ASPECTS – A Solution to Airport Secure Perimeter Control against sUAS

Chris Boselli, Jason Danis, Sandra McQueen, and Alex Breger
Undergraduate Students
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
100 Natural Resources Road
Amherst, MA 01003
E-mail: {cboselli, jadanis, smmcque, abreger}@umass.edu

Tao Jiang,
Graduate Research Assistant,
Civil and Environmental Engineering
University of Massachusetts Amherst
130 Natural Resources Road
Amherst, MA 01003
E-mail: taojiang@umass.edu

Douglas Looze, Ph.D.
Professor
Department of Electrical and Computer Engineering
University of Massachusetts Amherst
113F Knowles Engineering Bldg
151 Holdsworth Way
Amherst MA 01003-9284
E-mail: looze@ecs.umass.edu
Phone: (413) 545-0973

Daiheng Ni*, Ph.D.
Associate Professor
Civil and Environmental Engineering
University of Massachusetts Amherst
130 Natural Resources Road
Amherst, MA 01003
E-mail: ni@ecs.umass.edu
Phone: (413) 545-5408
Fax: (413) 545-9569

* Corresponding author

Number of Figures: 5
Number of Tables: 4
Word Count: 6130
Abstract:

Small unmanned aircraft systems (sUAS, also known as drones) are becoming increasingly popular among hobbyists, and with this popularity there comes the risk of a runway incursion between a commercial aircraft and sUAS around airports. Despite a 5-mile safe airspace designated by the FAA, there are increasing reports of close encounters between sUAS and manned aircraft as well as full on collisions. To keep airports safe and secure, this research proposes to create a module, called the Airport Secure Perimeter Control System (ASPECTS), that can be attached to every hobbyist's sUAS for the purpose of notification and prevention. Upon startup, the module installed on the sUAS connects to a database containing the central coordinates of every airport in the United States. A 5-mile critical radius as well as an additional one-mile buffer region is established around each point. The buffer region is created in order to inform the user that he or she is approaching a safe airspace and needs to take corrective action. The one-mile radius was chosen to be sufficiently large enough to give the user time to react after receiving a warning message. If the user breaches the 5-mile safe region, autopilot software takes over the manual controls, and the sUAS is landed in a controlled manner. As the sUAS is being landed, the user still has lateral control of the vehicle in order to avoid any potential hazards below it. Upon landing and disarming the operator and the airport receive messages that a sUAS has entered the 5-mile critical space, and has been landed via autopilot software. To demonstrate the proposed design, a prototype was developed that successfully implemented this system, and was formally tested within a controlled environment.

Keywords: Small unmanned aircraft systems (sUAS), Drone, Geofencing, Airport
MOTIVATION AND OBJECTIVE

As small unmanned aircraft systems (sUAS, also known as drones) operation has become increasingly popular among hobbyists, there is a growing risk of runway incursions around secure airspaces such as airports. This is a relatively new hobby that people have begun to take up, and current laws are struggling to adapt as quickly as the small sUAS technology and capabilities becoming available to the public. There have been a number of reports recently that are making the public more aware of the potentially deadly consequences of not abiding by federal regulations concerning sUAS use.

USA Today stated in 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, prompting lawmakers to tag the popular remote-controlled aircraft with electronic collars that would keep them away”[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”[2]. Earlier in 2016, at the same airport as the previously mentioned collision, there was a near miss between an Airbus carrying over 150 passengers and a sUAS.

BBC news reported that a “near miss at Stansted saw a drone fly over the Boeing 737 by about 16ft, as the aircraft was at about 4,000ft during take-off. In another incident, a drone narrowly missed hitting the wing of a Boeing 777 shortly after take-off from Heathrow Airport on 22 September...It is calling for stricter rules and a registration system so drone operators can be easily traced and prosecuted for any "irresponsible flying". Pilots also want technology routinely fitted to drones to stop them from being able to fly in areas where they could meet commercial traffic”[3]. Although there is a major threat imposed on airport safety and security by sUAS, there is currently no universal means to ensure that the Federal Aviation Administration (FAA) mandated five mile radius around airports is not breached. Whether or not the actions of sUAS pilots are malicious or simply ignorant, there needs to be a way to ensure the safety of all passengers aboard commercial aircraft.

The objective of this research is to propose a solution to address the issue of sUAS intrusion at airports. An example solution of this kind was reported as the first place winner of the 2014-2015 Airport Cooperative Research Program (ACRP) University Design Competition for Addressing Airport Needs: Runway safety/incursions/excursions: Eye in the Sky - Drone Detection & Tracking System. In summary, this particular solution proposed that manufacturers install RFID chip on each individual unit sold. The corresponding RFID tracking systems with individual Adjustable Active RFID Readers would be responsible for detecting drones in the critical airspace set up all around the perimeter of the FAA mandated 5-mile radius. The range of these RFID tracking systems were expected to be 200-300 meters each, and individually cost around $700 (USD) [4]. Upon breach of this critical airspace by a sUAS, the airport would be notified of the sUAS RFID tracking number, which would be a part of a national database of all sUAS registered to users. The implementation of this entire innovative system would cost an airport approximately $34,212.56 (USD) [4]. This solution would drastically reduce the security threat of sUAS breaching the critical 5 mile critical regions surrounding airports, and its implementation is relatively simple.

Although the previously discussed solution would effectively eliminate the threat of sUAS breaching critical airspaces, we are proposing a unique solution which we believe to be more effective. It is our firm belief that the responsibility of any unforeseen consequences resulting from improper sUAS operation should lie with the user, and not with the airport that was breached. When a hobbyist makes the decision to purchase and operate a sUAS, the user should take on all the responsibility of operation whether or not he or she is aware of federal laws and regulations. The operation of a sUAS is a privilege, not a right. On top of this shift of responsibility from the airport to the user, our solution can be
implemented with zero additional cost to airports, which includes the cost of equipment damage and maintenance. The details of this solution are summarized in the following sections of this document.

LITERATURE SURVEY

A wide range of resources were consulted as we progressed from initial conception all the way to final systems testing and public demonstration. In the earlier stages, our research focused on surveying news and media outlets in order to identify specific areas of public need with respect to airport runway safety. The advantages and disadvantages of existing geofencing systems were also researched and evaluated early on. As the research evolved into its current form, our focus became significantly more technical and economic in nature. The research team consulted many handbooks and manuals in order to familiarize ourselves with the multiple software interfaces involved in our solution. We also researched vendors, part prices, and part specifications for various system components in order to minimize overall system cost and footprint.

News and Media Survey

A survey of recent news concerning sUAS activity has shown a rise in several noteworthy trends. The number drone sightings within the restricted FAA 5-mile radius has been steadily increasing, as has the number of close-calls or collisions reported between sUAS and commercial aircraft. Intel and AT&T also recently announced a partnership to design integrated 4G LTE and other enhanced smart capabilities for future sUAS, indicating that recreational drone activity will only continue to rise in popularity and sophistication going forward [5].

In response to this increased risk, political action towards developing more robust sUAS regulation has been undergoing as well. On April 21, 2016 the U.S. Senate voted 95-3 in favor of requiring the FAA to develop new sUAS safety standards that include geofencing [6]. This bill is moving onto the U.S. House of Representatives for consideration this term and if enacted would commence the FAA rulemaking process immediately.

Review of Existing Geofencing Systems

In addition to consulting current news for inspiration, we also reviewed existing geofencing solutions in order to examine potential areas for improvement. Some of these systems included previous Airport Cooperative Research Program (ACRP) University Design Competition project winners such last year’s Eye in the Sky RFID-based system. We also examined emerging GPS-based geofencing technology being implemented by certain sUAS manufacturers on their higher end consumer models. Existing manufacturer solutions tend to be expensive, proprietary, and specific to particular models or firmware—none of which is conducive to an affordable and generalized solution for all recreational users.

Review of Technical Literature

Once settled on a conceptual design, we had to review necessary technical documentation in order to identify system components that could meet our desired specifications. Our communications subsystem uses the AT interface, which required us to become familiar with the AT Commands: Interface Guide [7]. Similarly, our autopilot subsystem uses the MAVLink interface, requiring us to become proficient in the MAVLink Common Message Set [8]. Lastly, our main controller executes approximately 400 lines of Python code in order to automate control over our GPS, communications, and autopilot subsystems. Gaining the necessary proficiency with Python required us to study the Python Language Reference and other supporting documentation extensively [9].
Conclusions Drawn from Literature Review

Based on our multipronged review of current news and technology, we determined that the consumer sUAS market is missing a universal geofencing solution with robust features such as autopilot landing override and a mobile-based notification system capable of leveraging existing cellular infrastructure. With mandatory nationwide geofencing likely on the horizon for the FAA in 2016, regulators, manufacturers, and other stakeholders will need reliable and affordable solutions that meet those criteria and more. Our final design was fueled with these key literature considerations in mind.

DESIGN CONSIDERATIONS

This research aspires to improve runway safety for commercial airplanes at airports due to the steadily increasing popularity of sUAS, and the solution is envisioned to be financially and logistically feasible from the perspective of both the airport and the user. To serve the airports’ needs, we wanted to create an autonomous system which would not require any additional action by air traffic control and would not incur any monetary expense for the airport. For the user, the marginal cost would have to be low in comparison to the overall cost of the sUAS, since excessive financial burden would likely discourage compliance. The ultimate goal was to regulate traffic in the airspace and prevent potentially dangerous and fatal interference with passenger planes.

A preliminary survey of geofencing reveals that it is a way of defining virtual geographical boundaries by setting a central point and a radius around that point. Geofencing has many industrial and marketing applications; when paired with hardware and software, it can be used to simulate a physical barrier to keep aircraft outside (or inside) of the specified area. Further research showed that some high-end sUAS have already incorporated geofencing through firmware updates. However, this did not apply to less expensive models and those which are built entirely by the user. Our new goal was to provide a geofencing module which could be retrofitted on an existing sUAS or a “do-it-yourself” model, and therefore address the vast majority of sUAS that do not currently have geofencing capabilities.

Making use of existing infrastructures was another priority in minimizing costs, and so we used the cellular network to set up communication with the user. Outside of the no fly zone, a designated buffer zone can be defined, where the user must fly cautiously, see an illustration later in Figure 2. This is where the user receives the first warning that there is a restricted area nearby. While high-end sUAS often rely on a smartphone application to interface with the user and display warnings or alerts, the SMS notification that we use is simple and compatible with any phone. Several possibilities were considered when deciding on the best way to establish contact with the pilot, and it was concluded that a 3G device would serve our purpose. Among the other options was a Hotspot which would connect our on-board processor to the internet via 4G and facilitate the sending of an email to the local airport, but this would require a data plan which costs the user approximately $30 USD per month.

With geofencing module being conceptualized (the technical details of which are discussed in a later section), the next question is to consider what kind of sUAS best suite our need. Table 1 is a comparison chart defining some of the desired specifications we envisioned for an ideal sUAS.

Table 1. Desired specifications for the envisioned sUAS

<table>
<thead>
<tr>
<th>Model</th>
<th>Flight Time (minutes)</th>
<th>GPS Enabled</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeresRoad CXHobby CX-20</td>
<td>15</td>
<td>Yes</td>
<td>320</td>
</tr>
<tr>
<td>Storm RC X3</td>
<td>12-15</td>
<td>Yes</td>
<td>300</td>
</tr>
<tr>
<td>Parrot AR Drone 2.0</td>
<td>12</td>
<td>Yes</td>
<td>349</td>
</tr>
<tr>
<td>Custom Arduino-based Quadcopter</td>
<td>~12</td>
<td>No</td>
<td>~400</td>
</tr>
</tbody>
</table>
Though commercial products are readily available, the most effective way to provide proof of concept for this research would be to build our own sUAS because this would allow us an open-source platform, more control over the software, and a deeper understanding of the functionality and capabilities of the sUAS itself. Indeed, we build the module using one of the most widely-used flight controllers in DIY as well as commercial sUAS: the 3D Robotics Pixhawk. In addition, our module will work with any flight controller featuring a telemetry input port, ensuring compatibility with about 75% of publicly available sUAS.

In order to fully consider the safety concerns of this research, we consulted section 3.1 of the FAA Safety Management System Manual. A successful Safety risk management system determines associated risks, identifies the severity and probability of the occurring risks, develops mitigation strategies as appropriate, applies, tracks, and monitors the mitigation strategy, and assesses and modifies strategies as necessary [9]. According to the FAA Safety Management System Manual, redundancy is a key focus point for equipment defense strategies [9], and thus should be integrated into the design to ensure that the system against failure.

THE PROPOSED SOLUTION

Our proposed solution, a module called the Airport Secure Perimeter Control System (ASPECTS), was designed to eliminate the threat of hobbyist sUAS within the critical five mile restricted airspace mandated by the FAA around all airports in the United States. In order to successfully demonstrate complete functionality, many individual systems such as the Geofencing module, notification system, and static file server system were integrated to implement the final design.

Overview of the Design

As a universal geofencing solution, ASPECTS may be incorporated on future sUAS 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 4 provides a more detailed diagram of the system.)

By constantly monitoring the location of the sUAS 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/outer boundary in Figure 2, allowing an opportunity for the user to manually redirect the flight path.

Figure 1. High-level block diagram of ASPECTS
If the user does not take corrective action before the sUAS reaches the No-Fly Zone, the on-board controller will then assume control of the sUAS in order to prevent a breach of the critical airspace (indicated by the red/inner boundary in Figure 2) 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.

**Components of the Design**

Schematically illustrated in Figure 1, the Geofencing module of the ASPECTS system is the portion of the system that acts much like an electric dog fence in order to keep sUAS threats out of an airport’s restricted airspace. The module itself also consists of two main subsystems: the GPS antenna/receiver, and the controller hardware. The Geofencing module is responsible for monitoring geofenced areas associated with airports by interpreting location information from the GPS antenna/receiver and comparing the location of the sUAS with any nearby no-fly zones. This process is done using software programmed into the control hardware. The distance between the sUAS and any nearby airports is monitored by continuously by the programmed controller hardware in real time. When that distance is calculated to be inside of the five mile critical radius, the controller hardware signals the notification system and then proceeds to execute a program which takes over the flight controller on the sUAS in order to safely land the vehicle. Key components of the proposed design are further elaborated as follows.
The sUAS Body

One of the earliest changes made to our original vision for this project was the decision to build our own sUAS “from scratch” rather than simply purchasing a fully integrated sUAS. 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.

One of the major aspects that needed to be considered when designing our own sUAS 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 sUAS. 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 price and weight of each component can be seen in Table 2.

Table 2. On-board hardware and corresponding weight contribution to the design

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Price</th>
<th>Qty</th>
<th>Total Cost</th>
<th>Weight (g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadcopter Frame</td>
<td>Tarot 650 carbon fiber frame</td>
<td>$110.38</td>
<td>1</td>
<td>$110.38</td>
<td>485</td>
</tr>
<tr>
<td>Flight Controller</td>
<td>Pixhawk</td>
<td>$119.89</td>
<td>1</td>
<td>$119.89</td>
<td>100</td>
</tr>
<tr>
<td>Battery</td>
<td>nano-tech 4000mA</td>
<td>$37.65</td>
<td>2</td>
<td>$75.30</td>
<td>333</td>
</tr>
<tr>
<td>Propellers</td>
<td>3DR</td>
<td>$9.99/2 sets</td>
<td>3</td>
<td>$30.00</td>
<td>6</td>
</tr>
<tr>
<td>Motors</td>
<td>3DR Black Top Motor</td>
<td>$15/ea</td>
<td>4</td>
<td>$60.00</td>
<td>272</td>
</tr>
<tr>
<td>ESCs</td>
<td>3DR 20 A</td>
<td>$13/ea</td>
<td>4</td>
<td>$52.00</td>
<td>84</td>
</tr>
<tr>
<td>On-board controller</td>
<td>Raspberry Pi Model B</td>
<td>$40/each</td>
<td>1</td>
<td>$40.00</td>
<td>45</td>
</tr>
<tr>
<td>3G/GPS Communication</td>
<td>Adafruit FONA 3G Breakout</td>
<td>$79.95</td>
<td>1</td>
<td>$79.95</td>
<td>9</td>
</tr>
<tr>
<td>Hand-Held Controller</td>
<td>Turnigy 9-Channel Transceiver</td>
<td>$59.99</td>
<td>1</td>
<td>$59.99</td>
<td>--</td>
</tr>
<tr>
<td><strong>Grand total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>$627.51</strong></td>
<td><strong>1334</strong></td>
</tr>
</tbody>
</table>

Another key aspect of the design process was selecting which battery to purchase that would power all of the on-board components of the sUAS. We had to take into consideration the power consumption of the sUAS 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.

Controller

In order to interpret GPS data, run code for overriding the sUAS’s microcontroller, and run code for landing the sUAS 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 sUAS 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 sUAS’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 sUAS’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 sUAS has breached a restricted area, the Raspberry Pi communicates with the Pixhawk microcontroller, which is the flight controller installed on the sUAS, to override its software and safely land the sUAS. The software to land the sUAS is executed after it has been confirmed that the Raspberry Pi has priority over the Pixhawk. Upon landing, the sUAS disarms itself and is incapable of being rearmed until the user takes it outside of the restricted space and rearms the system via reboot.

![Figure 3. Breakdown of regions surrounding a geofenced area](image)

**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 [10].

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 sUAS, users can provide a cell phone number at which they can be contacted for alerts. If the sUAS 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 sUAS 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 sUAS:

\[
B = \text{avg. max speed (m/s)} \times [\text{reaction time + max delay}] \ (s)
\]

\[
= 22.3 \ (\text{m/s}) \times [15 + 8] = 512.9 \text{ m}
\]

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 sUAS 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.

**GPS Antenna/Receiver**

The GPS Antenna/Receiver system consists of an Adafruit Ultimate GPS Breakout Version 3 chip along with the corresponding external active GPS antenna with a 5 meter uFL to SMA adapter cable. This system itself is responsible for receiving and parsing GPS satellite information that is to be used by the rest of the Geofencing module’s subsystems as well as the notification system.

The Adafruit Ultimate GPS Breakout Version 3 chip is a compact, lightweight chip providing -165 dBm sensitivity over 66 channels with a compatible +3.3-5 VDC design. At only $39.99, this chip provides a fast location refresh rate of 10 Hz while exhibiting minimal power consumption due to the maximum 20mA current draw during navigation. The chip also provides some data logging capability due to some internal flash memory in its on-board microcontroller [6].

The corresponding Adafruit external active antenna that is compatible with the GPS breakout chip mentioned above uses a 3-5 V, 5 meter long uFL to SMA adapter cable. This antenna is suitable do to its adequate 28 dB of signal amplification while only drawing an average of 10 mA of current. It also provides a substantial operating bandwidth of 10 MHz along with an operating temperature range of -30-85 degrees Celsius. One issue is that the 5 meter long cable is the shortest that comes with this particular antenna. A smaller cable would correspond to a smaller enclosure for our module and also a lower overall weight of the system [6].

**Geofencing Coordinate Fetch Server (2nd Raspberry Pi)**

In order to upload the most current geofencing information to our device during operation, we 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 [11]. 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, network configuration, and database programming. 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 involved issuing a manual server call using a GUI-based FTP program on the client device, which allows us to visually capture the transfer. When the host’s IP address and port information is properly entered into the client, our test file 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.

In addition to the server component, we also designed a database 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 sUAS’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.

FIELD TEST

With the software complete and the hardware integrated, we set up a 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 sUAS 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. 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 sUAS.

The primary function of the system is to avoid breaches of the five mile safe airspace surrounding an airport, so the fundamental subsystem on the prototype is the autopilot landing software. This software has been thoroughly tested and has been shown to respond within 10 seconds of a breach. Although there is a bit of latency regarding the speed of the software response, it is negligible on the scale of five miles. The autopilot itself has the feature of allowing the user to laterally control the sUAS as it descends. This allows the operator to avoid collision with hazardous objects, especially in adverse weather conditions. Once the sUAS is landed, it disarms itself and fails to rearm until the user takes the vehicle outside of the restricted airspace and resets the system via reboot. If the autopilot fails to respond to the sUAS entering a restricted airspace, the notification system adds redundancy that notifies the airport as well as the user that a breach has occurred and corrective action needs to be taken. This ensures that the airport will have time to respond to the undesired sUAS if the user fails to take any action. Also, because this prototype module is powered from the sUAS power supply, instead of an external source, it will continue to work as long as the vehicle is able to operate. The multiple redundancies that the ASPECTS system relies on maximizes the probability that corrective action will be taken in some form upon a breach of a restricted airspace.
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.

FINAL DESIGN SPECIFICATIONS

This research satisfies the ACRP goals by taking a number of suggested ideas and implementing them into a single system. The system utilizes direct warning systems to alert air traffic controllers for situations leading to runway incursion, innovative approaches to reducing runway excursions and associated risks, and innovative processes to identify the hazards that present the greatest risk to air carrier operations within the runway environment and strategies to mitigate those hazards and improve safety of airport surface operations. All of the above solutions were suggested by this competition as general guidelines for determining a sufficient plan [4]. 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 excursion. 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 sUAS 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. The proposed design of Geofencing model is further detailed in Figure 4 and the resulting prototype is pictured in Figure 5.

![Figure 4. A visual representation of our Geofencing solution](image-url)
Table 3 summarizes the cost analysis of producing the prototype as well as projected costs for mass production.

**Table 3. The cost analysis of producing the prototype**

<table>
<thead>
<tr>
<th>Component</th>
<th>Prototype Development</th>
<th>Production (Mass manufacturing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi (on board)</td>
<td>$39.95</td>
<td>$34.31</td>
</tr>
<tr>
<td>GPS Chip</td>
<td>$39.95</td>
<td>$31.96</td>
</tr>
<tr>
<td>GPS Antenna</td>
<td>$12.95</td>
<td>$10.36</td>
</tr>
<tr>
<td>TTL to USB Adapter</td>
<td>$9.95</td>
<td>$7.96</td>
</tr>
<tr>
<td>FONA 3G Chip</td>
<td>$79.99</td>
<td>$31.96</td>
</tr>
<tr>
<td>FONA Battery</td>
<td>$9.95</td>
<td>$7.96</td>
</tr>
<tr>
<td>FONA Battery Charger</td>
<td>$12.50</td>
<td>$10.00</td>
</tr>
<tr>
<td>SD Card with OS</td>
<td>$11.95</td>
<td>$7.96</td>
</tr>
<tr>
<td>Enclosure</td>
<td>$19.40</td>
<td>$5.00</td>
</tr>
<tr>
<td>Monthly Texting Plan</td>
<td>$3.00</td>
<td>$3.00</td>
</tr>
<tr>
<td>Raspberry Pi (server)</td>
<td>$29.99</td>
<td>—</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>$269.54</strong></td>
<td><strong>$145.47</strong></td>
</tr>
</tbody>
</table>

In this case, mass manufacturing refers to the cost to build 100 units, which implies that production on a larger scale would be even cheaper per unit. As previously stated, the solution proposed by the *Eye in the Sky - Drone Detection & Tracking System* in the 2014-2015 competition submission would cost each airport approximately $34,212.56 (USD) per unit [4]. While simple to implement, this solution places a high cost on each airport that agrees to use the system. Our proposed solution takes this responsibility off of the airports and places it on the user. At $145 (USD) or less per unit, this solution is affordable, and the cost is nearly negligible compared to cost of most publicly available sUAS. Reaching out to sUAS manufacturers with a prototype that is affordable and has been demonstrated to be reliable would be the first step of the process of getting this system mass produced. Installing this prototype module on each publicly available sUAS would allow manufacturers to adhere to legislation that is predicted to become federal law.

More details of the final specifications of the proposed design are provided in Table 4.

**Table 4. Quantitative Final System Specifications**
<table>
<thead>
<tr>
<th>Specification</th>
<th>Goal</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Weight</td>
<td>&lt;1000 g</td>
<td>428.5 g</td>
</tr>
<tr>
<td>GPS Update Rate</td>
<td>&gt;1 Hz</td>
<td>10 Hz</td>
</tr>
<tr>
<td>GPS Accuracy</td>
<td>&lt;10 m (weather-dependent)</td>
<td>3 to 7 m</td>
</tr>
<tr>
<td>Time Sending Text</td>
<td>&lt;10 s</td>
<td>6 s</td>
</tr>
<tr>
<td>Drone Battery Life</td>
<td>8 to 12 min</td>
<td>8.5 min</td>
</tr>
<tr>
<td>Server Sync Latency</td>
<td>&lt; 10 s</td>
<td>2 s</td>
</tr>
<tr>
<td>System Cost</td>
<td>&lt; 100 USD</td>
<td>154 USD</td>
</tr>
<tr>
<td>System Usability</td>
<td>National (All US Airports)</td>
<td>International (all airports in global coordinate database)</td>
</tr>
<tr>
<td>Compatibility</td>
<td>Majority (&gt;50%) of U.S. Consumer Market</td>
<td>Significant Majority (&gt;75%) of U.S. and EU Consumer Market – requires widely supported MAVLink compatible flight controller</td>
</tr>
<tr>
<td>Automation</td>
<td>Fully Automated</td>
<td>Mostly Automated (Requires manual restart after each run)</td>
</tr>
</tbody>
</table>

CONCLUSIONS

ASPECTS addresses a number of safety and security concerns of the FAA regarding runway incursions between sUAS and commercial aircraft. 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 or any secured area. The notification system implements a warning system for sUAS 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.

We were able to successfully demonstrate this design 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 design show that the GPS is sufficiently fast and accurate, the total module weight is much less than our envisioned goal, the latency of server synchronization and sending messages is short enough, and the reduction in battery life for the sUAS is negligible. The only specification that was slightly higher than what was anticipated 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 have been provided.

As sightings of sUAS operation in the vicinity of airports is increasing, legislation is being proposed that all drone manufacturers install geofencing software that prevents them from breaching critical airspaces. 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 research takes that solution a step further by also implementing a notification system for both the drone operator and the airport. The prototype module developed herein has proven to negligibly reduce the flight time of a sUAS 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.

ACKNOWLEDGEMENTS
This work was entered into the 2015-2016 Airport Cooperative Research Program (ACRP) Design Competition for Universities and won first place in the category of Runway Safety/Runway Incursions/Runway Excursions Challenge.

This research is partly supported by the United States Department of Transportation (USDOT) University Transportation Center (UTC) grants.

The authors thank Mr. Jared Geller, a Civil Engineering undergraduate assistant at UMass Amherst, for his help in this research.

REFERENCES