EXPLOITING THE AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST SYSTEM VIA FALSE TARGET INJECTION

THESIS

Domenic Magazu III, Captain, USAF

AFIT/GCO/ENG/12-07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/12-07

EXPLOITING THE AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST SYSTEM VIA FALSE TARGET INJECTION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Domenic Magazu III, MS

Captain, USAF

March 2012

AFIT/GCO/ENG/12-07

EXPLOITING THE AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST SYSTEM VIA FALSE TARGET INJECTION

Domenic Magazu III, MS

Captain, USAF

Approved:

Robert F. Mills, PhD, (Chairman)	Date
FOR David J. Robinson, Lt Col, USAF (Member)	Date
Jonathan W. Butts, Maj, USAF (Member)	Date

Abstract

A new aircraft surveillance system, Automatic Dependent Surveillance-Broadcast (ADS-B), is being introduced by the Federal Aviation Administration (FAA) with mandated implementation in the United States by the year 2020. The rapid deployment of the system with current test-beds spread across the U.S. leaves very little chance for anyone to test the abilities of the system and more importantly the flaws of the system.

The research conducted within this thesis explores some of the weaknesses of the system to include the relative ease with which false aircraft targets can be injected. As part of a proof of concept, false ADS-B messages were successfully generated using a system comprised of GNU Radio, a Universal Software Radio Peripheral (USRP), and software developed by the author.

The ability to generate, transmit, and insert spoofed ADS-B messages on the display of a commercial ADS-B receiver, identified and exploited a weakness of the ADS-B system. Four demonstrations, conducted within an experimental environment, displayed the potential uses of the system created through this research and its associated impacts.

Acknowledgments

I would like to thank my thesis advisor(s) Lt Col David Robinson and Dr. Robert Mills for their patience and stellar support throughout the entire process of developing this thesis.

I'd also like to thank my entire family back at home. You've had a hand in the successes I've been able to enjoy and molded me into the person I am today. I am eternally grateful for all that you have done.

Last, but certainly not least, I'd like to thank my wife and son. You gave me the strength and support to pursue a monumental task and push through the many obstacles along the way. I could not have done this without you!

Domenic Magazu III

Table of Contents

Page

List of Figures

Figure Page

List of Equations

EXPLOITING THE AUTOMATIC DEPENDENT SURVEILLANCE-BROADCAST SYSTEM VIA FALSE TARGET INJECTION

I. Introduction

1.1 General Issue

Around December 17, 2009, it was reported that militants in Iraq had purportedly used $26 off-the-shelf software to intercept unencrypted live video feeds from United States Predator drones. Data captured could have provided the militants with information detrimental to US operations. The Wall Street Journal marked this incident "the emergence of a shadow cyber war within the U.S.-led conflicts overseas" [1]. Bringing this issue to modern day, imagine an adversary capable of receiving and displaying the exact location of all U.S. and allied aircraft performing missions over hostile territory. This hypothetical scenario is becoming a reality with the implementation of the latest Federal Aviation Administration (FAA) aircraft surveillance system, Automatic Dependent Surveillance-Broadcast (ADS-B).

In an effort to save fuel and money while enhancing aircraft safety, the FAA has begun actions to overhaul their traditional radar based surveillance system with a next generation (NextGen) solution based on ADS-B technology. Rather than relying on ground based radar to determine an aircraft's position, a unit onboard the aircraft determines the exact coordinates of the aircraft using the Global Positioning System (GPS) satellite constellation. That information is then automatically reported via air-to-ground and air-to-air data communication links at a fixed rate, depending on the aircraft's current state (enroute, taxiing, etc.). Because aircraft traffic volume is growing at a phenomenal rate, approaching the limits of current systems, the swift implementation of a

new system is vital. For that reason, like any new technological advancement, security may not always be at the forefront of the system's development.

1.2 Problem Statement

The implementation of a new aircraft surveillance system has far reaching implications on the commercial sector as well as military. The NextGen system significantly enhances aircraft safety and efficiency, but the security of the system is a great concern. In particular, the air-to-air communications are unencrypted and unauthenticated. One must consider the vulnerability of such a system that does not encrypt or authenticate their communications. In fact, research by McCallie et al. [2] investigated several specific attack scenarios and the severity level of each attack; however no solutions were developed to carry out these scenarios.

Commercial aircraft companies are in the process of determining how to equip their fleets with the appropriate equipment in order to meet the FAA mandate. In addition, military program offices are cautiously planning the extent of implementation with security and financial concerns at the forefront. As the mandated deadline for ADS-B implementation draws closer, aircraft are forced into compliance without answers to multiple security questions. The problem does not stop at American borders. There are already areas of the world that have implemented ADS-B. For example, the entire continent of Australia has full coverage of ADS-B and utilizes this resource for commercial aircraft surveillance [3]. Other large projects to implement the new system include China [4]. A complete look at the security of ADS-B should be instituted before the world commits whole heartedly to a new system shrouded with security concerns.

This research, in particular, presents a proof-of-concept demonstration of how false targets can be generated.

1.3 Research Objectives

The purpose of this research is to demonstrate the relative ease of generating ADS-B messages using arbitrary data. Furthermore, the system developed to generate ADS-B messages relies on inexpensive hardware and software to display the versatility one may have in designing such a system. The hardware to be used is a Universal Software Radio Peripheral (USRP) as the radio frequency (RF) front end and GNU radio as the development toolkit to build the software defined radio (SDR) application for signal processing.

Once the system is created, it will help resolve many questions proposed towards the safety and security of ADS-B. In particular, can an individual receive and display aircraft in real-time? Can aircraft messages be spoofed? If spoofing is possible, can the messages be inserted and displayed on a commercial ADS-B receiver?

1.4 Research Focus

The focus of this research is on positional ADS-B messages. ADS-B contains multiple subtype formats for broadcasting aircraft data, this research will focus on downlink format (DF) 17, subtype 5 messages. These messages contain information such as aircraft ID, altitude, latitude, and longitude, which provide the potential for false targets to be inserted on a radar display.

In addition to the 1090 MHz frequency, the FAA has approved the 978 MHz Universal Access Transceiver (UAT) link for use by general aviation aircraft flying at

lower altitudes. Although this link is used for ADS-B data communication it was not included within the scope of this research.

1.5 Investigative Questions

This research hopes to answer several questions

1. What is required to generate properly encoded ADS-B messages?
2. How can a system comprised of a USRP and GNU Radio be developed both to generate and transmit ADS-B messages?
3. If a system can be built, will it remain relatively inexpensive?
4. How can false ADS-B messages be inserted into the ADS-B network?

1.6 Approach

Using various sources, most notably, Radio Technical Commission for Aeronautics (RTCA) meeting notes [6], a system was created with the ability to generate ADS-B messages and subsequently broadcast them. In conjunction with a digital signal processing application, C++ code was written to generate the messages and perform proper preparations to transmit those messages. The process included multiple encoding equations and finally the transformation into hexadecimal representation to ensure the proper waveform for ADS-B messages.

After building the system, several demonstrations were generated and tested. The success of the demonstrations was recorded and discussed to prove or disprove the potential uses of the system created.

1.7 Implications

McCallie et al. [2] introduced some potential vulnerabilities of ADS-B. This research goes even further by demonstrating how those weaknesses can be exposed successfully. Those affected by these weaknesses include multi-billion dollar commercial airlines and military aircraft, within the U.S. and around the globe.

1.8 Preview

This document is outlined in the following manner. Chapter II provides an in-depth literature review on aircraft surveillance history, ADS-B, digital signal processing, software defined radio, and the USRP family. Chapter III describes the system development for the proof of concept system. Chapter IV contains results, discussion, and analysis of the system's capabilities. Finally, Chapter V contains conclusions, a summary of the research performed, and recommendations for future research related to ADS-B message exploitation.

II. Literature Review

2.1 Chapter Overview

This chapter presents an overview of past, present, and future Air Traffic Control (ATC) systems in order to provide background necessary to understand how the current system came into use, why this system needs to be replaced, and an introduction into future generation ATC systems. With regard to the future ATC systems, the focus of this review will be on the main area of this research, Automatic Dependent Surveillance-Broadcast (ADS-B). The chapter also includes information pertaining to software defined radio, digital signal processing (DSP), and the USRP family.

2.2 ATC History

2.2.1 Early ATC Techniques

During the early stages of flight, little to no equipment existed for in-flight navigational aids. In addition, there was no governing body within the United States to set and enforce standards, giving the appearance that improvements to navigation would not come quickly. Pilots relied on a single ground controller using flags during the day or a light at night to signal instructions to takeoff, land, or turn. Because there was no capability, such as radios, to communicate information during flight, the controller would inform the pilot of weather information or the presence of other aircraft in the same route before takeoff. Weather changes or other pertinent information could not be relayed to the pilot once they were off the ground [7].

Pilots also had to use known landmarks such as cities, railroad tracks, and/or water towers to determine their position and make any necessary corrections. This can be

compared to Visual Flight Rules (VFR) used today. VFR are established by the Federal Aviation Administration (FAA) and allow a pilot to operate the aircraft in a free-flight manner when conditions allow proper operation and control [8]. The pilot assumes responsibility for separation from other aircraft.

Shortly after these techniques, a slight improvement was introduced utilizing lighted runways or beacons set apart by a mandated distance [7]. This improved VFR flying, however, as air traffic increased each year, the need for better instruments when navigating through poor conditions (i.e. adverse weather) was essential, resulting in radar based systems becoming the navigational aid of choice.

2.2.2 Introduction of RADAR

Radio detection and ranging, also known as radar, was developed during World War II. Its design was shrouded in secrecy, but shortly after the war completed, its benefits were revealed, and the use of radar for both military and commercial use became more prevalent. Radar works through the emission of electromagnetic (radio frequency) waves by a directional or rotating antenna dish. Those radio waves then reflect off an object and return back to the source where the signal is gathered by a receiver and the target is placed on the plan position indicator (PPI), the display within the ATC tower [9]. A typical radar system is comprised of secondary surveillance radar (SSR) and primary surveillance radar (PSR). These are coupled with radio communications to form a complete ATC system. These systems came about during the 1950's which marked the beginning of radar use for ATC purposes [10].

2.2.3 Present ATC System

During the late 1950's, the U.S. government became involved with air traffic management and regulation through the Federal Aviation Act of 1958, signed by President Eisenhower [22]. This Act created the Federal Aviation Agency which later became the Federal Aviation Administration (FAA). Since that time, the FAA has taken action to administer and enforce regulations, develop and maintain the national infrastructure, and monitor the air picture across the United States of America. To date, the FAA controls 750 ATC facilities, 18,000 airports, and 4,500 air navigation facilities providing complete coverage of the National Airspace System (NAS) [2]. The cost to maintain and operate these facilities is quite large. Approximately 150 million dollars is spent each year in operational and maintenance costs by the FAA [11]. As the systems continue to age, maintenance costs continue to escalate, which is a primary motivation for ADS-B implementation.

As discussed earlier, most modern ATC systems are comprised of PSR and SSR. PSR works solely on the signal reflecting off an aircraft or other object and the reflected signal returning to the receiver, normally positioned in the same location as the transmitter. The orientation of the radar antenna determines the bearing of the aircraft relative to the radar site, while the distance is determined by the length of time it takes for the signal to be emitted and for the reflected signal to return to the origin of the transmission [12]. PSR is completely passive and does not require any action from the object being tracked. The main disadvantage of PSR is that it can reflect off birds, ground objects, and atmospheric phenomena, which can cause issues for air traffic

controllers. Due to this disadvantage, PSR has been augmented with Air Traffic Control Radar Beacon System (ATCRBS), also known as SSR.

SSR employs special equipment onboard the target aircraft. Interrogation signals are sent from the ground station antenna just as with PSR. Once an aircraft receives the interrogation signal, a transponder aboard the aircraft returns a coded reply signal containing information such as aircraft identification and altitude. Because the aircraft transmits the reply message, SSR provides greater signal strength which translates to greater range and improved performance. SSR also decreases the power needed by the ground station transmission since the signal will not need to reflect and return [Figure 1].

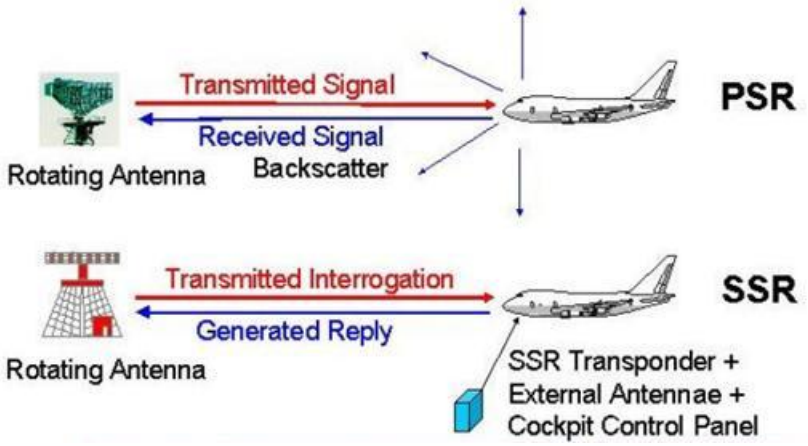


Figure 1 - Graphical representation of PSR and SSR [13]

One drawback of SSR is its dependent nature, with dedicated equipment on the ground and on the aircraft. In most cases, PSR and SSR will be coupled together to accommodate both aircraft with and without transponders. Additionally, PSR is required for air defense, detecting non-cooperative targets.

2.2.4 Current ATC Limitations

Current radar systems are quickly reaching their maximum capacity due to shortcomings in visibility propagation, limitation of line-of-sight, limitation of voice communications and the lack of digital data links [14][15]. Additionally, synchronous garbling is another concern in which the interrogation signal of SSR invokes a response from more than one aircraft, causing the replies sent by both aircraft to overlap at the receiver leading to loss of information at the ATC facility.

Compounding the interference of messages is the rapid growth of air traffic. Air traffic has increased 32% within the last decade and is projected to nearly double by the year 2028 [Figure 2]. Currently, air traffic controllers handle 9 to 15 aircraft at any one point. With the current radar system and projected increase in air traffic, experts believe controllers could be required to handle approximately 45 aircraft at any one point, a situation that is infeasible to manage and completely unsafe [16] [17].

Figure 2 - Rapid air growth over the next decade and a half. Expected to double air traffic by 2025 [18].

2.3 NextGen Solution

2.3.1 FAA Decision

With the release of the FAA's final decision [19], they solidified their choice to move forward with the implementation of ADS-B technology as part of the next generation (NextGen) national airspace system. In addition, in August of 2007, the FAA awarded the International Telephone and Telegraph (ITT) corporation a contract to field a national system capable of providing a comprehensive set of ADS-B services for the United States [20] [21]. Because the FAA is charged with providing a safe and efficient airspace for both civil and military aircraft [22], the armed services are also affected by the FAA's decision.

2.3.2 ADS-B Around the World

ADS-B is also being adopted on a global scale. Europe has a slightly more aggressive implementation plan with hope of wide spread implementation by 2015 as compared to 2020 for implementation within the U.S. [23] [24]. ADS-B standardization in Europe is being driven by the Requirements Focus Group (RFG), containing members from the European Organization for the Safety of Air Navigation, FAA, European Organization for Civil Aviation Equipment, Radio Technical Commission for Aeronautics (RTCA), and participation from nations such as Australia, Canada and Japan [23]. The RFG is also working closely with the International Civil Aviation Organization (ICAO) to ensure global interoperability for ADS-B. ICAO codifies the principles and techniques of international air navigation and fosters the planning and development of international air transport to ensure safe and orderly growth.

2.3.3 NextGen Components

NextGen is comprised of several components working simultaneously to collect information and disseminate it via a broadcast transmission [Figure 3]. Its main source of information is from the constellation of GPS satellites. GPS sensors aboard an aircraft determine its location, which is then joined with information from the aircraft's navigational system or Flight Management System (FMS). The FMS provides information such as the flight plan to ensure the aircraft is following its designated path.

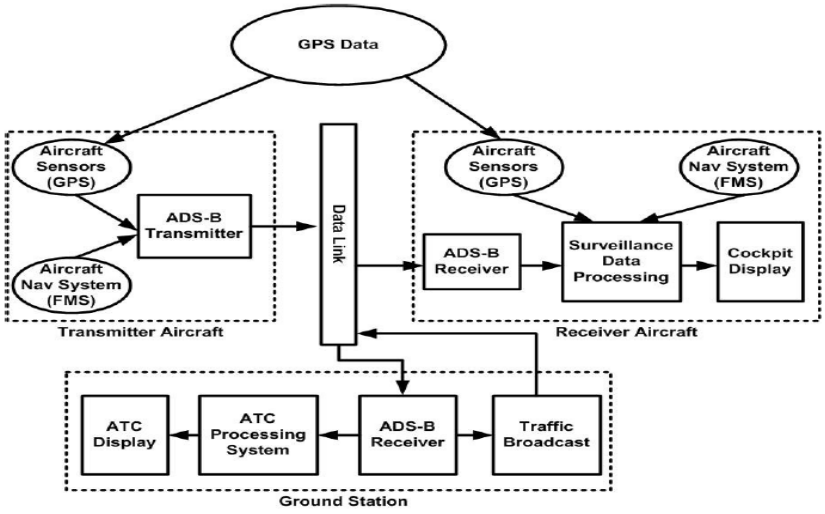


Figure 3 - Major components of the ADS-B system [2]

The FMS and GPS data is fused and broadcasted to other aircraft and ground stations. Both receiving aircraft and ground stations perform similar operations on the message to interpret and display the information on the control display unit (CDU) [2].

2.3.4 ADS-B Explained

ADS-B's main purpose is to determine the position of an aircraft and then broadcast that information, along with its altitude, call sign, heading, and aircraft type automatically (i.e., without an SSR interrogation signal) to other aircraft and to air traffic control ground facilities. ADS-B is *automatic* in that it does not require any action or input by the pilot and there is no interrogation from the ground required. It is also *dependent* because it relies on onboard equipment to gather the ADS-B data and *broadcast* it to other ADS-B users and it is a means of providing *surveillance* and traffic coordination.

ADS-B was created with compatibility and ease of transition in mind. It was built using similar aspects of the current aircraft surveillance transmission mode called Mode S or mode select. Mode S operates by interrogating aircraft by a specific aircraft identification number. Only the aircraft possessing the correct identification number will reply to an interrogation with its flight information, eliminating issues with synchronous garbling. Transmission types prior to Mode S include Modes A and C. Mode A provided aircraft identification and Mode C provided altitude. Mode S provides greater capabilities, primarily in the form of aircraft information to include identity, intent, capability and location [25].

ADS-B is similar to Mode S in that it uses the same transmission frequency of 1090 MHz. It differs in that the message is 112 bits, 120 μs long and are "squitter" messages [17] [26]. A squitter message is simply a transmitted message not invoked by any interrogation [Figure 4].

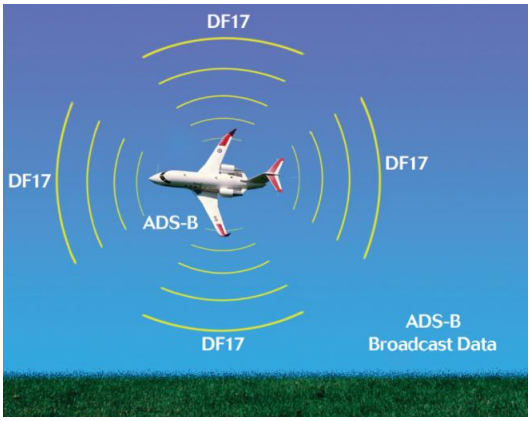


Figure 4 – Broadcasting of ADS-B squitter message [27]

As shown in Figure 5, 56 of the 112 bits are for ADS-B specific data to include altitude and airborne position (latitude and longitude). The remaining bits are used for message format, aircraft address, parity check, and finally a few bits for the transponder communication capability.

Figure 5 - Diagram showing the fields of the ADS-B message [2]

Pulse Position Modulation (PPM) is used for encoding and transmitting the message [26] [29]. The PPM for each pulse results in the data occupying either the first or second half of the entire pulse, which in effect is the same as Manchester encoding [Figure 6].

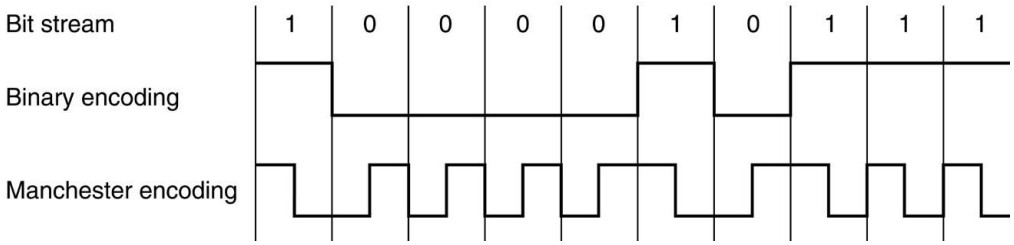


Figure 6 - Manchester encoding

There are two types of equipment for ADS-B messages, with one being *ADS-B In* and the second being *ADS-B Out*. ADS-B Out provides the ability to broadcast messages to ground stations and to other aircraft within the appropriate receiving range. As demonstrated by the Los Angeles ADS-B trials, air-to-air reception ranged from 0 to 110 nautical miles [30] [31] [32]. ADS-B In is responsible for receiving, decoding, and displaying messages within the cockpit or ATC tower. As mentioned previously, by 2020, all aircraft flying within the United States will be required to be equipped with ADS-B Out. Currently there is no mandate for ADS-B In onboard aircraft, but without the usage of this equipment, many of the advantages of ADS-B cannot be utilized.

2.4 Advantages of NextGen

One of the biggest advantages of ADS-B is the ability to provide coverage where radar could not reach before. The primary area in which this is relevant is transoceanic navigation [33]. With current systems, the radar picture is limited by land based radar. ADS-B overcomes this limitation through the utilization of GPS satellites, broadcasts from other aircraft, and ground stations to generate its air picture. Strategically placed

broadcast stations provide the ability to receive nearby transmissions and broadcast them out to anyone within range. Therefore with careful placement of ATC facilities, aircraft will maintain the air picture around them, providing greater accuracy, resolution, integrity and safety.

Related to coverage, an added advantage is the smaller footprint of ADS-B facilities. This allows the FAA to deploy ADS-B transmitters/receivers on structures such as oil rigs many miles out from land which can then act as Automatic Dependent Surveillance-Rebroadcast (ADS-R) stations [37] [Figure 7]. ADS-R receives ADS-B position broadcasts and rebroadcasts the information to near-by aircraft [34] [35]. In addition to a smaller footprint, operation and maintenance of ADS-B equipment and facilities will be significantly cheaper. The average radar-based facility costs approximately $1-$4 million dollars while an ADS-B site will cost approximately $100-$400 thousand [36].

Figure 7 - ADS-B equipment aboard an oil rig [37]

Another advantage of ADS-B is more precise management of aircraft on their approach for landing. As described by Randy Babbitt, the FAA Administrator, ADS-B provides "greater precision and reliability" [38]. Because the aircraft transmit location every second, data flows at nearly real-time, as compared to updates of five to ten seconds for traditional radar. The controller can take advantage of this feature and are able to reduce the amount of space between each aircraft, thus increasing the airport's throughput [39] [40]. A 2000 FAA study, demonstrated how aircraft could reduce their separation from the current standard of 4300 feet to only 750 feet [10]. This is a drastic change from the current methods which focus on aircraft flying a more rigidly structured and uniform flight path [41]. The ability to decrease separation allows aircraft to perform continuous descent landing, as opposed to the current stair-step practices [42]. In the stair-step method, aircraft alternately descend and then level off by accelerating engine speed. These short bursts of engine speed burn enormous amounts of fuel, increase noise, and cause more emissions. The elimination of the stair-step process could result in airlines saving millions of dollars in fuel costs. The United Parcel Service used ADS-B for one year and realized a savings of 250,000 gallons of fuel. The fuel savings generated additional benefits such as a 30% reduction in emissions and a 34% reduction in noise [43] [44].

Widespread usage of ADS-B also introduces the concept of Free Flight [45] to the NAS. The Free Flight model could introduce further savings in fuel as well as faster flight times for passengers with the ability of aircraft pilots to formulate a more direct route from their point of origin to the aircraft's destination. Aircraft routes would no longer be so rigidly structured and uniform or determined by ground controllers before

the flight takes place [41]. This is the result of more frequent reporting of the ADS-B surveillance system (reports every 1 second), *ADS-B In* technology, and the ability to receive precise aircraft position data in locations where PSR and SSR could never reach.

In addition to these positive results, Alaska's Capstone program, an experiment in testing ADS-B technology and its effect on air traffic controller workload, produced even more optimistic results. During the trial period, 208 aircraft were equipped with ADS-B. Normal flights in and out of the Alaskan region were monitored. After program completion, surveys of controllers found that 57% said they had spent less time providing Instrument Flight Rules (IFR) separation services, and 79% felt their overall efficiency increased with ADS-B. Additionally, ATC saw a reduction of 18% in controller communications while at the same time reducing the fatal accident rate by nearly half [42] [46].

Providing a better air picture to the pilots is yet another advantage. An aircraft with both ADS-B Out and ADS-B In can provide pilots with a more fluid response to situations since they have a clear picture of the air as well as the ground without relying on voice communications between themselves and air traffic controllers, as is the current case with radar-based systems. This allows for faster response and corrective actions while providing the pilot with strong confidence in the air picture.

Finally, ADS-B has the potential to augment the Traffic Collision Avoidance System (TCAS) [47]. Although TCAS is not at the end of its life cycle, ADS-B operates in a similar fashion, giving the systems the capability to enhance their ability to avoid mid-air collisions. TCAS works by interrogating aircraft (similar to SSR) within the vicinity and tracking aircraft by their replies to those interrogations. TCAS then

determines if the aircraft has entered the 'protection volume' or safety zone of another aircraft. If an aircraft has entered that zone, a traffic advisory is sent to the aircraft. If the aircraft does not make the necessary correction a resolution advisory (vertical maneuver command) is released to avoid a mid-air collision [48]. Not only will ADS-B provide mid-air collision avoidance, but aircraft continue to transmit ADS-B messages while on the ground. This provides surface surveillance for runway incursion avoidance and ramp management [49]. Although many advantages exist, like all new technologies, there also exist disadvantages or vulnerabilities.

2.5 Vulnerabilities of ADS-B

ADS-B consists of GPS for an aircraft's coordinates, a transponder, a barometric altimeter, and the associated wiring to connect them. While each of these components offer potential avenues of attack, most are beyond the scope of this research and will not be discussed further. The attacks to be focused on are those aimed at exploiting the ADS-B messages being transmitted and received by an aircraft. Additional reading on GPS failures and information integrity can be found at [50] [51] [52]. Other sources [14] [52] outline ADS-B message attack scenarios in which the system could be exploited by nefarious users which will be described in the following paragraphs.

2.4.1 Passive Monitoring

In December of 2009, the Wall Street Journal reported that insurgents had purportedly intercepted live video feeds from unmanned aerial vehicles controlled by the United States military [1]. A related concern with ADS-B is the ability for any individual to purchase equipment that is capable of receiving and translating ADS-B messages

providing a mechanism to passively monitor aircraft for possible ill intent. Dick Smith, former chairman of Australia's Civil Aviation Administration, explained how terrorists with a laptop, transponder and antenna could intercept ADS-B messages and utilize them to track aircraft of law enforcement, high profile politicians, commercial aircraft, or military aircraft [53].

Additionally, with the advent of smart phone markets, applications have been released that will plot an ADS-B equipped aircraft on a map. The user can download this application for free and monitor their local air traffic. The paid version of this application provides additional information such as live aircraft movements, flight path contrails, the ability to search for an aircraft by flight number, aircraft altitude, speed, plane type, and transponder "squawk" codes. This application costs a user only $3.99. Applications of this sort are purely passive and only allow an individual to monitor without affecting an aircraft's display or causing any harm to an ATC facility. The next section discusses actions that will lead to the alteration of messages or the display aboard an aircraft or at an ATC facility.

2.4.2 Active Attacks

The first type of active attack on the ADS-B system is the ability for a malicious person to jam the signal at the ground station. There will be approximately 800 ground stations placed 150 to 200 miles apart [19] [20]. Development of facilities for performing ADS-B communications have already begun and completion is projected for 2013 [42]. Locating a ground station and gaining close proximity to it will likely not be difficult because they are numerous and their general locations have already been designated [Figure 8].

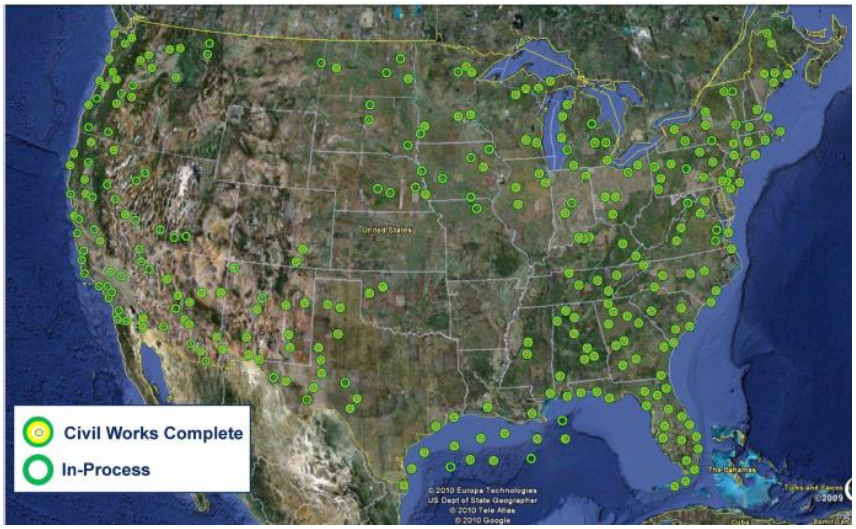


Figure 8 - Map showing ADS-B ground facility status [37].

Related to this attack is jamming a specific aircraft. Although an attacker could jam a single aircraft's receiver from the ground, once the aircraft goes beyond line-of-sight it will no longer be susceptible to the attack.

Another active attack vector is the injection of ghost targets on the display of aircraft as well as ground stations. The user would need to transmit the 112 bit ADS-B message, which would then be received by the ground station and displayed for pilots and air traffic controller(s).

Spoofing aircraft on a ground station facility will not yield high success as the facilities will use primary surveillance radar to verify the ADS-B messages [28], however this verification cannot be implemented aboard an aircraft [54]. Because there is no current methodology allowing interoperability of ADS-B and legacy radar systems on an aircraft, what is shown on the display cannot be verified or corrected [55][56]. A

controller within an ATC facility also has the ability to reference logged flight plans to determine if a target is a ghost inject or not. Again, the ability to perform this cross check is not available to pilots. Suggestions have been made through prior research that signal tracking, ADS-B data filtering, evaluation directional characteristics of an ADS-B signal, track change reports, and flight intent, can provide information validation, eliminating the ghost inject scenario [57]. Because these validation techniques are not currently being employed they will not be discussed further.

Although it is out of the scope of this research, something of importance to note is the ADS-B system will use the backbone infrastructure of AT&T to connect the air traffic management picture. This provides a vector through the Internet to infiltrate and/or disrupt the air traffic control system. Access to the system could grant a malicious user the ability to disrupt, destroy, or deny ATC services.

2.6 ADS-B Test Equipment

Many companies have begun production of light-weight and inexpensive hardware for both aerial enthusiasts and pilots alike to become familiar with the new ADS-B system. One such piece of equipment, the SBS-1eR, is made by Kinetic Avionic Products Limited based out of the United Kingdom. The SBS-1eR is about the size of the human hand and provides many features to include tracking of Mode-S/ADS-B equipped aircraft, built-in air traffic and FM radio, and software capable of being run on a laptop or PC with Windows operating system. The software provides a virtual radar screen displaying any aircraft broadcasting ADS-B in the area. Figure 9 is a screen capture of the virtual display generated by the software, showing a location within Ohio,

currently tracking two aircraft.

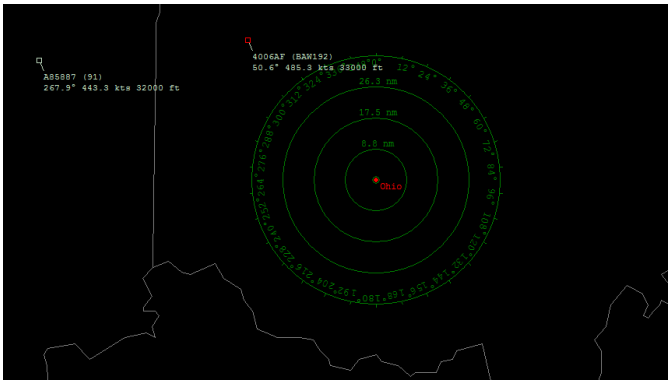


Figure 9 - SBS-1eR virtual radar screen currently tracking two aircraft

Along with displaying the aircraft on the "radar screen", additional information can be obtained for a given aircraft to include altitude, speed, exact location via GPS coordinates, country of origin, planned track, and aircraft identification in hexidecimal format [Figure 10].

Figure 10 - Aircraft information displayed by SBS-1eR

The SBS-1eR will be used throughout this research for monitoring live and ghost inject traffic. This equipment will be used to validate the proof of concept system proposed by this thesis.

2.7 Universal Software Radio Peripheral

The USRP is a flexible USB or Ethernet device capable of connecting to a laptop or PC [Figure 11]. The USRP provides a front end for software defined radio applications. It is made up of a motherboard which contains four digital-to-analog converters, four analog-to-digital converters, a field programmable gate array (FPGA) discussed in the next section, and a USB or Ethernet controller [Figure 12]. Additionally, the USRP motherboard has the ability to support four daughterboards, two for receive and two for transmit.

Figure 11- USRP N200

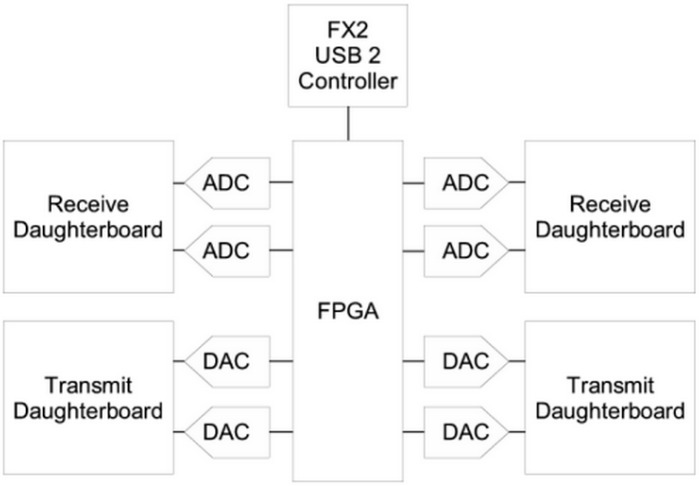


Figure 12 - USRP block diagram

Both the USRP and associated daughterboards are developed by Ettus Research, LLC. The daughterboards vary greatly in their capability, with some performing receive actions and others performing both receive and transmit functions, also called transceivers. The daughterboards allow a software defined radio application to operate at various frequencies depending on the specific board purchased [58].

2.8 FPGA/Digital Down Conversion

One of the main components of the USRP is a Xilinx FPGA. The FPGA performs the process of digital down conversion (DDC). This involves taking a band pass RF signal and mixing it to a lower frequency, which lowers the sample rate while retaining all information. DDC is used when the signal of interest occupies a small portion of the entire bandwidth. To illustrate the process, take a signal at the 45-47MHz range, the band of interest is only 2MHz. The RF signal would be digitized at a sample rate of 100 million samples per second if it were not down converted. This follows the

Nyquist theorem which states the sampling rate should be no less than two times the rate of the highest frequency [54]. DDC brings the signal of intent down to baseband, a signal measured from zero hertz to a cut-off frequency which would be 2 MHz for this example. At this point the baseband signal can be sampled at a much lower rate, on the order of 4-5 MHz. DDC adds the advantages of simplifying any further signal processing on the data, allows more processing to fit within the FPGA, and reduces the power requirement of the FPGA.

2.9 GNU Radio Basics

GNU Radio is free software that provides a framework for developing software defined radio applications using readily-available, low-cost external RF hardware and commodity processors. In addition, the recommended hardware for GNU Radio interface is the USRP.

GNU Radio uses the Python programming language for high-level organization, policy and GUI control, while using C++ for performance-critical signal processing blocks. Different source files are released for free download on the GNU Radio wiki page. Implementing the different software radios is as easy as downloading the source files and installing on a Linux based machine [59].

One such release was provided by Nick Foster, an employee of Ettus Research, who developed a simple Mode S/ADS-B receiver. It was released in October of 2010 for download from the GNU Radio site. It has the ability to decode and display ADS-B messages to the command window. Once decoded, the output can be exported to

multiple interfaces to include keyhole markup language (KML) for Google Earth, which is a popular application for plotting data in three dimensional space.

2.10 Summary

Using the information gained through this literature review an individual now has the general knowledge to begin the development of an ADS-B message generator as well as an ADS-B transmitter. In the following chapter, the author will discuss the methodology in building such a system.

III. Methodology

3.1 Chapter Overview

This chapter will describe the methodology used in designing a system capable of both receiving and transmitting ADS-B messages. The software, hardware, and coding will be discussed as well as the limitations experienced throughout the process of creating the system. The system described will then be used to demonstrate that ADS-B messages can be spoofed causing ghost injects to be inserted and displayed on the commercial ADS-B receiver.

3.2 Hardware and Software

3.2.1 Computer

The computer used during these experiments was a Dell® Precision 690 with Intel® Quad-Core Xeon processers operating at 3.00GHz. The choice of this system was based on availability. When designing a system similar to this, considerations should be made in the processing power of the computer used. The processer operating at 3.00GHz will provide enough calculations per second to meet requirements for this system. All modern processors should handle the calculations for digital signal processing as the majority of processing will occur within the USRP.

3.2.2 Operating System

The operating system chosen was Ubuntu 10.10, Maverick Meerkat, released in October of 2010. Ubuntu is an open source operating system built around the Linux kernel. Ubuntu 10.10 has proven to be an extremely stable version and much of the development for GNU Radio has taken place on this version, hence the decision to use it.

Ubuntu operating systems are freely available at ubuntu.com. There are multiple ways to install the operating system, all of which are described at the Ubuntu website. The install for these experiments were accomplished through disc boot. It is assumed that the reader has a working knowledge of Ubuntu Linux and specific details for installing/configuring are not provided here. The next step is deciding on a platform for building SDR applications.

3.2.3 SDR Design Platform

GNU radio was chosen for the experiments in this thesis. It is written using two different programming languages. The higher level language to create SDR applications is written in Python. The lower level language which will perform the critical signal processing is C++. This SDR toolkit was chosen for the following reasons. GNU radio has a very large following and comes with pre-configured signal processing blocks for generating SDR programs, alleviating a significant portion of the learning curve required to transmit and receive radio frequency (RF) signals. In addition to having pre-configured processing blocks, the toolkit is completely free for use and experimentation. A final reason was the availability of previously developed code for receiving ADS-B messages using the standard GNU radio library [60]. This program, called *GR-AIR-MODES*, provides insight into the decoding of ADS-B messages and helps to provide the foundation for creating, manipulating, and transmitting ADS-B messages and its associated data.

3.2.4 USRP

The USRP was chosen for its great flexibility and modular architecture. The USRP family is developed and sold by Ettus Research, LLC. This device provides the

front end for the SDR application, the two specific devices used are the USRP N200 and USRP2 [Figure 13]. The USRP2 was released in September of 2008. The specifications of the USRP2 include a Xilinx Spartan 3-2000 FPGA, gigabit Ethernet interface, two 100MS/s, 14 bit analog-to-digital converters (ADC), two 400 MS/s, 16-bit digital-to-analog (DAC), and an SD card reader. The USRP N200 is very similar to the USRP2 with the only differences being the RF bandwidth increasing to 50 MHz from 25MHz and an onboard Flash memory instead of an SD card for firmware and configuration. This allows for easier programming over the network.

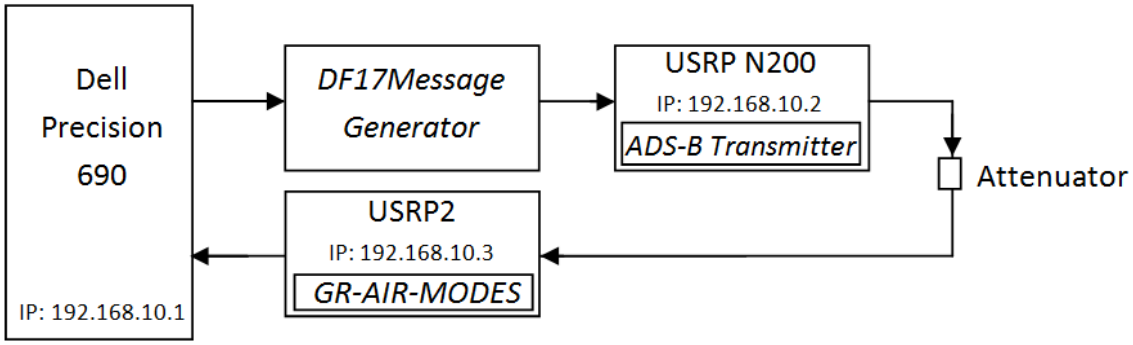


Figure 13 - System block diagram for testing/troubleshooting

3.2.4.1 USRP Daughterboards

With each USRP the appropriate daughterboard must be installed to provide the required RF coverage. The daughterboards are also manufactured and sold by Ettus Research. Since ADS-B messages are transmitted at the 1090MHz frequency the WBX daughterboard was chosen [Figure 14]. The specifics of the WBX daughterboard include a frequency range of 50 MHz to 2.2 GHz, transmission power of 30 to 100 mW, and dual

synthesizers for independent transmit and receive frequencies. Both the USRP2 and USRP N200 will be using the WBX daughterboard.

Figure 14 - WBX daughterboard

3.2.4.2 USRP Firmware

The USRPs were configured with the latest version of the Universal Hardware Driver (UHD) for better compatibility between the computer system and the USRP devices. The UHD provides a single driver that can be used across all USRP devices. Prior to the integration of UHD, each USRP device had its own specific driver and the ability to communicate between USRPs was extremely difficult if not impossible [59]. To use both UHD and GNU Radio a bash script is freely available at http://code.ettus.com/redmine/ettus/projects/uhd/wiki. In order to install the latest UHD and GNU radio versions simply copy and paste the code to a shell file within the home directory. Through the command line navigate to the location of the file and perform an execution command (./UHDandGNURadio.sh). The script will download the most current software and complete all installation steps.

3.2.5 Testing GNU Radio and USRP Install

To ensure proper installation of the above items, the *UHD+GNURadio* build provides some useful tools to include 'uhd_find_devices', which finds all properly configured URSPs and another application 'uhd_usrp_probe', which provides all specifications of those USRPs, to include firmware and daughterboard information. These applications can be found within the directory of the *UHD+GNURadio* installation within the /uhd/host/utils directory. When running these scripts, be sure the devices are connected to a gigabit Ethernet port or they will not be recognized on the network and errors will result. When networking USRPs, be sure to set the Internet Protocol (IP) address to 192.168.10.#. Setting the IP address can be accomplished through another tool installed with *UHD+GNURadio*. Appendix A contains information on the USRP device configuration settings and how to change them.

3.3 Commercial ADS-B Receiver

In order to validate the information received, a commercial device was also purchased called the SBS-1eR. The SBS-1eR, developed by Kinetic Avionic Products Limited, is a commercial device developed to receive and decode ADS-B messages. The device also contains a built in very high frequency radio receiver for listening to air traffic communications. It has an extremely small footprint and can connect to a PC or laptop via USB or Ethernet [Figure 15].

Figure 15 - Front and rear view of SBS-1eR

The SBS-1eR also includes Basestation Virtual Radar software, which provides a virtual radar screen on the PC for viewing aircraft movements [Figure 16]. The display will provide information such as aircraft ID, altitude, latitude, longitude, callsign, and much more within the right side of the display. The aircraft's position will be displayed on a virtual radar screen within the top left corner and finally the altitude in the bottom left corner.

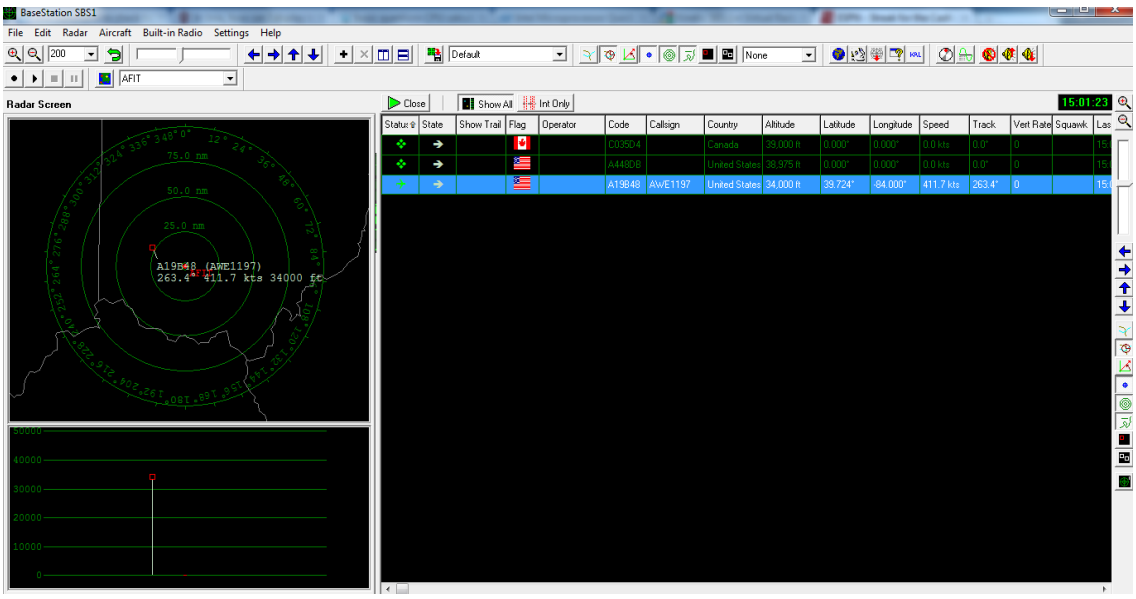


Figure 16 - SBS-1eR virtual radar screen currently tracking an aircraft

3.4 USRP ADS-B Receiver Install

There are two methods used within this research to receive ADS-B messages. The first is the SBS-1eR, which will display the aircraft on its software-based virtual radar screen. The second method is to download, build, and run the *GR-AIR-MODES* program. The subsequent paragraphs will go into the process of building the receiver and some detail on the options the program provides to the user.

Similar to the UHD and GNU Radio build script, the ADS-B receiver program is available for free and is easy to download through a git repository [60]. Github is a web-based hosting service for software development projects. Developers can store their code on the website and make changes as needed, which can then be downloaded by users of the application. Using a simple command will download all the necessary files for building and running the program :

git clone git://github.com/bistromath/GR-AIR-MODES.git

After completion, a *GR-AIR-MODES* folder is located in the current directory. If any errors occur they will be displayed within the command prompt window. All errors should be corrected before continuing. In order to begin receiving data, a USRP must be functioning properly with the latest UHD version and a proper daughterboard, capable of receiving within the 1090MHz range.

After all hardware and software has been installed and compiled correctly, the program can be run by navigating to the *GR-AIR-MODES* folder and opening the src/python directory. Simply executing the uhd_modes.py file will begin the capturing of

ADS-B/Mode-S messages. There is no GUI, but messages will be displayed within the command prompt showing all relevant data fields [Figure 17]. If an error is received the USRP may have a different IP than the program expects. Simply open uhd_modes.py in a text editor. Line 63 of the code sets the IP of the USRP source, change it to the IP set during the steps followed in section 3.2.5.

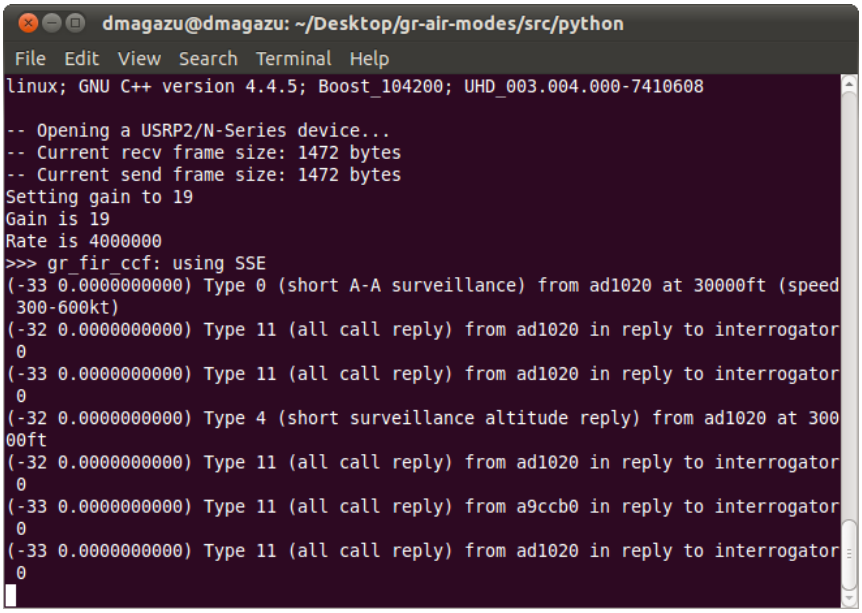


Figure 17 - Command window displaying the *GR-AIR-MODES* program running

There is also a command line switch, -K, that will allow the user to 'record' all data captured to a KML file. KML uses eXtensible Markup Language to encode geographic modeling data. KML files can then be used within Google Earth. After running the *GR-AIR-MODES* program with the –K option invoked, open Google Earth and import the KML file. All aircraft broadcasting ADS-B messages will be displayed

on the three-dimensional map. Although elementary, be sure to choose the correct antenna before running the program. The antenna can be chosen through the –A command followed by the antenna name. With the WBX daughterboard, there are two options, the receive (RX) only antenna or the transceiver (TX/RX). There is no difference between the two antennas, but the USRP will default to RX. If the antenna being used is not connected to RX, it may give the appearance the program is not receiving ADS-B messages.

3.5 Demodulating the Pulse Train

As discussed in Chapter II, a typical ADS-B message is 120 μs long and uses PPM for transmitting the bits. PPM is a form of signal modulation where the bits are transmitted in one of two possible time slots; for ADS-B the time slot equals one microsecond [Figure 18]. A "1" or "0" is denoted by a pulse in the first or second half (0.5 μs) of the time slot, respectively.

3.5.1 Preamble Detection

The first step in demodulating the pulse train is to determine the presence of a signal. The first four pulses in Figure 18 are considered the preamble. The preamble serves as a flag and time sync, notifying a receiver that a message will begin 8μs from the first preamble pulse. The preamble does not follow conventional PPM principles (a pulse in every time slot). A method for overcoming this deviation is to represent the gaps as consecutive "0" bits.

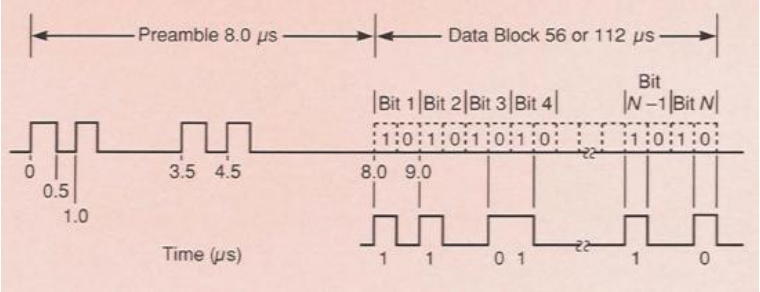


Figure 18 - 120 microsecond ADS-B message waveform

3.5.2 Message Reception

After the preamble has been detected, the individual bits need to be identified for message decoding. The bits are detected through a threshold setting based on the pulse amplitudes of the preamble. This requires the pulses to remain at relative constant amplitude throughout transmission. Pulses violating the threshold could cause bit errors and ultimately the message would be discarded. Once the bits have been received successfully, *GR-AIR-MODES* begins to decode the bits based upon the message type and the message format. DF17 messages have a particular placement for the individual components of the message. To ensure proper decoding, the format must be known to ensure the correct bits are used for altitude, aircraft ID, etc. The ADS-B message is broken into six parts including the preamble [Figure 19].

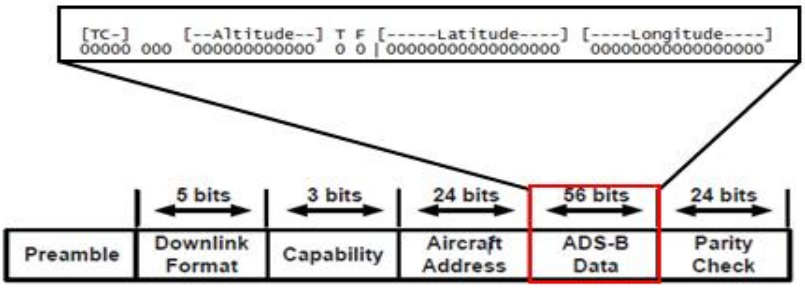


Figure 19 - ADS-B message format. Modified from [2].

The first field of the message is the DF type; as discussed ADS-B messages are set to 17 or 10001 binary. The next field is the capability field or subtype field. This field describes the specific data being transmitted by the ADS-B message. The subtype message focused on for this research is subtype five, positional reports. This field will be set to "101" for all tests. Following the capability field is the aircraft's ID, which is a three byte field that contains the ICAO designation for each aircraft. The aircraft ID is assigned for the life of the aircraft but can be changed if necessary. Following the aircraft ID are the 56 bits of ADS-B data, containing information about the altitude, latitude, and longitude of the aircraft. Finally, the last field is the parity check, which is 24 bits. The parity bits are obtained using a cyclic redundancy check polynomial applied to the first 88 bits of the message.

3.5.3 Downlink Format and Aircraft ID Decoding

The first two fields can be converted to decimal format from its current binary format to determine the DF type and capability field. Because the aircraft ID is in hexadecimal notation (6 hex characters) no decoding is necessary. Decoding the altitude, latitude, and longitude requires multiple calculations and will be discussed next.

3.5.4 Altitude Decoding

The altitude is comprised of 12 bits (bits 41-52). In order to decode the altitude, bit number eight of the 12 bits, counting from left to right, is removed. This is known as the Q bit. This bit determines whether the altitude is being reported in 25 foot increments (set to '1') or 100 foot increments (set to '0') [Figure 20]. After determining the increment value, the first seven bits are shifted to the right, essentially eliminating the Q bit. This will leave a binary number that can now be converted to decimal. Once in decimal format, the number is multiplied by the increment (25 or 100), and the final step is to add 1,000 to it.

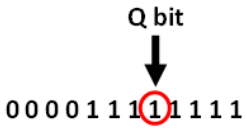


Figure 20 - Q bit representation for altitude

For example, assume the altitude bit pattern is 000011111111. Since the Q bit is set to one, the altitude is being reported in 25 foot increments. The Q bit is then eliminated, which leaves 000001111111 or 127 decimal. Next, the decimal number is multiplied by 25 for a product of 1,175. Finally, 1,000 is added to give a decoded altitude of 2,175 feet.

3.5.5 Latitude and Longitude Decoding

The next decoding to take place is the latitude and longitude. These two fields are each 17 bits. There is an additional two bits designated for Compact Position Report

(CPR) format, which makes 36 bits total for this information. CPR was developed for ADS-B messages to reduce the number of bits required to transmit positional data. Accuracy for ADS-B messages is within 5.1 meters, since the Earth's circumference is around 40,000 kilometers that would require almost 7.8 million position values (40,000m/5.1m) or 23 bits of data (2^23 = 8,388,608). In order to fit the positional data into 17 bits, the higher order bits are eliminated. The higher order bits contain information such as the hemisphere in which the aircraft is located. Since an aircraft in most cases will remain within the same hemisphere for the lifetime of the aircraft, they can be eliminated. In order to accomplish this without causing ambiguity, the Earth is divided into odd and even latitude and longitude zones [Figure 21]. In order to determine the precise location, both odd and even CPR messages must be received. These two formats differ slightly in their calculations, but will be used to narrow down the location of the aircraft to an X and Y coordinate within one zone to within the 5.1 meter accuracy mentioned earlier.

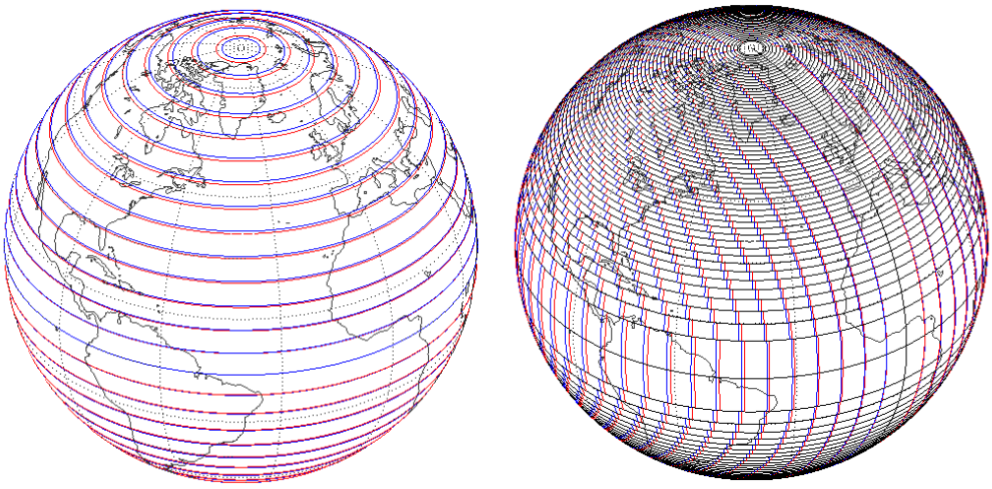


Figure 21 – Latitude (left) and longitude (right) zone boundaries [6]

The equations throughout the decoding process were found at [61]. The first step in decoding is to calculate the latitude index designated by the letter *j* as shown in Equation 1. *Lat(0)* and *Lat(1)* are the odd and even format messages received.

$$j = \left(\frac{59 * Lat(0) - 60 * Lat(1)}{131072}\right) + \frac{1}{2} \quad \textbf{(1)}$$

Following that the recovered latitude (*Rlat*) is calculated. This calculation is performed for both the odd and even messages [Equation 2]. For this equation, the zone size is designated by the variable $Dlat_i$. $Dlat_1$ and $Dlat_0$ differ slightly with an odd message equal to 360/4*(Number Of Zones-1) and an even message equal to 360/4*(Number of Zones). The number of zones equals 15 as this is the amount of zones within each quadrant of the Earth. The modulus function within this equation returns the remainder of dividing the first variable by the second variable.

$$rlat(i) = Dlat_i * \left(\text{modulus}(j, 60) + \frac{Lat(i)}{131072}\right) \quad \textbf{(2)}$$

The next variable to determine is the number of longitude zones (*NL*). This is determined by using the *rlat* value from the previous equation. A lookup table [Appendix C] is used to determine the *NL(rlat)* value. Next is the width of the longitude

zone (*ni*) which is calculated by dividing 360 by *NL – i*. Further decoding leads us to the *longitude index* [Equation 3].

$$Longitude\ Index = \frac{\big(Lon(0) * (NL(i) - 1)\big) - \big(Lon(1) * NL(i)\big)}{131072} \quad \textbf{(3)}$$

Finally, the longitude of the second message received (the odd message) is found using [Equation 4].

$$Longitude = Dlon(i) * \left(\frac{\big(modulus(longitude\ index, ni)\big) + Lon(i)}{131072}\right) \quad \textbf{(4)}$$

The final field to decode is the parity check. The parity bits are calculated through a cyclic redundancy check (CRC) with a generating polynomial and using the first 88 bits of the ADS-B message [Equation 5]. It is assumed the user has general knowledge of how CRC bits are generated and will not be covered in great detail. Information for generating CRC bits can be found at [62].

$$\boldsymbol{G(X) = 1 + x^3 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} + x^{16} + x^{17} + x^{18} + x^{19} + x^{20} + x^{21} + x^{22} + x^{23} + x^{24}} \quad \textbf{(5)}$$

The preceding discussion focused on decoding the ADS-B message bits. Encoding an ADS-B message is essentially the reverse.

3.6 USRP and ADS-B Transmission

Just as in receiving, the transmission of ADS-B signals will require the correct hardware to transmit the messages at 1Mb/second at the carrier frequency of 1090MHz. For this setup, all equipment previously described was used. The use of the GNU radio companion (*GRC*) was used in the creation of the transmitter. *GRC* is a GUI application installed with the *UHD+GNURadio* script. To run *GRC* the command *gnuradio-companion* should be executed. Within the *GRC* are pre-built blocks that allow construction of SDR applications in a drag-and-drop fashion. The flow of information is then created by connecting each block's output to another block's input. The beginning block is called the source and the flow of information will end with a sink. When a *GRC* application is executed, Python scripts that control the USRP radios are automatically generated.

Figure 22 shows a *GRC* representation for the *ADS-B Transmitter*. A file source is used to provide the binary representation of the ADS-B message. This message has been properly encoded with all ADS-B message components as discussed in section 3.5, Figure 19. It has also been generated using proper PPM coding. Next is the *packed-to-unpacked* block, which extracts message bits from the file one bit at a time and prepares it for transmission. As the name states, this block receives packed data (hexadecimal format) and unpacks it into its binary format. For example, a hexadecimal "A" received from the file will be unpacked to "1010". A "1" read from the binary file will result in a pulse with amplitude of one, a "0" will result in no pulse.

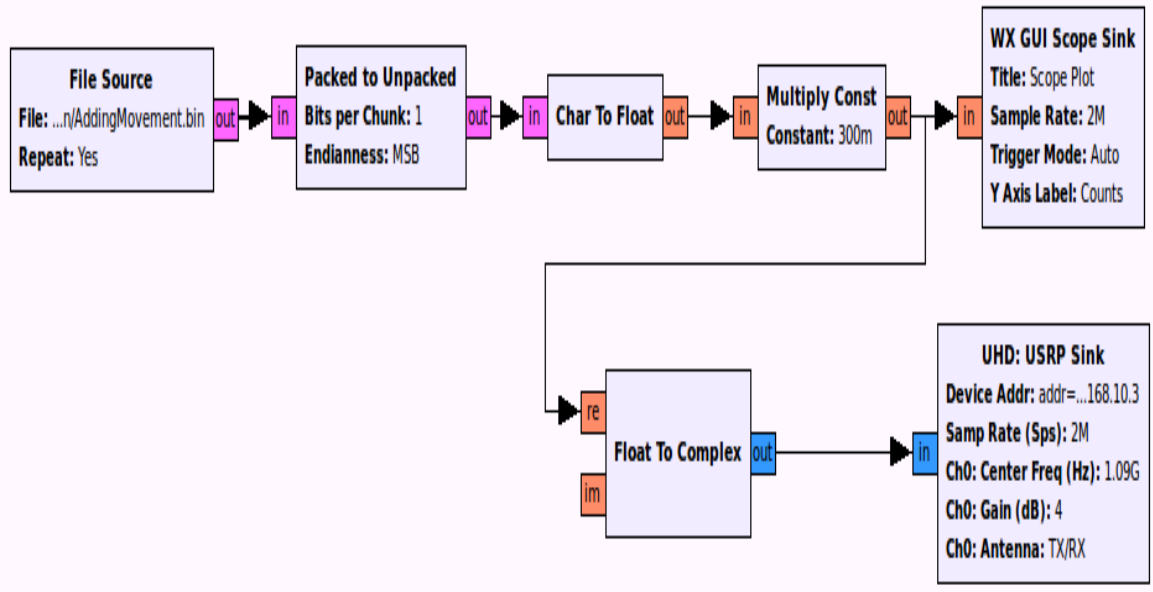


Figure 22 - GNU Radio Companion screenshot containing the *ADS-B Transmitter* application

The *char-to-float* block converts the binary bits to a floating point representation, which allows manipulation of the data using floating point numbers. For this case, the amplitude of one is multiplied by 0.3, decreasing the amplitude of the pulse from a height of 1 to 0.3. This is performed because the USRP cannot handle amplitude of 1 without causing pulses to overlap (smear), which would result in errors in our message.

There are two sinks for this *GRC* program. The *scope sink* is included to aid in troubleshooting issues with the signal and does not affect the application. The second sink is the USRP. Before the signal flows to the USRP, it goes through a *float-to-complex* block, this converts floating point data to complex data for upconversion to RF (1090 MHz).

The parameters of the USRP include the IP address of the USRP, sample rate, transmission gain, and antenna choice. The sample rate should be set to 2Ms/s in order to satisfy Nyquist's Theorem, described earlier. The transmission gain was set to default at four, best results when running the program were found between a gain of one and four. Finally, the antenna used throughout the research was the transceiver (TX/RX).

Another SDR application was designed within *GRC* to assist in troubleshooting any issues with the transmission of ADS-B messages [Figure 23]. The blocks involved with this application are a USRP source, a *complex-to-magnitude* block, and a *scope sink*. The USRP is tuned to the 1090MHz frequency and uses the *complex-to-magnitude* block to perform amplitude demodulation on the received signal, which is then fed into a *scope sink* for a visual look of the received message.

Figure 23 - *GRC* view of the ADS-B troubleshooting application

Using two USRPs connected by a 30 dB attenuator, a message was transmitted and received [Figure 13]. The results of that transmission were observed with the *scope sinks*. These two scope plots show the message before transmission [Figure 24] and the message upon receiving it with the other USRP [Figure 25].

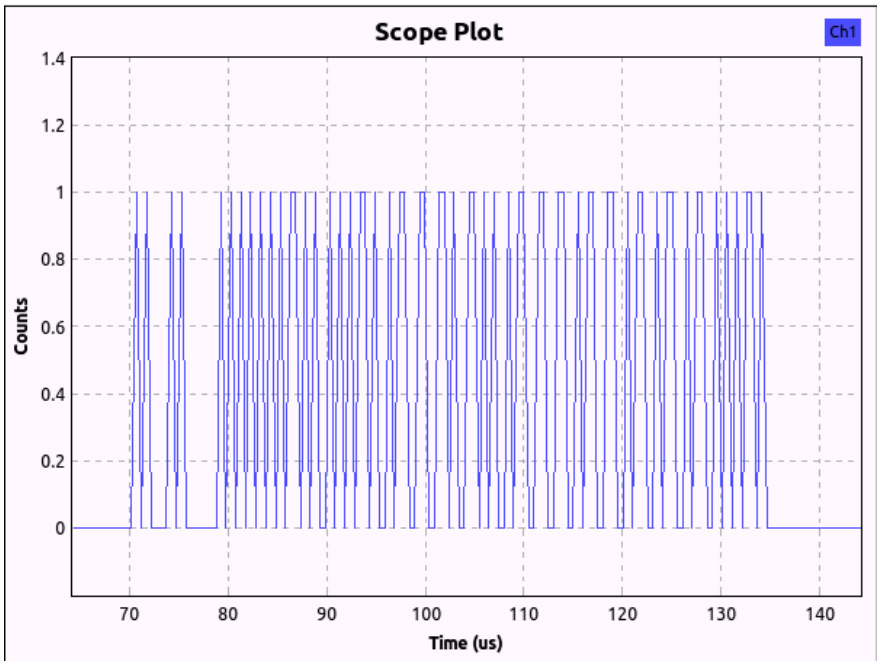


Figure 24 - Scope plot of transmitted message

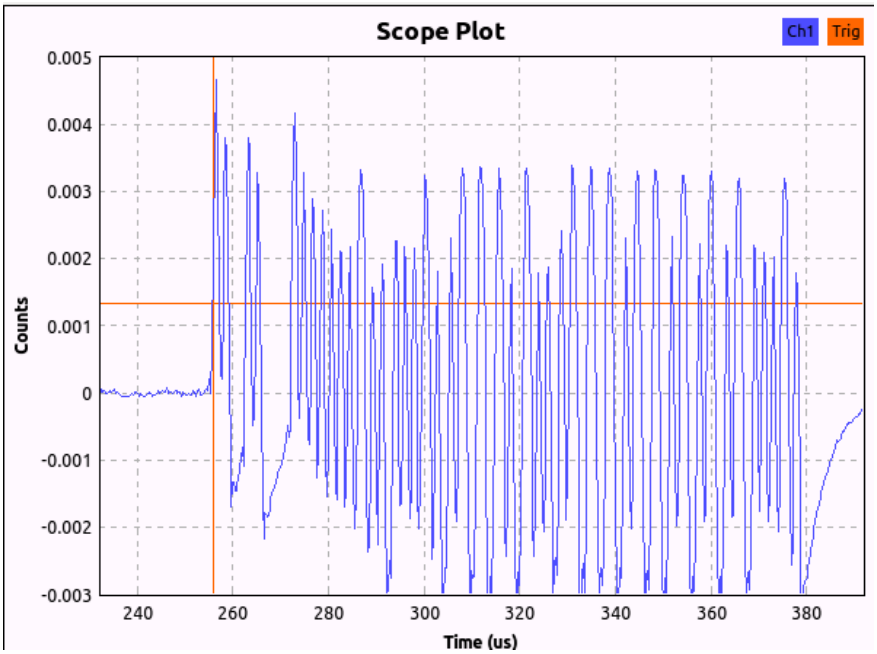


Figure 25 - Scope plot of received message

3.7 ADS-B Message Generation

The preceding sections have created the foundation for generating and transmitting ADS-B messages. This section discusses the calculations necessary to create the data file containing the encoded aircraft ID, altitude, positional data, and CRC bits. Appendix B contains a C++ program that accepts data input and generates a properly encoded ADS-B message.

Encoding an ADS-B message requires four inputs: aircraft ID, altitude, latitude and longitude. The aircraft ID will be entered as hexadecimal characters (0-9 & A-F). The altitude will be an integer divisible by 25, as all altitude will be encoded in 25ft increments for greater precision (compared to 100ft increments). Finally, the latitude and longitude inputs represented by degrees, minutes, and seconds (DDDMMSS) will be entered in decimal format. The DDDMMSS information is converted to decimal degrees for latitude and longitude using the following formula:

$$Decimal\ Degrees = Degrees + \frac{Minutes}{60} + \frac{Seconds}{3600} \quad \textbf{(6)}$$

After converting latitude and longitude into decimal numbers, it must be determined if they will be negative or positive. Using Figure 26, locations north of the equator will yield positive latitudes while locations south of the equator will be negative. Finally, locations east of the prime meridian will result in negative longitudes and vice versa for locations west of the prime meridian.

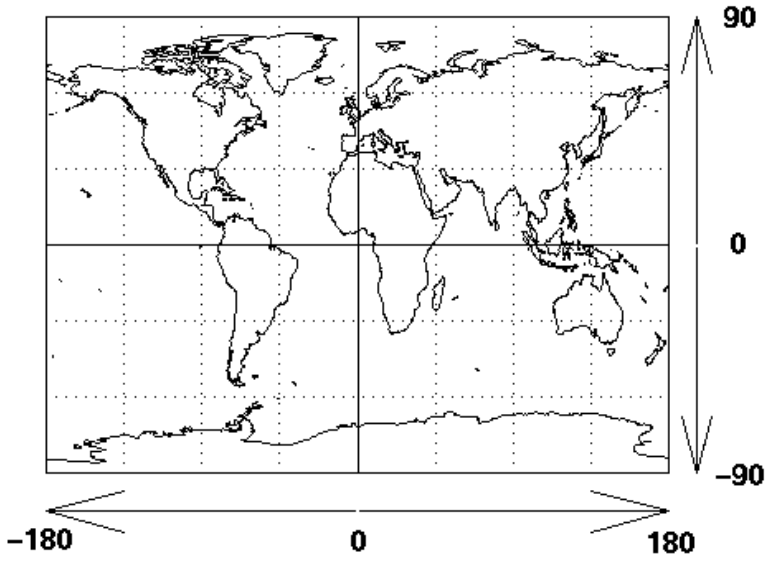


Figure 26 - Latitude and Longitude Measurements

3.7.1 DF Type

To construct the message as shown in Figure 19, the first step is to append the DF format type. This value, 0x8D or 10001101, is added to the message automatically by the C++ program. The first five bits (10001) equates to 17 decimal or the desired DF format type. The last three bits are a capability field referring to the specific data being transmitted, throughout this research this field will refer to positional data reports (subtype five).

3.7.2 Aircraft ID Generation

The next data to be entered is the aircraft ID. This is represented by six hex characters or three bytes of data. Since this information is entered as hexadecimal characters it does not need to be manipulated. It can be appended directed to the DF type and stored within the message array.

3.7.3 Altitude Encoding

Following the aircraft ID generation is the altitude calculations. This data is entered as a decimal number. To encode it requires subtracting 1,000 and then dividing the number by 25. As discussed earlier, room must be made for the Q bit. The values in the last four bits are held in memory. Following that, the first eight bits are shifted to the left by one. An OR is then performed to concatenate the shifted bits and the last four bits. To complete the encoding, the Q bit set to 1 (25 ft increments), is added in the eighth bit position. Similar to the aircraft ID, hexadecimal 0x58 is then appended to the front of the altitude to denote a type code of 11 (**01011**000), which equates to an airborne position report. The three zeros in the remaining in this field provide surveillance status within the first two bits and the antennas used in the last bit. These three bits are irrelevant to this research and will be set to "000" for all tests. The value for altitude is then stored within the message array.

3.7.4 Latitude and Longitude Encoding

The next two data inputs are the latitude and longitude. These require many more calculations but follow similar procedures as the decoding section. The first step is to determine $Dlat_i$, the latitude zone size [Equation 7]. These numbers will remain constant throughout the life of the program. *NZ* is the number of zones per quadrant which will be 15 at all times. The variable *i* refers to the format type (1 odd or 0 even).

$$Dlat_i = \left(\frac{360}{4*NZ-i}\right) \quad \textbf{(7)}$$

After determining the *Dlat* values, the next calculation is the *YZ* value or Y coordinate within the Zone. This is calculated using the *Dlat* value and the latitude value input by the user [Equation 8]. The result of this equation will give an integer within the 17 bit limit. *Floor(x)* is defined as the greatest integer *k* such that k ≤ x. For example, *floor(5.6)* will be equal to 5, while *the floor(-5.6)* equals -6. This integer is then assigned to a variable for later use.

$$YZ_i = \text{floor}\left(2^{17} * \left(\frac{\text{modulus}(lat, Dlat_i)}{Dlat_i}\right) + \frac{1}{2}\right) \quad \textbf{(8)}$$

Realized latitude is then calculated. This value will be approximately the same value as that entered by the user. The latitude, *YZ* and *Dlat* values are all used for this calculation as shown in Equation 9. The realized latitude or *Rlat* value, calculated during this step is then used to determine the specific longitude zone or *NL*. *NL* is set via a look up table as described earlier.

$$Rlat_i = Dlat_i * \left(\left(\frac{YZ}{2^{17}}\right) + \text{floor}\left(\frac{latitude}{Dlat_i}\right)\right) \quad \textbf{(9)}$$

Using the *NL* value returned by the lookup table [Appendix C], the longitude zone size (*Dlon*) can then be determined with the simple equation [Equation 10].

$$Dlon_i = \begin{cases} \dfrac{360}{NL(Rlat_i) - i} & \text{when } NL(Rlat_i) - i > 0 \\ 360 & \text{when } NL(Rlat_i) - i = 0 \end{cases} \quad \textbf{(10)}$$

Finally, the encoded longitude value is calculated. This is known as the *XZ* value or X coordinate within the Zone:

$$XZ_i = \text{floor}(2^{17} * \left(\left(\frac{\text{modulus}(longitude, Dlon)}{Dlon}\right) + \frac{1}{2}\right)) \quad \textbf{(11)}$$

After both numbers have been generated (*YZ* and *XZ*), they are concatenated onto each other. It should be noted that these numbers are not added together. The *YZ* value is shifted to the left 17 positions and then OR'd with the *XZ* value. This 34 bit value is now ready for transmission as an even message. In order to transmit as an odd message the bit in position #35 ("F" bit from Figure 19) must be set to "1". This value is then added to the message array in preparation for the final calculations before being written to a binary file.

3.7.5 CRC Bit Generation

The final step of encoding is to apply the CRC polynomial to compute the parity bits. The code for generating CRC bits has some preliminary bit manipulation before beginning the XORing. The first seven lines of code within the CalculateCRC112BitsOdd function ensure 4 bytes of data are in each of the variables to be XOR'd. Once this takes place, the individual portions of the message begin a loop that iterates until all 88 bits have gone through. The result of XORing the 88 bits with a 25

bit polynomial will result in a remainder of 24 bits. These bits are stored within the message array, which can now be written to a binary file. The file will be called *FinalMessage.bin*, located in the same directory as the *DF17MessageGenerator*. The user can then set this file to be the source of the *ADS-B Transmitter* [Figure 22].

3.7.6 Transmitting the Messages

Once the file is set as the source, there are two options for running the *ADS-B transmitter*. Those options are to run it from the *GRC* application or run it via command line. After execution begins, the scope plots for the *ADS-B Transmitter* as well as the ADS-B message analyzer (discussed earlier), will prove useful.

3.8 Summary

This chapter presented the development of a system comprised of a computer, USRP, and C++ code for generating acceptable ADS-B messages. Additionally, it discussed the process of decoding and encoding ADS-B messages for transmission. The next chapter discusses the validation of the system and covers some applications of how the system can be used to generate and inject false messages.

IV. Analysis and Results

4.1 Chapter Overview

This chapter will discuss the techniques used to validate the system developed in Chapter III. The first section will discuss the *GR-AIR-MODES* validation process. Following that will be the validation of the contributions of this research, the *DF17MessageGenerator* and the *ADS-B Transmitter*. Finally, demonstrations will be conducted and analyzed to display the potential capabilities of the system.

4.2 Validation of ADS-B Systems

It is reasonable to question the validity of *GR-AIR-MODES* and the *ADS-B Transmitter* created with the USRPs. *GR-AIR-MODES* depends on the code developed in [60] while the *ADS-B transmitter* application is dependent on the code developed in this research. The SBS-1eR commercial receiver was used to validate the operation of both *GR-AIR-MODES* and the *ADS-B Transmitter*. After validating the *GR-AIR-MODES* program it was used to validate the *DF17MessageGenerator*.

4.2.1 *GR-AIR-MODES* Validation

In order to validate the *GR-AIR-MODES* program, it was run side-by-side with the SBS-1eR multiple times. The intention of these experiments was to compare the data captured by both receivers. Should the commercial device and *GR-AIR-MODES* program capture similar data, it is assumed *GR-AIR-MODES* performs accurately.

During the side-by-side experiments, *GR-AIR-MODES* was run with the –K option invoked to store all ADS-B data within KML files for viewing within Google

Earth. This option was used to replicate the virtual radar screen generated by the SBS-1eR software. Throughout all experiments, the devices captured the exact same information. The results of one of the experiments conducted are shown in Figure 27. During this experiment, there was one aircraft broadcasting ADS-B data. This data was captured by both the SBS-1eR and *GR-AIR-MODES*.

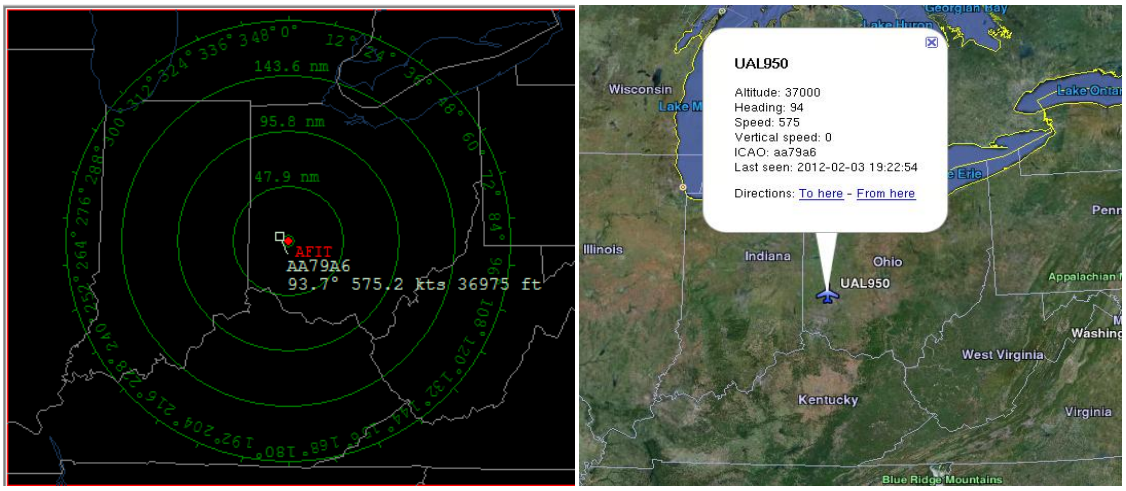


Figure 27 - Side-by-side view of SBS-1eR (left) and *GR-AIR-MODES* (right)

Additional data captured during this particular experiment included altitude, speed, and heading [Figure 28] [Figure 29]. Both devices were able to capture and decode precisely the same information with a heading of approximately 94°, a speed of 575 knots (~661 mph), an altitude of 37,000ft and a location of latitude 35.854721 and longitude -84.266113. A slight difference in longitude and altitude, shown within the figures below, is a result of the delay in capturing a screen shot for both programs (i.e., the aircraft kept broadcasting data so *GR-AIR-MODES* received one additional message

with updated information).

Figure 28 - SBS-1eR captured data

```
(-31 0.0000000000) Type 17 subtype 05 (position report) from aa79a6 at (39.854721, -84.266113) (27.78 @ 350) at 37000ft
(-31 0.0000000000) Type 17 subtype 09-1 (track report) from aa79a6 with velocity 575kt heading 94 VS 0
```

Figure 29 - *GR-AIR-MODES* captured data

4.2.2 *DF17MessageGenerator* Validation

GR-AIR-MODES can then be used to validate the *DF17MessageGenerator*. For the purpose of validation, the –w command was used during *GR-AIR-MODES* execution. This command will display the ADS-B message in its hexadecimal format in the command prompt window [Figure 30]. The aircraft information, to include aircraft ID, altitude, and position, were then used as input for the *DF17MessageGenerator*. The message displayed by *GR-AIR-MODES* was then compared to the message generated by the *DF17MessageGenerator*. The *DF17MessageGenerator* produced the exact same results [Figure 31].

```
(-32 0.0000000000) Type 17 subtype 05 (position report) from ab7437 at (39.925827, -84.193542) (33.09 @ 347) at 36025ft
17 8dab7437 58b9929e02f39d 984e70 000000 0.0005643221084 0000000000
```

Figure 30 - Raw data displayed by *GR-AIR-MODES*

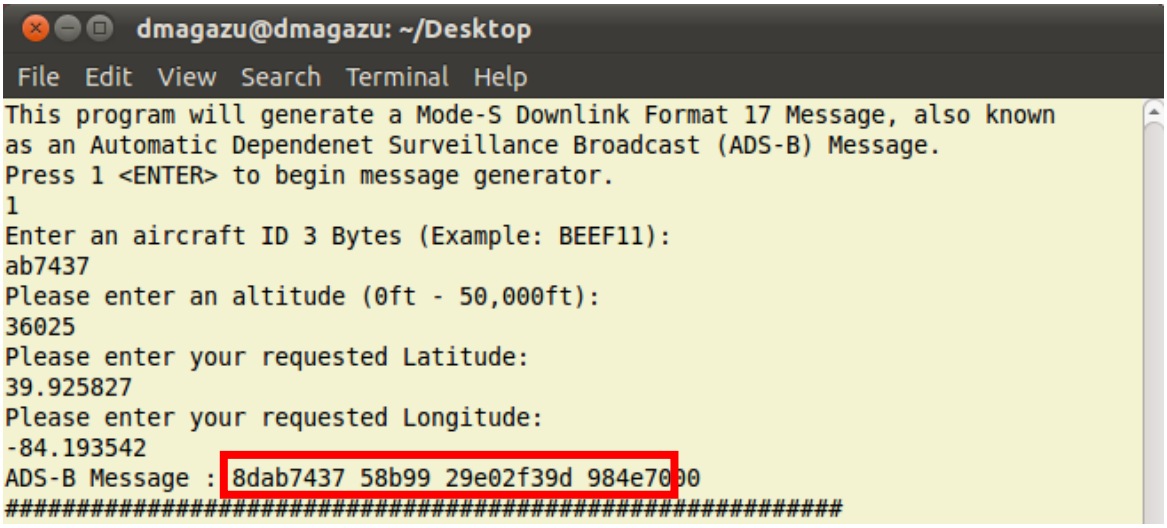


Figure 31 - *DF17MessageGenerator* validation message

4.2.3 *ADS-B Transmitter* Validation

The final step is to validate the *ADS-B Transmitter*. To perform this validation, the USRP N200 ran the *ADS-B Transmitter* with a file source containing the message from the previous validation (Aircraft ID AB7437, altitude 36025, latitude 39.925827 and longitude -84.193542). The USRP N200 then transmitted the message, which was received successfully by the SBS-1eR [Figure 32]. The results showed the successful reception and display of the ADS-B message [Figure 33]. The bottom left corner of the figure displays the altitude, the top left corner shows the virtual radar display with the aircraft plotted in the correct position, and the right side of the figure displays the aircraft's information to include the latitude and longitude points.

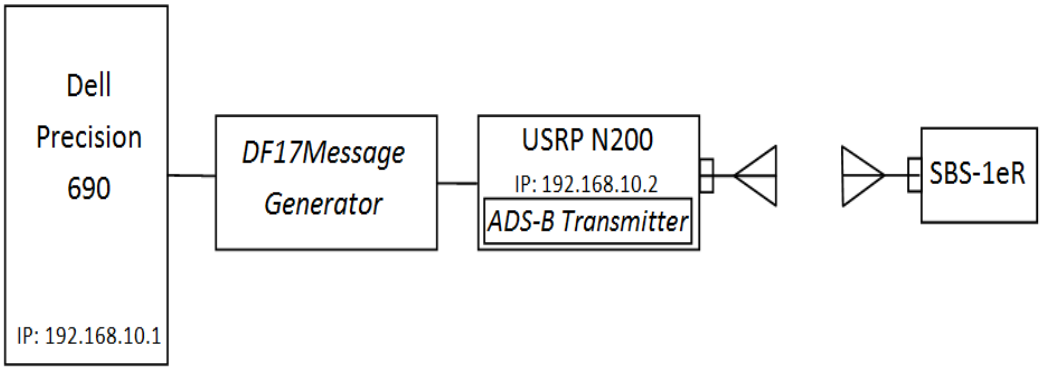


Figure 32 - System block diagram for implementation/use

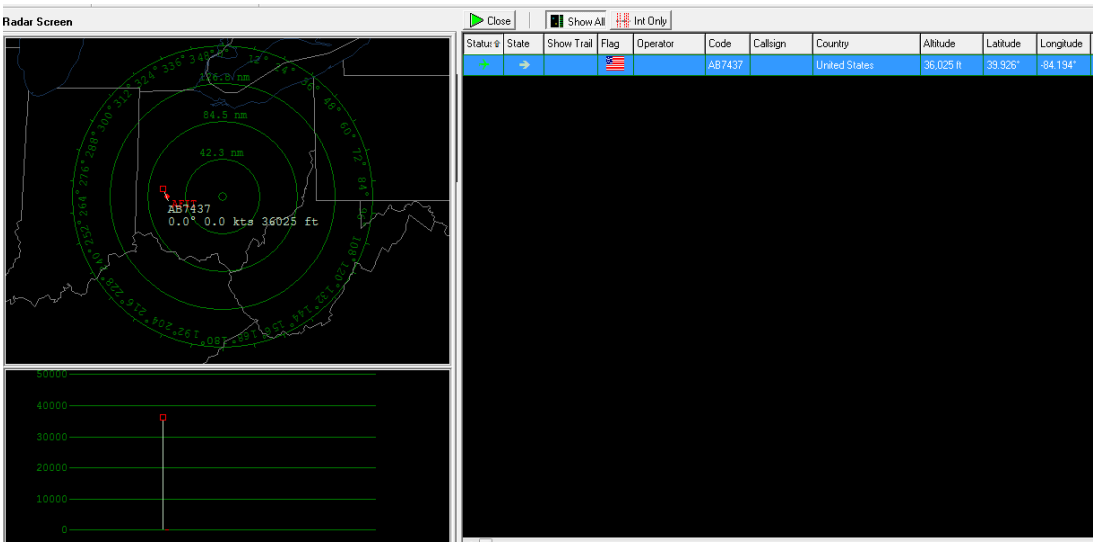


Figure 33 - *ADS-B Transmitter* Validation

4.3 Demonstrating the *DF17MessageGenerator* and *ADS-B Transmitter*

After completing the validation of the systems, four demonstrations were performed to exhibit the system's potential. All experiments were conducted using the

DF17MessageGenerator and the *ADS-B Transmitter*. The first demonstration was to plot multiple aircraft on the radar display of both *GR-AIR-MODES* and the SBS-1eR. Next, live data received by the SBS-1eR was observed, altered, and rebroadcast. The third demonstration plotted false aircraft following a live aircraft. Finally, an aircraft track was generated, which involves having a false target "move" through the airspace.

4.3.1 Flooding Radar Display

The first scenario tested was flooding the radar screen with false targets. To demonstrate this scenario, 10 aircraft messages were generated using the *DF17MessageGenerator* and then transmitted using the USRP/*ADS-B Transmitter*. Both *GR-AIR-MODES* and SBS-1eR were used to receive and plot the aircraft with varying locations and altitudes, as shown in Figure 34 and Figure 35. This technique could be implemented using as few or as many aircraft as desired to cause confusion within the airspace and in ground facilities.

Figure 34 - Google Earth view of multi-aircraft scenario

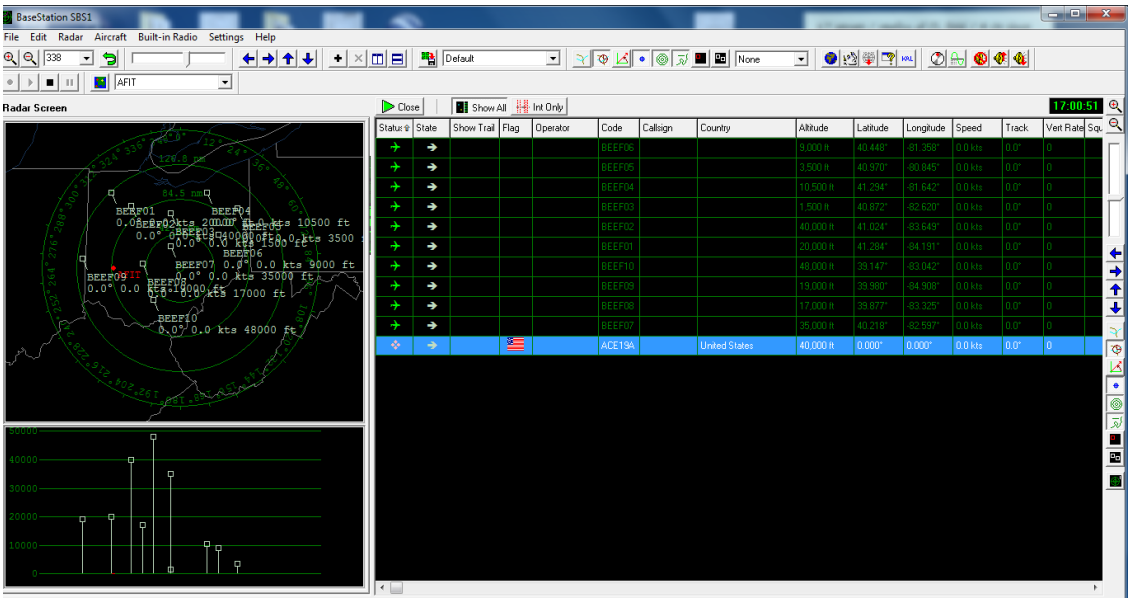


Figure 35 – SBS-1eR software view of multi-aircraft scenario

4.3.2 Altering Live Data

The next scenario requires the reception of live data, altering the information such as the altitude or location, and broadcasting the new spoofed message. This scenario could be used to "change" the position of an aircraft and cause confusion within the cockpit of an aircraft or at an ATC ground facility.

To demonstrate this scenario, live data was captured using the SBS-1eR. Once an aircraft broadcasting ADS-B messages appeared on the radar screen, its data was entered into the *DF17MessageGenerator* [Figure 36] with a different altitude and position.

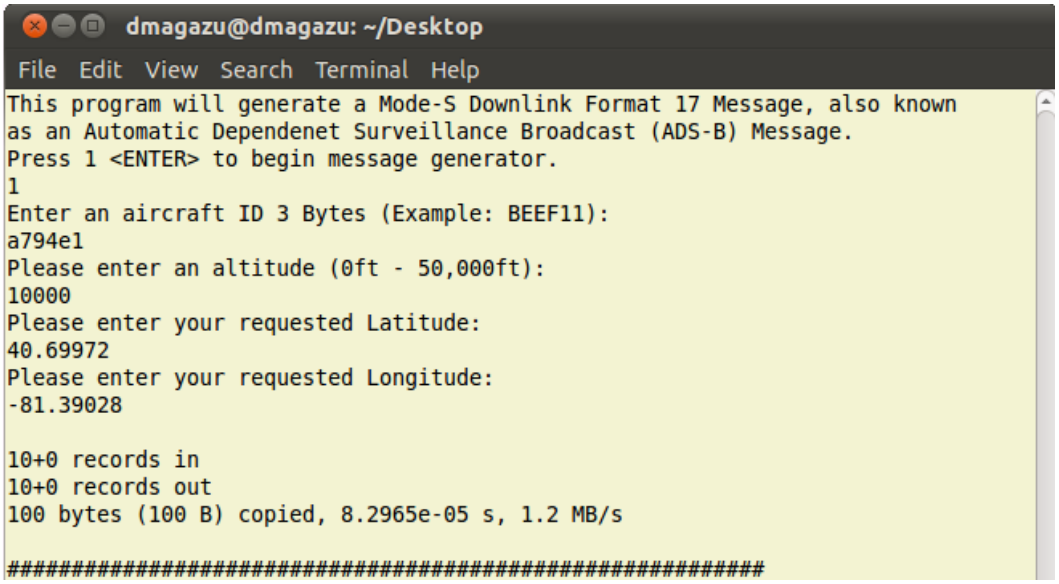
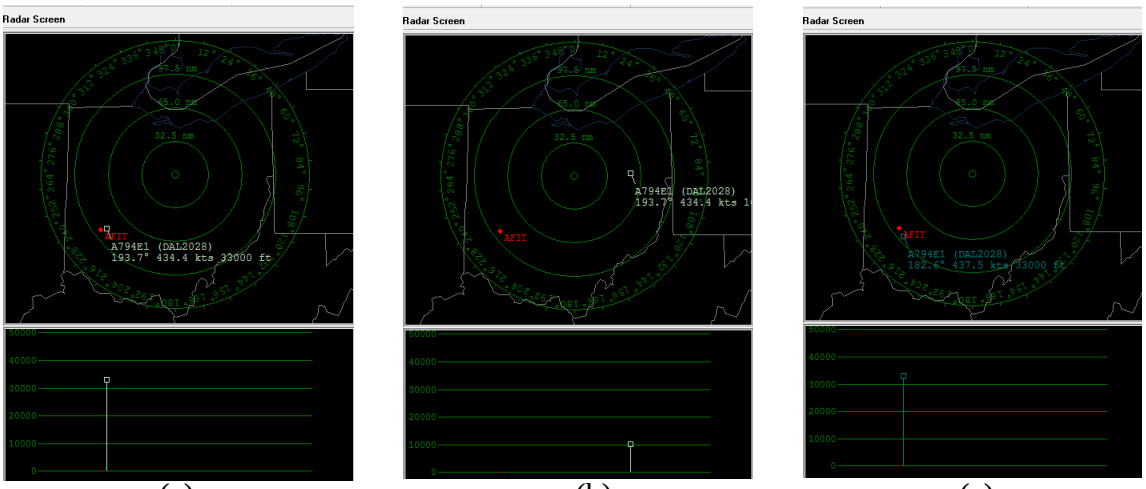


Figure 36 – Entering altered data for live data scenario

Using the *ADS-B Transmitter*, the spoofed message was then broadcast. After a few seconds, the SBS-1eR shows a change in data for the aircraft with aircraft ID A794E1. First, a change in altitude (from 33,000 ft to 10,000 ft) is observed and then the position changes to the spoofed position coordinates. The lag in positional change is due to the calculations involved in converting the odd and even messages to an exact X and Y coordinate within the latitude and longitude zones. Once the position has been determined, the aircraft seems to jump from its current location (a) to the false position (b). The position can be viewed in the top portion of the figure and the altitude can be observed within the bottom section. After turning the *ADS-B transmitter* off, the aircraft's true position and altitude return [Figure 37]. In (c), the aircraft is highlighted blue as the receiver interprets the change in altitude (from 10,000 ft to 33,000 ft), as the aircraft ascending. Of note, the figure gives the impression that the true aircraft position

vanishes. This is due to the close proximity of the *ADS-B Transmitter* to the SBS-1eR, causing the commercial receiver to be flooded with the spoofed message, which is transmitted repeatedly. In reality, both the real aircraft and spoofed aircraft would appear on the display of an aircraft or ATC ground facility.



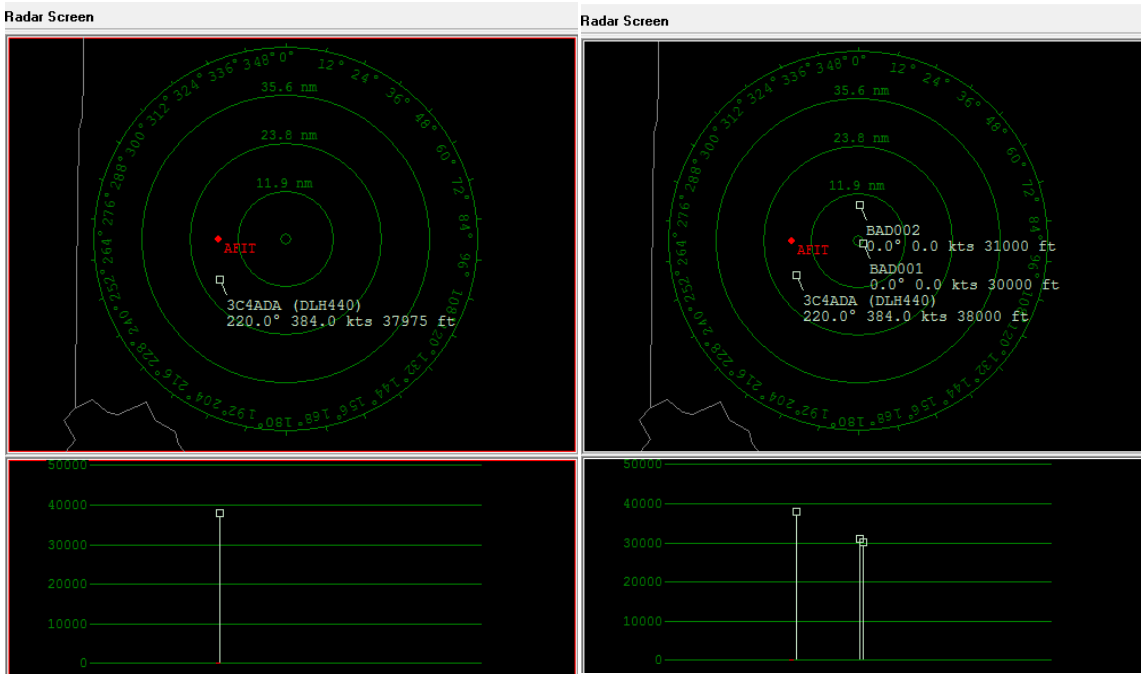
(a) (b) (c)

Figure 37 - Series of SBS-1eR screen captures (a) shows original aircraft data, (b) shows aircraft data of spoofed messages, and (c) shows aircraft returning to original location

4.3.3 False Aircraft Following

The third demonstration also awaited the reception of live aircraft data using the SBS-1eR. Once an aircraft was displayed, two ADS-B messages were generated with aircraft IDs of BAD001 and BAD002. Both false messages had coordinates trailing the aircraft by approximately 15-20 nautical miles. This demonstrates the ability to inject false aircraft in the vicinity of real aircraft in an expedient manner. If the locations of the false aircraft were closer to the target it may cause confusion for the pilots, which could

lead to unnecessary corrections to avoid a mid-air collision.



(a) (b)

Figure 38 - Radar display for aircraft following scenario (a) shows the live aircraft and (b) shows the two spoofed aircraft trailing

4.3.4 Aircraft Track Generation

The final demonstration to be discussed is the production of an aircraft track. This involves having a false target "move" through the airspace by changing the position values in the ADS-B messages that are transmitted on a regular basis. This demonstration generated movement for one aircraft, however multiple false tracks could also be generated. This can also be applied to any of the previous demonstrations.

To execute this scenario, a message must be generated to establish the point of origin of the false aircraft. Subsequent messages should then follow within 1-10 seconds of each other using appropriate latitude and longitude points for movement in any direction [Figure 39]. The scenario produced an aircraft with Aircraft ID D3D3D3 moving in an eastward direction. The dots on the figure represent the reception of individual ADS-B messages. An update was sent once every second giving the impression the aircraft traveled over 50 nautical miles.

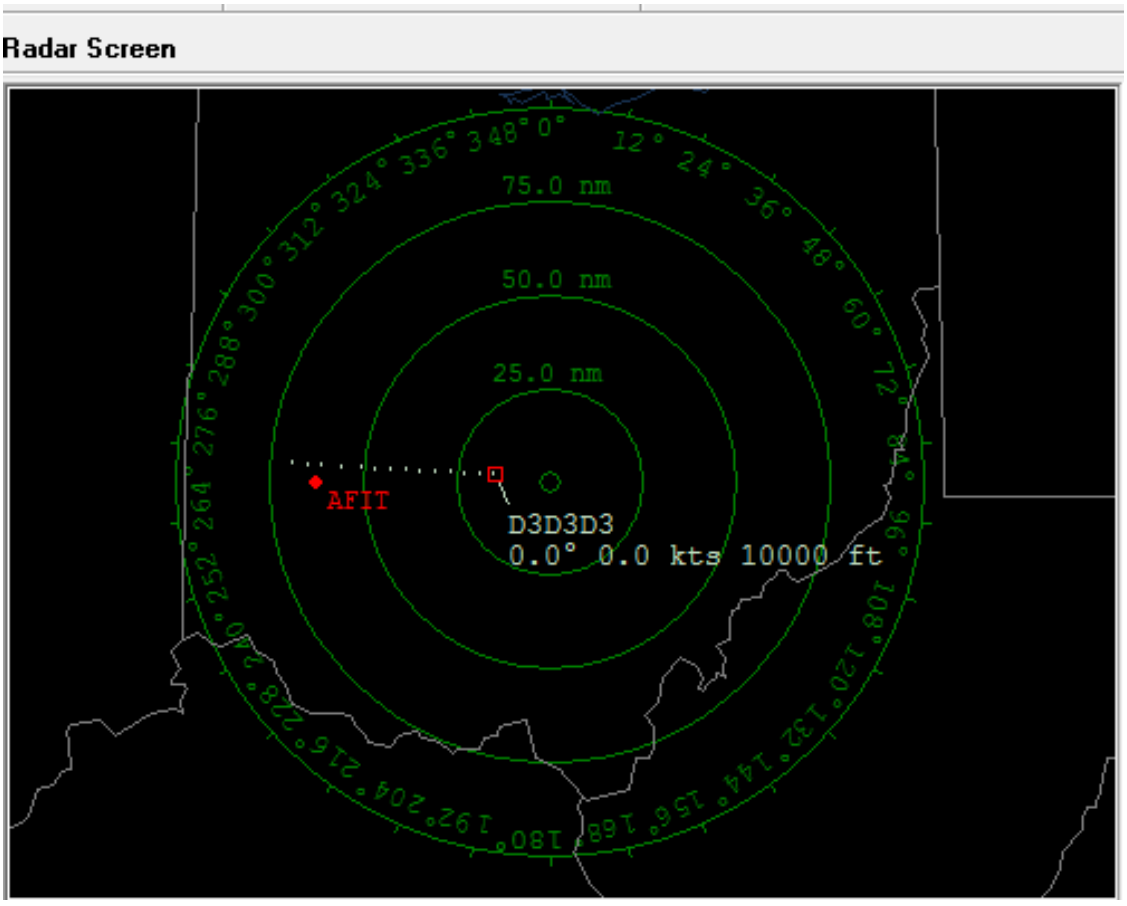


Figure 39 – SBS-1eR radar picture of false aircraft track

4.4 Summary

The demonstrations within this chapter emphasize the potential of the *DF17MessageGenerator* and *ADS-B Transmitter* to generate false/deceptive ADS-B messages. With a limited amount of time, it was not feasible to construct more demonstrations, however the work provided here creates a strong foundation for the capabilities of the system developed in this thesis. It is assumed by the amount of success already achieved that the uses of the system are only limited by the imagination.

V. Conclusions and Recommendations

5.1 Chapter Overview

This chapter reviews the work completed and offers conclusions on the information discovered through the experiments. The investigative questions from Chapter I will be reviewed first, then discussions on further research related to this topic, and finally a summary.

5.2 Conclusions of Research

The purpose of this research was to investigate the weaknesses of the NextGen system to include the ability to inject false targets on the display of a commercial receiver. Additionally, this work investigated the ability to develop a proof of concept system using relatively inexpensive equipment. The author concludes that this research successfully developed a system capable of generating ADS-B messages and transmitting them in accordance with ADS-B message protocol. Furthermore, the messages transmitted were successfully received and displayed by a commercial ADS-B receiver, the SBS-1eR.

5.3 Investigative Questions Answered

The research and methodology described in the previous chapters revealed the steps required to encode raw data and transform it into a properly formatted ADS-B transmission. The majority of the information for encoding could be found within RTCA DO 260 Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast which is purchasable for approximately $150.00. The reason the author did not purchase and use this document was to illustrate

the ease in which an individual could find the necessary information as well as maintain anonymity.

The accomplishments of this thesis showed that with some knowledge of digital signal processing (DSP) and the ability to write code in Python and C++, it is possible to develop a system capable of generating acceptable ADS-B messages. "Acceptable" means good enough to be recognized by commercial off-the-shelf equipment that are used to decode ADS-B broadcasts. The messages ranged from single aircraft to multi-aircraft broadcasting. Additionally, the messages were used to generate aircraft tracks to imitate the travel of an aircraft across the radar screen of a receiving station.

The creation of the system took approximately six months for an individual with little to no background in DSP and RF communications. The system developed was composed of two USRPs with appropriate RF daughterboards and antennas, a Dell personal computer, and freely available software. In reality, once the system is built, two USRPs are not necessary. Excluding the second device and its associated daughterboard, the cost for creating this system was less than $5,000.00, with the USRP costing $1700.00, an antenna $35.00, WBX daughterboard $450.00, and the Dell Precision 690 costing approximately $2,000.00. The software for DSP applications along with the operating system came at no cost as they are freely available for download and use.

Finally, the potential capability of the system was also shown through the use of the system in four demonstrations. Those scenarios exhibited the use of the system in generating multiple false targets, altering live data being received, inserting false targets near real aircraft and lastly, creating an aircraft track.

5.4 Recommendations for Future Research

Although the system developed in this thesis has the ability to generate ADS-B messages, it currently can only create one message at a time. Future research should include generating multiple messages in a dynamic manner. The code, as currently written, will stack one message on top of another within the same file. This can be successful for most scenarios, to include all those discussed within chapter four. Future work could also include the development of separate functions to perform certain tasks such as generating a complete track with only a starting (both latitude and longitude) and end point.

In relation to work with generating a complete track, work with the 'great circle' calculations could prove to be useful in generating tracks for aircraft and the associated latitude and longitude points along a given track. The great circle premise states that for any two points on the surface of a sphere, there exists a circle that goes through both of those points. With further research into these calculations, it could then be incorporated into the current code for expedited long track message generation.

The system proved to work well within the experimental environment. However, it was not tested on real systems within an aircraft or at an ADS-B ground station. The low power of the USRP makes it ideal for the environment in which this thesis took place and prevents the transmissions from causing any interference or reception by aircraft equipped with ADS-B In. This is a limiting factor in that the system could not be tested aboard aircraft to determine if the messages displayed upon the SBS-1eR would also show on the displays of a cockpit. Further real-world testing is therefore needed.

Additional research could be performed in the use of ADS-B subtype 9 messages. This provides aircraft speed and track information. When broadcasting spoofed messages throughout this research, it can be observed that the speed and track information located below the aircraft displayed on the SBS-1eR shows zeros. Having the ability to add this information to the current system would increase the appearance of an authentic aircraft broadcasting ADS-B data.

Finally, future research could be conducted with the non-commercial ADS-B data link, UAT or 978 MHz. Experiments could be conducted to test if the current system would have the ability to generate false targets using this link.

5.5 Summary

The system produced through this research was comprised of the following components, GNU Radio for the software defined radio application, a USRP front end, and the development of a C++ program to properly encode messages. Through the validation techniques described in Chapter IV the system proved it has the ability to generate and transmit acceptable ADS-B messages. Moreover, the use of a commercial receiver was used to provide validation of receiving and displaying these false ADS-B messages.

The tools created through this research, the *DF17MessageGenerator* and the *ADS-B Transmitter*, have proved the hypothesis proposed by McCallie, et al [2] of spoofing ADS-B messages. These tools further show the weaknesses of the NextGen system. In conclusion, future research could identify further uses for this system.

Appendix A

Multiple devices per host

For maximum throughput, one ethernet interface per USRP2 is recommended, although multiple devices may be connected via a gigabit ethernet switch. In any case, each ethernet interface should have its own subnet, and the corresponding USRP2 device should be assigned an address in that subnet. Example:

Configuration for USRP2 device 0:

- Ethernet interface IPv4 address: 192.168.10.1
- Ethernet interface subnet mask: 255.255.255.0
- USRP2 device IPv4 address: 192.168.10.2

Configuration for USRP2 device 1:

- Ethernet interface IPv4 address: 192.168.20.1
- Ethernet interface subnet mask: 255.255.255.0
- USRP2 device IPv4 address: 192.168.20.2

Change the USRP2's IP address

You may need to change the USRP2's IP address for several reasons:

- to satisfy your particular network configuration
- to use multiple USRP2s on the same host computer
- to set a known IP address into USRP2 (in case you forgot)

Method 1: To change the USRP2's IP address you must know the current address of the USRP2, and the network must be setup properly as described above. Run the following commands:

```
cd <install-path>/share/uhd/utils
./usrp_burn_mb_eeprom --args=<optional device args> --key=ip-addr --val=192.168.10.3
```

Method 2 (Linux Only): This method assumes that you do not know the IP address of your USRP2. It uses raw ethernet packets to bypass the IP/UDP layer to communicate with the USRP2. Run the following commands:

```
cd <install-path>/share/uhd/utils
sudo ./usrp2_recovery.py --ifc=eth0 --new-ip=192.168.10.3
```

Appendix B

```
//////////////////////////////////////////////////////////
//Code developed to generate ADS-B messages///////////////
//////////////////////////////////////////////////////////

#include <iostream>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string>
#include <iomanip>

using namespace std;

long int completeMessageOdd[5];
long int completeMessageEven[5];
int Dlat0 = 6;               //even messages
double Dlat1 = 6.10169491525423728813559322033 9;  //odd messages
double pi = 3.14159265;
char buffer[30] = {0xA1, 0x40};

double Modulus(double val, double modval)
{

      if (val < 0) //Checking for negative angles
      {
            val = val + 360;
            int result = floor(static_cast<int>( val / modval));
            return val - static_cast<double>( result ) * modval;
      }
            else
            {
                  int result = static_cast<int>( val / modval);
                  return val - static_cast<double>( result ) * modval;
            }
}

int GenerateNlatValue(double latitude)
{
      double Nlat;
      double sqrtHold;
      double buff[63];
      int j = 2;
      for (int i = 2; i <=59; i++)
      {
      Nlat = 0.0;
      sqrtHold = 0.0;
      Nlat = (180/pi)*(acos(sqrt((1-(cos(pi/30)))/(1-
      (cos((2*pi)/i))))));
      buff[j] = Nlat;
      j++;
      }
```

```
	for (int k = 59; k >=2; k--)
	{
		if(latitude > buff[k])
		{

		}
		else
		{
			if (k < 2)
			{
				return 1;
			}
			return k;
		}
	}
}


int CalculateLatBitsOdd(double lat, double longitude)
{

	//Calculate YZ which will be what is put into our message
	double YZ;
	double modHold;
	int LatHex;
	modHold = Modulus(lat, Dlat1);
	modHold = (modHold/Dlat1);
	modHold = modHold * pow(2,17);
	modHold = modHold + .5;
	YZ = floor(modHold);

	//Calculate Rlatiude for airborne
	long double Rlat1;
	int floorHold;
	Rlat1 = (double)YZ / pow(2,17);
	floorHold = (lat/(double)Dlat1);
	floorHold = floor(floorHold);
	Rlat1 = Rlat1 + (double)floorHold;
	Rlat1 = Rlat1 * Dlat1;
	int NlLat = GenerateNlatValue(Rlat1);

	//Calcule DLongitude
	double Dlon1;
	if ((NlLat - 1) > 0)
	{
		Dlon1 = (360/((double)NlLat-1));
	}
	else
		Dlon1 = 360;

	//Calculate XZ the decimal representation of our longitude
	double XZ;
	modHold = 0;
	modHold = Modulus(longitude, Dlon1);
```

```
	modHold = (modHold/Dlon1);
	modHold = modHold * pow(2,17);
	modHold = modHold + .5;
	XZ = floor(modHold);

	//Ensure this fits into our 17 bit space
	long int YZ1 = Modulus(YZ,pow(2,17));
	long int XZ1 = Modulus(XZ,pow(2,17));
	long int LatLon;
	YZ1 = YZ1 << 17;
	LatLon = (YZ1 | XZ1);
	LatLon = (LatLon | 17179869184);
	completeMessageOdd[2] = LatLon;
return 0;
}

int CalculateLatBitsEven()
{
	double lat;
	double longitude;
	cout << "Please enter your requested Latitude: \n";
	cin >> setbase(10) >> lat;
	cout << "Please enter your requested Longitude: \n";
	cin >> setbase(10) >> longitude;
	CalculateLatBitsOdd(lat, longitude);

	//Calculate YZ which will be what is put into our message
	double YZ;
	double modHold;
	int LatHex;
	modHold = Modulus(lat, Dlat0);
	modHold = (modHold/Dlat0);
	modHold = modHold * pow(2,17);
	modHold = modHold + .5;
	YZ = floor(modHold);

	//Calculate Rlatiude for airborne
	long double Rlat0;
	int floorHold;
	Rlat0 = (double)YZ / pow(2,17);
	floorHold = floor(lat/(double)Dlat0);
	Rlat0 = (double)Rlat0 + floorHold;
	Rlat0 = Rlat0 * (double)Dlat0;
	int NlLat = GenerateNlatValue(Rlat0);

	//Calcule DLongitude
	double Dlon0;
	if ((NlLat - 1) > 0)
	{
		Dlon0 = (360/((double)NlLat));
	}
	else
	{
		Dlon0 = 360;
	}
```

```
	//Calculate XZ the decimal representation
	//of our longitude
	double XZ;
	modHold = Modulus(longitude, Dlon0);
	modHold = (modHold/Dlon0);
	modHold = modHold * pow(2,17);
	modHold = modHold + .5;
	XZ = floor(modHold);

	//ensure they will fit into a 17 bit message
	long int YZ1 = Modulus(YZ,pow(2,17));
	long int XZ1 = Modulus(XZ,pow(2,17));
	long int LatLon;
	YZ1 = YZ1 << 17;
	LatLon = (YZ1 | XZ1);
	completeMessageEven[2] = LatLon;

return 0;
}

long int CalculateAltitude()
{
	long int altitude;
	long int hold;
	cout << "Please enter an altitude (0ft - 50,000ft):  \n";
	cin >> setbase(10) >> altitude;
	altitude = (altitude + 1000)/25;
	hold = (altitude & 0x00F);
	altitude = (altitude & 0xFF0) << 1;
	altitude = (altitude | hold); //concatenate to the entire message
	altitude = (altitude | 0x010);
	altitude = (altitude | 0x58000); //Takes on the TC (0x58) field

return altitude;
}


long int GenerateAircraftID()
{
	long int address;
	long int df17;
	cout << "Enter an aircraft ID 3 Bytes (Example: BEEF11): \n";
	cin >> setbase(16);
	cout << setbase(16);
	cin >>  address;
	df17 = 0x8D;
	df17 = df17 << 24;
	df17 = (df17 | address);

return df17;
}
```

```
long int CalculateCRC112BitsOdd()
{
	long poly = 0xFFFA0480;
	long int a = completeMessageOdd[0];
	long int a2 = completeMessageOdd[1];
	long int a3 = completeMessageOdd[2];
	int j;
	long int b = a2;
	long int c = a3;
	long int d;
	long int hold;

	b = b << 12;
	d = (c & 0xFFF000000);
	d = d >> 24;
	a2 = (b | d);
	hold = (a3 & 0x000FFFFFF);
	hold = hold << 8;
	a3 = hold;



	for (j=1; j <= 88; j++)
		{
			if((a & 0x80000000) != 0)
			{
				a = a ^ poly;
			}
			a = a << 1;
			if((a2 & 0x80000000) != 0)
			{
				a = a|1;
			}
			a2 = a2 << 1;
			if((a3 & 0x80000000) != 0)
			{
				a2 = a2|1;
			}
			a3 = a3 << 1;
		}
	completeMessageOdd[3] = a;
return 0;
}

long int CalculateCRC112BitsEven()
{
	long poly = 0xFFFA0480;
	long int a = completeMessageEven[0];
	long int a2 = completeMessageEven[1];
	long int a3 = completeMessageEven[2];
	int j;
	long int b = a2;
	long int c = a3;
	long int d;
	long int hold;
```

```
	b = b << 12; //Bit manipulation to push the message together
	d = (c & 0xFFF000000);
	d = d >> 24;
	a2 = (b | d);
	hold = (a3 & 0x000FFFFFF);
	hold = hold << 8;
	a3 = hold;

	for (j=1; j <= 88; j++)
		{
			if((a & 0x80000000) != 0)
			{
				a = a ^ poly;
			}
			a = a << 1;
			if((a2 & 0x80000000) != 0)
			{
				a = a|1;
			}
			a2 = a2 << 1;
			if((a3 & 0x80000000) != 0)
			{
				a2 = a2|1;
			}
			a3 = a3 << 1;
		}
	completeMessageEven[3] = a;
return 0;
}

long int ConvertForHexEditor(long int a, long int b, long int c, long
int d)
{
	long int ACID = a;
	int i = 1;
	int m = 2;
	while (i<=8)
	{
		ACID = (ACID & 0xF0000000);
		if (ACID == 0x00000000)
			buffer[m] = 0x55;
		else if (ACID == 0x10000000)
			buffer[m] = 0x56;
		else if (ACID == 0x20000000)
			buffer[m] = 0x59;
		else if (ACID == 0x30000000)
			buffer[m] = 0x5A;
		else if (ACID == 0x40000000)
			buffer[m] = 0x65;
		else if (ACID == 0x50000000)
			buffer[m] = 0x66;
		else if (ACID == 0x60000000)
			buffer[m] = 0x69;
		else if (ACID == 0x70000000)
```

```
            buffer[m] = 0x6A;
      else if (ACID == 0x80000000)
            buffer[m] = 0x95;
      else if (ACID == 0x90000000)
            buffer[m] = 0x96;
      else if (ACID == 0xA0000000)
            buffer[m] = 0x99;
      else if (ACID == 0xB0000000)
            buffer[m] = 0x9A;
      else if (ACID == 0xC0000000)
            buffer[m] = 0xA5;
      else if (ACID == 0xD0000000)
            buffer[m] = 0xA6;
      else if (ACID == 0xE0000000)
            buffer[m] = 0xA9;
      else if (ACID == 0xF0000000)
            buffer[m] = 0xAA;
      else


      ACID = a;
      ACID = (ACID << (i*4));
      i++;
      m++;
}
cout << " ";
ACID = b;
ACID = (ACID << 12);
i = 4;
while (i<=8)
{
      ACID = (ACID & 0xF0000000);
      if (ACID == 0x00000000)
            buffer[m] = 0x55;
      else if (ACID == 0x10000000)
            buffer[m] = 0x56;
      else if (ACID == 0x20000000)
            buffer[m] = 0x59;
      else if (ACID == 0x30000000)
            buffer[m] = 0x5A;
      else if (ACID == 0x40000000)
            buffer[m] = 0x65;
      else if (ACID == 0x50000000)
            buffer[m] = 0x66;
      else if (ACID == 0x60000000)
            buffer[m] = 0x69;
      else if (ACID == 0x70000000)
            buffer[m] = 0x6A;
      else if (ACID == 0x80000000)
            buffer[m] = 0x95;
      else if (ACID == 0x90000000)
            buffer[m] = 0x96;
      else if (ACID == 0xA0000000)
            buffer[m] = 0x99;
      else if (ACID == 0xB0000000)
```

```
            buffer[m] = 0x9A;
      else if (ACID == 0xC0000000)
            buffer[m] = 0xA5;
      else if (ACID == 0xD0000000)
            buffer[m] = 0xA6;
      else if (ACID == 0xE0000000)
            buffer[m] = 0xA9;
      else if (ACID == 0xF0000000)
            buffer[m] = 0xAA;
      else

      ACID = b;
      ACID = (ACID << (i*4));
      i++;
      m++;
}
cout << " ";

ACID = c;
i = 1;
while (i<=9)
{
      ACID = (ACID & 0xF00000000);
      if (ACID == 0x000000000)
            buffer[m] = 0x55;
      else if (ACID == 0x100000000)
            buffer[m] = 0x56;
      else if (ACID == 0x200000000)
            buffer[m] = 0x59;
      else if (ACID == 0x300000000)
            buffer[m] = 0x5A;
      else if (ACID == 0x400000000)
            buffer[m] = 0x65;
      else if (ACID == 0x500000000)
            buffer[m] = 0x66;
      else if (ACID == 0x600000000)
            buffer[m] = 0x69;
      else if (ACID == 0x700000000)
            buffer[m] = 0x6A;
      else if (ACID == 0x800000000)
            buffer[m] = 0x95;
      else if (ACID == 0x900000000)
            buffer[m] = 0x96;
      else if (ACID == 0xA00000000)
            buffer[m] = 0x99;
      else if (ACID == 0xB00000000)
            buffer[m] = 0x9A;
      else if (ACID == 0xC00000000)
            buffer[m] = 0xA5;
      else if (ACID == 0xD00000000)
            buffer[m] = 0xA6;
      else if (ACID == 0xE00000000)
            buffer[m] = 0xA9;
      else if (ACID == 0xF00000000)
            buffer[m] = 0xAA;
```

```
        else


        ACID = c;
        ACID = (ACID << (i*4));
        i++;
        m++;
}
cout << " ";

ACID = d;
i = 1;

while (i<=6)
{
        ACID = (ACID & 0xF0000000);

        if (ACID == 0x00000000)
                buffer[m] = 0x55;
        else if (ACID == 0x10000000)
                buffer[m] = 0x56;
        else if (ACID == 0x20000000)
                buffer[m] = 0x59;
        else if (ACID == 0x30000000)
                buffer[m] = 0x5A;
        else if (ACID == 0x40000000)
                buffer[m] = 0x65;
        else if (ACID == 0x50000000)
                buffer[m] = 0x66;
        else if (ACID == 0x60000000)
                buffer[m] = 0x69;
        else if (ACID == 0x70000000)
                buffer[m] = 0x6A;
        else if (ACID == 0x80000000)
                buffer[m] = 0x95;
        else if (ACID == 0x90000000)
                buffer[m] = 0x96;
        else if (ACID == 0xA0000000)
                buffer[m] = 0x99;
        else if (ACID == 0xB0000000)
                buffer[m] = 0x9A;
        else if (ACID == 0xC0000000)
                buffer[m] = 0xA5;
        else if (ACID == 0xD0000000)
                buffer[m] = 0xA6;
        else if (ACID == 0xE0000000)
                buffer[m] = 0xA9;
        else if (ACID == 0xF0000000)
                buffer[m] = 0xAA;
        else


        ACID = d;
```

```
            ACID = (ACID << (i*4));
            i++;
            m++;
      }
      cout << "\n";
}

long int GenerateCompleteDF17Message()
{
      long int address;
      long int altitude;
      long int position;
      long int crcBits;


      //Generate Aircraft ID and store in our buffer
      address = GenerateAircraftID();
      completeMessageOdd[0] = address;
      completeMessageEven[0] = address;

      //Generate Altitude and store in our buffer
      altitude = CalculateAltitude();
      completeMessageOdd[1] = altitude;
      completeMessageEven[1] = altitude;

      //Generate the Latitude and Longitude Bits, functions will store
      values in our buffer
      CalculateLatBitsEven();

      //Generate the CRC bits and store them in our buffer
      CalculateCRC112BitsOdd();
      CalculateCRC112BitsEven();


      //Can be used for troubleshooting purposes if you need to see the
      odd message remove comments from code below.
      //cout << setbase(16) << "Odd Message : " <<
      completeMessageOdd[0] << " " << completeMessageOdd[1] << " " <<
      completeMessageOdd[2] << " " <<           //completeMessageOdd[3]
      << "\n";

      //Convert for easy entry into hex file
      ConvertForHexEditor(completeMessageEven[0],completeMessageEven[1]
,completeMessageEven[2],completeMessageEven[3]);
      ofstream myfile;
      myfile.open ("FinalMessage.bin", ios::binary | ios::app);
      myfile.write(buffer, 30);
      myfile.close();
      system("dd if=/dev/zero of=zeroes.bin bs=10 count=10");
      system("cat FinalMessage.bin zeroes.bin > SecondaryMessage.bin");

      //Can be used for troubelshotting purposes if you need to see the
      even message.
      //cout << setbase(16) << "ADS-B Message : " <<
      completeMessageEven[0] << " " << completeMessageEven[1] << " " <<
```

```
        completeMessageEven[2] << " " <<
        //completeMessageEven[3] << "\n";

        ConvertForHexEditor(completeMessageOdd[0],completeMessageOdd[1],
        completeMessageOdd[2],completeMessageOdd[3]);
        myfile.open ("SecondaryMessage.bin", ios::binary | ios::app);
        myfile.write(buffer, 30);
        myfile.close();
        system("cat SecondaryMessage.bin zeroes.bin > FinalMessage.bin");

        //clean up
        system("rm SecondaryMessage.bin");
        system("rm zeroes.bin");
        cout << "\n";
        cout << "The ADS-B Message has been generated and is ready for
        transmission." << "\n";
        return 0;
}



int main()
{
        int choice;
        int exit = 0;
        system("clear");
        cout << "This program will generate a Mode-S Downlink Format 17
        Message, also known\nas an Automatic Dependenet Surveillance
        Broadcast (ADS-B) Message.\n";


        while(exit ==0)
        {
        cout << "Press 1 <ENTER> to begin message generator.\n";
        cin >> choice;
        if (choice == 1)
        {
        printf("############################################################
        ## \n", GenerateCompleteDF17Message());
        }

        else
        {
        printf("Exiting.............\n");
        exit = 1;
        }
        }

        return 0;
}
```

Appendix C

Condition	Transition Latitude		Number of Longitude Zones, NL	
	Degrees (decimal)	32-bit AWB (hexadecimal)		
If |lat| <	10.47047130	07 72 17 54	**Then NL(lat) =**	59
Else if |lat| <	14.82817437	0A 8B 63 03	**Then NL(lat) =**	58
Else if |lat| <	18.18626357	0C EE B5 50	**Then NL(lat) =**	57
Else if |lat| <	21.02939493	0E F4 48 D6	**Then NL(lat) =**	56
Else if |lat| <	23.54504487	10 BE 3E 9F	**Then NL(lat) =**	55
Else if |lat| <	25.82924707	12 5E 12 29	**Then NL(lat) =**	54
Else if |lat| <	27.93898710	13 DE 23 2C	**Then NL(lat) =**	53
Else if |lat| <	29.91135686	15 45 32 43	**Then NL(lat) =**	52
Else if |lat| <	31.77209708	16 97 EF 0B	**Then NL(lat) =**	51
Else if |lat| <	33.53993436	17 D9 C2 3B	**Then NL(lat) =**	50
Else if |lat| <	35.22899598	19 0D 3E 35	**Then NL(lat) =**	49
Else if |lat| <	36.85025108	1A 34 62 2C	**Then NL(lat) =**	48
Else if |lat| <	38.41241892	1B 50 C4 78	**Then NL(lat) =**	47
Else if |lat| <	39.92256684	1C 63 AE 77	**Then NL(lat) =**	46
Else if |lat| <	41.38651832	1D 6E 2F 6C	**Then NL(lat) =**	45
Else if |lat| <	42.80914012	1E 71 2A 68	**Then NL(lat) =**	44
Else if |lat| <	44.19454951	1F 6D 5F 49	**Then NL(lat) =**	43
Else if |lat| <	45.54626723	20 68 71 E6	**Then NL(lat) =**	42
Else if |lat| <	46.86733252	21 53 F0 01	**Then NL(lat) =**	41
Else if |lat| <	48.16039128	22 3F 54 E9	**Then NL(lat) =**	40
Else if |lat| <	49.42776439	23 26 0C C7	**Then NL(lat) =**	39
Else if |lat| <	50.67150166	24 08 77 22	**Then NL(lat) =**	38
Else if |lat| <	51.89342469	24 E6 E8 E0	**Then NL(lat) =**	37
Else if |lat| <	53.09516153	25 C1 AD DF	**Then NL(lat) =**	36
Else if |lat| <	54.27817472	26 99 0A 48	**Then NL(lat) =**	35
Else if |lat| <	55.44378444	27 6D 3B A2	**Then NL(lat) =**	34
Else if |lat| <	56.59318756	28 3E 79 B3	**Then NL(lat) =**	33
Else if |lat|	57.72747354	29 0C F7 42	**Then NL(lat) =**	31
Else if |lat| <	58.84763776	29 D8 E2 B2	**Then NL(lat) =**	30
Else if |lat| <	59.95459277	2A A2 66 89	**Then NL(lat) =**	30
Else if |lat| <	61.04917774	2B 69 A9 E5	**Then NL(lat) =**	29
Else if |lat| <	62.13216659	2C 2E D0 D5	**Then NL(lat) =**	28
Else if |lat| <	63.20427479	2C F1 FC B2	**Then NL(lat) =**	27

Condition	Transition Latitude		Number of Longitude Zones, NL	
	Degrees (decimal)	32-bit AWB (hexadecimal)		
Else if |lat| <	64.26616523	2D B3 4C 60	**Then NL(lat) =**	26
Else if |lat| <	65.31845310	2E 72 DC 8C	**Then NL(lat) =**	25
Else if |lat| <	66.36171008	2F 30 C7 D8	**Then NL(lat) =**	24
Else if |lat| <	67.39646774	2F ED 27 0C	**Then NL(lat) =**	23
Else if |lat| <	68.42322022	30 A8 11 2E	**Then NL(lat) =**	22
Else if |lat| <	69.44242631	31 61 9B A1	**Then NL(lat) =**	21
Else if |lat| <	70.45451075	32 19 DA 2E	**Then NL(lat) =**	20
Else if |lat| <	71.45986473	32 D0 DF 12	**Then NL(lat) =**	19
Else if |lat| <	72.45884545	33 86 BA F3	**Then NL(lat) =**	18
Else if |lat| <	73.45177442	34 3B 7C CB	**Then NL(lat) =**	17
Else if |lat| <	74.43893416	34 EF 31 C5	**Then NL(lat) =**	16
Else if |lat| <	75.42056257	35 A1 E4 F8	**Then NL(lat) =**	15
Else if |lat| <	76.39684391	36 53 9E FA	**Then NL(lat) =**	14
Else if |lat| <	77.36789461	37 04 65 38	**Then NL(lat) =**	13
Else if |lat| <	78.33374083	37 B4 38 EB	**Then NL(lat) =**	12
Else if |lat| <	79.29428225	38 63 15 64	**Then NL(lat) =**	11
Else if |lat| <	80.24923213	39 10 ED 48	**Then NL(lat) =**	10
Else if |lat| <	81.19801349	39 BD A5 B3	**Then NL(lat) =**	9
Else if |lat| <	82.13956981	3A 69 0D 67	**Then NL(lat) =**	8
Else if |lat| <	83.07199445	3B 12 CB 8A	**Then NL(lat) =**	7
Else if |lat| <	83.99173563	3B BA 3A 96	**Then NL(lat) =**	6
Else if |lat| <	84.89166191	3C 5E 0E 31	**Then NL(lat) =**	5
Else if |lat| <	85.75541621	3C FB 4C 0F	**Then NL(lat) =**	4
Else if |lat| <	86.53536998	3D 89 48 8A	**Then NL(lat) =**	3
Else if |lat| <	87.00000000	3D DD DD DE	**Then NL(lat) =**	2
Else			**NL(lat) =**	1

Bibliography

1. Gorman, S., Dreazen, J. Y., & Cole, A. (2009). "Insurgents Hack U.S. drones: $26 Software Is Used to Breach Key Weapons in Iraq; Iranian Backing Suspected." *Wall Street Journal*, A1.

2. McCallie, L. D., Butts, J., Mills, R. (2011). Exploring Potential ADS-B Vulnerabilities in the FAA's NextGen Air Transportation System. Retrieved from Defense Technical Information Center database.

3. Smith, D. (2011). *Has compulsory ADS-B for Australia been properly thought out?* Retrieved February 15, 2011 from http://www.dicksmithflyer.com.au/article_111.php

4. Niles, R. (2006). *China looks at ADS-B.* Retrieved March 12, 2011 from http://www.avweb.com/avwebflash/briefs/191750-1.html

5. Office of the Joint Chiefs of Staff (2006). Joint Publication 3-13: *Information Operations* (February 13, 2006).

6. Marshall, A. (2009). *An Expanded Description of the CPR Algorithm*: *ADS-B 1090 MOPS, Revision B, Meeting #29*. July 21-24, 2009.

7. Gilbert, G. (1973). *Historical development of the air traffic control system. IEEE Transactions on Communications*, 364-375. Retrieved from IEEE Xplore.

8. Hal Stoen. (2001) VFR Flight. Retrieved February 15, 2012 from http://stoenworks.com/vfr%20flight.html

9. Cohen, B., & Smith, A. (1998). Implementation of a low-cost SSR/ADS-B aircraft receiver decoder (SY-100). Paper presented at the *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. the AIAA/IEEE/SAE,2* F44/1-F44/8 vol.2.

10. Livack, G. S., McDaniel, J. I., Battiste, V., & Johnson, W. W. (2000). The Human Element in Automatic Dependent Surveillance-Broadcast Flight Operations. Paper presented at the *Digital Avionics Systems Conferences, 2000. Proceedings. DASC. the 19th, 2* 5D4/1-5D412 vol.2.

11. Dunstone, G. (2010). *ADS-B Introduction/Tutorial* [PowerPoint slides]. Retrieved from http://www.scribd.com/tracon900/d/44649913-ADS-B-Introduction

12. Wolff, C. (2012). *Radar tutorial.* Retrieved February 15, 2012 from http://www.radartutorial.eu/02.basics/rp05.en.html

13. Ferrer, A., Carbonell, A., Perez, T., Larrosa, X. & Gonzalez, D. V. (2012). *Riding the skies.* Retrieved February 8, 2012 from http://surcandoloscielos.es/blog/frequently-asked-questions-ix-el- radar-3parte/

14. Purton, L., Abbass, H., & Alam, S. (2010). Identification of ADS-B System Vulnerabilities and Threats. *Australian Transport Research Forum 2010,* Canberra, Australia.

15. Boci, E. (2009). RF Coverage Analysis Methodology as Applied to ADS-B Design. Paper presented at the *Aerospace Conference, 2009 IEEE,* 1-7.

16. Harrison, M. J. (2006). ADS-X the next gen approach for the next generation air transportation system. Paper presented at the *25th Digital Avionics Systems* (Gorman, J.Y., & Cole, 2009) *Conference, 2006 IEEE/AIAA,* 1-8.

17. Wing-Shih Huang, Narayanan, R. M., & Feinberg, A. (2008). Multiple targets estimation and tracking for ADS-B radar system. Paper presented at the *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th,* 3.C.1-1-3.C.1-10.

18. Babbitt, R. (2010). FAA Aerospace Forecast: Fiscal Years 2011-2031. *U.S. Department of Transportation Federal Aviation Administration Aviation Policy and Plans.* Retrieved April 3, 2011 from http://www.faa.gov/about/office_org/headquarters_offices/apl/aviation_forecasts/aerospace_forecasts/2011-2031/media/2011%20Forecast%20Doc.pdf

19. Federal Aviation Administration, Automatic Dependent Surveillance Broadcast (ADS-B) Out Performance Requirements to Support Air Traffic Control (ATC) Service; OMB approval of information collection, 14 CFR Part 91, Federal Register, vol. 75(154), August 11, 2010.

20. Bruno, R., & Dyer, G. (2008). Engineering A US National Automatic Dependent Surveillance - Broadcast (ADS-B) Radio Frequency Solution. Paper presented at the *Digital Communications - Enhanced Surveillance of Aircraft and Vehicles, 2008. TIWDC/ESAV 2008. Tyrrhenian International Workshop on,* 1-6.

21. Boci, E., Sarkani, S., & Mazzuchi, T. A. (2009). Optimizing ADS-B RF coverage. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.* 1-10.

22. United States Congress. *Federal Aviation Act of 1958.* Public Law No. 85-726, 85th Congress, 2nd Session. Washington: GPO, 1958.

23. Rekkas, C., & Rees, M. (2008). Towards ADS-B Implementation in Europe. Paper presented at the *Digital Communications - Enhanced Surveillance of Aircraft and Vehicles, 2008. TIWDC/ESAV 2008. Tyrrhenian International Workshop on,* 1-4.

24. Cedrini, V., Zacchei, M., & Zampognaro, V. (2008). ADS-B 1090ES Implementation: The CRISTAL-MED project. Paper presented at the *Digital Communications - Enhanced Surveillance of Aircraft and Vehicles, 2008. TIWDC/ESAV 2008. Tyrrhenian International Workshop on,* 1-5.

25. Garcia, M. L., Hoffman, J. M., Rowley, J. L., & Stone, D. L. (2007). Test for success: Next generation aircraft identification system RF simulation. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2007. ICNS '07,* 1-10.

26. Valovage, E. M. (2009). A Method to Measure The 1090 MHz Interference Environment. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.* 1-8.

27. Stamper, W. (2005). "Understanding Mode S Technology". *RF Design Magazine,* 18-21.

28. Valovage, E. (2007). Enhanced ADS-B Research. *Aerospace and Electronic Systems Magazine, IEEE, 22*(5), 35-38.

29. Helfrick, A. (2004). *Principles of Avionics* (3rd ed.). Leesburg, VA: Avionics Communication Inc.

30. Bernays, D. J., Thompson, S. D., & Harman, W. H. (2000). Measurements of ADS-B Extended Squitter Performance in the Los Angeles Basin Region. Paper presented at the *Digital Avionics Systems Conferences, 2000. Proceeding DASC the 19th, 2* 7B1/1-7B1/8 vol.2.

31. Harman, W., Gertz, J., & Kaminsky, A. (1998). Techniques for improved reception of 1090 MHz ADS-B signals. Paper presented at the *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. the AIAA/IEEE/SAE, , 2* G25/1-G25/9 vol.2.

32. Nichols, R., Bernays, D. J., Spriesterbach, T., & Dongen, V. (2002). Testing of Traffic Information Service Broadcast (TIS-B) and ADS-B at Memphis International Airport. Paper presented at the *Digital Avionics Systems Conference, 2002. Proceedings. the 21st, , 1* 3A2-1-3A2-13 vol.1.

33. Sampigethaya, K., & Poovendran, R. (2011). Security And Privacy Of Future Aircraft Wireless Communications With Off-board Systems. Paper presented at the *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, 1-6.

34. Gilbert, T., & Bruno, R. (2009). Surveillance and Broadcast Services - An Effective Nationwide Solution. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.* 1-19.

35. Garcia, M. A., & Bruno, R. (2009). Ensuring Interoperability Between The Surveillance Broadcast Services System And ADS-B Avionics. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2009. ICNS '09.* 1-7.

36. IATA, Air Traffic Surveillance Views and Expectations of Airspace Users, Gongora, M. Retrieved November 20, 2011 from http://legacy.icao.int/nacc/meetings/2007/SURV_SEMI/Day03_IATA_Gongora.pdf, 2007.

37. ITT, U.S. ADS-B Ground Infrastructure Program, Herndon, Virginia, Retrieved October 3, 2011 from (http://events.aaae.org/sites/091001/assets/files/India%20Aviation%20Summit.pdf), 2009.

38. Babbitt, R.J. (2010). FAA announces performance standards for critical NextGen Avionics. *GPS World, Vol 5.*

39. Powell, J. D., Jennings, C., & Holforty, W. (2005). Use Of ADS-B And Perspective Displays To Enhance Airport Capacity. Paper presented at the *Digital Avionics Systems Conference, 2005. DASC 2005. the 24th,1.*pp. 4.D.4-4.1-9 Vol. 1.

40. Hollinger, K. V., Nickum, J. D., Peed, D. T., & Stock, T. M. (2006). Forecasting Aircraft Owner Equipage Responses To Potential FAA Actions. Paper presented at the *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, 1-9.

41. Cotton, W. (1973). Formulation of The Air Traffic System As A Management Problem. *Communications, IEEE Transactions on, 21*(5), 375-382.

42. McHale, J. (2010, December 2010). ADS-B In Brings Air Traffic Control Elements to The Cockpit. *Military & Aerospace Electronics,* 14-19.

43. Hall, T., Mackey, A., Nichols, B., Marksteiner, J., & Volpe, J. A. (2008). Prototype ADS-B System In The Midwest: Description And Lessons Learned. Paper presented at the *Integrated Communications, Navigation and Surveillance Conference, 2008. ICNS 2008,* 1-11.

44. Mozdzanowska, A. L., Weibel, R. E., & Hansman, R. J. (2008). Feedback Model of Air Transportation System Change: Implementation Challenges For Aviation Information Systems. *Proceedings of the IEEE, 96*(12), 1976-1991.

45. Raghavan, R. S. (2002). Performance Analysis Of 1090 Mhz Automatic Dependent Surveillance Broadcast (ADS-B) Using OPNET Modeler [ATC]. Paper presented at the *Digital Avionics Systems Conference, 2002. Proceedings. the 21st,1* 3E6-1-3E6-11 vol.1.

46. Smith, A. P., & Mundra, A. D. (2006). Impact Of ADS-B On Controller Workload: Results From Alaska's Capstone Program. Paper presented at the *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA,* 1-9.

47. Holdsworth, R., Lambert, J., & Harle, N. (2001). In-flight Path Planning Replacing Pure Collision Avoidance, Using ADS-B. *Aerospace and Electronic Systems Magazine, IEEE, 16*(2), 27-32.

48. Bernays, D. J., Drumm, A. C., & Shank, E. M. (2002). Validation Techniques for ADS-B Surveillance Data. Paper presented at the *Digital Avionics Systems Conference, 2002. Proceedings. the 21st, , 1* 3E2-1-3E2-9 vol.1.

49. Hicok, D. S., & Lee, D. (1998). Application of ADS-B For Airport Surface Surveillance. Paper presented at the *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. the AIAA/IEEE/SAE, 2* F34/1-F34/8 vol.2.

50. Hammer, J., & Elliott, D. (2010). Stochastic Analysis of ADS-B Integrity Requirements. Paper presented at the *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th,* 3.A.2-1-3.A.2-8.

51. Sampigethaya, K., Poovendran, R., & Bushnell, L. (2010). Assessment And Mitigation Of Cyber Exploits In Future Aircraft Surveillance. Paper presented at the *Aerospace Conference, 2010 IEEE,* 1-10.

52. Sitzabee, William E., Stepaniak, Michael J., Feng, Peter, P. Mission Assurance Issues with t Federal Aviation Administration's Policy to Implement GPS Navigational System. Air Force Institute of Technology, Wright-Patterson, AFB, Ohio.

53. Smith, D. (2006). ADS-B Terrorists' Dream. Retrieved September 8, 2011 from http://www.dicksmithflyer.com.au/Terrorists_dream.php

54. Sampigethaya, K., & Poovendran, R. (2010). Visualization & Assessment of ADS-B Security For Green ATM. Paper presented at the *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, 3.A.3-1-3.A.3-16.

55. Smith, A., Cassell, R., Breen, T., Hulstrom, R., & Evers, C. (2006). Methods to Provide System-Wide ADS-B Back-Up, Validation And Security. Paper presented at the *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, 1-7.

56. Besada, J. A., Garcia, J., De Miguel, G., Casar, J. R., & Gavin, G. (2000). ADS Bias Cancellation Based On Data Fusion With Radar Measurements. Paper presented at the *Information Fusion, 2000. FUSION 2000. Proceedings of the Third International Conference on, , 2* WEC5/23-WEC5/30 vol.2.

57. Zu Feng, & Zhang Xuejun. (2010). An Application Of Fuzzy Mathematics In ADS-B Data Validation. Paper presented at the *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on, , 3* 882-886.

58. Ettus, M. (2012). *Ettus Research LLC.* Retrieved August 10, 2011 from http://www.ettus.com

59. Lang, J-P. (2011). *GNU Radio Wiki.* Retrieved August 10, 2011 from http://gnuradio.org/redmine/projects/gnuradio/wiki

60. Foster, N. (2012). *GR-AIR-MODES.* Retrieved August 10, 2011 from https://github.com/bistromath/gr-air-modes

61. Cardew, Edward J. (2010). Decoding ADS-B Position. Retrieved August 10, 2011 from http://www.lll.lu/~edward/edward/adsb/DecodingADSBposition.html

62. Heriot-Watt University (2012). Cyclic Redundancy Check Polynomials Tutorial Retrieved January 15, 2012 from http://www.macs.hw.ac.uk/~pjbk/nets/crc/

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
22-03-2012	Master's Thesis	August 2010 – March 2012

TITLE AND SUBTITLE
Exploiting The Automatic Dependent Surveillance-Broadcast System Via False Target Injection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Magazu III, Domenic, Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENY)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCO/ENG/12-07

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)
Fill in

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

A new aircraft surveillance system, Automatic Dependent Surveillance-Broadcast (ADS-B), is being introduced by the Federal Aviation Administration (FAA) with mandated implementation in the United States by the year 2020. The rapid deployment of the system with current test-beds spread across the U.S. leaves very little chance for anyone to test the abilities of the system and more importantly the flaws of the system.

The research conducted within this thesis explores some of the weaknesses of the system to include the relative ease with which false aircraft targets can be injected. As part of a proof of concept, false ADS-B messages were successfully generated using a system comprised of GNU Radio, a Universal Software Radio Peripheral (USRP), and software developed by the author.

The ability to generate, transmit, and insert spoofed ADS-B messages on the display of a commercial ADS-B receiver, identified and exploited a weakness of the ADS-B system. Four demonstrations, conducted within an experimental environment, displayed the potential uses of the system created through this research and its associated impacts.

15. SUBJECT TERMS
ADS-B, FAA, Aircraft Surveillance, NextGen

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert F. Mills, PhD
U	U	U	UU	101	19b. TELEPHONE NUMBER *(Include area code)* 937-255-3636 x4527 Robert.mills@wpafb.af.mil

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18