ASPECTS

Chris Boselli, EE, Jason Danis, EE, Sandra McQueen, EE, Alex Breger, EE

***Abstract*—The increased popularity of unmanned aircraft vehicles (UAVs) has expanded the possibilities for innovation with aerial photography, package delivery services, and other exciting applications. However, many new users are inexperienced and unaware of the guidelines and regulations in place to ensure safety and prevent potentially disastrous collisions. The ASPECTS module – a Raspberry Pi computer connected to a communication device – mounts on the UAV and interfaces with the flight controller to restrict flight in FAA-designated No-Fly zones.**

I. Introduction

RECREATIONAL drone use has increased dramatically in recent years as the hobby becomes more popular. The number of Certificates of Allowance permitting the flight of UAVs in civil airspace grew from 146 in 2009 to 545 in 2013 [2]. These numbers are of course in addition to the recreational drones available for public purchase. Furthermore, according to a report by USA Today in October 2015, "An examination of 891 drone sightings reported to the Federal Aviation Administration over a 17-month period found more than half flew too close to an airport" [1]. On top of these sightings which clearly present a very real danger to the safety of aircraft leaving and entering airports, there have been a variety of instances that nearly had catastrophic consequences. According to BBC news on April 17th, 2016, "The British Airways flight from Geneva, with 132 passengers and five crew on board, was hit as it approached the London airport at 12:50 BST on Sunday. No debris has been found and police have asked for anyone who finds drone parts in the Richmond area to come forward...After safely landing the plane, the pilot reported the object had struck the front of the Airbus A320" [3].

UAV interference with passenger planes could result in property damage and, in some cases, fatality. Therefore, with this trend comes a new challenge: defining regulations that will lower the risk of incursions with passenger and commercial planes, and ensuring safe practices among vehicles sharing the airspace. One way this is implemented is by establishing restricted areas where drones cannot legally fly. The Federal Aviation Administration (FAA) currently enforces a five mile No-Fly Zone around all major airports, military bases, national parks, and other sensitive landmarks in the U.S. [2], but very often those perimeters are breached by unmanned aircraft. This poses a threat to safety and security.

This issue has gained the attention of the FAA, the United States Department of Transportation, and several key Senate members, who recently proposed that all consumer drones must feature *geofencing* capabilities in the future. Geofencing is a way of creating virtual geographical boundaries which are defined by a central GPS coordinate with a specified radius around that point. The idea of geofencing expands on the basis for other existing technologies such as invisible fences for dogs. This concept can be paired with specified hardware and software components to simulate a literal "fence." This technology requires innovation by manufacturers, but also presents an obstacle for hobbyists who build their own copters; our design will make geofencing technology available to all UAV owners.

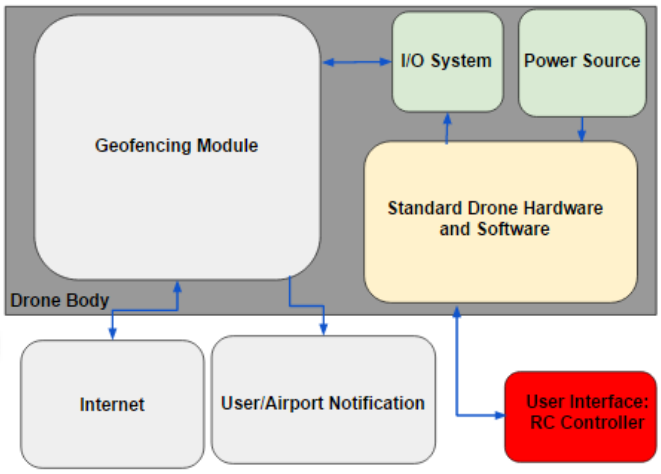

Fig. 1. High-level block diagram of ASPECTS system.

II. Design

A. Overview

ASPECTS (Airport Safety Perimeter Control System) is a proposed universal geofencing solution which may be incorporated on future drone models or retrofitted on previous designs. The on-board unit consists of a Raspberry Pi computer, a GPS chip, and a 3G communication module, all of which will interface with the existing flight controller to execute commands, as seen in the high-level block diagram in Figure 1. (Figure 6 on page 6 provides a more detailed diagram of the system.)

By constantly monitoring the location of the drone via satellite, ASPECTS will determine the proximity to local No-Fly Zones through a software algorithm on the Raspberry Pi. Using the on-board 3G communication module and SIM card, ASPECTS will send a message warning the user when he or she is within a predefined buffer zone around the No-Fly Zone (indicated by the blue boundary in Figure 2 below) allowing an

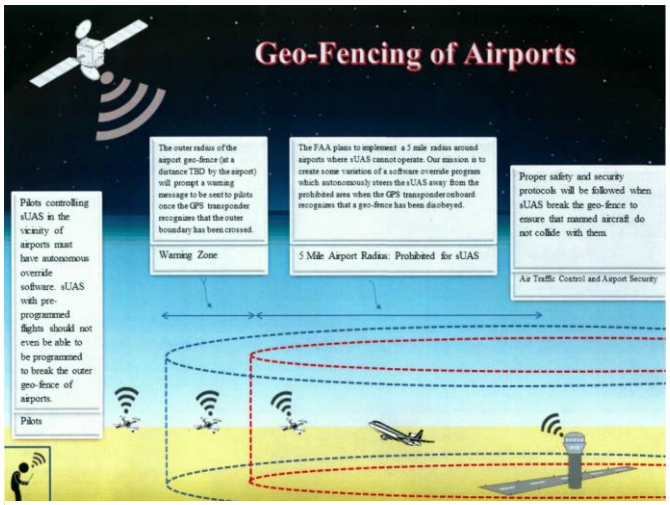

Fig. 2. A visual representation of our Geofencing solution. The blue outer perimeter represents the buffer zone surrounding the critical No-Fly zone denoted by the red inner perimeter.

opportunity for the user to manually redirect the flight path.

If the user does not take corrective action before the drone reaches the No-Fly Zone, the on-board controller with then assume control of the drone in order to prevent a breach of the critical airspace (indicated by the red boundary in Figure 2 below) by landing the aircraft. This effectively creates a physical barrier around the perimeter of the airport or other sensitive area and averts a potentially dangerous situation.

B. Drone Body

One of the earliest changes made to our original vision for this project was the decision to build our own drone "from scratch" rather than simply purchasing a fully integrated drone. A major factor in this decision was to keep all associated hardware and software components of the project as open source and compatible as possible. In disassembling a 3DR Iris+ quadcopter, we soon realized that we would likely need special permissions from the manufacturer in order to be able to hack into and alter the flight software to execute our algorithm once inside a geofenced area. To avoid additional troubles, we carefully researched all of the major components required to build a sufficient quadcopter of our own.

Part	Description	Weight (g)	Qty	Total Weight Contribution (g)
Quadcopter Frame	Tarot 650 Carbon Fiber Frame	485	1	485
Flight Controller	Pixhawk	100	1	100
Battery	nano-tech 4000mA	333	2	333
Propellers	3DR	N/A	3	0
Motors	3DR Black Top Motor	68	4	272
ESCs	3DR 20 A	21	4	84
Our on-board controller	Raspberry Pi Model B	45	1	45
3G/GPS Communication	Adafruit FONA 3G Breakout	9	1	9
TOTAL WEIGHT				**1328 g**

Table 1. On-board hardware and corresponding weight contribution to the design.

One of the major aspects that needed to be considered when designing our own drone was the overall weight of the quadcopter and additional on-board hardware which we were going to install as part of our design. This key factor ultimately finalized decisions between alternative hardware possibilities when purchasing the parts for the drone. We were able to stay within our target limit of 1.5 kg for total weight of the quadcopter parts and additional on-board hardware components. The final parts list with the corresponding weight of each component can be seen in Table 1.

Another key aspect of the design process was selecting which battery to purchase that would power all of the on-board components of the drone. We had to take into consideration the power consumption of the drone hardware as well as our additional components, and we had to make some calculations in order to be sure that the battery provided enough power for significant flight time without compromising the balance between increased power and added weight.

C. Controller (Raspberry Pi)

In order to interpret GPS data, run code for overriding the drone's microcontroller, and run code for landing the drone at a geofence boundary, an on-board controller was needed. We use a Raspberry Pi Model B to implement this controller unit, which will be housed on-board the drone during flight. Upon startup the unit connects to a database, located within a separate subsystem, containing the central coordinates of every airport in the United States. The controller determines which central airport coordinates are located within ten miles of the drone's initial position, and a five mile critical radius as well as an additional one mile buffer region is established around each point.

Coordinates that are read into the controller are classified into three regions: (1) Clear of all restricted areas, (2) inside of a buffer region near a geofenced area in which the user and owner of the restricted space will be notified of the drone's presence, and (3) inside of a geofenced area. Distance from the central geofence coordinates are determined using the Pythagorean theorem assuming the difference in latitude is the y-axis distance and the difference in longitude is the x-axis distance. A visualization of this algorithm is shown in Figure 3. After it is determined that the drone has breached a restricted area, the Raspberry Pi communicates with the Pixhawk microcontroller, which is the flight controller installed on the drone, to override its software and safely land the drone. The software to land the drone is executed after it has been confirmed that the Raspberry Pi has priority over the Pixhawk. Upon landing, the drone disarms itself and is incapable of being rearmed until the user takes it outside of the restricted space and rearms the system via reboot.

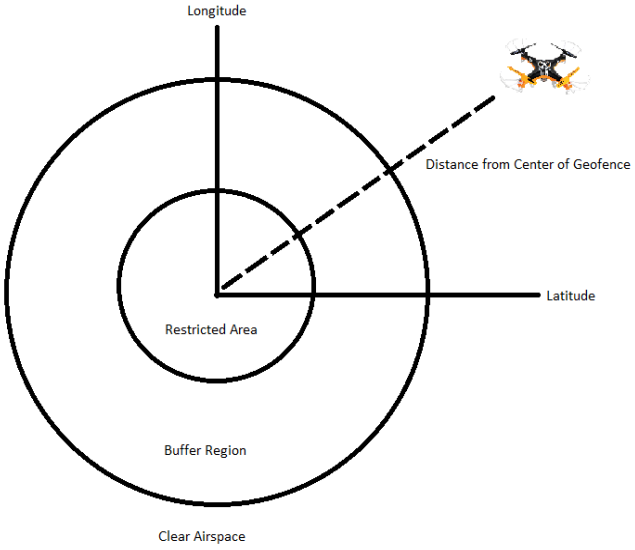

Fig 3. Breakdown of regions surrounding a geofenced area

For this project we set up a temporary testing site to assess the overall system performance. We assigned a coordinate to an open field on the university campus and set LEDs as a visual indication of where the drone was relative to the buffer "warning" zone and the no fly zone. Preliminary tests involved simply walking through the area to ensure that the module recognized its current location. After the autopilot commands were proven to function smoothly, we were able to execute full test flights successfully. It is also worthy of note that this module will work with any flight controller featuring a telemetry input port, ensuring compatibility with about 75% of publicly available drones.

D. Communication Device

The geofencing module employs 3G communication with the user via the cellular network. The Adafruit FONA breakout modem, designed for compatibility with the Raspberry Pi, provides this 3G capability and was selected for its efficiency and compact size. Data is sent to and received from the Raspberry Pi through the hard-wired transmit and receive pins. The various 3G functions of the FONA are accessed using a library of built-in "AT" commands provided in the datasheet of the SIM5320 chip built onto the FONA board [8].

Utilizing existing communication technology, such as T-Mobile's cell towers, is critical to the ASPECTS design because it minimizes overall cost of the system and facilitates implementation. Upon registration of their drones, users can provide a cell phone number at which they can be contacted for alerts. If the UAV approaches the No-Fly zone it will encounter the geofence coordinates defined in the controller software. The FONA will then receive a command from the Raspberry Pi's transmission (Tx) pin, which contains the target cell phone number of the user along with a text message warning the user of his or her proximity to the airport, instructing him or her to turn around. In order to allow the user sufficient time to read the message and react, we estimate that the text must be received between 10 and 20 seconds before the drone reaches the critical 5-mile radius. Since this requires fast and reliable communication, we chose to use the 3G FONA in favor of the previous 2G model. The maximum latency observed was 8 seconds from the time the message was sent to when it was received by a mobile device. From this delay, the radius (B) of the buffer zone added on to the initial five miles could be calculated based on the maximum speed of the average UAV:

$$B = \text{avg. max speed } (m/s) * [\text{reaction time} + \text{max delay}] \ (s)$$
$$= 22.3 \ (m/s) * [15 + 8 \ ] = \mathbf{512.9} \ m$$

In addition to notifying the user when the UAV has breached the buffer zone surrounding the critical airspace, the FONA sends another text message update following the warning. This message will indicate that either the user has successfully cleared the No-Fly zone and the surrounding buffer, or that the flight controls have been taken over and the UAV will land or return to its launch point.

Before deciding on the FONA, the team explored multiple options for communication schemes, including 4G and Wi-Fi. One alternative, the USB Huawei Hotspot, is capable of connecting to the internet through the cellular network. This would allow for a simple implementation of an automated email to the airport, in addition to providing all the functionality of the FONA. However, connecting to the internet requires the user to purchase a data package from their provider, which may cost a minimum of 30 USD per month. The FONA requires only a texting plan, at a tenth of the cost, making it a far more practical and realistic alternative.

A Google Voice account designated to the ASPECTS project email account also receives a message from the FONA in the event of a breach of critical airspace, giving information as to the location of the drone and the user ID assigned at registration. This aids in collecting data on airport breaches and provides a way to keep track of repeat offenders.

E. Geofencing Coordinate Fetch Server (2nd Raspberry Pi)

In order to upload the most current geofencing information to our device during operation, our team has designed a remote server to host this information over the Internet. Our physical server (the host) consists of a second Raspberry Pi device connected to the Internet and configured to function as an FTP server using GNU's vsftpd protocol [9]. Our geofencing controller (the client/first Raspberry Pi) can access the server's file system at custom intervals using a Linux command line script when connected to the Internet. The client need only ping the server's static IP address over the FTP protocol, and it receives immediate read permission to download the hosted files (stored as .txt files). Once downloaded by the client, these text files will be parsed by our GPS code into meaningful navigational data.

Designing the server mainly involved UNIX shell programming (ECE 353 and 558), network configuration (ECE 374), and database programming (ECE 242). On the server side, the main challenge was using the Linux shell editor to alter the server's FTP configuration file in order to enable and customize the server. The host device and local network also had to be configured to forward FTP ports 20-22 and maintain a static IP address. On the client side, we must

still design a Linux command line script to be called by our GPS program during operation in order to fetch new geofencing coordinates. For example, the server could contain a database of different coordinate files for various different regions. The server script would be configured to execute upon entering each new region, in addition to executing an automatic check for updates at a custom interval.

In addition to our primary physical backup server, we also designed a secondary backup server that is hosted in the Amazon cloud. Our secondary server also uses the same FTP protocol and access can be similarly automated using the same Linux shell script. In the event that, the primary server is offline the script can be set to repeat the upload using the backup server's IP address instead.

There are two experiments we used to verify the functionality of our data servers. The first, which we had already demonstrated by MDR, involved issuing a manual server call using a GUI-based FTP program on the client device, which allows us to visually capture the transfer. As pictured in Fig. 4 below, when the host's IP address and port information is properly entered into the client, our test file 'MDR_test.txt' is accessed and downloaded to the client's file system. The FTP program reported a file transfer of 84 Bytes in 1 second, which is more than optimal for our needs.

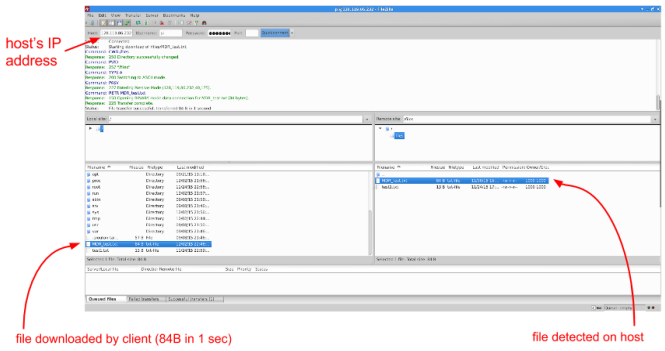

Fig 4. Example of a Manual Server Call using Linux GUI

In addition to the server component, the team also designed a database—seen in Figure 5—containing the name, GPS location, and contact information (contact only for airports that opt-in) for every airport in the United States. Our geofencing module periodically downloads the most current version of this database. From there, our geofencing code can parse through and monitor the drone's proximity relative to any nearby airports. Our system is also capable of synchronizing with and processing any number of geofencing coordinate files, whether airport-related or not, which makes it ideal for the broad range of expected future applications.

171	41.8031	-78.6401	Bradford_Regional_Airport	amherstairport@gmail.com
172	41.938889	-72.683222	Bradley_Intl	amherstairport@gmail.com
173	19.760056	-155.553717	Bradshaw_Aaf	amherstairport@gmail.com
174	46.398308	-94.138078	Brainerd_Lakes_Rgnl	amherstairport@gmail.com
175	41.9335691	-85.0522935	Branch_County_Memorial_Airport	amherstairport@gmail.com
176	39.5924	-75.3455	Brandywine_Airport	amherstairport@gmail.com
177	36.531994	-93.200556	Branson_LLC	amherstairport@gmail.com
178	47.490244	-122.764814	Bremerton_National	amherstairport@gmail.com
179	47.5619	-122.625	Bremerton_Terminal	amherstairport@gmail.com
180	65.331389	-166.465833	Brevig_Mission_Airport	amherstairport@gmail.com
181	41.552	-112.062	Brigham_City	amherstairport@gmail.com
182	44.3048	-96.8169	Brookings_Regional_Airport	amherstairport@gmail.com
183	58.556539	-155.777311	Brooks_Camp	amherstairport@gmail.com
184	42.2511932	-84.9554443	Brooks_Field_Airport	amherstairport@gmail.com
185	38.8819456	-83.8827367	Brown_County_Airport	amherstairport@gmail.com
186	32.5722722	-116.9801611	Brown_Field_Municipal_Airport	amherstairport@gmail.com
187	25.906833	-97.425861	Brownsville_South_Padre_Island_Intl	amherstairport@gmail.com
188	31.2588	-81.4665	Brunswick_Golden_Isles_Airport	amherstairport@gmail.com
189	61.266381	-149.653119	Bryant_Ahp	amherstairport@gmail.com
190	37.706444	-112.145806	Bryce_Canyon	amherstairport@gmail.com
191	37.9896667	-122.0568889	Buchanan_Field_Airport	amherstairport@gmail.com
192	33.420417	-112.68618	Buckeye_Municipal_Airport	amherstairport@gmail.com
193	65.981667	-161.149167	Buckland_Airport	amherstairport@gmail.com
194	39.701668	-104.75166	Buckley_Afb	amherstairport@gmail.com

Fig 5. Screenshot of Geofencing Database for U.S. Airports

III. Project Management

While some particulars of our design evolved over the course of the project, the major goals set by the team at the Preliminary Design Review were achieved by the completion of the Senior Design Project course. This was achieved through individual efforts as well as collaboration to integrate a successful system.

Our team met weekly to review our progress with our advisors Professor Douglas Looze (ECE) and Professor Daiheng Ni (CEE), as well as Civil Engineering undergraduate students. We also met separately as a team to talk about our individual contributions and how each piece would fit into the overall system design. Alex Breger was primarily responsible for the server; Christopher Boselli, for developing flight control software and assembling drone hardware; Jason Danis for GPS processing; and Sandra McQueen, for communication with the UAV owner and local airport. When obstacles arose due to delayed hardware shipment, we adapted our deliverables by focusing on other subsystems so that all contributed equally to the final FDR design. While each had his or her particular subsystem to develop, all team members assumed responsibility for project completion.

Earlier this semester we had the opportunity to present our project at Bradley International Airport, where we received valuable feedback from airport officials regarding their experiences with UAV sightings and their concerns for the future of drone use. In addition to providing further motivation to implement our solution, the conversation at Bradley emphasized the importance of monitoring the frequency of breaches of secure airspaces by drones. This provides them with the data necessary for observing trends and expressing the magnitude of the problem through statistics.

IV. Conclusion

Our team accomplished our major goals for the completion of the ASPECTS project, and it is our hope that the project will be used by ECE teams in the future. ASPECTS addresses a number of safety and security concerns of the FAA regarding runway incursions between small unmanned air vehicles and commercial vehicles. The national database of all the coordinates that require a geofence addresses the issue of identifying the areas that are at the greatest risk of a runway excursion or incursion. The software is flexible enough to be able to alter the radius of the restricted airspace with ease, which would allow the system to change its geofence radius based on the needs of the specific airport or no fly zone. The

notification system implements a warning system for drone operators and air traffic controllers for situations leading to runway incursions. Finally, the autopilot software is a direct solution to reducing runway incursions and associated risks.

Our team was able to successfully demonstrate this project by creating a dummy airport in a small field. The central coordinate of this dummy airport was read into the Raspberry Pi along with the database of real critical airspaces, and the critical and buffer regions were sized appropriately for demonstration. The final specifications of this project show that the GPS is sufficiently fast and accurate, the total module weight is much less than our projected goal, the latency of server synchronization and sending messages is short enough, and the reduction in battery life for the drone is negligible. The only specification that was slightly higher than projected was the total system cost (calculated for 100 units), but this would be greatly reduced if the system were to be mass produced. The final specifications of this project are depicted in the tables and figures below.

Component	Prototype Development	Mass Production
Raspberry Pi (on board)	39.95	34.31
GPS Chip	39.95	31.96
GPS Antenna	12.95	10.36
TTL to USB Adapter	9.95	7.96
FONA 3G Chip	79.99	39.99
FONA Battery	9.95	7.96
FONA Battery Charger	12.50	10.00
SD Card with OS	11.95	7.96
Enclosure	19.40	5.00
Monthly Texting Plan	3.00	3.00
Raspberry Pi (server)	29.99	—
TOTAL	**269.54**	**153.50**

Table 2. Cost Analysis of System Production

Specification	Goal	Actual
Module Weight	<1000 g	428.5 g
GPS Update Rate	>1 Hz	10 Hz
GPS Accuracy	<10 m	3 to 7 m (weather-dependent)
Time Sending Text	<10 s	6 s
Drone Battery Life	8 to 12 min	8.5 min
Server Sync Latency	< 10 s	2 s
System Cost	< 100 USD	154 USD

Table 3. Quantitative Final System Specifications

Specification	Goal	Actual
System Usability	National (All US Airports)	International (all airports in global coordinate database)
Compatibility	Majority (>50%) of U.S. Consumer Market	Significant Majority (>75%) of U.S. and EU Consumer Market – requires widely supported MAVLink compatible flight controller
Automation	Fully Automated	Mostly Automated (Requires manual restart after each run)

Table 4. Qualitative Final System Specifications

An aspect of the system that could be improved upon is the initialization on startup. The system works by entering commands into a terminal after boot up. Ideally, the system would start up automatically upon powering the drone, but this would require a USB to TTL device that differs from the one currently being used for the GPS subsystem. With a unique TTL to USB device, we could have defined different persistent USB driven devices that would have been recognized by the Raspberry Pi on startup. Unfortunately, we were only able to find one sufficient USB to TTL device, which prevented the Pi from recognizing each device until it was fully booted. Although the system needs to be started via command line, it still functions properly, and this aspect can be fixed in the future if a separate TTL to USB device can be found.

As sightings of drone operation in the vicinity of airports is increasing, legislation is being proposed that all drone manufacturers install geofencing software that prevent them from breaching critical airspaces [7]. This implies that manufacturers will need to devise a solution to not only recognize every location that requires a geofence, but also to prevent drones from physically entering these regions. This project takes that solution a step further by also implementing a notification system for both the drone operator and the

airport. The prototype module created by this group has proven to negligibly reduce the flight time of a drone over a single battery life, and it should be sufficiently cheap to manufacture on a broad scale. Installing this prototype module on each publicly available drone would allow manufacturers to adhere to legislation that is predicted to become federal law.

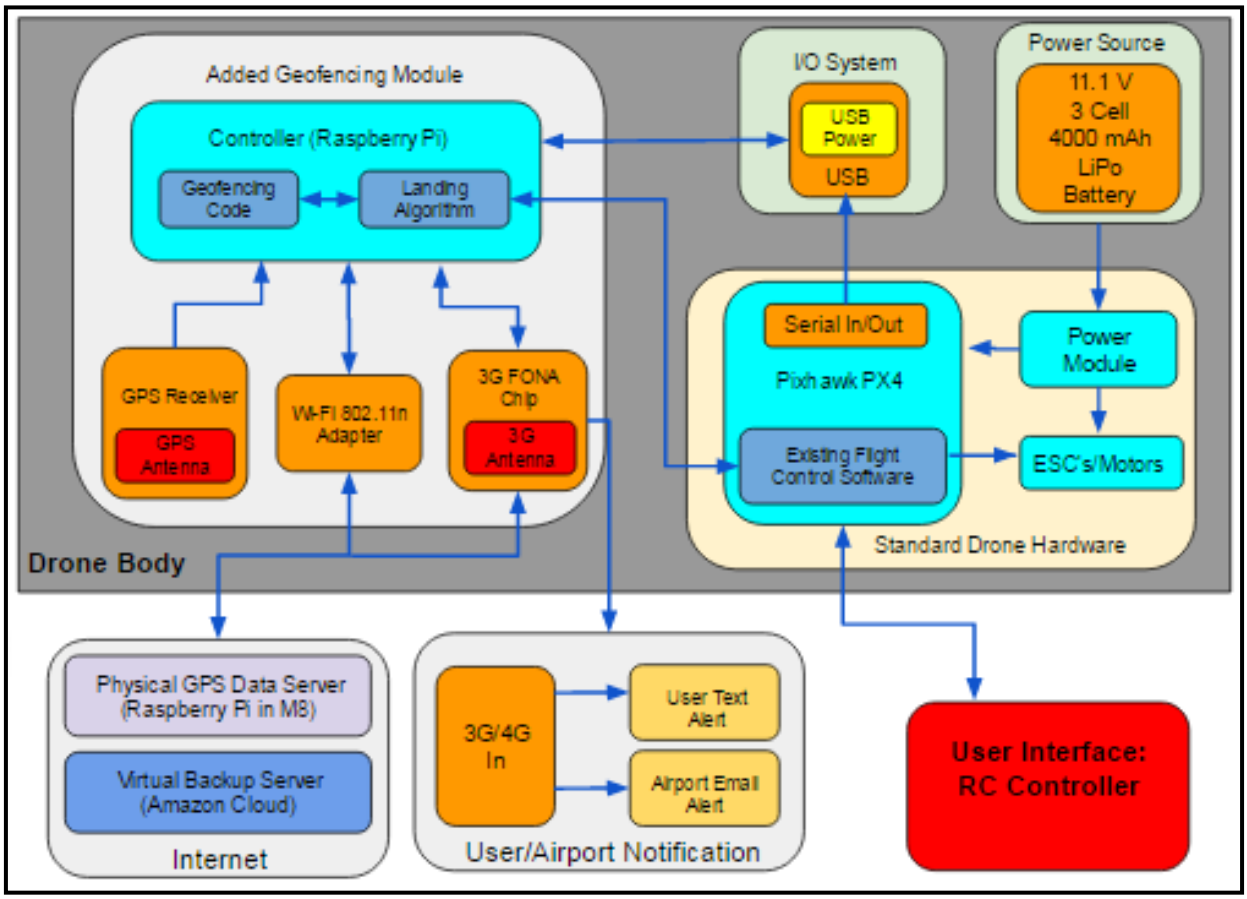

Figure 6: Detailed Block Diagram

ACKNOWLEDGMENT

All of us on the ASPECTS team would like to thank our ECE faculty advisor, Doug Looze, for his valuable insight and contribution to team ideas. We would also like to extend a thank you to our Transportation Engineering advisors, Daiheng Ni and John Collura, for their help and advice along the way. We would also like to thank Bradley International Airport's Air Traffic Manager, Ayaz G. Karzi, for his suggestion and for providing us with the perspective and insight of an airport staff member dealing with these issues every day. His critical advice helped us understand the solution our project is attempting to solve from outside the engineering point of view, and made us realize the severity of the issue at hand.

REFERENCES

[1] B. Jansen, '*Drone sightings spur legislation to fence them in away from planes*', USA TODAY, 2015. [Online]. 7 December 2015. <http://www.usatoday.com/story/news/nation/2015/10/21/drone-rulebreaking-geofencing-dianne-feinstein-charles-schumer/74292314/>.

[2] Federal Aviation Administration, Fact Sheet – *Unmanned Aircraft Systems (UAS).* Washington, DC: US Department of Transportation, January 2014.

[3] "'Drone' Hits BA Plane: Police Investigate Heathrow Incident." *BBC News*. April 18, 2016. Web. April 20, 2016.

[4] "Drones Involved in Near Misses at UK Airports." BBC News, 29 January 2016. Web. April 26, 2016.

[5] "Schumer Bill Would Require Safety Features, Like Geo-Fencing And Sense & Avoid Technology On All Drones; Would Improve The Ability Of Law Enforcement To Take Action Against Reckless Users By Making Drones Detectable And Identifiable To Pilots & Air Traffic Control," *Charles E. Schumer United States Senator for New York*, Oct. 14, 2015. <http://www.schumer.senate.gov/newsroom/press-releases>.

[6] "US Senate Passes Legislation to Let Airports Disable Nearby Drones." *Business Insider*. Web. April 22, 2016. <https://www.schumer.senate.gov/newsroom/press-releases/after-schumers-urging-drone-geo-fencing-proposal-passes-the-senate-newtechnology-could-prohibit-drones-from-flying-near-airports-sensitive-areas-sporting-events-and-above-500ft>.

[7] "AT&T and Intel Want Drones Connected to an LTE Network." *Ars. Technica*. Web. February 22, 2016. . <http://arstechnica.com/business/2016/02/att-and-intel-want-drones-connected-to-an-lte-network>.

[8] SIM Tech, "AT Command Set," SIMCOM5320 command set, July 2014.

[9] Christopher Evans, "vsftp – Secure, fast FTP server for Unix-like systems" (2015) <https://security.appspot.com/vsftpd.html>.