Quadcopter Delivery Verification

Design Review

ECE 445 SP2016

TA: Luke Wendt

Team Members: Sachin Weerasooriya - Electrical Engineering

Raymond Hoagland - Computer Engineering

Rahul Joshi - Computer Engineering

February 29th, 2016

Table of Contents

1.0
Introduction....2
1.1 Statement of Purpose....2
1.2 Objectives....2
1.2.1 Goals....2
1.2.2 Function/Features....2
1.2.3 Benefits....2
2.0 Design....3
2.1 Block Diagrams....3
2.2 Block Descriptions....3
2.2.1 Drone....3
2.2.2 Camera....3
2.2.3 Odroid C1+....4
2.2.4 Pixhawk....4
2.2.5 Phone App....4
2.2.6 Motion Sensor....5
2.2.7 Force Sensor....5
3.0 Schematics....6
3.1 PCB Schematic....6
4.0 Simulations....7
4.1 OpenCV Algorithms....7
4.1.1 QR Code Detection....7
4.1.2 HSV Filtering....7
4.1.3 Distance Detection....8
5.0 Requirements and Verification....8
5.1 Requirements and Verification....8
5.1.1 Power Module....8
5.1.2 Sensors....9
5.1.3 Analog to Digital Converter....9
5.1.4 OpenCV....10
5.1.5 Phone App....11
5.2 Tolerance Analysis....11
6.0 Software Flow Chart....12
7.0 Ethical Analysis....12
8.0 Safety Measures....14
9.0 Cost Analysis....15
9.1 Parts....15
9.2 Labor....16
9.3 Total....16
10.0 Schedule....16
11.0 References....18

1.0 Introduction

1.1 Statement of Purpose:

When you order a package that is shipped via ground, it is either left at your door or you must be present to sign off on the package. Services like Amazon PrimeAir look to speed up delivery time with drone shipping, which claims to be able to come to your door in 30 minutes. Their current delivery model has a shipment loaded onto a drone which takes off, converts to a plane, flies to the vicinity of the landing local, converts back to a drone, and lands on a marker put out by the recipient. This drone then deposits the package and returns to the factory for its next package. Here lies a major flaw; in the event that something valuable is being shipped, it would be desirable for their to be a confirmation that someone is available to pickup the package that is being delivered. We aim to solve this by creating a drone that will wait for customer verification before the drone is delivered.

1.2 Objectives

1.2.1 Goals:

- Design an embedded system on a drone to authenticate delivery with customer
- Determine proper landing location when in close proximity of customer
- Wait until notified by customer to drop off package or leave with package if customer doesn't make contact

1.2.2 Functions/ Features:

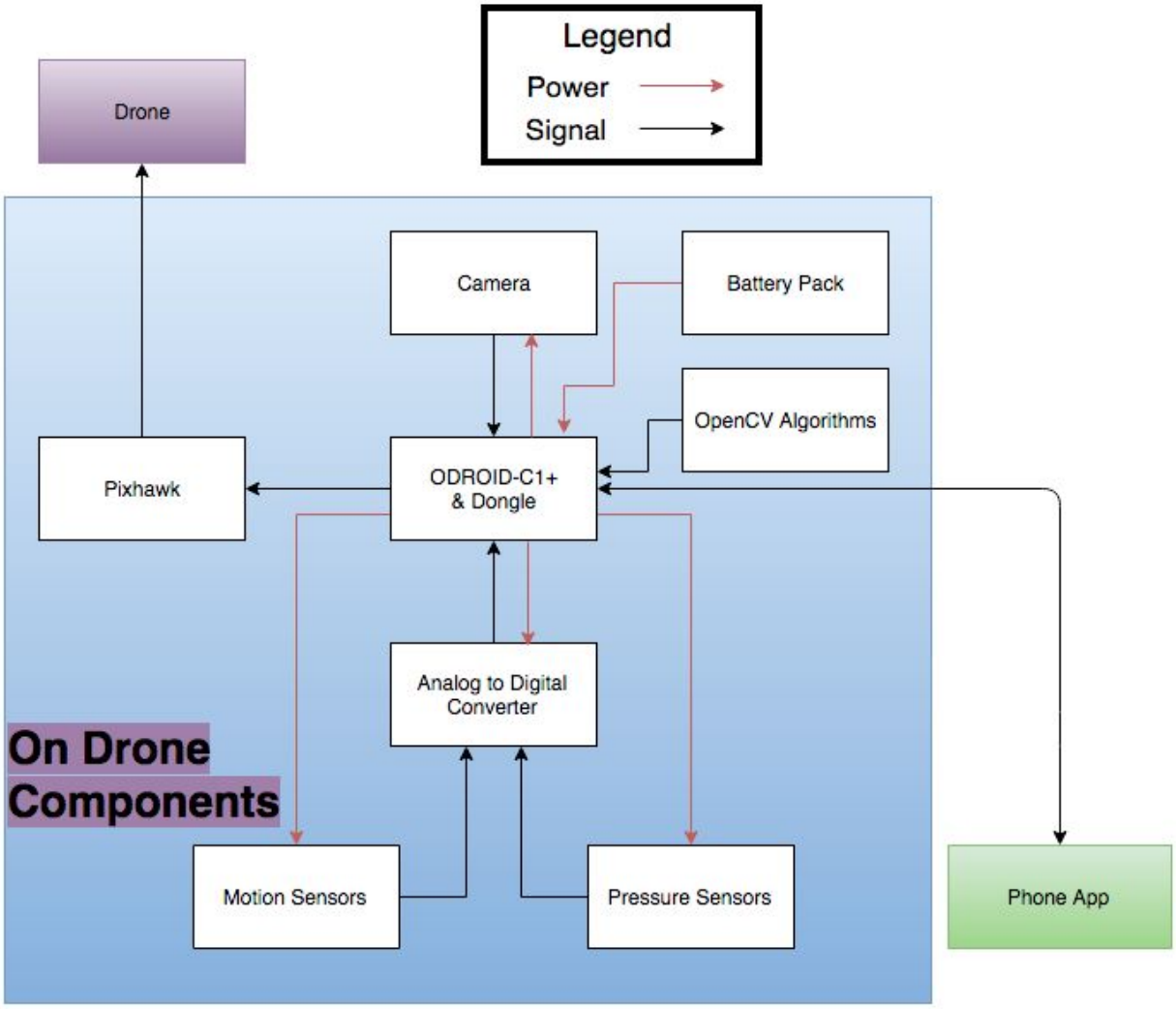
- Communicate with end-user through app when landing location has been found so that user may acknowledge delivery
- Identify and land in zone signified by fiducial marker
- PCB containing
- Observe surroundings for potential hazards to package using motion sensors
- Takeoff to evade any detected threats and light up red LED
- Have green LED light up when customer confirms package delivery
  (will not be implementing mechanics to drop off package as this is out of our scope)
- Takeoff with package if contact with customer is not made

1.2.3 Benefits:

- Enables speed of drone delivery but security of standard ground signature authorized deliveries for valuable packages
- Cost-effective automated delivery system

2.0 Design

2.1 Block Diagrams



2.2 Block Diagram Descriptions

2.2.1 Drone:

Input: Commands through MAVProxy will be sent to manipulate positioning and height.
Output: LEDs to indicate flight status (searching, landed, confirmed).

This is the main component of the project, which all other components will be mounted to. The drone will leverage these components to navigate to a designated drop zone to deliver a package. The drone will also have safeguards for low battery warnings or other concerns.

2.2.2 Camera:

Input: Visual information on surroundings on the ground. Used to identify landing location and determine distance from the ground.

Output: Video feed processed by OpenCV to determine the drone's position in reference to the landing location.

The camera is used for border detection, QR code decryption, and height detection. The feed output from the camera will be used to capture and process the images to locate the landing position.

2.2.3 Odroid C1+:

Input: Video feed captured from camera using OpenCV.
Output: Commands through MAVProxy shell to control drone positioning.

Once the drone has reached the designated drop off point, it needs to notify the recipient. The C1+ will send a notification to the recipient's phone app to indicate arrival. The C1+ will also receive confirmation that the recipient is ready to pick up the package, indicating that the drone can leave. The C1+ will also receive the output of the camera for image processing through OpenCV. OpenCV code in C++ will be used to perform HSV filtering on the captured images to identify the pattern specific to the landing pad border. This border will be denoted by alternating rectangular sections of red and blue surrounding the QR code in the middle. OpenCV will also be used to perform contour detection to match the specific nested square patterns found in the corners of the QR code and then decode the QR code. Finally, OpenCV will be leveraged to determine the drone's height after identifying the focal length of the camera using a default initialization image with a known distance. Using the results of analysis using these OpenCV methods, the C1+ will run commands through the MAVProxy shell to communicate with the Pixhawk to adjust throttle power for each rotor. The C1+ will also receive signals from the motion sensors and force sensors through GPIO pins which will be analyzed to adjust the throttle response as well. If the motion sensors are triggered, the drone will take off. If the force sensors - on the bottom of the drone legs - are triggered, the drone will decelerate to land softly.

2.2.4 Pixhawk:

Input: Commands through the MAVProxy program to communicate with the drone's motors.
Output: Throttle inputs to the drone motors.

This is an open source controller used by our drone. It allows for a connection to the C1+ through the Telem-2 port so that the C1+ can execute commands to control drone operations.

2.2.5 Phone App:

Input: Notification that the drone has arrived and is waiting for delivery.
Output: Confirmation that user is ready to retrieve package.

The phone app receives the notification that the drone has arrived at the designated landing location and allows the user to send the drone a confirmation signal that he or she is in the vicinity and ready to pick up the package.

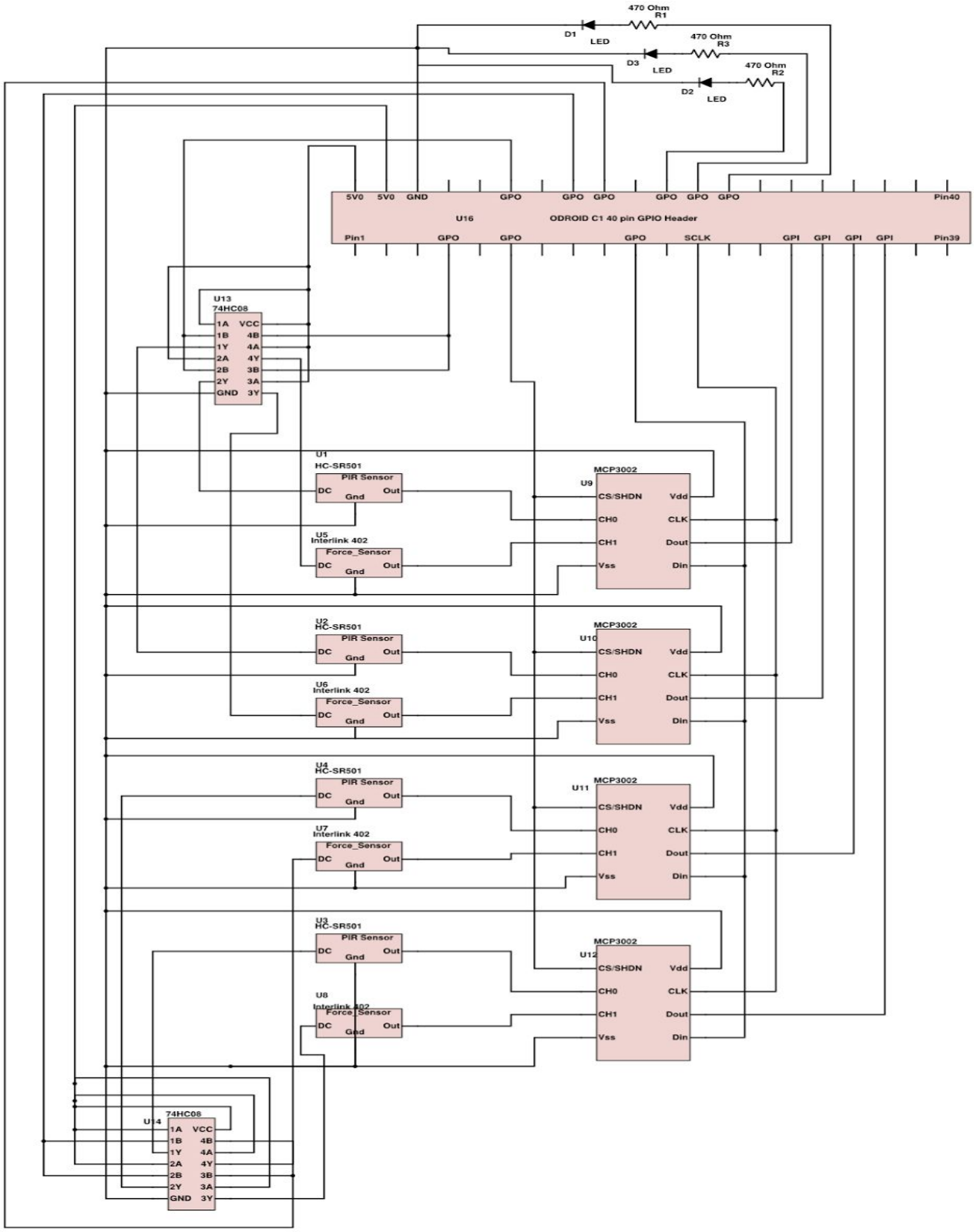
2.2.6 Motion Sensors:
Input: Visual feed to analyze for changes in the environment.
Output: High signal if motion is detected by any of the sensors.

There will be one motion sensor placed on each side of the drone. These 4 sensors will monitor each side of the drone for potential movement. In the event that movement is detected, a signal will be sent to the Odroid C1+ to take off. The type of sensor being used is a basic passive infrared (PIR) motion sensor.

2.2.7 Force Sensors:
Input: Information on the pressure exerted on the legs of the drone.
Output: A high signal if significant pressure is detected by any of the pressure sensors on the drone legs.

There will be one force sensor placed on the bottom of each drone. Image processing will be used to an extent to determine the height of the drone. Once it is known that the drone is only a few feet off the ground, the drone will slowly descend the final few feet until the force sensors are triggered on the ground. At this point, the drone will decelerate.

3.0 Schematics



4.0 Simulations

4.1 OpenCV Algorithms

4.1.1 QR Code Detection

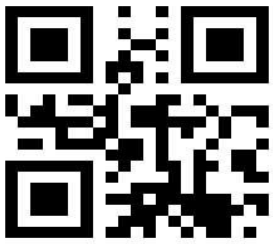


Figure 1 Original QR-Code



Figure 2 Threshold imaging to sharpen contour areas for decoding



Figure 3 Detection of contours in QR-Code

The first figure above is the baseline QR-code that we are parsing for and wish to decode at a certain distance. The second figure shows the result of a applying a threshold filter to a captured image to sharpen the contour areas of the QR-code to help the decoder recognize/scan the image. The third figure shows the detection of the contour patterns in the corners of the QR-code which helps us crop the captured image to zoom in on the QR-code and decode it more easily.

4.1.2 HSV Color Filtering

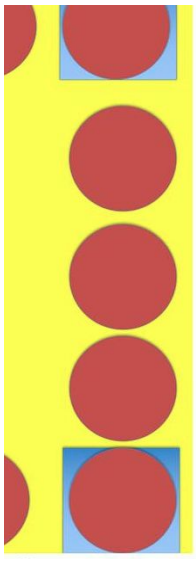


Figure 4 Landing pad



Figure 5 Detection of red circles using OpenCV

Figure 4 above show a part of the landing pad and figure 5 shows being able to detect the red circles using hsv color detection in OpenCV. This color detection will help identify the landing pad from the height the drone will be hovering at.

4.1.3 Distance Detection

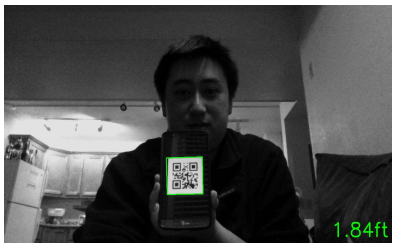


Figure 6 Initial Distance from Camera

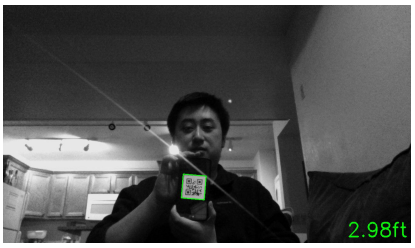
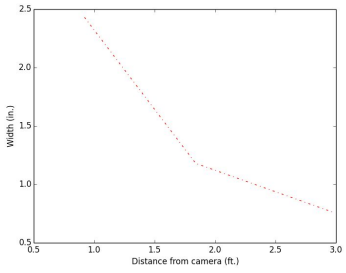


Figure 7 Second Distance from Camera



The above images were taken from three different distances, each progressively farther away. Using the predetermined length of the QR code from the camera for the first image, we were able to determine the focal length and distance from the camera for each of the next images. These images are to help figure out at what height the QR code will still be recognizable. We wish to determine the ratio between the area of the QR-code and the distance from the ground. Using this ratio, we can determine the maximum height at which the drone can scan the QR-code.

5.0 Requirements and Verification

5.1 Circuit

5.1.1 Power Module

Requirements	Verification	Points
Odroid C1+ must be provided biasing voltage of 5(+/-.1V) with a peak available current draw of 2(+/-.25A) and have the device run for at least 10 minutes	1. Power up device on the ground 2. Disconnect MavProxy signals in the test script so that the ODROID is still doing all the processing but the drone is safely on the ground. 3. Time to ensure the ODROID is powered for at least 10 minutes by ensuring the LED1 is lit	10

Odroid C1+ GPIO pins must provide biasing voltage of 5(+/-.25V) for the 4 PIR sensors	1. Probe DC pin on devices u1, u2, u3, and u4 (found in schematic on pg6) using digital multimeters 2. Ensure that the measured voltage falls within 4.75V-5.25V	

5.1.2 Sensors

Requirements	Verification	Points
Passive Infrared (PIR) Sensor - Must be able to detect an object approaching the drone within 12(+/- 2m)	1. Have drone landed with motion sensor attached or disjoint from drone 2. Have one person stand over 20 meters from drone and begin walking to drone 3. Probed output voltage using multimeter from sensor during this period should be at 1(+/-1)V 4. Once person approaching drone is within 12 (+/-2m), probed output voltage using multimeter from sensor should be 3.3(+/-.5V)	12.5
Force Sensing Resistor (FSR) - Resistor must have nearly infinite impedance (>50KOhms) when sensor isn't pressed. Each FSR must have resistance of 12.5 (+/- 2.5KOhms) when drone has landed	1. Lift drone so that none of the sensors are being pressed 2. Measure resistance of each sensor to ensure that they all have resistances greater than or equal to 50 KOhms 3. Place drone flat so that all of the sensors are being pressed 4. Measure resistance of each sensor to ensure that they all have resistances between 10 and 15 KOhms	12.5

5.1.3 Analog to Digital Converter

Requirements	Verification	Points

1. The ADCs must be provided with 3.3(+/-.3V) to be used as a reference voltage and a biasing voltage 2. Ensure that data is being clocked in when intended by user once every .5(+/-.1sec)	1. Probe Vdd pin on devices u9, u10, u11, and u12 (found in schematic on pg6) using digital multimeters 2. Ensure that measured voltage at each pin falls between 3V and 3.6V 3. Check the LED D2 to ensure that it flashes red every .4-.6 seconds to ensure that data is being gathered 4.	10

5.1.4 OpenCV Algorithms

Requirements	Verification	Points
Total 1fps (+/- .5s)	1. Python testbench wrapper using time module to measure time between image frame capture and output received from Border detection and QR code detection functions. 2. Ensure runtime less than 1 second maximum.	15
Distance calculation using focal length with accuracy of 85(+/-5)% at 10ft	1. Bring drone to height of 10 feet using given drone software 2. Run distance algorithm to get estimated height 3. Calculate the percent difference of the two heights to see if only 15%(+/-5)% different	10
Check validity of QR code with 85(+/-5%) accuracy at 3(+/- .5ft) off ground	1. Bring drone to height of 3 feet using drone software 2. Have many trial runs of QR-code decoder algorithm 3. Determine success rate by determining average success trials: # of successes/ # of trials	15
Ensure that the control algorithm gets the drone to land at the center of the	1. Modify initial takeoff software procedure so that drone flies at 5 feet off the ground as opposed to normal	10

landing pad +/- 2.5 ft from the middle.	30 feet to facilitate easy processing of the landing pad border 2. Run the main script without QR code reading and see how far from the center the drone lands	

5.1.5 Phone App

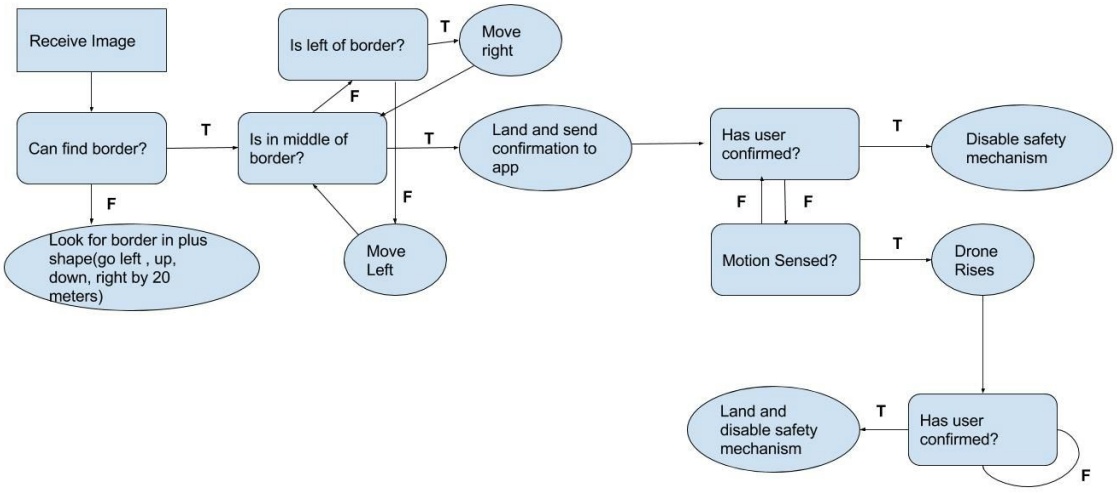
Requirements	Verification	Points
User confirmation sent and received within 1 minute(+-1 minute)	1. Send packet from app with timestamp, compare with time on odroid c1+ 2. Time within 1 minute to account for differences in clock speeds of app and odroid c1+	10

5.2 Tolerance Analysis

The major hardware component of the project that will most affect the outcome of this project is the accuracy and reliability of the PIR sensors. It is vital that the range that these sensors work accurately is relatively far away. Ideally, the sensors would work up to distances of 25 feet. With this range, there would be plenty of time for a sensor to detect motion, send a signal to the Pi, and fire up the motors so that the drone takes off in time to avoid the threat. In the event that the sensors cannot accurately tell that a threat is approaching until closer (i.e. 15ft) there is a chance that the assailant could get to the drone and steal the package before the drone is able to take off.

In addition, another problem that we could run into is receiving false positives from the PIR sensors. A false positive isn't necessarily catastrophic because it doesn't effect anything critical to completion of the mission; the drone would take off to a safe hovering height but would come down in the event that a confirmation message is received from the user. With that being said, it is still undesirable because it would use up a considerable amount of power in the event that the drone is forced to hover for any length of time. In order to remedy this, we have implemented ADCs to capture the analog data and convert it to digital for ease of processing in software. We will implement a filter where all signals that fall between a certain magnitude that are essentially noisy spikes but don't represent signals high enough for a motion signal to be attenuated. This should remove a majority of the false positives that we would incur.

6.0 Software Flowchart



7.0 Ethical Analysis

The project presented in this design review will be implemented with guidance of the IEEE code of ethics.

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment

The goal of this project is to find the specified customer landing location and safely deliver the package using a drone. The team working on this project must focus on safety of the customer and others around the drone. As the drone is developed and possible safety issues come up, they will be explicitly informed to those who may be affected.

2. To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist

Due to the nature of this project being for academic purposes, there is very little reason to have such conflicts of interest. If there are any conflicts, anyone affected by this project will be informed of immediately.

3. To be honest and realistic in stating claims or estimates based on available data

The data that is collected will not be falsified in any manner and be analyzed in strict accordance to the requirement and verification table in this design review.

4. To reject bribery in all its forms

As stated above, the academic nature project will make it very unlikely to receive such bribes and potential briberies will be rejected immediately.

5. To improve the understanding of technology; its appropriate application, and potential consequences

These ethical measures and safety measures outlined in this design review will guide us in understanding what are the proper applications and consequences of the use of certain technologies.

6. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations

This team has a diverse set of skills and will contribute to the project based on their experience with the two main areas being software knowledge and electrical engineering knowledge.

7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others

The advice from professors or TA's will be taken into full account in the duration of this project. The sources for many areas of our project from algorithmic ideas to hardware implementations will be properly cited.

8. To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression

All persons within the team will be properly conduct with each other and others outside the team fairly.

9. To avoid injuring others, their property, reputation, or employment by false or malicious action

The team will take precautions by following safety measure listed below when working on project testing and not engage in malicious actions against others.

10. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics

This ethical statement is to help this team follow the IEEE code of ethics and also to help other teams also uphold such a code.

8.0 Safety Measures

The goal of this project is to provide safe package delivery with user verification for drones. It achieves this goal by using a camera in conjunction with OpenCV running on a single-board computer to detect a known border pattern and unique QR code identifying the landing pad. When all LEDs are red, this indicates that the drone is actively searching for the border. Once this transitions to yellow, a prospective border has been found and the drone will begin positioning itself to move itself to the middle of the border. When this LED turns green, the drone is in the middle of the prospective border and will begin descending. When the second LED transitions to yellow, this indicates that a QR code has been detected and the drone will continue to descend until it can decode the QR code to check for validity. When this LED transitions to green, the QR code has been verified to be correct and the drone will begin landing. The third LED is used for the sensors. When the third LED is yellow this means the pressure sensors outputs are being analyzed while the motion sensor outputs are masked. If the LED is blinking yellow this indicates a pressure sensor has triggered high and the drone should stop moving down. When the LED turns green this indicate that the motion sensors are now being analyzed. If the LED is blinking green this will indicate that one of the motion sensors has triggered high and the drone will take over to hover in a safer location. The second LED will then blink red every time data is being analyzed via the ADC. That being, said please keep in mind the following security concerns.

1. Before operating the drone ensure the following checks have been completed.
    a. Drone is started from the off position.
    b. All wires are cleanly soldered and are not loose.
    c. Ensure the drone is programmed to stay within a 5 ft radius geofence and will return automatically if attempting to leave

d. No personnel are within the geofence and there are no objects impeding the drone's takeoff and touchdown areas.
e. New code has been peer reviewed and passes automatic testing
f. Override kill switch is functioning correctly.
g. Ensure the drone is fully charged.

2. During operation of the drone please be aware of the following safety hazards.
    a. The drone is completely controlled by software as no SIM card will be installed. Thus if the drone is not behaving as it should in any way, send the kill signal.
    b. The drone has a very limited battery so for testing purposes make sure to keep the drone within visual range at all times.
    c. Before approaching the drone make sure it is completely powered off or the verification code has been received. Perceived motion will cause the drone to rapidly elevate which may pose a safety threat to those in the immediate vicinity.

9.0 Cost Analysis

9.1 Parts

Item	Cost
Flytrex Sky Quadcopter	$549
180degree Fisheye Lens 1080p Wide Angle Pc Web USB Camera (USB500W02M)	$43
PIR Motion Sensor (HC-SR501) (4)	$8.49 ($33.96)
Force Sensor (FSR_0.5_DE2) (4)	$6.26 ($25.04)
Portable Charger External Backup Battery Power Bank by Mediabridge - 4600 mAh - 5V/2A	$19.99
Odroid C1+	$50

WiFi Dongle	$0 - already own
470 Ohm Resistor (4)	$0 - already own
LEDs (3)	$0 - already own
Total	$720.99

9.2 Labor

Name	Hourly Rate	Total hours invested	Total
Raymond Hoagland	$50	200	$25000
Rahul Joshi	$50	200	$25000
Sachin Weerasooriya	$50	200	$25000
Total			$75000

9.3 Total

Section	**Total**
Labor	$75000.00
Parts x 2	$1402.00
Total	$76402.00

10.0 Schedule

Week	**Sachin**	**Rahul**	**Raymond**
Week 1	● Finalize drone selection and purchase ● Finalize camera purchase ● Finalize sensor selection.	● OpenCV Installation on OSX ● Research Ardu-Copter source code ● Submit Proposal	● Contrast OpenCV and SimpleCV ● Research on MAVProxy/Pixhawk
Week 2	● Finalize ‘Circuit Schematic’ , ‘Calculation’ for Mock Design	● OpenCV research -- libraries and algorithms to use ● Finalize	● MAVProxy examples ● Image capture from live feed with OpenCV

	Review ● Submit Mock Design Review	'Simulation' for Mock Design review	● Finalize 'Safety', 'Block' Description for Mock Design Review
Week 3	● Finalize 'Circuit Schematic', 'Cost Analysis' for Design Review	● OpenCV HSV filtering to identify colored sections for border detection ● Finalize 'R/V Table', 'Simulation', 'Ethics/Safety' for Design Review	● OpenCV Contour Detection/Analysis for QR code corners ● Find Python QR decoder for testing ● Hough Circle Transformation for circle border detection. ● Finalize 'Schedule', 'Block Description' for Design Review
Week 4	● Finalize circuit design, noise filtering for sensors to eliminate false positives	● OpenCV HSV pattern matching for alternating red/blue patches.	● MAVProxy command integration with OpenCV output
Week 5	● Begin circuit construction ● Logic to match MAVProxy command blocks to OpenCV output	● Begin developing Android user application skeleton	● ODroid setup, OpenCV installation ● ODroid benchmarking
Week 6	● Test flight, check OpenCV algorithm effectiveness	● Finalize user application core	● Finalize QR code detector algorithm ● Finalize QR decoder
Week 7	● Test pressure and motion sensor detectors	● Finalize HSV pattern matching	● Finalize height detection algorithm
Week 8	● Power module and sensor verification check ● Test flight detection and landing test	● Single board computer verification check ● User application verification check	● Develop OpenCV testbench for timing measurements ● OpenCV verification check
Week 9	● 'Introduction', 'Block Diagram', 'Block Description',	● 'R/V' and 'Costs' for Final Report	● 'Simulation', 'References', 'Conclusion' for Final

	‘Schematic’ for Final Report		Report
Week 10	• Finalize ‘Introduction’, ‘Block Diagram’, ‘Block Description’, ‘Schematic’ for Final Report • Submit Final Report	• Finalize ‘R/V’, ‘Costs’ for Final Report	• Finalize ‘Simulation’, ‘References’, ‘Conclusion’ for Final Report • Project Code upload to Github

11.0 References

[1]"Introducing Copter." *Copter*. Ardupilot, n.d. Web. 29 Feb. 2016. <http://copter.ardupilot.com/wiki/introduction/>.

[2]"Getting Started with Images¶." *OpenCV 3.0.0-dev Documentation*. Opencv Dev Team, 10 Nov. 2014. Web. 29 Feb. 2016. <http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html>.

[3]"How to Create and Decode a QR Code in Python Using Qrtools."*Cerebro.toBlog()*. Ralgozino, 13 June 2011. Web. 29 Feb. 2016. <https://ralgozino.wordpress.com/2011/06/13/how-to-create-and-decode-a-qr-code-in-python-using-qrtools/>.

[4]Rosebrock, Adrian. "Find Distance from Camera to Object Using Python and OpenCV." *PyImageSearch*. PyImageSearch, 19 Jan. 2015. Web. 29 Feb. 2016. <http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/>.

[5]"OpenCV: Geometric Transformations of Images." *OpenCV 3.0.0-dev Documentation*. N.p., n.d. Web. 29 Feb. 2016. <http://docs.opencv.org/master/da/d6e/tutorial_py_geometric_transformations.html>.

[6]"Finding Contours in Your Image¶." *OpenCV 2.4.12.0 Documentation*. N.p., n.d. Web. 29 Feb. 2016. <http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/find_contours/find_contours.html>.

[7]"MAVProxy¶." *MAVProxy — MAVProxy 1.4.38 Documentation*. N.p., n.d. Web. 29 Feb. 2016. <http://dronecode.github.io/MAVProxy/html/index.html>.

[8]"Communicating with Odroid via MAVLink." Ardupilot, n.d. Web. 29 Feb. 2016. <http://dev.ardupilot.com/wiki/odroid-via-mavlink/>.

[9]"DroneKit Tutorial." Ardupilot, n.d. Web. 29 Feb. 2016. <http://dev.ardupilot.com/wiki/droneapi-tutorial/>.

[10]Mackay, Randy. "Ardupilot-balloon-finder." *GitHub*. N.p., n.d. Web. 29 Feb. 2016. <https://github.com/rmackay9/ardupilot-balloon-finder>.