Otto: The Personal Cameraman

Seth Kielbasa, CSE, Albion Lici, CSE, Noah Portnoy, CSE, and Andrew Sousa, EE

***Abstract*—Otto is the personal cameraman that introduces a new way to capture your life's most exciting moments. The system is an autonomous quadcopter that is designed to follow and record a user performing an individual action sport. By maintaining a visual lock on the user during his or her performance, Otto is able to capture the entire experience through an on-board high resolution video camera. Once finished, the user can gather video recordings from the drone and share them with loved ones.**

I. Introduction

HUMANS are social beings that yearn to share their experiences with family and friends. Some of the most exciting experiences to share are those that include extreme circumstances. Individual action sports, such as skiing, wake-boarding, mountain biking, and skateboarding are activities that people truly enjoy capturing and sharing. Filming these moments is extremely hard since the sports are done at high velocities or in relatively dangerous conditions. We have set out to eliminate this challenge from the lives of amateur extreme sports performers.

Previously, this challenge has been addressed by various subpar solutions. Some performers hold a camera [1] while doing their action sport in order to capture the moment. This solution is dangerous because the performer is putting some of their focus on filming and not necessarily on their own actions. Another option for the performer to attain a recording of himself is to have someone ride or otherwise move alongside him and record him [1]. This solution is even more unsafe because it requires the cameraperson to ride alongside the performer at a presumably high rate of speed and focus almost entirely on the recording. There are also several products that have the same goal of recording action sports performers. These solutions attempt to track their users with GPS localization; our solution is different in its implementation and, we expect, its operation. Furthermore, these products, such as AirDog and HEXO+, are in the development stages and are not yet on the market. Thus, action sports performers, of which there are 4.88 million in the United States [2], have not yet found the appropriate solution for capturing all of the amazing things they do.

Otto is the personal cameraman for capturing and recording amazing third-person aerial images. This product will make the recording process safer for action sports performers and enable them to capture a unique, aerial view of their performances. Additionally, it will deliver functionality that has never been feasible for everyday consumers. This technology could have applications in many fields, including medicine, military, and home security. Along with these positive alternatives, there are negatives as well. Criminals could use this technology to follow people, enabling them to stalk others in an obscure way. We attempt to eliminate this in our requirements through the use of the user's smartphone and the wearable jersey.

In order to deliver this ambitious prototype, we have developed requirements and constraints to keep our project within the proper scope. More importantly, we want to ensure that during the recording, the user can pay no attention to Otto and focus entirely on his or her performance. From this, we concluded that the system must initiate and maintain a visual lock on the user throughout the entire performance. We also require that the system must have safety features to minimize the possibility of injury to the user when operating Otto; the system must have a safety lock on both the smartphone application and the quadcopter itself to prevent undesired liftoff. A full list of our requirements can be found in Table IV of the Appendix. Along with these requirements are quantitative system specifications that Otto will need to abide by; these can be found in Table I.

TABLE I
System Specifications

Specification	Value
Maximum drone/user separation	30 meters
Minimum drone/user separation	5 meters
Average flight time	10 minutes
Maximum speed of drone	30 mph
Maximum speed of user permitted for tracking	30 mph
Maximum angular velocity of drone	1.8 rad/sec
Minimum quality of video recording	720p at 30 fps
Total mass	< 1.5 kg
Lift at 50% throttle	> 1.5 kg

II. Design

A. Overview

Otto introduces a new way for amateur extreme sports performers to film themselves. Otto is a drone, more specifically an autonomous quadcopter, that will track and record a user. To do this, we needed to start with a robust

S. K. Author from Westfield, MA (e-mail: skielbas@umass.edu).
A. L. Author, from Revere, MA (e-mail: alici@umass.edu).
N. P. Author from Newton, MA (e-mail: nportnoy@umass.edu).
A. S. Author from Dartmouth, MA (e-mail: acsousa@umass.edu).

foundation; details about the drone system's hardware are covered in Section B. The tracking software and hardware will physically reside on this drone platform. The drone will hover and fly using customized flight control software; more on that in Section C.

The tracking functionality relies on the user wearing a uniquely colored jersey and carrying a smartphone. The user will open the app and within the app be able to specify a drone-user separation distance, turn Otto on, start the recording, and press the takeoff button. Once the takeoff button is pressed, the drone will commence liftoff and rise to the appropriate drone-user separation distance. At the same time, the FollowMe feature starts up; this will enable the tracking and following of a user. When FollowMe initiates, it will attempt to acquire a visual lock on the user. Once this is achieved, the user will receive a notification through the app that Otto is ready for action.

The FollowMe feature works by leveraging two major technologies: GPS and image processing. The first component of FollowMe is GPS tracking, described in Section D, and it will follow the user based on the GPS coordinates of the user relative to those of the drone. The second component is camera tracking (detailed in Section E), which uses a low-resolution camera to align the drone with the user. The camera tracking is able to find the user in the frame because he/she is wearing a specially colored jersey. The outputs from these two components are then combined through a sensor fusion technique. Once fused, the FollowMe feature will calculate actions for the drone to make based on the fused data and output the proper controls to the flight control board in order to achieve those actions. The FollowMe feature is visually represented in the Otto Block Diagram; Figure 1.

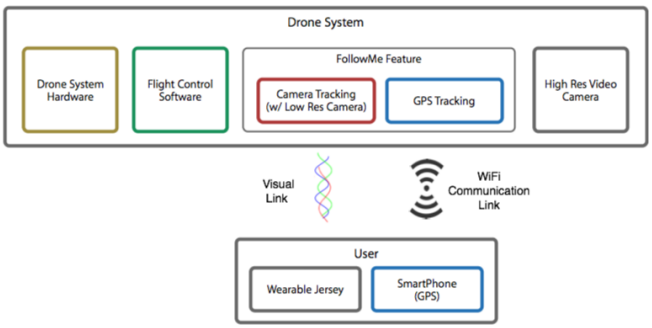

Fig. 1. A high-level view of Otto's main components.

B. *System Hardware*

First, we introduce the system's hardware. The airframe of the quadcopter is the DJI Flame Wheel F450, which is made of a hardened plastic material to ensure a rigid flight with enough strength to endure high impact landings [3]. To provide thrust for this frame, we have four SunnySky brushless DC motors which have a 980 kV rating; this means that for every volt applied, the motor will attempt to produce 980 revolutions per minute (RPM) [4]. The motors rotate 10x4.7 inch carbon fiber composite propellers. This motor-prop combination ensures that there is enough upward force to allow the drone to hover at approximately 50% throttle, under the assumption that the drone's total mass is 1.5 kg. This is done so that at 80% throttle, the motors can quickly maneuver the drone. These motors are operated by a three-phase signal; the system generates this locally with electronic speed controllers, or ESCs.

We chose four ESCs with the SimonK firmware loaded on them, as it seems to be the most promising in the industry [5]. The ESCs take in a pulse-width modulated (PWM) signal from the flight control board and turn it into a polyphase signal to rotate the 14-pole DC motors. The flight control board is the APM 2.6 board manufactured by 3D Robotics. It contains all of the necessary hardware for flying: a 3-axis gyroscope, an accelerometer, and a barometer.

The flight control board will receive movement commands from the FollowMe feature in the form of pitch, roll, and yaw commands. For the GPS tracking component of the FollowMe feature, there is a local GPS module on the drone: the u-blox LEA-6H GPS module with a Taoglas patch antenna [6]. This module collects GPS data and sends it to the FollowMe feature's computation platform, a Raspberry Pi Model B+ [7]. The GPS tracking also uses a 2.4 GHz Wi-Fi module which is attached to the Raspberry Pi through a USB port. More about the operation of GPS tracking can be found in Sections D and E. For the camera tracking component of the FollowMe feature, the Raspberry Pi will also be attached to a Logitech C310 USB webcam [8].

The drone is powered by a 5200 milliampere-hour (mAh) lithium-ion polymer (LiPo) battery [9]. This battery will ensure that the drone can meet the ten minute flight time requirement, as we have estimated that this battery will provide 11.4 minutes of flight time (see "Mixed Flight Time" in Figure 4 of the Appendix for calculated results). The power is distributed to the four on-board ESCs, each of which then converting that power, as well as the input from the flight control board, into three-phase power signals to its respective DC motor. Each of the ESCs also has a battery eliminator circuit (BEC) which outputs 5V 2A DC; this supplies power to the flight control board and the Raspberry Pi. All of the peripherals attached to the Raspberry Pi are then powered by the Raspberry Pi itself.

Finally, there is a high-resolution video camera on-board the drone: the GoPro Hero camera. This device has an internal battery as an independent power source as well as its own data storage. The GoPro camera will be used to capture the high-resolution video recording of the user.

C. *Flight Control Software*

The flight control software resides on the flight control board and maintains constant knowledge of the orientation of the drone. Using readings from on-board gyroscopes and accelerometers, the system deploys a set of algorithms to determine appropriate values to feed the four individual ESCs, which then update the motors. The software currently in use was based on that provided by [10]. Modifications have been

made to add functionality for our specific goals, introduce safety features, and increase flight stability. Currently, the flight control software is functional with test flights being successfully conducted under manual RC control.

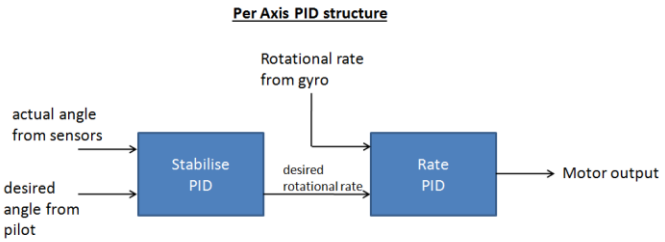

Fig. 2. A block diagram representation of the Flight Control Software. [10]

The algorithm centered at the heart of the flight control software is PID. Figure 2 provides a visual representation of the software handling incoming data and converting it to values capable of driving the motors [10]. Each PID block takes in a desired value that is compared to the actual values captured from the sensor chip. The software contains an array of adjustable PID constants used to weight the determined error, resulting in a weighted value being fed to the motors [10].

This crucial subsystem is being developed for the 3D Robotics APM 2.6 flight control board [11]. This Arduino-based microcontroller contains an MPU-6000 sensor chip that features three pairs of gyroscopes and accelerometers, one pair per axis. An Integrated Development Environment (IDE) tailored specifically for the APM board has been the site of all flight-related software design and testing.

Completion of this subsystem required knowledge obtained through the numerous Electrical and Computer Engineering courses that we have completed over the past three years. ECE 353 Computer Systems Lab 1 gave an initial exposure to the C programming language, which is syntactically and functionally very similar to the Arduino microcontroller language used in this project. Additionally, ECE 373 Software Intensive Engineering provided guidance on how to successfully plan the composition of a program. An elective junior-year ECE design project on firefighting robots provided foundational knowledge in robotics engineering and provided valuable experience utilizing sensors and motors with a microcontroller.

For obvious safety reasons, numerous tests are conducted on each new revision of the flight control software before being used with live motors. The first round of testing involved simple scripts to print sensor readings and motor outputs to the console for analysis. Through this technique, it could be confirmed that the sensors were reading appropriate values and that the software was providing reasonable output to the motors. From there, a testing rig was built to allow the drone, now with the attached motors and propellers, to have free motion over one axis. This allowed for visual confirmation that the drone could respond to the manual RC control of either pitch or roll and then stabilize itself with minimal oscillation. With further use of the testing rig, the PID constants mentioned above were tweaked to reduce the oscillation effects. Once the PID control was sufficiently refined, outdoor tests were conducted at low throttle levels before actual takeoff and flight was achieved. While airborne, effects of spinning or oscillation of the drone frame could be observed, and the software was adjusted to counteract these negative motions.

D. GPS Tracking

As one of two fundamental components of the FollowMe feature, GPS tracking will allow Otto to track the user performing his sport. This will be achieved by first acquiring the GPS coordinates of both Otto and the user, and then providing those coordinates as input to the overall FollowMe feature. The GPS tracking application will run on the drone's Raspberry Pi, and it will receive coordinates at a rate of 2 Hz from both the Android application and the GPS module on-board the drone, the u-blox LEA-6H GPS module [6]. The GPS tracking application and the Android application will be communicating with each other over Wi-Fi using the UDP protocol. In order to achieve this communication between the two applications, both applications must be multithreaded. This can give rise to several potential issues, including the lack of effective communication between threads and unresolved data dependencies within each application. To resolve such complications, mutexes (mutual exclusions) are implemented to ensure that shared resources are used in an orderly manner by all threads. Threaded programming and mutexes were introduced in ECE 373 Software Engineering. In addition, Wi-Fi technology and the Internet protocol stack are used in the GPS tracking software package to allow the two applications to exchange data; these networking topics were presented in ECE 374 Computer Networks and the Internet. The Raspberry Pi application is written in the C++ programming language and it utilizes the POSIX Threads (pthreads) library as well as Linux API system calls. The Android application was developed using the Java programming language and the Android API, which is provided and documented by Google. An Android extension tool package was also used alongside the Eclipse IDE in order to make the development of the app more efficient.

The user's coordinates will come from an Android smartphone that will be running a custom-developed application (henceforth referred to as the "app"). The smartphone must be on the user's person for GPS tracking to work properly. The tasks of the app have been broken down into three threads executed in parallel: the user interface (UI) thread, the network send thread, and the network receive thread. Any rendering of visual objects and UI updates are performed by the UI thread. The UI thread is also responsible for spawning all other threads for this application; it can therefore be thought of as the main thread [12]. The network send thread handles all outgoing network data intended for the GPS tracking application on the Raspberry Pi by transmitting messages that have been added to the transmit FIFO queue. The network receive thread handles all incoming traffic from the Raspberry Pi to the Android app. Receiving from the

network is a synchronous, or blocking, task.

The drone's coordinates are calculated by the on-board u-blox LEAH-6 GPS module. The GPS tracking application, running on the Raspberry Pi on-board the drone, is tasked with collecting GPS data from both the drone and the user, and then performing calculations on that set of data. The output of this application will then be fed into the higher-level FollowMe feature. The GPS tracking application is broken down in the same fashion as the Android app with the addition of an I2C thread. This means there is a network send, a network receive, and a main thread, each performing a similar task as that of the Android app. The main thread is responsible for spawning all other threads and performing calculations on GPS data. The network send thread is responsible for transmitting all data in the FIFO send queue. The network receive thread is responsible for receiving data from the Android app. Finally, the I2C thread requests the coordinates from the flight control board, which is connected to the u-blox GPS module via a UART connection. All future I2C communications will be executed on this thread.

The first four ASCII characters of a message packet in the GPS tracking software package are reserved for the payload identifiers for latitude and longitude (see Table II). All subsequent characters in the message are interpreted as the payload, which consists of two GPS coordinates identifying the location of either Otto or the user. Payload size is not specified in the packet. Because we use the UDP protocol, the communication on the Wi-Fi channel is inherently unreliable. This means that 2 Hz is not guaranteed at all times because communication packets containing GPS coordinates might get lost due to collisions, dropped because of a full queue, or other factors. Attempting retransmission will provide lagging coordinates as an input; therefore we ignore any dropped messages. The GPS tracking system is designed to be robust, and so does not halt when packets are dropped. However, the system will see a decrease in performance in this scenario. Under the current implementation of the messaging interface for the GPS tracking software package, only two payload identifiers are supported: $LAT and $LON. See Table II for an explanation.

TABLE II
PROTOCOL FOR ANDROID/RASPBERRY PI WI-FI COMMUNICATION

Payload Identifier (four bytes)	Description
$LAT	Latitudinal coordinate of Otto or the user
$LON	Longitudinal coordinate of Otto or the user

E. Camera Tracking

The camera tracking system serves as the second of two core components of Otto's FollowMe feature, allowing the drone to accurately follow the user and keep the user in the video frame. In order for the camera tracking system to uniquely identify and track the user in the environment, the user must wear a distinctly colored jersey. The camera tracking system will attain a visual lock on the user upon drone takeoff by scanning for the color of the user's jersey. Once the visual lock has been acquired and the FollowMe feature has been initiated by the user through the use of the Android application, the camera system will continuously track the user. If the user begins to veer out of the video frame boundaries, the camera tracking system will send yaw control output in vector form to the FollowMe feature. This output communicates how the drone should reorient itself along the yaw axis to maintain a visual of the user. The FollowMe feature will take this data from the camera tracking system and synthesize it with GPS tracking data to form a unified output to the flight control board, instructing the flight control software how to maneuver the drone. As of the team's Midway Design Review, camera tracking is a functional, closed-loop system operating at 7.5 Hz that can track an object of a certain color and keep the object in the video frame by commanding a servo motor to rotate the camera along the yaw axis.

The camera tracking software lives on Otto's main computer, a Raspberry Pi Model B+ [7]. The software is written in the Python programming language and harnesses the OpenCV (Open Source Computer Vision) library to detect the colored object by heavily processing the video frames that are captured by a Logitech C310 USB webcam [8]. Specifically, the camera tracking software performs the following OpenCV transformations [13] on each video frame: cvtColor to convert the image from the RGB to the HSV color space, inRange to get a mask [14] of the image consisting only of those pixels that fall within the desired object's color range, dilate to dilate the shapes present in the mask so as to smooth out the shapes' edges, findContours to detect all shapes present in the image, and contourArea to measure the area of the detected shapes and to select the most prominent shape in the image. From here, the software determines the center of the detected object within the video frame, and continuously checks to see if the center of the object moves outside a programmed set of bounds centered about the middle of the frame. Upon detecting that the object has moved outside of these bounds, the software outputs servo commands for the yaw axis over a serial output. The serial output then goes to an Arduino Uno, which sends digital output to the servo. [Note that the Arduino IDE and programming language were used to develop the servo-controlling software that runs on the Arduino Uno.] Once the object moves back within the specified bounds, the software no longer sends servo commands. This simulates how the camera tracking software will send yaw commands to the FollowMe feature to keep the user in the video frame.

Several tests have been conducted to assess the performance of the camera tracking system. The environment for these tests used flourescent lighting, thus having a color temperature of approximately 3000 K [15], and the camera was positioned to face a white background that filled the entire camera frame. Any object displayed to the camera was always kept approximately two feet away. For these tests, the system was configured to track a red object. The variables in the tests are as follows: (1) the color of the object, and (2) either moving the camera to simulate the effects of drone movement, or

keeping the camera stationary. For each test, we monitored the system's performance for two minutes in the same environmental conditions. Detection rates and false positive rates were calculated using frame counts provided by the camera tracking system. The results are summarized in Table III.

TABLE III
CAMERA TRACKING SYSTEM TESTS

Test Scenario	Detection (True Positive) Rate	False Positive Rate
Moving red object always in frame, stationary camera	98.7%	0%
Moving red object always in frame, moving camera	100%	0%
No object in frame, stationary camera	N/A	0%
No object in frame, moving camera	N/A	0%
Blue object always in frame	N/A	0%
Green object always in frame	N/A	0%
Yellow object always in frame	N/A	0%

While there are many factors contributing to the performance of the camera tracking system in these tests, we can conclude that the system is capable of tracking a red object under certain conditions with a high degree of accuracy. We can attain similar performance in tracking an object of another color by simply changing the HSV range in which to look for an object. At this point, it is unknown the system's performance outdoors with the user wearing a jersey and the camera a distance away from the user that meets the system specifications. Color temperatures outdoors range from 5500 K to 6500 K [15], which would affect the camera's perception of color; this can be accounted for by modifying the HSV range specified in the program. However, a potential area to be addressed is the presence of objects in the environment that are similar both in color and size relative to the user's jersey.

The development of the camera tracking system relied heavily on the material covered in several Electrical and Computer Systems Engineering courses. ECE 353 Computer Systems Lab 1 and ECE 373 Software Intensive Engineering together provided a deep and fundamental understanding of software that allowed for the creation of the camera tracking system. ECE 354 Computer Systems Lab 2 introduced image processing and manipulation techniques that were formative in the design of the camera tracking system.

III. PROJECT MANAGEMENT

As of our Midway Design Review, we have completed all of our proposed deliverables, which are summarized in Table IV. Through this, we have completed the four main subsystems of Otto. In the remaining five months comes the most challenging part of our project: integrating the subsystems to create a functional prototype. As a team, we must complete two main tasks: develop autonomous flight capabilities for the drone, and fuse the data from the GPS and camera tracking systems to output reliable pitch, yaw, and roll instructions to the flight control board.

TABLE IV
MDR DELIVERABLES

MDR Goal	Completion
Drone Hardware	100%
Flight Stabilization Algorithm	100%
GPS Tracking	100%
Camera Tracking	100%

Team Otto is comprised of three computer systems engineers (CSEs) and one electrical engineer (EE) who have diverse backgrounds. Seth Kielbasa has worked with robotics in the past as part of the UMass Amherst firefighting robot team; as part of Team Otto, he is responsible for the flight control and stabilization algorithms. Albion Lici has completed multiple internships at Teradyne, where he gained much insight into interfaces between hardware and software; this knowledge has proven to be very useful. Albion is responsible for the GPS tracking component of the FollowMe feature. Noah Portnoy also has robotics experience as he led the UMass Amherst firefighting robot team for two years; he is responsible for the camera tracking component of the FollowMe feature. Andrew Sousa is the EE of the team and he brings robotics experience from his work leading the IEEE Micromouse group to successful completion of an autonomous robot. Andrew is the team manager and is responsible for all of the systems hardware. Please see Figure 3 to learn more about our project timeline.

The team has been working quite well together; we all have a clear vision of the final product. There is always an enlightening discussion at our weekly all-hands meetings with our advisor, Professor Christopher V. Hollot, who continues to ask us thought-provoking questions and helps steer the team in the right direction. The team communicates daily, either through an online messaging service or in person, and meets weekly on Thursday evenings to discuss each individual's progress from the week. We also share important information such as data sheets, calculations, and experiment results on cloud-based storage hosted by Google, while we keep all of our code under version control in a GitHub repository. Finally, we conduct large-scale or high-level communication via email. We are in nearly constant contact due to both our project's difficulty level and our commitment to deliver Otto.

IV. CONCLUSION

Following the completion of the Midway Design Review, the team has all four subsystems successfully working independently of each other. Integration of the drone system hardware and flight control software has already begun, with successful test flights demonstrating stable flight and responsive control over both pitch and roll. The third axis, yaw, has proved a challenge to control. However, tests have been conducted with the on-board compass to combat the inaccurate yaw sensor on the flight control board. Stable readings have been observed from the compass; thus, user control over the yaw axis is now fully reliant on compass readings. The camera tracking component of the FollowMe

feature is currently operating on the Raspberry Pi, tracking colored objects in a simulated environment. The second FollowMe component, GPS tracking, is running on both the Raspberry Pi and an Android smartphone, and it is accurately transmitting coordinates and determining separation distances.

Over the course of the next few months, the team will see the development of the next crucial component of Otto: the fusion of the GPS and camera tracking into a single FollowMe feature. The success of this system is imperative to the completion of the project, as it responsible for user tracking and commanding the autonomous flight feature; both of these functions are essential to making Otto possible. The fused FollowMe feature will be developed in parallel with significant testing and simulation of the flight control software to ensure a smooth transition from the manual RC controller to the autonomous flight algorithm.

Potential areas of struggle include implementing the fusion of data from the camera and GPS tracking subsystems into a single flight command, and fixating a camera on a moving user while maintaining appropriate operating distances. Presently, the team is considering an algorithm based on a weighted sum of values from the two tracking components, where the weighting is a function of separation distance; GPS being more heavily weighted when the user is far away from the drone, and camera tracking being more heavily weighted in close-distance situations in which the GPS would become too inaccurate. From here, all flight movement calculations will be done on the Raspberry Pi before transmitting them to the flight control board for execution. To reliably record the user, the current implementation under consideration is a camera with fixed orientation on the drone. This requires that the FollowMe feature adjust the drone's orientation in such a way that the user will be maintained in the camera's field of view. The issue with this implementation resides with the vertical axis. In order to vertically reorient the camera to keep the user in the video frame, the drone must make a pitch adjustment, which would result in either forward or backward movement.

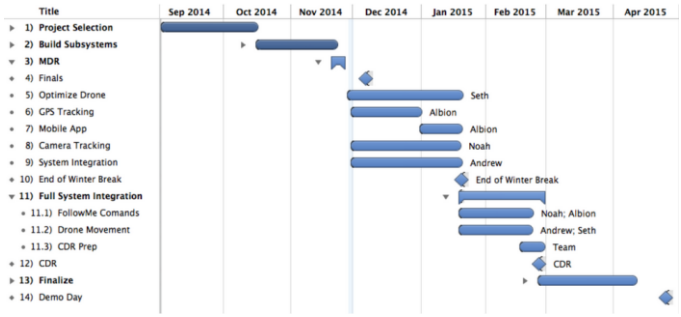

Fig. 3. Project timeline from Fall 2014 to Spring 2015.

The planned deliverable for CDR will be a fully functional prototype of Otto, incorporating the autonomous flight, FollowMe feature, and video recording. Figure 3 shows the expected timeline for the coming months. From this point until middle to late January, the team will be split into two essential sub-teams. The first team will be charged with perfecting the drone flight and developing the autonomous flight software. The second team will be responsible for the required fusion of data from the camera and GPS tracking. The time period from late January to CDR will be spent merging these components into an integrated prototype. During this time, the video recording will be developed and tested once the drone has successfully demonstrated the rest of its functionality.

APPENDIX

TABLE IV
SYSTEM REQUIREMENTS

	Requirement
1	Track user through a fusion of two sensors: GPS and camera
2	Collect GPS location of user through a Wi-Fi connection to user device
3	Collect finer location data of user through camera tracking
4	Carry out user-defined takeoff and land commands
5	Maintain a user-defined drone/user separation distance
6	Allow user to start and stop video recording
7	Video recording is high-definition (720p or better)
8	Must maintain visual lock on user for duration of recording
9	Drone will take preliminary measures upon reaching critical battery level
10	Safety lock in hardware and software

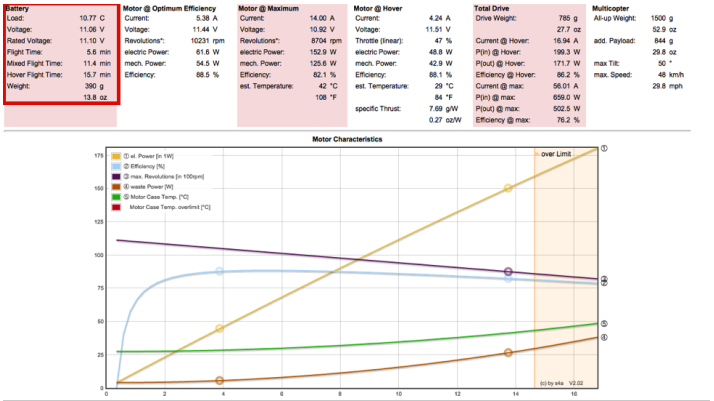

Fig. 4. Estimated performance of the drone based on its motors, propellers, battery, and drone weight. Battery performance estimates, including flight times, are highlighted with a box. "Mixed Flight Time" is a combination of hovering and maneuvering, the latter of which requires more thrust. Calculations provided by [16].

ACKNOWLEDGMENT

Our team would like to thank our advisor Professor Christopher V. Hollot for his excellent advising methodologies and Francis M. Caron for helping us with all requests related to the Senior Design Project lab.

REFERENCES

[1] B. Rose. (Accessed 2013, February 14). *How to Get Better Action Cam Footage* [Online]. Available: http://gizmodo.com/5983584/getting-better-action-camera-footage

[2] "Number of people who are very interested in extreme/ action sports in the United States (USA) from spring 2008 to spring 2014," Statista, New York, NY, Stat. Rep., Spring 2014 [Online]. Available: http://www.statista.com/statistics/229006/people-who-are-very-interested-in-action-sports-usa/

[3] DJI Innovations. (Accessed 2014, December 15). *Flamewheel ARF Kit* [Online]. Available: http://www.dji.com/product/flame-wheel-arf/feature
[4] P. Pine. (Accessed 2014, December 15). *What does KV mean?* [Online]. Available: http://www.flyelectric.com/ans.kv.html
[5] (Accessed 2014, December 15). *SimonK ESC User Guide* [Online]. Available: http://www.robotshop.com/media/files/pdf/lynxmotion-simonk-esc-guide.pdf
[6] (Accessed 2014, December 15). *3DR uBlox GPS with Compass Kit* [Online]. Available: http://store.3drobotics.com/products/3dr-gps-ublox-with-compass
[7] (Accessed 2014, December 15). *Model B+* [Online]. Available: http://www.raspberrypi.org/products/model-b-plus/
[8] (Accessed 2014, December 15). *Logitech HD Webcam C310* [Online]. Available: http://www.logitech.com/en-us/product/hd-webcam-c310
[9] (Accessed 2014, December 15). *Lumenier 5200mAh 3s 35c Lipo Battery* [Online]. Available: http://www.getfpv.com/lumenier-5200mah-3s-35c-lipo-battery.html
[10] G. Owen. (Accessed 2014, December 15). *How to Build Your Own Quadcopter Autopilot / Flight Controller* [Online]. Available: https://ghowen.me/build-your-own-quadcopter-autopilot/
[11] (Accessed 2014, December 15). *APM 2.6 Set* [Online]. Available: http://store.3drobotics.com/products/apm-2-6-kit-1
[12] (Accessed 2014, December 15). *Processes and Threads* [Online]. Available: http://developer.android.com/guide/components/processes-and-threads.html
[13] (2014, April 21). *OpenCV 2.4.9 Documentation* [Online]. Available: http://docs.opencv.org/
[14] (2014, March 26). *Mask (computing): Image masks* [Online]. Available: http://en.wikipedia.org/wiki/Mask_%28computing%29#Image_masks
[15] (Accessed 2014, December 15). *Color Temperature* [Online]. Available: http://en.wikipedia.org/wiki/Color_temperature
[16] (Accessed 2014, December 15). *xcopterCalc - Multicopter Calculator* [Online]. Available: http://www.ecalc.ch/xcoptercalc.php