Collision Avoidance at low cost using ADS-B

Auburn REU on SMART UAVs 2021
Technical Report# CSSE21-02

Martinez, Manuel
University of Illinois Urbana-Champaign
mmart330@illinois.edu

Poudel, Samikshya
New Jersey Institute of Technology
poudel4samikshya@gmail.com

July 8, 2021

Abstract

This work present the implementation of Automatic Dependent Surveillance-Broadcast (ADS-B) technology on Unmanned Aerial Vehicles (UAVs) for collision avoidance between aircraft at low cost. Currently, the FAA does not require ADS-B on UAVs, but this is likely to be changed with the growing popularity of UAVs. ADS-B is a surveillance technology that is used to broadcast information about the aircraft using satellite or other sensors. It provides pilots and ground controllers with real-time precision and shared situational awareness. Currently,ground controllers at Auburn Airport cannot accurately count aircraft in their vicinity in a cost-effective way.This work will develop an ADS- B receiver to address this problem. It will then be incorporated into the UAVs for aircraft collision avoidance. This work will use a software-defined radio receiver to decode ADS-B radio signals into binary bits. Then ADS-B decoder software will be installed on the raspberry pi to extract the ADS-B information. The decoded information will be used to maneuver UAVs without aircraft collision.

1 Introduction

Unmanned Aerial Vehicles **(UAVs)** refer to aircraft that do not directly have a pilot on board controlling it; also called a drone. Pilots controlling the drone can be several meters away or several thousand kilometers. In some instances, UAVs do not even need a pilot and can fly autonomously. Research in autonomous UAVs has become a major interest in academia and industry. There are many variations of UAVs in size, implementation, and type. The size of UAVs range from group 1 (<20lbs) with speeds of about 115 mph, group 2 and 3 (21-55lbs or <1320) with speeds <287.7 mph, group 4 (>1320 lbs) with operating altitude of <180ft, and group 5 (>1320lbs) with operating altitudes of >180 ft [2]. Some UAVs fly using fixed wings like most aircraft while others use several rotary propeller motors to hover, similar to a helicopter. The use of UAVs has become increasingly popular in recent years particularly for personal use but also in other industries. The cost of UAVs has also become more affordable which has contributed to this increased popularity. Their uses range from monitoring wildlife, conducting geological surveys, to finding missing people. The application of UAVs has already been deployed for military uses and because of their efficiency and enhanced safety, they are now being applied for commercial and private use. Integration of UAVs into the various national airspaces is a major concern particularly with Small Unmanned Aerial Systems **(SUAS)** [1]. In the US, the Federal Aviation Administration (FAA) has outlined a set of regulations that all aircraft

must follow to be able to fly in the National Airspace System **(NAS)** but explicitly excludes SUAS. In 2014 the FAA addressed this issue by enacting regulations on hobby-type UAVs between 250 g and 25 kg [1].

The FAA also published an Automatic Dependent Surveillance-Broadcast (ADS-B) requirement in heavy aircraft in 2010 and mandated that all aircraft must have ADS-B Out by 2020 [7]. Due to the massive growth in the number of flight operations and the growing popularity of UAVs in civil applications, it has made it difficult to work with traditional radar systems. Traditional radar systems use radio waves to determine the distance, flight angle of the aircraft which can be limited since it is heavily dependent on line of sight. In other words, radio stations built to a low altitude can have problems accurately detecting every aircraft. Unlike the traditional radar system, ADS-B uses GPS satellites, transmitters, and receivers to broadcast aircraft information. Since the ADS-B messages are broadcasted in the air, traffic can be monitored at low altitudes and on the ground. Furthermore, it is effective in remote areas where radar coverage is limited or completely absent. ADS-B systems are much cheaper to maintain and deploy than traditional radar systems [7].

ADS-B broadcasts provide real-time situational awareness to air traffic controllers and aircraft pilots. It tracks active aircraft information such as aircraft identification, altitude, speed, and velocity. An ADS-B systems comprises of two subsystems: ADS-B Out and ADS-B In. The ADS-B Out transmits ADS-B messages containing aircraft information such as location; which is received from GPS satellites, altitude, speed, etc. to aircraft and ground stations. The ADS-B In receives the ADS-B messages from the nearby aircraft transmitting. The benefit of using ADS-B is that it helps aircraft to safely navigate dense airspaces and avoid mid-air collisions [8]. The ADS-B regulation falls under one of the FAA's Next Generation Air Transportation System (NextGen) modernization programs which focuses on safer, more efficient, and more predictable flights. The concentration of this program is to develop and deploy innovative and transformative technologies to improve safety, efficiency, environmental performance, and passenger experience even in a busy airspace [3]. Regulation of UAVs and SUAS is likely to grow as popularity continues to rise.

(a) Raspberry Pi 4 (b) RTL-SDR

Figure 1: Raspberry Pi and RTL-SDR dongle

This paper focuses on the design and implementation of an ADS-B receiver to count the number of aircraft flying near Auburn airport and the incorporation of the receiver into UAVs for collision avoidance.

A computing device known as a raspberry pi and a USB software-defined radio **(SDR)**, both available at low cost, will be used to pick up radio signals from the air.An ADS-B decoding software such as Dump1090 will then be used in conjunction with the SDR to decode ADS-B packets. The ADS-B decoding software will then be modified to include a function that will accurately track all take-offs, landings, and overflights. The receiver will be used in UAVs to get the aircraft information in its proximity. The UAV's autonomous flying software will be installed and a collision avoidance algorithm will be implemented and modified according to the received data from the ADS-B receiver. The rest of the paper consists of problem description, related works, algorithm description, implementation, and conclusion.

2 Problem Description

For our research problem, we wish to address two issues that build upon each other. Firstly, Auburn Airport is unable to accurately count all aircraft in its vicinity. Auburn Airport uses educated guesses to estimate the number of aircraft taking off, landing, and passing through. Current software and hardware that can count aircraft accurately exists but is expensive which makes it cost-ineffective to buy/rent the equipment that allows for proper tracking of aircraft. Secondly, because of the FAA requirement of ADS-B Out in all aircraft we see a viable way to implement UAV aircraft collision avoidance using this technology along with existing collision avoidance algorithms.

3 Related Works

We will discuss some previous research applicable to our project that range from using ADS-B radar systems to using path planning algorithms.

3.1

Camera-based approaches have been proposed but are usually limited by long navigation distance and weather. The article *Ads-b based collision avoidance radar for unmanned aerial vehicles* by C. Lai, Y. Ren, and C. Lin, discusses the usage of ADS-B not only for situational awareness and also using the physical broadcast itself as a radar pulse [4]. In this way, ADS-B can be used as radar for collision avoidance and would be effective with all air traffic and in all weather. A standard ADS-B transponder was used which broadcasted from a UAV every second through an omnidirectional antenna. The broadcasted ADS-B radio signal was then reflected by the UAV which had 4 mounted directional antennas. The detection of the obstacles and their distances could be achieved by comparing amplitudes and angles-of-arrival phases of the signals received by each of 4 individual antennas. For this method to be effective, a lookup table must be created by simulation and/or lab measurements to accurately estimate distances and angles (0 degrees being the front of UAV). A problem found with this method though is that it is generally assumed that the target is detected in the direction of the main beam(max gain), but if the UAV tilts then the main beam may not point toward the target which results in an angle-of-arrival estimation error. It was found that a UAV tilt(bank) of +/- 15 degrees resulted in an acceptable error, but anything outside of this caused inaccurate obstacle estimations. Patch antennas were used instead of horn antennas because they were lighter and smaller, but the antenna patterns had to be known beforehand. In the experiment, a standalone ADS-B radar transceiver was constructed and was tested without the UAV. The ADS-B broadcasts were transmitted and reflected with different signal power from targets at different distances. They found that when the target was within 2km there was a high chance of detection and a low chance of false alarm. Past 2km detection probability is 90% and false alarm probability is high at 40%. This issue can be mitigated by increasing the transmitted power but this would require larger, heavier, and more expensive amplifiers [4] which would not be practical in our application.

The implementation in this report solely discusses how to detect but not how to autonomously avoid obstacles/aircraft which is what we wish to achieve. Also as discussed in the introduction, as the NAS continues to experience increased aircraft activity radar systems may not be optimal. Nonetheless, this implementation can likely be adopted into approaches similar to ours where once an obstacle is detected by a running algorithm uses this information to avoid the collision.

3.2

One such example is a research paper *Sense and avoid for unmanned aerial vehicles using ads-b* by Y. Lin and S. Saripalli discusses autonomously avoiding incoming aircraft using an algorithm known as Closed Loop Rapidly Exploring Random Tree (CL-RRT) [5]. In this paper aircraft collision avoidance is defined as a path planning problem where a collision-free path is generated connecting all waypoints that costs the shortest length given some initial state. The path must also maintain a given horizontal and vertical distance from other aircraft. Closed-loop dynamics are used with a low-level controller that takes two inputs: a reference command $\boldsymbol{r}$ = (x,y,z) which is the desired UAV location in 3D space and the UAVs current state $\boldsymbol{x}$ = (x,y,z, speed, yaw, pitch); a state is defined as some 3D space coordinates, speed, yaw, pitch. The controller then outputs a control $\boldsymbol{u}$ which controls and adjusts the UAVs motors' speed which maneuvers it to the next state. The controller uses the CL-RRT algorithm to run a forward simulation to predict the next possible states and then checks each against the constraints (i.e. horizontal/vertical distance from aircraft). The algorithm essentially predicts a path using closed-loop dynamics by generating random waypoints for some time interval, predicting a path from the waypoint to the start location, and checking to see if constraints are satisfied. If satisfied, temporarily add the waypoints as child nodes with the start node as the parent. At the end of the time interval, the temporary child node with the shortest length is added to the tree. In this manner, rapid tree expansion occurs which eventually yields a collision-free path to the goal location. For the experiment, real ADS-B data was collected and used in a simulated environment to test the CL-RRT algorithm. The UAV and the controller hardware ran the same in the simulation as they would in real flight[7]. More than 80 runs were conducted with various aircraft approaching the UAV from different angles and only 5 failed the given constraints of 50m vertical distance and 300m horizontal distance from all other aircraft. Of the runs that failed none resulted in a collision and were fairly close to the constraint boundaries.

This report shows that it is viable to use the CL-RRT algorithm to autonomously avoid collisions and approach collision avoidance as a path-finding problem. Another feasible implementation for autonomous collision avoidance could be a combination of the transceiver used in the previous report that used ADS-B as radar and the CL-RRT algorithm. The transceiver could be used to detect obstacles and provide this data to the CL-RRT algorithm which would then adjust the UAVs path to avoid the collision. However, this works aims to achieve autonomous collision avoidance of other aircraft at low cost, so we instead we could use the ADS-B broadcasted data in conjunction with CL-RRT.

3.3

While the popularity of UAVs has grown tremendously in recent years, it has also increased the threat of collisions among manned aircraft and UAVs at low altitudes. A Conflict Detection and Resolution (CD&R) system was developed and implemented on both manned aircraft and UAVs to address this problem. In the research *Quasi-ADS-B Based UAV Conflict Detection and Resolution to Manned Aircraft* by Lin, Chin E. & Lai, Ya-Hsien, a Quasi ADS-B transceiver was designed and implemented based on existing ADS-B

technology to transmit and receive flight information. The main purpose of this research was to develop a Quasi ADS-B transceiver in the CD&R context for General Aviation(GA) such as private aircraft and UAVs. The transceiver used 900MHz -XBee pro as a data transponder to broadcast flight information. It was used to broadcast and receive data packets, using standard ADS-B Compact Position Reporting (CPR) format, from its vicinity aircraft. In this research, UAVs were treated as moving objects such that they were not part of the flight control system. UAVs were considered a higher priority than manned aircraft so that all manned aircraft avoided collision with UAVs.

The 900 MHz radio station which followed the same CPR format as ADS-B was used. The Quasi ADS-B transceiver architecture was designed based on the Time Division Multiple Success (TDMA) mechanisms. Both manned aircraft and UAVs were equipped with ADS-B Out transmitters whereas only manned aircraft had ADS-B In receivers with situational awareness displays. ADS-B Out data output cycle ran every second and this cycle consisted of two sub-parts: 1) slot sorting section 2) the transmission cycle. The slot sorting section included sending a slot message which contained aircraft ID and slot number. In case of conflict of data, an automatic slot change mechanism with ID priority would be activated. In the transmission cycle, the data was transmitted for UAVs which included coordinates of the current position and next waypoint. Both data packets were broadcasted in a 50 ms slot by ID sequencing. During broadcasting, the data would be converted into binary bits by CPR and the higher-order bits would be used for the position of the aircraft.

After the transceivers were designed, they were incorporated in two ultralight aircraft to check the capability of flight data transmission. The manned aircraft was used to detect and avoid collisions whereas the UAV was used to simply fly from one waypoint to another. During the flight operation, the manned aircraft was able to detect the flight activities and intruders in its proximity and was able to sense the intrusion. It was able to detect any aircraft within 3 Km of it by using a CD&R algorithm. Two CD&R algorithms were adopted to validate the transponder performance of the transceiver in a real flight test: (1) Probabilistic Grid Detection (PGD) (2) Time and Sector Recognition (TSR). The PGD algorithm was used to detect the flight path for multiple aircraft based on the probability of the aircraft reaching the same sector at the same time. This algorithm would create the grid for each second based on predicted coordinates in a certain time interval using the Gaussian Function. The conflict prediction between the aircraft was estimated using the probability grid. TSR algorithm was used to generate the sector-shaped flight path in the tangential direction. TSR would trigger if PGD crossed the given threshold or the intruder was within a kilometer range from the aircraft.

During the experiment, the transceiver collected the UAV and GPS satellite data and this data was used in the CD&R algorithm for simulation and function verification. While the flight test experiment was carried out UAVs crashed and it was hard to locate them, so an additional function was added to detect the location of those crashed UAVs. As a result, this application was also used for search and rescue programs for injured mountain hikers. Most of the related works discussed in this paper shares a common idea of using existing ADS-B technology with some collision avoidance algorithm. In our research, we will use the ADS-B broadcasts as an input for our collision avoidance algorithm as well. In the previously discussed research papers (II and III), both used the collected data from the transceiver for simulation. However, we wish to implement live ADS-B data in conjunction with an algorithm such as the CL-RRT algorithm on UAVs for collision avoidance.

4 Implementation

In the development of the ADS-B receiver the hardware used was a raspberry pi 4 shown in figure 1a and a Nooelec R820T DVB-T RTL-SDR USB dongle as shown in figure 1b. A fully developed commercial product by Airspy called SDRSharp was used on a windows PC to verify that our hardware was functional. With the installation of appropriate drivers and SDR Sharp, the windows PC was successfully able to extract radio signals from various radio stations and playback the decoded messages. This verified that the RTL-SDR dongle was fully functional. The latest version of Raspbian was used for the raspberry pi 4 as the operating system due to its simplicity and easy-to-use interface. The Raspbian OS image was burned onto a 32GB micro sd card and then inserted into the raspberry pi.

The first encountered issue was connecting to Auburn University Wifi (AU_WiFi) because of a certificate issue. There are specific procedures that must be followed as described in the given link (https://www.eng.auburn.edu/admin/ens/helpdesk/on-campus/raspberry-pi.html). Upon successful setup to the university wifi, a base image of the Raspbian OS was created and saved to an external hard drive to be used for other future projects if needed. All latest updates and upgrades of the Raspbian OS and preinstalled software were then applied. Some essential applications and packages were also installed using the "sudo apt-get install" command: cmake, git, and build_essentials.

An online guide by Kenn Ranous was used and referenced to perform the following steps that install the needed drivers [6].

To access the RTL-SDR dongle via its USB port, a generic c library was also installed: libusb-1.0-0-dev. This library gives us access to the device but recognizes the device as a TV instead of a RTL-SDR device. For this reason, we need to install the correct drivers to use the USB dongle as RTL-SDR. The open-source drivers for SDR devices are available on Github. We decided to use a popular repository by Osmocom: https://github.com/osmocom/rtl-sdr. The following commands were used to get the repository, build, and install the drivers. git clone git://git.osmocom.org/rtl-sdr.git

```
cd rtl-sdr/mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
sudo cp ../rtl-sdr.rules /etc/udev/rules.d/
```

After this, the default driver for the RTL-SDR must be changed to the Osmocom driver that was just installed. This can be done by blacklisting the default driver. To do this, navigate to the /etc/modprobe.d folder and create a new file called blacklist-rtl. conf, and add just one line of text: blacklist dvb_usb_rtl28xxu. Save the file and restart the raspberry pi. Test the dongle in the terminal using this command: rtl_test -s 2400000. If successful a prompt within a terminal will indicate whether the RTL-SDR dongle is properly working. Then, a software-defined radio spectrum analyzer can be installed. Only open-source software was used for this work to allow for modifications wherever deemed fit and also for future work. For this reason, GQRX was chosen as the software-defined radio application.

Installation of GQRX is rather simple as it is available through the software manager. Simply navigate to Preferences ->Add/Remove Software then search for GQRX and install it. With GQRX installed, radio signals of different frequencies could be analyzed. To test, Radio signals were able to be received

and decoded from a local weather station at 162.525 MHz. At this point decoding radio signals was possible, but decoding the ADS-B broadcasts was not. MalcomRobb's forked version of dump1090 was chosen as the basic ADS-B decoder as it was open-source, could be modified, and included visualization features that can be implemented for future steps. The repository was cloned using the git clone command https://github.com/MalcolmRobb/dump1090).

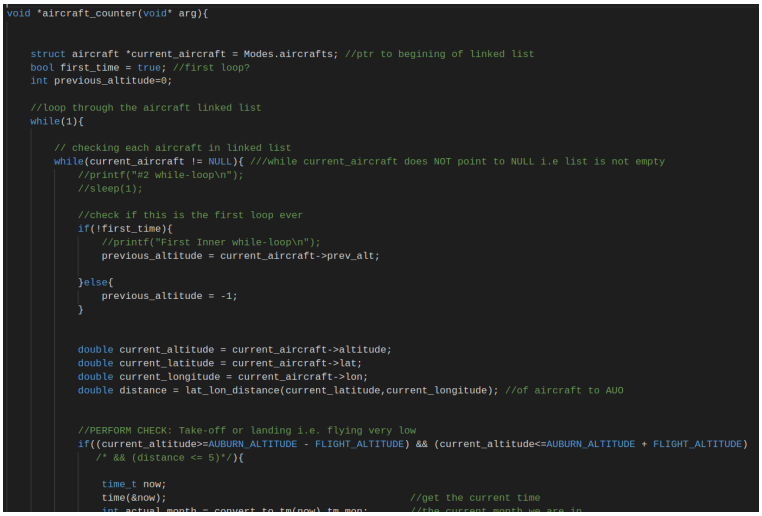
The high-level overview of this program is as follows. There is a single data structure called Modes that contains the program's global state. It contains important attributes about the current state of the program such as pointers to linked lists, current system time, mutexes, the max gain of the RTL-SDR, networking configuration, statistics, etc. There are also 3 posix threads running, each with its own specific tasks and different levels of importance. The first and most important thread is the main thread which is responsible for the overall flow and global state of the program using the Modes data structure. This thread initializes the Modes data structure, parses arguments, and decides what type of output mode that the user wants to use (i.e. debug mode, interactive mode, show incoming broadcasts one-by-one mode). It also handles the decoding of ADS-B broadcasts and puts messages in a queue.

The second thread handles RTL-SDR interaction between the program and the hardware device. The third thread handles reading the data from the message queues and populates it in the appropriate data structures. For this work modification of the existing code was necessary. In the Modes data structure, a 4th thread was added to run a function that will iterate through a linked list that contains information of all aircraft broadcasting to the receiver. The following variables were added to Modes data structure : num_takeoffs, num_landings, num_overflights,num_overflights,num_takeoffs_monthly,num_landings_monthly,num_overflights_monthly to keep track of flight operations. Other variables start_time and cur_month were also added to keep track of monthly counts.

The function Modes_init() which initializes the Modes data structure was also modified to account for these new attributes. The existing aircraft data structure format was also modified to include a prev_alt and status variable and was used to store an aircraft's altitude 5 seconds ago and store takeoff or landing status. An algorithm was then developed to recognize the different types of flight operations of aircraft based on their flying scenarios. The inner workings of our algorithm consist of two while loops. The first while loop is an infinite loop where an aircraft pointer is reset to point to the beginning of the aircraft's linked list. In the inner while loop, the whole list is iterated through one aircraft at a time until the last aircraft. In this inner loop, several if statements hit true/false which ultimately decide whether a takeoff or landing has occurred. If a plane is within +/- 500 ft from ground level, which is defined to be 767 ft, and its altitude has increased by at least 5 ft in the last 5 seconds, and it has not already been flagged as being in the air, it will be considered a take-off. Then the num_takeoffs attribute within the Modes data structure and the num_takeoffs_monthly attribute will be incremented if counting for the same month. Similarly, if a plane is within +/- 500 ft from ground level, has descended by at least 5 ft in the last 5 seconds, and has not been flagged as already on the ground, it will be considered a landing. Then, num_landings and num_monthly attributes within the Modes data structure will be incremented. Note that in the real ADS-B broadcasts when received, altitude changed by steps of 25 ft.

In the ADS-B broadcasts, an aircraft on the ground will broadcast "grnd" as its altitude. The approach of using +/- 500 ft rather than just ground level was to add versatility. If the receiver is placed at the airport, then it is accurate, but as you begin to shift it away from the airport, due to the curvature of the earth, the ground level is no longer seen and landings/takeoffs will not be counted accurately.

For overflights, a different approach was taken using an existing function. A function called interactiveR-



```
void *aircraft_counter(void* arg){


    struct aircraft *current_aircraft = Modes.aircrafts; //ptr to begining of linked list
    bool first_time = true; //first loop?
    int previous_altitude=0;

    //loop through the aircraft linked list
    while(1){

        // checking each aircraft in linked list
        while(current_aircraft != NULL){ ///while current_aircraft does NOT point to NULL i.e list is not empty
            //printf("#2 while-loop\n");
            //sleep(1);

            //check if this is the first loop ever
            if(!first_time){
                //printf("First Inner while-loop\n");
                previous_altitude = current_aircraft->prev_alt;

            }else{
                previous_altitude = -1;
            }


            double current_altitude = current_aircraft->altitude;
            double current_latitude = current_aircraft->lat;
            double current_longitude = current_aircraft->lon;
            double distance = lat_lon_distance(current_latitude,current_longitude); //of aircraft to AUO


            //PERFORM CHECK: Take-off or landing i.e. flying very low
            if((current_altitude>=AUBURN_ALTITUDE - FLIGHT_ALTITUDE) && (current_altitude<=AUBURN_ALTITUDE + FLIGHT_ALTITUDE)
              /* && (distance <= 5)*/){

                time_t now;
                time(&now);                                 //get the current time
                int actual_month = convert_to_tm(now).tm_mon;       //the current month we are in
```

Figure 2: Aircraft Counter Algorithm

emoveStaleAircrafts() removes an aircraft from the aircraft linked list if an ADS-B broadcast has not been received in over 5 minutes. A simple function called overflight_hlpr() was developed and it is called whenever an aircraft is removed from the linked list. This function first checks if the plane has already been counted as either a takeoff or landing and if not it counts as an overflight. Then, increment the num_overflights variable and also the monthly count, if counting for the same month. The figure below shows the code for the overflight_hlpr() function. The count of the various flight operations are then written to a text file as shown below.

```
=========AIRCRAFT ACTIVITY REPORT=====

-------------------------------------
THIS MONTH ( 6 )
-------------------------------------
Take-offs:       5
Landings:        16
Overflights:     95
-------------------------------------


SINCE:Mon Jun 28 10:20:18 2021
 -------------------------------------
Take-offs:       69
Landings:        61
Overflights:     619
```

Figure 3: Text file Output

To verify that the algorithm was working properly and accurately, an in-person visit to the auburn

airport to visually count aircraft landings and take-offs were conducted. It was observed that indeed the algorithm was working accurately and was incrementing flights as expected. Due to time constraints, the incorporation of this ADS-B receiver into a real UAV was not carried out. Instead, a basic framework for a simulation program was created that simulates the movement of a UAV from one start location to another destination location on a cartesian plane. The start and destination points are inputted and a function called simulate_path() then simulates the UAVs movement along a path from start to destination. Other aircraft are manually added at certain locations in the cartesian plane and are stationary. Every second another function called avoid_collision() checks to see if the UAV is within a certain distance of other aircraft. If it is within that certain distance then a collision avoidance algorithm is called. Currently, the collision avoidance algorithm is a simple function that moves the UAVs altitude up 500 ft then once reaching its destination it lowers 500 ft. If a collision is still detected after moving up 500ft, it will move up another 500 ft. The end goal for this simulation program is to incorporate real latitude, longitude, speed, and altitude from recorded flight activity and feed this data into our simulation. Before this, conversion from the cartesian plane to GPS coordinates must be done.

5 Conclusion

The ADS-B receiver design and framework for the simulation are presented in this work. An accurate aircraft counter algorithm was developed and tested in this work. Low-cost equipment and modification of existing open-source software were then used to include the aircraft counter algorithm. Auburn Airport is now able to accurately track flight operations at a low cost with this constructed ADS-B receiver. AUO no longer has to use educated guesses as a primary way of tracking flight operations. Future work of this receiver should include visualization features to plot planes on a map to make it more visually appealing to Auburn Airport staff. A website can also be created in which the receiver outputs data to the website and is accessible from anywhere. Developing a simulation framework allows for the testing of collision avoidance algorithms without using real physical aircraft. Due to time constraints, our simulation program was not completed, but a basic framework was established. FAA regulations on UAVs are expected to become stricter due to the increase in popularity so it is important to continue developing and testing collision avoidance algorithms. Because of the widespread use of ADS-B Out in aircraft, developing a collision avoidance algorithm for UAVs that uses ADS-B data would be a logical course of action. Further development of our work would include the completion of the simulation program that would allow for more robustness. More specifically, it would allow non-stationary aircraft, use of real latitude/longitude coordinates, use of real ADS-B data, and would take as input a collision avoidance algorithm. Upon successful simulated flights, The ADS-B receiver developed for Auburn Airport can be mounted on a UAV and used for collision avoidance.

6 Acknowledgement

This work is supported in part by the National Science Foundation Grant# 123456 and Auburn University Computer Science & Engineering Department. The authors would like to thank Dr. Richard Chapman and Dr. Saad Biaz for all their helpful discussion and technical support.

References Cited

[1] A. P. Cracknell, "Uavs: regulations and law enforcement," *International Journal of Remote Sensing*, vol. 38, no. 8-10, pp. 3054–3067, 2017.

[2] DOD, "Unmanned aircraft system airspace integration plan," *Archived from the original (PDF) on 2016-01-21,Retrieved 2021-06-14*, pp. 85–88, 2009.

[3] A. Krumbein, "Introduction to ads-b technology for enhanced aviation tracking and safety," *website, Accessed June 2021*, pp. 85–88, November 2016, <https://www.southwestantennas.com/articles/introduction-to-ads-b-technology>.

[4] C. Lai, Y. Ren, and C. Lin, "Ads-b based collision avoidance radar for unmanned aerial vehicles," *IEEE MTT-S International Microwave Symposium Digest*, pp. 85–88, 2009.

[5] Y. Lin and S. Saripalli, "Sense and avoid for unmanned aerial vehicles using ads-b," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6402–6407, 2015.

[6] K. Ranous, "Rtl-sdr for linux quick start guide." https://ranous.files.wordpress.com/2020/05/rtl-sdr4linux_quickstartguidev20.pdf. Accessed: 2021-06-01.

[7] W. Semke, "Analysis of radar and ads-b influences on aircraft detect and avoid (daa) systems," *Aerospace*, vol. 4.3, no. 49, 2017.

[8] M. Strohmeier, M. Schäfer, V. Lenders, and I. Martinovic, "Realities and challenges of nextgen air traffic management: the case of ads-b," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 111–118, May 2014.