4-29-2022

**ADS-B Aircraft Photography**

Gabriel Michael Adamson

Wesley Crouch

Jacob Elmore

Follow this and additional works at: https://louis.uah.edu/honors-capstones

Part of the Aviation Commons, and the Electro-Mechanical Systems Commons

**Recommended Citation**

Adamson, Gabriel Michael; Crouch, Wesley; and Elmore, Jacob, "ADS-B Aircraft Photography" (2022). *Honors Capstone Projects and Theses*. 682.

https://louis.uah.edu/honors-capstones/682

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.
ADS-B Aircraft Photography

by

Gabriel Michael Adamson, Wesley Crouch, Jacob Elmore

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 29, 2022

Honors Capstone Director: Dr.

Professor of Engineering

Student (signature)

Date 4/29/22

Director (signature)

Date 5/4/22

Department Chair (signature)

Date 5/4/22

William Wilkerson

Digitally signed by William Wilkerson

Date: 2022.05.04 14:09:27 -05'00'

Honors College Dean (signature)

Date
Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Gabriel Adamson____________________

Student Name (printed)

______________________________

Student Signature

4/29/2022

Date
Contents

Contents.......................................................................................................................................... 1

Executive Summary ......................................................................................................................... 5

Introduction .................................................................................................................................... 6

Big Picture ................................................................................................................................... 6
Requirements .................................................................................................................................. 6
  Marketing ................................................................................................................................ 6
  Engineering .............................................................................................................................. 7
Market Research ......................................................................................................................... 8
Design .............................................................................................................................................. 9

Alternate Designs/Tradeoffs ....................................................................................................... 9
Subsystems.................................................................................................................................... 10
  Raspberry Pi ........................................................................................................................... 11
  Servos .................................................................................................................................... 11
  LCD Touchscreen ................................................................................................................... 12
  Sense HAT .............................................................................................................................. 13
  GPS ......................................................................................................................................... 13
  Camera ................................................................................................................................... 13
  RTL-SDR .................................................................................................................................. 14
  Dump1090 ............................................................................................................................. 14
  Django Web Server ................................................................................................................ 14
Testing ....................................................................................................................................... 14
  Unit Testing ............................................................................................................................ 14
  Integration Testing ................................................................................................................ 17
  System Testing....................................................................................................................... 18
Project Management................................................................................................................. 20
  Scheduling.............................................................................................................................. 20
Executive Summary

The goal of this project is to engineer a device that will use the Automatic Dependent Surveillance-Broadcast (ADS-B) signals that are broadcasted by aircrafts to locate nearby aircrafts and take pictures of them. The final product will include a camera, a platform which allows pitch and yaw for the camera, a computer which controls the system, a user interface, an antenna which will receive ADS-B messages, Global Positioning System (GPS) and sensor modules for position and angle calibration, and the software which the computer will execute to perform the necessary tasks.

The main purpose of designing this system is to replicate the functionality of the NASA launch cameras in a cost-effective manner. The expected deliverable should be a self-calibrating device which will display an interactive user interface to ADS-B messages which allows the user to select an aircraft and take its picture. The primary purpose of this project is for hobbyists to be able to view pictures of different types of aircraft. Most hobbyists have to wait outside with their cameras in hopes that an airplane will pass in order to photograph it. However, this project waits for active ADS-B signals to come in from different airplanes, and it automates the picture taking process. As a result, this allows hobbyists to view photographs of airplanes without having to wait for the perfect opportunity to capture a photo themselves. This system could also be used by any organization which would like to record aircraft movement. For example, NASA has launch cameras which perform a similar function to our project. However, our team will develop a cost-effective version which could be more mobile and easily accessible for companies/individuals compared to the NASA launch cameras.
Introduction

Big Picture

Recording aircraft movement can require a significant amount of work. First of all, an individual must be within line of sight of an aircraft. This can actually be an issue because if a person is wanting to photograph aircrafts, then he or she must either go to a high air traffic location (such as an airport) or wait until one passes by overhead. The person must then be able to find the aircraft. This can be difficult, especially if the aircraft is at a high altitude. Finally, in order to photograph the aircraft, the photographer must have a camera on hand which is high enough quality to record at a long distance. The device constructed in this project eliminates these steps by performing the action automatically. This can either be used as a station to be controlled in-person, or it can be deployed at a separate location and controlled with a remote device.

To build this system, many engineering concepts from a variety of fields must be observed. The receiving of high-frequency messages requires the use of an antenna and software which can translate the messages based on the protocol. This project also required the integration of multiple electronic components. The Raspberry Pi operating system required some configuration changes as well. Web interface development also played a big role in the construction of the system. In order to determine the relative yaw and pitch needed for the camera, vector math calculations were also applied in the construction of this device. This project integrates concepts of radio communication, robotics, and software engineering to form a solution. The final product will be a system which will automatically photograph nearby aircrafts.

Requirements

Marketing

As the “Big Picture” section of this report mentioned, the overall objective of this project is to build a system which will automatically photograph a selected aircraft based on the ADS-B information received by the antenna. The overall system description can be broken down into multiple marketing sub-requirements. These requirements are:

1. The product must display a map showing aircraft locations.
2. The product must allow the user to select an aircraft to take a picture.
3. The product must automatically locate and photograph the selected aircraft.
These three requirements, although very general, outline the basic use of the system as a whole. Each of the engineering requirements are derived from this principle. These marketing requirements were verified with the customer’s project proposal and by the customer in meetings.

**Engineering**

The five main marketing requirements for the system are used to provide a guideline for design. The customer who initiated the project indicated that the specific implementation could be determined by the team. Using the marketing requirements, the following engineering requirements were developed:

1. The product must receive and process ADS-B messages from aircrafts. (Traced from marketing requirements 1 and 3).

2. The product must allow the user to view a map and select an aircraft. (Traced from marketing requirement 1).

3. The product must be calibrated such that it knows its location and rotation angles. (Traced from marketing requirement 3).

4. The product must calculate its relative position to a selected aircraft and determine the appropriate camera angle. (Traced from requirements 2 and 3).

5. The product must photograph an aircraft selected by the user. (Traced from marketing requirements 2 and 3).

These five main requirements are generalizations of specific functionality that is expected for the system. The marketing requirements described above are visualized in the following use case diagram.
Based on this use case diagram, some necessary components are apparent. In order to get the ADS-B messages from aircrafts, the system must be outfitted with an antenna to receive these messages. There must also be a software application which interprets the received signal. This system must then access a map and clearly mark the positions of aircrafts. The graphical user interface (GUI) must allow the user to select a specific aircraft and send a command to take a picture. In order to photograph the aircraft, a camera is needed on the system. To move the camera, motors are also required. In order to calculate the correct movements for the camera, sensors are needed to establish the system’s position and orientation. Therefore, the customer usage requirements imply certain design requirements as well.

**Market Research**

Similar projects to this one have also been developed, with some variation. The details for one example project can be found at this link: [https://medium.com/@arunvenkats/automating-the-capture-of-airplane-pictures-with-raspberry-pis-ads-b-and-iot-software-39e25ddcf3ea](https://medium.com/@arunvenkats/automating-the-capture-of-airplane-pictures-with-raspberry-pis-ads-b-and-iot-software-39e25ddcf3ea). This competitive project is very similar, but without the interface and requiring an internet connection, reducing its mobility. This
product is meant to replicate the NASA launch cameras, which would be the strongest competition. However, NASA launch cameras are expensive and not easily accessible. Our product, on the other hand, is very easy to reproduce and relatively cost-effective.

Aside from independent projects (such as referenced in the link above) and the NASA cameras, there does not appear to be any commercial competition for this product. This is likely due to the limited number of applications for this device, which implies that the market for this device would likely be very small.

Design

Alternate Designs/Tradeoffs

In order to meet the requirements of the project, the ADS-B Aircraft Photography Team collaborated to determine a design. The first design choice which had to be made was the type of computer to use to control the system. The team proposed using either a Raspberry Pi or a cheap laptop such as a Chromebook. One advantage of the laptop choice is improved usability. Laptops come equipped with default input peripherals (keyboard and mousepad) which the Raspberry Pi does not have which are more reliable than the Raspberry Pi LCD touchscreen option (which was the team’s decision for the interface with the Raspberry Pi). One main issue with the laptop option was the cost. The Raspberry Pi kit which the team purchased was $110. However, any other computer (such as a Chromebook) would likely cost several hundred dollars. Therefore, the Raspberry Pi option was a better economic choice. The biggest reason that the Raspberry Pi option was chosen, however, was due to its high configurability and hardware compatibility. The necessary devices for this project (camera, accelerometer/magnetometer, and camera platform motors) are compatible with the Raspberry Pi. Laptops generally have a limited number of ports, and they do not allow for the integration of hardware as does the Raspberry Pi via the pins. Also, the Raspberry Pi operating system (OS) is easily configurable compared to other operating systems. This allows for any customization necessary for the system to function. Because of these essential advantages, the Raspberry Pi option was chosen by the team.

Another design selection had to be made to determine the user interface for the Raspberry Pi. The two main options were to use a touchscreen or to use a combination of a screen and another input peripheral (such as a mouse/keyboard). The advantage of a touchscreen is that it would decrease the system complexity by eliminating the need for any other input devices. Because the system was meant to be portable, the elimination of a mouse and keyboard improves the system’s mobility. The downside to this option is that touchscreens tend to be unreliable at times. As will be discussed later in this report, it was for this reason that the team changed from using a touchscreen to using a mouse and keyboard. The specific
interface chosen was the Raspberry Pi 10.1inch LCD 10.1-inch Resistive Touch Screen. This type of screen was chosen because of its compatibility with the Raspberry Pi. There were several different sizes to choose from. However, it was determined that the screen should be no smaller than the interface chosen to improve the ease of use.

**Subsystems**

In this project, the team integrated the hardware peripherals with the computer and designed the software such that the device properly processes the input data and responds accordingly. The first step was to install the necessary software onto the Raspberry Pi. Then, all of the peripherals were connected and tested. The team ensured that the Raspberry Pi could receive the dump1090 information for processing purposes. After this was completed, software was developed which processes the dump1090 data and allows the user to determine which aircraft he or she wishes to photograph. Software was also designed which allows for the calibration of the camera. The GPS provides the location of the device, and the accelerometer gives its current orientation. It was necessary to build a program which reads the calibration data and uses it to calculate the appropriate camera angle. The team also provided software which interfaces with the camera stand to send the correct control commands. The system saves the captured images.

The following outline describes the final deliverables based on the system design determined in the planning phase of the project.

1. Hardware Components
   a. Raspberry Pi
   b. RealTek Software Defined Radio (RTL-SDR)
   c. LCD Display
   d. Automatically Rotatable Platform
   e. Camera
   f. Orientation Sensors
   g. GPS

2. Software Components
   a. User Interface Application
   b. ADS-B Message Processing Tool
   c. Calibration Application
   d. Camera Control Program
Raspberry Pi

For this project, the team elected to use a Raspberry Pi 4 for the computer which controls the entire system. The reason that a Raspberry Pi was chosen, as explained in the tradeoff analysis section of this report, was that it is compatible with the necessary hardware components such as the servo motors, the camera, and the sensors. There was also a need for four USB ports for connections with the RTL-SDR antenna, the input peripherals, the GPS, and the screen. For this reason, the team chose to purchase a model 4 Raspberry Pi.

In order to set up the Raspberry Pi, a micro-SD card was required to host the operating system. The team purchased a SanDisk 32 GB SD card to fulfill this requirement. The Raspberry Pi Bullseye OS was installed onto this card using the Raspberry Pi imager from https://www.raspberrypi.com/software. After installing the operating system, the card can be inserted into the Raspberry Pi. The device will then boot using the configuration in the /boot/config.txt file on the micro-SD card.

For the purposes of this project, the operating system configuration had to be altered. The touchscreen, camera, GPS, and Sense HAT devices all required separate configuration changes in order to operate. The setup, configuration, and installation procedures are detailed in the corresponding design sections for each peripheral device. The configuration file used is listed in the appendix.

Servos

In order to fulfill the need for an automatically rotatable platform, two servo motors were purchased. One of these provides rotational mobility for camera yaw, and the other servo controls camera pitch. The original servos purchased were small and relatively weak. Although the product was advertised as a camera platform, it quickly became clear that they were not strong or durable enough to support the needs of this project. Because of the insufficiency of these servos, another rotatable platform was purchased. Unit testing of these servos demonstrated that they could reliably support the camera and lens. This was an initial concern because of the added weight of the lens.

To integrate the servos with the Raspberry Pi, the three wires on the servo used for power and control were all attached to pins on the board. The two servos have identical schematics. The three wires that the servo uses are VCC (for voltage in), GND (ground), and PWM (pulse width modulation for control). The following image from the schematics shows a diagram for the servo.
The figure above shows that the operating voltage of the servo between 4.8 and 7.2 Volts. This means that the red pin will be connected to a 5 Volt pin on the Raspberry Pi. The control signal is sent through the orange wire using pulse width modulation. This means that the information which the servo receives is based on the duration of the duty cycle. This wire is plugged into a general-purpose input/output (GPIO) pin on the Raspberry Pi. The PWM period for these servos is 20ms. The Python API GPIO Zero is used by the Raspberry Pi to control the angle of the servos. This interface can use the 20-millisecond period to calibrate the servos such that the different pulse-widths produce the expected results by commanding the correct motor angle.

The first servo (which we will refer to as servo1) was used to control the azimuth of the camera. The red VCC wire was plugged into pin 2 on the Raspberry Pi for 5 volts input. The brown ground wire was attached to pin 9. The PWM wire for servo1 was plugged into pin 12, which is GPIO pin 18. Similarly, servo2 has GND connected to pin 14, VCC connected to pin 4, and PWM connected to pin 11 (GPIO 17). An image which describes the Raspberry Pi pins can be found in Appendix D. It is worth noting that these servos have only a 180 degree control angle, meaning the system will only be able to photograph aircrafts within the ±90 degrees of the direction it is pointing.

LCD Touchscreen

The final product does not actually use the touchscreen feature of the user interface. The issues with this device are outlined in the testing section of this report. Instead, a wireless mouse and keyboard are used to allow for user input. An HDMI connection between the screen and the Raspberry Pi is achieved through the HDMI0 port on the Raspberry Pi. The screen power comes from a USB port on the Raspberry Pi.

A lot of time was actually spent in the attempt to integrate the touchscreen with the rest of the system. The biggest issue was that certain boot configurations for the LCD screen and the camera and Sense HAT conflicted. A lot of trial and error was performed in order to determine the proper way to handle the multiple peripherals simultaneously.
Sense HAT

The Sense HAT (Hardware Attached on Top) is used for angular calibration. This sensor is essentially a compass for the system. The magnetometer on the Sense HAT provides a two-dimensional vector which does not account for a skewed orientation (meaning the magnetometer cannot independently report an accurate heading unless it is level with the ground). For this reason, the accelerometer from the Sense HAT is used to compensate for any tilt.

In order to receive data from the sensors, the Python API sense_hat was used. This allowed the system to receive raw x, y, and z data from both the magnetometer and the accelerometer. Code described in the following link https://ozzmaker.com/compass2/ was used to combine the accelerometer and magnetometer data in order to compensate for tilt. The final heading was then reported in degrees from north. This means that 0 degrees is due north, 90 degrees is due east, 180 degrees is due south, and 270 degrees is due west. The final heading is provided to the software application which calculates the camera angles.

GPS

The Global Positioning System in this product is also a calibration tool. The GPS is a dongle which is plugged into a USB port on the Raspberry Pi. When correctly configured, GPS information will be written to /dev/ttyACM0 on the Raspberry Pi. The Python GPSD software is used to interface with the GPS dongle. In order to correctly calibrate the system, the current position is needed in three dimensions. This means that the GPS dongle must get a 3D-fix, which requires at least four visible satellites.

Due to the unreliability of the GPS dongle as discussed in the testing section of this report, an alternate design strategy was implemented. Instead of waiting for a GPS response, the system can, instead, pull the latest GPS update from a saved location which is updated by another process as often as a 3D GPS position can be received. This speeds up the photography process by not relying on a quick response from the GPS.

Camera

The camera used in this project is the Raspberry Pi HQ Camera. It is connected to the Raspberry Pi’s camera port with a ribbon cable. Since the camera will be communicating through I2C, this option must be made available within the Raspberry Pi operating system. Because the computer is running the Bullseye version of the Raspberry Pi OS, the new camera library called “libcamera” is used to interface with the device.
RTL-SDR

The Realtek Software Defined Radio (RTL-SDR) is an antenna dongle that is plugged into one of the Raspberry Pi’s USB ports. The purpose of this antenna is to receive ADS-B messages from a frequency of 1090 megahertz. The antenna used is rotatable and extendable/retractable. rtl software installed on the Raspberry Pi can be used to interface with the RTL-SDR. For the purposes of this project, the RTL-SDR hardware and software does not require any customization because it is only intended to provide nearby ADS-B information for the dump1090 application.

Dump1090

Dump1090 is the software component we used to take the input from the SDR and decode the ADS-B signals. Dump1090 outputs the flight data to a json file, which is parsed by the JavaScript on the web interface, in the browser. There are multiple variations of the dump1090 software available on GitHub. The software that the team chose as a starting point for the web interface can be found at the following link: https://github.com/flightaware/dump1090.git.

Django Web Server

The default Dump1090 web interface can be served by any web server, as the code runs client side. For this project however, we needed the web interface to trigger the servos and photo on the server. Since the libraries for the servo is in Python, Django was used as the web server. A button was added to the default web interface which calls an API endpoint. This endpoint takes latitude, longitude, and altitude, as well as some data regarding plane type. This endpoint returns an id that can be used to request the photo once that subsystem takes the image.

Testing

Unit Testing

The independent testing of each peripheral device was conducted using Python tests. These tests have all been uploaded to the team’s GitHub organization within separate repositories.

Raspberry Pi

Raspberry Pi Bullseye operating system was installed onto the micro-SD card. Upon booting the computer with this micro-SD card, the operating system behaved as expected.
Configuration changes have been made to the operating system in order to accommodate the necessary hardware interfaces. These changes to the operating system do not disrupt expected behavior.

**Servos**

Initially, the servos had a “jitter” behavior while engaged. When a motor was commanded to go to a certain angle, the servo would go to the correct general orientation, but it would constantly shake instead of remaining still once it reached the correct angle. Additional research revealed that the problem was in the software configuration for the pin specification for the servos. After making the necessary changes, the servos performed as expected. Commands to certain angles resulted in the expected movement.

**Sense HAT**

**Sense HAT (B)**

The unit tests for the accelerometer indicate some issues either with the hardware on the sense HAT or with the software used to read this data. The vectors literally do not add up to the expected result based on the orientation of the accelerometer and based on basic trigonometric principles. The magnitude of the acceleration vector is expected to be approximately 1g when at rest. However, the magnitude of the acceleration vector is usually much higher than 1g when at rest, depending on the orientation. This could prove to be a problem down the line because the acceleration vector will be used in integration in order to help calibrate the system.

According to the unit test of the magnetometer, this device is fairly accurate. Third party software was used and modified in order to obtain the heading of the accelerometer. The heading was tested against calibrated compasses. The magnetometer is reasonably accurate in determining the heading if it is level with the ground. However, if the magnetometer is pitched, then the readings for the heading are incorrect. The acceleration vector should be used to account for this, but this vector is not always correct due to the accelerometer issues mentioned above.

**Sense HAT**

Due to the unreliability of the sensors based on the tests described above, another Sense HAT was purchased using personal funding. The original device was called Sense HAT (B), but the new device was the general version instead of the B-model. Unit tests performed on the new Sense HAT indicate that the new device will produce the expected results necessary for calibration.
GPS

The GPS has been tested using the GPSD Python software. When attempting to poll the
GPS with the unit testing software for the first time after booting the computer, there is rarely a
response. However, after resetting the processes and getting a GPS response, the GPS works
perfectly. It is because of the initial unreliability that the design change described in the
“Subsystems” section was made for the GPS.

HQ Camera

The HQ camera was tested with the Raspberry Pi. This did not initially work, but changes
to the Raspberry Pi’s boot configuration allowed the camera to work as expected. This boot
configuration was saved in the Tools repository under the ADS-B Aircraft Photography Github
organization.

Battery Module

The battery module was tested with the Raspberry Pi. The result of this test was a
corrupted micro-SD card. Although a new operating system had to be installed, no other
component of the system was damaged. It was determined that the battery module purchased
is faulty. Aside from the micro-SD card, the touchscreen was also damaged due to power
problems. This incident occurred during the final weeks of the semester. Because of this
problem, one micro-SD card, the touch input capability of the LCD touchscreen, and the battery
module itself were all lost. These components could not be replaced due to budget and
schedule constraints. However, our team did find workarounds for these issues in order to
produce a functional prototype.

As an alternate solution for powering the device, small portable batteries which
interfaced with the USB-C port were tested to determine if they could support the system.
Initial unit tests showed that, for the hardware tested, the external battery could power the
Raspberry Pi. However, the Raspberry Pi indicated that it was in a low power state.

RTL-SDR

After using the installation process outlined in this link:
youtube.com/watch?v=AfnHRTMWh44, the RTL-SDR antenna was tested using rtl_test. The
results of the unit tests indicate that the RTL-SDR antenna works as expected and is ready for
integration.

LCD Touchscreen

After installing the LCD-show software as described by the default setup, several
different configurations had to be adjusted. The ADS-B Aircraft Photography GitHub “tools”
repository contains detailed procedures followed in order to properly configure the Raspberry Pi touchscreen.

Integration Testing

**Dump1090 with RTL-SDR**

Following the installation and unit testing of the RTL-SDR, the dump1090 application was tested to determine if the surrounding aircrafts are being displayed. As expected, this functionality was successful. The use of the RTL-SDR within dump1090 has been determined to be very reliable. This functionality has remained unaffected by changes to the system, which is indicative of its robustness.

**LCD Touchscreen with HQ Camera**

Initial integration tests revealed that the LCD touchscreen configuration settings were not compatible with the HQ camera. Additional research and modification revealed that combining certain boot settings on the Raspberry Pi OS would allow for mutual compatibility for both the LCD touchscreen and the camera. These settings can be found in the configuration file listed in Appendix C.

After the battery module failure as described in the “Battery Module” unit testing section, the LCD touchscreen began exhibiting errors. It is likely that the damage to the touchscreen originated from the faulty battery module. The issue generated by this hardware failure was inconsistency with input position. After extensive testing, it was determined that the touchscreen could not be calibrated to a usable state. Therefore, the fourth and final USB port on the Raspberry Pi was used to install a mouse and keyboard for input.

**Calibration Software with Sense HAT and Servo Motors**

Initially, this software from the following source was used to get the necessary command angles for the servos: [https://github.com/paunstefan/view_angle_calculator](https://github.com/paunstefan/view_angle_calculator). However, upon integration testing between the Sense HAT and servo motors, it was discovered that the azimuth command was incorrect, causing servo1 (the servo which controls the camera yaw) to point the camera in an incorrect position. After further research into the issue, the following source was used for the relative azimuth calculation: [https://stackoverflow.com/questions/3932502/calculate-angle-between-two-latitude-longitude-points](https://stackoverflow.com/questions/3932502/calculate-angle-between-two-latitude-longitude-points). After making the required adjustments, the integration test was constructed using an arbitrary aircraft position to determine if the camera turned in that direction. The revised version of the system passed the integration test based on its rotation angles. However, system-wide testing was required in order to verify this functionality completely by seeing if an aircraft is actually within range.
Battery Module with External Hardware

When the portable USB-C battery was tested with all of the additional external hardware, it failed to provide consistent power to support the operation of the entire system. The battery was at best unreliable without attempting to use the servo motors simultaneously. When power was sent to the servo motors, the computer would crash due to the lack of power.

Due to budget and time constraints, a new battery module (besides the portable USB-C battery) could not be purchased to power the device. Instead, the team chose to use a regular outlet for the working prototype. It is determined that an external battery could likely use the USB-C to reliably power the system, but the battery would have to be significantly more powerful than the device tested.

System Testing

In order to perform system testing, the final product was deployed at the residence of the hardware design engineer. This location was chosen both because the product was already being stored there and because no location within a reasonable distance provided significantly more air traffic. To set up, the product was simply plugged into an exterior outlet. Because of the design, a system test does not actually require the use of a keyboard. The desktop contains the launcher script for the entire application. The only required steps is to click the “run.sh” application on the desktop, then use the application to select aircrafts for photograph.

For the system-wide testing, the test engineer would wait for an airplane to enter the field of view, select the airplane on the web interface, and then press the “Take Photo” button which appears at the top of the selected aircraft’s information.

A couple of issues were noticed when performing the system test. The first issue is that the camera platform does not always respond to every click of the “Take Photo” button. The second problem was that since the camera receives a single position with no updates during the time it takes for the camera to move into position and take a photo, the airplane will potentially move out of view before the photo can be taken. Several photos taken during the system testing exhibited this behavior, as shown in the following image:
In the above image, the airplane contrail can be seen at the bottom right, indicating that the airplane moved out of view. The third issue was that the camera is not powerful enough to take a picture of an airplane at the regular altitudes. This means that it can be very difficult for the camera to pick up an airplane unless it has a contrail. The fourth issue is that the camera angle, while usually very close, can sometimes be inaccurate. This can be for a number of reasons, such as expired GPS data, imprecise GPS data, imprecise Sense HAT data, imprecise servos, or erroneous software.

Other than the occurrences of the issues mentioned above, the system performed as intended. When an aircraft is selected on the interface, the system successfully gives the option for taking a photo. When this option is selected, the camera will then orient itself to point towards the aircraft. If the system has no errors, the aircraft is producing a contrail, and it has not moved out of view, then the system will take a photo of the aircraft. The following image shows the result of one such test.
Project Management

Scheduling

In this project, the team chose an agile development procedure. The implementation of this was a scrum-like development cycle. The main engineering requirements were broken down into several high-level tasks which needed to be performed. The team mapped out a schedule for the completion for each task based on the expected labor hours. The schedule below was roughly followed aside from some delays in task execution. The following list shows these high-level tasks:

- Set up Raspberry Pi – Fall 2021
- Set up RTL-SDR antenna and corresponding interface software – Fall 2021
- Set up dump1090 – Fall 2021
- Set up camera and camera control software – Spring 2022 Week 1
- Set up accelerometer and corresponding interface software – Spring 2022 Weeks 2-3
- Set up GPS and corresponding interface software – Spring 2022 Weeks 3-4
- Build camera mount – Spring 2022 Weeks 5-7
• Develop camera mount control application – Spring 2022 Week 8
• Edit dump1090 application – Spring 2022 Weeks 8-10
• Set up battery module – Spring 2022 Week 11
• Develop position/orientation calibration application – Spring 2022 Weeks 11-13
• Convert dump1090 output into camera platform commands – Spring 2022 Weeks 13-14
• Build physical platform to support the entire system – Spring 2022 Week 13-14
• Perform system tests – Spring 2022 Weeks 15-16

In periodic team meetings, each team member would be assigned a specific task from the list. If a team member had any difficulties, they could be address at these periodic meetings. Because of the clear constraints given by the customer, the overall design was quickly nailed down in team meetings prior to the creation of the list above. Once a task was completed, the member who completed it would be assigned a new task. This iterative approach was followed until every task was completed.

Costs

For the design of this system, many different components as described in the “Subsystems” section of this report had to be purchased. Funding for this project came from two main sources: UAH and the members of this team. A complete purchasing summary is provided in Appendix B. According to the records, UAH is responsible for $462 for this project minus tax. This includes the currently pending request amounting to $75 (which is presently out-of-pocket funding from a team member). In addition to the $462, an estimated $156 out-of-pocket has been spent on this project by the team members. Hardware failures, scheduling constraints, and budget limitations were all reasons why this money has not been filed to UAH purchasing. These numbers bring the total cost for this design project to be approximately $618. Although the design process required the purchasing of many different hardware components, it is clear that with a more streamlined production which uses only the necessary components, the total cost per unit would likely be closer to $450. Having already developed the software and designed the hardware, the production process would be very simple. This would just require the purchasing of the components, some time for assembly, extra time for software installation, and then testing.
Societal Impact Analysis

Social, Environmental, Health, and Safety

This project is not expected to have much of an impact on the environment. The project will not emit any harmful gases or pollution, and it also will not make much noise or affect the look of the surrounding environment. We expect the product to last for a number of years, as we are using components such as a Raspberry Pi, touch screen interface, and other hardware components. The only potential concern would be with the electronics. However, based on the small amount of power used in this system, this has been designated as a low risk with low impact. Based on the hardware and its application, there is not any likely event related to this project which could affect the physical safety of a user of this system.

Privacy and Legality/Political Impact

Based on the assessment of the team, it has been determined that legal and political aspects do not have a major impact on the design of this product or its use. However, there are a few things which must be taken into consideration. The use of a camera with this device could lead to privacy issues in the case of product abuse. In the United States, it is legal to photograph within public areas. Therefore, the regular use of this device is completely legal on public property or on other property for which the user of this device has the proper authorization to record. However, it is important that the user of this product confirm the local laws prior to use. Receiving ADS-B messages could also be viewed as a privacy concern, because any ADS-B messages broadcasted nearby can be logged by the device. This means that the behaviors of specific nearby aircrafts are known. Again, there is no local litigation against the reception of ADS-B messages, so the use of the RTL-SDR antenna to capture ADS-B data is in compliance with privacy laws.

Security and Ethics

There are some potential security concerns with this product. This system was designed to function entirely offline. However, additional functionality is available when connected to a network. There were no cybersecurity requirements for this design. However, our analysis of the system resulted in the discovery of some specific security weaknesses. One security weakness is the use of the default Raspberry Pi OS credentials without additional factors of authentication. This leaves the system more open to brute-force password attacks. Another potential flaw is in the software for the main application. Although the software design is resistant to client-side attacks, there are some internal processes which do not include robust error checking, which could lead to security issues affecting both privacy and availability.
However, the analysis of these weaknesses determined that they are low risk due to low likelihood of the security to be circumvented.

Due to the absence of security for the ADS-B protocol, there is a risk of malicious use of this system. It is worth reiterating that the system does not have the ability to perform any attacks on the ADS-B network. However, the information gathered using this system could potentially play a role in an attack on the ADS-B network. Since the messages broadcasted are in plaintext, this system can capture those messages to compromise integrity. For example, a message captured on this system could be rebroadcasted using a transmitter. A replay attack of this kind would cause great confusion for receivers of the counterfeit messages. There are several possible attacks which could be used against the ADS-B network. The following table lists some cybersecurity weaknesses within the ADS-B protocol.

<table>
<thead>
<tr>
<th>Attack classification</th>
<th>Level</th>
<th>Attack method</th>
<th>Harmful</th>
<th>Difficulty</th>
<th>Affected factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft Reconnaissance</td>
<td>PHY+APP</td>
<td>Eavesdropping</td>
<td>Low</td>
<td>Low</td>
<td>X</td>
</tr>
<tr>
<td>Replay Attack</td>
<td>PHY+APP</td>
<td>Message Injection</td>
<td>High</td>
<td>Low</td>
<td>X X</td>
</tr>
<tr>
<td>Aircraft Target</td>
<td>APP</td>
<td>Message Injection</td>
<td>Medium</td>
<td>Medium</td>
<td>X</td>
</tr>
<tr>
<td>Ghost Injection</td>
<td>APP</td>
<td>Message Injection</td>
<td>High</td>
<td>Low</td>
<td>X</td>
</tr>
<tr>
<td>Ground Station</td>
<td>APP</td>
<td>Message Injection</td>
<td>High</td>
<td>Low</td>
<td>X</td>
</tr>
<tr>
<td>Target Ghost Inject</td>
<td>APP</td>
<td>Message Injection</td>
<td>High</td>
<td>Low</td>
<td>X</td>
</tr>
<tr>
<td>Aircraft Flood Denial</td>
<td>PHY</td>
<td>Signal Jamming</td>
<td>Medium</td>
<td>Medium</td>
<td>X</td>
</tr>
<tr>
<td>Ground Station Flood Denial</td>
<td>PHY</td>
<td>Signal Jamming</td>
<td>Medium</td>
<td>Lower</td>
<td>X</td>
</tr>
<tr>
<td>Virtual Aircraft Hijacking</td>
<td>PHY+APP</td>
<td>Message Modification</td>
<td>High</td>
<td>High</td>
<td>X X</td>
</tr>
<tr>
<td>Virtual Trajectory Modification</td>
<td>PHY+APP</td>
<td>Message Modification</td>
<td>High</td>
<td>High</td>
<td>X X</td>
</tr>
<tr>
<td>Aircraft Disappearance</td>
<td>PHY</td>
<td>Message Deletion</td>
<td>High</td>
<td>Low</td>
<td>X X</td>
</tr>
<tr>
<td>Aircraft Spoofing</td>
<td>PHY+APP</td>
<td>Message Modification</td>
<td>High</td>
<td>Low</td>
<td>X X</td>
</tr>
</tbody>
</table>

Figure 5

The use of this system to assist in any of attacks against the ADS-B network would be both unethical and illegal.

Production

Manufacturability

This product would be relatively easy to manufacture. Since the necessary hardware, operating system configuration, and application software have all been designed and integrated, manufacturing copies of this system would be fairly simple. Since all of the necessary software is stored on GitHub, the applications necessary to drive the interface and hardware can be easily downloaded. Necessary hardware components and corresponding integration are clearly documented. This would greatly streamline the assembly process. The main difficulty in manufacturing that still needs to be address is the need for an outer “shell” to encapsulate the exposed electronics. The team recommends that a custom container be designed which covers the back of the screen and the Raspberry Pi itself. This was not
implemented for the current prototype due to both budget and schedule constraints. There are also additional hardware customizations which require extra work for installation.

**Sustainability**

The sustainability of this product depends not only on the availability of the current software and hardware used in the design and obtained from external entities, but also the availability of support personnel. It is inevitable that both software and hardware updates will be made in the future to the versions which are currently used. The relative compatibility of each subsystem as they are iteratively updated will determine the sustainability of this product. Sustainment could likely be managed sufficiently at low cost due to the low rate of person-hours which maintenance is estimated to require. As long as the overall structure of the product is held, it is expected that changes to software or hardware components can be easily accounted for.

**Robustness**

The current robustness of the product is determined to be low. As stated in the “manufacturability” section, the setup which is used in the initial prototype is not production ready. Any suboptimal weather conditions would greatly damage the entire system. There is also a concern with the servo motors used for the rotating platform. The servo motors have a “jitter” issue which is caused by an unstable power supply. This could lead to damage of the camera or wires. This also causes the results of the camera to be unreliable. Due to an issue with the battery module purchased for this project, it was not an option as a power source for this prototype. In order to achieve a level of robustness suitable for production, external power sources must be used to power the peripheral devices such as the servo motors and the Sense HAT. Since the servos are the hardware components which are in motion, they will likely be the first subsystems to fail. Overall, the system is expected to last as long as the servos remain functional, assuming that the product is carefully handled.

**Conclusion**

The final product from this design process was a working prototype of a device which will automatically photograph aircrafts. Although there are some issues with the system which cause unreliability, this prototype is an excellent proof of concept for the design. The design requirements were stated as follows:

1. The product must display a map showing aircraft locations.

2. The product must allow the user to select an aircraft to take a picture.
3. The product must automatically locate and photograph the selected aircraft.

This product successfully fulfills all of these requirements, with the caveat of some unreliability. As a prototype, this system demonstrates the overall success of the design. However, for this product to maximize its usefulness, higher quality hardware is required, such as a more powerful camera for recording distant aircrafts and servo motors for the camera mount which have full, 360-degree control angles.

Lessons Learned

One of the main lessons learned through this project would be to keep the main functional requirements the focus of the project throughout the design process. During hardware integration, the attention of the project was many times diverted to tasks such as touchscreen configuration, battery compatibility, and structural design, none of which were functional requirements. Although these aspects of the project would be useful for a final product, scheduling constraints require that the core functionality of the system be developed before adding any “finishing touches.” In other words, customer requirements should always drive the entire design process, even (and perhaps especially) on a scheduling level. The team did get a working prototype by the end of the semester; however, it would have been ideal to allocate more time for system testing after the implementation of the required capabilities rather than spend that time working on hardware design components which are helpful but unnecessary.

Another lesson learned is to thoroughly document your work and backup any important data. Because of an electrical issue described in previous sections of this report, the software on the device which was being used was lost. Fortunately, the team had prepared for such an occurrence by storing all necessary software and other important notes on GitHub. Redundancy can be a very important aspect of a project because disaster can occur without prior warning, which could jeopardize the project by putting it unreasonably behind schedule.
Appendix

Appendix A – Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADS-B</td>
<td>Automatic Dependent Surveillance-Broadcast</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-Purpose Input/Output</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HAT</td>
<td>Hardware Attached on Top</td>
</tr>
<tr>
<td>HDMI</td>
<td>High-Definition Multimedia Interface</td>
</tr>
<tr>
<td>HQ</td>
<td>High-Quality</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>MB</td>
<td>Megabytes</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>SD Card</td>
<td>Secure Digital Card</td>
</tr>
<tr>
<td>SDR</td>
<td>Software Defined Radio</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>VCC</td>
<td>Voltage Common Collector</td>
</tr>
</tbody>
</table>

Appendix B – Purchasing Links

UAH Funded

Note: Tax not included in UAH contribution.

**Raspberry Pi - $110**

https://www.amazon.com/Vilros-Raspberry-Basic-Cooled-Case/dp/B07TKFKKMP/ref=sr_1_8?dchild=1&keywords=raspberry%2Bpi%2B4&qid=1635438834&sr=8-8&sres=B08B6F1FV5%2CB077V5JTMV9%2CB07WHZW881%2CB07TC2BK1X%2CB08QQ4QCFJ%2CB07TKFKKMP%2CB07XTRK8D4%2CB08956P7LC%2CB07VYC6556%2CB07LY143F%2CB09HGKY8GM%2CB081TMCMSL%2CB08N4JK26%2CB0899VXM8F%2CB09DWG79T7%2CB07XTN5YRN&srpt=SINGLE_BOARD_COMPUTER&th=1

**Realtek Software Defined Radio (RTL-SDR) - $48**
https://www.amazon.com/NooElec-NESDR-Smart-XTR-Bundle/dp/B06Y1FDBBF/ref=sr_1_20?dchild=1&keywords=rtl+sdr&qid=1635440472&sr=8-20

**User Interface - $85**

https://www.aliexpress.com/item/1005003238622821.html?src=google&aff_fcid=7ffe6ccac049558d45de4f7f5b0bae-1635448381471-05611-UneMJZVf&aff_fsk=UneMJZVf&aff_platform=aaf&sk=UneMJZVf&aff_trace_key=7ffe6ccac049558d45de4f7f5b0bae-1635448381471-05611-UneMJZVf&terminal_id=390a9ac4e8e748f78d4ed1bd600840f

**Automatically Rotatable Platform - $10**

https://www.amazon.com/Camera-Platform-Anti-Vibration-Servos-Aircraft/dp/B0775R6JFF/ref=sr_1_6?dchild=1&keywords=PAN-TILT+camera+mount&qid=1635440912&qsid=146-8458449-9656663&sr=8-6&sres=B074WVL4NM%2CB07DLXWMVC%2CB000XZ3I22%2CB0775R6JFF%2CB01DXE6J90%2CB093W9JNCM%2CB077RJW34WB%2CB07X3Z3N8%2CB0787YXQP%2CB01C7X7YS%2CB07Z4GRG7%2CB08D3JNKFS%2CB00FHRV15C%2CB071GQLJLK%2CB008T17W9O%2CB07S3VJT3P&srpt=CAMERA_SUPPORT

**Camera - $50**

https://www.adafruit.com/product/4561?gclid=Cj0KCQjwkIgkBcARIsA1NMioKW6uSW17-J6uijWCuUM6tedqldrmWcY0VQ53chnxY7OC0kYpc28aAguFEALw_wcB

**Camera Lens + 3 year protection plan - $50 + $6**

https://www.amazon.com/gp/product/B08PYMBX9T/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

**Power Source - $46 total**

Battery HAT: $30

https://www.amazon.com/Geekworm-Raspberry-X706-Function-Compatible/dp/B096FT6THL/ref=sr_1_15?dchild=1&keywords=raspberry+pi+power+bank&qid=1635445937&sr=8-15&sres=B082CVWH3R%2CB01LAEX7J0%2CB07RC649ZC%2CB07Z6NXVNW%2CB09FN38Q7N%2CB07JYYRT7T%2CB07P5ZP943%2CB07L4RZP6B%2CB07H5T9J4L%2CB07CZDXDG8%2CB096FT6THL%2CB0899VXM8F%2CB09C5H5YYX%2CB07Y213F8S%2CB087FXLZZH%2CB087FV4W4NY

Batteries: $16

https://www.amazon.com/Skywolfeye-Flashlight-Tactical-Rechargeable-Adjustable/dp/B08LZPGRTK/ref=sr_1_14?dchild=1&keywords=18650+battery&qid=1635446085&sr=8-14&sres=B06ZZ8Y89F%2CB093FF73V3%2CB07SQLRMQH%2CB089LNM5B%2CB08BR1PD4S%2CB099RMW2C9%2CB08GQNKVXW%2CB089RBMLN%2CB08LZPGRTK%2CB08G181BMK%2CB0
89N1HS3Y%2CB08B8L4SKFNV%2CB08FJ53V3X%2CB07ZXZQMRS%2CB08RBQRL6N%2CB0874R85J2%2CB08Z1JDQNG%2CB07MHGRP5W%2CB08H1Q5B98%2CB08FDMGK2M

**Sense HAT (B) - $22**


**GPS - $14**


**Servos - $13**

https://www.amazon.com/gp/product/B08C7HTB2S/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

**Wires - $8**

https://www.amazon.com/gp/product/B01EV70C78/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1

**Independently Funded**

**Sense HAT – $31**

https://www.amazon.com/gp/product/B014HDG74S/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&psc=1

**Backup Ribbon Cables - $15**

https://www.amazon.com/gp/product/B089LM5D1T/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

**SD Cards - $40**
(One main, two backups)


**Supporting Platform - $50**


**Mouse and Keyboard - $20**

(Temporarily Donated)


**Appendix C – Configuration File**

/boot/config.txt

Note that lines beginning with ‘#’ are commented out, meaning they are not used. Also, be aware that the lines may wrap to appear as if they are new lines when they are actually continuations of previous lines. The ADS-B Aircraft Photography tools repository on GitHub contains an actual copy of the config.txt file used by the operating system.

```plaintext
# For more options and information see
# http://www.raspberrypi.org/documentation/configuration/config-txt.md
# Some settings may impact device functionality. See link above for details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
```
# overscan.
#framebuffer_width=1280
#framebuffer_height=720
# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1
# uncomment to force a specific HDMI mode (this will force VGA)
#hdmi_group=1
#hdmi_mode=1
# uncomment to force a HDMI mode rather than DVI. This can make audio work in
#hdmi_drive=2
# uncomment to increase signal to HDMI, if you have interference, blanking, or
#config_hdmi_boost=4
# uncomment for composite PAL
#sdtv_mode=2
# uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800
# uncomment to enable the lirc-rpi module
#dtoverlay=lirc-rpi
dtconfig=vc4-kms-v3d
dtconfig=irt-rpi
dtconfig=irt
# Uncomment this to enable the lirc-rpi module
# Uncomment some or all of these to enable the optional hardware interfaces
display_rotate=0
hdmi_group=2
hdmi_mode=1
#hdmi_mode=87
#hdmi_cvt 1024 600 60 6 0 0 0
# Additional overlays and parameters are documented /boot/overlays/README

#dtconfig=github
#tconphy=hdmi2 boost=4
#tconphy=hdmi2
#no display
# Uncomment to increase signal to HDMI, if you have interference, blanking, or
# Uncomment to force a HDMI mode rather than DVI. This can make audio work in
# Uncomment to force a specific HDMI mode (this will force VGA)
# Uncomment if hdmi display is not detected and composite is being output

framerate=720
framerate_width=1280
framerate_height=720
# overscan.
Appendix D – Raspberry Pi Pins

Sources

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9133434
https://www.raspberrypi.com/software
https://ozzmaker.com/compass2/