbers that represent the target's location in each frame. As the software processes the simulation, it tracks a few variables to represent the success factor of the software.

1. (Number of frames where target was found) / (Number of frames containing target)
2. Average distance between target's actual location and location where software found target
3. Time it took to complete test

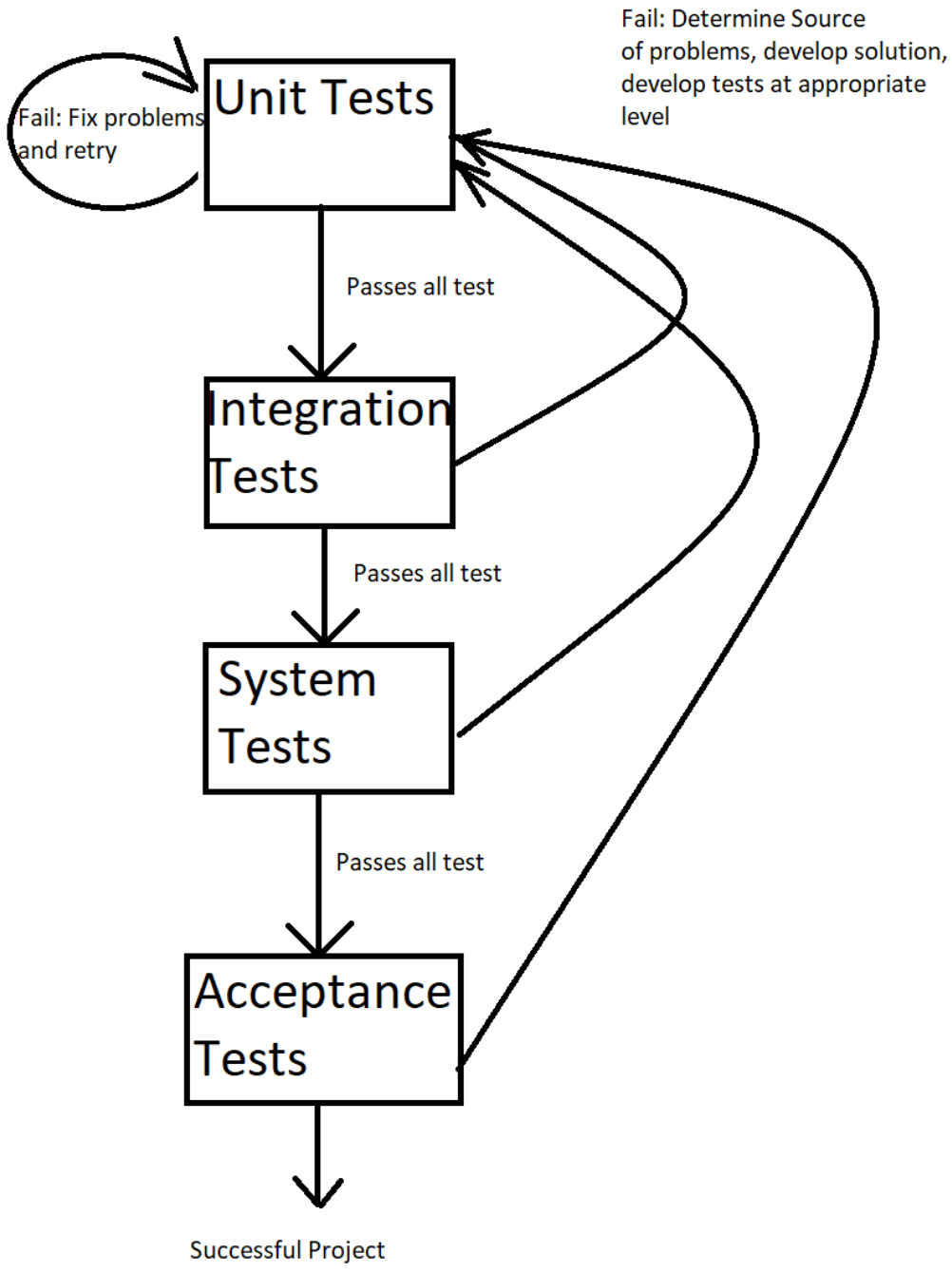

Figure 3.5.1. Test Process Flow Diagram

Flow diagram of the Test Process followed by this project

3.6. Modeling and Simulation

Testing the software's tracking ability can be simplified by simulating scenarios. Since it can be difficult to create real-time scenarios for our software to perform in, and even more difficult to accurately and consistently recreate scenarios, we need a way by which scenarios can be run and re-run consistently. We accomplish this by creating animations of people moving across a frame. The animation can be fed into the software in place of its live-feed camera, and it will then each frame and track targets as if it were a live run.

Several scenarios have been covered. In one scenario, a single target move across frame. In another, the target moves across frame while a non-target human remains stationary. The scenarios get more complex until we have multiple people moving and at times obstructing our target. When our software is working as intended, the software should very rarely lose track of our target.

3.7. Results and Implementation Challenges

Hardware

The first issue we encountered with our drone was that the amount of space available to attach components is limited. **Figure 2** shows the whole drone with the lipoly battery occupying the space in the bottom "shelf" of the drone, and the pi power source in front of the battery outside of the frame. We considered removal of the pi power source and using a power distribution board with a 5V voltage regulator to supply power to our pi. However, this diverts power from the motors thereby limiting flight time below our 5 minute threshold. Another option is to place components off center and next to each other. However, this results in uneven weight distributions. Our final option is to create an additional "shelf" to mount components to, but this will be at the cost of additional weight. One of of our engineers completed laser cutter training to be able to create a flat acrylic platform to be attached underneath the bottom platform that came with our drone. This is likely the best solution to be implemented.

Our next significant challenge is the late arrival of our flight controller. We placed an order in early October, and the component has yet to arrive. We decided to investigate the website selling the product and realized that the website has fairly negative ratings. We believe our part was never shipped to begin with and the company simply took our money without any intention to deliver the product. We have researched alternative flight controllers as well as looking at other suppliers of the controller we ordered. After this incident we also plan to consider the reliability of suppliers before ordering from them to ensure we don't have this issue occur again. We will likely also order a cheap flight controller to work with until our ideal controller arrives. This has ultimately affected our ability to produce a working drone since the flight controller is essential to all the motor systems working correctly to establish flight. Without this we cannot fully assemble the drone.

Figure 3.7.1: Power Supply to Frame Size Comparison

An image of the Power Supply compared to the size of the Frame.

Facial Recognition

At this time, the software has been tested its ability to recognize a human face. To test this, the camera recording the camera and the performer are kept in motion for 10 seconds. This produces 300 frames. Since all frames have a human face in them, we are able to divide the number of frames were a face was found by the total number of frames and this will directly reflect the percentage of effectiveness of our ability to recognize a face. From a frontal angle, approximately 93% of frames are successful. Tests demonstrate that with the current haar cascade for facial recognition, we have a very small range of angles we can take from the

camera to the face that is to be recognized. Once the camera takes a +/- 20 degree angle, we begin seeing a less than 50% success rate.

4. Closing Material

4.1. Conclusion

During the creation of this project there has been many learning opportunities. We've learned how to use the python language and android development. We've researched how the basics of aerodynamics work. We've learned algorithms for image processing and target tracking. Perhaps most importantly we've gained experience in system design and documentation.

Working on the project has benefits our group by teaching us how to work on a project that is actually long term, compared to a semester long project. This experience has taught us how to design systems and attempt to foresee any issues those designs might have. They've taught us how to work together as a group and to recognize other's strengths and try to supplement their weaknesses with our strengths.

Going forward we plan to continue improving our design to make the best product possible. To accomplish this we will continue to work closely with our client to ensure they receive the maximum result for this project.

4.2. References

Ageitgey, “ageitgey/face_recognition,” *GitHub*, 30-Oct-2018. [Online]. Available: https://github.com/ageitgey/face_recognition. [Accessed: 01-Dec-2018].

Alduxvm, “alduxvm/pyMultiWii,” *GitHub*, 08-Aug-2018. [Online]. Available: https://github.com/alduxvm/pyMultiWii. [Accessed: 01-Dec-2018].

“How To Build A Quadcopter,” *RSS*, 2018. [Online]. Available: https://myfirstdrone.com/build-your-first-quad. [Accessed: 01-Dec-2018].

sdmay19-42, “Project Plan v.3” *Google Doc*, 1-Dec-2018. [Online]. Available: https://docs.google.com/document/d/1kXX9-RPbnwRTnAHN69FHwD5lHd79avN86Fvp1hA8ZCA/edit#. [Accessed: 1-Dec-2018].

4.3. Appendices

CRIUS All in One Pro Flight Controller V2.0 Data Sheet - Explains how the Flight Controller Works

https://us.gearbest.com/development-boards/pp_69471.html

OpenCV - Open source Computer Vision. Provides a library for image transformation and image based machine learning.Our team is using this for facial recognition.
https://docs.opencv.org/2.4/modules/refman.html

Motors Data Sheet - Explains the operational parameters of the motors.
https://www.aliexpress.com/item/4PCS-GARTT-ML-2212-920KV-230W-Brushless-Motor-For-DJI-Phantom-QuadCopter-F450X525-Multirotor-Drone/32561949556.html