Passive Aircraft Detection System & Super Resolution Imaging Device

A means to do surveillance and increase security for less!

Group 2

Authors:

Alexander Alonzo	*Photonic Science & Engineering*
Xavier Colon	*Photonic Science & Engineering*
Carter Clavell	*Electrical Engineering*
Nathanael Olson	*Electrical Engineering*
Kai Doldan	*Computer Science*
Garrett Mosrie	*Computer Science*

Project Sponsor:

Roy H. Wubker Jr.	DOD	*Sponsor/Advisor*

Reviewer Committee:

Dr. Xun Gong	ECE	*Professor*
Dr. Stephen Kuebler	CREOL/CHM	*Professor*
Dr. Mark Heinrich	ECE	*Professor/Advisor*

Table of Contents

Ch1 Executive Summary……1
Ch2 Project Description……2
2.1 Motivation & Background……..2
2.1.1 RF Device……2
2.1.2 Camera Array……..3
2.2 Existing Products & Related Works……6
2.2.1 Hensoldt TwInvis……6
2.2.2 Thales Elvira……7
2.2.3 Dedrone RF-300……...7
2.2.4 Summary of Related Works……8
2.3 Goals……...9
2.3.1 RF Device & Camera Array Basic Goals……9
2.3.2 RF Device & Camera Array Advanced Goals……9
2.3.3 RF Device & Camera Array Stretch Goals……10
2.3.4 Software Basic Goals……10
2.3.5 Software Advanced Goals……10
2.3.6 Software Stretch Goals……..11
2.4 Objectives……11
2.4.1 RF Device & Camera Array Basic Objectives……..11
2.4.2 RF Device & Camera Array Advanced Objectives……...11
2.4.3 Camera Array Stretch Objectives……..12
2.4.4 Software Basic Objectives……...…..12
2.4.5 Software Advanced Objectives…….……12
2.5 Features & Functionalities……..……..13
2.5.1 RF Device……..13
2.5.2 Camera Array……...…..13
2.6 Engineering Specifications…….……..13
2.7 House of Quality……...16
2.7.1 RF Device House of Quality……..16
2.7.2 Camera Array House of Quality……..……..16
2.8 Hardware Block Diagram……...……..17
2.9 Software Block Diagram…….……..19
Ch3 Research……20

3.1 Camera Array……..……..20
3.1.1 Overview of Camera Technology (CMOS vs. CCD)……20
3.1.2 Evaluation for Selection……...…..25
3.1.3 Product Research & Selection…….……..28
3.2 Lens System……..31
3.2.1 Lens System Types for Multi-Camera Imaging……31
3.2.2 Evaluation for Selection……...…..34
3.2.3 Justification for Selection……..35
3.2.4 Specific Lens Selection & Optical Configuration…….……36
3.2.5 Conclusion of Lens System Research……..…..38
3.3 Image Stitching Methods for Super Resolution……..……..39
3.4 Adaptive Optics……....40
3.5 Evaluation for Selection for Image Stitching……42
3.5.1 Justification of Selection……43
3.5.2 Conclusion of Image Stitching Research……...45
3.6 RF Device……46
3.6.1 Overview of Antenna Technology……...……..46
3.6.2 Evaluation for Selection……...…..52
3.7 Overview of Spectrum Analyzer Technology…….……..52
3.7.1 Evaluation for Selection……...…..53
3.8 Overview of Mini PC……..…..54
3.9 Overview of Android Phone……...……..57
3.10 Overview of Wired Communication Protocol……..……..59
3.10.1 RF Device……..…..59
3.10.2 USB Hubs (Camera Array)……..60
3.11 Overview of Wireless Communication Protocols……..……62
3.11.1 Evaluation for Wired & Wireless Selection……...……..63
3.12 Power Requirements…….……..64
3.13 Overview of Reggedizing Components……..72
3.14 Programming Languages (Camera Array)……..74
3.15 Super Resolution Technique Analysis……..……..75
3.16 Front-End Frameworks…….……..81
3.17 Map APIs……..…..83
3.18 Database……..87

3.18.1 Relational Databases……....87
3.18.2 Non-Relational Databases……..……..88
3.19 Server Environment……..……..90
3.20 Web Stack…….…..92
3.21 Image Object Detection……..95
3.22 Anomaly Detection Algorithms……..98
3.22.1 Types of Unsupervised Anomaly Detection Algorithms….……99
3.23 Data Protocols……..104
3.24 Encryption Protocols……..…..107
Ch4 Standards & Design Constraints……...108
4.1 Optical Standards……108
4.2 Optical Design Constraints……109
4.3 Hardware Standards……110
4.4 Hardware Design Constraints……111
4.5 Software Standards……...……..113
4.6 Software Design Constraints……..……116
Ch5 Comparison of LLMs……..119
Ch6 Hardware…….……126
6.1 Camera Array Preliminary Hardware Diagram……..126
6.2 Spectrum Analyzer Hardware Design……...….127
Ch7 Software……...128
7.1 Data Collection and Transfer……..……128
7.2 SpectrumAnalyzer SDK…….……130
7.3 Algorithms for Signal Analysis……..……132
7.4 Front End……137
7.5 Anomaly Detection Algorithm Design……...……142
7.6 API Development……143
7.7 Deployment Planning and Cloud Usage…….…144
Ch8 Testing……..147
8.1 Camera Array……..147
8.1.1 Midterm Demo……147
8.1.2 Final Demo SD1……...……...148
8.1.3 Additional Optics Testing……...….150
8.2 RF Device……..…….……159

8.2.1 RF First Set of Tests**…….159**
8.2.2 RF Testing Continued**……...…168**
8.2.3 RF Testing Software**……...….169**
8.3 Object Detection Testing**…….…172**
8.4 Anomaly Detection Algorithm Testing**……..…….176**
Ch9 Administrative Content……..178
9.1 Finances**……..……178**
9.2 Project Timeline**…….……...…..180**
Ch10 Conclusion…….181
Appendix Bibliography…….….184

List of Tables & Figures

Tables:

2.1 Prototype Specifications and Features
2.2 Unit Components Specifications
2.3 Camera Array Engineering Specifications
3.1 Comparison of CMOS & CCD
3.2 Comparison of Specific Camera Technology
3.3 Comparison of Lens System Types
3.4 Comparison of Resolution Strategies and Corrective Optics
3.5 Comparison of Antenna Technology
3.6 Comparison of Spectrum. Analyzers
3.7 Comparison of MiniPC Technology
3.8 Comparison of Mobile Device Technology
3.9 Comparison of Wired Protocols
3.10 Comparison of USB Hubs
3.11 Comparison of Wireless Protocols
3.12 Power Requirements of Components
3.13 Comparison of Power Supply Technology
3.14 Comparison of Ruggedizing Components
3.15 Comparison of Material
3.16 Comparison of Super Resolution Methods
3.17 Comparison of Front-End Frameworks
3.18 Comparison of Map APIs
3.19 Comparison of Relational Databases

3.20 Comparison of Non-Relational Databases
3.21 Comparison of Server Environments
3.22 Comparison of Object Detection Models
3.23 Comparison of Anomaly Detection Algorithms
3.24 Comparison of Data Protocols
3.25 Comparison of Encryption Protocols
7.1 Comparison of Clouds
8.1 Preliminary Budget Table for RF
8.2 Preliminary Budget Table for Camera Array
8.3 SD1 Milestones
8.4 SD2 Milestones

Figures:

2.1 Passive Radar House of Quality
2.2 Camera Array House of Quality
2.3 RF Device Hardware Block Diagram
2.4 Optical System Hardware Block Diagram
2.5 Schematic Diagram of Camera Array System
2.6 Software Block Diagram
3.1 CMOS & CCD Sensor Charge Readout Mechanic
3.2 Exposure Time and Duration of Each Shutter Type
3.3 How super res. Functions
3.4 Motion Blur Example with Plane
3.5 Isotropic Antenna Radiation Pattern
3.6 VSWR of LPDA
3.7 Radiation Pattern
3.8 Aventas AX-37A Radiation Pattern
3.9 MSE Reliability Example
3.10 Nearest Neighbor, Bi-linear, & Bi-cubic Interpolation
3.11 Back Projection Algorithm
3.12 ESRGAN Texture Upscaling
3.13 Web-stack Comparison
6.1 Camera Array Hardware Setup
6.2 RF Device Hardware Setup

7.1 Moving Average Filter
7.2 Low-Pass Filter
7.3 Band-Pass Filter
7.4 Rough Draft of Desktop Design
7.5 Rough Draft of Mobile Design
7.6 User Flow Diagram
8.1 Final Demo Optical Setup
8.2 Test of Camera Array with KIMTECH Box
8.3 KIMTECH Box Stitched Image
8.4 KIMTECH Box Super Resolution Image
8.5 Test of Camera Array with Sprite Bottle
8.6 Sprite Bottle Stitched Image
8.7 Sprite Bottle Super Resolution Image
8.8 Test of Camera Array with Mug
8.9 Mug Stitched Image
8.10 Mug Super Resolution Image
8.11 Least Narrow FOV Lens System
8.12 Middle FOV Lens System
8.13 Most Narrow FOV Lens System
8.14 RF Device initial test setup
8.15 Key Fob Test Horizontal 1 Meter
8.16 Key Fob Test Horizontal 5 Meter
8.17 Key Fob Test Horizontal 10
8.18 Key Fob Test Horizontal 50
8.19 Key Fob Test Horizontal 100
8.20 Key Fob Test Horizontal 170
8.21 Key Fob Test Horizontal 175
8.22 Key Fob Test Horizontal 180
8.23 Key Fob Test Vertical 1 Meter
8.24 Key Fob Test Vertical 5
8.25 Key Fob Test Vertical 10
8.26 Key Fob Test Vertical 50
8.27 Key Fob Test Vertical 100
8.28 Key Fob Test Vertical 115

8.29 RF Environment, No Aircraft
8.30 Demonstration of Detection
8.31 Spike SDK
8.32 Data from RF Device
8.33 Data at Receiving End
8.34 Sample Image of Plane in Clear Sky
8.35 Result Image of Plane in Clear Sky
8.36 Result Image of Plane in Cloudy Sky

Chapter 1: Executive Summary

This project presents a low-cost, passive aircraft detection and super-resolution imaging system, addressing critical gaps in surveillance infrastructure. Leveraging passive RF (radio frequency) detection and computational super-resolution techniques, this system aims to provide affordable, stealthy, and scalable surveillance capabilities, particularly for radar “dead zones” where illegal aircraft activity may go unnoticed. Sponsored by Roy H. Wubker Jr. of the U.S. Department of Defense’s Southern Command (SOUTHCOM), this initiative unites students from Photonic Science, Electrical Engineering, and Computer Science to develop a field-deployable prototype.

Many global regions, particularly along borders, suffer from limited surveillance coverage due to the high costs and infrastructural demands of active radar systems. These "radar dead zones" can be exploited for illegal activities such as unauthorized aircraft entry, smuggling, and drone-based reconnaissance. The absence of reliable surveillance in such areas compromises national security and border integrity.

To address this challenge, our team designed a dual-component system capable of functioning independently or in the ideal case, in tandem. The first component, a passive RF detection unit, identifies airborne objects by capturing reflected ambient RF signals, including FM radio, GPS signals, or engine noise. Unlike active radar, which transmits signals and waits for their return, this system only listens—drastically reducing power consumption and increasing stealth. The second component, a camera array utilizing super-resolution imaging, captures synchronized images from multiple low-cost cameras, fusing them into a high-resolution composite. This optical system enables the verification of potential threats and improves detection reliability.

The passive RF detection system is the cornerstone of the electronic surveillance aspect. It uses a log-periodic antenna to capture ambient RF signals, which are analyzed using a spectrum analyzer connected to a compact computing system. This approach allows the detection of aircraft based on RF reflections alone, eliminating the need for signal transmission. The system is designed to cover airspace up to 12 miles away and detect objects with a radar cross-section (RCS) of at least 3 square meters. Future iterations will be powered using renewable energy sources such as solar panels, ensuring autonomous operation even in remote locations. To ensure timely decision-making, the detected information will be transmitted in real time via a web-based application.

The super-resolution camera array complements the RF system by providing visual confirmation. It employs three synchronized cameras, each equipped with a different lens system—telephoto, wide-angle, and ultra-wide—to capture varying fields of view at a specified magnification. These low-resolution grayscale cameras (Arducam OV9281) are chosen for their high frame rates and global shutter design, which eliminates motion blur and ensures accurate image capture. Images from each camera are stitched together and processed through a super-resolution algorithm, generating a higher-quality composite image. This enhanced image is then passed to detection software, which improves object identification performance.

Our proposed system draws inspiration from advanced commercial products, while maintaining a significantly lower cost. The Hensoldt TwInvis, for instance, is a passive radar system that uses environmental RF signals for stealth detection but lacks an optical verification mechanism. The

Thales Elvira integrates RF and optical sensors for comprehensive tracking, but its complexity and cost make it less accessible. The Dedrone RF-300 specializes in drone detection using triangulation and advanced software but requires multiple units and proprietary tools to function effectively.

In contrast, our design aims to uniquely combine RF and optical systems in a single, affordable unit. By integrating super-resolution imaging, we offer a secondary verification layer that enhances detection accuracy and reduces false positives. This system is optimized for simplicity, modularity, and cost-efficiency, making it suitable for both civilian and defense applications in resource-constrained environments

This project represents a promising step in the growth of advanced surveillance technologies. The success of this project will offer a cost-effective alternative to high-end detection systems, capable of both passive RF sensing and enhanced optical verification. The system's modularity and scalability allow it to be adapted for applications ranging from border security and military reconnaissance to wildlife monitoring and environmental research.

The ability to generate high-resolution images from low-cost cameras, combined with passive RF detection, demonstrates how computational imaging and signal processing can compensate for hardware limitations. This opens the door to broader adoption of intelligent sensing systems, particularly in regions or organizations constrained by budget or infrastructure. By bridging the gap between affordability and performance, this project lays the groundwork for future advancements in autonomous, multisensor surveillance platforms.

Chapter 2: Project Description

2.1 Motivation & Background

2.1.1 RF Device

Along the borders of many different countries, it is very hard for radar and air traffic controllers to detect aircraft. The reason being that it is generally far too expensive for many countries to place large high-power radar systems all over their country. These systems can cost tens of millions of dollars and require advanced infrastructure to implement. As a result, certain locations become dead zones, areas where there are no effective means of surveillance. These so-called "dead zones" make it very easy for illegal activity to happen such as illegal immigration, drug smuggling, or drone surveillance. Not only that, these aircraft may not be using a transponder of any form in order to avoid detection from air traffic control or similar systems. Naturally, this makes it an issue for governments all over the globe. Our project aims to provide a cost-effective and stealthy solution in an effort to help increase security and give governments the peace of mind they are seeking.

As aspiring engineers, senior design projects are a cornerstone of our education and provide several key benefits which include but are not limited to, hands-on application of knowledge,

real world problem solving, collaboration and teamwork, and portfolio development. As a group, we must learn to problem solve under pressure, work as a collective unit toward a common goal, and design and showcase a functional product using an innovative approach. With that said, we wanted to work on a project that addressed a significant, real-world challenge.

We knew we had an incredible opportunity before us and a chance to solve a real world problem that incorporated our unique skills and expertise. This problem was introduced to us by Roy H. Wubker Jr. who, through the Department of Defense's Southern Command (SOUTHCOM) is sponsoring this project. Based on the introduction from our sponsor we knew we had an incredible opportunity before us and a chance to solve a real world problem that incorporated our unique skills and expertise. Our sponsor's initial recommendations for solving this problem involved RF detection which led us to brainstorm more ideas, especially those that involved optics and photonics.

This problem poses many challenges but has widespread impact which is why our group decided to take on this project. From this project, we learned about how to implement a solution to this kind of system and optimize the strengths of our solution. Our group tried to make our solution as low cost as possible and, simultaneously, as stealthy as possible. The harder it is to detect, the longer our proposed solution can relay vital information. Longevity in this case, will also help keep the replacement cost down.

Active radar is a modern radar technique that is used all over the globe to track the direction, altitude, and speed of airplanes. It works by sending out a Radio Frequency beam from an antenna to the desired target direction; The RF beam bounces off of the desired object and then makes its way back to the antenna which is listening for the reflection. Based on the time-of-flight of the signal, it can determine how far the object is, its speed, direction, and more.

Our solution, however, is to use passive radar. This will not only lower the cost but it will also provide a more simplistic solution as compared to an active radar counterpart. Passive radar works by instead of sending out a beam of RF energy, which requires lots of power, space, and infrastructure, it uses existing RF signals in the environment, and listens for those RF signals to bounce off of the desired object. RF signals in the environment appear in the form of things like FM Radio, GPS, or even the natural resonance of an airplane's motor. By only listening to the radio signals that are reflected off of the object, the amount of power consumed will be cut down. Furthermore, the system will be much harder to find as there was no RF beam emitted from the system to be tracked by enemy countermeasures. It will essentially be completely hidden to any aircraft.

2.1.2 Camera Array

High-resolution imaging plays a vital role in applications such as aerial surveillance, environmental monitoring, and remote sensing. Traditional imaging systems capable of capturing fine details at long distances rely on high-performance cameras, which often cost upwards of $10,000. This high cost limits access to advanced imaging technology, particularly for smaller research teams and organizations with budget constraints. To address this issue, we propose a

cost-effective alternative by utilizing an array of low-cost cameras to generate high-resolution images comparable to those of expensive imaging systems.

The key innovation of this project lies in the application of super-resolution techniques to combine multiple low-resolution images into a single, high-resolution output. This technique enhances object detection capabilities by increasing image detail and clarity, making it particularly beneficial for tracking small and distant objects such as aircraft or wildlife. Once a high-resolution image is generated, a detection algorithm is applied to identify and analyze objects within the image. This detection performance will then be compared to results obtained using a standard, non-super-resolution image to evaluate the effectiveness of our approach.

Super-resolution is a computational imaging technique that reconstructs a high-resolution image by integrating multiple lower-resolution images. This approach is particularly advantageous in applications where imaging sensors are limited by physical constraints, such as cost, size, or atmospheric interference. By increasing image clarity, super-resolution allows for enhanced feature extraction, improving object recognition and detection accuracy.

In many real-world applications, such as aircraft detection in aerial reconnaissance, objects appear small relative to the overall image. Standard low-resolution cameras may struggle to detect these distant objects due to pixelation and loss of fine detail. Super-resolution reconstruction mitigates this issue by restoring edges, textures, and small-scale features that might otherwise be lost in individual low-resolution images. This enhanced resolution facilitates more reliable object detection, especially in scenarios where small details differentiate targets from their surroundings.

To evaluate the effectiveness of super-resolution in object detection, we compare detection performance between super-resolution images and non-super-resolution images. A trained detection algorithm, such as a convolutional neural network (CNN) or machine learning-based object recognition system, will be applied to both image sets. Key performance metrics, including detection accuracy, false positives, and processing speed, will be analyzed to determine the impact of super-resolution on object identification. By systematically testing the system with and without super-resolution, we can quantify its benefits and assess whether it significantly improves detection capabilities in challenging imaging environments.

The effectiveness of our super-resolution system heavily depends on the arrangement of the camera array. The optimal configuration must balance coverage, alignment precision, and computational feasibility. A curved camera array offers significant advantages by ensuring that each camera is directed toward a common focal point, reducing parallax errors and simplifying image stitching. However, alternative configurations, including rectangular, circular, and elliptical layouts, have also been explored to determine the most effective design. The final configuration was chosen based on its ability to minimize distortions and maximize image quality.

Another critical factor in our setup is camera synchronization. Since the system will attempt to capture images of moving objects, it is essential that all cameras take their images simultaneously. If there is a significant delay between exposures, objects may shift positions between frames, causing misalignment in the final composite image. This issue can lead to

ghosting artifacts or detection errors, particularly in applications such as tracking aircraft or monitoring fast-moving vehicles. To prevent this, hardware and software synchronization techniques have been implemented to ensure that all cameras capture frames at the exact same moment.

Additionally, each camera in the array is equipped with specialized lens systems tailored for specific imaging tasks. A combination of telephoto, wide-angle, zoom, and fisheye lenses are employed to enhance coverage and resolution. Telephoto lenses will be used to capture fine details in distant objects, while wide-angle and fisheye lenses will provide broader context. This multi-lens approach ensures that the final composite image benefits from both high detail and wide coverage.

Atmospheric turbulence and environmental factors, particularly in Florida's humid climate, introduce distortions that degrade image quality. Over long distances, heat waves and varying air densities cause light to refract unpredictably, resulting in blurring and loss of detail. To mitigate these effects, the system will, in the future, incorporate phase corrector plates and computational wavefront correction techniques. Phase corrector plates work by introducing an optical element that compensates for wavefront distortions before the light reaches the sensor. These plates can be custom-designed to correct for specific atmospheric aberrations encountered in our imaging environment.

The final stage of the project involves analyzing the effectiveness of super-resolution in object detection. A trained detection algorithm is applied to both the super-resolution and non-super-resolution images, and the results have been compared across multiple test scenarios.

One key metric to evaluate will be detection accuracy, which measures how effectively objects are identified within each image type. Higher-resolution images should theoretically allow the detection algorithm to recognize smaller or partially obscured objects that might be missed in standard images. The impact of resolution enhancement on false positive rates will also be analyzed, ensuring that the system does not introduce additional noise or artifacts that could interfere with object classification.

Additionally, we will examine computational efficiency—while super-resolution processing enhances image quality, it also introduces additional processing time. It is essential to determine whether the improvements in detection accuracy justify the computational overhead, particularly in real-time applications such as surveillance or aerial monitoring. This analysis will help optimize the trade-offs between image enhancement and system performance, ensuring that the final implementation remains practical for real-world use.

By systematically comparing the performance of super-resolution and non-super-resolution images, this project will provide valuable insights into the benefits and limitations of this approach. If the results demonstrate a significant improvement in detection accuracy, it would validate the feasibility of using low-cost camera arrays as an alternative to expensive high-resolution cameras.

The development of a cost-effective super-resolution imaging system has significant implications for various fields. In aerial surveillance, such a system could enable more precise tracking of

airborne objects, improving situational awareness for both civilian and military applications. In environmental monitoring, it could help researchers track wildlife populations, analyze deforestation patterns, or assess the impact of climate change. Additionally, in disaster response, high-resolution imaging from affordable drones could assist emergency teams in damage assessment and resource allocation.

Beyond immediate applications, this project lays the groundwork for future advancements in scalable imaging systems. By demonstrating that high-resolution imaging can be achieved using low-cost components, this research could pave the way for autonomous vehicle vision systems, space-based imaging, and next-generation remote sensing technologies. As computational imaging techniques continue to advance, the principles established in this study could be expanded to even more sophisticated imaging applications.

In conclusion, this project seeks to bridge the gap between affordability and performance in high-resolution imaging. By leveraging a camera array, super-resolution algorithms, and object detection analysis, we aim to demonstrate that cost-effective imaging systems can achieve results comparable to expensive commercial solutions. If successful, this project could contribute to a broader adoption of advanced imaging technologies, making high-resolution surveillance, monitoring, and research more accessible to a wide range of industries.

2.2 Existing Products & Related Works

2.2.1 Hensoldt TwInvis

The TwInvis system by Hensoldt is a passive radar that leverages signals in the environment, such as those from FM radio, digital television, or mobile communication devices. It does not emit any signals itself, making it undetectable by traditional radar-detection systems. The TwInvis system monitors reflections of the aforementioned signals off objects, including aircraft, to detect and track their movement. TwInvis passively listens to ambient RF signals emitted by sources in the area, processes how they bounce off objects in the environment, and with an array of these devices, can determine the speed and trajectory of the aircraft.

This technology is very similar to the solution we are proposing and can be applied in stealth aircraft detection, civil air traffic monitoring, and even compliments active radars when integrated with military systems. Since it does not emit its own signals, it avoids detection and interference from hostile forces and makes use of already existing infrastructure which simultaneously allows for wide regional coverage in areas dense with RF signals. However, in areas in which the density and quality of ambient RF signals is poor, this technology is not of much use. In addition, this system requires advanced algorithms to separate useful data from just noise, which increases computational demands.

A key difference between the TwInvis and our proposed solution is the integration of optical components. TwInvis can only function in areas where RF signals are dense enough and only works using RF detection. Our project incorporates an optical component in the form of an infrared camera. This approach can provide a more complete picture of the environment that is

being monitored. By implementing an optical component, we can introduce a sort of 2-step verification system to reduce false positives and provide more accurate results. Not only that, capturing video of the aircraft could help the user determine the threat level and could clue the user into the intentions of the aircraft entering their airspace.

2.2.2 Thales Elvira

The Elvira system by Thales is a multisensor electronic support system designed for detection, classification, and tracking of aircraft. It integrates RF signal detection with additional tracking systems, such as optical or infrared sensors. This approach provides a holistic view of the airspace. Elvira detects emissions from communication devices, radars, or other onboard systems of aircraft. It then correlates RF data with other inputs, such as infrared tracking, to improve target identification and accuracy. In a broader sense, Elvira can use advanced algorithms to analyze emission types, origin, and even determine threat level.

This system is ideal for border surveillance as it operates covertly without filing flight plans or transponders. It can also be integrated into ground stations or shipboard systems to enhance situational awareness. By integrating multiple sensors, it is able to minimize false positives and improves tracking in challenging environments. The issue is the cost. The integration of advanced sensors and processing units makes it very expensive to both deploy and maintain. Since the main method of gathering data is the RF system, it could be susceptible to jamming or other countermeasures. In addition, the system is not simple and would require specific training to be effective.

This system, although more complex, does have many similarities to our proposed solution. Both the Elvira system and our project work based on already available RF signals in the area. Using the already existing infrastructure of radio towers, television stations, and mobile devices is a great way to maintain stealth while obtaining the information you seek. However, Elvira systems main trigger is based on RF detection with the secondary trigger being the infrared search and track system. Our project aims to use the RF detection method as the main trigger and the infrared camera system as a means to verify the detected anomaly. With this approach, we could reduce the power consumption of the device which would make for a more effective system overall.

2.2.3 Dedrone RF-300

The Dedrone RF-300 is a passive network-attached radio frequency (RF) sensor designed to detect, classify, and locate drones and their remote controls. It is capable of detecting consumer, commercial, and hobbyist drones by analyzing the RF signals they emit. While primarily focused on drone detection, its capabilities can extend to other aircraft emitting RF signals.

Some key features of this system are the ability for the RF-300 to identify communication between drones and their controllers including both Wi-Fi and non-Wi-Fi drones. It classifies drones by their manufacturer, model, and unique ID. Through internal digital signal processing, the RF-300 is also able to determine the direction of signals from drones and pilots. With multiple RF-300 units, it can triangulate and pinpoint the exact location of both the drone and its

operator. Each RF-300 sensor covers a 120-degree area, requiring 3 units to achieve a full 360-degree coverage zone. The typical detection range is up to 1000 km, which can be extended to 1500 km under ideal conditions. The system is also built for extreme environments and can operate between -20 degrees to 55 degrees Celsius. Finally, to power the device, it must be connected via ethernet connection.

In addition to the key features listed, the RF-300 integrates with Drones DroneTracker software, a platform that aggregates and analyzes data from multiple sensors to provide comprehensive airspace security. The integration of this software enables real-time monitoring and alerts and when paired with a camera, offers advanced video analytics to enhance detection and tracking capabilities. The software can catalog all alerts, flight paths, and video footage of drone threats, which can be easily exported. Moreover, the software incorporates a classification engine that recognizes drones of all types. The system makes use of a sophisticated machine learning system trained on millions of data points.

This product, while much more advanced, has many similar components and restrictions as our project. One key thing to notice is that multiple units of the RF-300 are required to have a complete view of the airspace in question. Not only that, multiple units are required in order to determine crucial information like speed and direction of travel. One of the main goals of our project is to eventually be able to detect those key parameters so we have to keep in mind that we'll need multiple units to be able to do that. We also do not have the software, nor the capabilities to re-produce a similar piece of software so our project mainly focuses on detecting aircraft rather than classifying them. Eventually, a database could be implemented with software that could reference previous images and detection signals.

2.2.4 Summary

Overall, there is a lot of insight to be gained from the systems that exist already today. Given that RF technology has been around for a while, we can see many different interactions of RF detection implemented to many detection systems already due to its stealth capabilities and relative simplicity. Most of these systems also require multiple units to determine localization information. Our project will likely also need to incorporate that if we hope to get accurate location readings down the line. The innovation for our project will definitely be to do some of what these devices do, which cost a lot of money, for much cheaper.

In most cases, we see that RF detection, if enhanced with optics, usually involves infrared imaging. While IR allows for a more detailed view, it is still susceptible to environmental factors such as rain. For this reason, we chose to use visible light rather than IR. This will prove to be cheaper and will allow for better imaging during the day and can be optimized for different weather conditions in the future. Most of these systems also feature some sort of pan-tilt motor setup to move the camera in the direction of the aircraft. This proves to be a difficult addition and makes things much more complex, especially since our group does not feature a mechanical engineer. Our solution is not going to look like many of the aforementioned products but will instead be focused on competing with the RF device to show the pros and cons of each system. Using super resolution techniques, we will attempt to capture an image of an anomaly, in this case a plane or drone, and enhance it so that it may run through anomaly detection software to

help alert the user of any potential threats in the area. This will keep things much simpler and will still allow for the freedom to include the optical components as an enhancement of sorts should the customer be interested.

2.3 Goals

2.3.1 RF Device & Camera Array Basic Goals

- Build a unit of similar ability to the ones discussed at a very low cost
- The successful detection of a metal sphere using passive detection. The reason for a metal sphere is due to its ease of detection by radar.
- Measure antenna radiation pattern and classical range measurement.
- Arrange 3 low cost cameras in an array configuration.
- Capture an image at a distance of 1, 10, and 20 meters and effectively enhance the resolution of the image using image processing techniques.
- Design a simple and stable configuration for the cameras.
- Find the simplest way to connect the cameras together to the main computer so that image processing may be done.
- Use code to do image processing and to create a super resolution image.

2.3.2 RF Device & Camera Array Advanced Goals

- The system should be able to detect an aircraft in flight. This will prove that our passive detection system is working as intended.
- The system should be self-sustaining. If the system were to be deployed in the field, it should be able to provide its own power. This may be done via solar, wind, or other types of power generation depending on the deployment location.
- The time it takes to deliver the information from the system to the user should be relatively quick. This will ensure that our system is actually viable and worth implementing.
- The device should have a housing that allows it to operate in adverse conditions. Primarily heat and humidity.
- Design lens systems such that a very wide view, wide view, and telephoto lens system are able to be applied to the camera array.
- Enhance our 3 cameras with lenses that will help the process of super resolution effectively at a distance.
- Capture multiple images of a drone or other object through our camera array and use image processing techniques to create a super resolution image that may be run through a detection program/software.
- Design a way to place the lenses in front of the cameras and ensure proper optical alignment.

2.3.3 RF Device & Camera Array Stretch Goals

- The system should be easily implemented and easily manufacturable. The main goal of this project is to provide a low-cost solution which means ease of use and construction is important to consider.
- The system should be connected with other systems to form an array. This will allow for accurate localization and will help the user determine the speed, direction, and location of any aircraft within the detection distance.
- The system should be completely autonomous. This will allow the user to “set it and forget it” and will alert them of possible incursions within their airspace.
- The system should incorporate a false alarm feature.
- Create or utilize a database that can characterize aircrafts and identify the type of plane or drone that is entering the range of the antenna. The system should have the ability to easily install and uninstall enhancements depending on the user's needs. This will be vitally important when it comes to selling the unit. Depending on the user's needs or surrounding environment, they may or may not want certain add-ons or enhancements.
- Design multiple, easy to install, lens systems that can be used depending on the desired image capture distance.
- Design a sturdy, stable, and environmentally protected housing for the entire device.
- Design a connecting unit that allows the unit to be attached to a vehicle, such as an airplane, for mobile image capture.
- Design a housing for the lens system that allows for easy swapping in case the object is at a different distance.
- Incorporate a phase correction plate to account for any aberrations due to the environment.
- Upgrade optical components to a higher quality to further improve the performance of the assembled system
- Combine the RF device and camera array into a synergistic system for optimal performance in aircraft detection.

2.3.4 Software Basic Goals

- Build a lightweight and reliable software system that connects the spectrum analyzer to a web interface
- Create a basic user interface that displays aircraft signal data in real time, including location, signal strength, and detection source
- Develop a background service that runs independently to collect and clean signal data from the SDK
- Format and store incoming data in a way that can be quickly used by the front end without slowing the system down

2.3.5 Software Advanced Goals

- Integrate real-time WebSocket support so that aircraft signals are shown instantly on the map as they are detected
- Use signal processing filters to clean up noisy data and reduce false detections
- Develop logging features that keep track of signal patterns, errors, and system health in case of a failure or disconnect

- Build the system with cross-device compatibility so it works on both desktops and mobile phones without major changes

2.3.6 Software Stretch Goals

- Include signal classification features that can help identify different types of aircraft or filter out non-aircraft signals
- Build a simple database to log past detections and allow users to search or review older signal events
- Create a dashboard interface that summarizes total detections, time active, and any alerts or events

2.4 Objectives

2.4.1 RF Device & Camera Array Objectives for Basic Goals

- Research, select, and buy quality and cost effective materials and components to create the device
- Ensure proper calibration of spectrum analyzer
- Test all components once they have arrived to ensure they function properly.
- Measure range of detection by using a second antenna.
- Use a second antenna to find the effective beam pattern of the antenna.
- Research and purchase low cost, low resolution cameras to include in the array and ensure they can easily be integrated with our devices.
- Test multiple array configurations and shapes and observe which configuration and camera spacing results in the best image after image stitching.
- Use wood, cardboard, or foam to put the camera array on so we can ensure that they are aligned properly and stable.
- Connect the cameras to a computer via a USB connection.
- Write and test code using python to do image processing techniques and create a higher resolution image out of the many that were captured.
- Use Zemax to design a lens system that will allow for quality image capture at 100 yards and account for possible aberrations that we may encounter.

2.4.2 RF Device & Camera Array Objectives for Advanced Goals

- Create an algorithm that differentiates noise from aircraft.
- Test the prototype at Dade Collier Airport to see if we can detect passively
- Research means to power the unit autonomously and integrate it (phase 2)
- Test the amount of computer power and latency to process data.
- Increase efficiency of detection algorithms to decrease power draw.
- Purchase more of the low cost, low resolution cameras to arrange into a larger array configuration.
- Research a way to effectively connect all the cameras into a more cohesive system.

- Use the sign outside of the L3 Harris building on campus to test our camera array at a distance of 100 yards and see if we can produce a more clear image at that distance.
- Based on the separations of the cameras in the array, design a housing for multiple lens systems that can be easily attached to the array and are properly aligned.
- Research and work with CREOL professors to understand how a phase correction plate works and how to properly incorporate it into our design to produce the best possible image.

2.4.3 Camera Array Objectives for Stretch Goals

- Develop a robust enclosure to protect the camera system from harsh environments.
- Integrate with an aircraft mounting system and optimize the system for aerial imaging. Also conduct airborne tests.
- Research and invest in more cameras with better frame rates, resolutions, and the ability to sync capture.
- Create a larger array that has the capability to incorporate more cameras and lens systems to increase the potential of the super resolution system.
- Research and invest more time into super resolution algorithms including algorithms that make use of AI and deep learning.
- Create a larger database of aircraft and drones that may be called upon when an anomaly appears in the area. This can be used to compare what was captured and detected with the camera array with previous detections or, if enhanced with AI, what the deep learning was trained on.

2.4.4 Software Objectives for Basic Goals

- Use Node.js and Express to set up the server and connect it to the front end
- Design a responsive and clean UI using HTML, CSS, and JavaScript with a focus on readability
- Write code that pulls signal data from the SDK using a background service and sends it to the web interface
- Test the real-time signal display using simulated or replayed spectrum analyzer data
- Keep the code organized with clear comments and folder structures to make future updates easier

2.4.5 Software Objectives for Advanced Goals

- Connect the software system to a live spectrum analyzer device using its SDK
- Test signal filtering methods and select the best one for reducing false positives
- Set up WebSockets for constant updates between the backend and the front end
- Collect user feedback on the UI and make changes to improve clarity and performance

2.5 Features & Functionalities

2.5.1 RF Device

The unit is designed for those who purchase it to be able to detect aircraft via the passive detection of waveforms rather than via an active radar. They want it to be able to detect aircraft in radio and radar “dead zones” which have no active radars to detect aircraft, as well as no nearby infrastructure that would be able to detect aircraft in other ways. A common example of this would be an airplane that illegally passes through borders by flying into dead zones that don’t have radars to detect it, or by flying under the radars that would normally be able to actively detect it. This unit should be able to passively detect any such aircraft and with the use of multiple, be able to make an array of them that can be used to triangulate the position of any such aircraft based on the signal strength picked up by each of the devices in a way that can be shown on a phone or computer via starlink. Furthermore it has been requested that the data be livestreamed so that people can take appropriate action based on the data found.

2.5.2 Camera Array

The camera array device allows the user to capture an image at a desired distance and be able to discern higher amounts of detail compared to that of a lower resolution image, that is typically only able to be done by very expensive cameras. Our device, since it will be made up of cheap cameras and affordable lens systems, will allow for clarity at a cheaper cost. The design is also meant to be scalable which will allow for even more precision and clarity depending on the user's needs. By its nature, the array will be able to capture a wide field of view while maintaining its high pixel density through image fusion. With the configuration, blind spots are also eliminated which also allows for a much clearer image. With a larger amount of cameras in the array, it may also be possible to estimate depth which is vital when it comes to surveillance scenarios. Additionally, the device uses off the shelf components which verifies that ITAR restrictions are met. Finally, due to the systems design, the device has a natural fault tolerance meaning that if a camera happens to stop working, the remaining cameras could still produce the desired outcome with a lesser quality.

2.6 Engineering Specifications

Below are some specifications that were derived with the consideration of our project objectives, constraints and limitations of other similar products and projects, and quality versus cost analysis. The tables are also split into categories, namely those for the RF device and those for the camera array device.

Table 2.1 Prototype Specifications and Features

Parameter	Specification
Signal receive and detection time	**1 min or less**
Wide frequency detection range	300MHz to 3GHz
Detection Range	**.25 mile to 4 Miles**
Detection Accuracy	**>65%**
Size	<1m^2
Cost	<$6500

Table 2.2 Unit Components Specifications

Major Specification	Component	Parameter	Specification
Electrical Engineering	Log Periodic Antenna	Antenna radiation pattern	180 degrees
	Battery	Power	>600W
	Spectrum Analyzer	Dynamic range	-158dbm to 10dbm
	Android Phone	Power consumption	<10W
	Portable solar panel	Power	>60W
Photonic Science & Engineering	OV9281 Camera	a) Frame Rate b) Resolution c) Pixel Size d) Color e) Shutter Type	a) 100 fps b) 1MP (1280x800) c) 3um x 3um d) Monochromatic e) Global
	Telephoto Plano-Convex Lens	a) Focal Length b) Material c) Lens Diameter	a) 250 mm b) N-BK7 uncoated c) 1 inch
	Wide Plano-Convex Lens	a) Focal Length b) Material c) Lens Diameter	a) 175 mm b) N-BK7 uncoated c) 1 inch
	Ultra-Wide	a) Focal Length	a) 125 mm

	Plano-Convex Lens	b) Material c) Lens Diameter	b) N-BK7 uncoated c) 1 inch

Table 2.3 Camera Array Engineering Specifications

Engineering Requirement	Parameter	Specification
Camera(s)	Resolution Pixel Count Frame Rate Color Shutter Type	1MP 1280x800 100 fps Monochromatic Global Shutter
Camera Array	**Effective FOV** **Image Overlap**	**0.5-5 degrees** **>50%**
Super Resolution Algorithm	**Final Image Quality**	**>2x Original (at least 2MP)**
Lens System	**System Length** **Object Size at 60 ft Away**	**<125 mm** **<175 mm** **<250 mm** **~1 ft**
Entire Unit	Weight Operating Temperature	Less than 2 kg 10-30 degrees Celsius

2.7 House of Quality

2.7.1 RF Device

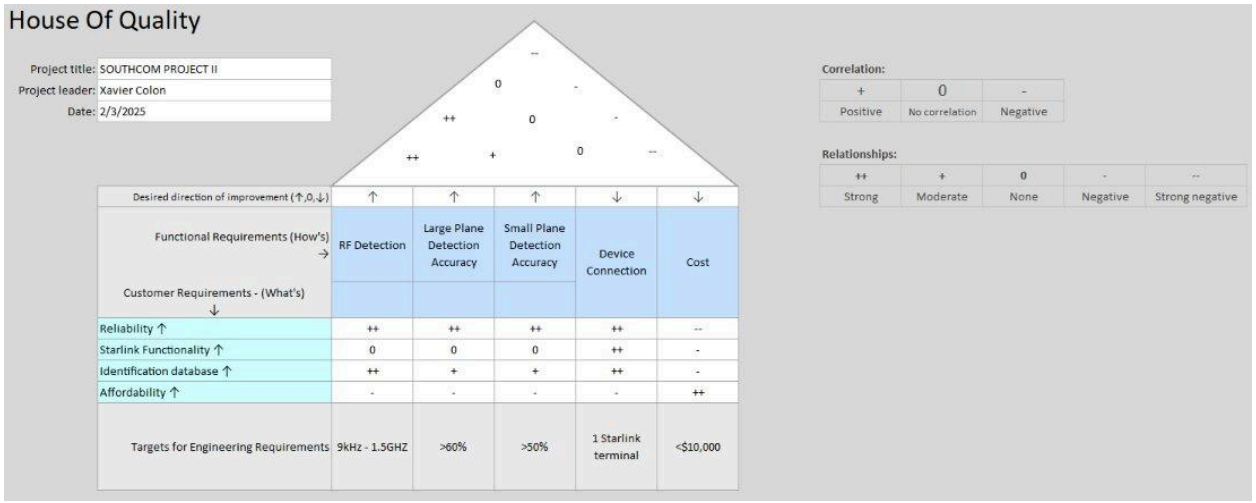


Figure 2.1 *Passive Radar House of Quality*

2.7.2 Camera Array

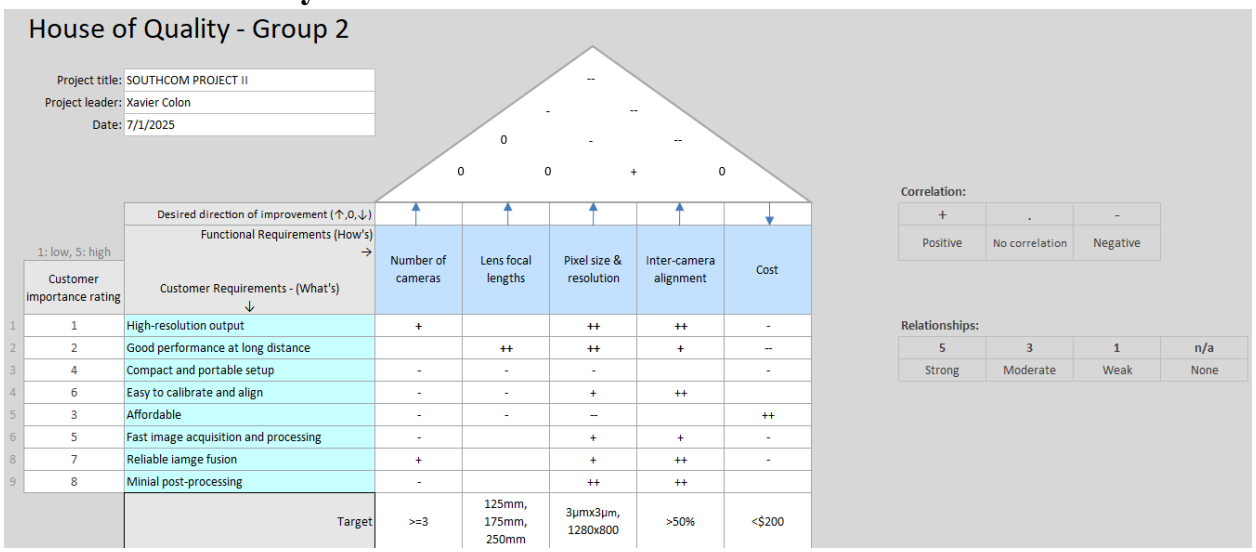


Figure 2.2 *Camera Array House of Quality*

2.8 Hardware Block Diagram

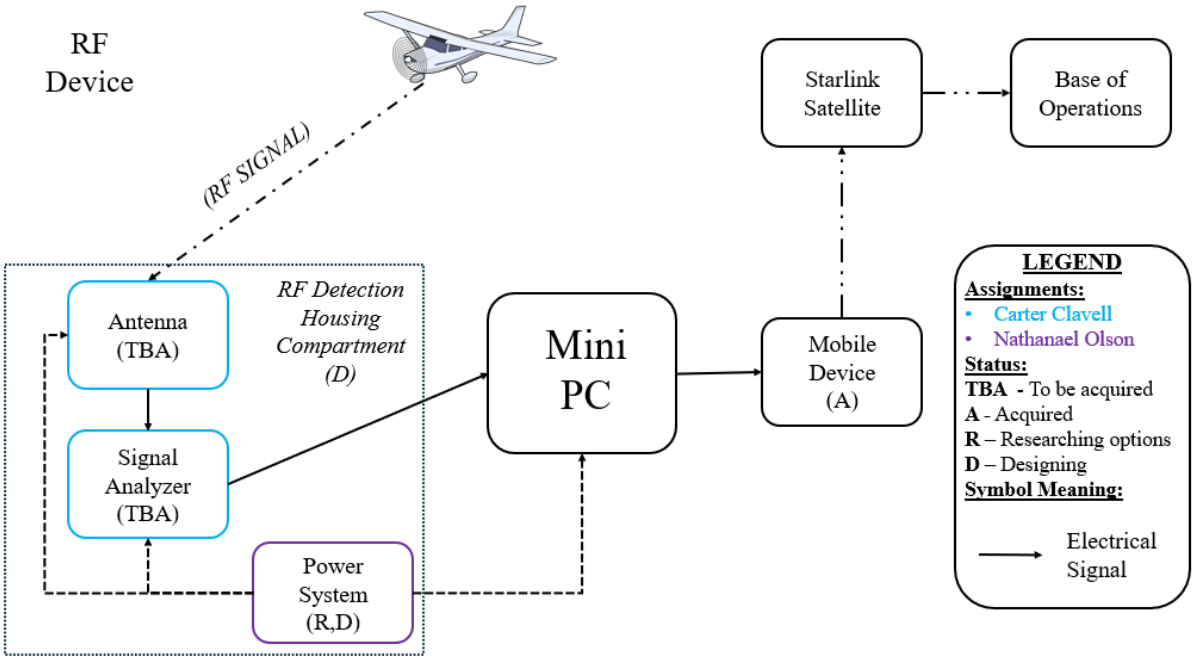


Figure 2.3 *Hardware Block Diagram*

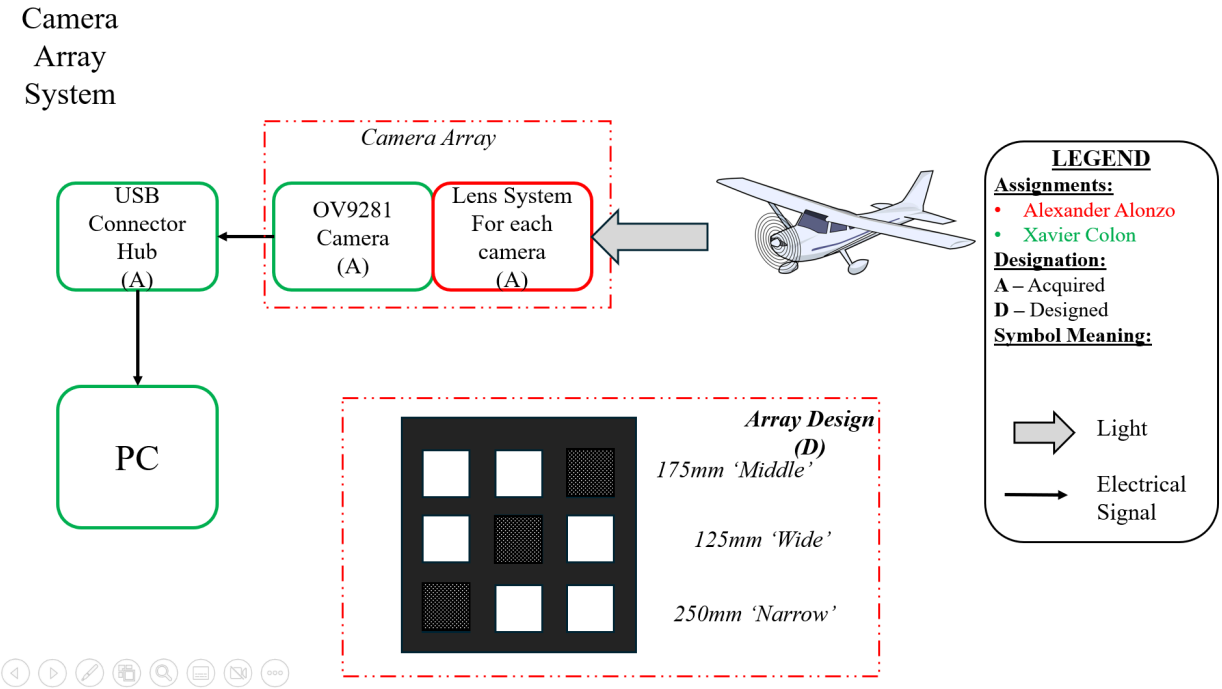


Figure 2.4 *Optical system hardware diagram*

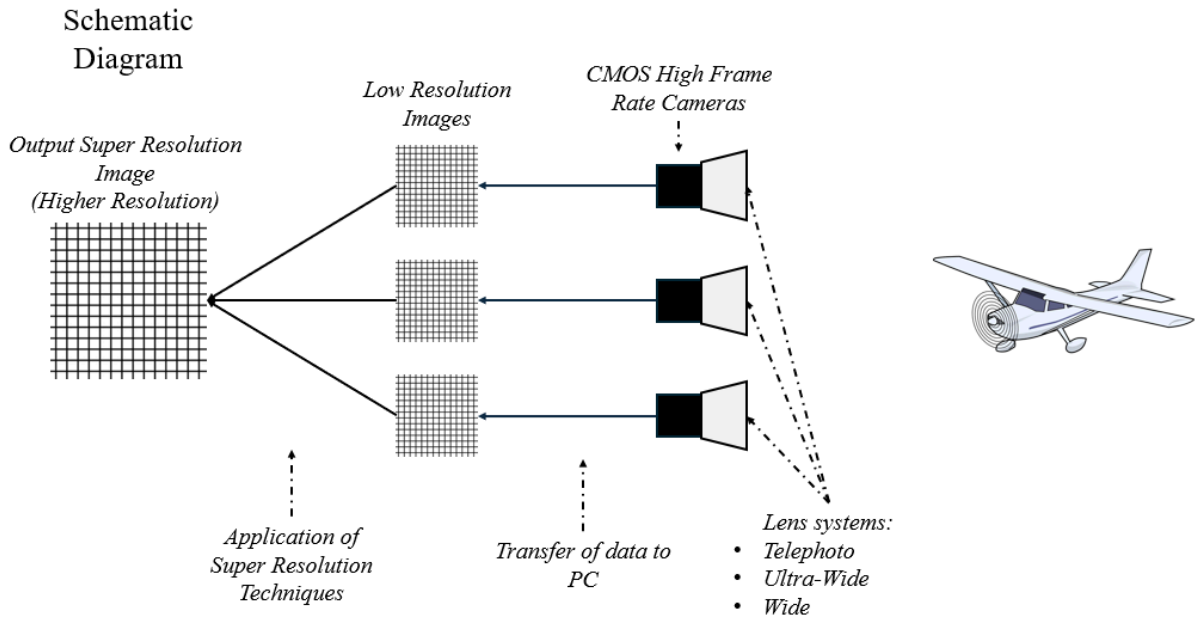


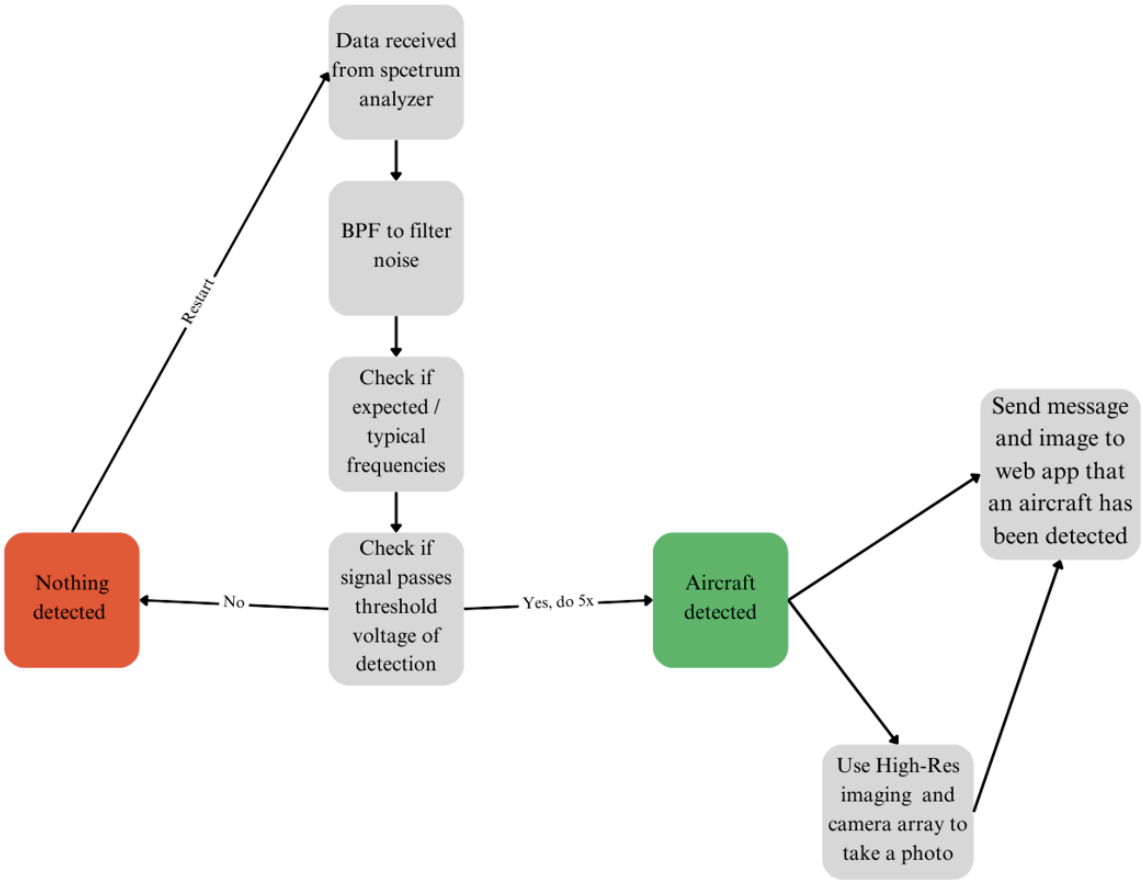
Figure 2.5 *Schematic diagram of camera array system*

In the figure above, the process of producing the super resolution image is shown. First the cameras, depicted by the black boxes, with their respective lens systems, capture the desired scene individually. The chosen lens systems were adopted from the iphone and other mobile phone cameras that feature 3 cameras with similar lens systems. In this case, it would most likely be a drone or aircraft that is in flight, not close to the cameras. This would be the ideal case to minimize motion blur which will help when using detection software later on. Once the images are captured, they are transferred to a pc so the image processing and fusion train can begin. In this phase, the low resolution images go through our specially designed super resolution code and then produce a single high resolution image at the output which is the super resolution image. Since this image has more detail than the low resolution images, when it is run through a detection software, it will ideally reveal the desired target object more effectively since the edges of the target are more well defined.

The hardware diagrams above illustrate some of the basic components that our project will implement and whose responsibilities they are. Further additions, either by our sponsor's request or by other groups on this project that are not in our senior design section may need to be accounted for. In the first diagram, a plane is depicted with its RF signal being emitted. That signal is captured by our antenna and is interpreted through our spectrum analyzer device. After that, the information from the spectrum analyzer is transferred to the mini pc where the computer science team takes over to transfer the data to the user. While the final idea is to implement this device in a low RF noise environment, for our phase of this project we will focus on detecting a signal from the target to prove our idea works. We currently do not know if other divisions, like the computer science team, will need different hardware parts or housing units yet. Moreover, the PCB requirement from senior design was voided for this project which is why it is not illustrated in the diagram.

The second diagram serves as an optical system hardware diagram and shows how the optical systems interact. A closer look at the camera array is provided and details the layout of the 3 cameras that will be used and their respective lens systems: telephoto, wide, and ultra-wide. With the 3 cameras, images will be captured with different levels of magnification and field of view to obtain the necessary details to construct a higher resolution image. From there, image processing and super resolution code will be used to attempt to create a super resolution image from the images captured from each individual camera. Once the super resolution image is produced, it will be run through a detection software and will act as a competitor to the RF approach to demonstrate that the problem can be solved both with electronics and optics. While our system is also rather basic and small scale, this project could act as a proof of concept to encourage more research and investment into super resolution techniques.

2.9 Software Block Diagram



a

Figure 2.6 *Software Block Diagram*

Shown above is a high level representation of how the software will determine the detection of aircraft. We have at first the data that is outputted from the spectrum analyzer. Which will then

be sent through a filter and an algorithm to check if it is within the typical frequencies that we may see. Then we will check if the received signal is great or less than our threshold voltage of detection. If the signal voltage is higher that means there is likely something there bouncing off the radio wave into our antenna. This is a good indicator that there is likely an aircraft. If it is too low then it will continue to repeat the process. However, if an aircraft is detected it will send a packet of data and a high res image caught by the camera system to our web app. From the web app the user will be able to check and see when and where the aircraft has been detected.

Chapter 3: Research

3.1 Camera Array

Research Goals

For the camera array device to be successful, a few important things need to be taken into consideration. Since we are dealing with image sensing and capturing images at a distance, the two main topics to discuss revolve around which sensors to use and which lenses to use. The goal for this section is to investigate and research the different types of technologies in the sensing and lens domain to then develop a justifiable choice for our project. Additionally, the programming language we choose to use is just as important. Different options will be investigated and evaluated to see which will best serve our needs for the project.

3.1.1 Overview of Camera Technologies (CMOS vs. CCD)

Selecting the appropriate image sensor technology is a critical factor in the performance of any imaging system. The type of technology that is chosen has a direct influence over the resolution, sensitivity, power consumption, and overall cost. In the context of this project, which involves spatial ray imaging where multiple low-cost cameras are utilized to achieve a high resolution image, understanding the trade-offs between CMOS (Complementary Metal-Oxide Semiconductor) and CCD (Charge-Coupled Device) sensors is vital. This section explores the fundamentals behind each of the technologies, as well as other sensor technology characteristics, and evaluates their advantages and limitations which will guide us to the best option for this project.

CMOS image sensors operate by converting light, or photons, into an electrical signal through an array of photodiodes. In this case, each photodiode represents one pixel in the sensor. When light enters the sensor, it makes contact with the photodiode and generates an electron-hole pair via the photoelectric effect. The number of electrons generated is directly proportional to the intensity of the incident light, resulting in an accumulated charge within each pixel. In CMOS sensors, each pixel typically has its own amplifier which allows for the immediate conversion of charge into voltage. With this type of design, the sensor is able to perform parallel readout, which is reading the voltages from each pixel at the same time, which then enables faster image capture and higher frame rates.

CCD cameras, on the other hand, use an array of photosensitive pixels to capture light and convert it to electrical signal. A CCD sensor consists of a grid of photodiodes which, like CMOS

sensors, are made of silicon. Similarly, these sensors also function by generating electron-hole pairs through the photoelectric effect. This, again, means that the number of accumulated electrons is proportional to the intensity of the incident light during the exposure time. The main difference in CCD sensors as compared to CMOS sensors is how the charge transfer mechanism works.

Unlike CMOS sensors, where each pixel has its own amplifier, CCD's store charge and then transfer it sequentially to a central readout amplifier. This process starts with charge accumulation, where each pixel collects charge based on the light it receives. For the charge transfer, after the exposure is complete, the charges are transferred row by row (or column by column depending on the specific device) through a series of potential wells. Here, a potential well refers to an electric field-created region within a pixel where photo-generated electrons are collected and stored. To transfer the electrons from the potential wells to the amplifier, an external clocking signal is used which basically adjusts the voltages to allow for the charges to move. Finally, at the readout and amplification stage, the charge from each pixel is moved to a readout register where it is converted into a voltage by a single on-chip amplifier. The resulting signal is then digitized and used to produce a digital image. Because the charge is transferred across multiple pixels, rather than at each pixel like in the CMOS, CCD's require precise timing mechanisms to ensure accurate charge movement. This type of sensor design results in lower fixed-pattern noise compared to CMOS sensors but also increases power consumption and readout time. (B. S. Carlson)

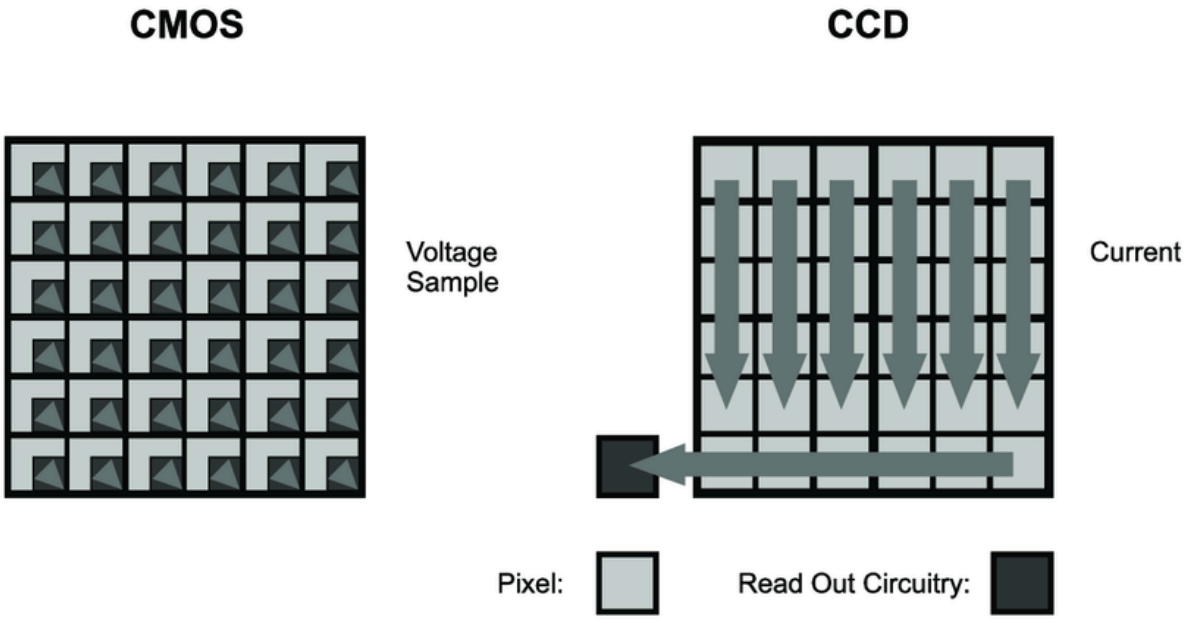


Figure 3.1 *CMOS and CCD sensor charge readout mechanism (ResearchGate)*

In terms of cost, CMOS sensors are significantly cheaper to manufacture than CCD's. This is because they are produced using standard silicon fabrication processes, similar to those used for microprocessors. This allows for mass production at a much smaller cost which is why CMOS

sensors are the dominant choice in consumer electronics. CCD sensors on the other hand require specialized fabrication methods. The main reason for the increased precision is because of the precise charge transfer structures and high voltage clocking circuits as previously mentioned. This process is therefore more expensive which leads to a higher cost per unit as compared to CMOS sensors.

Additionally, CMOS sensors have built-in processing capabilities, which include analog to digital (ADC) conversion, noise reduction, and signal amplification to name a few. The ability to do this with the sensor alone means that a simpler circuit design can be used which therefore lowers the system cost. CCD sensors require separate ADC's, external timing circuitry, and higher voltage drivers which increases both the overall system cost and complexity. This can also give us insight into some other reasons that costs are higher for CCD's. Since their circuitry is more complex and requires more components, the power consumption is higher which then leads to the incorporation of cooling requirements, another expense.

Another difference between these sensor technologies is in their shutter method. There are two types of shutter methods, rolling shutter and global shutter. The rolling shutter method, which is more widely used in CMOS technology, captures an image by scanning rows of pixels sequentially. This means different parts of the image are exposed to slightly different times. As a result, this method can cause motion blur or banding effects. This is commonly seen in consumer technology such as in smartphones, DSLRs, and mirrorless cameras, all of which usually equip a CMOS sensor. The rolling shutter method is mainly used because of its lower power consumption, simpler circuit design, and overall cost effectiveness. However, its biggest flaw comes in the form of distortion in scenes with fast moving objects or quick camera motion.

The global shutter method, which is used in CCD's and some high end CMOS technology, captures the entire image at the exact same time. This method eliminates the distortions experienced by the rolling shutter method. This, therefore, makes it much more viable for applications that require precise motion capture, machine vision, and scientific imaging. While global shutters offer a more superior accuracy when it comes to motion, they typically require more complex circuitry and higher power consumption. Global shutter is mostly found in professional and specialized cameras like those you may see at sporting events or in the film industry. Since our project will mainly focus on stationary objects, the rolling shutter method that is mainly found on CMOS technology would work best.

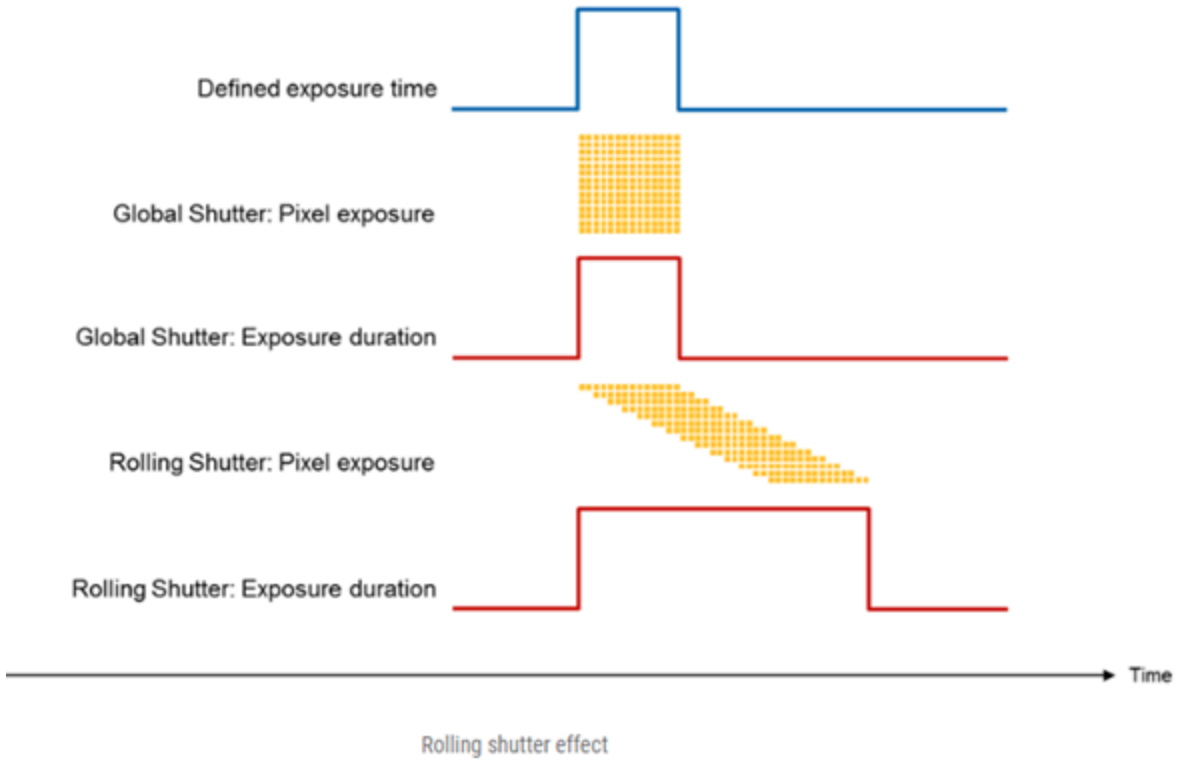


Figure 3.2 *Exposure time and duration for each shutter type (E-con Systems)*

The last main comparison in sensor technology we need to consider is mirrorless versus DSLR (Digital Single-Lens Reflex) cameras. DSLRs incorporate a mirror and prism system to reflect light from the lens of a camera into a photographer's eye. This method provides a real time view of the scene that the photographer is trying to capture without relying on any electronics. In contrast, mirrorless systems rely, typically, on liquid crystal display screens to display a digitally processed image. This method, while it does not provide a real time view of the scene, allows the user to see what the image would look like once the image is captured. Furthermore, because this method does not rely on a mirror, the size and weight of the system is much less compared to that of a DSLR system. Since our project will use a computer to see the scene that sensors are capturing, a mirrorless system will work best.

Table 3.1 Comparison of CMOS and CCD technology

Feature	**CMOS Camera/Sensor**	**CCD Camera/Sensor**
Manufacturing Cost	Lower - Uses standard silicon fabrication, allowing for mass production with lower cost.	Higher - Requires specialized fabrication, increasing cost of production.
System Cost	Lower - On-chip integration of analog to digital converter,	Higher - Requires separate analog to digital converters,

	amplifiers, and signal processing reduces need for additional external components.	timing circuits, and high voltage drivers, increasing cost and complexity.
Power Consumption	Lower - Each pixel has its own amplifier, requiring less power.	Higher - Charge must be transferred across the sensor, consuming more power and generating heat.
Image Quality	Good - Modern CMOS sensors rival CCDs in many applications.	Great - Lower noise, preferred for scientific imaging.
Motion Capture	Rolling shutter in some CMOS sensors can cause distortion. High-end CMOS sensors use global shutter.	Uses global shutter, making it ideal for capturing fast moving objects.
Applications	Smartphones, webcams, security cameras, etc.	Scientific imaging, astronomy, medical imaging, etc.

Overall, each type of sensor has its strengths and weaknesses. CMOS sensors are the most effective for capturing images of slow moving objects because of how frequently they are paired with rolling shutters. They are also usually much cheaper compared to CCD sensors because their circuitry is much simpler and requires less power to operate. CCD sensor strength lies in its ability to capture a better image when an object is moving. While they are more expensive than CMOS sensors, they typically also feature better resolutions, frame rates, and motion capture which allows them to do what CMOS sensors can and more. For our project, if we decide to go with a CMOS sensor, we would be limiting our ability to capture images of moving objects. However, if we were to use CCD sensors instead, we could compete with the RF detection device and see which system would be better when it comes to aircraft detection. Since they are more expensive, our array size would definitely have to be reduced compared to if we were using CMOS sensors, but our application range would be much larger.

Based on the table though, CCD sensors clearly outperform CMOS sensors. What is not represented effectively here is higher level CMOS sensors. Some CMOS sensors are able to compete with CCDs to a good degree with certain characteristics such as a high frame rate. For our purposes, and with our budget, a high performance CMOS camera with a higher fps should do the trick. The goal is for us to produce a super resolution image and we would be able to show that effectively with high level CMOS cameras which would keep our cost lower and still fit our needs. If we test this idea with a drone for example, these types of cameras should work very well.

3.1.2 Evaluation for Selection

In order to have an effective demonstration of our project, some evaluation of criteria have to be made. Firstly, we have to consider the resolution of the sensors that we will purchase. In our design, we want to compile an array of lower resolution cameras to then produce an enhanced super resolution image using image processing tools. This would then lead to the belief that we should purchase the lowest resolution sensors possible, but this would be a mistake. Since one of our goals is to capture an image at a distance, we would need sensors that have at least a decent resolution to make this work the way we intend. The sensors that we choose should be greater than 1 megapixel, with higher values providing better clarity but coming at a greater cost. When they are organized in an array, this will produce the necessary quality to be able to discern certain details while simultaneously maintaining the affordability.

The next parameter to consider is the frame rate. Since our goal is to capture an image at 100 meters, choosing an appropriate frame rate, commonly known as frames per second (FPS), depends on different factors. Among these factors are motion blur, exposure time, lighting conditions, and the purpose of the system. Since our system intends to capture a stationary object, we'll have to evaluate which frame rate would serve best for that situation. Typically, when imaging stationary or slow-moving objects, a low frame rate is usually sufficient. This is because there is minimal movement which means aberrations and distortions are also minimized. Typically, for fast moving objects like aircraft, vehicles, or athletes in game, a higher frame rate is necessary to minimize blur and improve tracking. For the latter, it is customary to have at least 30 or more frames per second, with the large values making the risk of motion blur even less. For our system, 30 frames per second is not necessary, however, a frame rate that is too low may prove detrimental. For this reason, we will evaluate options between 10-30 frames per second. This will allow for some minimal movement, which is expected especially if the user is taking the picture while holding the system or if there is slight movement in the object's area.

The consideration is crucial to the success of our project and it is the synchronization of capture. In order for the camera array to really show its strengths, synchronized capture is important. If the images from the cameras are captured at the same time, we would then be able to fuse the images together to get finer details. If the images are not captured at the same time, we run the risk of getting extremely distorted or unclear images. The latter technique is not impossible to do and is actually sometimes used by artists who capture two completely different scenes and merge them together. For our purposes though, we are trying to get more detail from one scene. In a surveillance setting, you cannot guarantee that the object you are trying to capture will not move at least a little bit, so being able to synchronize the capture is incredibly important.

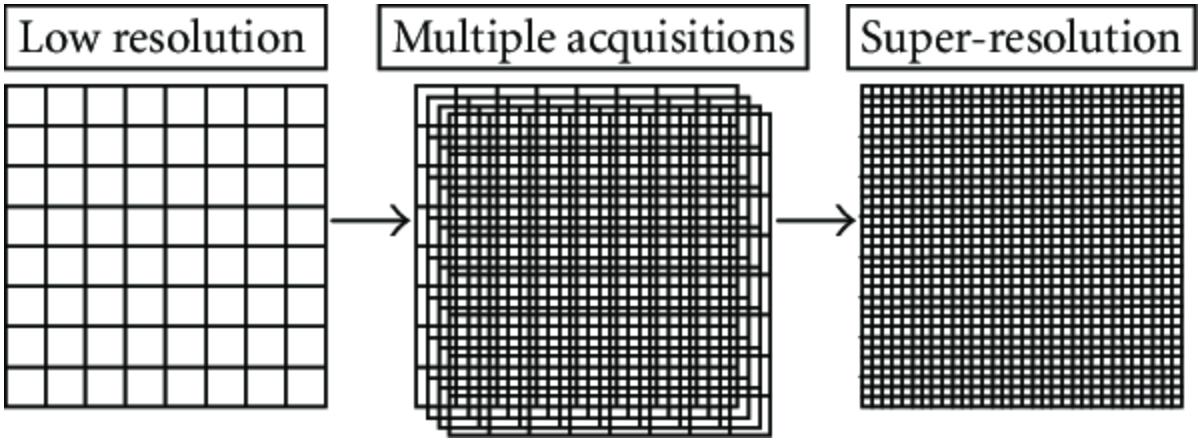


Figure 3.3 How Super-resolution functions (ResearchGate)

The image above demonstrates how the process of creating a super resolution image works. It can be easily visualized as multiple captures being superimposed on each other to increase the pixel density and thereby the resolution of the final image. If the multiple captures do not capture the same scene at the same time, there will be distortion and blur that will be hard to correct. The best example of this is when a moving object is attempting to be captured up close. When a fast moving object is captured up close without the proper shutter speed or frames per second to account for it, a motion distorted image is produced. While at a glance, it is still clear that there is an airplane in the image below, this type of motion blur could be detrimental for edge detection algorithms. Since the main goal of our super resolution project is to produce a high resolution image to be able to better detect an object in the scene, making sure clarity is as high as possible is really important. With blur, the edges of the object we are trying to detect could get lost which, in a real world scenario, would be detrimental to the user.

Figure 3.4 *Motion blur example with a plane* (Eyeoftheflyer)

The image above shows the importance of synchronized image capture. While in the depicted instance, it is done rather up close, the same ideas and considerations need to be taken into account for our design. To address this factor, we will need to look into cameras and sensors that have the proper frame rates, shutter types, and if possible, synchronized capture capabilities. If they cannot synchronize capture, they will need to be compatible with our computers so that we can write code that will direct them to capture the scene simultaneously. The code may need to involve some sort of computer vision as well to ensure that a viable image is able to be produced. This would add another safety net to our design which will allow for an easier user experience.

In terms of power consumption, we are aiming to keep that to a minimum. For our device to be effective and viable, power consumption cannot be too high. If there is a large power requirement, it may require the user to carry around a battery or power bank which reduces the appeal of our idea. Since our demonstration will likely involve the computer vision approach mentioned earlier, a power source that is easily accessible, such as a USB connection, may be the direction we want to take. This will allow for easy installation and allows us to both power the cameras and use them as intended with the code that we will write.

Another characteristic to consider is the bandwidth. During our PSE midterm demo, we encountered an issue we did not expect. Our setup, which consisted of 3 cameras, was connected via a USB hub to our computer. To our surprise, only 2 of the cameras were functioning, and

after looking into what caused the issue to occur, we learned about bandwidth, specifically video bandwidth. Our approach used a code that, when run, opened separate windows to show you what each camera was seeing. With just 2 cameras operating, it was clear that the cameras were operating sub-par to their regular individual performance. This was a result of the bandwidth limitations of our USB hub. To address this, we will need to look into effective means of reducing bandwidth and some devices that handle video well, especially if there are more than 3 cameras running at a time.

The last evaluation is the cost. The main idea of this SOUTHCOM sponsored project is to provide a more affordable solution to the problem while still maintaining the appeal of what makes the more expensive options attractive. Our goals for the camera array can be easily achieved by some products on the market, like high resolution cameras, most of which come with large lenses, but they have a steep cost. For this reason, our research will focus on cameras and sensors that can achieve all the previously discussed characteristics that we need but are also affordable. This means, for each camera, the goal is to find an option that is $50 or less. If we are able to find options in this range, our device's appeal stays high and when completed, will still be more affordable than professional grade setups.

3.1.3 Product Research (Specific Models)

The first product that we considered was the **Arducam OV9281** camera. The Arducam OV9821 is a 1 megapixel camera which uses a global shutter camera and is mainly designed for high-speed imaging applications where motion clarity is critical. The reason this camera is being considered is because of the goals of the RF side of the project. Since the RF side of the project is attempting to detect an aircraft, or other object, that emits an RF signal, the incorporation of this camera can rival and compete with the RF device to attempt to locate that same aircraft or object. Therefore, a global shutter may be necessary to incorporate since rolling shutter cameras introduce motion blur and make it difficult to discern detail with moving objects. Additionally, the use of a global shutter ensures that all pixels capture the image simultaneously as described earlier which actually benefits our project in terms of synchronized capture. Though it would require the use of a USB hub that can handle multiple streams at once, it would serve as a good option for motion and synchronized capture.

While the resolution of this camera is lower, listed at 1280 x 800 pixels, its high frame rate, which can reach up to 120 frames per second, would help capture finer details of a moving object. This camera also features a monochrome sensor, meaning it only captures the intensity of the light, which translates to a grayscale image. While this may seem like a disadvantage, a monochrome sensor would help capture more detail than a color sensor. Since the sensor does not have to process color information, the grayscale image would possess more detail in the picture. A monochrome sensor would also be more advantageous in lower light conditions since it is only capturing light intensity. This camera connects via a USB 2.0 which makes it easy to integrate with Python and OpenCV, a computer vision library. However, since its resolution is on the lower end, it may not be as useful to us if we are trying to get a very high resolution image at the end. Finally, the size and price of this camera are very attractive. Since the camera is only 28 x 38 mm in size, it makes it ideal for an array arrangement approach. This specific one is listed

on Amazon for around $50 which would make the final array more affordable than other options (Arducam).

The next product we looked at was the **IMX219 8 megapixel** camera module. This camera module is a compact and low cost imaging unit that is commonly used in embedded vision projects. It is typically paired with a Raspberry Pi and NVIDIA platform. It features the Sony IMX219 sensor which is an 8 megapixel, or 3280 x 2464 pixels and uses a rolling shutter CMOS sensor. This provides a good balance between resolution and power efficiency since CMOS sensors use less power than CCD sensors. The module is widely available and is easy to integrate which makes it an appealing option for our array setup. Its frame rate can go up to 30 frames per second which is suitable for capturing an image of a moving object, as long as it is not moving too fast.

The cost effectiveness of this option, being that it is one of the more affordable options that offers a higher resolution sensor, makes it very feasible for a multi camera array on a budget. Since it is not powered through a USB connection, we would have to use a battery to power the camera, though its power consumption is not very high which won't make that too much of a burden. (Seedstudio)

The next consideration was the **See3CAM_CU135**, a 13 megapixel camera. This camera was designed for computer vision applications, which is part of our approach with the camera array. The camera provides sharp images with an excellent dynamic range, meaning there is a lot more clarity in the image. It allows the user to manually control the exposure levels, gain adjustments, and white balance giving fine control over image conditions. It uses a USB 3.0 interface which allows for faster data transfer than its USB 2.0 counterparts, and makes it more ideal for multi-camera setups. It also features interchangeable lenses which will help us when integrating the lens system that we will design. The more glaring disadvantage of this camera is its price. While it is still relatively cheap when compared to the types of cameras on the market, for our applications, it will prove difficult to use. (See3CAM)

Another camera we researched was the **Raspberry Pi Camera Module 3**. This camera is very similar to the previously mentioned OV9281 camera. It features a 11.9 megapixel resolution, and a Sony IMX708 CMOS sensor. It has a frame rate of 100 frames per second when operating at 720p resolution just like the OV9281. However, this camera module uses a rolling shutter as opposed to a global shutter making moving objects much harder to capture. Additionally, the interface for the camera is not usb, instead, it is MIPI CSI. This would mean we would need an additional Raspberry Pi module to be able to use the camera. The main issue with the MIPI CSI is the number of allowed connections, which in our case, would not be helpful as our project consists of multiple cameras in use at the same time. Bandwidth limitations would also be an issue since the CSI bus shares bandwidth across all the connected cameras, whereas USB cameras can be spread across multiple ports for better performance. It is slightly cheaper than the OB9281 camera, but its lack of flexibility does not make it the best option for our project. (RaspberryPi)

Table 3.2 Comparison of Specific Camera Technology

Feature	Arducam OV9281	IMX219 8MP	See3CAM_ CU135	Rasp. Pi CM3
Resolution	1280x720	3280x2464	4208x3120	4608x2592
Shutter Type	Global	Rolling	Rolling	Rolling
Max Frame Rate	100 fps (720p)	30 fps (1080p)	60 fps (1080p)	100 fps (720p)
Pixel Size	3 um	1.12 um	1.1 um	1.4 um
Connection Type	USB 2.0	MIPI CSI	USB 3.0	MIPI CSI
Price	~$50	~$35	~$100	~$40
Best for Motion Detection	Yes, global shutter prevents motion blur	No, rolling shutter leads to motion blur	No, rolling shutter leads to motion blur	No, rolling shutter leads to motion blur
Super Resolution Feasibility	Good, multiple cameras with high frame rates	Possible, limited by rolling shutter	Possible, limited by rolling shutter	Possible, limited by rolling shutter

The table above compares the four camera modules that were previously discussed. While they each have their advantages and limitations, we believe the Arducam OV9281 will be the best option for our project. The global shutter, paired with a USB connection, and high frame rate make it the best option for super resolution applications. Additionally, the price makes it an affordable option and would allow us to purchase more modules. Although its resolution is lower than the other camera modules, our project will address that limitation by using multiple cameras at once. This will help increase the overall resolution of the final image since cameras capturing an image of the same object from slightly different perspectives provides more detail to the full picture.

As of this research, we also believe that 3 cameras should provide the necessary amount of information. This is based on how smartphone cameras such as the iphone work and use their cameras. The iphone 16 pro max, for example, utilizes 3 cameras, one with a ultra wide fov, one with a wide fov, and one with a telephoto lens. Using these 3 cameras, they are able to achieve high resolution imaging through image fusion. The use of 3 different fields of views also helps with depth control and focus which will help our super resolution process. Though their cameras are built with much more expensive technology that features much higher resolutions than our proposed solution, the 3 cameras should be enough to showcase what we want (Apple).
In terms of how to improve the system given a more extensive budget and less strict time constraints, a few options are available to improve the performance of the system. Firstly, choosing cameras that feature CMOS sensors, global shutters, and have much higher resolutions

and using more of them in the system would be the best way to improve performance. The more cameras that are available to capture data, the better the super resolution output will be since there is more information available to use. Additionally, the camera, if possible, should feature sync capture capabilities to provide the best possible inputs for the super resolution algorithm. One camera that may be a good option is the Sony IMX455, a camera commonly used in astrophotography and other scientific applications. This camera has a very high dynamic range, excellent low light performance, and has very high resolution for superresolution baseline. There are also variations that use IR imaging methods instead of visible spectrum which may have better applications in dark environments. While this is just one possible option, any camera with higher resolution and overall better performance would serve well especially if the number of cameras in the array was increased beyond our projects 3. The more information, especially unique information meaning with different perspectives or magnifications, that is able to be captured and used the better.

3.2 Lens System

Research Goals

The primary research goal for the lens design of each camera in the array is to optimize the trade-offs between magnification, field of view, and aberration control to ensure the best final image. Each camera will have a unique focal length and aperture size, selected to maximize resolution while maintaining sufficient overlap between adjacent fields of view. Additionally, due to the hot and humid atmospheric conditions of the region, wavefront distortions will degrade the image quality over long distances. To mitigate this, the system will incorporate a phase corrector plate, designed to counteract these phase shifts caused by differing refractive indices in the air. This corrector plate will function similarly to adaptive optics but in a passive manner, compensating for distortions before computational super-resolution algorithms process the final image. The effectiveness of the phase correction will be analyzed to ensure optical clarity and resolution in the constructed image.

In designing a multi-camera system for super-resolution imaging, careful consideration must be given to the selection of optical components and image processing techniques. Each camera in the array plays a unique role in capturing different magnifications, fields of view (FOV), and perspectives, all of which contribute to the final high-resolution composite image. To achieve optimal clarity, a combination of telephoto, wide-angle, zoom, fisheye, and aspheric lenses will be utilized. These lenses allow for diverse data collection, capturing fine details from distant objects while also providing broad contextual information. However, challenges such as parallax, distortion, and atmospheric turbulence—especially in Florida's hot and humid environment—necessitate advanced stitching and correction techniques. These include feature-based stitching, optical flow alignment, multi-resolution pyramid blending, deep-learning-based super-resolution, and wavefront correction. Together, these methodologies will ensure a seamless and high-quality final image, overcoming the limitations of individual camera units.

3.2.1 Lens System Types for Multi-Camera Imaging

A well-designed optical system requires multiple types of lenses to capture both fine details and broader scene information. Each lens type brings unique benefits that contribute to the multi-camera array's ability to construct a super-resolution image.

Telephoto Lenses for High-Resolution Detail

Telephoto lenses, which typically have focal lengths between 100mm and 600mm, are indispensable for capturing distant objects with high precision. These lenses provide a narrow FOV, which enables the system to isolate fine details that would otherwise be lost in wider lenses (Firoozfam, 2004). A key advantage of telephoto lenses is their ability to minimize perspective distortion, ensuring that objects retain their natural proportions even at long distances. However, their small FOV necessitates precise alignment within the array to prevent gaps between images (Brady et al., 2020).

Another challenge with telephoto lenses is their susceptibility to atmospheric distortions. Over long distances, heat waves and humidity variations in the air create wavefront distortions that degrade image sharpness. To mitigate this, phase corrector plates can be integrated into the optical system. These plates function similarly to adaptive optics in astronomical imaging, counteracting phase shifts induced by the atmosphere before the image reaches the sensor (Harfouche et al., 2023). Computational post-processing, such as wavefront reconstruction algorithms, can further correct distortions, enhancing the final image quality.

Wide-Angle Lenses for Contextual and Peripheral Imaging

Wide-angle lenses, with focal lengths ranging from 10mm to 35mm, play a critical role in capturing a broader FOV. While they lack the magnification of telephoto lenses, their ability to cover large portions of a scene ensures that contextual and peripheral information is preserved (Al-Harasis & Sababha, 2019). This is particularly important when multiple telephoto cameras are capturing different portions of a distant object; wide-angle images can provide alignment references to help with image stitching.

One drawback of wide-angle lenses is barrel distortion, where straight lines appear curved, particularly near the edges of the frame. This effect can be corrected using optical elements within the lens itself or through computational de-warping techniques. Additionally, while wide-angle lenses excel in providing scene context, their lower resolution at long distances makes them less suitable for capturing fine details. As a result, they are best used in conjunction with higher-magnification optics within the array.

Zoom Lenses for Adjustable Focal Ranges

Zoom lenses, with adjustable focal lengths such as 24-70mm or 70-200mm, provide versatility by allowing dynamic adjustments based on the imaging scenario (Chu et al., 2020). This adaptability is particularly useful when tracking moving objects or when the scene composition changes. Unlike fixed focal length lenses, zoom lenses enable a single camera to serve multiple purposes, making them valuable in a system where space and weight constraints must be considered.

However, zoom lenses introduce additional optical complexities. As focal length changes, focus shifting and chromatic aberrations can occur, requiring precise calibration (Lin et al., 2015). Additionally, mechanical zoom mechanisms may introduce minor vibrations or misalignments over time. Computational corrections, such as autofocus optimization and digital sharpening, can help maintain image integrity when using zoom optics.

Fisheye Lenses for Panoramic Coverage

Fisheye lenses, which typically provide a FOV of 180° or greater, are useful for capturing hemispherical images that can later be stitched together into a panoramic composite. These lenses are particularly valuable for surveillance and environmental monitoring applications (Zheng, 2010). By positioning fisheye cameras at key points in the array, the system can ensure complete coverage of a scene, reducing the likelihood of missing important visual information.

The primary drawback of fisheye lenses is their extreme distortion, which results in a curved representation of straight lines. Advanced de-warping techniques, such as cylindrical or spherical projection mapping, must be employed to correct these distortions before integrating the images into the final composite (Brady et al., 2020).

Aspheric and Custom Lens Designs for Aberration Control

To minimize optical aberrations such as spherical and chromatic aberrations, aspheric lenses can be incorporated into the system (Yuan et al., 2017). These lenses provide sharper images by reducing the need for additional corrective elements, which can introduce light loss or distortions. Custom lens designs, tailored for specific wavelength ranges or optimized for low-light performance, can further enhance image quality (Fang et al., 2025).

Table 3.3 Comparison of Lens System Types

Lens Type	Focal Length Range	Positive Features	Negative Features
Telephoto Lenses for High-Resolution Detail	100mm-600mm	Minimizes perspective distortion and high precision with long distances	Narrow field of view
Wide-Angle Lenses for Contextual and Peripheral Imaging	10mm-35mm	Wide field of view to give peripheral information	Barrel distortions and low magnification

Zoom Lenses for Adjustable Focal Ranges	24mm-200mm	Best used for moving objects	Focal length changes causing image blur and chromatic aberrations
Fisheye Lenses for Panoramic Coverage	Wide Ranging	Field of view 180 degrees or more	High distortions
Aspheric and Custom Lens Designs for Aberration Control	Wide Ranging	Minimizes spherical and chromatic aberrations	Can be expensive

3.2.2 Evaluation for Selection for Lens System

Focal length is a critical parameter in lens design as it directly affects both magnification and field of view (FOV). Longer focal lengths provide greater magnification, capturing fine details of distant objects, while shorter focal lengths offer a wider FOV, which is essential for capturing broader scene context. In a multi-camera array, employing a range of focal lengths—from telephoto lenses (100mm–600mm) for detail to wide-angle lenses (10mm–35mm) for context—ensures comprehensive scene coverage. Zoom lenses (24mm–200mm) add versatility by allowing dynamic focal adjustments based on the imaging scenario. However, managing the trade-offs between focal length and FOV is challenging, as telephoto lenses may introduce perspective distortions, while wide-angle lenses can suffer from barrel distortions. Selecting the appropriate combination of lenses helps optimize both detail and context in the final composite image.

Field of view (FOV) determines the extent of the scene captured by each camera. Adequate FOV overlap between adjacent cameras is crucial for seamless image stitching, as insufficient overlap can lead to artifacts while excessive overlap increases redundancy and processing time. Typically, an overlap of 20–30% is recommended for reliable feature matching during the stitching process. Wide-angle and fisheye lenses are beneficial for capturing broad scenes, with fisheye lenses offering a FOV of 180° or more for panoramic coverage. Correcting distortions from these lenses, such as barrel and edge distortions, is essential to maintain the accuracy of the stitched images. Balancing FOV across different lenses in the array is necessary to simplify stitching and ensure that all critical information is captured.

Aberration control is essential for maintaining image sharpness and clarity. Chromatic aberration, caused by different wavelengths focusing at varying points, and spherical aberration, resulting from peripheral rays focusing differently than central rays, are the primary concerns. Using aspheric lenses helps minimize spherical aberrations by optimizing light paths, while

low-dispersion glass and achromatic doublets can reduce chromatic aberrations. Custom lens designs tailored to specific wavelength ranges or optimized for low-light performance further enhance image quality. However, these solutions can significantly increase costs and complexity. Efficient aberration control not only improves the clarity of individual frames but also simplifies the super-resolution and stitching processes by providing cleaner, high-contrast images.

Cost considerations for lens design are a significant factor when selecting components for a multi-camera array. The type of lens—be it telephoto, wide-angle, zoom, or custom aspheric—directly influences costs due to differences in material quality, coatings, and manufacturing precision. Aspheric lenses, for instance, offer superior aberration control but are more expensive due to complex manufacturing processes. Selecting a mix of cost-effective lenses without compromising on essential parameters like focal length and aberration control can help balance performance and budget. Prioritizing lenses with high-quality coatings for key positions in the array while using standard lenses for less critical roles can optimize both performance and cost.

In conclusion, selecting lenses for a multi-camera array requires a careful balance between focal length, field of view (FOV), aberration control, and cost. Longer focal lengths provide magnification essential for capturing fine details of distant objects, while shorter focal lengths widen the FOV to accommodate contextual imaging. By employing a combination of telephoto, wide-angle, and zoom lenses, the system achieves comprehensive scene coverage while addressing challenges such as perspective distortions and barrel distortions. Ensuring adequate FOV overlap between lenses is also vital for seamless image stitching, optimizing feature matching while minimizing redundancy and computational overhead.

Aberration control further enhances image clarity and sharpness, with solutions like aspheric lenses and low-dispersion glass reducing chromatic and spherical aberrations. Custom lens designs tailored to specific imaging scenarios improve performance but require cost considerations, given their complexity and manufacturing requirements. Cost efficiency is achieved through strategic selection, utilizing advanced lenses for critical roles and cost-effective options elsewhere. This thoughtful approach ensures the system maintains high performance while balancing budget constraints, resulting in a reliable, adaptable, and robust imaging solution for diverse operational needs.

3.2.3 Justification of Selection

The selection of a three-camera array, consisting of telephoto, wide, and ultra-wide lens systems, is a carefully considered choice that directly aligns with the requirements and objectives of this project. Each lens is designed with a fixed magnification of approximately 5, paired with specific fields of view (FOVs) that cater to different imaging needs. The telephoto lens, offering an FOV of about 0.9 degrees, is crucial for long-range detection, providing the capability to identify and monitor aircraft at the farthest edges of restricted airspace. This ensures that even distant objects can be precisely detected, which is essential for early warning and situational awareness. The wide lens, with an intermediate FOV of approximately 1.75 degrees, plays a complementary role by capturing medium-range imagery. This balance between detail and coverage is particularly

useful for scenarios where aircraft are neither too close nor too far, providing versatility in detection. Finally, the ultra-wide lens, with its broader FOV of around 3 degrees, captures close-range and wider perspectives, effectively covering areas closer to the system and minimizing the risk of blind spots. This lens arrangement creates a versatile and comprehensive optical component that meets the system's requirements for monitoring and detecting aircraft across varying distances and scenarios.

This selection supports the project's primary goal of augmenting an RF system to passively detect aircraft entering restricted airspace. The addition of an optical layer ensures that the system can capture visual evidence of potential airspace intrusions, enhancing its ability to safeguard sensitive areas. By combining the strengths of all three lenses, the system is well-equipped to detect aircraft whether they are distant, at a medium range, or within close proximity. This layered approach is especially valuable when addressing the dynamic and unpredictable nature of aircraft movement, ensuring that no critical visual information is missed. Additionally, the use of fixed focal lengths simplifies the system's design, making it reliable and easy to operate. For stationary or slow-moving aircraft, this fixed design ensures consistent imaging with minimal operator intervention, allowing for dependable passive detection under a variety of conditions.

The design also addresses key challenges while balancing trade-offs effectively. One of the primary challenges in developing a detection system is managing costs without compromising functionality. The choice of fixed lenses with a magnification of approximately 5 helps achieve this balance, eliminating the complexities and expenses associated with motorized zoom systems. These fixed lenses not only reduce production costs but also enhance reliability by avoiding moving parts, which are more susceptible to wear and failure over time. This makes the system more durable and suitable for long-term operation. While fixed FOVs can be seen as a limitation due to their lack of adaptability to highly dynamic scenarios, the inclusion of three lenses with distinct and complementary perspectives mitigates this drawback. Together, they provide a range of imaging capabilities that balance spatial coverage and detail, ensuring that the system is versatile enough to handle diverse detection needs.

Another critical trade-off is operational simplicity versus flexibility. While motorized lenses or systems with adjustable focal lengths offer greater adaptability, they require more complex mechanisms and higher maintenance, which can be detrimental in passive detection applications. The fixed-lens approach prioritizes simplicity and reliability, ensuring that the system remains operational with minimal need for recalibration or repairs. This design choice is particularly suited for unmoving or slow-moving targets, where stability and repeatability are more important than dynamic adaptability. Moreover, the system's tailored FOVs ensure that it can effectively monitor airspace and detect aircraft without requiring continuous manual adjustments, making it highly practical for its intended purpose.

3.2.4 Specific Lens Selection and Optical Configuration

The core optical design of this multi-camera imaging system is centered around a carefully selected set of lenses—three plano-convex and three plano-concave—with focal lengths of 250mm, 175mm, and 125mm for the plano-convex group, and −50mm, −35mm, and −25mm for the plano-concave group. This selection is intentional and purpose-driven, enabling the system to

achieve a target 5× magnification while offering sufficient flexibility in field of view and object size management across varying ranges. These lens configurations provide both optical diversity and a modular platform adaptable to environmental conditions and imaging requirements.

Plano-convex lenses, with their converging optical geometry, are ideal for focusing light from distant objects and are well-suited for high-magnification applications. The longer focal lengths—particularly the 250mm and 175mm lenses—enable a narrow field of view, making them optimal for isolating distant targets and capturing fine detail. These lenses form the "telephoto class" within the array, tailored for long-range detection and high-resolution imaging of targets at the farthest observable limits. Their role is fundamental to resolving small object features and ensuring visual confirmation of targets such as aircraft at extended distances.

Conversely, the inclusion of plano-concave lenses with short negative focal lengths—−50mm, −35mm, and −25mm—serves a complementary optical function. These lenses diverge light rays, effectively expanding the field of view. When used in conjunction with the plano-convex counterparts in a compound optical system or matched per camera module, the concave elements help manage beam expansion, flatten optical paths, and correct field curvature, enhancing edge sharpness and minimizing spherical aberration. Importantly, their contribution allows the system to preserve image fidelity across the sensor area and prevent loss of focus in peripheral zones. This combination facilitates the necessary image clarity and scale to feed into downstream image stitching and super-resolution algorithms.

Together, this diverse focal length array creates a hybrid optical platform capable of imaging across a range of spatial resolutions and distances. By mixing focal powers—positive (converging) and negative (diverging)—the design accommodates both high-magnification central imaging and broader contextual scene capture. Additionally, this approach simplifies the hardware configuration by reducing the need for motorized zoom systems or adaptive optics, thereby enhancing reliability and reducing mechanical complexity. Each lens pairing is fixed and calibrated to work within specific scene parameters, ensuring the consistency and repeatability needed for passive detection systems, especially those used in unmanned or continuous monitoring operations.

The strategic spacing of focal lengths also ensures optimal overlap between fields of view, a requirement for seamless image stitching and multi-perspective fusion. For example, the 125mm plano-convex lens, paired with a −25mm concave counterpart, covers mid-range distances and bridges the gap between high-detail long-distance lenses and wider short-range optics. This thoughtful segmentation of lens roles allows the system to maintain consistent magnification (5×) across variable object distances while retaining field coverage sufficient for object tracking and target localization.

In sum, this specific lens selection provides a robust, modular, and cost-effective optical framework for the multi-camera array. It enables high-performance imaging without sacrificing simplicity or escalating costs, reinforcing the system’s mission to passively and precisely monitor large spatial volumes with minimal operator intervention.

3.2.5 Conclusion of Lens System Research

The selection of the lens architecture for this multi-camera imaging system is grounded in both optical precision and operational pragmatism. Among the various lens types evaluated—including wide-angle, zoom, fisheye, and aspheric custom optics—the telephoto lens design emerged as the optimal choice for the system's intended purpose: high-resolution, long-range passive detection with consistent magnification. Telephoto lenses provide the narrow field of view and high magnification necessary to isolate and resolve fine object details at a distance, making them ideal for applications such as aerial target recognition, surveillance, and remote monitoring.

While wide-angle and fisheye lenses excel in contextual or panoramic coverage, they introduce significant distortions and offer limited detail at long distances. Zoom lenses provide versatility, but their mechanical complexity, susceptibility to calibration drift, and higher cost render them less reliable for fixed, long-term deployments. In contrast, telephoto lenses offer optical simplicity, high-resolution fidelity, and reduced geometric distortion, enabling precise and stable long-range imaging without the need for active lens adjustment. This design choice aligns with the system's passive operating model, where the emphasis is on repeatability, low maintenance, and minimal operator input.

Within the telephoto framework, the selected array of three plano-convex lenses (250mm, 175mm, and 125mm focal lengths) and three plano-concave lenses (−50mm, −35mm, and −25mm) further refines the optical system's capabilities. The plano-convex lenses function as primary converging elements, delivering high-magnification views essential for distant object detection. Their long focal lengths facilitate narrow FOVs, allowing the system to zoom in on specific regions with optical clarity. Complementing these, the plano-concave lenses serve a critical corrective and field-balancing role—expanding beam paths, flattening image curvature, and improving sharpness across the sensor plane. This pairing creates a well-balanced optical subsystem that ensures consistent 5× magnification, accurate object sizing, and high-resolution output suitable for downstream super-resolution stitching.

The fixed focal length design, while inherently less flexible than motorized systems, provides significant advantages in terms of mechanical reliability, system robustness, and cost efficiency. It eliminates moving parts that can fail over time, simplifies calibration, and reduces both hardware and software overhead. The lens array's modular structure also supports streamlined alignment and maintenance, making it highly suitable for persistent monitoring in dynamic outdoor or operational environments.

In summary, the deliberate choice of telephoto optics, implemented through a dual plano-convex and plano-concave configuration, reflects a strong synergy between imaging performance and system-level constraints. By prioritizing detail, reliability, and passive operation, this design ensures that the optical layer of the system meets the demands of high-resolution, multi-perspective detection while remaining scalable, affordable, and technically robust.

3.3 Image Stitching Methods for Super-Resolution Reconstruction

Once images are captured, they must be seamlessly combined to produce a high-resolution composite. Several advanced stitching techniques are employed to achieve this.

Feature-Based Image Stitching

Feature-based stitching aligns images by detecting and matching distinct key points within overlapping frames. Algorithms such as Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) allow for accurate image registration even when minor shifts occur (Harfouche et al., 2023). This method is particularly useful for integrating images from cameras with different focal lengths, ensuring a seamless transition between high-magnification and wide-angle captures.

Optical Flow-Based Stitching for Dynamic Adjustments

Optical flow techniques estimate pixel displacement between frames, allowing for real-time image alignment (Lin et al., 2015). This approach is beneficial in dynamic conditions where objects may shift slightly due to atmospheric turbulence. By continuously adjusting pixel positions, optical flow methods ensure that images remain aligned, even in challenging environments.

Multi-Resolution Pyramid Blending

To minimize visible seams between stitched images, multi-resolution pyramid blending is used. This technique constructs a multi-scale representation of each image and selectively blends details across different resolution levels (Zhuang & Tran, 2020). This approach ensures smooth transitions between adjacent images, eliminating hard edges that might otherwise appear in the composite.

Deep Learning-Based Super-Resolution Fusion

Deep learning techniques, particularly convolutional neural networks (CNNs), have revolutionized image stitching and super-resolution reconstruction. These networks are trained on vast datasets to intelligently fuse multiple images, enhance resolution, and correct distortions (Fang et al., 2025). By leveraging machine learning, the system can generate high-resolution images that surpass the native resolution of individual cameras.

Wavefront Correction for Atmospheric Distortions

Phase corrector plates are optical elements designed to pre-compensate for predictable phase distortions introduced by atmospheric turbulence. They work by introducing an inverse distortion pattern that cancels out the wavefront perturbations caused by air turbulence. These plates can be particularly useful in a static imaging setup where the distortions follow a predictable pattern, such as near heat-generating surfaces or over large bodies of water. By correcting for known

aberrations before the light reaches the camera sensor, phase corrector plates help mitigate some of the long-range imaging issues that typically degrade resolution.

There are two primary types of phase corrector plates: static phase plates and deformable phase plates. Static phase plates have a fixed correction pattern etched into them, making them ideal for long-term deployments where the turbulence profile remains relatively constant. They are often designed for specific wavelengths, ensuring maximum correction efficiency. In contrast, deformable phase plates are made from liquid crystal or piezoelectric materials, allowing for tunable wavefront modulation. These plates can adjust dynamically to changing atmospheric conditions, providing a more versatile solution for varying turbulence levels.

Several manufacturers offer high-quality phase plates that can be integrated into optical systems. Thorlabs, for instance, produces a range of phase plates designed for atmospheric correction, while Meadowlark Optics specializes in liquid crystal-based phase correctors that allow real-time adjustments. Boston Micromachines provides MEMS-based deformable mirrors that serve a similar function by dynamically altering the wavefront to counteract turbulence. Depending on budget constraints, a custom-etched static phase plate may be a cost-effective solution, while a liquid crystal-based corrector could offer real-time adaptability at a higher cost.

3.4 Adaptive Optics for Real-Time Correction

Adaptive optics (AO) is a more advanced approach that actively measures and corrects for wavefront distortions in real time. Unlike static phase corrector plates, which address pre-determined distortion patterns, adaptive optics systems continuously analyze incoming wavefronts and apply corrections dynamically. This capability makes AO particularly effective for applications that require real-time adjustments, such as astronomy, military surveillance, and biomedical imaging.

A typical adaptive optics system consists of three primary components: a wavefront sensor, a deformable mirror (DM) or spatial light modulator (SLM), and a real-time control system. The wavefront sensor measures incoming wavefront distortions using techniques such as Shack-Hartmann sensing or curvature sensing. Based on this data, the real-time control system calculates the necessary adjustments and transmits them to the deformable mirror, which then alters its surface to compensate for the distortions. This process happens within milliseconds, allowing for continuous correction even in rapidly changing atmospheric conditions.

Several commercially available adaptive optics solutions could be integrated into the multi-camera system. Thorlabs offers adaptive optics kits that include wavefront sensors and deformable mirrors, making them suitable for laboratory testing and field applications. Boston Micromachines provides MEMS-based deformable mirrors that can achieve high-speed correction, making them ideal for dynamic turbulence effects. Imagine Optic produces HASO wavefront sensors that, when paired with adaptive optics hardware, can significantly improve long-range imaging performance. While adaptive optics offers the highest level of correction, it comes at a higher cost and increased complexity, making it less feasible for budget-constrained applications. However, a simplified AO system using liquid crystal spatial light modulators (SLMs) may provide a balance between performance and affordability.

Table 3.4 Comparison of Resolution Strategies and Corrective Optics

Stitching Method	Method	Advantages	Applications	Limitations	Cost
Feature Based Stitching	Detects and matches key points	Can be effective for multiple focal lengths	Aerial imaging and panoramas	Cannot do dynamic Scenes	Moderate
Optical Flow - Based Stitching	Estimates pixel displacement in frames	Real time alignment	Remote sensing	Cannot do significant motion	Moderate
Multi - Resolution Pyramid Blending	Blends images at different resolution levels	Minimizes seams	Medical imaging	Long computation time	High
Deep Learning - Based Super - Resolution	Uses CNNs to enhance resolution	Extremely high resolution	Surveillance and medical imaging	Requires training data	High
Wavefront Correction	Uses phase corrector plate to pre - compensate for distortions	Reduces predictable distortions	Long range imaging and astronomy	Needs predictable distortion patterns	Moderate to high
Adaptive Optics	Uses wavefront sensors and mirrors to adjust light	Real time alignment	Astronomy and military surveillance	Very complex and high cost	High

By integrating multiple lens types and advanced stitching techniques, the multi-camera array can produce a high-resolution, super-resolved image capable of capturing distant objects with exceptional clarity. Each lens system serves a unique function, while computational algorithms ensure seamless integration of captured data. This combination of optical and computational

strategies will allow the system to overcome environmental challenges and achieve superior imaging performance.

3.5 Evaluation for Selection for Image Stitching

Resolution is a fundamental parameter for achieving high-quality super-resolution imaging. Higher resolution captures more detail, which is essential for both feature-based and optical flow-based stitching techniques. Ensuring consistent resolution across different lenses simplifies the stitching process and reduces interpolation artifacts. Deep learning-based super-resolution methods can enhance resolution beyond the native capabilities of individual cameras by intelligently merging information from multiple images. However, achieving higher resolution also demands increased storage, processing power, and training datasets, making it necessary to balance resolution requirements with computational resources and costs.

Computational complexity is a critical parameter that impacts both the feasibility and cost of implementing advanced stitching and correction techniques. Feature-based methods, such as Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF), offer moderate complexity suitable for static scenes, while optical flow methods, which estimate pixel displacement, require significant computational resources for real-time applications. Deep learning-based methods, though capable of producing the highest resolution and accuracy, are the most computationally intensive, often necessitating specialized hardware like GPUs. Efficiently managing computational complexity involves selecting the right combination of algorithms based on the specific requirements of the scene—whether it is static or dynamic—and the available processing capabilities.

Computation time is a practical concern for applications that demand real-time or near-real-time imaging, such as surveillance and navigation. Longer computation times can introduce delays, making the imaging system less responsive to rapidly changing scenes. Optimizing computation time involves leveraging hardware acceleration, such as GPUs, for deep learning and optical flow algorithms, as well as implementing multi-threading techniques to parallelize tasks. Selecting algorithms that balance accuracy with speed, like optical flow for dynamic scenes and feature-based methods for static scenes, can help manage computation time effectively without compromising on image quality.

Distortion and aberration correction using corrective optics is essential for long-range imaging, especially in hot and humid environments where atmospheric turbulence is a concern. Techniques such as wavefront correction and adaptive optics significantly improve clarity by compensating for distortions before the image reaches the sensor. Wavefront correction, using phase corrector plates, is cost-effective for predictable distortion patterns, while adaptive optics, involving real-time adjustments with deformable mirrors and wavefront sensors, is necessary for dynamic conditions. Selecting between these techniques depends on the predictability of distortions and budget constraints, with adaptive optics offering superior performance at higher costs and complexity.

Cost of corrective optics is influenced by the type of technology employed. Static phase plates offer a lower-cost solution for correcting predictable distortions, while deformable mirrors and real-time control systems used in adaptive optics significantly increase costs. Integration

complexity also adds to costs, as precise alignment and synchronization between corrective optics and camera sensors are required. To manage these costs, a hybrid approach that combines static phase plates for predictable distortions and simplified adaptive optics using liquid crystal spatial light modulators (SLMs) for dynamic scenarios can provide a balanced solution.

Complexity of integration is a crucial parameter affecting the reliability and maintenance of the imaging system. Complex corrective optics and stitching algorithms require precise alignment and calibration to function effectively, which can increase failure rates and maintenance costs. Modular designs that allow individual components—such as lenses, phase corrector plates, and adaptive optics modules—to be replaced or upgraded independently can simplify integration. Standardizing communication protocols between components and using software-based calibration tools can also reduce integration complexity and enhance the overall robustness of the system.

Price and cost efficiency of stitching strategies and corrective optics are significant factors in the overall system design. High-end solutions, such as deep learning-based super-resolution and adaptive optics, offer superior performance but at a steep cost due to the need for specialized hardware and extensive computational resources. Open-source software solutions like OpenCV and TensorFlow can help reduce software costs, while cloud-based processing can offload the need for high-performance local hardware. Cost efficiency can also be improved by modularizing the system—using high-cost, high-performance components only where they provide the most significant benefits and standard components elsewhere.

In summary, designing an effective imaging system requires a thoughtful balance between resolution, computational complexity, computation time, distortion correction, and cost. Resolution is a key driver of image quality and is critical for super-resolution stitching techniques, yet achieving higher resolution demands careful management of storage, processing power, and computational resources. Computational complexity must align with the requirements of the scene and available hardware, ensuring that algorithms like SIFT, SURF, or deep learning-based approaches are selected based on feasibility and cost considerations. Similarly, computation time directly impacts system responsiveness, making hardware acceleration and algorithmic optimization vital for achieving real-time imaging performance.

Addressing challenges such as distortion and aberration correction is crucial for maintaining image clarity, particularly in demanding environments. Cost-effective solutions, including static phase plates and hybrid adaptive optics, provide viable options for balancing clarity with budget constraints. By leveraging modular designs and standardized integration, the system can reduce maintenance challenges and improve overall reliability. Ultimately, the strategic selection of components and algorithms ensures that the imaging system delivers high performance while remaining cost-efficient, robust, and adaptable to diverse operational scenarios.

3.5.1 Justification of Selection

In the design of high-resolution imaging systems, the image stitching method plays a pivotal role in determining the quality, performance, and feasibility of super-resolution reconstruction. Among various approaches—ranging from optical flow and deep learning-based fusion to

adaptive optics and corrective phase plates—feature-based stitching emerges as a highly pragmatic and technically sound choice. This decision is driven primarily by three critical considerations: cost-effectiveness, moderate computational demands, and manageable implementation complexity.

Cost Efficiency and Practicality

One of the foremost advantages of feature-based stitching is its relatively low operational and infrastructural cost. Unlike deep learning-based super-resolution methods or adaptive optics systems, which require extensive datasets, training cycles, and specialized hardware such as GPUs or deformable mirrors, feature-based algorithms such as SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), and ORB (Oriented FAST and Rotated BRIEF) can be executed using standard computing resources. These algorithms are widely supported by mature, open-source libraries such as OpenCV and VLFeat, which further reduces both software licensing and development costs.

In contrast, deep learning-based stitching often incurs hidden costs in data curation, model tuning, hardware acquisition, and long-term system maintenance. Adaptive optics and phase correction strategies, while powerful, are typically expensive and require high-precision alignment and control mechanisms, making them less suitable for budget-constrained or field-deployable applications. Feature-based stitching, by comparison, leverages readily available computational techniques and avoids dependency on proprietary optics or model training, making it the most financially sustainable solution.

Optimized Computational Complexity

Feature-based methods strike a favorable balance between computational load and stitching quality. Techniques like SIFT and SURF operate by detecting and matching distinct key points within overlapping images, which allows for precise alignment without relying on dense, pixel-by-pixel processing. This approach significantly reduces the computational burden compared to optical flow methods, which require real-time tracking of every pixel and are therefore resource-intensive, especially in large image datasets or high-frame-rate scenarios.

Furthermore, while deep learning models provide excellent results, their runtime and energy consumption are typically higher due to the volume of parameters and layers involved. Feature-based methods, however, can be executed on moderately powered CPUs and benefit from GPU acceleration when available, without mandating it. Their relatively low memory footprint and linear scalability with image size make them ideal for systems where real-time performance is a consideration but not at the cost of excessive complexity.

For super-resolution stitching involving multiple cameras with varying focal lengths, feature-based methods are particularly effective. Their robustness to scale, rotation, and slight illumination changes ensures reliable registration even when image frames differ significantly in perspective or content detail. This eliminates the need for elaborate pre-alignment or manual calibration.

Manageable Implementation Complexity

In addition to cost and computation, the complexity of integration is a key factor in selecting feature-based stitching. Implementation involves a well-defined pipeline: keypoint detection, descriptor extraction, feature matching, transformation estimation, and blending. These stages are modular, allowing for fine-tuning or substitution of components without overhauling the entire system. Moreover, these methods have been extensively studied, optimized, and documented, making them more accessible to engineers and researchers without deep expertise in computer vision.

By contrast, optical flow-based stitching demands continuous calibration and error handling for motion vectors, making it more prone to instability, especially in real-world, uncontrolled environments. Deep learning pipelines often introduce a "black-box" element, where tuning or debugging model behavior can become prohibitively complex for teams without dedicated machine learning expertise.

Feature-based stitching is also highly compatible with hybrid image pipelines. For instance, it can be used as the initial registration step before applying pyramid blending or even shallow learning-based refinement. This modularity makes it a strategic choice for systems that may be scaled or upgraded in future iterations.

A Balanced Choice for Static and Semi-Dynamic Scenes

Given the imaging context in question—comprising multiple cameras with varying focal lengths capturing largely static scenes for super-resolution purposes—feature-based stitching is not only sufficient but optimal. It offers robustness to minor misalignments and lens distortions, which are common in practical deployments. Unlike optical flow, which is better suited for dynamic scenes with continuous motion, or adaptive optics, which target real-time wavefront correction, feature-based methods provide a strong middle ground that accommodates moderate scene variability with high stitching accuracy.

The trade-offs are intentional and aligned with project goals. By choosing feature-based stitching, the system avoids over-engineering and maintains a focus on delivering cost-effective, efficient, and scalable super-resolution imaging.

3.5.2 Conclusion of Lens System Research

In the context of developing a robust and efficient super-resolution imaging system, the selection of feature-based image stitching represents a deliberate and well-balanced engineering decision. Compared to more computationally intensive or hardware-dependent alternatives, feature-based techniques offer a compelling combination of precision, affordability, and implementation simplicity. By leveraging algorithms like SIFT and SURF, the system achieves reliable image alignment across frames with varying focal lengths, ensuring the seamless integration of high-magnification and wide-angle views without the need for specialized optics or GPU-heavy processing.

This approach aligns well with the broader design philosophy of the imaging system—prioritizing modularity, cost-effectiveness, and operational efficiency. While deep

learning and adaptive optics provide higher-end capabilities, their added complexity and expense make them less suitable for scalable or field-deployable solutions. Feature-based stitching offers a practical pathway to achieving high-resolution composite images with minimal infrastructure overhead, making it an ideal choice for scenarios that demand a balance between performance and practicality. Through this choice, the system maintains its flexibility, reliability, and readiness for future enhancements in super-resolution and long-range imaging.

3.6 RF Device

Research Goals

The RF device will require a few different things in order for us to be successful in our testing. One of the main things is our antenna technology. We will have to be very careful with what kind of antenna we choose as that is how all of our information is received. If we do not get an antenna that is wideband enough or sensitive enough then the rest of the equipment will not work. The next part is our spectrum analyzer which has to be accurate and sensitive enough for our use case. Next is our power source which will prove the viability of the overall functionality. Having a lower maintenance power source is a good idea as it allows us to place these in remote locations. Finally, we hope to find what kind of communication protocols will happen between all of the different devices. It is important for our computer to be able to decode the information that the spectrum analyzer may produce. With all of these in mind we will be able to successfully create a working prototype.

3.6.1 Overview of Antenna Technologies

Antennas are a technology that allows us to send Radio Frequency electromagnetic waves across space. There are many different types of antennas and uses for them depending on the parameters that may be found useful for a specific use case. For example, your ear acts as an antenna for sound waves as opposed to electromagnetic waves. They redirect the sound waves flying around space and send them to your eardrum. Your ear drum then vibrates and translates it to language. Antenna work in two different distinct modes, transmitting and receiving. Transmitting is the use case in which we will send an electromagnetic wave. Receiving is where we will “listen” to the transmitted wave and convert it into electrical signals.

Furthermore, they are incredibly useful as it allows for wireless communication between devices. Before antennas, long lines of wire would have been needed to communicate between long distances. Such as when the telegram or early telephone were used, they needed to run a physical wire between the two places which is very costly and can be challenging in rough environments. With antennas, we can forgo a lot of that cost by simply sending it across space.

As previously mentioned, there are many different types of antennas which we will explore for the use of our system. First, we need to define the parameters of antennas and how they will be useful for this system. There are also many different frequencies at which the antennas operate which will have to be narrowed down for our use case.

First, we need to define some basic parameters so we can research the specific antenna we may use. The first one being frequency; Which simply put is how quickly the electromagnetic wave

will repeat in a given time. Naturally, we will need to define the wavelength which is the distance between the peaks of each wave. Finally is radiation intensity. Which is the "power per unit solid angle." (tutorialpoint, 2019)

Next, we need some antenna specific parameters which will help us in choosing the best antenna possible for our system. Before that, in antenna theory, we like to compare anything we do to a "isotropic antenna." Which is an antenna that radiates equally among all directions. Shown in *Figure 3.5*. We use this as a reference for all other antennas to measure their specific parameters. Isotropic antennas are impossible, but are used because they are easy to make calculations with and act as an ideal case. Simply put, they are a reference at which we like to compare.

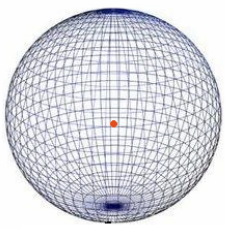


Figure 3.5 *Isotropic antenna radiation pattern (everythingRF, 2021)*

The first parameter discussed will be directivity. Which is the comparison of the maximum radiation intensity of the antenna in comparison to the radiation intensity of the isotropic antenna. Next is the Aperture Efficiency. Which is the ratio of the "effective radiating area to the physical area of the aperture." (tutorialpoint) Meaning, how much of the area of the antenna will actually be able to radiate. Next will be the antenna efficiency which is the ratio of the radiated power to the input power. Finally, is the antenna gain which describes how much power is transmitted in dB.

Furthermore, there are many different types of antennas that will need to be researched for the sake of finding the best type for our use case.
The first type that we will explore will be Dipole antennas. Also known as transmission line or wire antennas. They work by leaving one end of a wire open to space. The energy tries to escape and radiates the energy into the atmosphere. For this to work most effectively, the end of the transmission line must be impedance matched to free space.
There are a few different types of dipole antennas.

First we will start with the half wave dipole. This name simply comes from the fact that length of wire is half a wavelength. The half wave dipole is one of the most widely used antennas. It is typically used in radio receivers or television receivers. It can operate anywhere from 3KHz to 300GHz. This provides a very wide band of frequencies that we could choose from. However, the draw back of this is that you will need to design the antenna for specific frequencies. The bandwidth of this type of antenna is small and not good for our specific use case.

Dipole antennas are also omni-directional in the horizontal plane. Making this seem very desirable for our application as we want to cover as much air space as possible with as little

antennas possible. Not only that, but this type of antenna is very cheap to manufacture. Making this a great option for us as we are trying to keep costs as low as possible. It also has great directivity of about 2.15dbi on average. The efficiency of this kind of system is quite good as it allows less loss. As previously mentioned, this has major drawbacks such as the limited bandwidth. It can also be quite large as the size of the transmission line relies on the frequency. It also needs to be at least a quarter wavelength off the ground in order to work. It is also easily affected by other nearby objects and can degrade the performance. This all sums up for a bad implementation into our system.

The next technology that we will explore is the Half-wave folded dipole. This works a lot like the previous half wave antenna however it is folded closed together in a cylindrical shape. On both sides of the antenna the wire length is quarter wavelength as opposed to the half wavelength of the previous antenna. An image of this antenna can be found in *Figure 3.3*. The radiation pattern of the half-wave folded dipole is very similar to that of the previous antenna as well as the frequency range.

The main benefit and difference in comparison to the regular half wave antenna is that it has a higher input impedance of 300Ω in comparison to the half waves 70Ω input impedance. It's important to note that we can achieve an even higher input impedance by adding more wire elements. This is important for use in systems that may have a higher input impedance or where optimal power transfer is necessary. Such as its use in Yagi-Uda antennas where this acts as a feed for the system. Not only that it has better bandwidth characteristics which could be useful for our use case. However, it is still too small for what we need. It also has better signal to noise ratio performance which in turn allows higher resolution and signal quality. This type of antenna faces the same drawbacks as the previously mentioned half wave antenna.

Next, we will discuss the microstrip or patch antenna. The microstrip antenna is a very thin metal strip placed on a dielectric. The patch is usually half wavelength and when power is added it will radiate from the edge of the strip. The microstrip has a very large radiation pattern. However, its radiation power is very small and has a very thin bandwidth. One of the great things about this type of antenna is it is very low cost and lightweight. They typically have these inside of cell phones or other high speed and precision computing.

Furthermore, the types of antennas discussed so far have not been the most useful for our use case. They have primarily been the basic antennas that are used in very particular uses or for antenna arrays. As mentioned, the Half-wave folded dipole is typically used as a feed for Yagi-Uda antenna arrays which we will explore. Antenna Arrays is simply combining the strengths of a single antenna and multiplying it by adding many similar antennas. The antennas will be placed so close together that they lie in each other's induction field. Which is important as it allows "the radiation pattern to be the vector sum of the individual ones." (tutorialspoint, 2019)

This is extremely important as it allows us to have high gain, high directivity, and overall better performance for every part of the antenna while having very little drawbacks. Some more strengths include a higher signal to noise ratio, which is extremely important for our use case as the amount of power received from the aircraft will be very small. Having a higher signal to

noise ratio allows us to reduce the probability of a false alarm. The amount of power used is much lower.
Some of the major drawbacks of antenna arrays include: Large antenna dimensions, which leads to harder implementation and mounting. The resistive losses between elements are increased. They are typically used in satellite communications and wireless communications. One of the major use cases is also military radar. Which is a good indicator that they will be one of our best options. Antenna arrays aren't absolutely perfect, but seem to be the best trade in performance. We will take a look into a few different types and their advantages which will then lead to the selection of the antenna used in our project.

Collinear arrays are antenna arrays that are made of half wave dipoles placed end to end. They are called collinear because they are typically placed on a parallel line or axis. (tutorialspoint) This is a great benefit because as we explored, half wave antennas are very cheap to produce. They also have great forward gain and directivity. The frequency range can be from 30Mhz to 3Ghz. (tutorialspoint, 2019) This is the kind of frequency range that we really want to work with as a lot of aircraft and telecommunications lay on this frequency range. Those are the types of things that we are particularly interested in picking up. One of the other interesting things about the collinear antenna is that it's gain is dependent on the amount of elements. The more elements the higher gain. Anywhere from 1.9dB to 4.3dB which is a very high gain. (electronics-notes) Of course the radiation pattern will have some side lobes, However, this makes the radiation pattern uni-directional. An image of the beam pattern can be found in Figure 3.5. Some of the drawbacks are the size and deployment of this type of antenna. They are typically very large and reserved for outdoors which may pose an issue to the deployment of the system. An image can be found in Figure 3.6.

Next, we will discuss the Yagi-Uda antenna. It is made up of parasitic elements and dipole antennas. (tutorialspoint, 2019) Parasitic elements are elements that do not have their own feed. (tutorials point, 2019) As previously mentioned, the half wave folded dipole is the feed for this antenna type. An image can be found in Figure 3.7. This antenna has a very small bandwidth and a very high directive beam pattern. It also has an extremely high gain of 10dB or more. The frequency range can be from 3 Mhz to 3Ghz depending on the use. They are used everywhere in the world and can be found on rooftops everywhere. However, the main drawback of this antenna for our use case is the fact that the bandwidth is very small. Typically a few percent of the desired center frequency. (antenna theory) This could be an incredibly powerful antenna for our use case if we were to choose a frequency to detect planes. However, since we are going for a completely passive radar approach, it would likely be better to have a large bandwidth. The Yagi-Uda is a very strong contender for our use case.

The last antenna type we will explore is the log-periodic antenna.The log-periodic very similar to the Yagi-Uda antenna. It has a frequency range from 30 Mhz to 3Ghz. Making it a very wide band antenna. However, one of the major advantages over the Yagi-Uda is that it has "constant characteristics over a desired frequency range of operation." (tutorialspoint, 2019) Meaning that it has a very wide bandwidth and we can work with a very wide range of frequencies. It also has relatively strong and constant gain and radiation patterns across the operating frequency range. A chart of a log periodic dipole can be found in Figure 3.8 which plots the VSWR of the antenna across the frequency range. Low VSWR means that the antenna works well. The radiation

pattern is unidirectional or bidirectional depending on how you orient it. We would like the unidirectional pattern as it works very well for our application. A figure of the radiation pattern can be found in figure 3.9. It also has very good gain ranging from 6 - 12 dB depending on how many elements the antenna has. This wide range of gain will allow us to optimize our system for the absolute most amount of frequency coverage while providing good accuracy for detection. The major drawbacks of log periodic antennas is of course the size. Just like the Yagi, it is a little bit on the larger side. It also costs more than simple antenna systems, but that will ring true for any kind of antenna array. This is by far the strongest contender as it has a wide bandwidth and frequency range. While also providing a great gain. We will explore some specific commercial options for these types of antennas and the benefits. After that, we will evaluate the best antenna for our use. Below the VSWR or Voltage Standing Wave Ratio of a log periodic antenna can be found. The lower the VSWR the less noise and better, more accurate data will be present. It is important to see that the VSWR is extremely low for a large range of frequencies. Much larger than that of the other antennas. Not only that, but an image of the radiation pattern can be found and we can see how wide it is. Working for a large range of angles will increase our chances of detection. Making this an extremely strong contender.

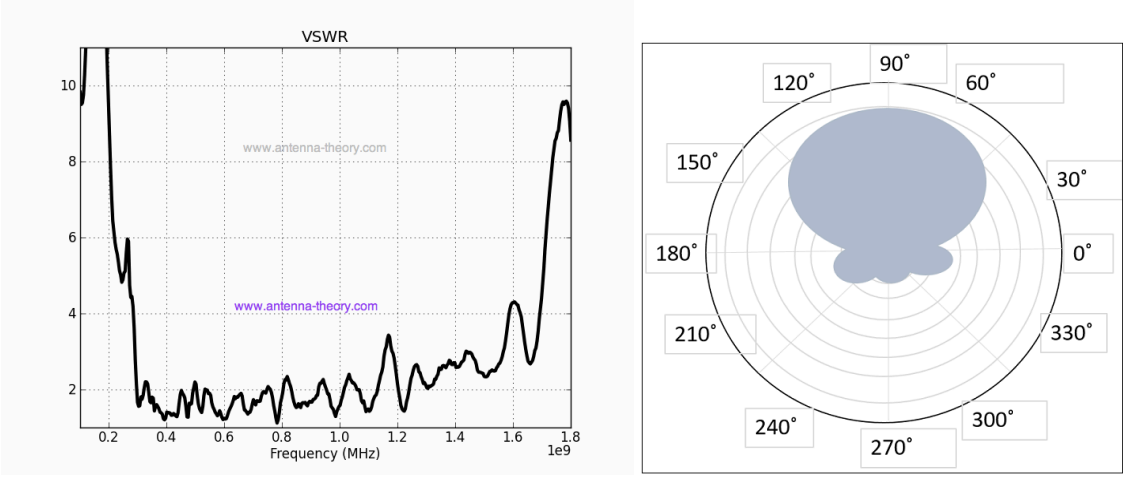


Figure 3.6 and 3.7 *VSWR of LPDA and Radiation pattern, (antennatheory, 2009) (tutorialspoint, 2025)*

Finally, we have compared a lot of different antenna technologies. However, we will be diving deeper into the antenna array technologies as they are an extremely good fit for our use case.

First we will start with the collinear array antenna. The antenna found was the Laird / Antenex BB8965C. It is a 14 in long antenna that has a good gain of 5dB. It works from 896 Mhz to 970 Mhz. As previously mentioned, it has a small frequency band. The reason that this band was specifically chosen was because it includes some aeronautic mobile communications and land / mobile communications as well. If we are trying to increase our chances of detection this would be a good range to go with. However, it was difficult to find antennas that worked specifically in the aeronautical radionavigation bands. Which typically range from 960 to 1164 Mhz. This one is good because there is a fair chance that a signal could bounce off a transponder on the ground. It also touches a bit into the aeronautical radio navigation band. Another benefit of this antenna is that it is very small and cheap. This could possibly be a good option. However, the bandwidth

is a little small for what we may need. In the image shown below you can also find the radiation pattern of a collinear antenna. It makes it a strong contender as it is very broad and allows us to be less picky about what direction the antenna is pointed. It will also increase our chances of detecting an aircraft as we are picking up a larger area of signals.

Next, we will focus on the Yagi antenna. The Kathrein TY900 is a great option for our use case. As previously mentioned, it works in the same frequency range as the last. This one has a peak gain of 12.2dB and is also weather proofed. It is a larger antenna measuring at 24 x 6.6 in. However, given the fact that it is weather proofed, it should be a great option if we would like to mount it in a tree. The high gain means it would be extremely accurate as well. The price is a little bit higher, but the trade off is well worth it as we lower our probability of a false alarm dramatically with higher gain. However as with the last the bandwidth may be a bit too small for our application. Below an image can be found of the radiation pattern of a yagi antenna. We can see that it is highly directional which may be a large problem for our use case.

Lastly, we will look at the log periodic dipole antenna. The Aventas AX-37A log periodic dipole antenna. This antenna has a frequency range from 300 to 3000 Mhz. By far the widest frequency range we have had yet. This would be incredible for our uses as it allows us to capture almost any communications or reflections that may happen. It also has a forward gain of 6dB. The antenna also has a built in low noise amplifier. This would help our spectrum analyzer and overall accuracy. We can see that the radiation pattern is also omni directional which will work well with our system. It is also dimensionally very small as there is a version that is meant to act as a portable antenna. The major drawbacks of this solution is the cost. This is by far the highest cost solution, but will provide the most benefits in the most amount of fields. Jack of all trades and the price comes with it. One of the important things to also know about this antenna is that it requires a 9V battery to work. This will not be a problem for the prototyping phase of this. However, it will need to be addressed likely with the Aventas AX-37AX which does not have a built in preamplifier and as a result doesn't need a 9V battery.

It is also important to note that any antenna that we use will have to be ruggedized to work in the environment that we intend to work in, which is the jungle. It is a similar environment to Florida's environment, but it may need to have some sort of enclosure that allows it to work in all cases.

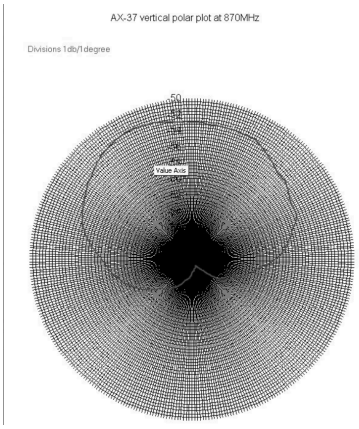


Figure 3.8 *Aventas AX-37A and Radiation Pattern (Aventas, 2025)*

Table 3.5 Comparison of Antenna Technologies

Antenna Type	Frequency Range	Beam Pattern	Gain (dbi)	Cost
Half Wave	3K to 300G	Omni	2.15	$80.00 - $200.00
Half Wave Folded	3K to 300G	Omni	4	$509.00
Microstrip	915-930 Mhz	Broad	2	$14.00
Collinear	896 - 970 Mhz	Omni	5	$85.15
Yagi-Uda	880 - 960 Mhz	Omni	12.2	$247.00
Log Periodic	300 - 3000 Mhz	Omni	6	$0.00

3.6.2 Evaluation for Selection

For our solution, we need an antenna that has a very wide bandwidth as we are not sure what kind of frequencies may reflect off of an aircraft. Not only that it needs a high gain in order for us to be able to accurately detect the aircraft. The antenna that works best is by far the Aventas AX-37A as it has good forward gain, and the largest by far band width. It is also dimensionally small and will work extremely well for our use. It is the most expensive but works best. For our use case our sponsor bought the log periodic antenna for us as we both agreed it would be the best technology. As a result the price is reflected in the table.

3.7 Overview of Spectrum Analyzer Technologies

Spectrum analyzers are an Electrical Engineering essential. They allow us to capture an electrical signal and display it in graphs that make it easy for us to understand. With this we are able to detect whether there was an aircraft detected based on the amount of power that we will receive with the antenna. Subsequently if we see a spike that means that we were able to detect an aircraft or something else like a radio transmitter. We will be doing analysis in the frequency domain meaning our spectrum analyzer has to be real time. The signals will be received in real time. This allows for wideband analysis, greater degree of accuracy. This allows us to get the best understanding and analysis of the electrical signals we are receiving.

Spectrum Analyzers have a few very important parameters that we will need to look out for. One of them being the Real Time Bandwidth or the RTBW. RTBW is how much of the frequency range can be analyzed without any gaps in the electrical signal. Subsequently, we will need a good frequency range. The more range that the spectrum analyzer and the antenna have the better chance we have at detection. Next is the dynamic range and sensitivity. Dynamic range is the ability for the spectrum analyzer to detect a low power signal in the presence of other stronger signals. Sensitivity or sometimes called Displayed Average Noise Level is how well it can react to weaker signals. Resolution Bandwidth is also very important as it determines how well it can distinguish two very close together signals. Another important parameter is size. The spectrum

analyzer needs to be small enough to be carried to remote locations. Finally, likely the most important parameter is the cost.

The first spectrum analyzer is the Tektronix RSA306B USB Spectrum Analyzer. It has a frequency range from 9 Khz to 6.2 Ghz. This will provide more than enough range for our use case. It has a RTBW of 40 Mhz. The dynamic range is -160dBm to 20 dBm. This is overall a great option as it is also USB powered making it compact and easy. The spectrum analyzer also comes with analysis software. Some of the performance may not be necessary as 6.2 GHz may be too high. Subsequently this could possibly be a place to save some money.

Next is the Siglent SSA3032X-R. This seems to be a great offering as it has the perfect frequency range of 9 Khz to 3.2 Ghz. This will likely be our most effective frequency range and should be optimized cost wise. This has an even greater difference between DANL than the last at -165dBm/Hz. It also has an RTBW of 40 Mhz. However, it is not necessarily compact and is more like a regular spectrum analyzer in terms of size. This will subsequently need a 120V wall outlet and will be a big trouble for our use case. Not only that but it has only 100 hours for free trials of the software. There is also an additional software based accessory you have to pay for which is the SSA3000XR-RT40.

Furthermore we have an offering from Signal Hound, the BB60D. This is a very compact spectrum analyzer that is powered by USB. It uses only 6 watts which is an incredibly low power figure. Not only that it has a very wide frequency range from 9 Khz to 6 Ghz. It has a dynamic range from -158dBm to 10dBm. The major drawback is that it has only a 27 MHz RTBW. However, it is the cheapest of the bunch so far and also has free software that you don't have to subscribe to. It has incredible system specs while also being compact and small. They may not be the best performing specs, but it is the cheapest smallest and does most of what may be necessary. One of the other big considerations with the BB60D is the fact that it has compatibility with Linux. With this in mind, we would be able to have the processing done on a small computer such as a Raspberry Pi or a Nvidia Jetson Nano. This is extremely useful for our use case as we are trying to lower the power consumption and size and this may be a viable option for us. The software also has a SDK which will allow us to make an incredibly powerful detection algorithm that will reduce our power consumption and increase our detection accuracy. We plan to implement machine learning or AI to help with our detection algorithm

Finally, we will talk about the Keysight N9023B PXA. It has the widest frequency range from 2 Hz to 55 Ghz. The RTBW is a staggering 2 Ghz and has a DANL of -167dBm. It has a dynamic range of -170 dBm to 30 dBm. This is by far the best performing spectrum analyzer with top of the line specifications. The drawbacks of this system are the size and power consumption. Not to mention this costs $207292.56.

3.7.1 Evaluation for Selection

Finally, we chose the Signal Hound BB60D. This is because it is simply the lowest cost. One of the design constraints of the system is that it has to be low cost. It has specifications that get extremely close to the rest of the spectrum analyzers. Excluding the Keysight. The main drawback of this is the smallest RTBW. However, once we test and find the most probable frequencies it shouldn't be that much of a problem as 27 Mhz is still quite a wide range of

frequencies. Not only that but it has an incredibly dynamic range and DANL. Which should prove extremely useful for our purpose as the power received from the aircraft will be extremely small. One of the important aspects of this spectrum analyzer is also the fact that the software is free and includes a Software Development Kit. The team and the sponsor also agreed that this would be the best option for our project. It has the most benefits for the least cost. As such, the price is reflected in the table as we didn't pay for the device.

Table3.6 Comparison of Spectrum Analyzers

Spectrum Analyzer	Frequency Range	DANL dBm/Hz	RTBW	Dynamic Range	Cost
Tektronix RSA306B	9 Khz to 6.2 Ghz	-160	40 MHz	-160 dBm to 20 dBm	$6490
Siglent SSA3032X-R	9 Khz to 3.2 Ghz	-165	40 MHz	60 dB	$6545
Signal Hound BB60D	9 Khz to 6 Ghz	-158	27 Mhz	-150 dBm to 10 dBm	$0.00
Keysight N9023B PXA	2 Hz to 55 Ghz	-167dBm	2 Ghz	-170 dBm to 30 dBm	$207292.56

3.8 Overview of Mini PC

When selecting a computer platform upon which to operate data from a spectrum analyzer, the Raspberry Pi, NVIDIA Jetson Nano, and Intel NUC 13 NUC13ANHi7 all have certain strengths and concessions. The candidates vary in calculation capacity, resource usage, and suitability for use in real-time RF analysis. While the Raspberry Pi is a low-power, low-cost solution and the Jetson Nano offers better GPU-accelerated processing, the Intel NUC 13 is the most practical and powerful solution for prototyping and development.

The Raspberry Pi 4 or 5 is an appealing choice as it is low power and low cost. With a quad-core ARM Cortex-A72 processor and 8GB of RAM, it is a good fit for simple signal processing tasks with software like GNU Radio or SDR#. It also has the ability to interface with spectrum analyzers via USB 3.0, so it is a solid option for simple RF data logging. But it is not efficient in processing big datasets in real time since it has limited CPU power and no separate GPU for parallel processing. The Raspberry Pi is a functional solution for those who need only light spectral analysis or who are working in low-power, embedded environments, but because of the computing limitations, it is not possible for heavy signal analysis.

The NVIDIA Jetson Nano is a higher-end model, with a quad-core ARM Cortex-A57 CPU and a 128-core NVIDIA Maxwell GPU. With its GPU acceleration, it can provide support for parallel signal processing tasks, which makes it a suitable one for machine learning-based spectrum analysis. Since it supports CUDA, TensorFlow, and PyTorch, it can be utilized to develop RF anomaly detection, interference classification, and real-time signal pattern recognition. However, for all its AI capabilities, there are some limitations with the Jetson Nano, such as lower general-purpose CPU performance, limited RAM (4GB LPDDR4), and higher power consumption (10W versus 5W for a Raspberry Pi). Though an excellent choice for AI-enhanced spectrum surveillance, it lacks the storage and CPU performance needed for heavy multi-threaded signal processing.

The Intel NUC 13 NUC13ANHi7 Arena Canyon Mini PC is an excellent option to process data from a spectrum analyzer with an excellent mix of performance, portability, and power efficiency. Equipped with an Intel Core i7-1360P processor, 16GB RAM, and a 512GB SSD, the mini PC is able to process the signals in real time, acquire and analyze data for applications such as RF spectrum monitoring and passive radar systems that are computationally demanding in nature. Since it has a small form factor, it can be conveniently deployed in the field or in small spaces, making it appropriate for mobile and fixed signal processing applications.

A prominent advantage of the Intel NUC 13 Arena Canyon is its multi-core performance. The Intel Core i7-1360P processor features 12 cores (4 performance cores and 8 efficiency cores) and 16 threads, which makes it highly capable of handling the high-speed data streams from spectrum analyzers like the Signal Hound BB60D. The combination of efficient cores and high clock speeds ensures that the real-time processing of data is not interrupted and happens seamlessly, even with large spectral data. Additionally, the 16GB RAM provides more than sufficient memory to run advanced signal processing software such as MATLAB, GNU Radio, or SDR# without the risk of delays or bottlenecks.

A second key feature of the NUC 13 Arena Canyon is that it is equipped with high-speed storage and connectivity. The 512GB NVMe SSD offers high-speed data logging, which is crucial in wideband RF signal capture and analysis. Spectrum analyzers have a tendency to generate large files of data that should be stored and accessed at high speeds, and the high read/write speeds of NVMe SSDs enable data processing in real time with as minimal latency as possible. On top of that, the multiple USB ports, Thunderbolt 4, and Ethernet connectivity allow simple integration with numerous RF equipment, external storage, and networked processing systems, making it an excellent hub for data acquisition and analysis.

Whereas Raspberry Pi and Jetson Nano are particularly good for low-power embedded computing and AI-processing, Intel NUC 13 NUC13ANHi7 is the best for prototyping and real-time RF data processing. It features an Intel Core i7-1360P processor (12 cores, 16 threads), 16GB RAM, and a 512GB NVMe SSD with significantly higher processing capacity and memory storage than either of the single-board computers. This makes it ideal for running high-performance software like MATLAB, GNU Radio, or SDR software with advanced DSP algorithms. The NUC 13 has the capability of real-time data capture, filtering, and analysis of wideband RF signals without performance bottlenecks that are associated with ARM-based platforms.

A further significant advantage of the Intel NUC 13 is that it is expandable and highly connected. It includes Thunderbolt 4, multiple USB 3.2 ports, and gigabit Ethernet, and hence is highly compatible with fast-speed spectrum analyzers like the Signal Hound BB60D. Unlike the Jetson Nano that relies almost totally on GPU acceleration for tasks connected to AI, the NUC features an end-to-end design with good CPU performance and also hardware acceleration due to Intel's integrated Iris Xe Graphics. Additionally, with Windows 11 Pro or Linux, it provides an easier and more flexible development platform for prototyping.

The Intel NUC 13 is the best solution for prototyping as it offers desktop-class performance with a compact size, which means it can process both real-time spectrum analysis and data-intensive programs. Compared to the Jetson Nano, which is optimized for AI, and the Raspberry Pi, which is limited by the restrictions of CPU performance, the NUC can easily complement professional RF analysis software and be easily expandable for more complex signal processing applications. Despite requiring more power (~90W to 10W for Jetson Nano and 5W for Raspberry Pi), its higher processing capacity, speed storage, and software compatibility with multiple platforms give it the best all-in-one solution for heavy-duty RF analysis and prototyping.

Besides its performance attributes, the NUC 13 Arena Canyon is a power-saving computing solution as well. It has a maximum power usage of approximately 90W and therefore consumes much less power than full-sized desktop workstations, making it ideal for remote or off-grid deployments. Paired with a solar panel and portable power station, it can be run for weeks in the field without access to a constant power supply. This makes it highly suitable for mobile spectrum monitoring tasks, e.g., passive radar detection, interference monitoring, or environmental RF scans.

Table 3.7 Comparison of Mini PC Technologies

Feature	**Intel NUC 13 (NUC13ANHi7)**	**Raspberry Pi 5**	**NVIDIA Jetson Nano**
Processor	Intel Core i7-1360P (12-core, 16-thread)	Quad-core Cortex-A76	Quad-core Cortex-A57
RAM	16GB DDR5	Up to 8GB LPDDR4X	4GB LPDDR4
Storage	512GB NVMe SSD	microSD or SSD (via adapter)	microSD or SSD (via adapter)
GPU	Intel Iris Xe Graphics	Broadcom VideoCore VII	NVIDIA Maxwell (128 cores)
Power Consumption	~90W	~5W	~10W
OS Support	Windows 11 Pro, Linux	Raspberry Pi OS, Linux	Ubuntu, Linux

Price	$0	$69.49	$224.99
Best Use Case	**High-performance data processing**	Low-power, simple data collection	AI-enhanced RF processing

In conclusion, the Intel NUC 13 NUC13ANHi7 Arena Canyon Mini PC is a powerful, compact, and low-power computing solution for processing data from a spectrum analyzer. Its powerful processor, large amount of memory, fast storage, and decent connectivity features all work together to make it a great platform for real-time signal processing use cases, while low power consumption and small form factor make it a great candidate for deployment in various environments, from lab setups to field-deployed RF monitoring stations.

3.9 Overview of Android Phone

When selecting a phone to obtain flight pattern data via Starlink, the key factors are whether the phone can connect to the Wi-Fi offered by Starlink or if it can send messages directly to satellites. While most phones today can access the internet through a Starlink terminal and router, only a few phones currently have the capability to send messages directly to satellites. This means that the best phone choice depends on the use case—whether one needs real-time access to information in remote areas with no Wi-Fi or just wants an internet-connected device for aviation tracking purposes.

Apple iPhone 14 and later models have been found to be good contenders for integration with Starlink. With iOS updates allowing direct-to-cell satellite messaging via Starlink, these phones offer basic satellite communication features, which should be expanded to voice and data soon. This feature makes iPhone 14, 15, and subsequent models an excellent choice for users who may work in regions with poor cellular coverage. However, even with this feature, complete data connectivity is still not possible using this method, so iPhones are still in need of a Wi-Fi connection to a Starlink router to receive real-time flight information. iPhones are also compatible with most aviation tracking apps, making them perfectly suited for data analysis and visualisation.

Similarly, Samsung Galaxy S23 and S24 series handsets provide an excellent choice. Samsung has already announced compatibility with Starlink's satellite text messaging service, starting with its flagship models like the Z Flip 6, Z Fold 6, and S24 Ultra. These handsets possess excellent processing power and big displays, which make them convenient for running aviation monitoring software. However, just like with the iPhone, full Starlink satellite data support continues to be in development, and Wi-Fi connectivity through a Starlink terminal remains a requirement for real-time flight tracking.

Google's Pixel 7, 8, and 9 series are also fantastic choices for receiving data via Starlink. While the phones do not yet have direct-to-cell Starlink support, they are constructed with high-performance processors and efficient AI-driven optimization, which enhance the visualization of data and real-time tracking. Pixel phones are also renowned for receiving timely

software updates and security patches, providing long-term support for aviation mapping and tracking software. Starlink satellite SMS support might be included in future updates, further enhancing remote connectivity.

For those customers who do not require direct-to-satellite capability but still need to get Starlink data, any Wi-Fi-capable contemporary smartphone will work when used with a Starlink router. So, older models of iPhones (such as the iPhone 12 and 13), budget Android phones (such as the Samsung A-series), and rugged phones (such as the CAT S75, including satellite texting on Inmarsat) can still provide decent data access. These machines can be linked to Starlink terminal Wi-Fi, allowing users to receive flight pattern updates as long as they are within the router's radius. This setup is ideal for users with fixed positions or with mobile workstations where Starlink Wi-Fi is readily available.

Table 3.8 Comparison of Mobile Device Technologies

Feature	**Samsung Galaxy S24 Ultra**	**Apple iPhone 15**	**Google Pixel 8 Pro**	**Older Android/iPhones**
Starlink Direct-to-Cell Support	**Yes (Texting now, voice/data future)**	Yes (Texting now, voice/data future)	No (Wi-Fi only)	No (Wi-Fi only)
Processor	Snapdragon 8 Gen 3	Apple A17 Pro	Google Tensor G3	Varies
RAM	12GB	6GB	12GB	4GB-8GB
Display	6.8” AMOLED (120Hz)	6.1” OLED (120Hz)	6.7” OLED (120Hz)	Varies
Battery Life	~14-16 hours	~13-15 hours	~12-14 hours	Lower than modern flagships
Starlink Wi-Fi Support	Yes	Yes	Yes	Yes

Price	$0	$729.99	$599.99	<$500

Of all the available smartphones, however, the Samsung Galaxy S24 series is best suited to accept flight pattern data through Starlink. Unlike the majority of smartphones, the S24 already accepts Starlink's direct-to-cell satellite messaging to maintain connectivity where other networks have no coverage. Also, with its high-performance processor, big high-resolution screen, and extended battery life, it is an optimal device for executing aviation tracking programs and handling real-time data in an efficient manner. Although iPhones also have satellite messaging and Google Pixel phones have robust AI-based processing, the S24's integration of Starlink capability, leading-edge hardware, and future-proofed connectivity makes it the most solid and capable solution for unbroken access to flight pattern information, anywhere.

3.10.1 Overview of Wired Communication Protocols (RF Device)

Naturally, since we have chosen a spectrum analyzer we will need that data to be transferred and processed locally so that we can transmit the data anywhere in the world. First, we will talk about the local communication protocols i.e. between the spectrum analyzer and our mini PC. Then we will do some discovery into how that data will be extracted for the sake of triangulation.

The communication protocol that we will explore is I2C. I2C works on a Master-Slave architecture. Meaning that our MCU will provide the clock and control the communication. The slave devices will simply respond to what is required of the master. It is a two wire communication system where there is a serial data line (SDA) and a serial clock line (SCL). It also operates at speeds from 100Khz to 3.4Mhz. I2C is a more low level technology that may be inside of the spectrum analyzer. It is typically used for things like interfacing with high resolution analog to digital converters. They can also be used for controlling a display which may be how our spectrum analyzer display is controlled. However, this is not the correct technology for our use case as it is a serial protocol. Meaning that it can only transmit one bit at a time and is severely limited by its clock speed. Even at the top of its frequency range it may only be able to transfer up to 400KB/s. This could be used if we had another low speed peripheral that is in the system. However, because we have extremely high resolution FFT data, I2C will not be sufficient for this specific use case.

The next technology we will explore is Universal Asynchronous Receiver-Transmitter or UART for short. It is a serial communication protocol that works asynchronously. Meaning that two devices can send and receive data without sharing a clock. It is a two wire protocol that has a transmit and receive data wire. UART is great because it is widely standardized and supported. Meaning that it is great for use with microcontrollers and embedded systems. However, this is also greatly limited by its speed, around 10 Mbp/s. This could be great for small peripherals or more simple low bandwidth things like I2C is. However, seeing as our spectrum analyzer is already assembled and has these small things already figured out it may not be the best for our use case. As previously mentioned, we need a much higher bandwidth in order to process our extremely complicated data.

Finally, we will talk about Universal Serial Bus. USB is without a doubt the best option for our use case. It is extremely versatile and robust. It is used in many different applications. It has the ability to not only send data between computers but also to charge or power things like our spectrum analyzer. It is a four pin system and has a power pin, a data in and out pin (Or data plus and minus), and finally a ground pin. In the early days of USB it had a data rate greater than I2C and UART at 12 Mbp/s. In more modern USB 3.0, it can transfer from 5 - 10 Gbp/s. Which absolutely trounces the previously mentioned protocols. As previously mentioned it can also provide power, with almost 5W of power which is more than enough for our spectrum analyzer. It is important to note that our spectrum analyzer connects to the computer through USB 3.0. This is great as we don't have to have a separate power source for the spectrum analyzer and antenna. Not only that, but it has extremely high data transfer rates which will be completely sufficient for our computer to process. We are likely going to be more compute power limited than we will be data rate limited. Our spectrum analyzer is an incredibly high precision tool and as a result needs only the best of the best data communication protocols which could only be USB.

Table 3.9 Comparison of Wired Protocols

Protocol	Power output	Bandwidth	Interface Compatible
I2C	1-5mW	400 KB/s	No
UART	3.3-150mW	10 Mbp/s	No
USB	.5-4.5W	5 - 10 Gbp/s	Yes

3.10.2 USB Hubs (Camera Array)

The main hurdle to deal with when using USB connections and cameras is bandwidth. To really make use of the OV9821's high frame rate, we'll need to avoid what is known as bottlenecking. This refers to the limitation of a communication method, like a USB connection, to handle data. Since each camera can have a max frame rate of 100 frames per second, connecting them all through a usb hub won't be enough. This was a problem we ran into during our midterm demo testing with cameras that only had 15 frames per second. With that said, the next section of research will be looking at different USB hubs that have the capability to handle the bandwidth of multiple camera connections.

The first USB hub we researched was the **Anker 10 port** USB 3.0 data hub. The total bandwidth of this device is 5 Gbps which is shared across all of its ports. It also features a 60 watt external power supply adapter which makes it a good choice for connections that require stable power distribution. Each USB 3.0 port can handle around 480 Mbps if all ports are used simultaneously. If we were to connect 3-4 cameras to the USB hub, each camera could get around 1.2 Gbps which would be suitable for our needs. The price of the USB hub is around $50 which makes it a little expensive for a connection hub, especially considering that when all connections are used, the full 5Gbps speed would not support all ports.

The second USB hub we researched was the **StarTech 4-Port** USB 3.0. This USB hub would be ideal for preliminary testing since it features only 4 ports and is less than other USB hubs out there that do the same thing. Like the Anker model, this hub has a total bandwidth of 5 Gbps which, when four cameras are in use, would likely cover the amount of bandwidth needed to run each camera. The main downside to this hub is the power. Since it doesn't have an external power adapter it runs the risk of not having sufficient power for each of the cameras. While the cameras may not consume a lot of power, this may be a risk we want to avoid.

The third USB hub we researched was also a StarTech brand hub, this time with 7 ports. The **StarTech 7-port** USB 3.0 hub is in all ways an upgrade from its 4-port counterpart. Like the 4-port, it also features a shared bandwidth of 5 Gbps but is better suited for multi camera use. Since, with this model, all ports would not be used at the same time with 5 cameras, the hub would be able to provide good bandwidth for each camera. It also features an external power adapter, listed at 36 watts, which should be plenty sufficient for our camera array setup. While the price is higher than the 4-port version, the additional features that this model provides reduces certain risks that we otherwise would've encountered.

The last USB hub we researched was the **Plugable 7-port** USB 3.0 hub. This USB hub also has 7 ports like the StarTech and features a total bandwidth of 5 Gbps. The external power adapter for this model however is listed at 60 watts which is more than sufficient for our case. This model would be the choice in the ideal case since it is more robust and industry level grade, however, it is not as readily accessible as the other models. Some of their other models are available on Amazon which would make the process of obtaining them much easier, whereas this model is ordered directly through the makers website, leaving uncertainty in the arrival time. Additionally, this model is more expensive but is definitely the best option.

Table 3.10 Comparison of USB Hubs

USB Hub	# of Ports	Bandwidth	External Power	Bandwidth for each port	Best for multi-camera use	Price
Plugable 7-Port	7 - USB 3.0	5 Gbps	60 W	Highest stability for each port	Best for multiple camera setup	~$50
StarTech 7-Port	7 - USB 3.0	5 Gbps	36 W	Higher stability for each port	Best if all ports are not used, 4-5 cameras	~$60
Anker 10-Port	10 - USB 3.0	5 Gbps	60 W	Lower bandwidth per port	Lots of ports, but when all are in used, performance	~$40

USB Hub	# of Ports	Bandwidth	External Power	Bandwidth for each port	Best for multi-camera use	Price
					issues	
StarTech 4-Port	4 - USB 3.0	5 Gbps	None	Lower bandwidth and possible power issue risk		~$35

Overall, the best choice for this project would be the Plugable 7-port USB hub. Its higher external power availability would ensure that power would not be an issue. Additionally, if we were to use around 5 cameras, each camera would have sufficient bandwidth and would have good stability. The alternative option, the StarTech 7-port, has the necessary bandwidth with slightly less power but would still work well for 4-5 cameras. The pricing is slightly higher, but we would be able to ensure a delivery time that is conducive to our success for senior design. The other options have either too many ports, in the case of the Anker, which reduces the bandwidth availability per port, or have a risk of not providing the necessary power for our cameras, in the case of the Startech 4-port. For this reason, the Plugable 7-port will most likely be our choice of USB hub. On Amazon, it is currently listed for $24 which makes it more affordable than the other choices and with more features.

If this project were to be done with a larger budget and more time, a USB hub may not be the best approach. Ideally, the cameras would be integrated through a dedicated computer and would be connected via a connection type that is able to transfer and stream high quality video. For our purposes, the USB hub allows us to connect the cameras to our laptop but in an ideal case, a dedicated device would be doing the computations and would not be limited by a USB connection. Additionally, with access to the proper resources, a more robust setup and organization of the system can lead to a more compact and efficient setup compared to the prototype setup we are making in this project.

3.11 Overview of Wireless Communication Protocols

Next, we have to learn about how we will transfer our data. These devices and prototypes will be placed remotely. As a result, it would be impractical to run hundreds of miles of wire around the world in order for these to all work properly. We will of course need wireless communications as they allow us to send the data across vast distances with extreme levels of accuracy.

The obvious solution would be to use cell service, 4G LTE, as it is mobile and is used all around the world. These transmitters also have extreme levels of power efficiency. Consuming from 1 - 2 watts on a phone. This is incredible for our system as we would have better fulfilled our power constraint. They also typically work from 700 Mhz to at most 2.6 Ghz. This lower frequency is

great because it allows long ranges with the trade off of data rates. The maximum data rate that is typically seen is from 100 - 300 Mbp/s which is more than enough speed and bandwidth for us to send a detection signal.They also have low latency so the data transmitted will be very accurate. These seem like the easy and best solution; However, one of the major issues is the location of these units. They are intentionally placed in extremely rural areas where there might not be a cell tower for hundreds of miles. This of course will be a huge problem for this communication protocol as the possibility of that data travelling will be none. This automatically eliminates this from our prototype as one of the constraints is the ability to work anywhere in the world.

Next, is Starlink which is a new service being offered. It works by using the large network of low orbit satellites to transmit and receive data. This is extremely useful for our use case as it allows us to communicate anywhere in the world. Which already has a leg up over the last solution. It operates in many different frequency ranges such as Ku-band (12 - 18 Ghz), Ka-band (26.5 - 40 Ghz), and V-band (40 - 75 Ghz). This is great as the more bands that we operate at, the more satellites we will be able to transfer data with. This is an extreme advantage as it has similar data rate speeds and bandwidth as 4G LTE. It also retains the low latency of 4G LTE which helps with the accuracy of the data. However, one of the major drawbacks of this technology is that it is power intensive. Typically anywhere from 10 - 100 Watts depending on the load. Also because of its high operating frequency it makes it more susceptible to interference from the rain which may lead to a large problem as the climate this will operate in is likely going to be South America. Starlink is really unbeatable when it comes to this use case and the power constraint will have to be worked around.

However, the FCC has recently started the approval of Starlink use in mobile phones. This would be a major driving point of our product as we can retain the low power consumption of 4G LTE while also having the ability to communicate anywhere in the world. This essentially turns any mobile phone into a satellite phone. This would also allow us to simplify our prototype by having all of the data processing done on a modern smartphone. Which has incredible computing power and power efficiency. Starlink is naturally the best communication protocol for our use case as we can use it anywhere in the world.

Unfortunately, despite the FCC approving of Starlink use in mobile phones, the service did not come out soon enough for it to be of use in the project. Starlink was still the best protocol for our project though, so we replaced the cellphone with a Starlink mini terminal, which was able to be easily setup and operational, though it had a higher power draw than a phone, it was still well within our power capabilities, drawing about 30 watts.

Table 3.11 Comparison of Wireless Communication Options

Protocol	Power Consumption	Download Speeds	Upload Speeds	Latency	Coverage
4G LTE	2 - 10 W	5 - 150 Mbp/s	1 - 50 Mbp/s	20 - 50ms	Near cell tower
Starlink	10 - 100 W	50 - 250 Mbp/s	10 - 40 Mbp/s	20 - 50ms	Worldwide

3.11.1 Evaluation for Wired and Wireless Selection

When it comes to how we are going to communicate between the processor and the spectrum analyzer, it seems as though USB is the best choice. It has the highest bandwidth and data rate which will be great for the accuracy and reliability of our system. As for how the data will be transferred from our local site to around the world, Starlink is by far the best option. It has extremely high data rates and works absolutely anywhere in the world which is a major driving aspect our sponsor wants.

3.12 Power requirements

Passive aircraft radar systems exploit available signals, such as commercial broadcast transmissions or cell networks, to detect and track aircraft without emitting their own radar pulses. This method significantly reduces power consumption compared to active radar systems, rendering them applicable for covert surveillance and cost-effective air traffic monitoring.

While passive radar systems have lower energy demands, they nonetheless require a highly optimized power supply for continuous signal processing, real-time analysis of data, and communication with other systems. Some of the key power demands are stable power supplies for high-performance computing, low-latency signal processing, and continuous data transmission. Power efficiency is also desirable for mobile or remote installations, where battery life, integration of renewable energy sources, and system optimization are of paramount concern.

The most power-consuming component of a passive radar system is the signal processing unit. Passive systems employ high-performance computing to process reflections, cancel out noise, and track aircraft movement in real-time. According to Griffiths & Baker (2005), passive radar processing requires significant computational power to support the correlation algorithms and the Doppler shift analysis that may consume from 50W to several hundred watts, depending on the complexity of the system and the number of receiving channels.

Passive radar systems make use of multiple receiver antennas to receive signals from different sources. The receivers' power consumption is a function of sensitivity and bandwidth. SDR receivers, which are now commonly used in passive radar, feature typical power consumption of between 5W and 50W per receiver module (Willis, 2011). While low-power consumption receivers improve energy efficiency, high signal-to-noise ratio (SNR) is necessary for making accurate detections.

Passive radar systems generate large amounts of data, which require large storage and communications bandwidth. In networked radar systems, transmission of processed data to centralized control points via satellite or wireless channels can be a large power consumer. Cherniakov (2008) reports that data transmission on long-range wireless channels can be 20W to 100W depending on the bandwidth and encryption requirements. Efficient data compression and low-power communication protocols are required to attain optimum power consumption.

The majority of passive radar installations are either remote or mobile and require ruggedized and power-efficient power supplies. Deployments on UAVs or on ground-based sensor stations often utilize battery power, fuel cells, or other forms of alternative energy sources like solar panels. It has been researched and proven that solar-powered passive radar systems with energy storage can operate with a total power consumption as low as 200W (Griffiths & Long, 2013).

Nonetheless, energy storage solutions need to be optimized for round-the-clock operation, with redundancy and resilience to withstand power fluctuations.

The power needs of an Aventas AX-37A log-periodic dipole antenna and a Signal Hound BB60D spectrum analyzer must be considered thoroughly to be able to continue running uninterruptedly and efficiently within a passive aircraft radar system. The Aventas AX-37A log-periodic dipole antenna is a passively operated receive antenna and does not contain an external power source for minimal use. But since it has an LNA that acts to enhance reception of the signal, it will require additional power, typically between 5V to 12V DC, depending on the type of amplifier model. The power consumption of an LNA is typically low, typically between 0.1W to 1W, but will have to be added to the overall system power consumption.

The Signal Hound BB60D spectrum analyzer, which itself processes and analyzes the incoming signals through the antenna, has more particular power requirements. It is powered through USB 3.0 and requires a stable 5V DC power source, at around 4.5W (or roughly 0.9A of current). Though this power is relatively low compared to much larger spectrum analyzers, it still requires a stable and consistent power supply so as not to disrupt the signal processing. Because the BB60D is powered via USB, it can be supplied from a connected laptop, but for extended field operation—especially under off-grid situations—an external power station or battery bank with USB 3.0 output is recommended.

For a complete power solution, a portable power station with USB and DC outputs is ideal for use in the field. A minimum capacity power supply of 300Wh can support several hours of operation before being recharged. For continuous usage over extended time, solar panel hardware to provide a minimum of 100W can support continuous operation. With a guarantee of stable, noise-free power supply, BB60D and AX-37A antenna would support effective operation and thus are efficient for passive radar aircraft use.

Additionally, the power requirements of the mini PC must be taken into account. The Intel NUC 13 NUC13ANHi7 Arena Canyon Mini PC is a tiny yet efficient mini PC intended for performance-intensive activities like data processing, media streaming, and even passive radar signal analysis. Its power requirements are supported by its Intel Core i7-1360P processor, 16GB of RAM, and 512GB SSD, all of which are factors in its power usage.

This model supports a 19V DC power supply, rated to deliver up to approximately 90W in full load configurations. In typical usage, e.g., web surfing or light calculations, power consumption will typically fall between 10W and 30W. Under extreme computational processing—e.g., signal processing for a passive radar system—it can draw around 50W to 75W depending on CPU loading and peripheral loading.

For off-grid or mobile applications, such as field deployment with a passive radar system, a portable power station with a 19V DC output or its equivalent AC inverter will be required. A 300Wh to 500Wh battery power station would provide several hours of operation time. For round-the-clock power supply, especially in remote areas, it is best to couple the power station with a 100W or larger solar panel. Also, enabling power-saving options on Windows 11 Pro and BIOS—such as reducing screen brightness, disabling unwanted background tasks, and making use of Intel's power-saving modes—can help extend battery life and improve power efficiency.

Table 3.12 Power Requirements of Components

Component	Power Consumption	Voltage Requirements	Power Source Compatibility
Aventas AX-37A Log Periodic Dipole Antenna	Passive (No power required)	9V	Direct connection to receiver
Signal Hound BB60D Spectrum Analyzer	~4.5W	5V DC (via USB 3.0)	USB 3.0 port from PC or external power source
Intel NUC 13 (NUC13ANHi7)	~90W (peak), ~10-30W (idle)	19V DC	AC adapter (120V/240V), battery via inverter
Starlink Mini Terminal	~25-30W	12V-48V	AC adapter(120V)

All portable power stations have special advantages and disadvantages, which make them suitable for different applications. For the purposes of this prototype, four possible power supplies have been selected to consider. The Westinghouse 100-Watt Solar Powered Portable Power Station, EF ECOFLOW RIVER 2 Pro, EF ECOFLOW DELTA Max 2000, and Jackery Explorer 300.

The Westinghouse 100-Watt Solar Powered Portable Power Station is highly portable with a weight of just 3.75 lbs, one of the lightest available in the market. It is an affordable and eco-friendly solution for charging small devices such as smartphones, tablets, and LED lights. It also comes with additional versatility with multiple output ports such as USB-A, USB-C, and AC. However, its low battery pack capacity of 155Wh and tiny power output of 150W make it unable to supply power to larger electronics or high-wattage appliances. Moreover, its six-hour charging duration from a wall outlet is inconvenient.

EF ECOFLOW RIVER 2 Pro stands out with its high-speed charging ability, charging up to full capacity in just 70 minutes, one of the fastest for a portable power station. Its 768Wh LiFePO4 battery offers extended lifespan and superior safety level, making it suitable for off-grid uses. At 800W of output and 1600W of X-Boost capability, it can power small kitchen devices and tools, and its multiple AC, USB, and DC outlets support dual device charging. Though, at 17.2 lbs, it's

far too heavy for smaller power stations and not very portable. Also, its mid-size capacity may not be sufficient for backup power for extended periods, and it is more expensive than some of its competitors in its class.

For high-capacity, the EF ECOFLOW DELTA Max 2000 is an ideal choice with its massive 2016Wh expandable battery, perfect for powering high-wattage devices. It boasts a stunning 2400W output capacity with 5000W surge and can hence support heavy use appliances such as fridges and power tools. The 1.8-hour fast charging maintains a minimal downtime period, whereas diverse charging provisions are an additional convenience under mixed situations. However, powerful as they are, they are heavy, weighing at 48 lbs, it is the easiest to maneuver on-the-go. In addition, it is one of the most expensive power stations, and its enormous capacity can be redundant for low power devices.

The Jackery Explorer 300 strikes the perfect balance between portability and mid-power. With a weight of just 7.1 lbs, it is extremely easy to transport and ideal for camping, traveling, and outdoor recreation. Its 2.5-hour fast charging using AC and USB-C is efficient for its size, and its 293Wh battery bank carries more energy than the Westinghouse model but in a smaller package. The variety of output ports, such as USB-C, USB-A, AC, and DC, provides compatibility with multiple devices. It is not ideal for charging large electronics, however, with its modest 300W (500W surge) power output. Also, its 100W solar input capacity supports slow solar charging and, while smaller capacity, is more expensive than some of the other competitors in the same category.

Much of the reason for the differences between the power supplies is the type of battery that they utilize, with the Westinghouse 100W and the Jackery Explorer 300 both using lithium ion batteries, while the EF ECOFLOW RIVER 2 Pro and the EF ECOFLOW DELTA Max 2000 use LiFePO4 batteries. In selecting a source of energy for a passive airborne radar system, the criteria are energy efficiency, lifespan, weight, safety, and suitability to the environment. Out of the two battery options, Lithium-Ion (Li-Ion) and Lithium Iron Phosphate (LiFePO4), there exist respective advantages and disadvantages in terms of utilization depending on the radar system's application demand.

Lithium-ion batteries possess greater energy density, typically 150-250 Wh/kg, and can facilitate a light and slim design. They are therefore suitable for applications in mobile or UAV-based radar systems, where weight saving is essential (Whittingham, 2004). They also provide quicker charging than LiFePO4 batteries, which makes them suitable where rapid replenishment of energy is essential. Additionally, lithium-ion batteries are readily available and used in numerous applications, thus easier to implement. However, these benefits come with some drawbacks. Lithium-ion batteries have a short lifespan, typically lasting 500-1,000 charge cycles, meaning they degrade quicker with extensive use (Chaykina et al., 2019). They also come with thermal instability risks, being more prone to overheating and thermal runaway, which can be a safety hazard in extreme conditions. Furthermore, lithium-ion batteries feature a shallow depth of discharge, i.e., they require advanced battery management systems (BMS) to prevent deep wear and degradation.

Compared to them, LiFePO4 batteries offer much increased life, i.e., 2,000–5,000 charge cycles, and can be used in constant or prolonged passive radar use (Kumar & Jain, 2022). They provide

enhanced thermal stability and safety and are resistant to overheating, hence are capable of working reliably even at high temperatures or under harsh environments. Stable discharge voltage is yet another key advantage, and this ensures the supply of a constant power source to radar processing apparatus and receivers and minimizes downtime in performance. LiFePO4 batteries do have their flaws, however. They have a reduced energy density of 90-160 Wh/kg and weigh and are heavier and bulkier than lithium-ion equivalents, being possibly a setback to UAV-based or highly mobile radar systems (Wang et al., 2021). Also, LiFePO4 batteries charge longer, which would limit their usage in situations calling for rapid refilling of energy. Their higher initial cost can also be a budget constraint for cash-starved organizations, though their long lifespan offsets long-term expenditure.

The choice between lithium-ion and LiFePO4 batteries is finally determined by the specific needs of the passive radar system. For permanent or fixed installations, LiFePO4 batteries are the better choice since they offer a longer life, consistent performance, and better safety. However, lithium-ion batteries are preferable for mobile, UAV-based, or portable radar systems due to their light weight and higher energy density, which allows for higher mobility. If reliability and lifespan are the most important concerns, LiFePO4 is the preferred choice, but if weight and size constraints are the most important factors, lithium-ion batteries are still a good choice.

For our purposes, going with a model that has a lithium-ion battery makes more sense because the prototype we make needs to be more mobile, and how long it lasts on its own battery-wise isn’t as important. Though an official release of the product would likely benefit more from a LiFePO4 type battery due to its long lasting nature and the product’s stationary nature.

The Westinghouse 100-Watt model is compact, eco-friendly solar charging but has limited solar input capacity of around 30-40W. This means solar panel charging can be sluggish, especially in low sunlight conditions. While third-party solar panel compatible, it may be adapter-limited, which could decrease convenience. With its small 155Wh battery, solar charging can remain a viable alternative for low-power consumers.

The RIVER 2 Pro has premium solar charging technology, with an input of up to 220W solar. This allows for a full charge in as little as 4 hours of optimal sunlight exposure, making it an excellent choice for off-grid applications. Not only is it compatible with EcoFlow solar panels, but it's also compatible with third-party solar panels, hence offering users freedom. The LiFePO4 battery also features a flat charge-discharge profile, making the efficiency of long-term solar charging better. Its MPPT (Maximum Power Point Tracking) controller ensures maximum efficiency in solar energy conversion under changing sunlight conditions.

The DELTA Max 2000 has the highest capacity of solar input among the compared power stations with the ability to handle a maximum of 800W solar input. It supports high-speed solar charging, and charging its 2016Wh battery to full capacity is feasible in 2.5–3 hours using multiple high-efficiency solar panels. This renders it one of the best options for home backup and off-grid power solutions that can power high-energy appliances without relying on the grid. Like the RIVER 2 Pro, it features an MPPT controller, which optimizes solar efficiency. However, to utilize it at its best, several high-wattage solar panels will be required, which is costly and takes up a lot of setup space.

Jackery Explorer 300 is moderately powered solar, with a maximum input of 100W. It will charge fully in about 5 hours using Jackery's SolarSaga 100W panel under perfect sun conditions. Not as powerful as the EcoFlow models, it supports efficient off-grid charging for low to medium-powered devices such as laptops, devices, and cameras. But its solar input limit guarantees that charging can take much longer if sunlight conditions are not ideal.

Solar power capabilities are very important for this prototype because the final project will be reliant on the ability to be self-sufficient when it comes to power. They are designed to stay in one place for incredibly long periods of time without needing to be actively maintained, meaning that they need to have batteries that don't run out, and more specifically, the ability to recharge themselves without anybody needing to manually recharge it. Though the low-power nature of the device means that the solar capabilities don't need to be very powerful, rather the device should be able to sustain itself off of a low-speed charge via the sun, so sacrificing solar power speed for cost will be worth it.

One of the largest determining factors between the power supplies is of course the cost. Since the main goal of this prototype and eventual full product is to introduce a very low-cost passive radar into the market of similar quality, the ultimate goal is to hit that low cost, high affordability threshold. Low costs are produced by choosing components and necessary pieces of the product that are similar quality to others on the market while maintaining a low price, this means that the lowest cost option that still meets all of the requirements is the ideal product.

Table 3.13 Comparison of Power Supply Technologies

Feature	**Westinghouse 100W**	**EF ECOFLOW RIVER 2 Pro**	**EF ECOFLOW DELTA Max 2000**	**Jackery Explorer 300**
Battery Type	Lithium-Ion	LiFePO4	LiFePO4	Lithium-Ion
Battery Capacity	155Wh	768Wh	2016Wh (Expandable)	293Wh
Power Output (Rated/Surge)	150W / Not specified	800W (1600W X-Boost)	2400W / 5000W	300W / 500W
Solar Input	Yes	220W max	800W max	100W max
Weight	3.75 lbs (Lightest)	17.2 lbs	48 lbs (Heaviest)	7.1 lbs

Price	$0(cheapest)	$379	$1299(most expensive)	$259
Dimensions (inches)	7.4 x 6.6 x 3.7	10.6 x 10.2 x 8.9	19.6 x 9.5 x 12	9.1 x 5.2 x 7.8
Best Use Case	Small devices, ultra-light travel	Mid-range power needs, tools, RVs	Home backup, high-power appliances	Camping, small device backup

Lastly, choosing the most appropriate power station is a personal need issue. The Westinghouse model is appropriate for light travel and low power demands, the RIVER 2 Pro offers power and portability, the DELTA Max 2000 is appropriate for heavy use, and the Jackery Explorer 300 is an excellent middle-of-the-road option.

Westinghouse 100-Watt Solar Powered Portable Power Station is the most suitable power supply for a passive aircraft radar system due to its portability, lower cost, and sufficient power output for low-power radar functionality. Passive radars, unlike active radars utilizing active transmissions, require low to moderate power consumption to drive the signal processing, data analysis, and communications hardware. With its 155Wh lithium-ion battery and pure sine wave output, the Westinghouse power station provides stable and clean power, ensuring solid performance for radar equipment without introducing electrical noise or instability.

One of the finest features of this power station is that it is portable, with a weight of only 3.75 lbs, which is ideal for mobile or field-deployed passive radar installations. Since passive radar systems are typically used for covert surveillance or mobile reconnaissance, a lightweight and easily transportable power supply is an excellent advantage. Its several output ports also allow for powering laptops, signal processors, and communications equipment at the same time, which are essentials for running radar.

The Westinghouse 100-Watt's solar readiness also enhances its suitability for passive radar use. With its ability to charge through solar panels, it can be topped off continuously even in the middle of nowhere, with a guarantee of long-term performance without the requirement for traditional power sources. While its 150W peak power delivery may not be sufficient to support power-intensive radar setups, it is more than adequate for low-power passive radar implementations where efficiency is given more priority than raw power.

While the Westinghouse 100-Watt represents the most pragmatic choice for a passive aircraft radar, the other power supplies all possess drawbacks which make them less ideal for this specific application. Passive radar systems tend to require low to moderate power consumption, with an emphasis on portability, efficiency, and affordability being more important than redundant energy reserves. The other power stations all have different advantages, but also have drawbacks that make them less suitable for passive radar applications.

The EF ECOFLOW RIVER 2 Pro, for instance, has a significantly larger battery capacity (768Wh) and faster charging rates compared to the Westinghouse 100-Watt. With the ability to charge from 0-100% in just 70 minutes, it is a high-power option for customers who need a reliable, high-capacity power station. In addition, its LiFePO4 battery lasts much longer than traditional lithium-ion batteries, making it a long-term option for long-term use. These advantages come with trade-offs that make it less ideal for passive radar systems. Its larger and heavier form makes it less portable than the Westinghouse lightweight model, an important factor for radar installations requiring mobility. Aside from this, the RIVER 2 Pro is also much more expensive, routinely retailing at nearly three to four times the price of the Westinghouse offering. Because a passive radar will not require such capacity, it is superfluous to spend more money on a higher capacity, larger power station.

Similarly, the EF ECOFLOW DELTA Max 2000 is even more overkill for the needs of passive radar, boasting a 2016Wh battery capacity and expandable power. While this makes it an excellent choice for home backup power or heavy off-grid applications, it is too heavy and expensive for a passive radar system, where efficiency and portability are more important than raw power. The DELTA Max is one of the heaviest and largest units in this comparison, so it's not suitable for field deployments where mobility is required. Additionally, its large power output is overkill for passive radar use, where power consumption is dominated by low-power signal processing and computing rather than powering high-wattage equipment. The DELTA Max's fast recharge time (1.8 hours full charge) is a nice feature to have but is not required here, as passive radars do not drain power as quickly as power-hungry systems. The DELTA Max's price is also exorbitantly high, making it a poor investment when the same operational needs can be met with a much smaller power station.

The Jackery Explorer 300, while a cheaper mid-range option, still doesn't give the same cost-to-performance ratio as the Westinghouse 100-Watt. It also has a higher battery capacity (293Wh), which might initially seem to be advantageous for extended radar use, but also means longer charge times and additional weight compared to the Westinghouse unit. Because a passive radar system does not require continuous high-wattage draw, the additional battery life is not a significant advantage. Moreover, the solar input limitations of the Jackery (100W max) mean that it doesn't charge as well as newer units, such as the EF ECOFLOW RIVER 2 Pro, and is also more expensive than the Westinghouse. The Explorer 300 also weighs more, making it less portable for use in the field where smaller, lighter power sources are preferable.

Ultimately, all three alternative power stations provide more power than necessary for a passive aircraft radar at the cost of portability and price. The EF ECOFLOW RIVER 2 Pro and DELTA Max 2000 are significantly larger, heavier, and more expensive than necessary for a low-power portable radar system. While the Jackery Explorer 300 attempts to achieve a greater balance, it still fails to be as compact or affordable as the Westinghouse 100-Watt. As passive radar installations prefer light, portable, and low-cost solutions, the Westinghouse 100-Watt remains the better choice, offering just enough power in a compact, affordable, and solar-friendly form factor.

Furthermore, the affordability of the Westinghouse 100-Watt makes it a financially feasible choice compared to big power stations like the EcoFlow DELTA Max 2000 or Jackery Explorer 300, which feature much more power than is needed for passive radar demands. This makes it an

ideal choice for economically limited deployments with a high degree of assured and stable power for important radar components. Lastly, the combination of portability, solar functionality, constant power output, and affordability makes the Westinghouse 100-Watt the perfect choice to supply power to a passive aircraft radar system in various operating conditions.

3.13 Overview of Ruggedizing Components

The nature of our project requires our components to be tough and ready for the environment. Florida is very similar to that of South America and it's hot and humid climate. Not only that, but the enclosure has to be sealed as small animals or bugs will make a home. As a result, we will need to build our prototype to withstand these elements. When we build our enclosure we will have to also tackle the heat that is generated from our unit and keep everything working optimally.

First, we will discuss the heat management that needs to happen. The power requirements will play directly into this as the heat will accelerate the degradation of our battery. To combat this, we will have to keep them cool and also consider this when we design the final prototype. For our use cases where we will be building prototypes to test and create an algorithm this is something that will be considered later. However, we will have to consider how to keep the prototyping components cool. Which leads into our enclosure.

There are two major ways that we can keep it cool. The first is to passively cool all of the components. This includes making things such as heat sinks or making a large thermal mass that will allow the heat to be absorbed and spread out. The major benefits of this method is that we don't have any powered system that drives up our already high power requirements. Not only that, but it will be extremely cheap as making a large heat sink out of metal also improves the rigidity and strength of the enclosure. Of course this will also come with the cost of weight. Ultimately, this is a good option as it doesn't add any power requirements and allows us to keep the cost low.

This also leads into heat pipes which we can use to extract the heat from the main components to an external heat sink. This works passively as it is a loop filled with a vapor that will evaporate at an extremely low temperature. Evaporation is a cooling process and as a result the system will also be cooler. It also moves heat away from the system. Of course the major drawback of this is the fact that it will be larger and also cost a lot if not more than a powered cooler. The heat pipes will also use the heat sink.

The next way to keep it cool is to have a powered cooler. This could be something like PWM controlled fans. Using a PWM fan means that we still need a heat sink. Not only that it adds in the necessity for more power or fan controller. The fan could also be set up to work with the NUC that we will be implementing for the prototypes. This will provide massively improved cooling over the passive setup. In return the components will work for longer and at more optimal performance.

All of these systems will need an air filter to keep bugs and such out to maintain effectiveness. Things such as the PWM fan may require more dust filters as there is more surface area that would allow bugs to enter.

Table 3.14 Comparison of Ruggedizing Components

Component	Ruggedness	Complexity	Effectiveness	Cost
Heat Sink	Highest	Lowest	Lowest	$27.25
Heat Pipes	Medium	Medium	Medium	$60.00
PWM Fan	Lowest	Highest	Highest	$15.95
Air Filter	Medium	Low	High	9.99

Evaluation for Selection

The reason that we would go with the PWM fan is because it is the cheapest option. CPU coolers have extremely high reliability with thousands of hours worth of run time. Not only that, but it will work the hardest and the best for our use case. Regardless the enclosure will have to have an opening of some sort for the air to get out. With this we can make the most effective use of our enclosure space and allow it to be an exit or entrance for more air to come in. Not only that, but the air filter will do a good job of keeping small critters out.

Next, we need to talk about waterproofing. The enclosure has to be waterproofed in order for the electronics to not get damaged. One of the easiest ways to stop this is by placement. Not allowing water to fall into the enclosure. We can also include small drain holes or redundancy in the enclosure by keeping the electronics that are sensitive in their own specific enclosure.

One thing that we could also do is apply silicone coating or an acrylic coating. This will be sprayed on to all of the sensitive electronics. This can also be sprayed on the antenna PCB or anything else that may need it. For example, we could pull apart the spectrum analyzer and coat all of its components. By spraying silicone on all of the components we are essentially coating it and making it humidity proof. Silicone coating is also extremely cheap and will not become brittle like acrylic coating will. It offers better longevity and the type of weather resistance that our system needs.

There are other options such as Urethane coating. Urethane is extremely strong and is more of a permanent thing. It not only waterproofs it, but makes the electronics more rigid. This would be good if we needed more than just humidity coating. For example if this were in the middle of the desert, the humidity isn't as big an issue as the dust may be. By having coating on it, it ensures that all of the sensitive electronics are protected.

Table 3.15 Comparison of Material

Coating	Humidity	Dust	Heat	Vibration	Longevity	Cost

	Resistance	Resistance	Resistance	Resistance		
Acrylic	Lowest	Lowest	Lowest	Lowest	Lowest	$17.98
Silicone	Highest	Medium	Highest	Highest	Highest	$18.06
Urethane	Medium	Highest	Medium	Medium	Medium	$41.20

Evaluation for Selection

As seen from the table above, silicone has the least trade offs for our use case. It offers most importantly humidity resistance. It does the best at sealing all of our sensitive components from the wetness of the environment. Not only that, but it is also far cheaper than Urethane and slightly more expensive than acrylic. It is by far the best option for our use case.

3.14 Programming Languages (Camera Array)

Another consideration we will have to make is which programming language or packages would help us to do super resolution. Since super resolution involves digital image processing, some basic machine learning, and algorithms that involve convolution, we'll have to find the programming language and any packages that would be best suited for that need. Since our part of the project, as PSE students, is only focused on creating a super resolution image, we won't have to worry as much about detection algorithms.

The clear frontrunner for our project was Python. Python features extensive libraries with image processing tools, and has lots of tools that can assist with deep learning, which is used in computer vision. Additionally, it is easy to integrate with OpenCV, a computer vision package, which is something we ended up using during our midterm demo. Python is also generally listed as one of the best languages for super resolution, both for AI based or classical interpolation based super resolution. As mentioned, one of the packages we would need would be OpenCV which is for general image processing. We would also need Numpy, this would allow us to do our mathematics more efficiently, especially when dealing with array and matrix operations. Overall, Python worked well for our midterm demo and may be the best option for our future iterations.

The next programming language we researched was C++. This programming language emerged as a possible option because of its real-time applications and its general speed compared to Python. This language, like Python, also works well with OpenCV and has similar libraries to Python that allow for fast matrix operations and good image processing. While this is a strong option, our experience with C++ is very minimal and using it would require a lot more time to get right. For this reason, C++ would not be great for our project.

The last programming language we looked at is one that PSE students are very familiar with, MATLAB. MATLAB features strong built-in image processing and interpolation tools which are vital to our project. Its main drawback is in its use case. Unlike Python or C++, MATLAB is better suited for algorithm development rather than real-time development. However, MATLABs strength in image processing could be good as a supplemental tool if we were to use Python.

Since Python would be in charge of running the cameras, to lessen the load or make computations slightly faster, we could use MATLAB to do the image processing while Python focuses on capturing usable images. With this system, our project's overall efficiency would be much faster, and we'd be using two programming languages we are very comfortable with. For these reasons, we will opt to use Python and MATLAB in tandem to produce the best results possible as quickly as possible.

For applications beyond this project, a coding language that is less elementary would serve best for the super resolution code. Since this type of algorithm requires an immense amount of computing power depending on which type of super resolution technique is chosen, a more robust coding software would serve best for this application. For our purposes, Python has enough of the tools that we need with a low entry barrier to use but for a more realistic project, like if this was in the government for example, more drawn out code and computers with large computational power would be preferred.

3.15 Super Resolution Techniques Analysis

Now that our programming language is established, we'll need to look into different ways to actually apply super resolution techniques. First, we have to understand what super resolution actually means. Super resolution refers to any technique that works to enhance the resolution of an image beyond the native capabilities of the imaging system. It mainly works by using one or multiple low resolution images and attempts to reconstruct a high resolution counterpart from the information of the low resolution images. The best example of how this would work in an ideal world actually appears in Hollywood. In some crime shows, it is very common for there to be a low resolution image that is then zoomed in and enhanced to reveal extraordinary detail about whatever scene is being shown. While that level of super resolution is certainly not realistic, it shows how powerful this type of technology could be and its potential applications.

So, how is it done? Super resolution leverages the idea that any low resolution that lacks detail can be reconstructed with multiple images or AI based methods. With that said, we can explain super resolution as a process of reconstructing a high resolution image, typically denoted as *I_HR*, from low resolution images, *I_LR*. This process involves signal reconstruction, image alignment, and learning based enhancements. Mathematically, we can model the process as the following:

$$I_{LR} = D(B(I_{HR})) + N$$

In this equation, we model the degradation process, or the process of getting a low resolution image from its high resolution counterpart, as a blurring function B, due to optics, sensors, or motion, and a downsampling function D, representing the loss of spatial resolution, onto the high resolution image *I_HR*. After those processes are applied to the high resolution image, there is also a compensation for noise N, due to the sensor noise or compression artifacts for example. Now that this process of degradation has been established, our goal when we do super resolution is to reverse this degradation and estimate the high resolution image from its low resolution counterpart.

While mathematically it appears very simple, there are many different approaches and methods that all have varying degrees of accuracy. Since it is generally impossible to get a perfect high resolution counterpart, often called the ground truth high resolution image, defining what success is when it comes to our context is important. For our purposes, we will define accuracy in 2 ways. The first way is just through general appearance to the naked eye, essentially answering the question of whether or not the image looks more clear or crisp. The other way we will define our accuracy will be through edge detection programs. While making the image look better is important, the edge detection protocol is how we prove that our concept and method of application is actually useful. Typically, when you want to detect if an aircraft or drone is in the sky above you from a picture, the image usually does not possess the clarity or resolution to effectively define edges, such as the wing of a plane, that a detection software can see. By defining our success as seeing if an edge detection software can detect more edges in the high resolution image compared to the low resolution counterpart, we will be able to claim that our approach is actually viable for the proposed application. The naked eye test is present as well to make sure that the images are actually representative of what the scene is showing. One method used to measure the quality of a super resolution image known as mean squared error shows why including the naked eye test is important. In the image below, all the super resolution counterparts have equal mean squared error values but the image at the top is clearly the best and most convincing high resolution image out of the group.

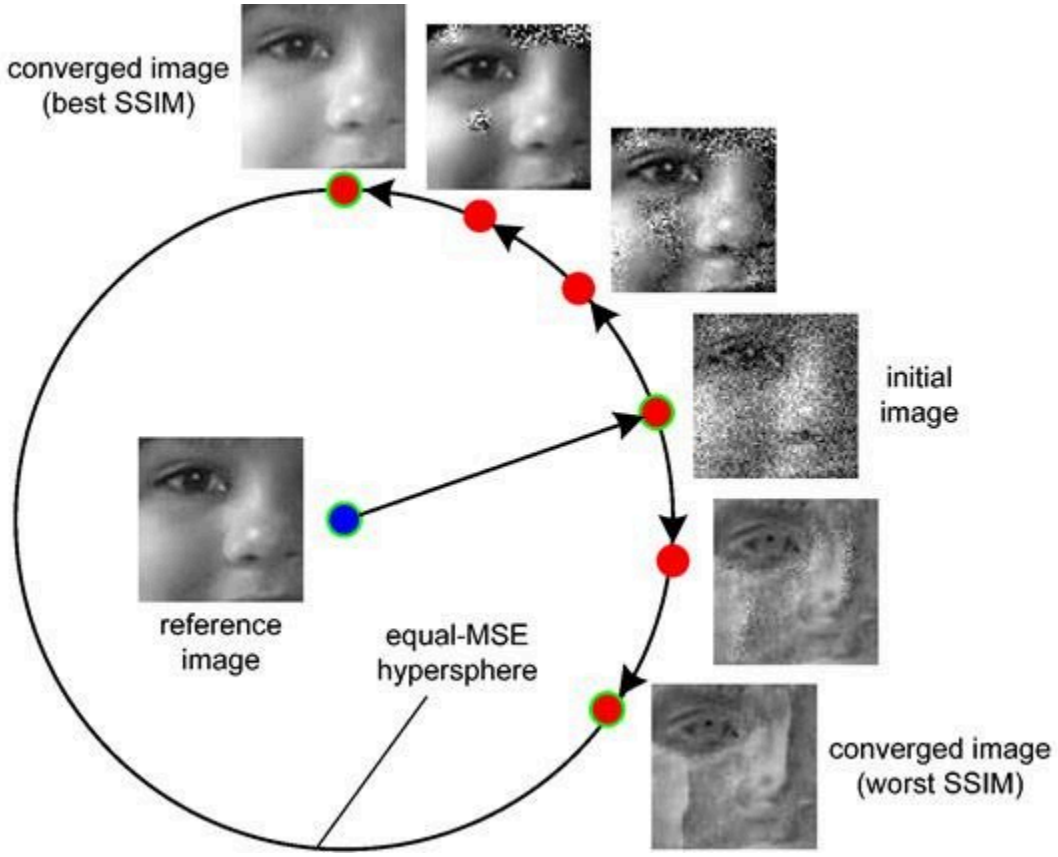


Figure 3.9 *Mean squared error reliability example (ResearchGate)*

Now that our success condition is established, let's look at the different methods of doing super resolution. One of the approaches to this problem is the interpolation based strategy. This approach tends to be one of the simplest super resolution methods and it works by estimating missing pixel values. While these methods are computationally efficient, they are unable to add any new information than what is already present in the image. Under this umbrella, there are three main types of interpolations: nearest-neighbor, bilinear, and bicubic. The nearest neighbor interpolation method uses the nearest pixel value and copies it to the pixel it is attempting to estimate. This usually results in blocky or pixelated images. Bilinear interpolation uses a weighted average of the four closest pixels to the pixel it is attempting to estimate. While similar to the nearest neighbor, this method produces a smoother image but generally lacks detail. The last method, bicubic interpolation, works similarly to bilinear but uses a weighted average of the nearest 16 pixels. This results in the smoothes images of the three methods but is still limited when it comes to restoring high frequency details.

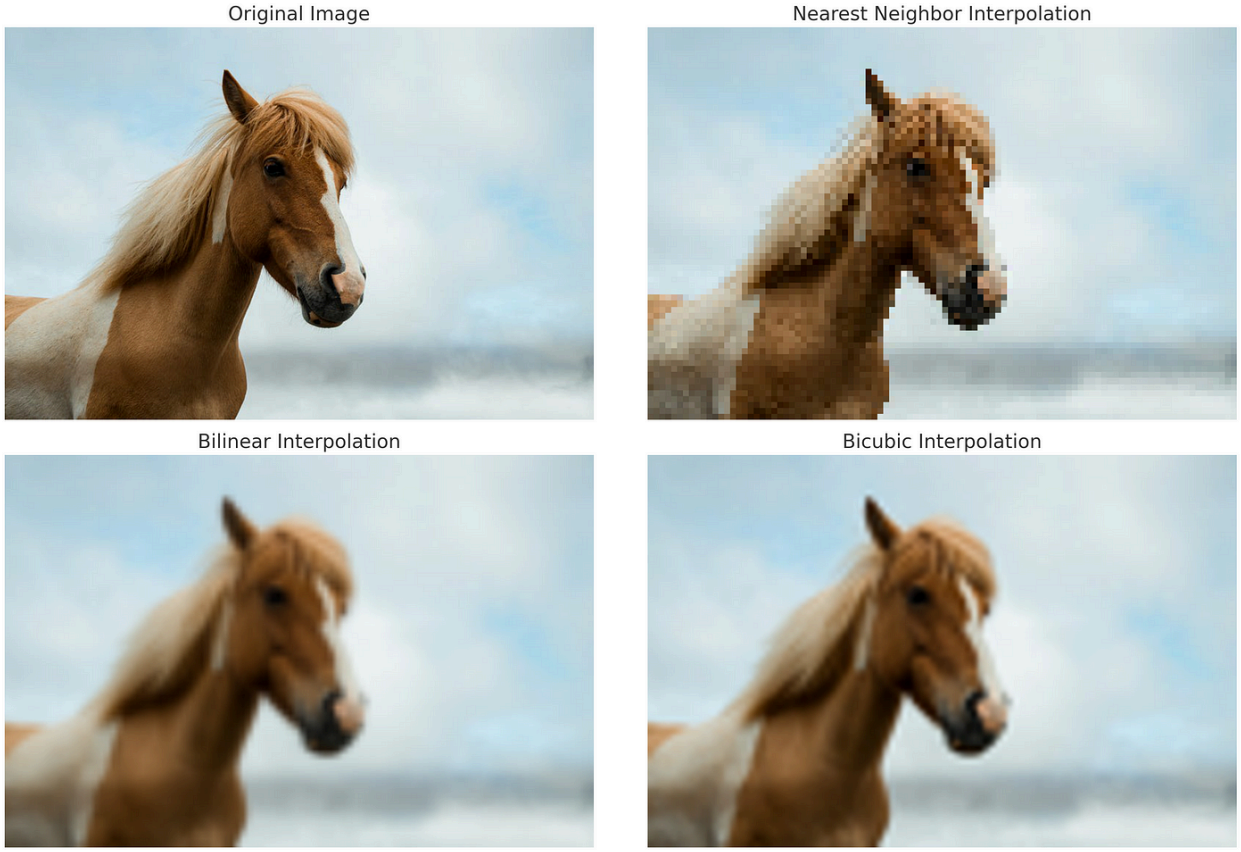


Figure 3.10 *Nearest neighbor vs. Bilinear vs. Bicubic interpolation (Medium)*

Above is an image that shows how these three methods compare when looking at the same scene. From the image above, it is clear to see that the more computational power that is being used, the better the result is. While these methods are not perfect, they are able to act on a single image to produce a higher resolution counterpart. The main limitations are of course the clarity of the image and the inability to capture finer details.

The next approach we'll look at is the multi-image super resolution approach or MISR. MISR reconstructs a high resolution image from multiple low resolution images captured from slightly different viewpoints or at different times. Before this can be done though, the images must be properly aligned. This involves estimating the transformation T that would map one image onto another. In this context, the transformation T can represent a translation, rotation, scaling, shearing, or homography which is perspective correction. This transformation can be done through feature based techniques, which looks for similar features in the images and attempts to align them, or through optical flow techniques, which estimates pixel displacements. Both techniques work best when the imaging devices are close to each other or are capturing a far scene. If those conditions are not met, there is a high risk of distortion appearing in the final image, an issue we dealt with during our midterm demo.

Once the images are aligned, the reconstruction process can commence. The two main techniques to do this are back projection and Bayesian estimation. Back projection is a common imaging technique that is often seen in tomography. It involves sending a signal from different perspectives to image something more clearly. In the case of MISR, back projection is an iterative algorithm that refines a high resolution image estimate by comparing it to the original low resolution image and correcting errors. The key to this technique is that if you downscale the estimated high resolution image, it should match the observed low resolution images. This method, therefore, ensures that the high resolution image is consistent with the input low resolution images and it corrects errors rather than blindly sharpening or upscaling parts of the scene. This technique works well for multi frame super resolution since different images are able to provide different perspectives of the scene but is limited when the image is large or is severely degraded. The algorithm is best depicted through a visual representation which can be seen below in a more advanced way, as proposed in a paper titled Deep Back-Projection Networks for Super Resolution.

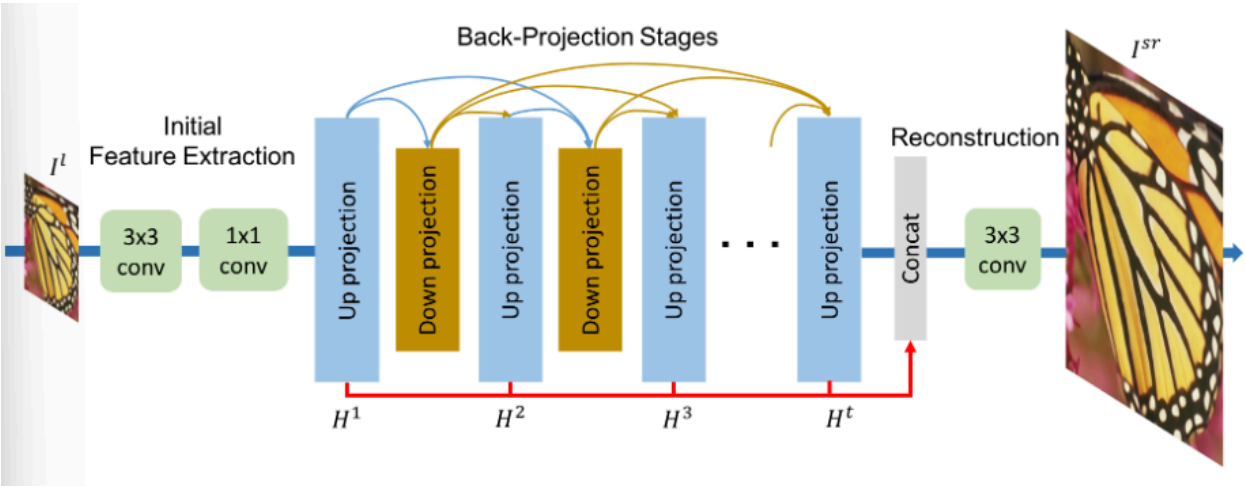


Figure 3.11 *Back projection algorithm visualization (Haris, Shakhnarovich, Ukita)*

The other technique, Bayesian super resolution, uses probabilistic models to infer the high resolution image by maximizing the posterior probability. In more simple terms, it maximizes the probability of the high resolution image it produces being the best image given the observed low resolution inputs.

$$P(I_{HR}|I_{LR}) \propto P(I_{LR}|I_{HR})P(I_{HR})$$

The equation above is known as Bayes' theorem and we want to find P(I_HR|I_LR), which is the posterior probability. Here, P(I_LR|I_HR) is how likely the low resolution images are given the high resolution image, and P(I_HR) is the prior which is what we assume about natural images such as the smoothness. While the math is a little complex, the main idea is that Bayesian super resolution is a technique that incorporates prior knowledge which makes it more robust to noise and missing details. It works well in low-light or noisy environments but is computationally expensive and intensive. However, it does yield more stable results than back projection alone.

The last group of super resolution approaches are defined by their machine learning based approach. This approach is the way modern super resolution techniques are done. It involves deep learning, particularly convolutional neural networks or CNNs, and transformers to learn complex image mappings. One technique using this approach is known as SRCNN or super resolution CNN. The model takes an interpolated low resolution image and learns to predict the high resolution details. This process involves feature extraction which is done by applying convolution to extract edge and texture features, non-linear mapping, which transforms features into a high frequency detail representation, and reconstruction to generate the final high resolution image. Mathematically it can be represented as the following equation where W is the CNN parameters and F is the process done on the low resolution image to obtain the high resolution image.

$$I_{HR} = F(I_{LR}; W)$$

Next is enhanced ESRGAN. ESRGAN is a generative adversarial network based model for super resolution designed to generate realistic high resolution images from low resolution inputs. This method is also a common way super resolution is done in the real world. It is sometimes preferred over other methods because of its ability to preserve and enhance textural content in images. ESRGAN consists of two main networks, a generator and a discriminator. The generator, which is based on a deep convolutional neural network (CNN), upscales the low resolution image and the discriminator, also a CNN, judges whether an image is a real high resolution image or a generated one. The generator is trained through many real world images to learn how to produce more realistic details and is also fine tuned to focus on textures and high frequency details, something that other methods severely lack. This method, admittedly, is very computationally complex but is able to produce the most realistic images. It is so good that it is used in satellite image enhancement, medical imaging, surveillance, and has become especially popular in gaming and animation.

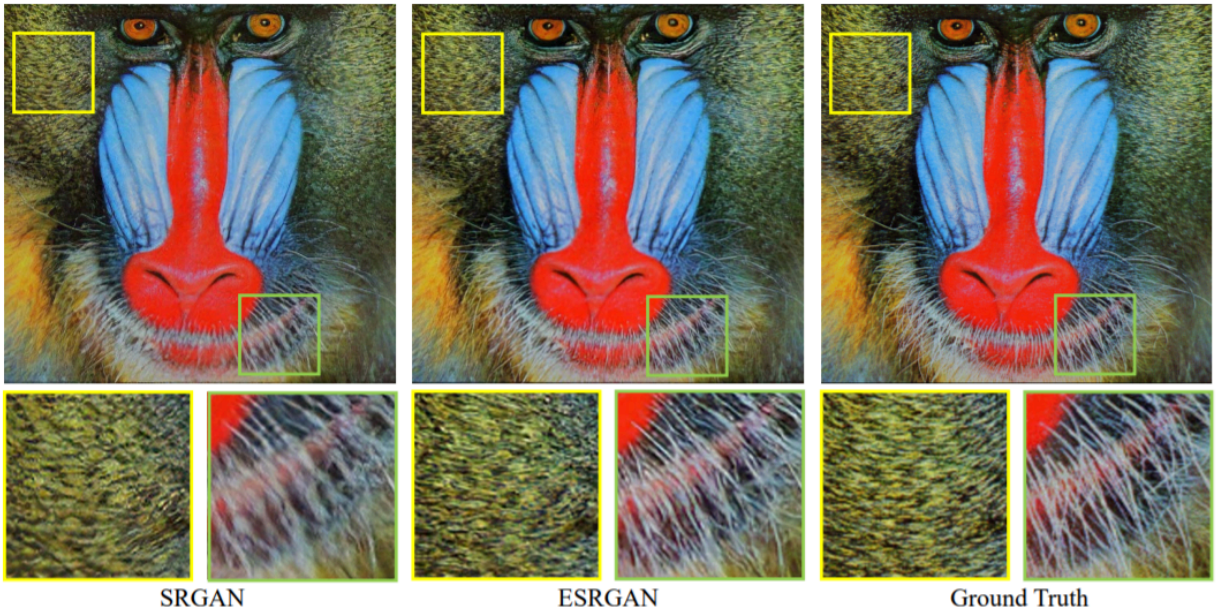


Figure 3.12 *ESRGAN texture upscaling example (Medium)*

Table 3.16 *Comparison of SR methods*

Method	Type	Input	Computational Cost	Image Quality	Pros	Cons
Nearest Neighbor	Single Image, Interpolation	1 Image	Very Low	Poor	Fast, Simple, zero processing time	Blocky artifacts, jagged edges, poor detail retention
Bilinear	Single Image, Interpolation	1 Image	Low	Fair	Smooth results, slightly better than NN	Blurry, soft edges, still loss of detail
Bicubic	Single Image, Interpolation	1 Image	Moderate	Good	Smoother than bilinear, better edges	Blurs fine details, no resolution gain
MISR (Multi-Image SR)	Multi-Image & Alignment	Multiple Images Aligned	High	Very Good	Uses actual new info from	Requires accurate image

					image shifts	alignment, sync capture, more processing time
ERGSAN	Multi-Image & Deep Learning	Multiple Images Aligned	Very High	Excellent	Learns complex patterns, preserves fine details, robust to noise	Requires GPU, training data or pretrained model, sensitive to misalignment

In terms of the super resolution techniques that were outlined, the clear best choices are between MISR and ERGSAN. While ERGSAN has incredible fine detail preservation, it requires a lot more computing power than MISR. For our purposes MISR will be our choice of super resolution technique. This method provides the necessary amount of detail we desire from our output and uses multiple images from slightly different perspectives which aligns perfectly with our project setup. The main thing we will have to consider when using MISR is image alignment and general optical alignment when implementing our lens systems. If there is too much going wrong alignment wise this method will prove rather difficult to implement. Regardless, this method serves our needs best and can be done using Python, though the processing time is not as quick as we desire.

Given a more significant budget and access to more resources, ERGSAN would be the way to go. While the amount of fine detail it is able to provide may seem like overkill, the more information that is able to be gathered from an image the better. Additionally, since this method is trained using AI methods, the project could be taken another step forward by implementing an AI database that is able to remember and recognize different types of aircrafts and anomalies. This addition would help reduce the risk of false alarms and can alert the user of more dangerous situations autonomously. This would require much more computational power than we have access to but would be the absolute best version of this project. If this were to be implemented, the applications of this project could also grow to security cameras, and other surveillance applications making it a jack of all trades unit.

3.16 Front-End Frameworks

Table 3.17 Comparison of Front-End Frameworks

	Javascript (Vanilla)	**React**	**Angular**
Type	Language	Library	Framework
Complexity	Simple	Moderate,	Complex, structured,

		component-based	and opinionated
Setup Needed	None	Moderate	High
Performance	Moderate, inefficient	High	Good
Ideal Use Case	Small projects	Medium to large projects	Large-scale enterprise applications

Javascript (Vanilla)

The first option we evaluated for building the front-end was pure JavaScript. JavaScript is highly versatile and widely supported, making it suitable for both front-end and back-end development. Compared to React and Angular, JavaScript is simpler and requires no additional setup, enabling rapid prototyping and immediate deployment. However, JavaScript relies on direct DOM (Document Object Model) manipulation, which can become inefficient and difficult to manage as applications scale, causing potential performance and code base maintainability challenges in larger projects. It's important we pick the right front-end framework from the start, as changing frameworks mid-development would involve rebuilding the entire front-end.

React

The second option we evaluated for building the front-end was React, a highly popular and powerful JavaScript library developed by Facebook. React is specifically designed for creating interactive user interfaces and is great for developing dynamic, single-page applications, much like what we are planning to build. Its key feature is the use of reusable and modular components, allowing developers to break complex user interfaces into smaller, self-contained units. This modular structure greatly enhances code readability, reusability, and maintainability, making it easier for developers to collaborate and scale projects over time. React employs a Virtual DOM, an efficient mechanism for rendering user interfaces, significantly enhancing performance by reducing costly DOM manipulations. Additionally, React utilizes JSX, a syntax extension that blends HTML and JavaScript, making components more intuitive and easier to debug. Its extensive community support ensures ongoing improvements, numerous third-party libraries, comprehensive documentation, and plentiful online resources, further streamlining the development process. These strengths make React an excellent choice for this project, as it's a medium scale application that needs to run on a wide variety of hardware, and the structure of the framework makes it easy to expand if needed.

Angular

The third option we explored for building the front-end was Angular, a robust and feature-rich framework developed and maintained by Google. Angular is designed primarily for creating complex, large-scale single-page applications (SPAs). Unlike React and pure JavaScript, Angular adds a lot of features for developers, adding features such as built-in routing, state management, form handling, validation, dependency injection, and comprehensive HTTP communication capabilities. Angular uses TypeScript, which introduces static typing and advanced tooling support, significantly enhancing code quality, error handling, and code base maintainability. Additionally, Angular employs two-way data binding, simplifying the synchronization between the application's data model and template view. While this powerful feature set makes Angular

highly suitable for large-scale applications, its structured and opinionated approach presents a steeper learning curve and requires a lot of initial setup effort. While nice to have, complex features like these may not be necessary for a project of this nature, and may needlessly complicate the project.

TypeScript

When using any of these frameworks, we also have the option of using TypeScript instead of just JavaScript. TypeScript is an extension of JavaScript that adds static typing. By default, JavaScript is a dynamically typed language. This means that developers don't have to define the type when defining a variable, and the type of the variable is checked during runtime. This is great for quick prototyping or small projects, but can quickly introduce bugs and form a complex code base. In static typing, the type is checked during compilation, reducing the chances of run-time bugs. Developers must declare the type when declaring variables, making the code base cleaner and easier to maintain. Since we are developing with scalability in mind, TypeScript would be an advantage to use over JavaScript for this project.

Evaluation for Selection

For the front-end of this project, we have many considerations to take when picking the framework we will use. First, performance and efficiency is key in a product like this. We want the web application to run smoothly and fast, ensuring users will be able to access it wherever they are and on whatever devices they are using. We'll evaluate frameworks based on their initial load times, runtime performance, and capability to handle complex interfaces seamlessly. Next, flexibility and customizability is an important factor to consider, since we will be integrating many other third-party APIs and tools into the web application. Frameworks will be assessed on their ease of integration, compatibility with diverse tools and services, and the degree to which they support custom development to meet our specific requirements. Lastly, Security and stability is also very important to us. We have to make sure the framework we are using is safe and secure, to minimize the risk of attacks. Thus, frameworks will be chosen based on their record of handling security vulnerabilities, regularity of security updates, and overall stability in production environments.

For this project, we have chosen to use React as the front-end framework. One reason for choosing React is that it efficiently manages real-time data updates, which is essential for an aircraft detection system. The component system, which allows reusability and scalability, is also very valuable to have during development, simplifying the code base. React also allows for a responsive and interactive user interface, critical for monitoring and surveillance purposes. React is very flexible and customizable, allowing our team to quickly adapt new project requirements to the web application as they arise. React is also a very proven technology, being used in a wide array of web applications for different purposes.

3.17 Map APIs

In this web application, with its purpose of analyzing aircraft detections from remote sensor nodes, integrating a map is essential for providing users with a clear and intuitive understanding of where detections are occurring. Using a map allows users to see the precise locations of each remote node and estimate the general area where an aircraft was detected. To achieve this, the application will require a map API capable of rendering geographical or topographical maps, as

well as supporting overlays such as markers, detection zones, and alert indicators. During initial testing, the system may only be able to identify which node detected an aircraft, without pinpointing an exact location. However, by overlaying each sensor node's estimated detection range on the map, users can visually assess potential aircraft positions and receive immediate alerts when a detection occurs within that range. This not only enhances the web application but also lays the groundwork for future upgrades involving localization, tracking, and trajectory prediction once multi-node detection and triangulation become available.

Table 3.18 Comparison of Map APIs

	Google Map API	**Mapbox**	**Leaflet.js**
Pricing	Paid (usage-based pricing with free tier)	Paid (usage-based pricing with free tier)	Free and open-source
Customization	Moderate, limited control	High, fully customizable with style editor and JSON	High, customizable with CSS/JS and plugins
Performance	High, fast and reliable with global infrastructure	Very High, built on vector tiles and WebGL	Moderate, designed for simplicated, not for heavy datasets
Data Sources	Google's proprietary map and satellite data	OpenStreetMap	OpenStreetMap
Ease of Use	Easy, extensive documentation and support	Moderate, well documented but steep learning curve	Very Easy, simple API and lightweight
Offline Support	Limited, requires setup	Good, supports offline tiles and caching	Good, supports offline tiles and caching
Ideal Use Case	Fast implementation, global accuracy, commercial use	Fully customized apps, data-driven visuals	Lightweight apps, open-source projects, low-budget use.

Google Maps API

The Google Maps API is one of the most popular and widely used tools for integrating geographic maps and mapping services into web and mobile applications. Developed and maintained by Google, this API provides developers with an extensive set of features that allow real-time map rendering, geolocation services, interactive navigation, and dynamic data visualization directly inside of a web application. At its core, the API allows developers to add customizable maps into their web applications and then overlay them with custom content such as markers, lines, polygons, labels, and info windows. What makes Google Maps API different from other APIs is its detailed and globally updated map data, including terrain, satellite, and

street-level views, making it suitable for a wide range of applications, from commercial to personal web applications. With real-time traffic data, route optimization, and live geolocation tracking, Google Maps API is especially effective in applications that demand situational awareness and spatial analysis. For this project however, these features may not be needed, and may add extra complications to the code base. In the context of our project, a passive aircraft detection system, the Google Maps API would give a detailed and responsive visual map to display aircraft detections, sensor coverage areas, and, eventually, flight trajectories when they become available. The support for layering and dynamic event handling allows developers to integrate custom icons that respond to system alerts, draw dynamic paths to show aircraft motion, and even incorporate heatmaps to visualize frequency or intensity of detections over time. Also, the integration of Places API and Geocoding API allows the system to translate coordinates into meaningful location data, offering context to operators monitoring potential airspace incursions. While the Google Maps API is robust and well-documented with a lot of community support, its pricing model, based on usage volume, can become expensive for large-scale or high-frequency applications, which must be considered in budget-conscious projects. For this project however, especially in the testing phase, we should be well within the free tier offered by Google.

MapBox

Mapbox is a flexible mapping platform that provides developers with a wide set of tools to create highly customizable and interactive maps for web and mobile applications. Unlike traditional map APIs that offer fixed styles and limited customization, such as Google Maps API, Mapbox allows complete control over the visual design and functionality of maps, enabling developers to tailor every element, from color schemes and labels to 3D terrain and animated layers, to match the needs of their application. Built on vector maps and using WebGL for rendering, Mapbox is optimized and has smooth performance, with dynamic visualizations. For this project, advanced features such as 3D terrain and animated layers shouldn't be needed, but since it's hard to switch APIs later on, it's good to have options available when picking an API at the start. The smooth performance makes the API particularly well-suited for applications that require real-time updates, such as our project. In the context of our low cost aircraft detection system, Mapbox can be used to overlay sensor ranges, show aircraft detection events, and display historical paths or heatmaps. MapBox also integrates seamlessly with modern development frameworks and supports data sources like GeoJSON (a format designed for representing geography data in a file), making it easy to feed live detection data into the map. Also, Mapbox offers offline capabilities and access to open-source mapping data, which can be great for users with poor bandwidth connection.

Leaflet.js

Leaflet.js is a lightweight, open-source JavaScript library designed for building interactive maps, making it an excellent choice for web applications that require basic geographic visualization without the complexity or cost of bigger mapping platforms like Google Maps API. Its simplicity and ease of use have made it a popular tool among developers, especially for projects like ours that prioritize performance and customization on a smaller scale. Despite its smaller size, Leaflet offers a wide set of features, including support for map layers, custom markers, pop ups, zooming, and panning, as well as integration with popular tile providers like OpenStreetMap, a free open-source set of map data. In the context of our project, an aircraft detection system,

Leaflet can be used to render a clean and responsive map showing the locations of remote sensor nodes, highlight active detection zones, and trigger alerts through map overlays or interactive icons. One of Leaflet's key strengths is its extensibility. Developers can enhance its functionality using a wide range of plugins for heatmaps, clustering, real-time data, and more. It also works well in offline or bandwidth-limited environments using pre-cached tile sets. Because it does not rely on proprietary APIs or paid services, Leaflet is a cost-effective and highly flexible solution for visualizing spatial data in real-time, particularly in projects focused on affordability and open-source development.

Evaluation For Selection

For our project, picking the correct map API from the start is important to us, as a lot of the web application will be built upon it. Firstly, integration and compatibility with other APIs is important, since we are going to have to use other third-party services, and it would be great if they worked together. Performance and reliability is also very important for this project, and when picking the right map API for the front-end, we have to make sure it doesn't affect performance too much. A map API that is too demanding will not work for us, since this web application has to run on most devices. Customization and flexibility is also a factor to consider when choosing a map API. Though not too important, being able to customize the map to fit our needs would be a very nice benefit to have. Scalability is also very important, as we don't know how large this project will eventually become.

Leaflet.js is the best choice for this project due to its lightweight design, ease of integration, and full compatibility with the goals of building a low-cost, scalable, and user-friendly web application. Since the primary objective is to visualize aircraft detections from remote sensor nodes on a map without having to deal with significant additional costs, Leaflet's open-source nature makes it an ideal solution. It's completely free to use and doesn't rely on expensive licensing or proprietary data, instead using open-source data such as OpenStreetMap. Its simplicity allows for quick deployment and customization, essential for the amount of time we have for development, enabling developers to add overlays, sensor range circles, detection markers, and alerts with minimal work and overhead. Leaflet's wide array of community-developed plugins provides flexibility to add real-time tracking, heatmaps, and dynamic data updates as the system evolves. Additionally, Leaflet is highly responsive and performs efficiently even on modest hardware, making it perfect for field-deployed systems or limited-bandwidth environments. Overall, Leaflet.js strikes the right balance between functionality, cost-effectiveness, and simplicity, making it the most practical mapping API for this aircraft detection project.

Offline-Map Data

Another option we have available for developing this web application is downloading open source map data and hosting it from our servers directly to users. Instead of relying on external services, serving the maps ourselves will offer the greatest flexibility, giving us more control on performance, availability, and customization of the data. This approach also helps reduce dependency on third-party APIs, and avoid issues with pricing, rate limiting, and service outages. Users with poor internet connection will also experience better reliability and availability.

3.18 Database

Choosing the right database for the Spectrum Analyzer SDK is critical to ensuring efficient data storage, retrieval, and scalability. There are many database models to choose from, each with their own advantages and disadvantages. This section explores the strengths and weaknesses of different database options and evaluates their suitability for this project. When selecting the Database, GIS capability is going to be important, but other features like maintaining configuration and status of each sensor are also going to be important to note.. Not all users will have access to all sensors, so we will have to be able to manage sensor-user permission relationships. Additionally, the database will need to support features that go beyond just storing signal data. For example, it should be able to handle maintenance tracking, so each sensor's health status and service history can be recorded and reviewed. This is important for long-term system reliability, especially if sensors are deployed in remote or difficult-to-access areas. The database

3.18.1 Relational Databases

Relational databases use structured datasets to store data in tables with predefined relationships. Here are some examples:

- **MySQL**
- **PostgreSQL**
- **Microsoft SQL Server**

Table 3.19 Comparison of Relational Databases

Features	MySQL	PostgreSQL	MicrosoftSQL Server
Efficiency	Moderate	High	Moderate
Flexibility	Fixed schema	Supports JSONB	Fixed schema
Query Performance	High	High	Very high
Geospatial Support	Limited	Yes	Yes
Data Consistency	ACID-Compliant	ACID-Compliant	ACID-compliant

MySQL

For this project, MySQL could be a solid choice if the main priority is handling structured, transactional data efficiently. It is reliable and widely supported, and also performs well with read-heavy workloads, making it a good option for storing user accounts and settings. However, its limited support for real-time streaming data could be a drawback since this project involves continuous updates from a remote sensor node. While MySQL can handle frequent writes, it is not as optimized for high-velocity data ingestion, which might cause performance issues if the system scales. Overall, MySQL could be a great option for the project since it may require structured and transactional data. However, it may not be ideal for handling high-velocity real-time data updates from the remote sensor.

PostgreSQL

PostgreSQL offers more flexibility with its support for both relational and semi-structured data, making it another strong candidate for this project. PostgreSQL has the ability to handle JSONB data, which allows for storing sensor readings in a structured yet adaptable format, which could be useful when dealing with information that might have varying attributes over time. PostgreSQL also has very strong ACID compliance, which ensures that all database transactions are reliable, making it a safer option for critical application data. However, there are some drawbacks to this choice. For example, PostgreSQL's write performance is not as fast as some NoSQL alternatives, which could be a limitation if we need high-speed data intake from the remote source node. Overall, PostgreSQL is a strong candidate due to its hybrid support for relational and semi-structured data. The major potential hindrance is it's real time-performance, but depending on the severity of this issue, it may still be a great choice.

Microsoft SQL Server

Microsoft SQL Server offers strong data integrity and geospatial support, making it reliable for structured applications. However, since this project only needs to check whether an aircraft is in a designated space, its complexity and resource demands may be unnecessary. While SQL Server does provide ACID compliance and strong query capabilities, these features are not essential for storing and retrieving simple real-time position data. With this being said, it may be ideal to choose a lighter, more flexible solution for handling quick lookups and checks with less overhead.

3.18.2 Non-Relational Databases

Non-relational databases are more optimized for flexibility and scalability instead of structured data, making them ideal for handling large volumes of unstructured or semi-structured information that can change over time. These kinds of databases could be useful to this project if we do not always have consistently-structured data coming in from the readings on the spectrum analyzer. Examples of these include:

- **MongoDB**
- **Cassandra**
- **Redis**

Table 3.20 Comparison of Non-Relational Databases

Features	MonogoDB	Cassandra	Redis
Efficiency	High	High	Very High
Scalability	Good for large datasets	Excellent	Limited (memory-based)
Flexibility	Dynamic schema	Wide column store	Key-value storage
Query Performance	Moderate	Limited	Minimal
Geospatial Support	Yes	No	No
Data Consistency	Eventual Consistency	Eventual Consistency	Not ACID-compliant

MongoDB

MongoDB could be beneficial to our project since it requires flexibility in handling rapidly changing data. Since aircraft data, including signal strength and location, might need a dynamic structure, MongoDB's document-based approach makes it easy to store and retrieve this information without strict predefined columns. MongoDB scales well horizontally, which would be useful if we had to deal with data coming from multiple sensor nodes at once, but since this is not the case, this feature would not be applicable here. MongoDB also has great geospatial indexing capabilities, which is another reason it is a strong option. If the system needs to handle geospatial queries to check whether an aircraft is inside a boundary, MongoDB's built-in geospatial features can efficiently determine whether a given coordinate falls within the designated space. Overall, MongoDB provides a good balance between flexibility and performance, allowing for efficient real-time position checks without unnecessary complexity. Its ability to store and query location data makes it a strong fit for our needs.

Cassandra

Cassandra is designed for high availability and distributed scalability, making it a strong choice for handling massive real-time data ingestion. However, since this project only receives data from one node rather than multiple distributed sources, these advantages, such as its ability to handle large-scale distributed writes, are not very relevant. While it can still efficiently manage time-series data like aircraft location, its complex schema design and weaker query capabilities make it a less practical choice for this project. Overall, Cassandra would work well for an application with massive real-time data ingestion, but is less suitable for this project, Since there is no need for massive scalability or multi-node synchronization, a more lightweight solution would be preferable.

Redis

For this project, Redis could be a highly efficient option due to its fast in-memory lookups. Since the only requirement is to quickly check whether an aircraft's coordinates fall within a designated boundary, Redis would allow for near-instant lookups without the overhead of a traditional database. It can efficiently store the latest aircraft positions and provide fast comparisons against the defined airspace. However, Redis is not ideal for storing persistent historical data, meaning that if there is any need for logging past detections, another database would be needed alongside it. While Redis would work well for maintaining the most recent aircraft positions, it might need to be supplemented with another database if long-term data storage is required.

Conclusion

Given the requirements of this project, MongoDB is likely the best choice due to its flexibility, efficiency, and built-in geospatial capabilities. Since the primary goal is to determine whether an aircraft is inside a designated space, MongoDB's document-based structure allows for easy storage and retrieval of real-time position data without enforcing a fixed table structure. While there are other solutions like Redis that favor speed, or PostgreSQL that provide strong relational integrity, MongoDB has the right balance by supporting fast lookups, scalability, and persistent storage without the overhead. This makes it the most practical and adaptable choice for handling aircraft tracking in this system.

3.19 Server Environment

Server environments are platforms or frameworks that run on a web server and handle backend logic, routing, and communication between the front end and the database. They process client requests, manage APIs, and enable features like real-time data updates, authentication, and storage. For this aircraft tracking project, a server environment is necessary to receive incoming position data and deliver that information to the front end efficiently and securely. Servers also play a key role in maintaining session management, enforcing security protocols, and managing background tasks like logging or scheduled updates. In addition, server environments determine which programming languages and tools developers can use, shaping the overall workflow and compatibility with third-party services. The right server environment affects how easily the system scales and how fast it responds, so it is important we pick a good one.

Table 3.21 Comparison of Server Environments

Features	Express	Flask	ASP.NET Core	Spring Boot
Performance	High	Moderate	High	High
Ease of use	Very Easy	Very Easy	Moderate	Moderate
Scalability	Moderate	Limited	High	Very High
Language	JavaScript	Python	C#	Java
Real-Time Support	Excellent (built-in WebSocket	Requires additional setup	Excellent (SignalR)	Requires additional setup

Express(Node.js)

Express is a minimalist web framework built on top of Node.js and is extremely popular for building lightweight, scalable web applications. For this project, Express is an especially strong fit because it uses JavaScript, which is the same language potentially used on the front end. This allows for a streamlined development process across the full stack. It is easy to learn and set up, making development fast and straightforward. On top of this, Express offers excellent built-in support for real-time communication via WebSockets, which is essential for tracking aircraft positions in real time. It also benefits from a massive ecosystem of middleware and strong community support, making it easy to add features or troubleshoot issues. Express scales reasonably well for medium-sized projects like this one and strikes a solid balance between flexibility and performance.

Flask(Python)

Flask is a lightweight Python web framework known for its simplicity and ease of use. It's great for quickly building and deploying small to medium applications. For a project like this, Flask could work if the real-time requirements are limited or handled with additional libraries such as Flask-SocketIO. However, Flask does not include native WebSocket support, so integrating real-time features requires extra configuration. While this adds a layer of complexity, Python's readability and the framework's simplicity make it attractive for small teams or rapid prototyping. It's a good option if we decide we will be working in Python and want something that's easy to understand and maintain. However, a downside of this choice is that additional tooling will be required to handle live aircraft updates.

ASP.NETCore(C#)

ASP.NET Core is a powerful, high-performance framework designed for modern web development. It offers built-in support for real-time communication through SignalR, which is one of the most reliable solutions for managing WebSocket connections. ASP.NET Core is known for its scalability and enterprise-grade stability, making it ideal for systems that may grow over time. For this project, it could be a good fit based on its real-time communication abilities, and if we decide to work in C#. However, it is important to note that it is generally considered to have a steeper learning curve compared to Express or Flask, and setup is more involved. With that being said, it offers strong tooling, excellent documentation, and great support for maintaining long-running, high-availability applications. This makes it a reliable option for aircraft tracking.

SpringBoot(Java)

Spring Boot is a powerful framework that simplifies the setup of Java-based applications by providing auto-configuration and embedded servers. It's widely used in enterprise environments and is known for its strength and scalability. For this project, Spring Boot is a viable choice if we need to scale significantly or integrate with other enterprise-level systems in the future. It does not include native real-time communication tools out of the box, but supports WebSockets through Spring Messaging and external libraries like SockJS. This makes real-time implementation possible but more complex compared to frameworks like Express or ASP.NET Core. Spring Boot also involves a steeper learning curve, especially for those unfamiliar with the Java ecosystem. However, if performance and integration with other services are priorities, Spring Boot could be a solid foundation for the backend.

Conclusion

Given the requirements of this project, Express seems to be the best server environment for this project because it offers the best balance of performance, simplicity, and real-time support. Its native compatibility with JavaScript streamlines development across the front and back ends, reducing complexity and speeding up the workflow. Express also has built-in support for WebSockets, which is crucial for delivering live aircraft position updates to the user interface. Unlike heavier frameworks, Express remains lightweight and easy to maintain, making it ideal for a project that needs fast, real-time communication without the overhead of enterprise-level infrastructure. Its strong community and vast ecosystem of middleware further ensure that the project can be extended and supported with ease as it evolves.

3.20 Web Stack

A web stack is the combination of technologies used to build and run a web application. It typically includes a front-end framework for the user interface, a back-end framework for handling server logic, a database system for storing data, and sometimes a runtime environment or supporting tools. Choosing the right stack is important because it affects development speed, performance, ease of maintenance, and how well different parts of the system work together. For this aircraft tracking project, the stack needs to support real-time data flow, dynamic user interfaces, and a scalable server setup. Since the system involves live aircraft position updates

and user accounts, each part of the stack needs to work efficiently and reliably. Because the system deals with real-time signal data and user interaction at the same time, it's important that the stack supports both speed and flexibility. The front end must be able to quickly reflect changes, like when a new aircraft is detected, without needing to refresh the page. Choosing a web stack that supports all of these goals will ensure that the system is reliable under pressure and can grow when needed. The following are four well-known stacks considered for this project:

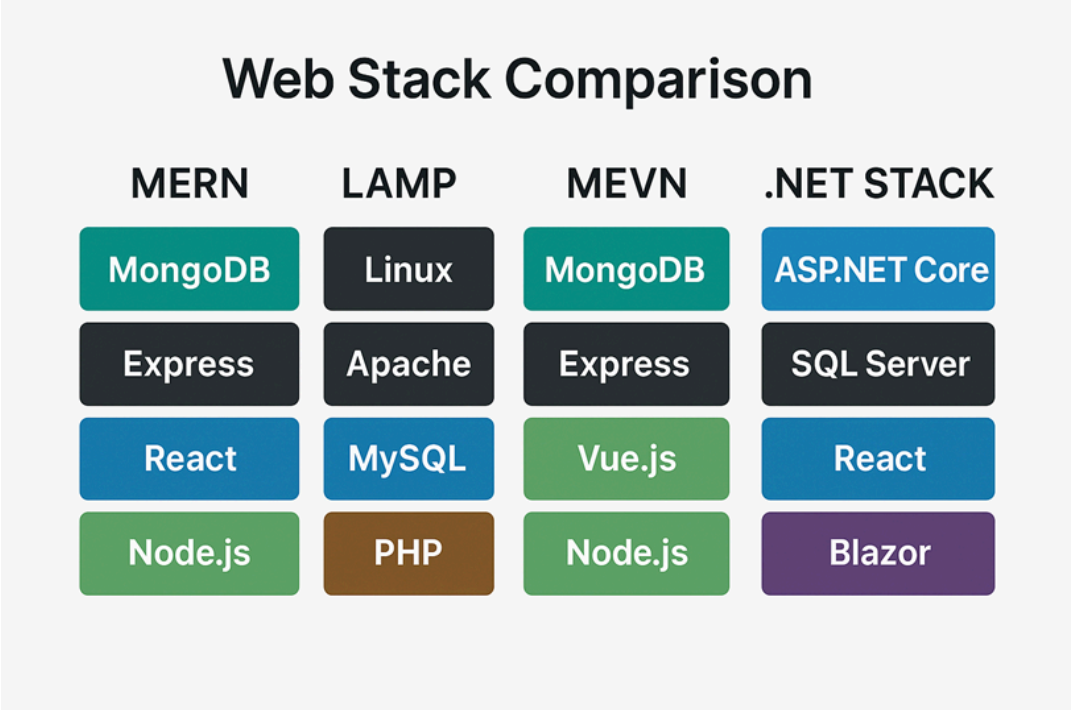


Figure 3.13 *Web Stack Comparison*

MERN

This MERN stack consists of MongoDB, Express, React, and Node.js. This stack is a popular choice for full JavaScript development. All components use JavaScript or variants of it, which makes development consistent across the entire application. React handles the user interface with a focus on dynamic, component-based layouts. Express and Node.js provide the backend logic and server routing. MongoDB stores application and tracking data using a flexible document-based format.

This stack is especially well-suited to real-time applications. WebSocket support is built into the Node ecosystem, and MongoDB's geospatial querying makes it ideal for aircraft boundary detection. MERN is also widely used and well-supported, which means strong documentation and lots of reusable components.

LAMP

The LAMP stack is a more traditional setup. It uses Linux as the operating system, Apache as the

web server, MySQL as the relational database, and PHP for server-side scripting. It's known for its stability and simplicity, and it's still used in many websites today.

While reliable, LAMP might not be the best choice for this project. Real-time data support is not built into the stack and would require additional tools. PHP and MySQL can certainly be used to process and store aircraft data, but integrating a live WebSocket-style system would involve extra work. LAMP is better suited to static or form driven websites than live tracking interfaces. It also lacks built-in tools for handling dynamic, continuously updating data. Implementing modern front-end frameworks or real-time updates in a LAMP environment usually involves additional configuration and bridging technologies. This adds complexity to a project that benefits from a more real-time capable stack.

MEVN

The MEVN stack is very similar to MERN but uses Vue.js instead of React for the front end. Vue is known for being lightweight and easier to learn than React, especially for small teams or projects that don't need highly complex UI logic. Like MERN, MEVN supports JavaScript across the stack and is compatible with WebSockets and real-time features.

For this project, MEVN would work well in situations where the interface needs to display dynamic data such as aircraft position and boundary checks without requiring an overly complex front-end architecture. Vue's reactivity system allows it to handle live updates efficiently, and its built-in features make real-time data binding more intuitive. When paired with Express and MongoDB, the stack remains fast and scalable. Although Vue has a smaller ecosystem than React, it is still well-supported and has a growing number of libraries that could be useful for creating advanced UI features.

.NET Stack (ASP.NET Core, SQL Server, React or Blazor)

This stack uses ASP.NET Core for backend development and is usually paired with SQL Server as the database and either React or Microsoft's Blazor for the front end. It is a powerful choice for high-level applications and supports real-time features through SignalR. SignalR is a library designed for persistent two-way connections, which will have applications for our project.

The .NET stack is highly capable but might be heavier than necessary for this project. The tools and setup are more complex, and development often requires familiarity with C#. SignalR is excellent for real-time aircraft data delivery and would handle location updates reliably. It also provides a solid foundation for user authentication and secure data handling, which could be useful if we expand account management features. In addition, this stack integrates well with Windows-based hosting environments and cloud services like Azure, making it a strong option for long-term deployment. Despite those strengths, its learning curve and system demands may not align well with the relatively lightweight goals of this particular application.

Conclusion

For this project, the MERN stack stands out as the most practical and well-matched option. It provides a unified development environment using JavaScript across all layers of the application, which simplifies the workflow and reduces the time required to switch between different languages. This consistency helps keep the codebase easier to maintain, especially as the project scales or adds new features.React, as the front-end framework, is well-suited for building the

interactive interface required to display aircraft positions on a map. It supports real-time data updates efficiently and integrates easily with mapping libraries and other dynamic components. Express and Node.js on the server side provide built-in support for WebSockets, which is essential for delivering live aircraft data with minimal delay. MongoDB rounds out the stack with flexible document-based storage and native geospatial querying, which is useful for checking whether aircraft coordinates fall within the defined airspace.

Other stacks like MEVN and .NET do offer strong capabilities, but for our needs, MERN provides the best balance of ease of use, performance, and real-time support while remaining relatively lightweight. We also have had experience using a MERN stack in our previous courses, which will reduce the learning process. Its popularity also means access to many libraries, community resources, and examples, which can speed up development and problem-solving. For a real-time aircraft tracking web app that needs to be responsive and developer-friendly, MERN is the best overall fit.

3.21 Image Object Detection

Object detection in images is a computer vision technique that involves identifying and locating objects within an image. Generally, it uses algorithms to analyze pixel patterns and features to recognize specific objects such as people, vehicles, or animals. This process typically involves two key steps: classification, where the system determines what is present in the image, and localization, where it places bounding boxes around each detected object. Object detection can be enhanced by super-resolution techniques, which increase the quality and detail of low-resolution images. This is especially important in scenarios like aerial surveillance, where targets (like aircraft) are far away and may be difficult to detect clearly, especially in cloudy or stormy conditions. By combining multiple low-resolution images from a camera array, the system can create a higher-resolution composite image, improving the accuracy of object detection and tracking.

Table 3.22 Comparison of Object Detection Models

	TensorFlow	**YOLO (You Only Look Once)**	**Detectron2**	**OpenCV**
Speed	Moderate-High	Very High	Moderate	Very High
Accuracy	High	Moderate-High	Very High	Low-Moderate (Less accurate for complex scenes)
Ease of Use	Moderate (requires setup)	Moderate (If using pre-trained models)	Moderate-Easy	Easy, straightforward setup and deployment

Deployment Environments	Web, mobile, cloud	Embedded devices, desktop applications	Server, desktop, research environments (heavy computational demand)	Embedded systems, desktop applications
Ideal Use Case	Industrial automation, traffic surveillance, object tracking	Real-time video analysis, surveillance, drones	High-accuracy applications, scientific image analysis	Rapid prototyping, low-resource devices, simple detection tasks

TensorFlow

TensorFlow's Object Detection API, developed and maintained by Google, is one of the most widely used libraries for performing object detection in images. TensorFlow provides a wide range of pre-made models and architectures, including SSD (Single Shot MultiBox Detector), Faster R-CNN, and the EfficientDet family, offering various accuracy and different computational performance based on requirement needs. The flexibility provided by these architectures allows users to choose models based on their specific needs, Faster R-CNN for high accuracy, or SSD and EfficientDet for faster inference and resource-constrained environments, with decreased accuracy. For our project, since at first the images will mostly be images of the sky and a single object, an architecture like SSD or EfficientDet will be sufficient. Additionally, TensorFlow’s extensive model zoo simplifies access to pretrained models, facilitating rapid development. The API is great in environments where robust, scalable deployment is necessary, such as cloud-based services or production-level applications in industrial automation and traffic monitoring. However, the initial learning curve can be significant, particularly for developers unfamiliar with TensorFlow's ecosystem or dependency management. Also, resource consumption is relatively high for certain models like Faster R-CNN, making them less suitable for deployment on low-powered devices. However, its powerful capabilities, active developer community, thorough documentation, and compatibility with hardware accelerators (GPUs/TPUs) make the TensorFlow Object Detection API a one of the top choices for our project.

YOLO (You Only Look Once)

YOLO is known for its fast speed and real-time object detection capability, revolutionizing how quickly and effectively objects can be identified within images or video streams. YOLO achieves this by analyzing images through a single neural network forward pass, processing the entire frame simultaneously rather than relying on sliding window approaches. Versions such as YOLOv4, YOLOv5, and YOLOv8 have significantly improved detection accuracy and inference speeds, making them widely popular in both academic and industry circles. YOLO is particularly advantageous in scenarios where rapid response times are crucial, including autonomous driving, drones, security systems, and real-time video analytics. Its lightweight variants can operate effectively on mid-tier GPUs or optimized CPUs, extending its reach to embedded and edge devices. However, one limitation of YOLO is its comparatively reduced accuracy for detecting

very small or densely packed objects, where models like Faster R-CNN often outperform it. Furthermore, architectural adjustments beyond standard configurations can become intricate, though tuning hyperparameters and training custom datasets remains relatively straightforward. Due to the exceptional speed, accessibility, extensive community-driven support, and the availability of pretrained models, it is clear why YOLO remains the preferred choice for projects prioritizing speed and efficiency in real-time detection.

Detectron2

Detectron2, created and maintained by Facebook AI Research, represents one of the most sophisticated libraries for object detection and instance segmentation built upon the PyTorch framework. It offers comprehensive support for state-of-the-art architectures like Mask R-CNN, Faster R-CNN, RetinaNet, and Keypoint R-CNN. This makes Detectron2 particularly powerful for research-intensive applications, where cutting-edge performance, accurate object localization, and segmentation are very important. The ease of integration with PyTorch significantly simplifies custom model creation, modification, and experimentation, appealing strongly to academic researchers and advanced developers. The extensive Detectron2 Model Zoo provides a variety of pretrained models, allowing quick baseline implementations. Despite its strengths, Detectron2 can be resource-intensive, demanding high GPU memory and powerful computation capability, particularly during training and inference, making deployment on low-powered devices challenging. Its complexity, however, is offset by its rich documentation, strong community support, and ongoing enhancements driven by active contributions from the AI research community. As a result, Detectron2 is exceptionally suited for scenarios demanding precise detection, instance segmentation, and fine-grained analysis, typically encountered in research labs, scientific imaging, medical applications, and high-end security systems.

OpenCV

OpenCV (Open Source Computer Vision Library) remains a cornerstone for traditional computer vision tasks, including object detection using classical methods such as Haar cascades and Histogram of Oriented Gradients (HOG) combined with Support Vector Machines (SVM). These methods offer distinct advantages, including computational efficiency, ease of implementation, and suitability for rapid prototyping or deployment on low-resource embedded systems and mobile platforms. Haar cascades, for example, enable real-time face and pedestrian detection with minimal computational overhead, making them ideal for embedded systems or IoT devices. Similarly, the HOG descriptor method is efficient at detecting well-defined objects under controlled conditions, such as pedestrian detection in surveillance scenarios. However, traditional OpenCV-based methods fall short in complex environments, variable illumination conditions, or scenes requiring detection of objects with high variability or occlusion, due to their lower accuracy compared to deep learning approaches. Additionally, classical methods necessitate custom training and tuning to achieve acceptable accuracy levels, and pretrained models are limited. Recently, OpenCV has enhanced its support for deep learning-based object detection by providing integration with TensorFlow and PyTorch models through its DNN (Deep Neural Network) module, extending its capabilities beyond classical techniques. Nevertheless, OpenCV's main advantage lies in applications demanding straightforward implementation, lightweight processing, real-time detection on low-resource hardware, or as a complementary approach to more resource-intensive, deep learning-based detection methods.

Evaluation for Selection

Initially, we thought OpenCV would be the best option for this project, since it's ideal for low-resource applications and easy to set up. However, due to several key limitations that arise from the traditional detection methods used by OpenCV, it is clear that it's not the best option. Aircraft, especially at significant distances and varying angles, exhibit wide variability in appearance, contrast, size, orientation, and illumination. This complexity poses substantial detection challenges for these traditional algorithms, greatly reducing their accuracy, increasing false-positive rates, and thus limiting reliability. Additionally, OpenCV's classical methods struggle significantly when faced with complex, changing backgrounds and environmental conditions such as atmospheric distortion, variable lighting, and weather effects like haze or cloud cover. Furthermore, OpenCV's traditional methods perform poorly with detecting small or distant objects, creating a weakness for the intended application of detecting far-away aircraft.

Therefore, for our passive aircraft detection and super-resolution imaging system, YOLO clearly emerges as the best choice due to its unmatched real-time processing capability, high efficiency, and adaptability to our specific project constraints. Given that one of our main goals is achieving a cost-effective and rapid detection system, YOLO's architecture aligns perfectly, providing excellent frame rates even on lower-cost hardware platforms such as Raspberry Pi or Jetson Nano—exactly what we're looking for. It can detect and track objects swiftly, which is crucial for immediate aircraft identification, alerting, and situational awareness. Although traditionally YOLO may have slightly lower accuracy in detecting very small or distant objects, our super-resolution imaging technique effectively compensates for this limitation by providing enhanced, clearer input images, improving detection performance significantly. Furthermore, YOLO's extensive and active community support, along with numerous pretrained models tailored to aerial and drone imagery, simplifies integration into our project, helping us move faster through testing and development phases. Additionally, its ease of use and well-documented procedures fit well within our group's experience, especially since we're planning to utilize Python and MATLAB. Ultimately, selecting YOLO not only meets our immediate project requirements of speed, efficiency, and ease of implementation but also ensures scalability and future-proofing.

3.22 Anomaly Detection Algorithms

There are many different environmental factors that will affect the ability of the system. Things such as: The amount of leaves that may be in the way. The amount of humidity and or rainfall that may be present. The ambient temperature. Or how RF noisy the environment may be. Data taken near Orlando's airport, MCO, will be very different from that taken in the middle of nowhere because of the amount of RF energy that is present. This will add up when it comes to how our algorithm will be able to detect as there are a lot of factors that can change the baseline noise level. With the algorithm we will try to minimize the amount of false alarms. All of this data will be collected and tabulated raw for the sake of creating an algorithm that works effectively.

Anomaly detection algorithms are used to identify data points that are significantly different from the normal values, helping to uncover outliers and errors. For our use case, an anomaly detection algorithm will be perfect for detecting an airplane flying overhead an antenna, since the RF signals will spike when something enters its detection range. This spike is what we would classify as an anomaly, and catching it accurately is very important for our system. Anomaly

Detection Algorithms can be split into two main categories, unsupervised anomaly detection and supervised anomaly detection.

Unsupervised anomaly detection aims to identify anomalies in data without using labeled examples of anomalies. It works by learning the normal distributions of the data, and flagging any data that is not near the normal range. For example, our RF antenna is receiving a constant steady stream of RF energy, and then it suddenly registers a large spike in power at a specific frequency, the algorithm will recognize that spike as an outlier and trigger an event. One common method here is using a rolling mean and standard deviation. If a RF signal exceeds a certain number of standard deviations from the average, it's flagged as an anomaly.

Supervised anomaly detection involves training a machine learning model on labeled examples to identify anomalies in datasets. This works by learning to distinguish between normal and non-normal patterns, which can be used to classify new data points as either normal or anomalous. A supervised model can be trained using machine learning applications like Random Forests, Neural Networks, or Support Vector Machines to distinguish between regular RF background noise and the signatures of aircraft.

For our use case, unsupervised anomaly detection is the best option because it's more lightweight and practical than supervised methods. Supervised anomaly detection relies on having a large, labeled, dataset that defines what data is normal and what data is an anomaly. Gathering data is very time consuming, and training models requires a lot of computational power, especially when using more complex machine learning models. Where efficiency is important on a system like ours, a lightweight method is more desirable. Unsupervised anomaly detection doesn't need any labeled data, instead it just observes incoming RF signals over time, learning the typical patterns, and flag anything that is an outlier. This lightweight method is ideal for embedded systems, where low-cost, low-power, and ease of deployment is wanted.

3.22.1 Types of Unsupervised Anomaly Detection Algorithms

Table 3.23 Comparison of Anomaly Detection Algorithms

	Real-time Friendly	Computational Cost	Ease of Implementation	Detection Power
Z-Score / Statistical Thresholding	Yes	Very Low	Very Easy	Moderate, best for obvious anomalies
Moving Average + Threshold	Yes	Very low	Very Easy	Moderate, great for sudden jumps
Isolation Forest	No, best for offline or batch detection	High	Medium, easy to use with libraries	High, good at identifying non-obvious anomalies

One-Class SVM (Support Vector Machine)	No, not ideal for streamed data	High	High, requires lots of tuning	High
K-Means Clustering	Limited, needs batch updates, not optimized for streamed data	Medium	Medium, requires some tuning	Moderate, struggles with noise
DBSCAN (Density-Based Spatial Clustering)	Limited, better for static datasets	Medium	Hard, requires lots of tuning	Moderate, good for finding noise, but may miss outliers
Autoencoders (Unsupervised Neural Networks)	No, needs offline processing	Very High	Hard, requires data prep and lots of tuning	Very High, great at identifying non-obvious anomalies
Principal Component Analysis	Limited, can be adapted for live use	Medium	Medium, needs preprocessing	Moderate
Histogram-Based Outlier Score (HBOS)	Yes	Very Low	Easy	Moderate, great for simple signals

Z-Score / Statistical Thresholding

One of the simplest and most effective unsupervised anomaly detection techniques is Z-Score-based thresholding. It's a statistical method that assumes the data follows a normal distribution and works by measuring how far a particular data point deviates from the mean in terms of standard deviations.

In this method, the system constantly calculates the mean and standard deviation of the incoming signal data, like RF signal values. Each new data point is then assigned a Z-score. If the Z-score of a data point exceeds a certain threshold, it is considered an anomaly. In our case, that would indicate a sudden spike in RF signal energy, possibly caused by an aircraft reflecting ambient signals toward the antenna. This method is ideal for real-time, resource-constrained environments like ours because it's lightweight and doesn't require much memory or CPU power, it doesn't need any labeled training data, and it's easy to tune and adjust, just change the threshold based on how sensitive you want the detection to be.

Moving Average + Threshold

Another lightweight and practical anomaly detection method is the Moving Average + Threshold approach. This technique is great for systems that process streaming data in real time, like potentially ours, where we're constantly monitoring RF signal strength from the antenna.

The algorithm maintains a moving average of recent signal values over a fixed time window and compares new incoming values to that average. If a new value exceeds the moving average by a predefined threshold margin, it's flagged as an anomaly. In our context, that anomaly could very likely be an aircraft reflecting a strong RF signal back toward the antenna. This method works well for a few reasons, it's simple to implement and doesn't require any complex math or data structures, it's computationally efficient, making it a great option for embedded systems or low-power hardware, and it adapts over time, since the average updates continuously, it automatically adjusts to slow changes in ambient RF levels.

Isolation Forest

Isolation Forest is a more advanced unsupervised anomaly detection algorithm that takes a different approach compared to traditional statistical methods. Instead of modeling what normal data looks like and identifying outliers based on distance or density, it works by trying to isolate anomalies as quickly as possible using random splits. Anomalies are rare and different in large datasets, so it should take fewer random "cuts" to isolate them from the rest of the data. Isolation Forest builds a set of binary trees by randomly choosing a feature and a split value at each node. Points that are easier to isolate are more likely to be anomalies. The result is an anomaly score for each data point based on how deep it lies in the tree structure.

For our project, Isolation Forest could be used to detect unusual patterns in the RF signal data that might not always look like obvious spikes. It's especially useful if the signal becomes more complex over time, with more subtle anomalies caused by drones, different aircraft, or interference patterns.

Isolation Forest could be a good option for a few reasons. First, it doesn't assume a distribution, which makes the algorithm flexible for real-world applications. Next, it works well with high-dimensional data, which could be useful if we ever want to feed in other sensor data. Lastly, Isolation Forest scales well, handling large datasets with ease. Unfortunately, Isolation Forest also comes with some downsides. It's not very lightweight compared to other methods, which is a factor to consider since we will be running on low-power hardware. Also, Isolation Forest requires a batch of data to be able to analyze, which limits our options on how we can transmit data.

One-Class SVM (Support Vector Machine)

One-Class SVM is another unsupervised anomaly detection algorithm that can be used when we only have access to normal data and want to flag any data that is not normal. It's based on the same principles as the traditional Support Vector Machine (a supervised learning algorithm used for classification and regression tasks in machine learning) used in classification, but adapted for one-class learning. The algorithm draws a boundary around data that is considered normal and flags anything outside that boundary as an anomaly. The algorithm learns a decision function that separates the normal data points from the origin in a high-dimensional feature space. If a new data point lies outside this boundary, it's considered suspicious. In our case, we could train a

One-Class SVM on a dataset of normal RF background noise, and then use it to detect anomalies like spikes caused by aircraft.

One-Class SVM can handle complex, non-linear patterns, and works without labeled anomalies. Unfortunately, One-Class SVM is computationally expensive, which is not ideal considering our hardware. Parameter tuning is also very sensitive, taking a lot of work to get right and running properly and efficiently. Also, like Isolation Forest, the algorithm requires batches of data and would not support a constant stream, limiting our options.

K-Means Clustering

K-Means Clustering is a classic unsupervised learning algorithm that's often used for anomaly detection by grouping data into clusters and identifying points that don't belong to any group well. It works by partitioning data into K distinct clusters based on similarity, usually using distance from the cluster center as the metric. For anomaly detection, we treat points that fall far from all cluster centers as outliers. In our case, this could be used to cluster normal RF signal patterns, and then any sudden spike or unusual frequency signature that doesn't fit into those clusters could be flagged as an anomaly, like an aircraft reflection showing up in the data.

K-Means Clustering is simple and fast, making it ideal in real-world applications where performance is important. The algorithm also works well when the data is naturally grouped into clusters, like noise levels or frequency patterns. Also, the algorithm doesn't require labeled data, which makes it easy to implement. Unfortunately, K-Means Clustering comes with some complications. When using the algorithm, you have to choose the number of clusters ahead of time, which might be hard to do considering our use case. The algorithm also assumes the data forms spherical clusters, which RF signals might not form. Lastly, the algorithm doesn't always perform well with real-time data.

DBSCAN (Density-Based Spatial Clustering)

DBSCAN is a clustering algorithm that's commonly used for anomaly detection, especially when the data doesn't form neat or spherical clusters. Unlike K-Means, which groups data by minimizing distance to a cluster center, DBSCAN groups data based on density, how closely packed data points are. Anything that falls outside these dense regions is labeled as noise, which in our case could indicate an RF anomaly. For each point in a dataset, the algorithm checks how many neighboring points exist within a certain radius. If there are enough neighbors, that point is considered part of a cluster. Points that don't meet the requirement and don't fall within any dense cluster are marked as outliers. This method is great for detecting RF events that don't fit normal background patterns, like an aircraft reflection that suddenly introduces a power spike or frequency signature that's not part of the usual noise pattern.

DBSCAN can be a good option because when using the algorithm, you do not need to specify the numbers of clusters ahead of time, unlike K-Means Clustering. DBSCAN is also more flexible, with the ability to detect different shaped clusters. The algorithm also automatically identifies outliers as part of the design. Unfortunately, using the algorithm can be tricky since you have to fine tune it to the data it will be receiving. The algorithm can also struggle if data has

varying densities, which is possible with RF signals spending on interference or background conditions.

Autoencoders (Unsupervised Neural Networks)

Autoencoders are a type of unsupervised neural network that can be used for anomaly detection by learning how to reconstruct normal data. Autoencoders work by training the network to compress and then reconstruct its input. If the input is something it's seen before, or something similar to what it's been trained on, it can reconstruct it pretty accurately. But if the input is something unusual, like a sudden RF spike caused by an aircraft, the reconstruction error will be high. Autoencoders can be great for complex, non-linear patterns of data, but unfortunately it is very computationally expensive, requiring good hardware or a separate server. Due to this, Autoencoders are not a good option for our use case.

Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method that's often used for dimensionality reduction, but it can also be applied to unsupervised anomaly detection. It works by identifying the main directions (called principal components) in which the data varies the most. PCA tries to find the shape of your normal data and project new data onto that shape. If a new point doesn't fit well into the learned pattern, it's flagged as an anomaly. The approach PCA takes is useful in cases like ours where we expect RF signal data to fall into certain repeatable patterns, and anomalies, or RF signal spikes, would distort that pattern.

Principal Component Analysis is fast and lightweight, performing much better than neural networks and clustering methods. PCAs also don't need labeled data, which makes setting up and implementing the algorithm easier. Unfortunately, this method assumes that data is linearly separable, whereas RF signals can be more complex and not match that requirement.

Histogram-Based Outlier Score (HBOS)

Histogram-Based Outlier Score (HBOS) is one of the simplest and fastest unsupervised anomaly detection algorithms available. The algorithm works by building histograms of each feature in the dataset and then scoring how rare or unusual a new data point is based on where it falls in those histograms. If a point falls into a bin with very low frequency, meaning it doesn't show up often in the training data, it gets a high outlier score. HBOs differ by assuming all features are independent, so it evaluates each feature separately and combines the scores. That independence assumption makes it very fast, and in many cases, accurate enough to catch anomalies without needing complex models.

Histogram-Based Outlier Score is extremely fast and scalable, working well with large datasets and real-time use. Model training is also not required, and HBOS uses very low memory and CPU power, working nicely with embedded systems. Unfortunately, HBOS assumes independent features, which isn't always true with RF signal data. The algorithm also requires a lot of tuning to work efficiently, needing to be carefully adjusted.

Evaluation For Selection

When selecting a method to use for unsupervised anomaly detection, some important factors we have to consider are the performance of the algorithm, how much computational power it

requires, and what form the data will be in (real-time streamed or in batches). Computational efficiency is also important. Since our system is intended to be low-power and potentially solar- or battery-operated, we can't afford to run anything that requires GPU acceleration or heavy matrix operations. The algorithm has to work on small, embedded hardware and still produce reliable results. Additionally, we need something that's predictable and explainable.

Z-Score / Statistical Thresholding will be the best option since it fits our use case the best. This method is very lightweight, not requiring any model training, memory-intensive operations, or any complex math. This method also works great for real-time detection, and makes no assumptions about the structure of the data, making it very flexible to use. This algorithm is also very easy to implement and perfect for the data obtained from the RF sensor.

During development, we decided to switch to using a Convolutional Neural Network (CNN) due to the difficulty of finding reliable features to extract for traditional machine learning. Originally, we tried using basic statistical methods and manually selected features like peak height, width, or slope from the RF signal. But the results were inconsistent. Depending on noise, signal interference, or slight changes in the background environment, those features would vary too much, making it hard to train a reliable model.

CNNs, on the other hand, are great at picking out patterns in structured data, even data like RF signals. Instead of us trying to handcraft every feature, the CNN learns what patterns actually matter just by looking at the training data. In our case, we fed it RF signal slices from the spectrum analyzer, where each sample was essentially a sequence of signal amplitudes over time. The CNN starts by applying small filters across that signal, picking up basic shapes like sudden jumps or slow rises. As it goes deeper, it combines those into more complex patterns that might represent a spike from an aircraft, for example.

This made our detection system way more flexible. It doesn't matter if the noise floor shifts slightly or if the signal looks a little different depending on the time of day or distance, once trained, the CNN focuses on the underlying pattern. It worked especially well when we tested with both clean RF samples and noisy ones. The CNN was still able to detect potential aircraft signatures that traditional methods would either miss or flag incorrectly.

Security Measures

Security is a critical aspect of our front-end system, and we're taking several measures to ensure user data protection and integrity. First, all data transmitted between the camera array and the user interface will be encrypted using standard security protocols (e.g., SSL/TLS), protecting sensitive information from unauthorized access. Additionally, authentication mechanisms such as secure login credentials and multi-factor authentication (MFA) will be implemented, ensuring only authorized personnel have access to the visual data and system controls. Stored images and metadata will also utilize encryption and restricted access permissions to safeguard data at rest. Furthermore, regular security audits and updates will be part of the system's maintenance plan to quickly address vulnerabilities and adapt to emerging threats. These combined security measures ensure that our solution remains resilient, trustworthy, and suitable for applications where data privacy and integrity are essential.

3.23 Data Protocols

Table 3.24 Comparison of Data Protocols

	Type	Reliability	Speed	Use Case
IP (Internet Protocol)	Network-layer	N/A (used for addressing and routing)	Depends on other protocols used	Core routing for all communication
TCP (Transmission Control Protocol)	Connection-oriented	High	Moderate	Reliable control and data synchronization
UDP (User Datagram Protocol)	Connectionless	Low (no guarantee of delivery or order)	High	Low-latency video or sensor data transmission
HTTP (Hypertext Transfer Protocol)	Application-layer	Moderate (reliable but not real-time)	Low to Moderate	Remote configuration and data access
HTTPS (Hypertext Transfer Protocol Secure)	Application-layer (secure)	High (secure and reliable)	Low to Moderate	Secure remote system access and data transfer
WebRTC (Web Real-Time Communication)	Real-time communication	High (optimized for real-time streaming)	Very High	Real-time video/audio stream between components

Data protocols are standardized rules and formats that govern how data is transmitted, received, and interpreted between devices or systems. In the context of this project, they play a crucial role in ensuring efficient communication between the various components of the aircraft detection and imaging system, such as RF detectors, spectrum analyzers, and the camera array. These protocols help maintain synchronization, reduce transmission errors, and ensure compatibility across devices, especially when multiple systems are integrated, such as USB-based camera modules and signal analyzers connected to processing units. Effective data protocols are essential for enabling real-time data streaming, accurate sensor fusion, and reliable system responses in surveillance and detection applications.

IP (Internet Protocol)

IP is the backbone that all these protocols ride on. It handles addressing and routing data packets to their destination. Whether we're using TCP, UDP, or any other protocol, IP is always involved. For our detection system, IP enables communication between all the devices—whether they're connected via Ethernet or Wi-Fi—making sure data gets to the right component in the setup.

TCP (Transmission Control Protocol)

TCP is a reliable, connection-oriented protocol that ensures data is delivered accurately and in the correct order. It's perfect for situations where data integrity matters, like sending control commands between the camera system and processing units. In our project, TCP can be used to guarantee reliable communication between modules, which is especially important for synchronizing detection results and ensuring the system behaves predictably under different conditions.

UDP (User Datagram Protocol)

UDP, on the other hand, is a faster, connectionless protocol that doesn't guarantee delivery or order. It's useful in scenarios where speed is more important than reliability—like streaming real-time video from the camera array or RF data bursts. In the context of our system, using UDP can help minimize latency for time-sensitive data, which is crucial for real-time surveillance and immediate detection alerts.

HTTP (Hypertext Transfer Protocol)

HTTP is a protocol commonly used for transferring data over the web. While it's not designed for real-time performance, it's still useful in this project for sending configuration settings, updates, or even accessing logged data from the system remotely. It acts more as a general-purpose communication layer for human interaction or background processes rather than the core of the real-time detection system.

HTTPS (Hypertext Transfer Protocol Secure)

HTTPS is basically HTTP with a security layer (SSL/TLS) on top. It encrypts data during transmission, which is important when we need to protect sensitive system data or camera feeds, especially if the system is accessible over a network. In this project, HTTPS could be used for secure remote access or when transmitting sensitive detection results or configurations to a central monitoring system.

WebRTC (Web Real-Time Communication)

WebRTC is designed for peer-to-peer communication in real time, which makes it super helpful when streaming live video or sensor data directly between components without needing a server in the middle. For our project, this could be a great option for setting up real-time streams between the camera array and the analysis software, especially if we want low-latency communication and reduced overhead.

Websockets

WebSockets are a powerful communication protocol used to facilitate real-time, two-way interactions between a web browser and a server. Unlike traditional HTTP, which relies on repeated requests from the client to the server to fetch updates—a process called

polling—WebSockets maintain an open, persistent connection. This persistent connection enables the server to send data directly to the client without waiting for a request, resulting in reduced latency and lower network overhead. When a WebSocket connection is established, it begins with a standard HTTP request from the client, which includes a handshake process. Once the handshake is complete, the connection upgrades from HTTP to the WebSocket protocol, allowing real-time data flow. WebSockets are especially useful in applications where instant data transfer is critical, such as chat applications, multiplayer online games, live-streaming platforms, and real-time monitoring systems. They provide a more efficient and responsive user experience compared to other methods like AJAX long polling. Additionally, since WebSockets use a single connection for both sending and receiving data, they significantly reduce network congestion, making them ideal for applications requiring rapid and frequent communication between clients and servers.

WebSockets would be perfect for this project, especially given that our passive aircraft detection system and high-resolution imaging device rely on transmitting real-time data to ensure quick and accurate responses. Since our system is designed for surveillance purposes, particularly tracking aircraft in RF "dead zones", it's crucial to deliver data instantly from the sensors to a monitoring interface or command center. Traditional HTTP methods, like polling, would introduce unnecessary latency and strain network resources by continuously requesting data, especially when monitoring fast-moving aircraft. WebSockets solve this by creating a constant, two-way communication channel between our device and the monitoring computer or mobile device, allowing the system to push immediate updates as soon as an aircraft is detected or identified. Moreover, the reduced latency provided by WebSockets ensures minimal delay in transmitting critical information, enhancing the effectiveness of security or response actions. Because our project could potentially involve multiple detection devices integrated as an array, WebSockets make it easier and more efficient to synchronize data from different sensors, maintaining the accuracy and integrity of our overall surveillance system. This approach would also be highly beneficial in our camera array configuration—where images or processed data from several cameras need to be rapidly combined into high-resolution composites—by facilitating quick data exchange and seamless integration into the final image output.

3.24 Encryption Protocols

Table 3.25 Comparison of Encryption Protocols

	SSL/TLS	IPsec	SSH	DLTS
Used for	Encrypting web traffic between browsers and websites	Encrypting data at the network level, VPNs	Secure remote login, file transfer, command execution	Encrypting UDP traffic
Encryption	RSA (setup),	AES, SHA-256	AES or	AES, HMAC

	AES (data), SHA-256 (integrity)		ChaCha20, Public Key for setup	
Mode	Works over TCP	Transport and Tunnel modes	Works over TCP	Works over UDP
Main Features	HTTPS, secure certificates, strong authentication	IP-layer encryption, protects all traffic, VPNs	Encrypted remote terminal and tunneling	TLS for datagrams, handles packet loss/reordering
Connection Type	Connection-oriented (TCP)	Both (TCP/UDP, encapsulated in IP)	Connection-oriented (TCP)	Connectionless (UDP)
Common Use Cases	HTTPS, email security (SMTPS), VoIP over TCP	VPNs, secure site-to-site communications	SFTP, remote shell, Git over SSH	VoIP, video conferencing, online gaming

SSL/TLS

SSL/TLS (Secure Socket Layer/Transport Layer Security) is one of the most common protocols. It's used to protect data sent between web browsers and websites. TLS replaced SSL because SSL had some security weaknesses that were being exploited. TLS was created to fix those issues and expand upon SSL. TLS helps ensure that data stays private, isn't changed during transmission, and comes from a trustworthy source. It uses special certificates provided by trusted groups, and different types of encryption, one type for setting up a connection (asymmetric) and another type for sending data (symmetric). Common methods include RSA for starting secure connections, AES for keeping data secret, and SHA-256 for checking data integrity.

IPsec

IPsec, or Internet Protocol Security, protects data at the network level. It's commonly used in VPNs (Virtual Private Networks) to safely send information across unsecured networks like the internet. IPsec works in two ways, transport mode, which encrypts only the actual message inside an IP packet, and tunnel mode, which encrypts the entire IP packet and wraps it inside another packet. It uses Authentication Header (AH) to confirm that data hasn't been altered and Encapsulating Security Payload (ESP) for encrypting data. Common encryption methods include AES for security and SHA-256 for verifying data integrity.

SSH

SSH, or Secure Shell, helps secure connections when logging into other computers remotely. SSH replaced older, unsafe protocols like Telnet, due to having superior security methods. SSH

makes sure data stays encrypted between a client and a server during remote sessions. SSH lets you securely run commands, transfer files (using SFTP), and even secure other types of network communication. It uses public-key cryptography for verifying users and symmetric encryption like AES or ChaCha20 for protecting data during the session. SSH also has ways to prevent attacks, such as man-in-the-middle and password-guessing attacks.

DLTS

DTLS, or Datagram Transport Layer Security, is similar to TLS but designed for securing data sent over UDP. UDP is a way to send data quickly without making sure all packets arrive or in the correct order. DTLS was created to add security to this type of fast data transfer. It provides confidentiality, verification of data authenticity, and ensures data hasn't been tampered with, just like TLS, but it also deals with issues like missing or out-of-order packets. DTLS is commonly used in real-time applications like video calls, gaming, and online media streaming. Common encryption methods used are AES for privacy and HMAC algorithms for checking data authenticity.

Chapter 4: Standards & Design Constraints

4.1 Optical Standards

As engineers, understanding standards and constraints is one of the most important parts of our profession. Standards, which are established guidelines or requirements that define how something should be done, help make sure that safety and regulatory laws are followed. Constraints, on the other hand, help you to define what is actually possible or realistic in the context of the problem you are trying to solve. In senior design, understanding these two things is crucial to our success and help us to create feasible projects.

While there are many standards that relate to our project, some of the most important to consider and understand are ISO 12233 which defines the method for measuring spatial resolution of an imaging system, USB3 Vision which defines how cameras transmit video over USB 3.0 connections, and IEEE 754 which defines floating point arithmetic which is used in image processing algorithms for precision in super resolution calculations. These three industry standards all relate to the camera array and address different parts of the super resolution process, being the definition of resolution, communication using USB 3.0, and image processing with floating point arithmetic. For optics purposes, we will explore two of these standards, ISO 12233 and IEEE 754, and see how each one really contributes to our project and help us through our senior design process.

First is ISO 12233. This is a standard that defines test methods for measuring the resolution of digital cameras and imaging systems. It is relevant to our project since we are attempting to use super resolution techniques and this standard provides a quantitative way to evaluate and improve image resolution. Super resolution techniques aim to reconstruct details from multiple low resolution images and this standard has standardized charts and methods to quantify the spatial resolution of a system before and after super resolution processing. This helps us to validate if our system is actually enhancing detail. Additionally, this standard can help evaluate

lens performance by inspecting different types of characteristics such as distortion, sharpness, and contrast loss which helps guide our optical design process. While we are comparing different approaches or arrangements of our camera array, this standard will help provide a benchmark to go off of and will ultimately help us determine the best configuration and most effective setup. So, while this standard does not seem like it's being used often, everytime we consult the resolution of an image or the effectiveness of a certain setup or technique, we are actually making use of this standard. Without this standard, measuring resolution would be much more difficult and would likely require the naked eye, which may not be able to pick up slight improvements in resolution depending on the image that is being used. Overall, this standard is not just important to our project but to any imaging system.

The second standard that affects our optical design is IEEE 754. This standard is the floating point arithmetic standard which is primarily used for representing and computing numbers in digital systems. While it is not directly related to image resolution like the previous standard, it plays an important role in numerical accuracy for our super resolution application. Super resolution is an inherently mathematically intensive process and when things like image alignment, filtering and convolution, and interpolation and reconstruction are done, high precision is required. Floating point arithmetic helps make sure that these processes are done accurately and more efficiently. Without this standard you would run the risk of introducing rounding errors which are present in other algorithms, precision loss, and overflow or underflow when dealing with large or small pixel values. IEEE 754 helps mitigate these risks by defining standardized floating point arithmetic formats, such as single precision and double precision, ensuring predictable results during our image processing process. Additionally, based on which level of precision we are looking for, we can use this standard to help balance speed and accuracy throughout the super resolution process.

To summarize, there are many different standards that relate to the context of our problem. The main ones we have considered help us define key parts of our project such as resolution and parts of the image processing train. Without these standards, finding a consistent way to evaluate the performance of our system or the effectiveness of our image processing techniques would be nearly impossible. For that reason, it is important to understand how these and other standards affect our design process.

4.2 Optical Design Constraints

One of the main constraints of this project is the budget. While there is no set budget, our project is attempting to find a low cost solution to the aircraft detection problem. For optics, this can result in many challenges. When trying to deal with low cost optical components, it can be hit or miss. For example, when researching cameras that are cheap, there may be specifications that are listed that look very good on paper, but once actually implemented, fall short of the expectation. These types of components typically tend to be pretty fragile and are generally not as reliable as higher cost, higher quality components. In addition, the affordable cameras that we deal with in this project typically come with a predesigned lens system which does not leave a lot of room for variation. The main limitation that cost has on our project though is resolution and frame rate. Many of the cameras that are available and affordable feature less than 3MP resolutions and usually less than 100 fps and that's if a global shutter is used, which costs more money. While they serve their purpose, higher resolutions and better frame rates would make for a much more

robust system that can have many more applications than the small niche we are trying to address.

The next major constraint is time. If we were allotted more time to do our project, the code that we could develop for our super resolution algorithm would be much better than it is now. A possibility that could've been explored is deep learning and AI enhanced super resolution techniques. This type of super resolution algorithm is easily

4.3 Hardware Standards

Our project relies heavily on the fact that we are using RF as a means to detect an aircraft. When it comes to RF, there are lots of standards and it is heavily regulated as all of the Radio Frequencies are in fact used for communication. Sending signals out on specific frequencies may interrupt communication between many different channels. Which could be things such as emergency communications or aircraft communications. As a result, there are many standards that we need to follow as the Federal Communications Commission has lots of rules regarding sending or receiving radio waves.

First, we will be following ITU-R SM.2152 which defines passive spectrum analysis and analysis methods. Some of the key technical parameters that we will need to follow are things such as minimum sensitivity. Which has to be less than or equal to -120dBm. This will ensure that we are able to detect weaker signals from an aircraft. We need a dynamic range greater than or equal to 80 dB. So that the system won't get destroyed from strong signals. A noise figure of less than or equal to 5 dB, which will help reduce loss. This standard also requires us to have RF shielding and interference mitigation. It states that passive monitoring stations must not radiate unwanted signals. The system may emit signals from things like the oscillator in the receiver, noise from the USB or improper grounding. By following this standard, we will be able to successfully analyze and monitor RF signals passively.

The next standard is ITU-R M.1461, which is spectrum sharing between radar and passive sensors. It states that we must not interfere with any communications such as air traffic controls or military radars. Which primarily operate from 1.2 - 1.4 GHz. It also has guidelines to not saturate or damage passive radar systems as radars can transmit from 1KW to multiple MW. They recommend some specifications so that the system may not get damaged such as having a receiver overload threshold of -20dBm max or having good antenna selectivity to filter stronger pulses.

Furthermore, ITU-R SM.2125 is a standard which defines techniques for detecting and identifying signals in a noisy environment. It talks about statistical signal detection methods, SNR estimation, and techniques such as noise cancellation. This standard is interesting as it provides a guideline that may be useful when we start prototyping. Our use case will not be incredibly noisy, but by providing a reference signal it may be helpful to help us determine what is noise versus what may be an aircraft. This standard is a great guideline to help us do so.

Next, our antenna will have to follow IEEE standards to work properly. We will have to follow IEEE STD 149-1979 and 291-1991 as they have methods and procedures for measuring things like antenna radiation patterns, polarization and overall performance.

The system will also have to follow IEEE 754-2019 which is the standard for floating point signal processing in RF uses. This is important as it provides the most accurate signal processing and data collection for our use. Our NUC mini PC will also have to follow this standard to ensure precision when we are making our algorithm. It also has storage standards such as IEEE 1667 which is the standard for storage protection. Ensuring that it is safe from outside sources and the data is secure. Of course, we will also be following IEEE 802.11ax and IPv6 which are the standards for wireless connection and IP addressing and security functions that our computer will use. Since we will be broadcasting through Starlink it is important that we are able to adhere to these standards.

4.4 Hardware Design Constraints

Firstly, we have to discuss our ITAR restrictions as one of the main goals that our sponsor requires is that it has no ITAR restrictions. If we do not comply with ITAR restrictions it means we wouldn't be able to export any hardware or software components without a state department license. Which would violate what our sponsor wants most.

One of the restrictions that we fall under is USML Category XI, Military Electronics. Which states that our system cannot be advanced enough to detect classified military aircraft such as stealth airplanes. With this kind of system we needed as much capability as possible. However, with our component selection, it will not have the ability to detect something that even active radars struggle to detect. Some of the capabilities of something that may be ITAR restricted are things like high gain directional antennas, spectrum analyzers that can work in the Gigahertz and above frequency range, or simply working in military frequency bands.

The next restriction is USML Category XV, Space systems and satellites. This is a restriction surrounding surveillance equipment and ensuring that it doesn't become airborne. This is something that we are able to follow because the system will be ground based and therefore bypass this restriction. This also naturally leads into USML Category XII, Fire control, optics and sensors. Which covers items that may be used for electronic warfare such as RF detection systems or advanced spectrum monitoring equipment. The best way to mitigate this is to make sure that we are not detecting military aircraft as that would violate ITAR.

We are able to mitigate our ITAR restrictions by simply using commercially available parts that do not have any military RF components. Our part selection follows all ITAR restrictions and are commercial off the shelf. They have also been approved by Roy, who is an expert in ITAR restrictions and government dealings.

Furthermore, this is something that we would like to use world wide and will have to follow not only United States communication protocol, but other countries as well. We will try to follow the International Telecommunication Union regulations so we can avoid trouble with any governments or military.

A few of the other constraints pertain to the environment. These are likely going to be set up in the jungles of South America. As a result, it will be very hot and very humid, similar to that of Florida. In order for us to be successful, we will have to be very careful with our enclosure design and how we will store the battery. If it is too hot all of the equipment may not work which would be considered disastrous. Not only that, but it can't be an open enclosure as there may be things like bugs or animals that will try to inhabit it. We will have to make it a closed box system and manage the heat from there.

Another constraint is of course cost. We are trying to keep the system cost as low as possible for the sake of commercializing the product for our sponsor. When it comes to the prototype, we have almost no cost constraints as our sponsor has already paid for our equipment. However, when it comes to the final design we are going to try and reduce the running costs of the system. Perhaps we have a spectrum analyzer that is too strong, or our antenna is too good for our use case. These are all things that we will discover as we work towards our final product.

A key physical constraint is the hardware requirement for the BB60D itself. The Spike software does not start its SCPI server while in demo mode. This means that any Python automation using SCPI communication requires the physical presence and successful initialization of a BB60D device. This constraint does not permit full development and testing of automation code in the absence of hardware and demands controlled access to the device for development.

A second important physical constraint addresses the real-time data rate capabilities of the BB60D. The sensor can generate spectral data at high throughput—up to 24 million samples per second. This volume of data is far greater than what a mobile phone like the Galaxy S24 can display or process in real-time. The data, therefore, must be downsampled, averaged, or selectively reduced before being sent to the phone. This constraint shapes the structure of the Python backend to include filtering or aggregation phases that render the content mobile-friendly without overwhelming the interface or depleting battery life.

Client-side mobile constraints must also be taken into account. Continuous data streaming, plot updating, and user interaction on the mobile client result in excessive battery and temperature stress. Galaxies like the S24 are powerful but not designed for aggressive, sustained data visualization tasks without throttling. This necessitates the need to implement control such as pause/resume buttons, dynamic refresh rates, or data windowing in order to reduce usage on the mobile client.

Lastly, the measurement accuracy of the BB60D is dependent on calibration and environmental conditions. Although Spike performs inherent correction and calibration routines, any practical application of the measurement data must take these limitations into account. Results may vary due to frequency response, gain variability, and extraneous interference, so measurements must be verified against known references or compensated based on calibration data.

Finally, we have the design constraint of implementing Starlink. Which is an important part in the success of our project. Our sponsor has specifically asked us to use Starlink as it is a new service that is being offered and will allow us to transmit the data throughout the world.

Especially where it will be placed in locations that are extremely remote and may not have line of sight to something like a cell tower.

4.5 Software Standards

First, we need to find what kind of software standards may affect our system. There are many parts in our system that will have to abide by these standards such as the spectrum analyzer, the computer, and the communication method that we choose.

First, we have IEEE-754 which governs how floating point arithmetic is handled. This is important for anything that may require computing power. Such as the spectral density, Fast Fourier Transform, or any other computational intensive task. All of our systems have to follow this standard for deterministic execution, low latency processing, and precision in floating point arithmetic. There are also POSIX.1b Real- Time extensions. This supports scheduling inter-process communication, and time sensitive execution for Unix-like OSes. Which is important because our system will later be implemented on Unix OS for the sake of increasing the efficiency and reliability of the system by eliminating any redundant background tasks.

These standards enable predictable behavior for tasks such as signal acquisition, pre-processing and detection. If the timing of these events are off we could have things like corrupted data. This would be catastrophic for our system as we are worried about how accurately we are able to detect an aircraft.

Next are our cybersecurity and data protection standards. Our data is sensitive because it holds aircraft data. This goes back to our ITAR restriction where we aren't allowed to detect military aircraft. Naturally, we need our data to be secure so that we don't raise any flags amongst our allies. First, we will talk about NIST SP 800-53 and NIST SP 800-171. This standard provides structured controls for access control, encryption, system integration, and logging. Our system will be communicating over a network or server of some sort. These two standards ensure that the data confidentiality and integrity are preserved during transmission. Our data will be highly sensitive so it is important for us to diligently follow these standards. For our system because it is aviation related, we have to ensure compliance with Common Criteria ISO/IEC 15408. We also have to follow FIPS 140-2 encryption.

For our software to interact properly with our hardware as well with other programs we need to VITA 49. Meaning we will have to properly adhere to well defined APIs and interface standards. The aforementioned standard, standardizes how the spectrum data is formatted and transported across different systems. This will ensure that our data will be easily transferred and used in our application.

Our system also has Standard Commands for Programmable Instruments. Which is a standardized command language used to control test and measurement equipment such as our Signal Hound. It is built on IEEE 488.2 which defines a consistent human readable command. This will allow us to configure and control our test equipment. We will also ensure that we adhere to RESTful API conventions especially because we are developing a monitoring system. This means we will have to comply with formats such as JSON, XML, or SHR. Which is what our system will use.

In terms of our usability, it will have to be as easy to use as possible. There will be many government agents using our system and will require it to work as smoothly and simply as possible. This is ISO 9241, ergonomics of human system interaction. This will ensure that the user interface of our system is intuitive and ensure the most efficient use and decision making. This also leads into accessibility standards. We will have to make it usable and easy to use for anyone that may have a disability. It will also need multi-language support for it to be used by our supporting countries that may need to use it.

In constructing an automated spectrum analysis system using the Signal Hound BB60D, Spike software platform, and Python scripting, compliance with and adoption of pre-existing technical standards are crucial in making accuracy, interoperability, and performance possible. They are the foundation on which both the communication with the instrument and subsequent data processing are built. Without these pre-existing frameworks, any kind of integration between the hardware and software would likely be unstable, inaccurate, or non-scalable.

The deployment of an automated spectrum analysis system utilizing the Signal Hound BB60D, the Spike software suite, and its corresponding SDK is strongly rooted in the practice of adhering to widely established technical standards. Such standards offer not only operational interoperability but also long-term maintainability, accuracy, and scalability across many hardware and software environments. At its heart is the employment of SCPI (Standard Commands for Programmable Instruments), a vendor-neutral command language that is used heavily on electronic test and measurement instruments. SCPI offers a standardized ASCII-based syntax for commands that enables a client application written in Python to talk freely to the BB60D using the Spike software. Commands such as "FREQ:CENT 315e6" to adjust center frequency or "TRAC? TRACE1" to retrieve spectrum trace data reflect the system's reliance on an extensible, readable, and unambiguous control protocol. Use of SCPI promotes orderly communications habits and allows the system to be ported onto other SCPI-supported equipment, not just Signal Hound's products.

In addition to device-level interactions, the system employs the IEEE 802.11 standard for wireless networking. This standard governs Wi-Fi communication protocols, which are critical in current implementation for immediate transmission of data from the host PC (on which Spike is run) to a mobile device, for example, Samsung Galaxy S24. Wireless implementation of IEEE 802.11 offers guaranteed, low-latency, and high-throughput data transfer over local networks to support mobile visualization and client-side monitoring without a physical connection. Design focuses on responsiveness of real-time streaming of spectrum data, and without a 802.11 compliance guarantee, data latency or packet loss can jeopardize the monitoring system's efficiency.

IEEE 754, the de facto binary floating-point arithmetic standard, is integrated with the communication layers. The levels of power returned by the Spike API are expressed as floating-point values in dBm and need to be treated with mathematical precision to prevent rounding errors and for the sake of analytic correctness. The use of numerical libraries like NumPy by Python, which are IEEE 754 compliant, enables precise manipulation of such measurements. Accurate floating-point computation is particularly critical in baseline setting,

computing threshold deviations, and identifying signal spikes because a single error would lead to false positives or miss significant anomalies.

Data transfer and accessibility-wise, the system complies with modern web and network standards for flexibility and real-time performance. HTTP/1.1 specification, as defined in RFC 2616, accommodates REST-based communication between backend Python and the client-side interfaces like web browsers or mobile dashboards. It accommodates clients to post requests for spike events or spectrum data through well-understood HTTP verbs like GET and POST. Where permanent, real-time streams of data are needed, the system may employ WebSocket communication, as prescribed by RFC 6455, to maintain a permanent, full-duplex TCP connection. This is particularly convenient when it comes to streaming time-critical spike events in real-time.

To support data interchange, the system employs structured data formats conforming to current standards. JSON (JavaScript Object Notation), as specified in RFC 8259, is used to serialize and ship data between the backend and outside clients or remote databases such as Firebase. JSON's simple syntax, ease of use, and support in almost all development environments make it the ideal data format for logging timestamped spike events. For data that can be analyzed offline or stored to spreadsheets, CSV (Comma-Separated Values), as specified by RFC 4180, offers a plain, tabular format that can hold human-readable data and which is compatible with analytical tools such as Excel or pandas in Python. All these specifications combined constitute a good framework to provide effective control, communication, storage, and visualization of the spectral data through the system.

Airbnb JavaScript Style Guide

The Airbnb JavaScript Style guide is widely used in the industry and provides clear rules for things like how to name variables, where to place brackets, how to write functions, and how to structure files. By following this guide, our code remains easy to read, even when switching between different parts of the system like the server and the map UI. In a project like this, where signal data is coming in and being displayed in real time, keeping the code simple and clean is important for debugging and improving performance. This standard also helps us avoid errors. For example, by using consistent spacing and syntax, we’re less likely to have bugs caused by missing semicolons or inconsistent logic. Although it’s not enforced by a compiler, following this style guide makes our team’s workflow smoother and reduces confusion when we need to review each other’s code.

ISO/IEC 25010 software quality standard

On a higher level, we also considered ISO/IEC 25010, which is a standard that outlines important qualities that good software should have. These include things like functionality, reliability, usability, efficiency, maintainability, etc. For example, under the "efficiency" category, we will make sure our background service doesn’t overload the server by only sending relevant data. Under "usability," we will keep the front-end interface simple so that users can clearly see aircraft locations and interact with settings without confusion. And for "reliability," we will add fallback behaviors in case the spectrum analyzer disconnects or fails to send data. ISO/IEC 25010 gave us a general checklist to follow to make sure we weren’t just focused on making the system work, but also making it work well.

We will also keep code maintainability in mind by using version control and organizing our project into clear folders with separate responsibilities. For example, all signal processing code will be kept separate from user interface logic. We used meaningful names for variables and functions, and leave comments in areas that are important or hard to understand. This helps make sure that if we, or anyone else, come back to the project months later, we can understand how it works without starting over from scratch.

Finally, while not an official coding standard, we will also be sure to include consistent testing practices. We created test cases for our image processing portion of the project to detect aircrafts and ensured the output matched expected results. Even something as simple as highlights where an aircraft is in an image is important in catching bugs early. If we add more complex logic later, like filters or classification algorithms, we plan to continue this habit of testing in small steps. To summarize, our project benefits from software standards both in how we write our code and how we think about the system as a whole. Following the Airbnb Style Guide helped us write cleaner and easier to read JavaScript, while ISO/IEC 25010 helped us stay focused on building software that performs well and can be maintained over time. Without these standards guiding our design choices, we would be more likely to run into bugs, confusion, or code that's too messy to build on later. Following standards strictly may initially be viewed as a hindrance, but it is quite the opposite. Using standards will give us a stronger foundation to build something reliable and useful.

4.6 Software Design Constraints

Our software will also play into our hardware design constraints. We have to manage our resources well because if our system is incredibly power hungry we will reduce the amount of time it can be out in the field. The amount of maintenance that it will require will shoot up and reduce the amount of false alarms. The computer will have to run with limited compute power. It will have to have an extremely optimized algorithm that will reduce the amount of calls to processing power while increasing the amount of reliability and accuracy of the system. We can do this by limiting the amount of CPU utilization for the spectrum analysis and processing. This will also allow us to control the amount of heat and stability that the system may output.

Having a system that runs cooler as opposed to hotter increases the lifespan of all the components. As a result, we will cut down on maintenance costs. We can also restrain the RAM usage to ensure there is a real time buffer and computing availability. Of course, by restraining all of these important computer characteristics, we will improve the thermal performance of it and reduce the amount of power draw that the computer may use.

The system must be able to handle the amount of data that we are transmitting. The Fast Fourier Transforms will be very large data sets and sweeps. We want to be able to scan as many frequencies as possible, which means that when we do our sweeps our data sets will be extremely large. The system must be able to read, process, and provide a buffer in real time. The system must also have low latency. This will be a real time monitoring sensor and as a result the latency must be as low as possible. This will come with a great cost to our heat and power management. For our use case where we are using a NUC this will be a smaller problem. However, as we move further along with designing and we try to reduce the amount of power

usage this will be a much larger problem. If we plan to integrate it onto a Raspberry Pi or similar controller we will have to optimize our algorithm as much as possible.

We will be using Linux as we get further in our development as it has a larger range of control over the computer in comparison to windows. There may be a lot of background tasks that Windows automatically enables that we can disable in Linux. This allows us to control our priority and handling of the CPU itself. We can prioritize the critical tasks and our algorithm which will help with all characteristics of our system.

Next, our data logging and storage is a big constraint of our use case. It must be processed and sustained locally on the processing unit. The storage must support writes without blocking the reading thread. It has to be able to implement the data rotation automatically and get rid of old outdated data. The data will also have to be structured very well, giving information such as the time, frequency, the amount of gain that we receive. This is one of the things that our software must provide for the customer as it allows them to make a decision as to how they want to interdict.

As previously mentioned, the software must comply with NIST SP 800-53 and SP 800-171. It is going to be used in a military setting and as a result the data is quite sensitive. It must be restricted only for authorized users. It will have to also be encrypted for wireless transmission and not reveal where the sensor is hiding.

While the system is deeply based on standardization, there are numerous implementation constraints that must be taken into consideration in order to facilitate proper, secure, and efficient operation. One of the most evident limitation lies in the SCPI communication latency and throughput. Although SCPI has a standardized syntax, it runs over TCP/IP and is interpreted by the Spike SCPI server, which in turn causes latency depending on network conditions, server response time, and command parsing effectiveness. This can make the system unresponsive, particularly when running at high polling frequencies or when there are several parameters requested sequentially. The sequential processing and limited buffer space of SCPI commands also suggest that high-rate data acquisition would have to be throttled or multithreaded with caution to avoid bottlenecks or lost messages.

In wireless environments, reliance on IEEE 802.11 networks introduces variability of performance with exposure to environmental noise, congestion, or signal quality deterioration. Wi-Fi is inherently prone to variability of bandwidth and latency levels, particularly over shared or public media. During streaming of real-time data to mobile devices like a Galaxy S24, network instability of short duration might impede user experience, delay sending spike notices, or introduce errors during synchronization of visualization and backend data.

Floating-point operations, although standardized by IEEE 754, introduce their own set of issues. Cross-platform floating-point equivalence cannot always be guaranteed, especially when comparing values generated on different hardware architectures (e.g., ARM vs x86). This could result in slight differences in baseline calculations or threshold comparisons. Additionally, depending on the floating-point precision (single vs double), the performance and memory requirements could differ, especially when handling large data streams or long historical logs.

From a software perspective, Firebase Realtime Database usage, while offering excellent ease of integration, is bounded by free-tier limits. Firebase imposes write and read limits, connection limit per second thresholds, and bandwidth thresholds. Under high-traffic scenarios or applications with prolonged monitoring periods, these limits are likely to be surpassed rapidly, requiring billing plans to be upgraded or architectural changes. Besides, Firebase projects are built with "test mode" security rules, which allow public read/write access to the database by default. This configuration, unless changed before deployment, offers a severe security vulnerability, allowing unauthorized clients to post fake data or delete existing entries.

On the client side, HTTP and WebSockets usage introduce asynchronous management and state management on the client, particularly in the scenario of real-time dashboards. Without proper implementation, this can lead to data races, stale visualizations, or even overpolling of APIs. Similarly, JSON's loose schema, while very convenient, is not strongly typed, and therefore data inconsistency will happen if clients or APIs return unexpected structures or omit some critical fields.

Finally, synchronization of timestamps between devices is an apparently trivial but critical limitation. Since the host system's clock is used to locally stamp spike events, any time differential between systems (e.g., the BB60D host and a mobile client) will result in ambiguity when comparing data for visualization or relative analysis. Without a central time-sync service like NTP (Network Time Protocol), it can affect timelines' accuracy or comparisons with history.

Finally, we have discussed all of the important software design constraints. They play heavily into our hardware design constraints as we are worried about things such as the compute power. We have also taken into account the security and integrity of the data that we are transferring. By following our standards and constraints we will be able to provide data that is encrypted and safe for our allies to use.

Chapter 5: Comparison of ChatGPT With Other Similar Platforms

AI has quickly become one the most popular and revolutionary technologies of our generation. With platforms, like ChatGPT, making interacting with AI easier than ever, it is important to understand its strengths as well as its limitations. As engineers, we need to be able to understand how to use AI as a tool since it is a technology that will be part of our future. To showcase how helpful or harmful the use of AI can be when dealing specifically with senior design, each of us conducted a case study specific to our area of study.

Case Study 1:

When first starting out on this project, figuring out the scope of the photonics and optics side was a huge challenge. Figuring out a way to detect an aircraft with our background and skill set turned out to be rather challenging, especially since we were limited to passive detection means, not to mention anything involving laser technology was immediately taken out of consideration due to ITAR restrictions and the legality of sending a laser beam at an aircraft. Since I didn’t understand what super resolution really was, how it worked, or how it could even be considered a detection method, I asked ChatGPT to explain. I asked “What is super resolution and how does

it work?" It was able to provide a very detailed explanation and listed different techniques including multi-image super resolution techniques, single image resolution techniques, and AI aided super resolution techniques. The explanation, in hindsight, made a lot of sense but, in the moment, left me wondering about how it was actually done and what hardware would be needed to do it.
The next prompt I had for ChatGPT was focused on which cameras would best suit my project's needs and why. I asked "What affordable cameras would make the most sense to use in a super resolution camera array and why?" ChatGPT then provided a detailed breakdown about 3 different cameras including a strength and weakness comparison in the context of the project, their price ranges, and the type of software I would need to run each type of camera. From here, I learned about the importance of frame rate, shutter type, and resolution. These 3 factors would determine the best choice for my project. Following this, I prompted ChatGPT to explain and compare shutter types since I wasn't too familiar with the options. It then provided a detailed description explaining rolling shutter and global shutter, what type of cameras they were associated with, and their applications. Here, I learned that global shutters worked best for capturing a scene with a moving object since the shutter captures the whole scene at once. This type of shutter, however, was mainly associated with CCD cameras, which ChatGPT also indicated were much more expensive than the other type of camera, CMOS. After further explaining the scope of the project and the limitations and constraints I was under, ChatGPT was able to provide other types of camera options, again with their strengths and weaknesses. This is where we found details about the cameras we discussed in chapter 3.

The next step was to figure out what code we would need to actually attempt to do super resolution. Here, I asked ChatGPT "Can you provide a code that can do super resolution?" From here, ChatGPT provided a python code that used opencv, a computer vision library from python. Although it provided a brief explanation of what the code was doing, I did not understand most of it when I looked at it. After asking ChatGPT to explain it in more detail, it was able to provide a very methodical breakdown of what each section of the code was doing and why it was doing it. I then copied the code it spat out to try with the cameras we had purchased at the time to see if it worked. On my first attempt, I attempted to use the code with one usb camera and this is where I ran into problems, the cameras were not being detected by the code. In response, I asked ChatGPT to rewrite the code so that the first thing it does is check if there are cameras connected, then capture an image, then do super resolution. ChatGPT provided a new set of code and I attempted to run it. Here, I discovered that the main reason the code was failing was because of my computer's webcam. Once I turned it off, the code actually began to work. With one camera, I was able to successfully detect it, capture an image. The code stopped working when it got to the super resolution part of the code because it required at least 2 images. The next attempt, I ran the code with 2 cameras connected via usb directly to my computer. In this case, the code ran all the way through, but the output super resolution image was weird. It was very stretched out and choppy. From here, I asked ChatGPT what was going on.

ChatGPT explained that in order for super resolution to be done properly, the cameras had to be capturing as close to the same scene as possible. It even provided the percentage of image overlap that the cameras should have between their field of views. While those numbers were pretty arbitrary for me, I did attempt to run the code again with the cameras closer like ChatGPT suggested and the resulting image was more in line with what was expected. Although the outer

edges still had a slight chop/shift, the main focus of the image appeared to have gone through the super resolution process. To further troubleshoot and understand what was happening, I described the resulting image to ChatGPT and asked it to evaluate what was happening. The AI explained that the code, which was utilizing opencv, was using feature matching to align the images so that it could then undergo the super resolution code, this is what was leading to the choppiness in the output image. After this, I wanted to verify that the output image was actually a super resolution image, since I couldn't really tell at a glance. Here, I submitted 2 images, one was a single camera image and the other was the output from the super resolution code, and asked ChatGPT if the 2nd image was actually a super resolution image. ChatGPT explained that the 2nd image had twice as many pixels as the first image so it was technically a super resolution image even though it didn't look like it with the naked eye. It also then reinforced the definition of super resolution and explained that with better equipment or more cameras, a better output would be possible.

Similar processes like this have happened and continue to happen throughout my time in SD1. Since this class is very self-learning based, AI like ChatGPT are invaluable to my experience. Before starting this project, I had no idea where to even begin and ChatGPT was able to provide well detailed explanations about the scope of my project, what type of hardware would be necessary, and even code. Additionally, if I ever ran into trouble either when thinking of a possible solution or when investigating which products to purchase, ChatGPT was able to guide me in the right direction and truthfully saved a lot of time and money. ChatGPT was also excellent for organizing information, especially when doing comparisons. I frequently used ChatGPT to help me organize my ideas and even make some of the comparison tables that are used in this document.

There were plenty of times ChatGPT was not perfect though, especially when prompted with very general prompts. Sometimes ChatGPT would provide information that wasn't correct because it didn’t properly understand the prompt in the way that I understood the prompt I was providing. This really reinforced the importance of self validation of the outputs ChatGPT provides. If I blindly followed ChatGPT’s outputs and tried to do this project, I could guarantee it would not be successful. To prove this, I asked ChatGPT to come up with a passive aircraft detection device that used optics to capture an image of a plane at 5000 ft for a senior design project. The solution it provided was a stereo vision setup but the calculations that accompanied it were slightly unrealistic. Firstly, the distance between the two cameras that was necessary to capture an image of a plane at that altitude was too large to be viable for a senior design project. Second, the solution it proposed involved IR cameras which are way out of budget for a senior design project. Thirdly, and most importantly, it did not consider any sort of environmental induced aberrations.

All in all, ChatGPT has been extremely helpful throughout the senior design process. It has helped me understand a complex topic like super resolution and has built up my understanding of the imaging field. It has greatly improved my efficiency, especially when organizing information and finding resources to further investigate the topics I had unanswered questions about. It is, however, not perfect, and has taught me the importance of verifying information and not to take its output as the ultimate truth. Through this process, I have also learned how to better prompt AI to get the result that I want/expect. Before I was very general in the type of prompts I provided

and now I get very specific about what I ask from the AI model. In the end though, AI like ChatGPT is just a tool. By properly leveraging this tool as an engineer, I was able to learn at a quicker rate, increase efficiency, and problem solve better throughout my time in senior design.

Case Study 2:
One of the problems for the electrical engineers when looking for the Displayed Average Noise Level (DANL) of the spectrum analyzers. For a lot of the sources that we used it was not very apparent where to find that information. Seeing as it is a fairly common and important specification of spectrum analyzers. Which prompted a question for chatGPT being, "What's the best way to advertise DANL." This was prompted as there may be lingo or jargon involved in this field that isn't apparent to someone that doesn't particularly know much about it. ChatGPT took the question from a marketing perspective. It mentions using a standardized format, dBm/Hz or specifying key conditions such as Preamp, RBW or Frequency range. It also mentions using graphical representations may help drive our point. Or that we need to highlight the real world benefits and even compare against competitors. This was not quite the engineering approach that was specifically wanted from ChatGPT. It tried its best, but ultimately may have misled someone that may have not known what they were looking for. In our case, this was immediately seen as it took the wrong way and promptly led to a google search to see if there were any other ways to find this important specification.

Case Study 3:
One of the examples relates to the power requirements of the passive radar for detecting aircraft. I have very little experience with RF devices in general and there is a lot of variance between different products based on how complex they are, how accurate they are, whether they are active or passive, etc. Chatgpt was an excellent resource in finding not only the kind of information I was looking for, but also where to find that information by having it provide me the sources that it was getting the information from, allowing me to check and read them for myself so I could better grasp how much power a device like this would require. This led directly into searching for power sources for the passive radar.

Chatgpt was also very helpful when it came to comparing the data that I had found. Once I had found multiple power sources that met the requirements for the passive radar I needed to compare and contrast them, normally this would require a lot of in-depth searching and take a lot of time to create a table that easily compares every metric that I was looking at for making the decision between them, but by prompting Chatgpt to compare the power sources and their usefulness when it comes to powering a passive radar, it put all of the data I could need and then some into a table that directly compares each of the power banks based on output, battery capacity, battery type, price, solar input capability, weight, and size dimensions. These were the factors that were taken into account when choosing which of the 4 options I had to pick from for powering the passive radar.

Furthermore, Chatgpt also helped with the writing process for comparing each of the power banks. After prompting it to compare and contrast them for use in powering a passive radar for aircraft detection, it gave me the table, and also listed out pros and cons for each of the devices in

bullet point format, giving me ideas and making the writing process for actually explaining my decision a bit easier, since I had all of the important points that I needed to hit listed out in separate sections for each of the power banks and a small summary of what situations they would be better or worse than the other options for this specific device.

Although this paints its strengths very well, I also wanted to test its weaknesses. To do that I began by adding some bias into prompts essentially asking the same things as before. One at a time I prompted Chatgpt to compare and contrast each device and explain which one is the best for use in powering a passive radar for aircraft detection, but for each of these prompts I specified that it needed to explain why a different option was the best option for our purposes-for example: “explain why the Westinghouse 100W is the best option for powering a passive radar”- and doing that style of prompt for each of the four options. Chatgpt did a great job arguing and explaining exactly what it was asked, but this is a weakness of the AI because it means that the information it gives is biased to what you ask it, so regardless of what the best option for a passive radar actually is, it will just tell you what it is asked at the time.

This is a major problem with AI because it means that regardless of how helpful it is, it is important to be very careful with the prompts that you ask it, because the slightest amount of bias in a question will result in a biased answer, further compounded by the fact that previous answers in a conversation with it will influence later answers, so if you ask it biased questions in the beginning, and later ask it an unbiased one in the same topic, it is likely to give you an answer based on the previous bias given.

Ultimately the important takeaway from this study is that it’s important to use Chatgpt for what it is, a tool. It is an amazing tool for summarizing and explaining information, it’s a great tool for getting ideas for what to research on your own and where, and it’s really good at helping in the writing process when you get stuck and need to move forward with your thoughts. But reliance on it to do the research and work for you will result in biased information and decisions that should be made through critical thinking, instead being made based on what has been done before, rather than what the best option truly is.

Case Study 4:

One of the major challenges in developing our multi-camera super-resolution imaging system was understanding the concept of super-resolution itself and determining the best lens configurations for our setup. Initially, I had a limited grasp of how super-resolution worked and why it was necessary for our project. I turned to ChatGPT with a simple question: "How does super-resolution work?" The AI provided a detailed explanation, describing how multiple low-resolution images can be combined computationally to reconstruct a high-resolution image. It introduced me to different methods such as multi-frame super-resolution, deep-learning-based enhancement, and interpolation-based upscaling. While the explanation was helpful, I needed more specific guidance on why we should use multiple cameras with different lenses instead of identical ones.

After prompting ChatGPT with "What are the advantages of using different lens types in a multi-camera system?", it provided a thorough breakdown of why a diverse lens configuration would yield better results than a uniform one. I learned that using a mix of several lens systems,

such as: telephoto, wide-angle, zoom, and fisheye lenses would allow us to capture both fine details and a broader field of view. Telephoto lenses would provide high magnification for distant objects, ensuring that small targets like aircraft remain visible and detailed. However, their narrow field of view meant that they alone would not be sufficient for large-scale scene capture. Wide-angle lenses, on the other hand, could provide a much broader field of view, making them ideal for capturing large portions of the scene and ensuring that all relevant objects appear in the final composite image. Zoom lenses would offer flexibility, allowing us to adjust focal lengths depending on imaging conditions, while fisheye lenses would enable panoramic coverage, reducing blind spots.

Before using ChatGPT, I had assumed that using identical lenses across all cameras would simplify the system and reduce calibration complexity. However, its explanation helped me realize that diversity in lens selection would significantly improve the quality, accuracy, and usefulness of our final image. This led me to rethink how the cameras should be arranged within the array and how different lens types could be combined to create an optimized imaging system.

Once I understood the importance of using multiple lens configurations, I needed to compare specific lens models and specifications. ChatGPT proved to be an incredibly useful tool for this step, as it was able to process large amounts of data and organize it into structured tables for easy comparison. Instead of manually sorting through specifications across different manufacturer websites, I provided ChatGPT with a list of lens systems and their key characteristics. It then generated a structured table summarizing focal length, aperture size, field of view, distortion characteristics, and cost. This streamlined the decision-making process and allowed me to compare lenses side by side without spending excessive time gathering and organizing the data manually.

ChatGPT also played a key role in helping me write and structure explanations for our research findings. After reviewing the comparison table, I asked it to summarize the pros and cons of different lenses for our use case. It provided a clear, structured analysis of each lens type, outlining situations where a given lens would be most effective and where it might introduce challenges. This helped me articulate why certain lenses were better suited for specific roles within our camera array. By having these pros and cons clearly laid out, I was able to more effectively justify our design decisions in technical discussions and documentation.

Beyond research and writing, I also tested ChatGPT’s limitations and potential biases. To see how it responded to leading questions, I deliberately asked it biased prompts, such as "Explain why telephoto lenses are the best option for super-resolution imaging", and noticed that it was able to construct a compelling argument for that viewpoint—regardless of whether it was truly the best option. I then repeated the experiment with other lens types, and in each case, ChatGPT provided strong arguments supporting whatever I had asked it to justify. This demonstrated an important limitation: ChatGPT does not independently determine the best option—it tailors responses to the framing of the question. If I had relied solely on its initial answers without critically analyzing the information, I could have been misled into making suboptimal design choices.

Another way I tested ChatGPT’s capabilities was by asking it to compare and contrast different camera configurations and justify which was best. When I phrased the prompt neutrally, it

provided a balanced analysis of each configuration, listing advantages and drawbacks. However, when I asked it to argue why a curved array was the best option, it presented a detailed case in favor of that configuration, despite the fact that there were still trade-offs to consider. This reinforced the need to approach ChatGPT's outputs with critical thinking and independent verification, ensuring that decisions were based on objective analysis rather than AI-generated justifications tailored to specific prompts.

Ultimately, ChatGPT has been an invaluable tool in helping me understand complex imaging concepts, compare technical specifications, and structure research findings efficiently. It provided clear explanations of super-resolution imaging, lens configurations, and image processing techniques, which allowed me to make well-informed decisions throughout the project. However, this experience also reinforced the importance of using ChatGPT as a research aid rather than a final authority. While it was incredibly useful for summarizing information, organizing data, and generating comparative analyses, its responses were limited by how questions were framed and needed to be verified with external research.

In the end, ChatGPT significantly improved the efficiency of my research and decision-making, but it also highlighted the importance of critical thinking and independent validation. By leveraging AI as a supplementary research tool rather than a definitive source of truth, I was able to enhance my understanding, refine our imaging system design, and ensure that our final decisions were based on objective, well-researched information.

Case Study 5:
In the early stages of our project, we were tasked with building a system to monitor and track aircraft signals using a spectrum analyzer and a live web interface. Since this topic combines hardware, software, and real-time data, I found myself struggling to fully understand how everything worked together, especially the part about extracting useful information from radio signals. I knew that the spectrum analyzer was going to provide frequency data, but I didn't have much experience with signal processing, or what steps came after receiving that raw data. This is where I first used ChatGPT to help bridge the gap.

I started by asking a basic question, "How do spectrum analyzers work?". ChatGPT gave me a good response that provided a solid overview, which got me a little more comfortable with the tool we were working with. I then moved on to more detailed questions, which were "How can I use frequency data to detect aircraft?" and "What filters are used to clean up signals?" The responses surprisingly really helped guide my understanding of low-pass filters, moving average filters, and peak detection, which I was only somewhat familiar with. I was able to take these suggestions and dig deeper into each one through additional reading, and even began testing some of the filtering methods in FL Studio. While ChatGPT wasn't always perfect, it gave me the foundation I needed to know what terms to look up and what techniques were worth trying.

As our project moved forward, we needed to figure out which algorithm or combination of algorithms would work best for cleaning up noisy frequency data from the analyzer. I used ChatGPT again, this time asking it to compare different options for signal filtering. It listed moving average filters, low-pass filters, and band-pass filters, with explanations of what each one does and what kind of signals they're good for. This gave me lots of information that helped me decide which ones made the most sense to use in our specific case.

That said, ChatGPT also had a few limitations that became obvious as I kept using it. When I asked it questions like “Which filter is best for aircraft signal detection?” it always gave me a balanced answer but never actually picked a side unless I forced it to. And when I did prompt it to argue for one approach over another, it would do so convincingly, almost too convincingly. I realized that it would support whatever side of the argument I wanted it to, even if that wasn’t the most logical choice. For example, I asked it to explain why a low-pass filter is the best choice for real-time signal tracking, and then asked the same thing about a band-pass filter. It supported both arguments equally well, which showed me that it wasn’t evaluating the actual best option, just responding to the way I phrased the question.

In conclusion, using ChatGPT improved my research a lot. It saved me hours I would have spent digging through unfiltered search results, and helped me find answers to specific questions that would have taken much longer to piece together on my own. It felt like I had my own personal assistant researching important topics for me and giving me the information I was asking for directly, versus searching through an entire article to find the parts I needed. It didn’t always give the perfect solution, but it made the entire research process faster and more focused. As long as I kept verifying the information I got, it became one of the most helpful resources I used throughout the project.

Case Study 6:

When it came time to figure out how we were going to detect aircraft using only the RF signals, I knew I had to find some kind of anomaly detection algorithm that would actually work for our use case. There are many different options available, and I didn’t want to waste time messing around with something that was too complex or not even meant for this kind of data. So I used ChatGPT to help pick the right option for our use case.

At first, I used it to get a list of possible algorithms that made sense for unsupervised detection. I asked what would work well with real-time signal streams, and it gave me a clean breakdown of the most common options: Z-Score / Statistical Thresholding, Isolation Forest, One-Class SVM, DBSCAN, Autoencoders, PCA, and a few others. Each one came with a short summary of how it worked and what kind of data it was best for. That already saved me a bunch of research time, and from there, I could start narrowing things down.

I knew I needed something lightweight and simple, no complex model training, no huge memory requirements. ChatGPT helped me rule out options like Autoencoders and Isolation Forest early on. They might work well in other scenarios, but not for a system that's supposed to run out in testing environments with limited power and computing capability. Z-Score stood out right away since it’s dead simple and doesn’t assume anything about the data. Plus, it works with both batch and streamed data, which gave me some flexibility depending on how we wanted to handle data transfer later on.

After deciding on Z-Score, I started prototyping and testing. I created a few sets of fake RF data with embedded anomalies and ran them through a basic implementation of the algorithm in Python. I also tested it in batches, since I already handled real-time streamed testing in another section, and wanted to see how well it held up when analyzing a chunk of signal data all at once.

I made sure to simulate scenarios where anomalies lasted multiple points in a row, like 10 or more data points spiking, to mimic something like an aircraft slowly coming into view.

The results from testing the algorithm were great. Z-Score picked up on the spikes easily without needing any heavy preprocessing. Because it's so fast and simple, I didn't run into any latency issues, even when scaling up the number of data points. It gave me a reliable way to test different threshold levels and figure out what counts as normal or suspicious. ChatGPT helped there too, I asked how to tune the threshold and it gave me some general guidelines on balancing sensitivity and avoiding false positives.

Overall, ChatGPT proved to be a useful tool during the research and early development phase of the anomaly detection portion of the project. It provided a clear overview of the available options and helped narrow down which methods would be the most viable for a lightweight, low-power system like ours. By quickly identifying the pros and cons of each algorithm, it allowed me to make informed decisions without having to spend hours reading through technical papers or documentation. This accelerated the prototyping process and gave me more time to focus on testing and integration, which ultimately helped move the project forward more efficiently.

Chapter 6. System Design

6.1 Camera Array Hardware Diagram

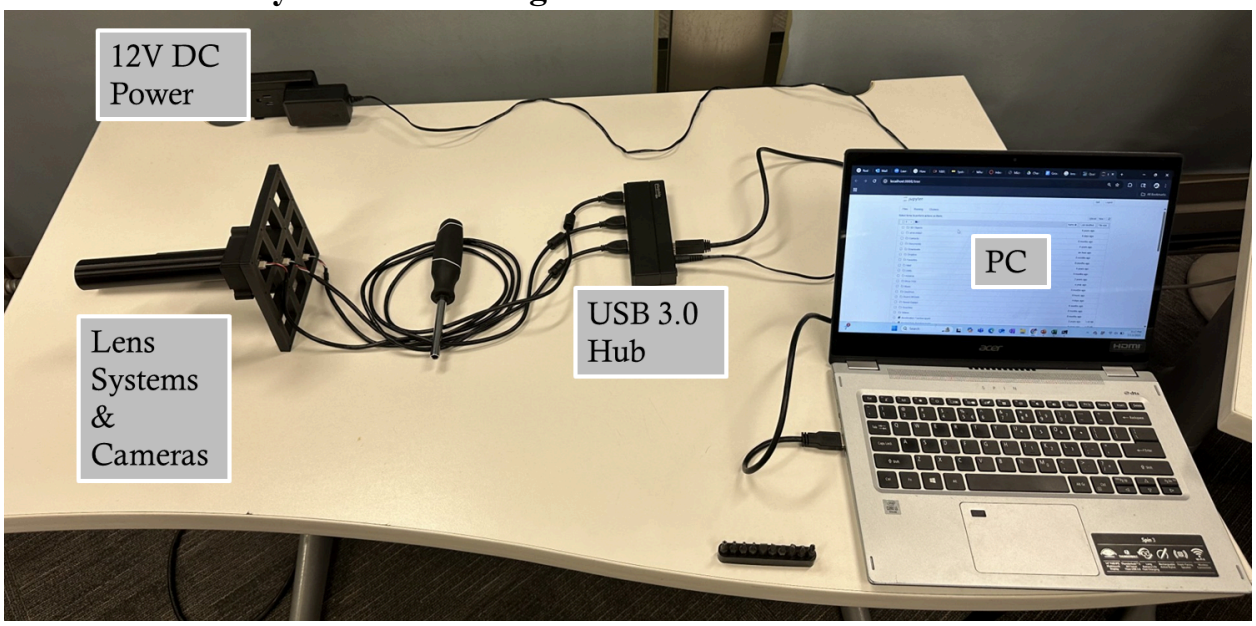


Figure 6.1 *Camera Array Preliminary Hardware Setup*

Above is a diagram that displays the setup for the preliminary camera array. While this diagram is rather crude, it shows the necessary components of the system. There is a different lens system for each camera. Each will provide a differing angular resolution with slightly varying field of views for image enhacenement. As can be seen, the OV9281 cameras are connected to a USB hub which features a 12V power supply. The USB hub is connected to a standard wall outlet to

get power and is connected to the laptop via a USB 3.0 connector. The USB hub and connector allow for up to 5 Gpbs of data transfer speed which is what we need to effectively stream video content from the cameras.

6.2 Spectrum Analyzer Hardware Diagram

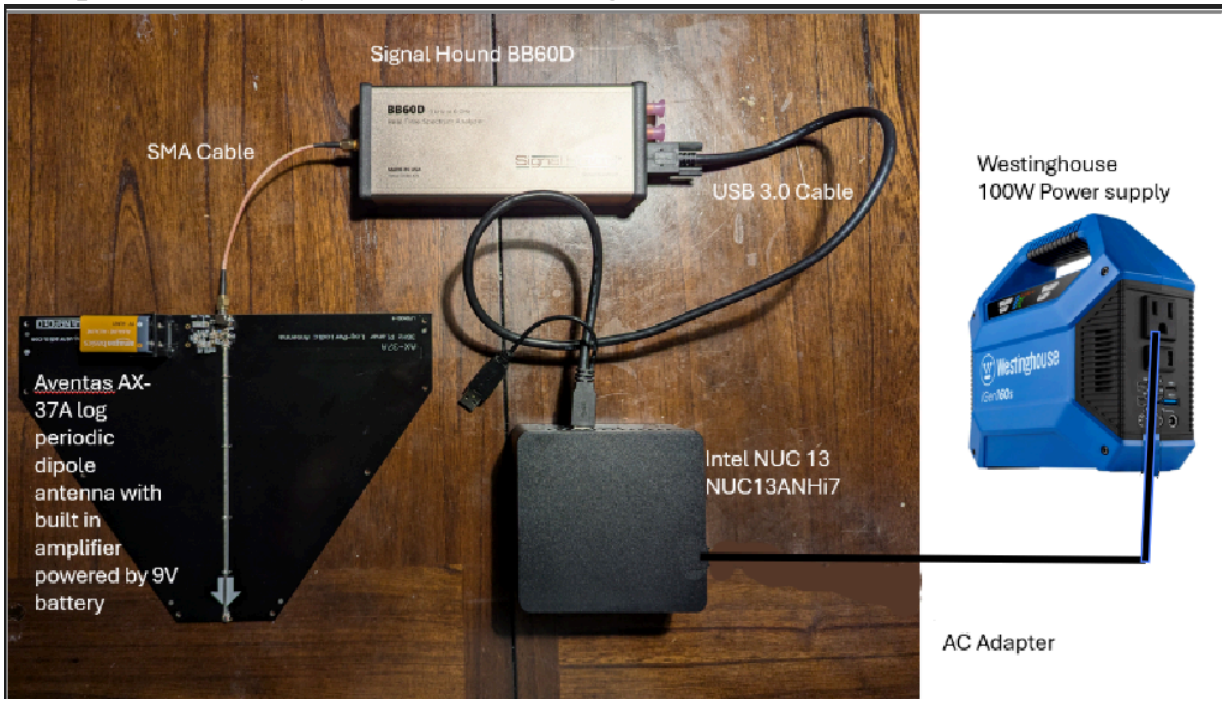


Figure 6.2 *RF Device Hardware Setup*

As shown from our wiring diagram up above there are a few details that need to be discussed. First, we will start with the antenna. It requires a 9V battery in order for the built in preamplifier. For our prototyping, we will be using and replacing the battery for the sake of simplicity. Aventas / WinRadio does offer an antenna / DC solution to the power. Meaning we could plug it into our power supply. However, for our use case we will leave the 9V battery as it reduces the amount of wiring that we may use in our testing. Next, we will talk about the connection between the spectrum analyzer and the spectrum analyzer. Of course, this will use the standard SMA cable. The SMA is incredibly common and used in almost all RF applications. We used a cheap Amazon SMA cable, which provided us plenty of resolution and accuracy for our testing. Next, we have a standard USB 3.0 cable to connect between the NUC. This will work with any USB 3.0 port. The spectrum analyzer is also powered by the USB 3.0 port, which removes a lot of complexity in our wiring. Finally, we will use a standard power / AC adapter cable to connect to the power supply. By having this our entire system will be powered and work flawlessly.

Chapter 7: Software

7.1 Data Collection and Transfer

Signal Hound Spike is the built-in control and analysis software for Signal Hound instruments such as the BB60D spectrum analyzer. From its foundation in high-speed RF measurement and

analysis, Spike offers a graphical user interface (GUI) as well as an automation interface for programming and controlling BB60D hardware in real-time. If Spike is connected using USB 3.0 with a BB60D, Spike communicates directly to it and performs actions such as frequency sweeping, bandwidth setting, signal demodulation, and real-time visualisation of data. Users have direct control of the BB60D from Spike GUI, with parameters such as center frequency, span, reference level, resolution bandwidth (RBW), video bandwidth (VBW), and detector mode set up, displaying live spectral data on screen. Spike environment also supports plugins for computing channel power, occupied bandwidth, harmonic distortion, etc.

In addition to direct control, Spike also offers a full SCPI (Standard Commands for Programmable Instruments) interface for high-level automation of communication with the BB60D using Python. This allowed us to create a script that can directly add data points to a database using Firebase, specifically by constantly polling the data collected by the antenna and spectrum analyzer and adding any spikes in the data past a threshold of 10dB past the norm into the database with the date, time, and magnitude of the spike. The code used creates a whole system for identifying spikes in radio frequency signals and recording them with test data and stores such anomalies into a Firebase Realtime Database. It begins by importing primary modules, which are time, logging, random, and datetime, and Firebase's admin SDK to securely interact with a database. It establishes wide logging both to file (signal_monitor.log) and console for runtime visibility, critical for long-running background operations like signal monitoring. It then establishes the critical parameters, such as the center frequency (315 MHz), frequency span (20 MHz), reference level (-20 dBm), and resolution bandwidth (10 kHz). These parameters emulate the configuration of a real RF receiver system like the Signal Hound BB60D. In addition, detection parameters are defined with 10 dB above the current baseline as the threshold and 10 measurements being utilized as a window for dynamically calculating the baseline or noise floor. Near-real-time sampling of data is assured by one-second polling.

The Firebase database URL and credentials are subsequently established using a JSON credentials file from a Google Firebase project and a database URL pointing to a Realtime Database instance. All signal monitoring and Firebase communication features are wrapped in the class FirebaseSignalMonitor. This class, when initialized, establishes internal settings for the signal context and for detection parameters and subsequently attempts to connect to the Firebase backend. It loads credentials from the provided JSON file and uses the Firebase Admin SDK to construct the app, initializing it with a reference to the named database collection. Following success, it configures the application to output detected signal spikes to the cloud. In the class, log_spike() functionizes a spike event for formatting and posting as a human-readable date and time, numerical timestamp, along with the magnitude of the spike as a JSON object. This data is pushed to the database with a unique ID key on every spike detection so that a clear log of out-of-bounds events is preserved.

The simulate_signal_data() function simulates real RF measurements by generating magnitude values randomly. Eighty percent of the time, it produces a signal value that varies around the current baseline noise level (simulating normal environmental RF), and twenty percent of the time it adds a greater value, indicating a spike or anomalous transmission. It has a rolling window of baselines, which are updated only when the signal value is considered normal. This ensures the baseline to reflect true background noise and not be affected by spikes. The system

flags a magnitude crossing the baseline by over the threshold (10 dB) as a spike and marks the event, simulating detection mechanisms in spectrum monitoring software. For log clutter reduction, regular readings are logged only after 10 measurements, while spike events are logged immediately.

The start_signal_monitoring() function is the signal monitoring loop. It writes the start of monitoring to a local text file (monitor_active.txt) and enters an infinite loop that calls the simulate_signal_data() method. When it detects a spike, it calls the log_spike() function to write the event to Firebase. It also writes status messages to the local file every 100 readings, like total readings taken and spikes found so far. This is a persistent progress log and can be used to track the activity of the system even in the event of the application crashing or being shut down. The script also exits cleanly on interruptions or failure. If it is interrupted with a KeyboardInterrupt (i.e., the user presses Ctrl+C), it terminates normally and logs a shutdown message to the log and status file. If some other unexpected error, it logs a full traceback for debugging purposes.

The main() function is the program's entry point. It initializes the FirebaseSignalMonitor using the preconfigured credentials and database URL, and starts the monitoring loop. Like the other functions, it includes aggressive error handling and logs any important events. The script is coded in such a way that it can be executed directly (because it includes the test for __name__ == "__main__"), so it's simple to execute as a background process or encapsulate into larger monitoring systems. Overall, this code is a clean but sound solution for monitoring an emulated RF environment and logging anomalies to a cloud database, leaving room for adaptation to real-world signal acquisition hardware. Its modularity, clear logging, and cloud support make it well-suited for remote monitoring applications, classroom demonstrations, or first-pass prototyping of spectrum awareness systems.

7.2 Spectrum Analyzer SDK

One of the central components of this project is the use of a spectrum analyzer to monitor and interpret radio signals. This device plays a vital role in identifying aircraft positions by capturing and analyzing frequency data emitted by various sources. In order to interact with the spectrum analyzer, the project relies on its built-in Software Development Kit (SDK), which acts as a bridge between the hardware and the application layer. The SDK provides the necessary tools to communicate with the device, retrieve signal data, process that data, and incorporate it into the rest of the system’s workflow.

The SDK is a collection of prebuilt libraries, drivers, and documentation that exposes functions allowing developers to directly access the spectrum analyzer's hardware-level capabilities. These functions typically include device configuration, frequency sweep control, data sampling, and signal extraction. In this project, the SDK is used to automate the process of scanning for signals, collecting real-time data, and passing that data to the backend server for processing and visualization.

Use and Integration

The integration of the SDK begins at the lowest level of the system architecture. A dedicated software module will be developed to interact with the SDK directly. This module is responsible

for initiating scans, setting signal thresholds, and selecting appropriate frequency bands. Once a sweep is completed or when a signal event is detected, the SDK allows us to retrieve the raw signal data, which often includes information such as frequency, signal strength, and timestamp. In the context of this aircraft tracking project, the primary use of the SDK is to collect position-related information transmitted via known frequency ranges. Many aircraft emit identifiable signals using protocols such as ADS-B (Automatic Dependent Surveillance–Broadcast) or other communication standards. The spectrum analyzer, with the help of the SDK, listens to the frequency space where these signals occur and extracts relevant data when patterns matching these transmissions are found.

After collecting this data, the software module processes it to convert raw signal values into meaningful information. This step may include calculating signal strength, determining proximity based on intensity, and associating frequency patterns with known aircraft identifiers. Usually, the SDK provides built-in methods to assist with filtering and transforming this raw data into more usable forms. For our project, the most important information will be if an aircraft is in a designated area we have selected. Once processed, the resulting data is formatted and passed to the backend server using a real-time communication protocol. This step is vital for keeping the web-based interface updated. Users interacting with the system see the results of this process in the form of aircraft icons on a map, each one representing a signal that was captured, decoded, and translated into a spatial position with the help of the SDK.

Data Flow

The SDK supports real-time and near real-time data operations, which is essential for a system like this that relies on accurate, live data. Polling intervals will be configured to ensure that as soon as new signal data is available, it is immediately handled and forwarded through the system. This design minimizes latency and allows the web interface to reflect real-world aircraft movement almost as it happens.

In most configurations, the SDK also provides access to historical signal data, buffer management, and device diagnostics. These features can be useful for debugging and improving system performance. For example, if signal dropouts or weak transmissions are noticed, we can access lower-level logs and diagnostic information to determine whether the issue lies in the spectrum analyzer hardware, its placement, or the surrounding environment.

Development and Testing

During development, the SDK's simulation mode can be used to test the full data pipeline without requiring a constant live signal input. This can help with debugging the server and refining the data formatting logic. If we don’t use the simulation mode, recordings of real signal sweeps can be captured and replayed through the SDK to test the handling process under more realistic conditions.

The SDK also includes documentation and usage examples that will help describe the development of this integration module. Depending on the language support offered by the SDK, bindings for languages like C++, Python, or C# may be used. This separate background service will provide a simplified way for the Node.js server to interact with the spectrum analyzer SDK. This service will handle all direct communication with the SDK and return formatted data to the

main server. By offloading this work, we keep the front-end fast and responsive while the heavier signal analysis runs in the background, ensuring efficiency.

Role in the System

The SDK is not only a tool for interacting with hardware. It is a critical part of the larger system. The accuracy, stability, and performance of the entire project depend on how well the SDK is integrated and how reliably it delivers signal data. Care must be taken to handle edge cases such as signal interference, incomplete sweeps, or device disconnection. The software layer built on top of the SDK will include error handling and recovery mechanisms to ensure the system continues to function even in less-than-ideal conditions.

To maintain consistent performance, the system will monitor the flow and responsiveness of data coming from the SDK. If performance drops or inconsistencies are detected, alerts can be generated to prompt administrative attention. This could potentially be especially helpful in a system where delays or missing data can degrade the quality of our results.

SDK Configuration and Customization

The SDK gives us a lot of settings that we can change to control how the spectrum analyzer works. We can adjust things like how wide of a range it scans, how sensitive it is, how often it collects data, and how long it listens. For our project, changing these settings helps us pick up aircraft signals more accurately. It also lets us ignore signals that aren't important, which makes everything run smoother and faster.

The SDK might work on different operating systems like Windows or Linux, depending on how it's made. This matters because we need to make sure it works well on whatever system we end up using. If the SDK only works on one platform, it might be harder to set things up or give us fewer choices later. If that's the case, we might use a tool that lets us run it in a virtual environment to get around that problem. Before using the SDK, we also need to look at its license. The license tells us what we're allowed to do with it. Some licenses let us use the software freely, while others have rules about sharing or changing it. Good documentation is also really helpful. If the SDK comes with clear instructions and examples, it makes building our system a lot easier. If the documentation is bad or missing, it can slow us down and cause mistakes.

7.3 Algorithms for Signal Analysis

Once the spectrum analyzer collects frequency data, we need to process that information in a way that helps us understand what it means. The raw data alone doesn't give us anything useful unless we apply algorithms to organize, detect patterns, and match that data to known signal behaviors. These algorithms help us figure out if an aircraft is nearby, how strong the signal is, and whether the data is trustworthy. In this project, we can explore a few different kinds of algorithms that focus on filtering, detection, classification, and prediction. Each one plays a different role in turning signal data into clear results for the user.

Signal Filtering

The first thing we often need to do is clean up the raw data. A lot of the frequency information picked up by the spectrum analyzer will include background noise or interference from other sources. To reduce this and focus on the important parts of the signal, we can use filtering algorithms. There are many different filtering algorithms, each with different purposes and existing to remove different frequencies. Here are some of our options:

Moving Average Filter

One basic option is a moving average filter. This type of filter smooths out the signal by averaging a set number of recent values. It helps eliminate short spikes or random changes in the signal that don't represent real aircraft transmissions.

Instead of reacting to every small fluctuation, the filter gives us a more stable line that reflects the overall trend of the data. This is helpful when trying to track steady signals from aircraft, especially if the environment has a lot of background noise. By reducing those quick jumps in the signal, the moving average filter makes it easier for the system to focus on what's important and ignore sudden, random spikes that would otherwise confuse the analysis.

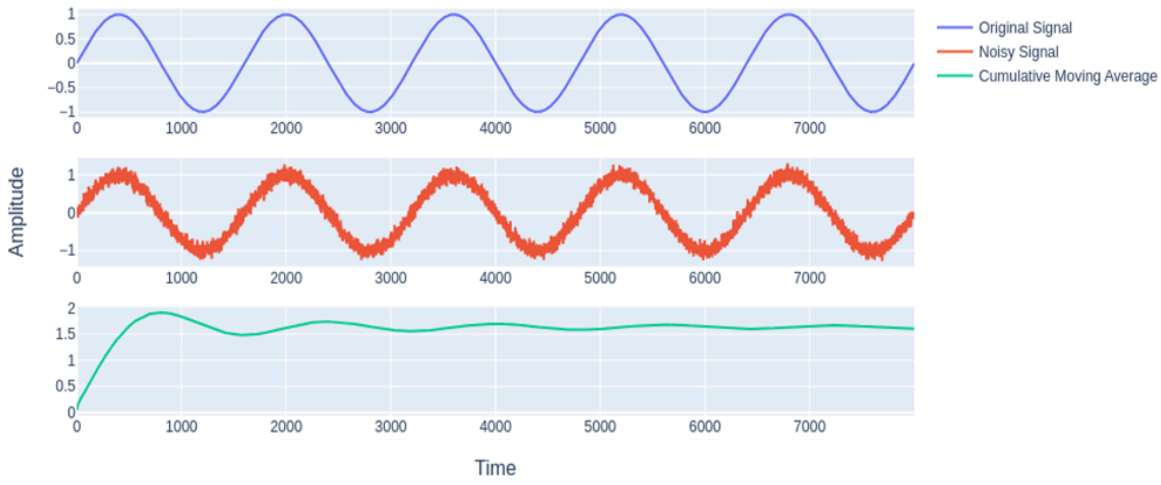


Figure 7.1 *Moving average filter*

This filter is also easy to implement, which makes it a good choice for a first step in signal processing. Since it only requires averaging a few numbers at a time, it doesn't use much computing power, so we can run it often without slowing the system down. For this project, we can apply the moving average filter to incoming signal data in real time. That means as each new point comes in from the spectrum analyzer, it gets smoothed together with the points that came

before it. The result is a signal that changes more gradually, making it easier for later parts of the system, like detection or classification, to work more accurately.

Low-Pass Filter

Another option is a low-pass filter, which cuts out the high-frequency noise above a certain point, letting only the smoother and slower-changing signals come through. For aircraft tracking, this helps us focus on steady signals that are more likely to be real transmissions instead of random fluctuations. By cutting out the high frequencies, we are limiting our data to only frequences that are under a certain threshold, which will help us make more meaning of our data and get rid of unnecessary noise.

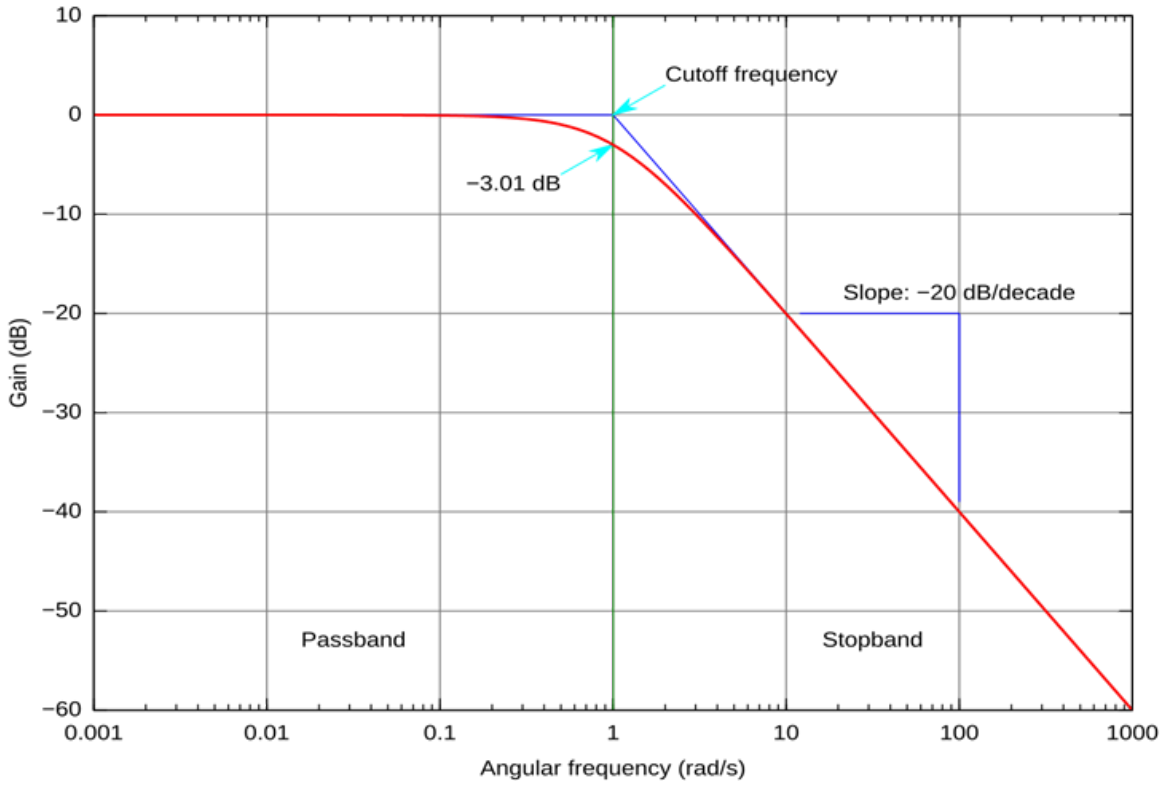


Figure 7.2 *Low-Pass Filter*

Low-pass filters are useful when we expect real signals to change slowly and stay stable over time, like with aircraft that send out steady signals while flying through monitored space. These filters reduce the chance of picking up fast, sharp changes that could be caused by other wireless devices or background interference. If we were to process every part of the frequency spectrum, including the high-frequency noise, it would take more time and possibly confuse the system with signals that aren't actually from an aircraft.

We can also adjust the filter's cutoff frequency depending on the situation. In a quiet environment, we might allow more frequencies through. In a busier setting, we could lower the

cutoff point to block out more of the noise. This makes the low-pass filter flexible and something we can tune as the project grows or moves into different testing areas.

Band-Pass Filter

We might also use a band-pass filter to only listen to specific frequency ranges. For example, if we know that aircraft send out signals in a certain range, we can filter out everything else. This speeds up the system and makes our signal analysis more accurate.

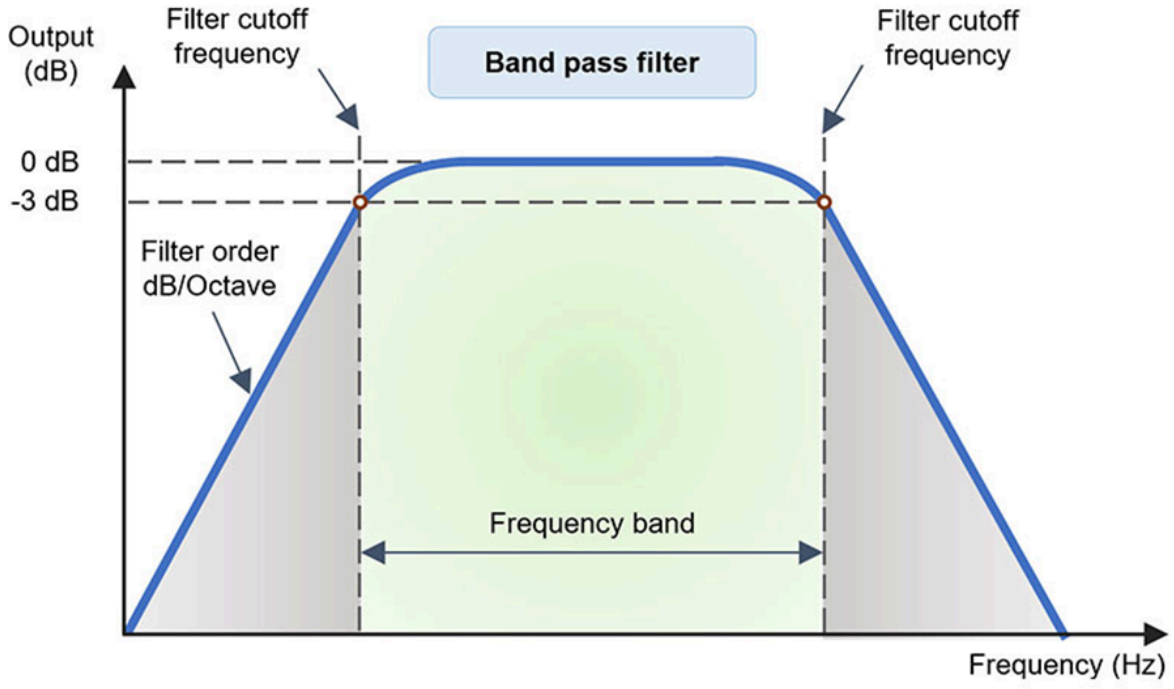


Figure 7.3 *Band-Pass Filter*

Band-pass filters are especially useful when we already know the exact frequency band used by the signals we want to track. In aircraft tracking, many signals occur in a very specific frequency range. By using a band-pass filter, we make sure that our system focuses only on those signals while ignoring all other data that falls outside the target range. This makes it easier to detect and analyze aircraft transmissions without being distracted by unrelated signals.

Another benefit of using a band-pass filter is that it helps the system work faster. Since we're only processing a limited range of frequencies, we don't waste time or computing power on data that doesn't matter to us. This is important for real-time systems like ours, where fast response times help keep the user interface up to date. When the system is getting constant updates from the spectrum analyzer, it's helpful to only pass along the useful parts of that data stream.

Band-pass filters can also improve accuracy by reducing false detections. Random noise or signals from other devices can appear across the frequency spectrum. If we let all of that data through, we might misidentify something as an aircraft when it isn't. By narrowing our focus, the band-pass filter helps make sure that we're only analyzing the signals that are actually related to our goal. We can also choose different frequency ranges based on the type of aircraft we're tracking or the environment we're working in. For example, commercial planes might use one range, while smaller aircraft use another. Being able to adjust the filter to match the situation makes it a flexible tool that we can adapt as our project continues.

Signal Detection

Once the data is cleaner, the next step is detecting whether there's actually a signal present. This is where detection algorithms come in. One common method is called energy detection. This involves measuring how much power or energy is present in a given section of the frequency range. If the energy crosses a certain threshold, we can assume a signal has been detected. We can adjust this threshold depending on how sensitive we want the system to be.

Another option is matched filtering. This method compares incoming signals to a known pattern or shape. If the match is strong enough, the algorithm counts it as a detection. This method can work well if we're only looking for specific types of signals and we already know what they're supposed to look like.

In some cases, we may want to look at frequency peaks, which are sharp increases in signal strength at certain frequencies. Peak detection algorithms scan through the data to find these spikes. If a peak is found in a known aircraft signal range, it may mean a plane is nearby and transmitting.

Signal Classification

After detecting a signal, we need to figure out what kind of signal it is. That's where classification algorithms come in. These algorithms try to label the signal as something meaningful, like "aircraft," "noise," or "unknown."

One way to do this is using threshold-based classification, which simply groups the signals by how strong they are or where they fall on the frequency scale. For example, we might say any signal over a certain strength in the ADS-B frequency range is probably from an aircraft. This is a simple method, but it can lead to mistakes if the environment is noisy or if there are other devices using similar frequencies.

Another method is using machine learning to train a model that can recognize patterns in the signal data. For example, we could collect a bunch of labeled signal data, some from aircraft and some from random noise, and use that to teach a classifier (like a decision tree or a neural network) how to tell the difference. Once trained, this model could analyze new signals and make a guess about what they are. While this method is more complicated, it has the potential to be more accurate, especially in unpredictable environments.

We could also look into time-series classification, where the algorithm looks at how the signal changes over time. Aircraft signals tend to have certain rhythms or repeating patterns, and those

patterns can help us identify them. Time-based algorithms would look at more than just the strength or frequency, they would analyze how the signal behaves over a short period.

Location Estimation and Tracking

In addition to detecting and identifying signals, we may want to estimate where the aircraft is located. If we only have one sensor, this becomes more difficult, but we can still make some guesses based on signal strength. A basic algorithm for this is received signal strength indication (RSSI) tracking. RSSI tracking involves measuring how strong a signal is and assuming that stronger signals come from aircraft that are closer.

We might also look at triangulation if we ever expand the project to use multiple sensors. In this case, we measure the time it takes for a signal to reach different points and then use geometry to figure out where it came from. This method is much more accurate, but it needs more hardware and more advanced math to pull off.

Another interesting algorithm for tracking is called Kalman filtering. This is used in a lot of real-world tracking systems, like GPS or radar. It helps smooth out noisy position estimates and make predictions about where something will go next. If we know an aircraft's signal keeps showing up at certain times, we can use a Kalman filter to guess where it's heading, even if the signal drops out for a few seconds.

Anomaly Detection

Sometimes, we want to know not just what's happening, but what's going wrong. That's where anomaly detection comes in. These algorithms look for unusual signal behavior that might mean a sensor isn't working correctly, or something unexpected is happening in the environment.

We can use statistical models to figure out what "normal" looks like for the system, and then flag anything that falls way outside of that range. For example, if signals usually appear at a steady rate but suddenly stop for a while, the system can raise a warning. We can also use clustering to group similar signals together and spot anything that doesn't belong. This type of analysis might not be used all the time, but it can be very helpful for debugging or improving the system's accuracy over time.

Conclusion

In this project, we don't need to use every single algorithm listed above, but we can use what we learned researching them to pick the best option depending on the actual data we end up receiving later on when we fully tackle this project. Filtering and detection are must-haves because they're the foundation of the signal pipeline. Classification helps us decide what to show on the map. Tracking and anomaly detection might come later, but they can make the system smarter and more reliable.

Each of the algorithms here helps turn raw data into something meaningful. Without them, the spectrum analyzer would just be a device picking up noise. With the right set of tools, we can turn that data into something useful, like showing a plane's location, checking if it's inside a safe area, and updating the display in real time. Choosing the right algorithms helps us make our system more accurate and easier to use.

7.4 Front End

The front end of our web application is designed with a clear focus on usability, accessibility, and performance, ensuring an efficient and intuitive user experience. The frontend will either be built using a framework such as React or Angular, or just use Vanilla JavaScript.

Desktop User Interface Design Overview

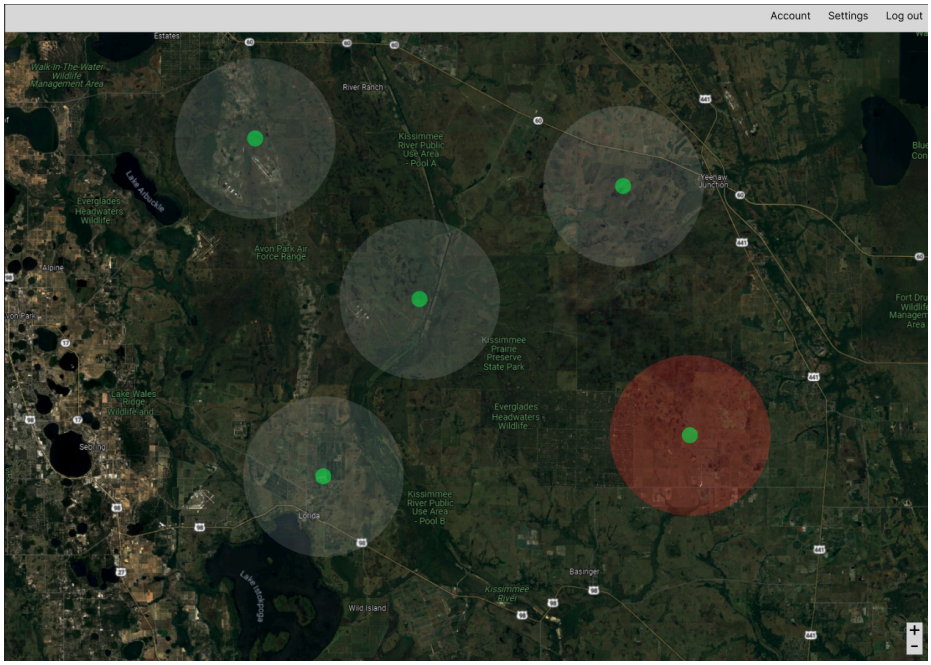


Figure 7.4 *Rough draft of Desktop design*

The desktop user interface (UI) is made up of two primary sections. On the top of the page, a control panel consisting of clearly labeled buttons for users to modify their account, change application settings, and log out. This intuitive layout helps users navigate quickly and easily. On the rest of the page, a geographical map is displayed, loaded either through a third-party API or stored directly on the web server. The map features circles that show the range of remote nodes, changing color when an airplane is detected from them. When the sensor nodes detect aircraft, the circles representing the sensors change colors, to let the user know when an aircraft is detected. Users can also click on the circle to view information about the sensor and logs of detections.

The account management and settings dialog boxes allow users to modify their accounts and change application settings. The account management dialog clearly displays user account information and allows simple modification of details, such as usernames and passwords. The settings dialog box offers application customization options, letting users select different map visualization modes, configure notifications of detected aircraft, and personalize the application to their individual needs and preferences.

Since it's important that the web application is accessible on mobile and web, and needs to be reliable and smooth, performance is very important. Lazy loading, a technique that delays loading assets and resources until they are needed, is one of the many different methods we'll utilize to ensure smooth performance.

Mobile User Interface Design Overview

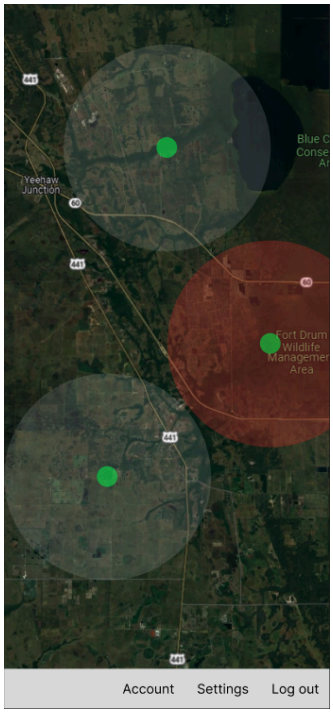


Figure 7.5 *Rough draft of Mobile design*

The mobile user interface (UI) will maintain the same functionality as the desktop counterpart using a response design. Using CSS media queries, the web application will be able to dynamically change formats, so the user can use the website on any device. The main part of the screen will be the geographical map, with circles overlaid showing remote sensor nodes and their ranges. When an aircraft is detected, the circles change color, to let the user know what remote node detected the aircraft. On the bottom part of the screen is a control panel consisting of buttons to let the user manage account settings, change application settings, and log out.

The mobile view if the web application changes menus designed to work with mobile controls, context-aware elements, and larger interactive zones. Elements such as the map and sensor node icons are scaled to the size of the screen, and performance remains important since some mobile devices can be underpowered.

Accessibility

When designing the front end of the web application, accessibility is a very important factor to consider. Accessibility allows users to access and interact with websites, regardless of limitations. Settings should exist to change the user interface, by including adjustable font sizes

and high-contrast display modes to accommodate users with visual conditions or different lighting conditions, especially considering outdoor scenarios. Interactive buttons and elements, such as the navigation buttons and sliders, should adapt to the screen-size, making it easy for mobile users to interact with them. They should also be clearly labeled, so users understand what they do, and be accessible with keyboard controls, for people who don't have access to a mouse or prefer a keyboard. Alerts from the web application should include an audio and visual indication, so users with no speakers or a bad screen can receive them. In regards to the high-resolution imaging capabilities, it's important that the interface supports zooming and panning, either with mouse controls, keyboard controls, and touch screen controls, allowing detailed analyzation of images without loss of usability. Additionally, the software needs to ensure compatibility across browsers and operating systems (like Windows, macOS, Android, and iOS), letting users access the web application on any device.. Lastly, implementing clear error messages and easily accessible support documentation within the interface will significantly reduce user frustration and improve overall system reliability.

The Web Content Accessibility Guidelines (WCAG) outline different levels of accessibility ratings for web applications. The WCAG keeps four principles in mind, perceivable, operable, understandable, and robust.The first WCAG first level is A, the minimum level. Level A covers the most essential accessibility considerations, making sure that web application content is accessible to a wide range of users, but some barriers to accessibility may still cause issues. The second WCAG level is AA, the common target. Websites that adhere to level AA provides broader accessibility, while still meeting the previous levels requirements. The highest WCAG level is AAA. Websites that aim for AAA level offers the most accessibility, addressing a wide range of barriers. Level AAA is very difficult to achieve, and may not be practical. In designing the front end of this web application, we are aiming for level AA. We believe level AA strikes the right balance of accessibility and practicability for our needs.

Accessibility in the front-end of the web application extends beyond visual design and navigation. An essential part of this includes ensuring compatibility with common assistive technologies such as screen readers like JAWS (Job Access With Speech), NVDA (NonVisual Desktop Access), and VoiceOver, as well as speech recognition and magnification software. To achieve this goal, we will use Semantic HTML, HTML tags that define the structure and meaning of websites. ARIA (Accessible Rich Internet Applications) are sets of roles and attributes that define ways to make web content and web applications more accessible. ARIA will be used in the website design to achieve the goal of more accessibility. Additionally, multimedia components such as instructional videos or audio alerts should include synchronized captions, audio descriptions, or transcripts, to ensure that users who are deaf or hard-of-hearing, or those working in noisy environments, can fully access information. Also, implementing controls that allow users to disable or minimize animations and transitions of elements is necessary to accommodate those sensitive to motion, including users who may experience dizziness or other related disorders. Considering our project's potential international application, localization should be included into our design by providing support for multiple languages. Users should have the ability to easily switch languages, with the interface designed to handle text expansion and size changes during translation. Accessible documentation, including user manuals and inline help, must also adhere to accessibility guidelines, providing clear instructions

and information in various formats such as accessible PDFs, HTML, and simple text, ensuring usability for all users.

While testing, we will gather valuable user feedback related to accessibility, and make changes to the web application when needed. We predict that we won't cover all accessibility needs at first, and the application will require a lot of accessibility changes and modifications throughout development.

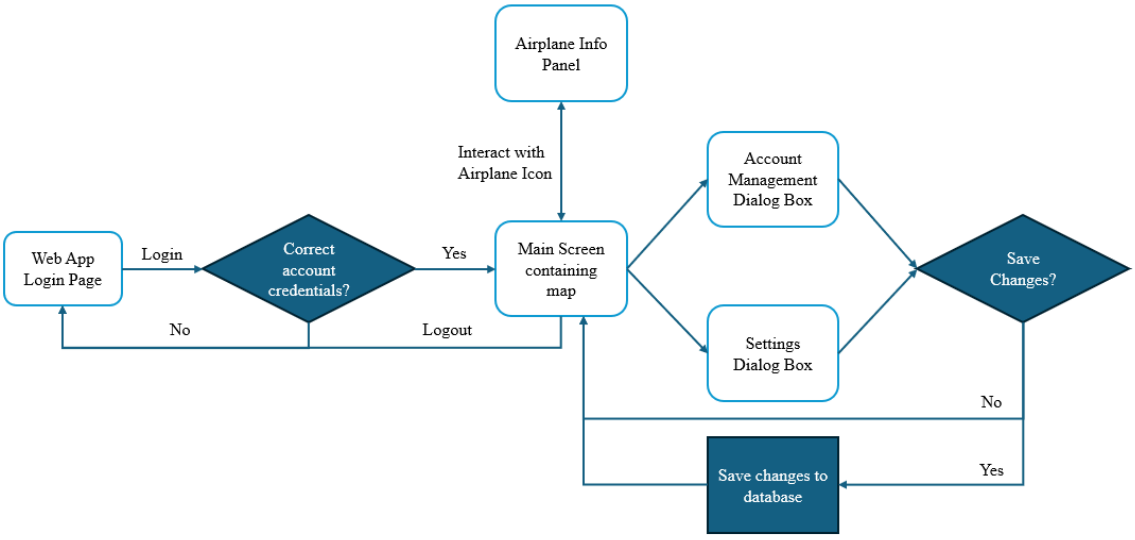


Figure 7.6 *User Flow Diagram*

Main Screen Details

This User Flow Diagram (UFD) outlines the navigation process within the web application. Initially, users arrive at the login page, where they must enter their credentials. If these credentials are incorrect, the user is sent back to the login page to try again. Upon successful login, the users are moved to the main page, containing the map showing current locations of airplanes detected, and buttons to change settings, manage their account, and log out of the application.

Settings and Account Dialog Boxes

From the main screen, if the user presses the settings button, a dialog box appears containing several different customization options. If the user presses the close button on the dialog box, then the dialog box disappears without saving any options. If the user presses the save changes button, the application stores the changes in the database, and the dialog box disappears. The same set of interactions apply to the account management button and dialog box.

Airplane Information Panel

When on the main screen, if the user interacts with an airplane icon displayed on the map, a panel showing info about the airplane appears. This panel contains the airplane's position, the signal strength from the remote node, and which remote node is detecting the airplane. Click or touching anywhere else on the map closes this information panel.

Flight Comparison Using OpenSky API

To enhance the accuracy and utility of our aircraft detection system, we integrated the OpenSky Network API into our application. This public API provides live air traffic data, allowing us to retrieve a list of flights within a specified radius around our sensor's coordinates. For our purposes, we defined a geographic bounding box of 0.05° latitude and longitude around the sensor—approximately 3.5 miles—to focus on flights most relevant to our detection zone. Every time a new signal is received by the sensor, the system compares it against the most recent flight data retrieved from OpenSky. This comparison enables the system to determine whether the detected aircraft corresponds to a known, registered flight. If a match is found, the system logs the aircraft's callsign, altitude, origin country, and timestamp as a known flight; otherwise, it flags the detection as potentially unknown. This integration not only helps validate detections in real-time but also highlights anomalies, making the platform valuable for monitoring unregistered or unexpected air traffic.

7.5 Anomaly Detection Algorithm Design

A traditional Z-Score / Statistical Thresholding algorithm works by constantly analyzing data, which could be in batches or streamed, and assigning a Z-score to each piece of data. To do this, the algorithm keeps track of a recent batch of signal values and calculates the mean average and standard deviation of those values. Then, for each new signal that comes in, it computes a Z-Score, which basically tells us how far that value is from the mean, measured in standard deviations.

$$Z = \frac{(x - \mu)}{\sigma}$$

If the Z-Score is too high, meaning the value is way off from the average, it gets flagged as an anomaly. This kind of behavior could indicate that an aircraft is reflecting a strong RF signal back toward our antenna. What makes this method perfect for our project is that it's extremely lightweight, doesn't need any model training, and works in real time. It also doesn't make any assumptions about the shape or behavior of the data, which is ideal when dealing with unpredictable RF environments. For a system that needs to be both fast and efficient, especially when running off limited power and computing resources, the Z-Score algorithm is perfect.

Though, a traditional Z-Score / Statistical Thresholding algorithm isn't exactly perfect for our use case. If an anomaly is detected, it gets counted towards the mean, raising the value of which anomalies will be detected. This is a problem because airplanes can be over our antenna for an extended period of time. Instead of just picking up that initial spike and staying triggered, the algorithm slowly starts to accept the anomalous values as the new normal. Over time, the

baseline shifts upward, and the algorithm stops detecting what should still be considered an active anomaly. In our case, that could mean the system fails to report a persistent signal caused by a plane directly over the antenna, which defeats the whole purpose.

To address this, we'll need to modify the algorithm to keep the baseline fixed for longer periods or implement a decay system that slowly forgets older values. To solve the problem of anomalies skewing the mean over time, we're going to switch from a rolling mean to a rolling median in our implementation. Unlike the mean, the median isn't affected by extreme values, which makes it much better for situations where an anomaly might persist, like an aircraft hovering or slowly flying over our antenna. When a plane is overhead for a long time, those high signal values would normally raise the average, making it harder to detect the anomaly as it continues. But with a rolling median, the baseline stays more stable because it simply reflects the middle value of the data window, not the average of everything, including spikes.

This way, even if several high signal values show up in a row, the algorithm won't immediately consider them normal. It allows the system to keep flagging those values as anomalous until the signal levels genuinely return to baseline. It also helps prevent false negatives, where the system might otherwise start ignoring what should still be a valid detection. Overall, switching to a rolling median makes our algorithm much more resistant to long-duration events and ensures consistent anomaly detection, even when the signal doesn't immediately go back to normal.

Another benefit of using the rolling median is that it improves the algorithm's resilience to noise. In RF environments, there's often random fluctuation or bursts of interference that can briefly spike signal levels. A rolling mean would treat those outliers as part of the baseline, which could lead to false positives or an unstable detection threshold. The median, on the other hand, filters those out naturally, which helps maintain more reliable performance over time.

We can also pair the rolling median with the Median Absolute Deviation (MAD) instead of the standard deviation to compute the threshold. MAD is a more robust measure of spread that works better with medians and is less sensitive to outliers. The modified Z-Score formula would then be based on how far a value deviates from the median, scaled by MAD. This gives us a full statistical thresholding method that's built for noisy, real-world signals.

Lastly, we're planning to tune the window size of the rolling median based on testing. A larger window gives a more stable baseline but reacts slower to changes, while a smaller window adapts faster but is more prone to fluctuation. We'll run experiments to figure out what size gives the best balance between responsiveness and stability for aircraft signals.

7.6 API Development

As our project becomes more advanced, one of the most important features we need to think about is building a full API. An API, or Application Programming Interface, is what allows other systems, tools, or programs to connect to our backend and get the data they need without having to go through the user interface. This matters a lot if Southcom eventually wants to plug into our system and pull data directly into their own dashboard or tools.

API Importance

Right now, everything in our project depends on the front-end interface. It shows aircraft detections on a live map and allows users to interact with the data. But in the future, not everyone will want to use our interface. Some users might want to take the data and feed it into a program they already use. An API would make that possible. Even more importantly, if the system is ever used in other countries or expanded to more users, an API would give them a way to access the signal information without needing to log into a website.

There's also the possibility that the interface could be removed altogether for some users. If Southcom has their own software that tracks alerts and aircraft detections, they might not need our map at all. In that case, they would only care about the raw data. With a solid API, the backend can provide exactly what they need in a clean and organized way, no matter what they do with it on their end.

Another reason to have an API is automation. With API access, people can program their own systems to pull data from us automatically. This means users won't have to sit and stare at the map all day. Their systems could check for updates every hour or get notified when something new is detected. The API would help turn our project into more than just a visual tool. Instead, it would become part of a larger, more flexible system.

API Purpose

To make this work, our API would need to include a few important features. For starters, it should be able to provide the latest aircraft detections, allow users to search by time or area, show signal strength, and indicate which sensor picked up the signal. It should also give access to basic system information like which users can see which sensors, and maybe even allow certain users to update sensor statuses, like marking them for maintenance.

All of the data would be returned in JSON format, which is easy for most programming languages to read and work with. A few examples of what our API might look like include endpoints such as: /api/detections/latest to get the newest data. There could also be options for submitting sensor updates or checking system health, depending on the user's role.

Security and Access Control

Security is a big deal when it comes to APIs. We don't want just anyone pulling information out of our system. That's why we would use token-based authentication. This would make it so every user gets a special key when they log in, and they must include that key when making requests. The system would also check what roles or permissions the user has and limit the data they can access. For example, a user who is only allowed to see data from Orlando wouldn't be able to request anything from Georgia. On top of that, we'd add protections like rate limits, which stop people from making too many requests in a short amount of time. All data would also be sent securely using HTTPS so it's protected while traveling across the internet.

Long-Term Benefits

The benefits of adding an API are huge. First, it makes the system much more flexible. We wouldn't be locked into one type of user interface. If someone wants to build a mobile app later, the API could power it. If another organization wants to plug into our data for their own tracking, they could do that too. It also makes our system more scalable and easier to maintain. Instead of

having a ton of different tools that all do their own thing, we'd have one central system that shares data the same way for everyone. In conclusion, building a well-organized and secure API is one of the best steps we can take to make sure our project keeps being useful in the future. The interface we've built is helpful now, but it won't meet every user's needs forever. An API opens the door to more advanced users, automation, and new tools we might not even think of yet. By building this into our system, we make sure it's ready for whatever comes next, and we give future users the freedom to use the data however they want.

7.7 Deployment Planning and Cloud Usage

If this system were used by a group like Southcom or another organization, it would need to be online and always available. The best way to do that would be using cloud hosting. AWS (Amazon Web Services) is a good option for this because it has lots of tools, is very reliable, and gives us choices depending on how we want to run the system. Two main options in AWS are EC2 instances and serverless tools like Lambda.

EC2 Instance

An EC2 instance is like having your own server in the cloud. You can install whatever software you want and keep it running all the time. That's useful for parts of our system that need to stay on, like the Node.js backend or the WebSocket server that sends data to users in real time. Since EC2 acts like a normal computer, it gives us the freedom to run any background services we need, such as the signal processing code that listens for input from the spectrum analyzer.

Lambda

On the other hand, AWS also has serverless options like Lambda, which run code only when needed. For example, if a user requests the latest signal data, a Lambda function could quickly grab it from the database and return the result. Serverless can save money because you're not paying for time when nothing is happening, but it's not great for real-time systems where something needs to be listening all the time.

Comparing EC2 and Serverless

To figure out which is better, we should consider which parts of our system need to run all the time and what parts don't. Here's how they compare:

EC2 Pros:

- Runs 24/7 without timing out
- Good for real-time data and background services
- Works well for live monitoring and fast updates

EC2 Cons:

- More expensive since you pay even when idle

- You have to manage and update the server yourself
- Higher chance of wasting resources if traffic is low

Serverless Pros:

- Only runs when triggered, so it's cheaper for light usage
- No need to manage servers or update software
- Automatically scales depending on the traffic

Serverless Cons:

- Not good for long-running services like live data feeds
- Has time limits and cold start delays
- Harder to use for things like WebSockets or background filtering

Table 7.1 Comparison of Clouds

Component	EC2	Hybrid (EC2 + Serverless)	Serverless
Backend	T2.medium EC2: $32/month	T2.micro EC2: $10/month	AWS Lambda only: $2–$5
WebSocket	Included in EC2	Included in EC2	Polling / AppSync $1-$5
Frontend Hosting	Served by EC2	S3 + CloudFront: $3-$5	S3 + CloudFront: $3-$5
Database	$5–$10	$5–$10	DynamoDB or Aurora Serverless: $5
API tasks	Handled by EC2	AWS Lambda: $1–$2	Handled via Lambda/API Gateway: $1–$2
Bandwidth	$5	$5	$5
Total Monthly Estimate	$45–$55	$25–$35	$18–$25

Conclusion

After comparing all the different options and thinking carefully about how our system actually works, the best choice for us would probably be a hybrid cloud setup. This means we would combine two different types of cloud hosting: EC2 instances and serverless functions like AWS Lambda. This will help to make sure we get both speed and reliability where we need it, and cost savings where we can afford to scale things down.

Some parts of our project need to stay active all the time, no matter what. For example, the backend service that connects to the spectrum analyzer and listens for aircraft signals needs to be constantly running. It's responsible for handling live signal data, cleaning it, and sending it to the rest of the system right away. These real-time features are important and can't be delayed, which is why we think they should be hosted on an EC2 instance. EC2 gives us the control we need to run background services 24/7 without worrying about them timing out or stopping unexpectedly.

But not every part of our system needs to run nonstop. There are also lighter, less urgent features, such as letting a user log into their account or requesting data from a past detection. These kinds of tasks are perfect for a serverless model. With AWS Lambda, we can run small pieces of code only when someone needs them. So instead of paying for a server that sits around waiting for a user to log in, we only pay when someone actually logs in. The same goes for simple API requests that don't need to be real-time. Lambda functions can handle them quickly and efficiently.

This combination gives us a really good balance. By keeping the most important and time-sensitive parts on EC2, we make sure the system stays fast and responsive. But by moving slower, low-traffic features to Lambda, we don't waste money or power running a full server for things that happen just once in a while. It also gives us more flexibility to scale the system later. If we need to support more users or sensors in the future, we can adjust how we use EC2 and serverless tools without having to rebuild the whole system.

Chapter 8: System Testing & Evaluation

8.1 Camera Array

8.1.1 Midterm Demo

As part of the PSE requirements for senior design 1, there is a midterm demo around 2 months or 8 weeks into the semester. The goal of this midterm demo is to demonstrate a subset of your optical design to a group of faculty members in order to receive feedback. In this case, Dr. Hagan, Dr. Likamwa, and Dr. Kar reviewed our midterm demos.

For our group's midterm demo, we displayed a simple 3 camera super resolution setup. The goal of this demo was to simply show that multiple images could be fused together to get more detail. To do this, we researched and purchased 3 cheap, low resolution USB cameras from Amazon, each at $15. Additionally, we purchased a simple USB hub to connect all cameras to a laptop. We then played around with code, occasionally seeking guidance from Chatgpt since we were not as familiar with many of the OpenCV capabilities. The first problem we ran into was the bandwidth problem. When the 3 cameras we purchased were connected via the USB hub, only

one of them would actually operate properly. To work around this, we had to plug each camera directly into one of our laptops, which limited our demo to only 2 cameras. Although this problem was not expected and definitely a little confusing, we figured out that we were still able to demonstrate what we wanted with 2 cameras.

After testing many different configurations of code, we ended up getting an iteration we thought was good. The code, when run with 2 cameras, would first detect the operating cameras and then assign them an integer index. This helped to define which image came from which camera for the fusion, although with 2 cameras, it does not make much of a difference. After assigning each camera an indice, a small window would appear to show you the feed from each camera, this feature was why our 3 camera setup and USB hub connection was not working. Once each window popped up, we made sure that the cameras were capturing as much of the same scene as possible from different perspectives, as is necessary for super resolution. Once that was confirmed, we pressed the ‘s’ button on our laptop to capture an image and the code was allowed to run. The code we used, which was created with help from Chatgpt, attempted to locate similar features from each image to try to produce the best super resolution image possible. Because of this feature, our success rate was actually only around 70%. This was due to some misalignment issues and image overlap inconsistencies. Sometimes, if either of the previously mentioned conditions were a factor, a super resolution image would be produced that would look heavily distorted, except for one common feature from the scene. This led us to think of our demo case and what conditions would be best for our demonstration.

After testing different scenes around the CREOL building, we realized the best way to demo our 2 camera setup was to use an object about 1-5 meters away. We also noticed that the background behind the object would sometimes heavily confuse the code and lead to images that looked heavily distorted. For this reason, we decided that our best demo case was a QR code against a solid background. The use of a QR code would help us show that super resolution was in fact taking place, and by comparing the super resolution image to the image of a single camera, we could effectively show that some corners of the QR code were more well defined in the super resolution image. The white background was used as a means to increase our success rate and limit the production of distorted images. Overall, the demo taught us about bandwidth issues with USB cameras, the density of code that goes into super resolution, and the best ways to be able to demonstrate it. Dr. Hagan was also able to get us thinking about what types of cameras or scenes each imaging device should be capturing, since multiple low resolution cameras capturing the same thing doesn't provide as much detail as you’d hope. He mentioned the iPhone and led us to start thinking about how its camera systems operate and how our project could benefit from learning about that.

For our final demonstration at the end of the semester, we aimed to validate key aspects of our optical design, even though the exact lenses specified for our final system—three plano-convex and three plano-concave lenses with focal lengths supporting 5× magnification—had not yet arrived. To move forward with testing, we assembled a substitute optical system using a plano-convex lens with a focal length of 100 cm and a plano-concave lens with a focal length of −10 cm. This combination resulted in a 10× magnification setup, effectively doubling the magnification planned in our final implementation. While not an exact match, this system allowed us to test the core principles behind our imaging architecture—specifically, whether

magnified observation through this type of lens arrangement would behave as predicted and support our resolution goals.

8.1.2 Final Demo SD1

Unlike our earlier midterm demo, which used low-resolution USB cameras and a QR code as the target, this final test focused solely on demonstrating the optical magnification effect using a laser beam. The laser beam was used as a precise and narrow light source, allowing us to visibly track how the beam’s image changed as it passed through the lens system. This served as an effective proxy for image detail, as the beam's diameter and position could be closely monitored at both the input and output of the lens assembly. The beam provided a clean, high-contrast feature that made it easy to detect changes in image scale and allowed us to validate the spatial behavior of the lens system under high magnification.

We aligned the plano-convex and plano-concave lenses on an optical rail, with the laser projected through the system and onto a target surface. By measuring the apparent diameter and beam divergence at various distances, we confirmed that the system produced a consistent 10× magnification, as expected from the focal length ratio. This helped us validate the core design assumptions and build confidence in the geometric behavior of the final lens system. While this test didn’t involve digital imaging or feature-based super-resolution processing, it clearly demonstrated that the optical magnification portion of our design was functioning correctly, which is a critical first step.

Additionally, this test allowed us to explore challenges in lens alignment, vibration sensitivity, and spatial calibration. At such high magnifications, even slight misalignments or tilts in the lens setup produced noticeable deviations in the projected beam. These observations highlighted the need for precision mounting and vibration isolation, especially in the final system where smaller magnification tolerances will still require consistent alignment for effective stitching and fusion. This experience will directly inform how we design our multi-camera frame and lens holders next semester. The image below shows the optical setup we used for this demo. The laser is going through the system beginning on the right, then the final image can be seen on the wall. This demonstration did prove the optical property of magnification we were attempting to show, this can be seen by comparing the spot size of the laser hitting the first lens and the spot size on the wall, which is approximately 10x larger.

Figure 8.1 *Final Demo Optical Setup*

Although limited in scope, this final demonstration effectively proved that our dual-lens approach can achieve and maintain high magnification, and that we understand the optical principles required to scale that design down to our target 5× magnification system. With our actual lenses set to arrive in the following semester, we are now well-positioned to integrate the imaging pipeline with a verified optical system, informed by real-world testing and physical validation of our magnification architecture.

8.1.3 Additional Optical Testing

In addition to the demonstrations that were done as part of the requirements for PSE students in senior design, testing was conducted with different variations of code to find which produced a result we were satisfied with, at least for preliminary purposes. The images below were done with a code that was produced using ChatGPT with finetuning and debugging for what we wanted specifically. While this code works well, it will likely not be the final set of code we use for the final demo because of its partial success rate. We tested the code, which uses feature based matching, with different scenes within the CREOL building at UCF.

The code that is used produces a ‘stitched’ image and a ‘super resolution’ image. The stitched image is the code's attempt to align the images together using its feature based matching algorithm. The super resolution image is what is produced when the stitched image is passed

through super resolution code. Though it is labeled as super resolution, the code is a very basic version of super resolution so the results are not as good as they could be, but they will inform us about how certain scenes or objects may act when it goes through the code. Additionally, the cameras are placed on a rig that was found in the CREOL SD lab and was used as a means to crudely maintain alignment.

Figure 8.2 *Test of camera array with KIMTECH box*

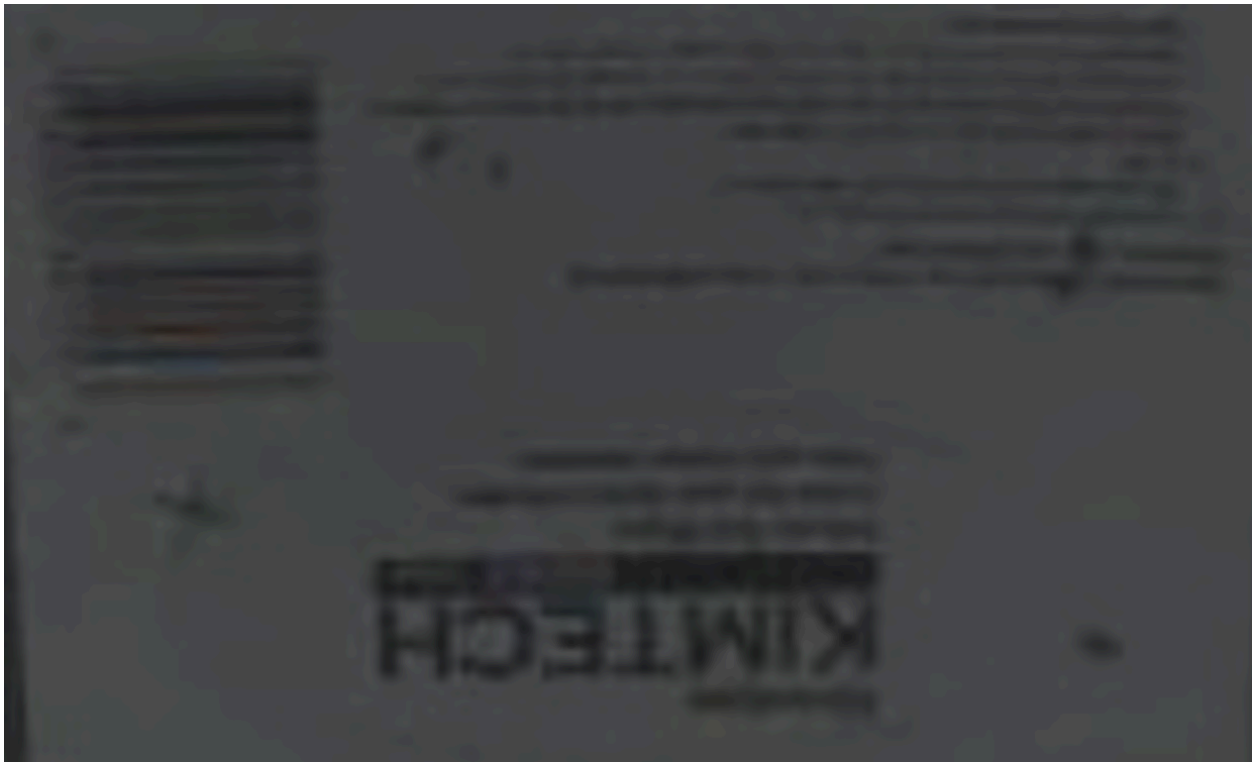


Figure 8.3 *KIMTECH box 'stitched' image*

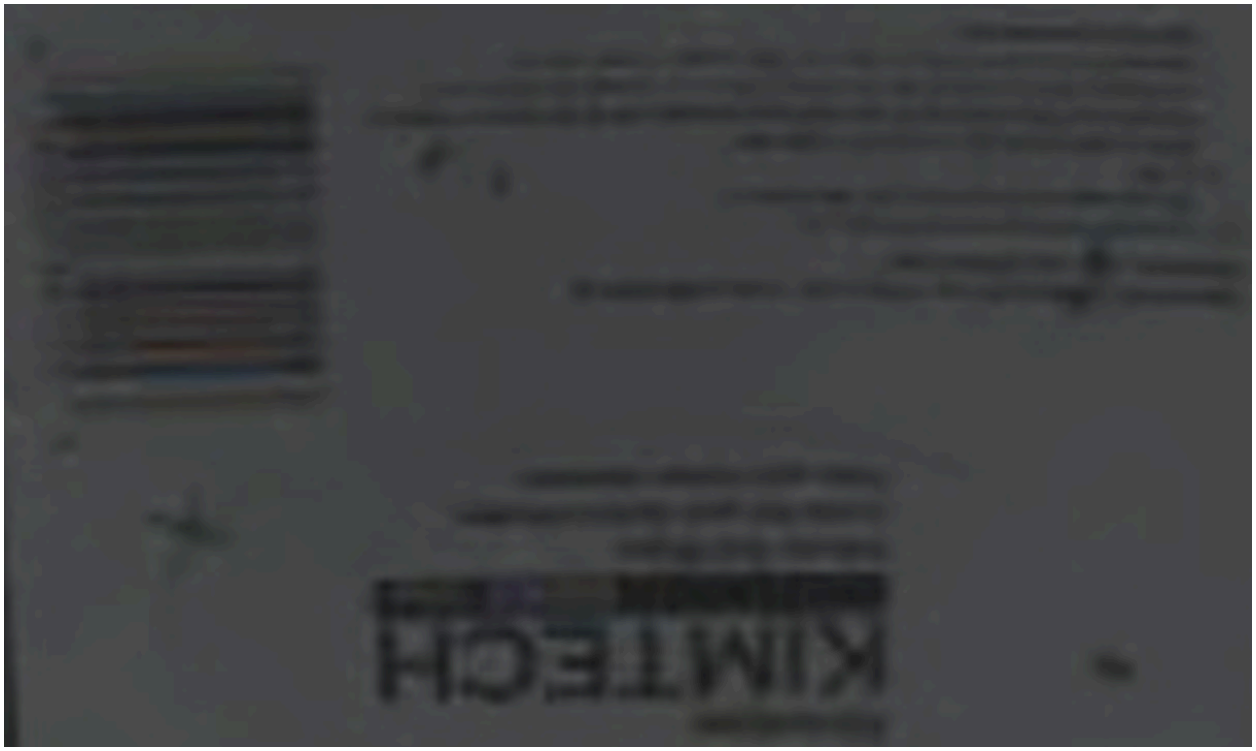


Figure 8.4 *KIMTECH box 'super resolution' image*

With the KIMTECH box, the goal was to see if the camera array could help enhance some of the letterings or the edges of the bar code. What is shown above in gray is a section of the entire stitched and super resolution images. It can be seen that overall, the resolution from the super resolution image still leaves a lot to be desired. However, though it may not be as apparent, upon further inspection of that image, there are more well defined edges than were present in the stitched image. This is a positive result since our goal is to ultimately better define the edges of aircraft or drones that may not appear as clear to some cameras.

Figure 8.5 *Test of camera array with sprite bottle*

Figure 8.6 *Sprite bottle 'stitched' image*

Figure 8.7 *Sprite bottle 'super resolution' image*

The next test was with a sprite bottle. The sprite bottle provided an interesting result because of the fact that it is rounded at its edges and contains liquid. The image above shows an interesting result when the sprite bottle is run through the code. It appears that during the image stitching part of the code, the bottle was split, likely due to the presence of the clear liquid within the bottle. This unexpected result provided an important note for us to consider. If the object we are attempting to image has some sort of camouflaging technology or paint, it may mess up our code's ability to properly image it. Obviously the sprite bottle does not have camouflage but the liquid within disturbs our code's image stitching ability to function properly. After processing, the images edges were still made more clear so in terms of that, it was a success but this output would not be and should not be acceptable for the potential applications.

Figure 8.8 *Test of camera array with mug*

Figure 8.9 *Mug 'Stitched' Image*

Figure 8.10 *Mug 'Super Resolution' Image*

The last test we ran for the camera array was with a mug we found in the CREOL SD lab. This was an interesting object to image because of its shape and the presence of small text on its body. After being run through the code, it can be seen that the mug was slightly distorted around the handle. However, the super resolution image once again was able to better define the edges of the object which is the ultimate goal. Though the image is slightly distorted, the integrity of the object is still maintained unlike with the sprite bottle. While the text on the mug cannot be read, small blurs can still be seen in the text's place. Ideally, we wouldn't need to read text but instead see important shapes like identification such as countries flags that may be present on aircraft and drones.

Overall, the preliminary tests were a success. Though the setup is quite crude, it allowed us to prove that our system works on a very basic level and has introduced new problems that we didn't consider before like techniques that may be intended to disrupt our system like camouflage. The main goal of defining edges better was achieved but still leaves lots of room for improvement.

The next series of testing was conducted using Zemax software. Zemax is an optical design program that can conduct ray tracing tests given a lens system. We used it to validate the feasibility of the three lens configurations within the camera array. The primary objective of

these tests were to confirm that a 5x magnification could be achieved consistently across all systems, and to evaluate the physical spacing requirements between two lenses in the same system. These tests were crucial for ensuring the systems would work before the construction of the lens housing began.

	Surface Type		Comment	Radius		Thickness		Material
0	OBJECT	Standard		Infinity		Infinity		
1	STOP	Standard		Infinity		20.000		
2		Standard	125 Lens 1 S1	64.400		3.300		BK7
3		Standard	Lens 1 S2	Infinity		122.0...	M	
4	IMAGE	Standard		Infinity		-		

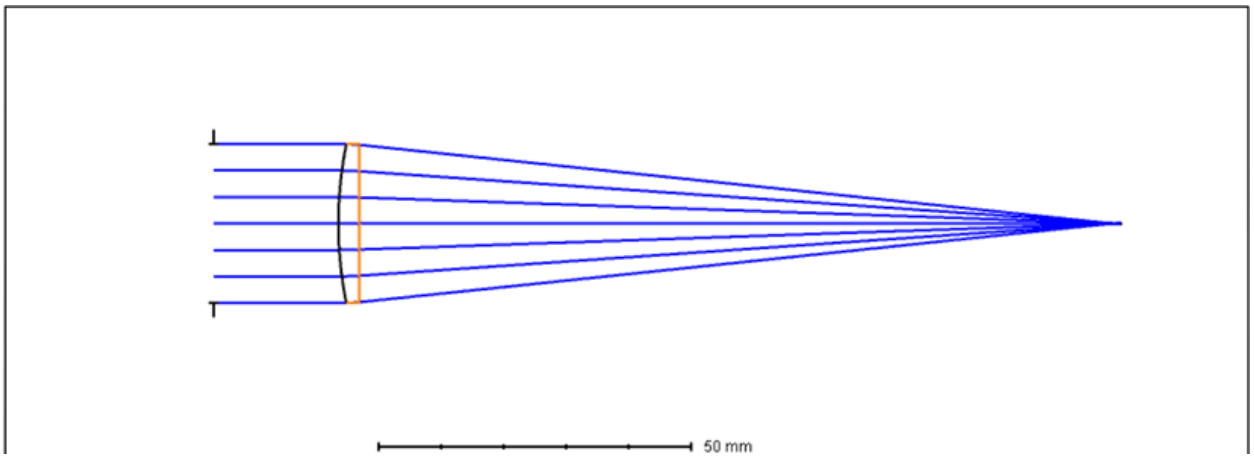


Figure 8.11 *Least Narrow Field of View Lens System Zemax Tests*

	Surface Type		Comment	Radius		Thickness		Material
0	OBJECT	Standard		Infinity		Infinity		
1	STOP	Standard		Infinity		20.000		
2		Standard	175 Lens 1 S1	90.100		2.900		BK7
3		Standard	Lens 1 S2	Infinity		171.8...	M	
4	IMAGE	Standard		Infinity		-		

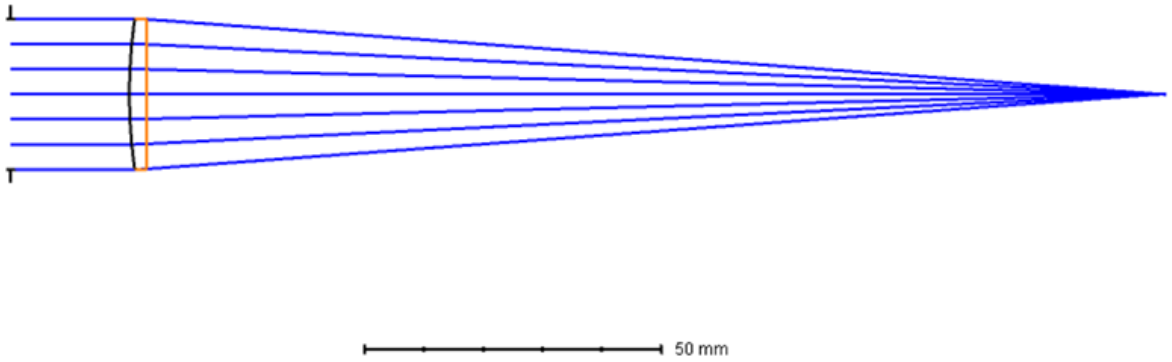


Figure 8.12 *Middle Field of View Lens System Zemax Tests*

	Surface Type		Comment	Radius		Thickness		Material
0	OBJECT	Standard		Infinity		Infinity		
1	STOP	Standard		Infinity		20.000		
2		Standard	250 Lens 1 S1	128.800		2.600		BK7
3		Standard	Lens 1 S2	Infinity		246.6...	M	
4	IMAGE	Standard		Infinity		-		

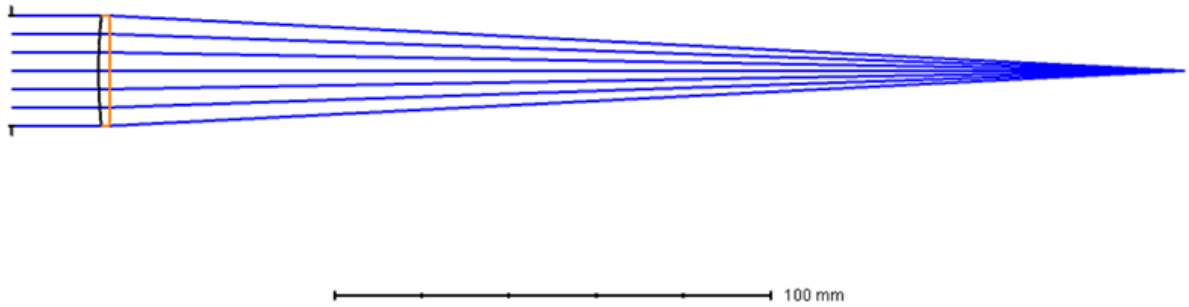


Figure 8.13 *Most Narrow Field of View Lens System Zemax Tests*

Originally, a two-lens system combining a plano-convex and a plano-concave lens was used to achieve the desired 5× magnification. However, further analysis revealed that to enable effective super-resolution, we needed to capture images at three distinct angular resolutions. To address this, we moved away from the dual-lens design and adopted three separate single-lens systems, each consisting of a plano-convex lens with a unique focal length. We have since changed the Zemax calculations to demonstrate these three systems. With the one lens system, we still achieved similar image quality and desired object size in our images. Figure 8.11 shows our widest field of view system, 8.12 shows the middle, and 8.13, the narrowest. These results provided valuable insight into the trade-offs between optical quality and mechanical feasibility, guiding the selection and refinement of lenses for the final array configuration.

For SD2, our main priority was testing. Once we discovered that we needed different angular resolutions from each source, we began to play around with distances, object sizes, and other things. The main goal was to get an image that actually was enhanced, at least visually. We achieved this by testing several different items including a large recycling bin, license plates, signs around campus and much more. After many days of testing, we understood that one big limitation of our system dealt with lighting and camera setting adjustments. Since the Arducams are manually adjusted, this meant that each lighting scenario had its own special combination of settings that fit that specific time and place. This was most evident during our D-Day demo, where we spent around 20 min trying to adjust settings to get the clearest images we could. As a result, our outputs were not ideal but our proof of concept got across.

Ultimately, the system worked as it needed to but requires better definitions for success, especially numerically. Dr. Kuebler was keen to point out that when delivering on engineering and customer requirements, plain eye tests absolutely do not cut it. For our systems, the metrics actually supported our input images better than our output which is likely due to one of the inputs or algorithms introducing a lot of noise to the system. Regardless, in future iterations of the system, should it be invested in, can involve larger arrays, with much better cameras, and better processing units to increase efficiency and usability. As cameras continue to get smaller and smaller, this concept of super-resolution will quickly evolve and be introduced into our society as the new norm. The concept of taking a large amount of lower-quality images to produce a high-quality counterpart, or simply telling a computer to ‘enhance the image’ is no longer science fiction or the magic of Hollywood. And the technology is only getting better!

8.2 RF Device

8.2.1 RF First Set of Tests

One of the most important characteristics of our system is the range. This will vary widely depending on how much power we may receive. Not only that, but it depends on whether we are Line of Sight or if we are slightly covered. Other factors include weather, the noise level of the area, and polarization.

For our preliminary range tests, we will do a classic range measurement by using a car’s keyfob. The reason for this is that it operates at 315 Mhz. This is a frequency that we can transmit all day and not get in trouble for. Also, because it isn’t a common frequency we will have an almost noiseless environment to test in. Of course, there will be other people pressing their keyfob, but this is something that is easy for us to discern as it is a pulse of energy. Meaning we can time it with each other. The keyfob also puts out a very small amount of energy, only 10 nW. This provides a realistic scenario for us as the power reflected may be the same if not more power.

Next, we will test the polarization. The polarization affects how much of the signal we will be able to detect. Not only that, but it will determine how we are able to orient the antenna when we get to install it in the scenario for our sponsor.

For our test, we will have Line of Sight to the keyfob to keep things simple. We are doing a classic range measurement, so we would like to keep it as controlled as possible by eliminating variables such as leaves. From there, we will test various distances and see what maximum range

we are able to detect. For our experiment, we will be using the included software with our spectrum analyzer, Spike. It has a real time analysis mode, which will provide us with the highest level of accuracy and precision that our spectrum analyzer.

First, is a photo of our experimental setup. As shown, we have a genius system for holding the antenna horizontally or vertically. This will allow us to test each polarization type. Not only that, but having the antenna exposed will eliminate as many variables as possible.

Figure 8.14 *RF Device initial test setup*

For the experimental part we wanted to test the horizontal Polarization first. We started by making a control distance, 1 meter. From there, we moved to known distances away from the spectrum analyzer in order to find its maximum range. Once we reach the top of the range, we will slowly increment the distance until we cannot find the signal anymore. It is to be noted we didn’t jump up to 150 meters because it was in the middle of the road.

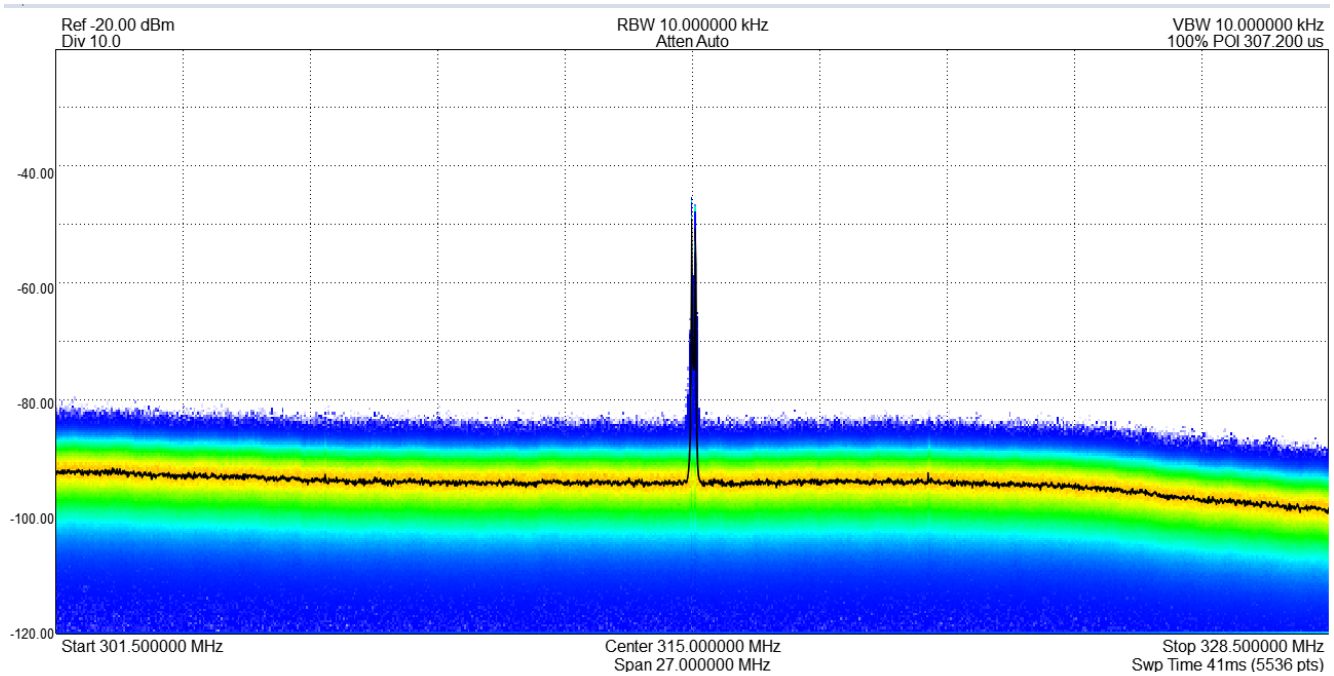


Figure 8.15 *1 meter test results*

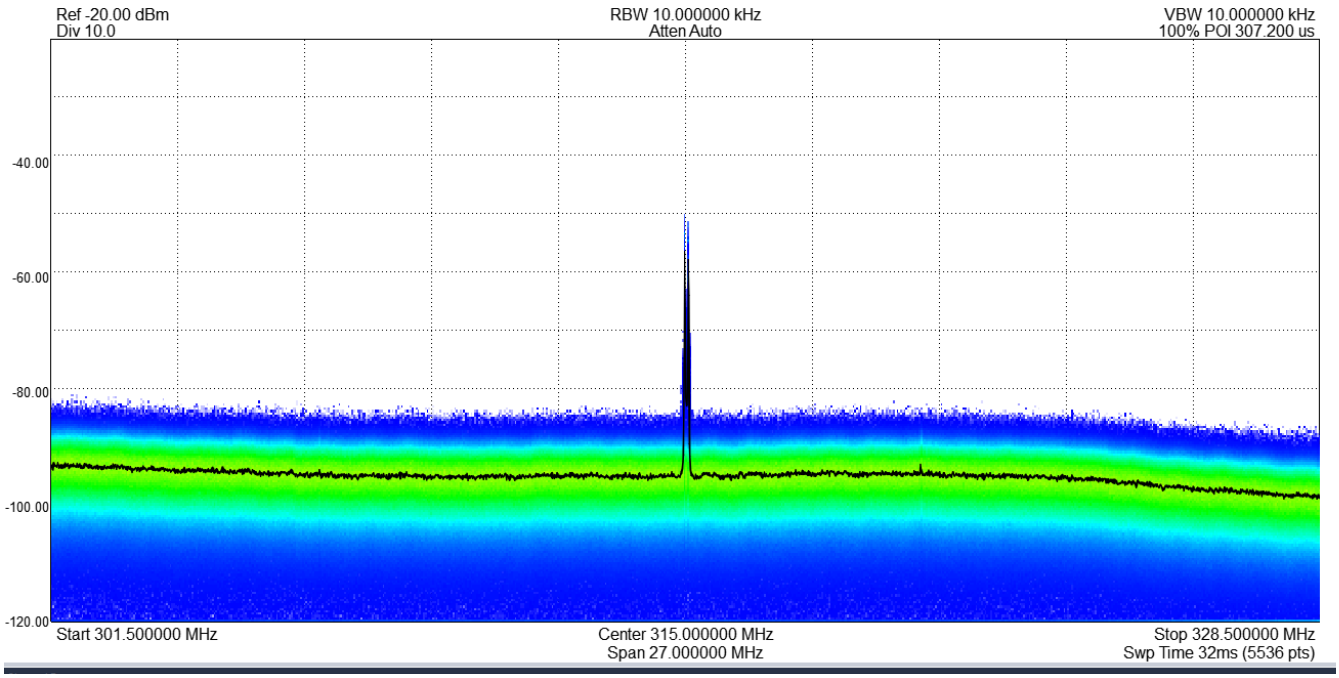


Figure 8.16 *5 meter test results*

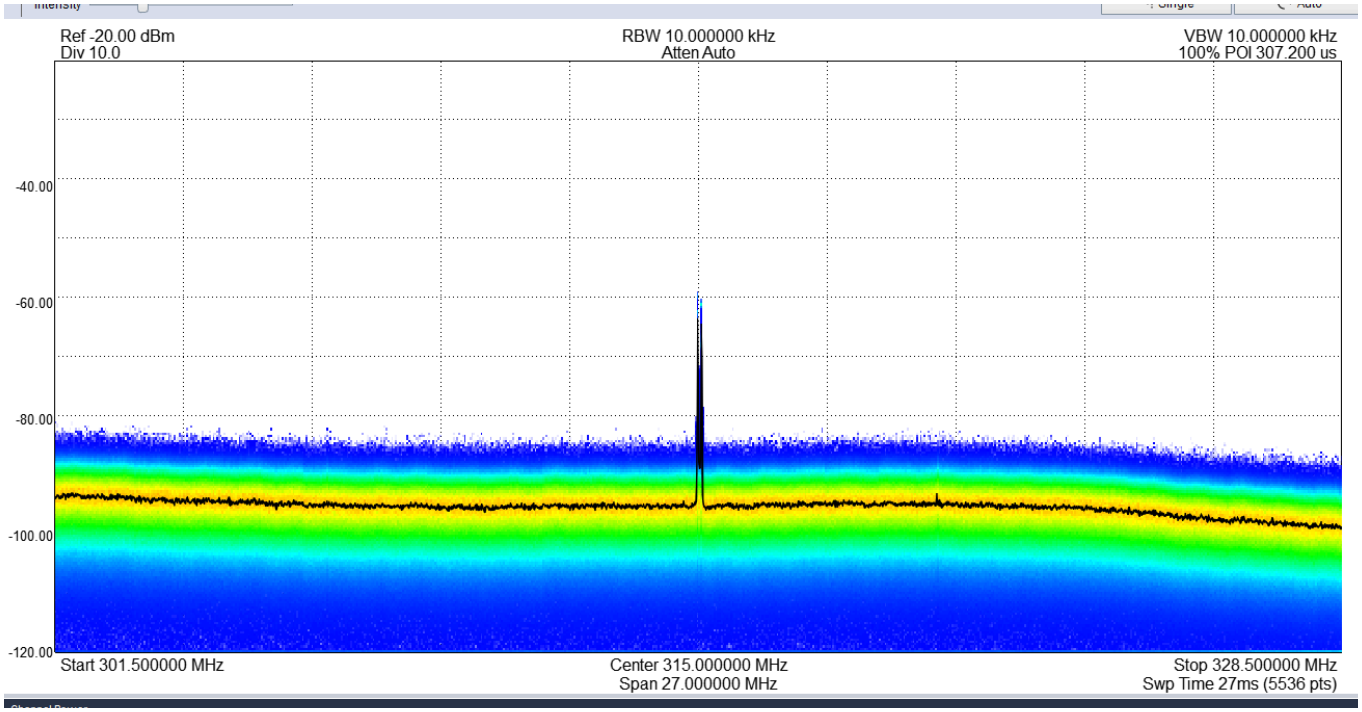


Figure 8.17 *10 meter test results*

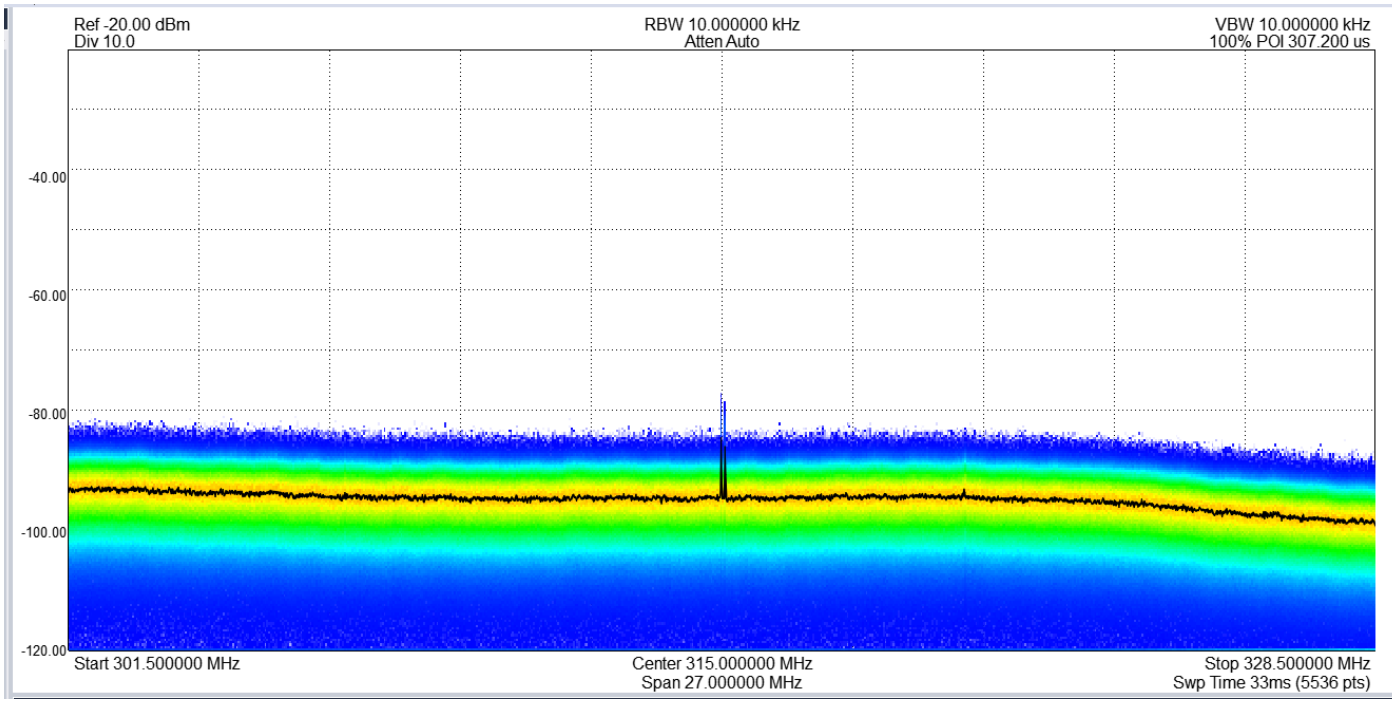


Figure 8.18 *50 meter test results*

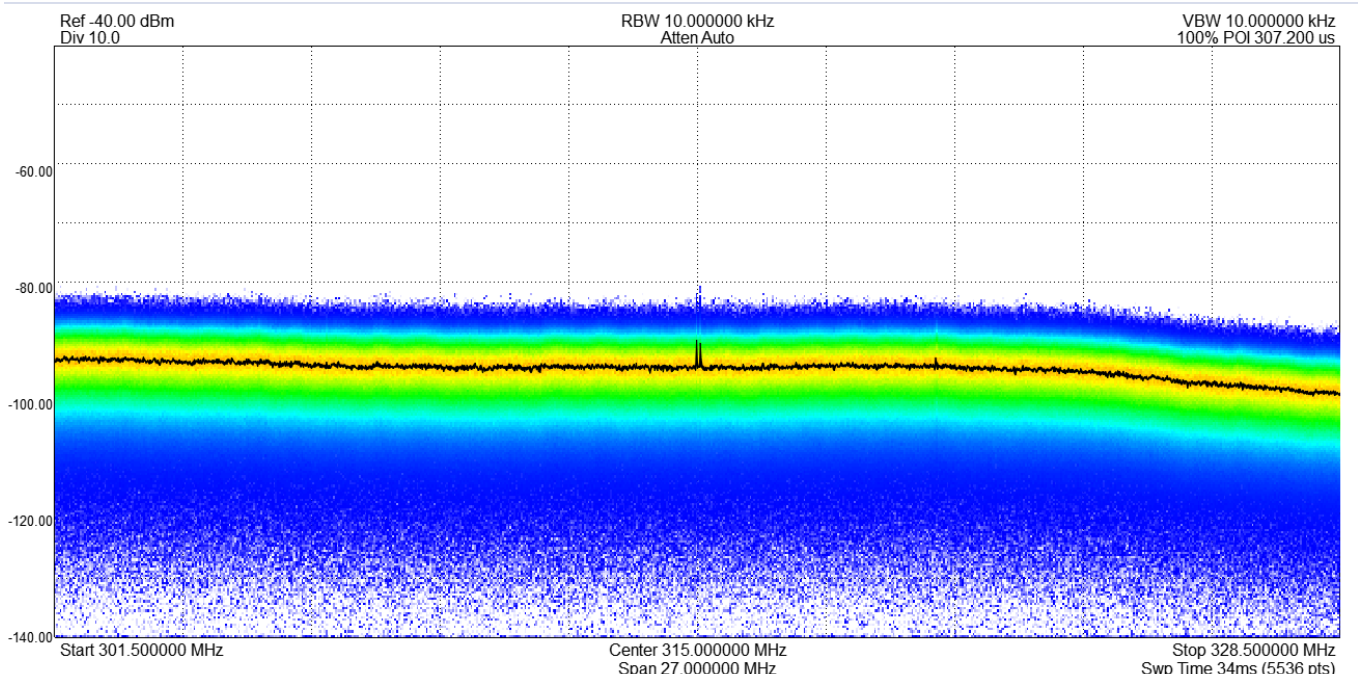


Figure 8.19 *100 meter test results*

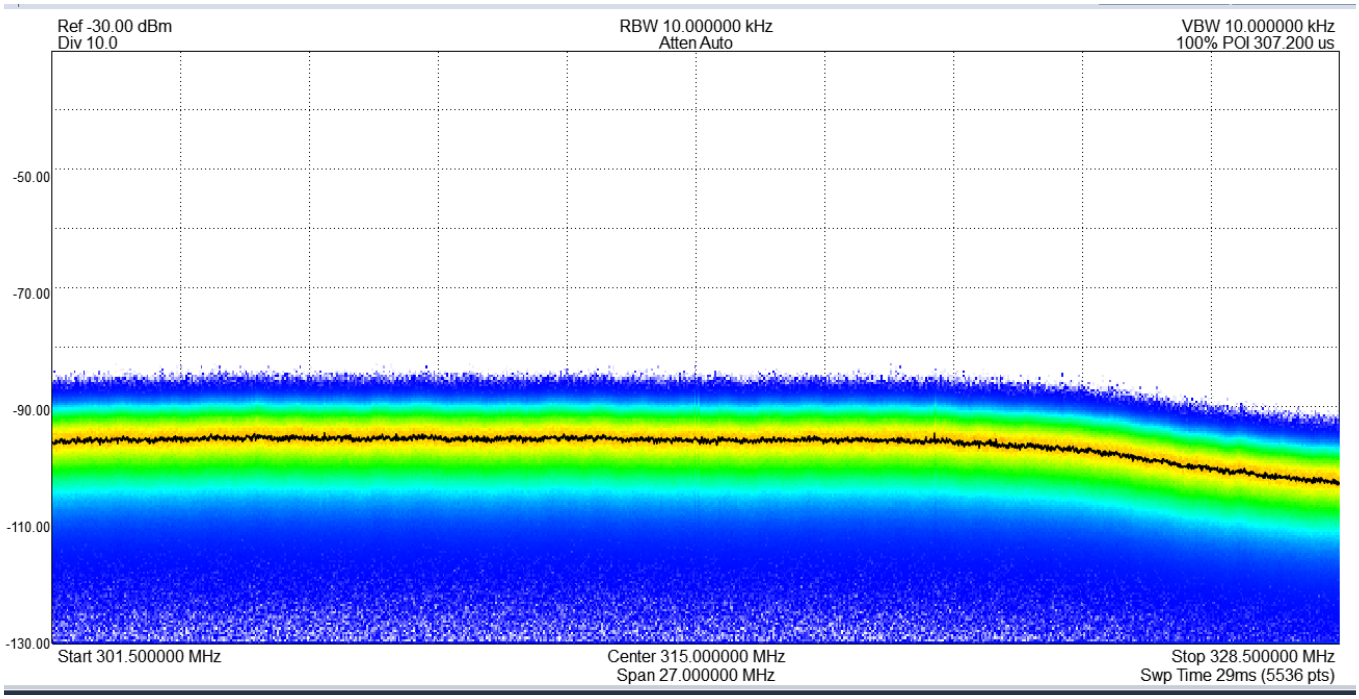


Figure 8.20 *170 meter test results*

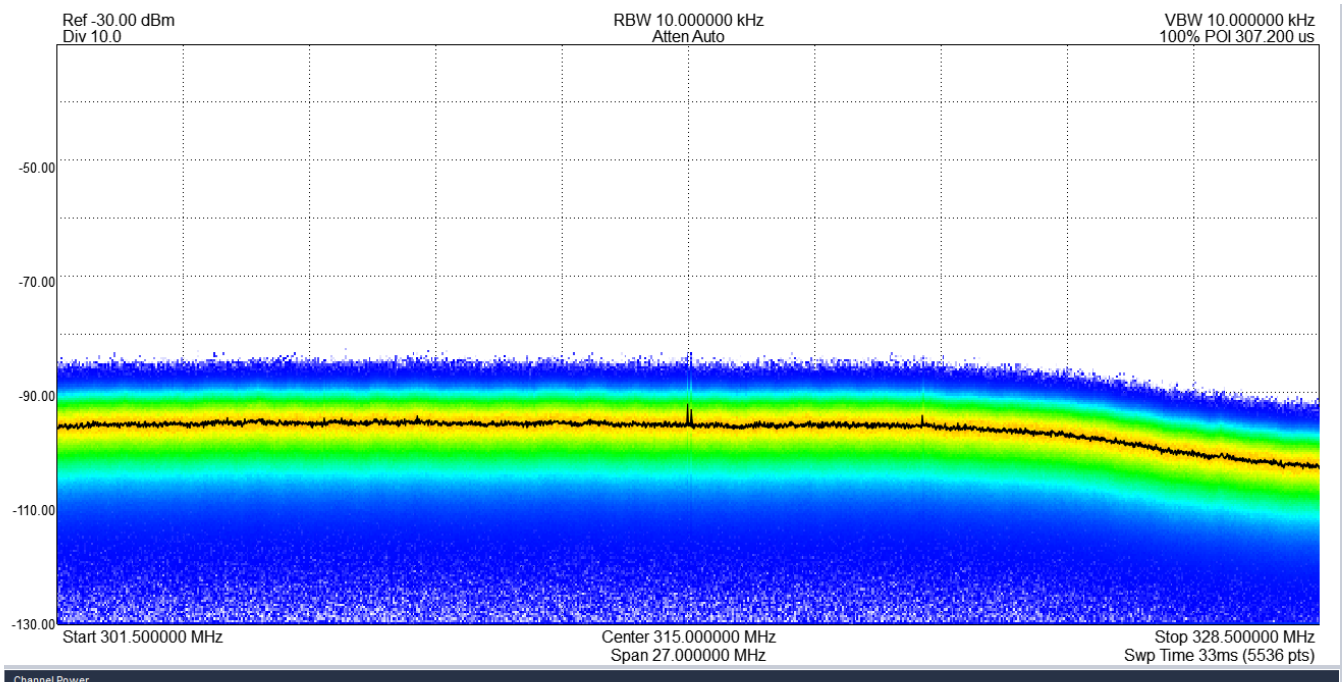


Figure 8.21 *175 meter test results*

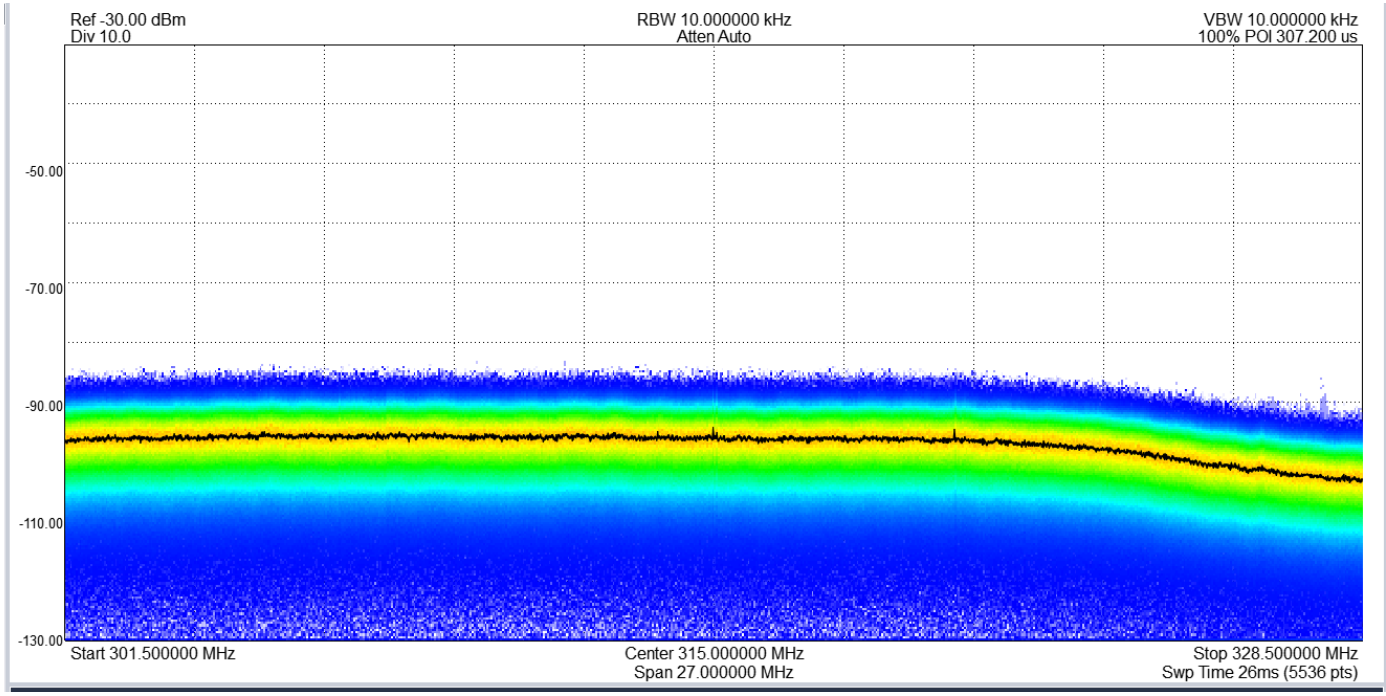


Figure 8.22 1*80 meter test results*

Results

As shown in the image above, we can see that the spectrum analyzer was not able to detect at 180 meters. This provides us a good baseline as to what the system may be capable of when it is horizontally polarized.

Next, we wanted to test the vertical polarization to see what would be most effective. Using the same test procedure as last time.

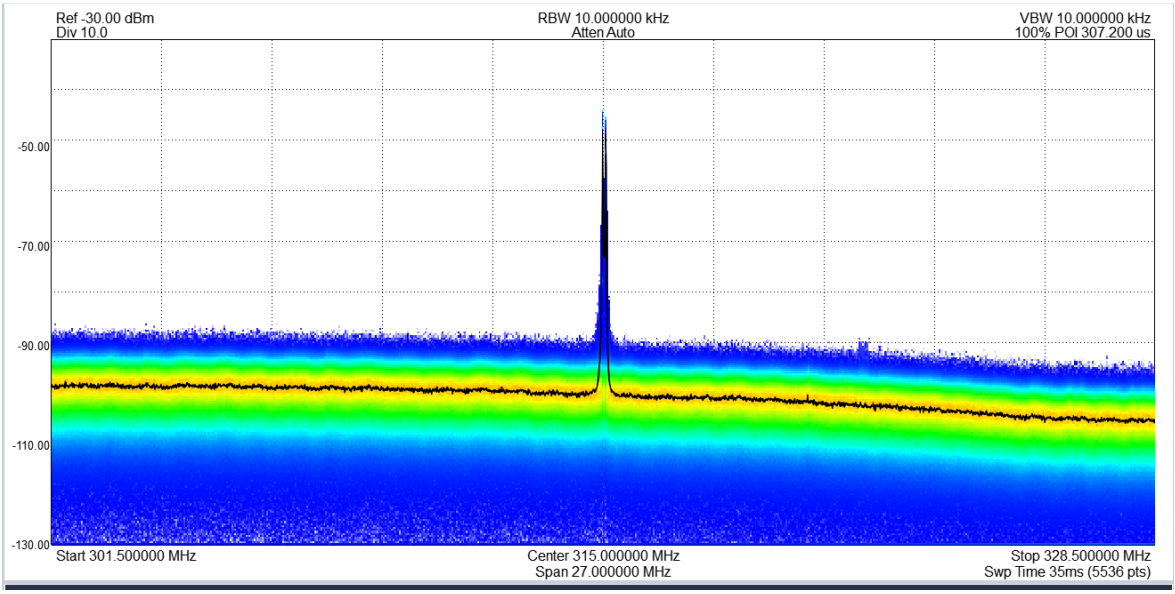


Figure 8.23 *1 meter test results (VP)*

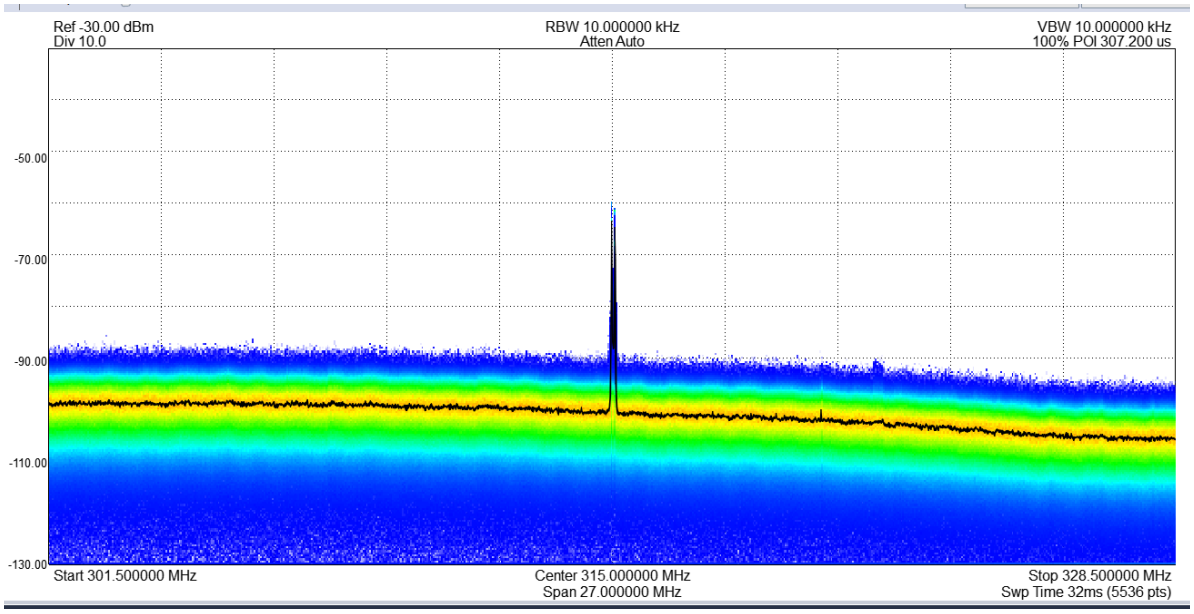


Figure 8.24 *5 meter test results (VP)*

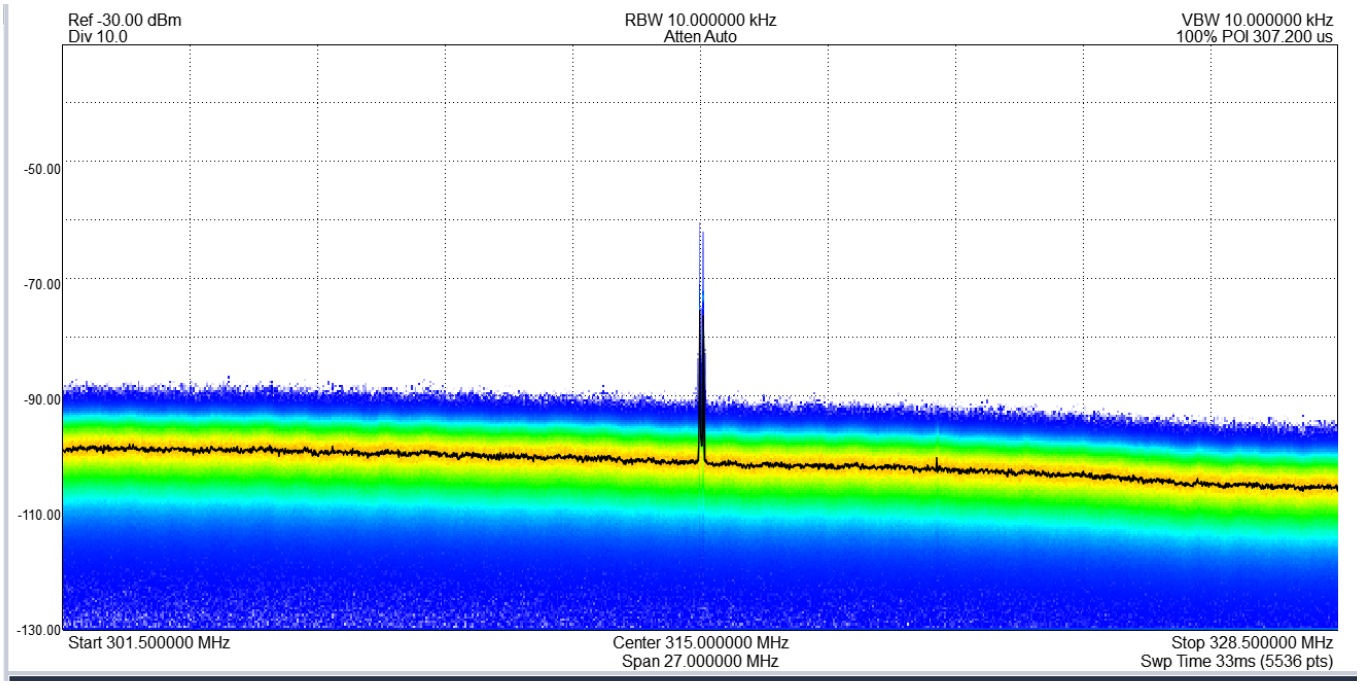


Figure 8.25 *10 meter test results (VP)*

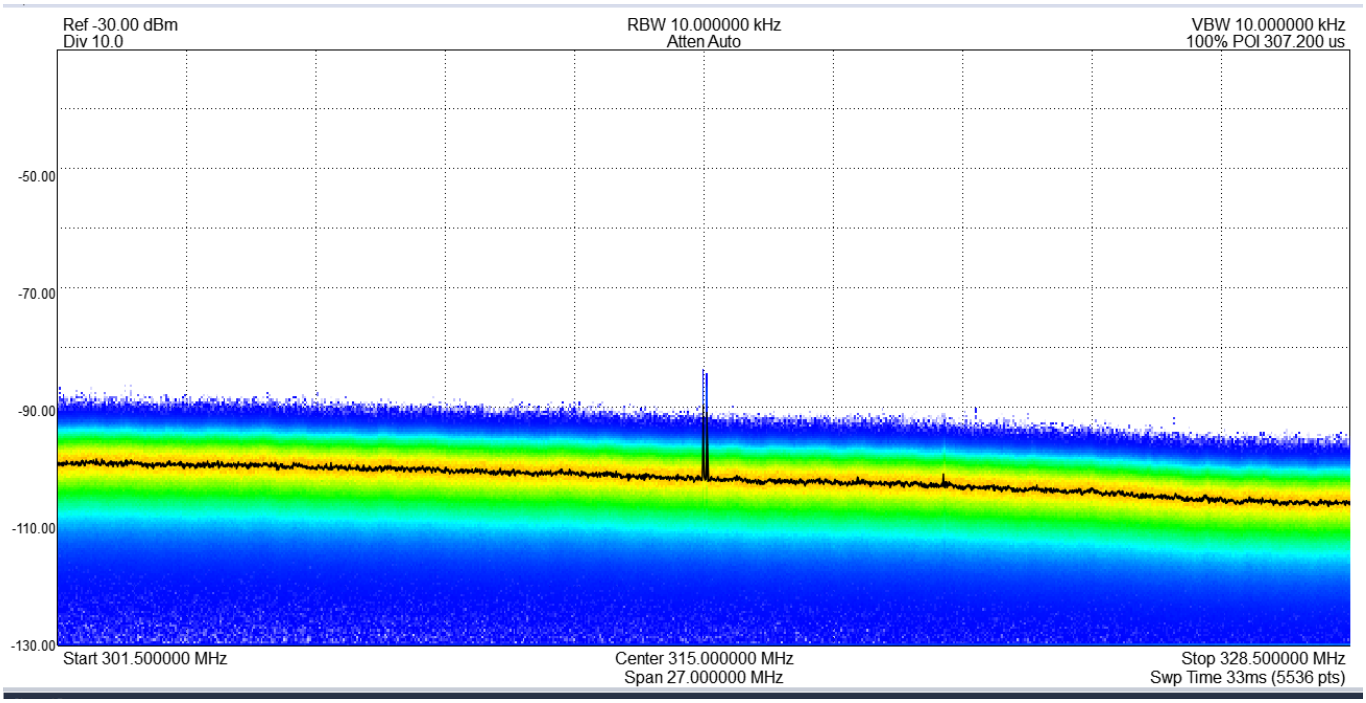


Figure 8.26 *50 meter test results (VP)*

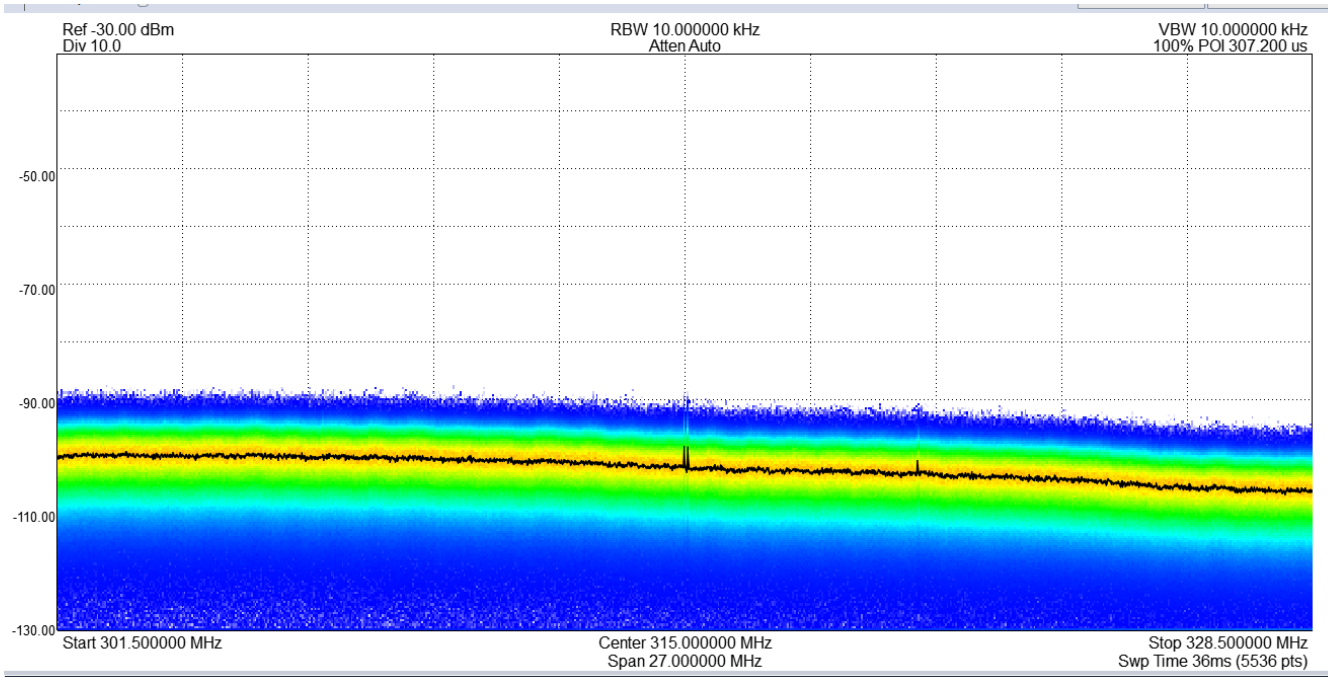


Figure 8.27 *100 meter test results (VP)*

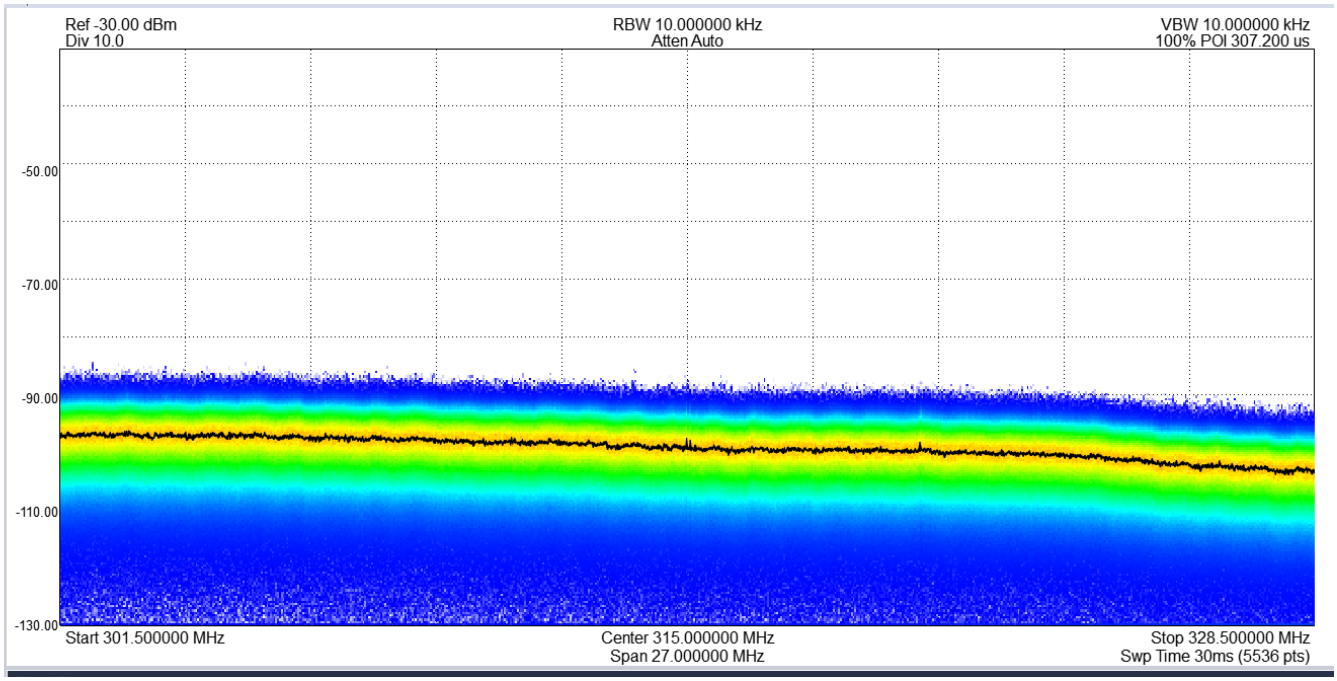


Figure 8.28 *115 meter test results (VP)*

Conclusion

As shown above, we can see that vertical polarization is much worse than horizontal polarization. It stops detecting at 115 meters in comparison to the latter at 180 meters. This is a significant improvement, and is very helpful for when we go to mount the system. That is a lot of performance that may have been left on the table if not tested.

8.2.2 RF Device Testing Continued

I was also curious about the fidelity and aptitude of our spectrum analyzer and antenna system. Naturally, we tested to see how able it would be to detect aircraft communications. By finding these aircraft communications channels, it would prove the ability of the spectrum analyzer and antenna to find these signals as they pass through many miles of different atmosphere.

For the test procedure, the spectrum analyzer and antenna will be placed inside of the house near a window. This is done so that we can really test the ability of the system. Next, we will be looking at 1090 Mhz which is the ADS-B transponder signal frequency. By using this transponder frequency, we know that it will be sent and received from the air and ground. Not only that, but it will prove that we can detect these signals from unknown distances.

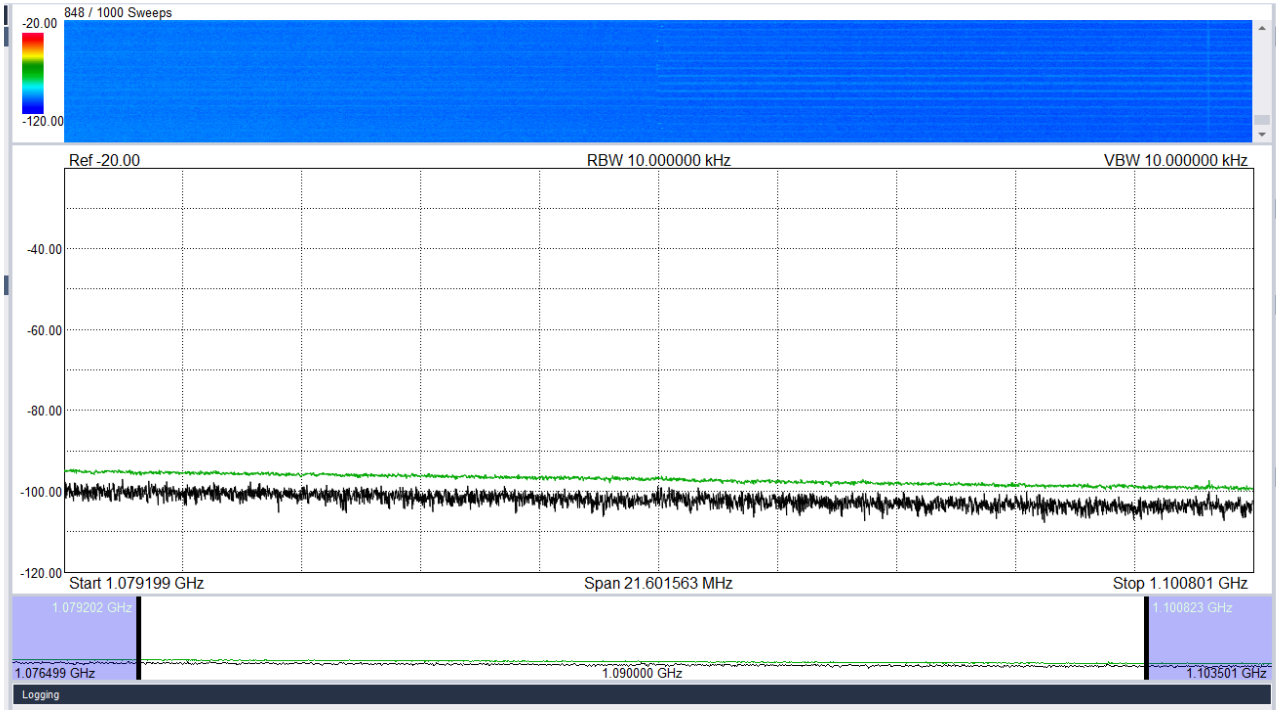


Figure 8.29 *RF environment, no aircraft*

Results

The image shown above shows what our RF environment will look like without any aircraft communications or transponders working. The green line also acts as a baseline that we have

established in the software. It eliminates all of the noise that is in the area by recording it then will be able to detect an anomaly by crossing that threshold of detection.

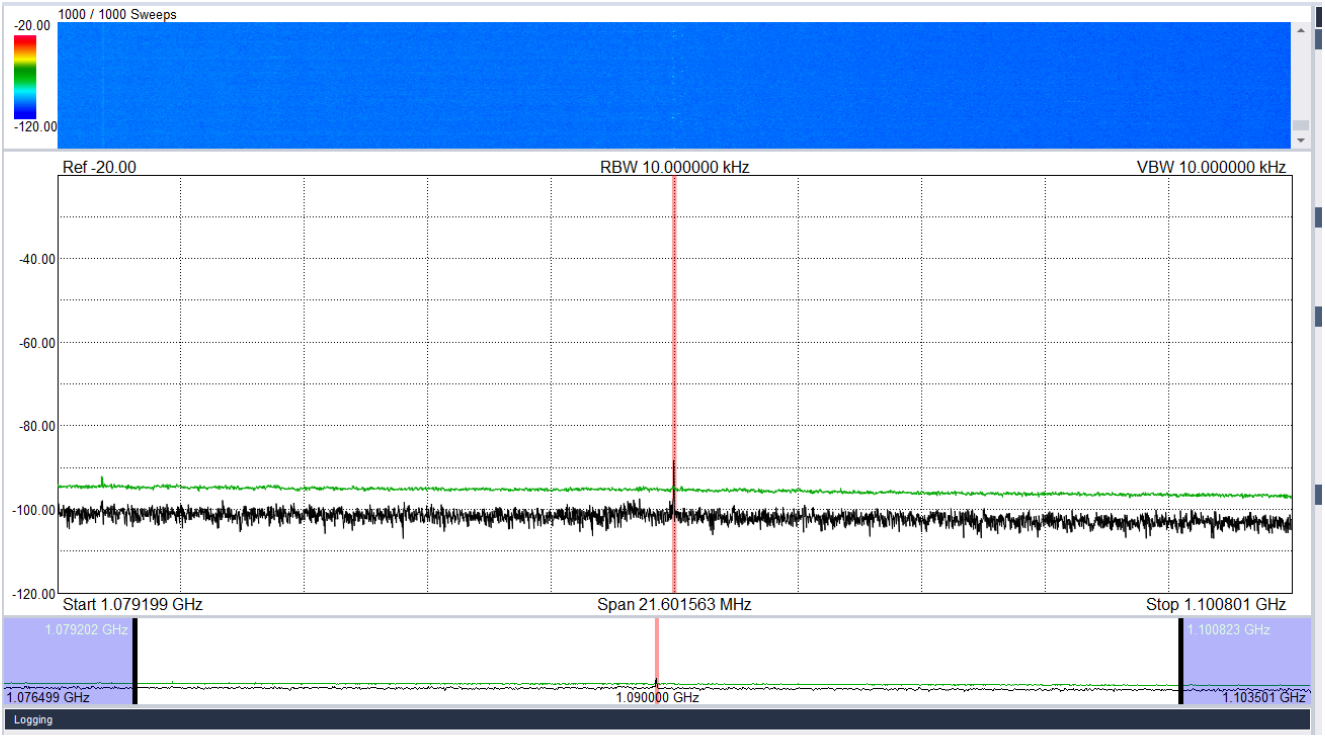


Figure 8.30 *Demonstration of detection*

Shown in the above image is a red line or ping that shows us that we have detected something. It is over that threshold that we have recorded and will light up.

Conclusion

This is incredibly important for our project as this is an unknown distance away. Making it possible for us to be able to detect anything as long as it is RF even if it is extremely weak. It's also really important as this was inside of the house and pointed outside of a window. This is the least optimal of conditions for our case. If it were in a better situation we would only get better resolution. Not only that, but it was recorded at midnight when there is less RF activity which may be more indicative of the environment this will be working in. This provides us a great baseline to the capabilities of our system and how we will implement it as we go further. The resolution provided is extremely useful if we want to passively detect an aircraft.

8.2.3 RF Device Testing Software

One of the major challenges that we will face when doing this project is sending the data. It is important for us and our upcoming demo in Jacksonville FL, to prove that Star Link is capable of such a task. As a result, we needed a way to send our data securely and in real time.

First, we needed to program our "sensor" to process the data and pick out what was useful. We did this by gathering a baseline of the surrounding areas RF energy. Then, we looked for things that stood out. For this stage, we used a threshold of 5dB over the baseline. Once a signal that

peaked over this threshold, such as a car keyfob or a walkie talkie, it would send the data. This of course meant we had to use the Spike SDK to program it. We used SCPI or Standard Commands for Programmable Instruments. The interface that we used is shown below.

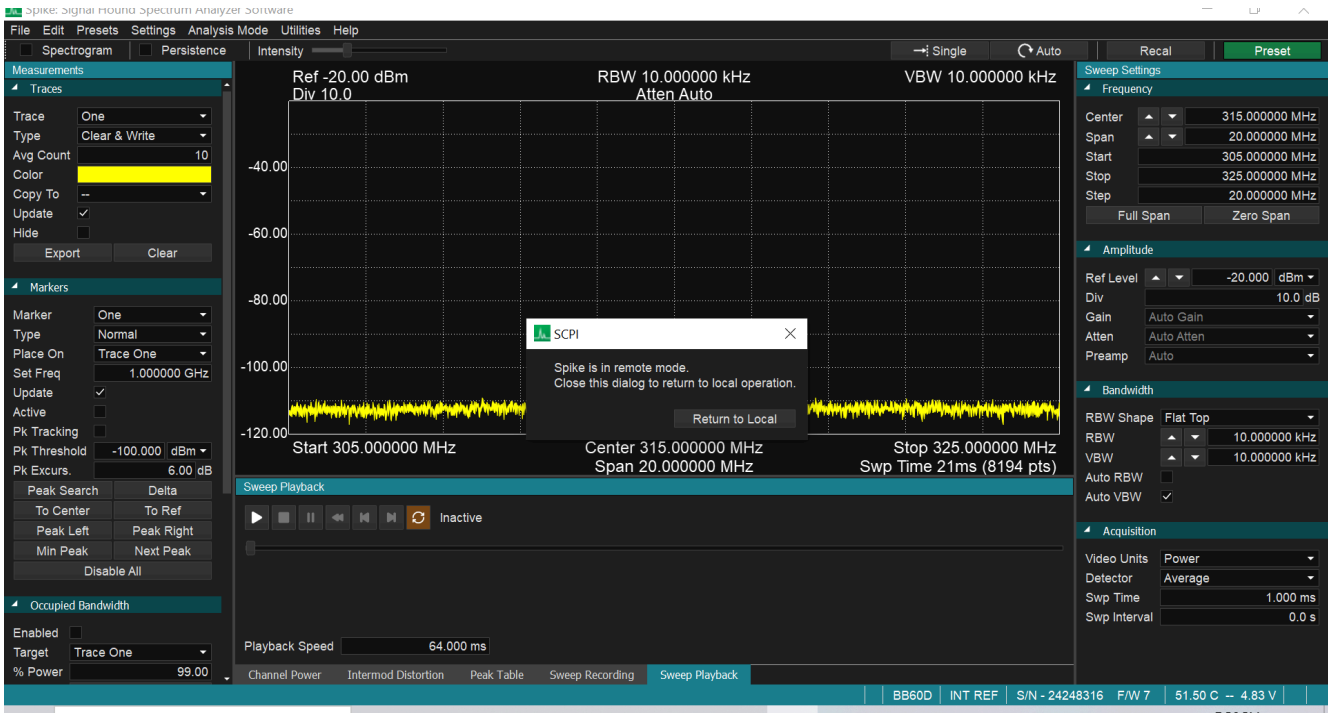


Figure 8.31 *Spike SDK*

Next, we had to send the data across a real time database. For the demo, we opted to use Firebase. There were two different sides that could be seen. First, we will show the client or instrument side where the data was processed and showed our spikes in a signal. Then, the real time database, which will update when we have received something that sets an alarm. It was labeled data with the date, magnitude, and the time.

The images shown below represent what both sides would see when sending data. First, we have the data being processed locally on the device. It of course shows a lot of valuable information. Such as when the data stream starts and stops. The center frequency and span. The reference level is the RBW and finally and most importantly the detection threshold. Meaning that the signal has to be over that threshold to be sent to the client. On the receiving / client side we can see what the data will look like. It will send it in packets that can be opened at any time. Inside of the packet we get only the most valuable information such as, the date, the magnitude, and the time. This is setup for a specific frequency so it is assumed that it is known information. Going forward it will likely have the frequency as well as we expand our range and algorithm.

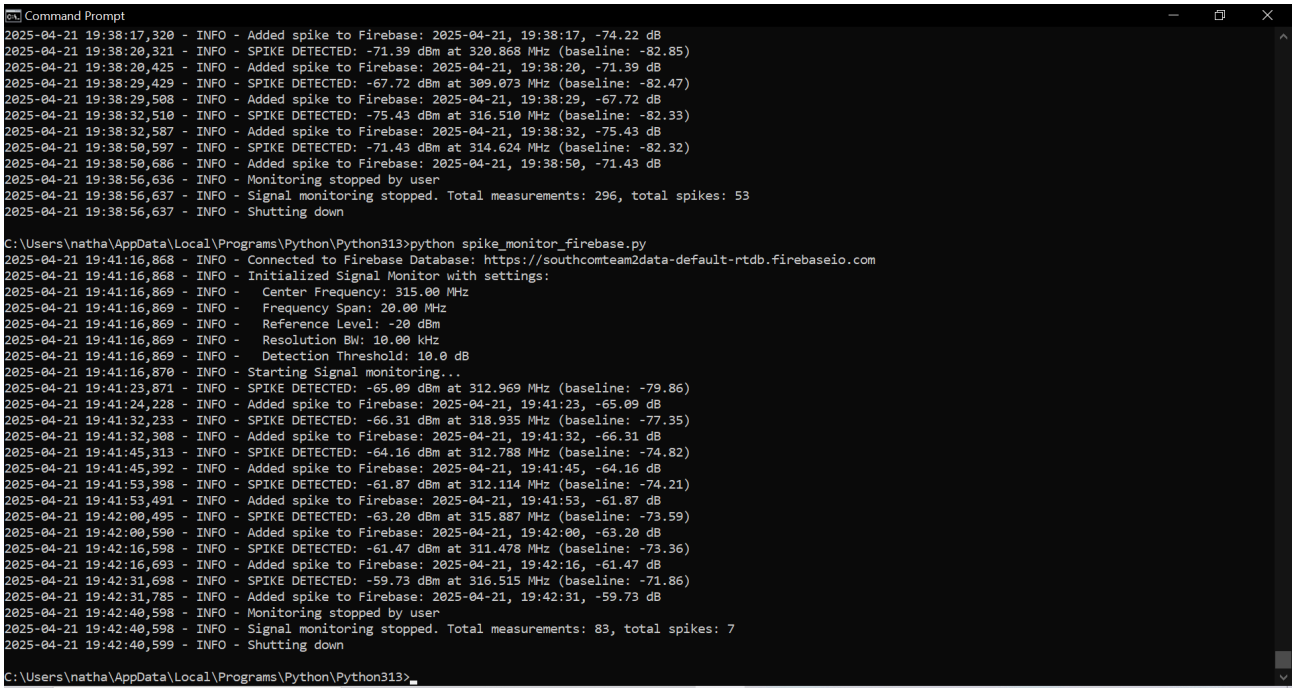


Figure 8.32 *Data from RF device*

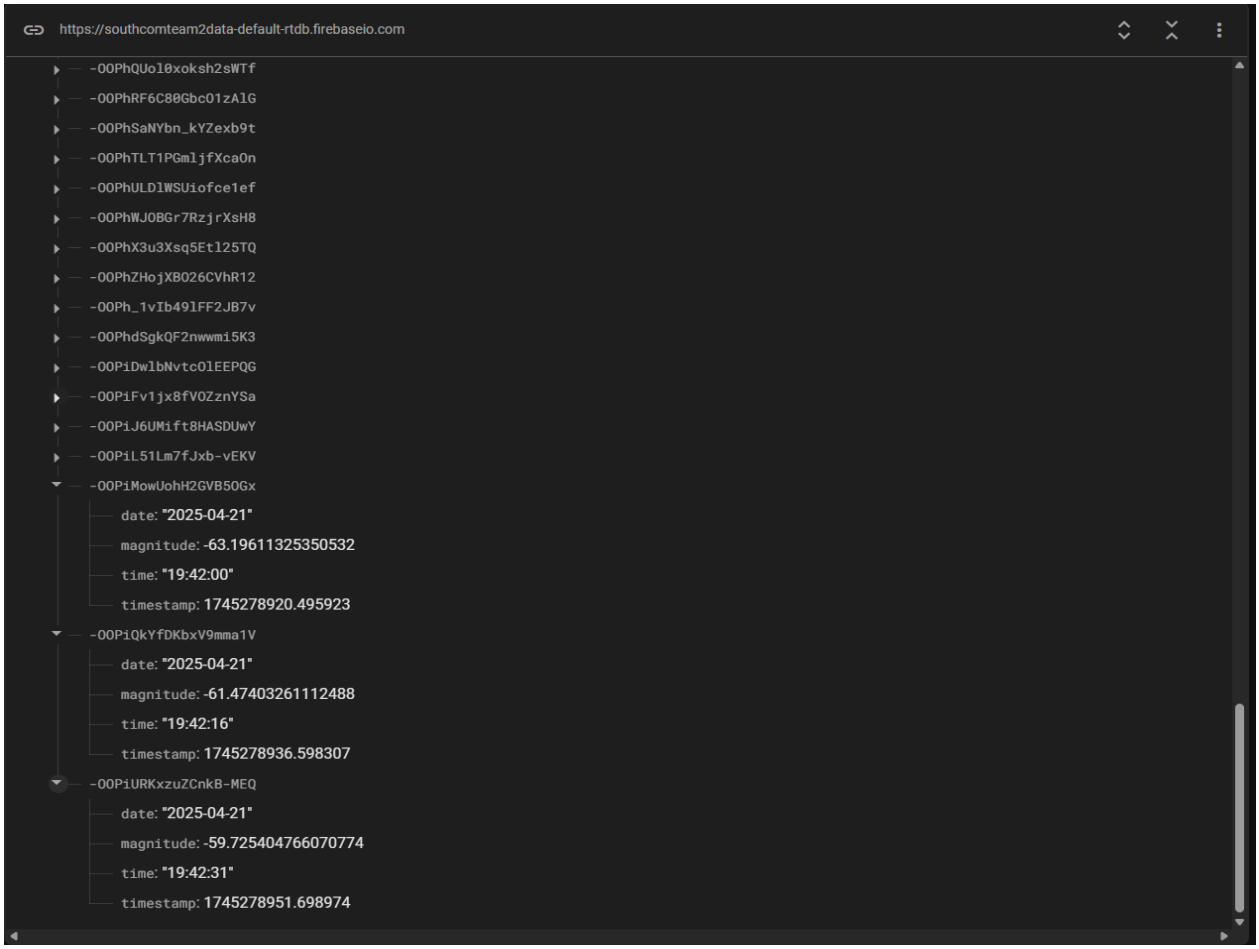


Figure 8.33 *Data at receiving end*

Conclusion

It is important to note that this kind of real time database will work over any network. For example, Nathanael was able to send the data from his home network 20 miles away. Then, I was able to see when he would activate his key fob in real time. This is incredibly useful as we go further, this will be implemented using Starlink. It will provide a great demonstration showcasing the ability of not only our equipment, but Starlink as well. However, this testing was highly successful as we were able to transfer the data across the internet with relative ease. It is extremely easy for the end user to be able to access this data and view when an incursion may occur. Which is the final objective for our project. As we go further, the algorithm will be much more sophisticated. However, this is a great preview into what we will be able to accomplish with our software.

8.3 Object Detection Testing

For this part of the project, we wanted to make sure our camera system could actually detect objects the way we planned. Since we're using a bunch of low-cost cameras and combining their images into one higher-resolution output, it was important to test whether the final image had enough detail to spot and recognize objects. Unfortunately, for this testing phase we did not have any super high resolution photos. However, with the test performing well for low resolution photos, high resolution photos should produce better results since far away objects will be more visible. We ran tests with photos at different distances and lighting conditions to see how reliable the AI detection really was.

To perform these tests, we made a Python script that uses multiple tools to accomplish detecting an object in a photograph. First is OpenCV, which is a really popular tool for working with images and video. It's written in C++ but has a Python wrapper that makes it super easy to use in Python scripts. We used OpenCV in this test because it has a ton of built-in functions for image processing, methods to read images, resizing images, detecting edges, and stitching multiple images together. For this test, we used OpenCV to export the output image after the script was done, as the AI model took care of opening the image, although OpenCV comes with the same functionality.

Next, we used PyTorch, an open-source machine learning library that's mainly used for deep learning and building neural networks. It was developed by Facebook and has become super popular because it's flexible and easy to work with, especially when compared to older frameworks like TensorFlow. One of the best things about PyTorch is that it lets you build models dynamically, which means you can change things on the fly during training instead of being locked into a predefined structure. In our case, if we wanted to build a model to help with object detection in images, PyTorch would be one of the top choices because it gives us access to tons of pre-trained models and makes it easier to train our own if needed. For this test, we used PyTorch to load YOLO, the object detection model when running the script. PyTorch is also responsible for opening the initial image, and making changes to the image, such as drawing a box and labeling objects.

Other smaller libraries were used to test object detection, such as NumPy. NumPy is a Python library used for working with arrays and doing math operations on large sets of numbers. It's

used as the backbone for a lot of scientific computing in Python. What makes NumPy useful is that it's faster than regular Python lists, especially when you're dealing with image data or anything that needs a lot of matrix operations. For this test, we used NumPy to store the tensor detections from PyTorch. This made manipulating the detection coordinates and values faster and easier. The last library used was Argparse, a Python library allowing you to build command line interfaces for scripts. When running a Python script, you can use Argparse to add input parameters, generate help messages, and add default settings. We used Argparse to give the script the file path of the image we want to test, and the path we want for the output image.

The model we used for object detection was YOLO (You Only Look Once). YOLO is one of the most well-known object detection algorithms, being used for many applications. YOLO is fast and efficient, which is important for us as we want to be able to run this on most devices, while minimizing performance issues. Unlike older approaches that scan over parts of an image multiple times, YOLO looks at the image once and predicts all bounding boxes and class probabilities in one shot, making it suitable for real-time applications, like surveillance and tracking applications. There are many different versions of YOLO, with some being more optimized for performance and more lightweight, while others are more accurate but use more resources. We went with YOLOv5, as it's optimized to be lightweight and have improved performance, while still being accurate enough for our use case.

Since we didn't always have access to real-time footage or physical objects for testing, we used images from Google Images to build a small collection of images to be used for testing. We searched for various aircraft types, planes, helicopters, drones, and other flying objects, at different distances and angles, with varying weather conditions. We also tested images with birds in it, since when looked at from below birds and planes have the same shape, which could set off false positives. This gave us a decent variety of sample images to test detection accuracy and resolution performance. It wasn't perfect, since lighting and quality vary a lot between images, but it helped us test how well models like YOLOv5 could recognize targets in various conditions. It also lets us simulate different real-world scenarios before running physical tests with the actual hardware.

Metrics For Success

To figure out whether the object detection system was working the way we wanted, we had to define some clear success metrics. The main metric was seeing if we can detect an object in an image. For each test, we checked if the object was not just visible, but also recognizable enough for YOLOv5 to detect it with a high confidence score, usually above 0.6. We also looked at how detection changed based on distance, lighting, and resolution. We also tracked detection consistency, meaning, could we detect the same object multiple times under similar conditions. Lastly, we watched for false positives, as we only want to detect aircraft, such as planes and helicopters, and not birds. This is important as, from below, birds have the same shape as planes when far away, and a normal edge detection algorithm would not be able to know the difference. We also made sure the detection algorithm wasn't marking objects that weren't actually visible in the original image.

Testing Procedure

Using our gathered collection of images, mainly sourced from Google Images, we ran them through the Python script, recording the results of each image. First, we tested images with clear conditions and a clearly visible flying object. This acted as our baseline to make sure YOLOv5 was working correctly and could detect obvious objects without any major issues.

Figure 8.34 *Sample image of plane in clear sky*

After running the image through the detection script, a new image was generated, which was a copy of the original image but with a bounding box and label showing the detected object. In this example, YOLOv5 successfully detected an airplane in a clear sky, with a confidence score of 0.91. This confirmed that the model was able to identify objects in plain view without any obstructions or background noise interfering with the result.

Figure 8.35 *Result image of plane in clear sky*

This showed that the detection algorithm is capable of reliably identifying flying objects when conditions are ideal, good lighting, high contrast, and the object being large enough in the frame. Next, we wanted to make sure false positives did not occur. We used a few images that included birds in the sky to test whether YOLOv5 would mistakenly label them as aircraft. The script outputted that no flying objects were detected, which is the result we wanted. This helped

confirm that the model could tell the difference between actual aircraft and smaller objects that might look similar. Next, we used images of planes in stormy conditions, to know if the detection algorithm can detect objects when they are partly obstructed by clouds.

Figure 8.36 Result image of plane in cloudy sky

The model excelled in detecting the plane in stormy conditions, with 83% confidence. This result was promising because it showed that even when the background was busy and lighting wasn't ideal, YOLOv5 was still able to pick up key features and identify the object correctly. Cloud cover didn't seem to throw off the model as long as enough of the plane was visible.

We repeated this test with a few more cloudy and low-visibility images, and while the confidence scores were slightly lower on average, the model still performed well. There were no major false positives, and detection remained consistent as long as the object wasn't too small or completely blended into the background. These tests helped confirm that our detection system can still function in less-than-ideal conditions, which is important if we plan to use it outdoors or in real-world scenarios where lighting and weather aren't always predictable.

Moving forward, we'll continue testing with super-resolution enhanced images to see how much further we can push detection accuracy, especially in cases where objects are far away or partially obscured.

Results

After running all the tests, we were able to get a clear picture of how our image-based detection system performed. For images with clear lighting, minimal background noise, and high visibility of the object, YOLOv5 consistently detected targets with confidence scores ranging from 75% to 95%. These were mostly images where the aircraft or drone was centered, well-lit, and large enough in the frame to pick up strong features.

One of the biggest issues we noticed was with lower-resolution images. When the image quality dropped, either from poor lighting, compression, or distance, the detection results took a hit. In some cases, the model failed to detect anything at all, even when a plane was technically visible

to the human eye. This was especially true when the object was too far away or too small in the frame. YOLOv5 struggled to identify any useful features, and confidence scores either dropped below the detection threshold or didn't register a detection at all.

That's where super-resolution comes into play. The plan is that by using a stitched image from multiple cameras, we can recover enough detail to allow the detection model to recognize targets even at longer ranges. If the super-resolution works as expected, it should solve the resolution problem and make object detection more reliable across a wider range of scenarios.

8.4 Anomaly Detection Algorithm Testing

To evaluate the efficiency of the Z-Score / Statistical Thresholding algorithm for anomaly detection in our passive RF system, we made a quick prototype using some fake data with anomalies in it. To do this, we generated a stream of numeric values that would simulate the kind of signal amplitude readings we'd get from the spectrum analyzer, mostly low-level background noise with occasional higher values representing potential aircraft reflections. We intentionally added a few spikes to mimic what a plane might look like as it passes overhead. The main goal here was to see how well the algorithm could flag those spikes as outliers in real time.

Testing Procedure

The algorithm worked by calculating the mean and standard deviation of the incoming data, and any new data point that exceeded a certain number of standard deviations from the mean was flagged as an anomaly. For initial testing, this approach was very responsive, clearly catching each of the spikes we added to the stream. But we noticed a key problem very quickly, once a spike was detected, it was still being factored into the rolling statistics, which caused the mean to rise over time. This made the algorithm less sensitive with each new anomaly, especially if the anomaly was prolonged, like in the case of an aircraft hovering or slowly moving overhead.

To counter this, we modified the algorithm to use a rolling median instead of a rolling mean. The rolling median gave us a more stable and robust baseline because it wasn't as heavily affected by outliers. We also calculated a rolling median absolute deviation (MAD), which helped us define a threshold range that didn't move too much, even with multiple anomalies. After implementing this change, the algorithm was able to detect each spike consistently without losing sensitivity over time.

In addition to streaming data, we also tested the Z-Score / Statistical Thresholding algorithm using batches of pre-collected data. This was useful for simulating scenarios where we might analyze signal recordings after the fact or run detection on periodic chunks of RF data captured in the field. For each test, we fed the algorithm a fixed-length array of signal amplitude values that included both normal background noise and manually inserted anomalies that mimicked aircraft reflections. By using entire batches at once, we were able to fine-tune the detection thresholds more precisely and observe how the algorithm responded across a wider snapshot of activity.

One of the key benefits of batch processing was the ability to visualize the full statistical distribution before applying any thresholds. This gave us a much clearer idea of how far outliers

deviated from the baseline, and made it easier to choose a standard deviation threshold that worked well for our use case. It also helped us experiment with techniques like using the median and median absolute deviation (MAD) as a more robust alternative to the mean and standard deviation, especially when the batch contained multiple anomalies that could skew the results.

Overall, batch testing allowed us to validate the consistency of the algorithm and ensure it would still work in situations where real-time detection isn't possible, like when storing data remotely and reviewing it later. It also gave us a controlled environment to test parameter sensitivity and edge cases, which is hard to do with live data. This helped us build confidence in the algorithm's performance before deploying it in a real-world setup.

This flexibility is especially useful when considering different deployment environments. In situations where constant data streaming isn't possible, due to bandwidth limits, connectivity issues, or power constraints, the batch processing mode offers a solid alternative. We can prioritize capturing and storing raw data on-site, then either transfer it periodically over a secure link or analyze it locally in timed intervals. Additionally, when we do have stable and fast connectivity (like via Starlink or a local server), we can switch to real-time streaming mode for immediate detection and faster response. Having an algorithm that supports both modes seamlessly opens up a lot of options in how we design the rest of the system, especially the parts responsible for communication and power usage.

These tests were all done offline using pre-recorded or simulated data, but they gave us confidence that this approach could scale to real-time analysis when we hook it up to live RF data. The simplicity of the Z-Score algorithm, especially with our modifications, makes it ideal for deployment on a lightweight system where computing power is limited. Later on, we plan to do similar tests using real-time signals from the antenna and spectrum analyzer to further validate its effectiveness in the field.

Results

After running our tests on both real-time streamed data and batch-processed datasets, the Z-Score / Statistical Thresholding algorithm showed strong performance in detecting anomalies under different conditions. In the streaming tests, simulated aircraft reflections that lasted for 10 consecutive data points were consistently detected when using a threshold of 2.5 standard deviations above the rolling median. With the standard mean-based approach, we observed a noticeable drop in sensitivity after the first few points in each burst due to statistical skewing. Switching to a rolling median and MAD completely fixed this issue, maintaining consistent detection across the full 10-point anomaly duration.

In batch mode, the algorithm identified all injected anomalies with a high degree of accuracy. Because the entire dataset was available up front, we were able to tune the detection threshold more easily and validate detections visually. The use of robust statistics like the median also made it less sensitive to outliers, ensuring that short-term spikes didn't skew the analysis. In both modes, false positives were minimal, and the system responded reliably to different anomaly patterns we tested, ranging from isolated spikes to sustained signal bursts.

The algorithm's consistent behavior across both real-time and batch modes confirms its suitability for deployment in a range of environments. Whether the data is streamed continuously from the antenna or stored and processed in batches, the detection logic remains lightweight, responsive, and effective. This versatility not only improves reliability in the field but also simplifies the data pipeline, since one algorithm can be used regardless of the system's connectivity or deployment constraints.

Chapter 9: Administrative Content

9.1 Finances

Table 9.1 Preliminary Budget Table for RF Device

Item	Approximate Price
Antenna(x2)	$900
Signal Analyzer	$5000
Solar / Power System	$420
Android Phone	$500
PC/laptop	$500
Total:	$7,500

Table 9.2 Preliminary Budget Table for Camera Array

Item	Approximate Price
Cameras (x3)	$50 each
Lenses (x6 but used 3)	~$22-$33 each
USB Hub	~$30
Material for Housing (3D Printed)	$15
Total:	$285

The budget for this project is set by our sponsor, Roy H. Wubker. The goal of this project is to create a cost-effective solution. Components that require a higher standard of quality were allocated a larger portion of the total funds. As it currently stands, many of the components could

be subject to change based on quality versus cost assessments or customer standards. In addition, the solar power system may not end up being part of the final product, but we have funds allocated for it in the case that it does become a part of the system.

While the main part of this project, the RF device, receives majority of the funding, the optical components were sponsored by SOUTHCOM after a request. As was mentioned earlier, the sponsor's main focus is an RF detection device, however, since the optical part of the project would pique the interest of the same customers as the RF device, a prototype setup will be demonstrated to show the potential of this application in hopes that more funding and time go into this area of research.

The cost per unit comes out to around $7000, but since the sponsor provided all the materials for our project, the out of pocket cost for us is $0. The price of the unit, which is flexible based on optimization and inclusion or exclusion of the optical element portion, is a much more appealing price especially since the user would need multiple to localize effectively. Compare this to radar systems from big companies that do contract work with the government. Our device is much more affordable while still providing essential information. While we do not want our device to cost the user anywhere near the price of a radar system, there are a lot of potential enhancements that can be made to elevate the quality of the unit.

9.2 Project Timeline

Below we have created both a table and Gantt chart to help organize our group and get a clearer view of the approaching deadlines. We have currently set up our schedule in even increments of about 3 weeks per step. This may turn out to be unrealistic as certain milestones may take more or less time to achieve. The main idea, however, is to give us insight into the quick turnaround we will need to have, especially going into the summer session of senior design 2. In addition to the dates listed below, our group met regularly with Dr. Chan and Group D to ensure the progress on the two projects was on pace.

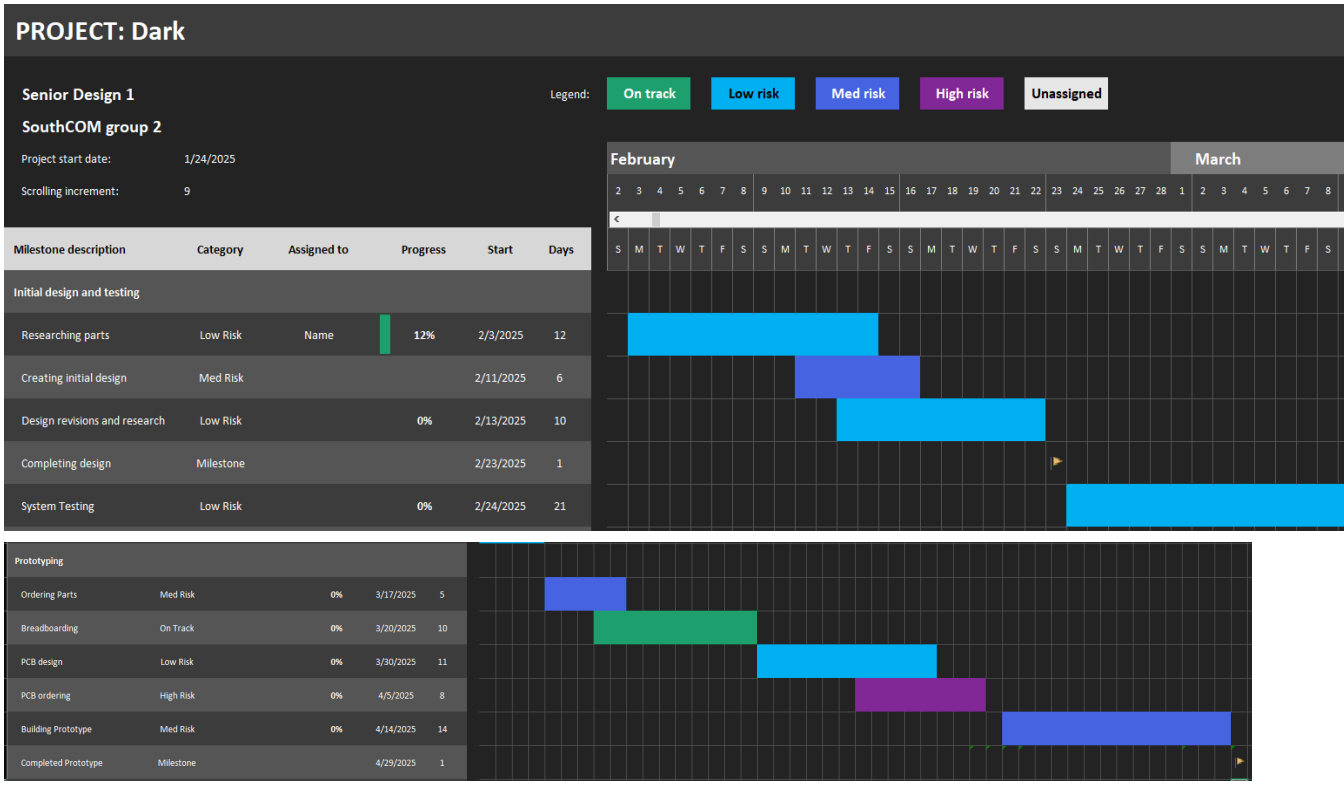
Table 9.3 Senior Design I Milestones

Milestone	Estimated Start	Estimated End	Duration
System Design	02/03/2025	02/23/2025	3 weeks
System Testing	02/24/2025	03/16/2025	3 weeks
Breadboard Prototyping	03/17/2025	03/30/2025	2 weeks
Finished Prototype	03/31/2025	04/29/2025	4 weeks

Table 9.4 Senior Design II Milestones

Milestone	Estimated Start	Estimated End	Duration
Multiple prototypes made	05/19/2025	06/16/2025	4 weeks
Designing Array setup	06/02/2025	06/22/2025	3 weeks
Testing Array	06/23/2025	07/20/2025	4 weeks
Final Product Completed	07/21/2025	08/23/2025	2 weeks

Gantt Chart for Schedule:



Chapter 10: Conclusion

In conclusion, the Southcom Passive Aircraft Detection project is a large leap in remote RF sensing. The system will be capable of detecting things such as aircraft or radio transponders in areas that are remote and have limited access to expensive active radar systems. We have first looked at the design constraints given to us by our sponsor. We will take into account the amount of power that the system will require, the amount of processing power, and the reliability of such a system.

All of the components that are used have no ITAR restrictions as specifically requested by the sponsor. This will allow this to be made into a product and exported elsewhere to countries that may be in need of such a cheap system. Doing so will improve our relationships with them and allow them to combat drug smuggling or other illicit activities. Using a passive system also allows us to use the system anywhere as we are not in violation of FCC guidelines.

The initial design and testing of the system have been done by the UCF senior design team and will work as the reference as to how our system will be implemented as we find and create solutions to any problems that arise as we go further.

There are a few major components of our system that really make the project unique. First will be our RF technology such as the spectrum analyzer and antenna. We started by using the Signal Hound BB60D which is a high end spectrum analyzer that comes with some very useful software. Next, we have our antenna which needs to cover a large range of frequencies. As a result, we used a Log periodic dipole antenna from Aventas. This is important as it has a very high forward gain and a large range of frequencies. The antenna also has a preamplifier which will help us to find the minute differences that may be hiding in the analyzed signals. These two components will prove extremely useful in our mission as we try to passively detect RF anomalies in remote locations.

Next, we have used some very clever hardware that is not only extremely powerful but is also power efficient. We are using our Intel NUC which has incredible computing capabilities with its i7 processor, but also has incredibly low power draw which will prove extremely useful in our pursuit of longevity. With our incredible hardware, we will have an incredible software algorithm that will be able to pick out something as small as the reflections of radio waves from an aircraft. It will also have a web app that is simple to use and will allow any user that may know nothing of RF to use it and make effective decisions. Combining these two things we will have an incredibly accurate and efficient system that will work entirely by itself.

We also have the simplicity of our system such as the fact that it is powered from one source. We will be using a WestingHouse power supply to power all of our components. Not only that, but our spectrum analyzer will be powered by the USB header of the NUC which is incredibly useful in decreasing the amount of cables hanging off.

The ruggedness and design of our system has also been greatly considered. We have considered our sponsor's end use which is to place these in the middle of humid and hot jungles. By pre-planning for the obstacles that may be faced in the jungle, we will simply implement them as we go to create enclosure for the systems. Using PWM fans to keep them cool and also layering all of the sensitive electronics with a coating of silicone spray.

All of this combined will prove as a powerful tool for our sponsor as we go to develop an entirely unique remote RF sensing device. All of the documentation above will serve as the starting and end point of our research and the difficulties that we faced through our testing and learning as we developed our product.

For the camera array, the application of super resolution techniques appears to have opened the door to new and exciting ideas in the world of surveillance and defense. Though our setup for this project uses only 3 cameras priced at a mere $50 with the addition of our own lens systems, it shows the potential of this setup and approach. If this approach was taken up with a larger budget, more time, and access to more innovative resources, this project could quickly make the leap into an advanced yet affordable super resolution system. The applications could then expand beyond aircraft detection and could include general security camera enhancement, anti-drone defense, and could even expand beyond defense applications.

In the ideal case, the optical components and RF device would be integrated together to provide the best possible device for the user. While in our project, this was not the case, our research and investment have opened the door to the possibilities of having devices that can greatly increase defense and surveillance capabilities at an affordable cost.

Appendix:
References

Dedrone, "Dedrone introduces localization technology to stop unauthorized drones," [Online]. Available: https://www.dedrone.com/press/dedrone-introduces-localization-technology-to-stop-unauthorized-drones. [Accessed: Jan. 23, 2025]

Dedrone, "RF Sensors Overview," [Online]. Available: https://www.dedrone.com/products/drone-detection/rf-sensors/overview. [Accessed: Jan. 23, 2025].

Robin Radar Systems, "ELVIRA Radar: 2D Drone Detection Radar," [Online]. Available: https://www.robinradar.com/products/elvira-radar. [Accessed: Jan. 21, 2025].

Robin Radar Systems, "Drone Detection Radar Solutions," [Online]. Available: https://www.robinradar.com/solutions/drone-detection-radar . [Accessed: Jan. 21, 2025].

Hensoldt, "TwInvis Passive Radar: Surveillance of noiseless objects," [Online]. Available: https://www.hensoldt.net/products/14/twinvis-passive-radar-surveillance-of-noiseless-objects . [Accessed: Jan. 20, 2025].

Defense News, "Hensoldt unveils a deployable package of its TwInvis passive radar," Aug. 10, 2023. [Online]. Available: https://www.defensenews.com/global/europe/2023/08/10/hensoldt-unveils-a-deployable-package-of-its-twinvis-passive-radar/. [Accessed: Jan. 20, 2025].

Al-Harasis, R., & Sababha, B. H. (2019). On the Design and Implementation of a Dual Fisheye Camera-Based Surveillance Vision System. Springer.

Brady, D. J., Hu, M., Wang, C., Yan, X., Fang, L., Zhu, Y. (2020). Smart Cameras. arXiv preprint.

Cherniakov, M. (2008). *Bistatic Radar: Emerging Technology*. John Wiley & Sons.

Chu, J., Dong, L., Liu, H., Lü, P., Ma, H., & Peng, Q. (2020). High-resolution measurement based on the combination of multi-vision system and synthetic aperture imaging. Optics and Lasers in Engineering.

Firoozfam, P. (2004). Multi-camera Imaging for 3-D Mapping and Positioning: Stereo and Panoramic Conical Views. Citeseer.

Griffiths, H., & Baker, C. (2005). "Passive Coherent Location Radar Systems." *IEEE Proceedings on Radar, Sonar & Navigation*, 152(3), 153-164.

Griffiths, H., & Long, N. (2013). "Renewable Energy Solutions for Passive Radar Systems." *Journal of Radar Systems Engineering*, 8(2), 79-92.

Harfouche, M., Kim, K., Zhou, K. C., Konda, P. C., Sharma, S. (2023). Imaging Across Multiple Spatial Scales with the Multi-Camera Array Microscope. Optica.

Team, Editorial. "Everything RF." *Everything RF*, everything RF https://www.everythingrf.com https://cdn.everythingrf.com/logo.png 236 48, 2 Oct. 2021, www.everythingrf.com/community/what-is-an-isotropic-antenna.

Willis, N. J. (2011). *Bistatic Radar*. SciTech Publishing.

· · Chaykina, N., Sychkov, V., & Kropachev, A. (2019). "Lithium-Ion Batteries in Modern Power Applications." *Journal of Energy Storage*, 25, 100822.

· · Kumar, R., & Jain, A. (2022). "Comparative Analysis of LiFePO4 and Li-ion Batteries for Renewable Energy Storage." *IEEE Transactions on Energy Conversion*, 37(4), 879-889.

Lin, C. W., Chen, K. W., Chen, S. C., Chen, C. W. (2015). Large-area, multilayered, and high-resolution visual monitoring using a dual-camera system. ACM Transactions on Multimedia Computing, Communications, and Applications.

Li, L., Tan, B. (2024). A Multi-View Reference Image Super-Resolution Framework for Generating Large-FOV and High-Resolution Image. Journal of Visual Communication and Image Representation.

Yuan, X., Fang, L., Dai, Q., & Brady, D. J. (2017). Multiscale Gigapixel Video: A Cross Resolution Image Matching and Warping Approach. IEEE International Conference on Image Processing.

· · Wang, X., Zhang, Y., & Li, H. (2021). "Advancements in Lithium Iron Phosphate Battery Technology for Industrial Use." *Energy Reports*, 7, 2493-2505.

· · Whittingham, M. S. (2004). "Lithium Batteries and Their Applications." *Journal of Power Sources*, 136(2), 345-350.

"The comparison of CCD and CMOS sensors," ResearchGate. Available: https://www.researchgate.net/figure/The-comparison-of-CCD-and-CMOS-sensors-55_fig3_370578099.

"What is global reset release? How does it work in e-con cameras?" e-con Systems, [Online]. Available: https://www.e-consystems.com/blog/camera/products/what-is-global-reset-release-how-does-it-work-in-e-con-cameras/.

"What is Panorama mode and how do I use it?" Samsung, [Online]. Available: https://www.samsung.com/uk/support/mobile-devices/what-is-panorama-mode-and-how-do-i-use-it/#:~:text=Panorama%20mode%20allows%20you%20to,how%20do%20I%20use%20them.

"Funny animal panorama fails," Bored Panda, [Online]. Available: https://www.boredpanda.com/funny-animal-panorama-fails/?media_id=1631163.

"Arducam 16MP autofocus camera module," Amazon, [Online]. Available: https://www.amazon.com/Arducam-Distortion-Microphones-Computer-Raspberry/dp/B096M5DKY6.

"IMX219-160 Camera, 160° FOV, Applicable for Jetson Nano," Seeed Studio, [Online]. Available: https://www.seeedstudio.com/IMX219-160-Camera-160-FOV-Applicable-for-Jetson-Nano-p-4603.html.

"4K USB camera," e-con Systems, [Online]. Available: https://www.e-consystems.com/4k-usb-camera.asp.

"Camera Module 3," Raspberry Pi, [Online]. Available: https://www.raspberrypi.com/products/camera-module-3/.

"Anker power bank," Anker, [Online]. Available: https://www.anker.com/products/a7515?variant=37437056745622.

"StarTech.com 4-port USB 3.0 PCIe card," Amazon, [Online]. Available: https://www.amazon.com/StarTech-com-Port-USB-3-0-Built/dp/B00DSCYSB4.

"StarTech.com 4-port USB 3.0 hub," Amazon, [Online]. Available: https://www.amazon.com/StarTech-com-Port-USB-3-0-Hub/dp/B00SCE4E0I.

"Plugable 7-port USB charging station," Amazon, [Online]. Available: https://www.amazon.com/Plugable-Port-USB-Power-Adapter/dp/B075NMVGP7.

B. S. Carlson, "Comparison of modern CCD and CMOS image sensor technologies and systems for low resolution imaging," SENSORS, 2002 IEEE, Orlando, FL, USA, 2002, pp. 171-176 vol.1, doi: 10.1109/ICSENS.2002.1037011. keywords: {Charge coupled devices;CMOS image sensors;CMOS technology;Image resolution;Charge-coupled image sensors;Consumer electronics;High-resolution imaging;Digital cameras;Cams;Power dissipation},

Team, Editorial. “Everything RF.” *Everything RF*, everything RF https://www.everythingrf.com https://cdn.everythingrf.com/logo.png 236 48, 2 Oct. 2021, www.everythingrf.com/community/what-is-an-isotropic-antenna.

“Antenna Theory - Antenna Arrays - Tutorialspoint.” *Tutorialspoint.com*, 2019, www.tutorialspoint.com/antenna_theory/antenna_theory_arrays.htm.

“Antenna Theory - Collinear Array.” *Www.tutorialspoint.com*, www.tutorialspoint.com/antenna_theory/antenna_theory_collinear_array.htm.

“Antenna Theory - Log-Periodic Antenna - Tutorialspoint.” *Www.tutorialspoint.com*, www.tutorialspoint.com/antenna_theory/log_periodic_antenna_theory.htm.

“Antenna Theory - Parasitic Array.” *Www.tutorialspoint.com*, www.tutorialspoint.com/antenna_theory/antenna_theory_parasitic_arrays.htm.

“Antenna Theory - Yagi-Uda Antenna - Tutorialspoint.” *Www.tutorialspoint.com*, www.tutorialspoint.com/antenna_theory/yagi_uda_antenna_theory.htm.

“Collinear Antenna» Electronics Notes.” *Www.electronics-Notes.com*, www.electronics-notes.com/articles/antennas-propagation/phased-array-antennas/collinear-vertical-antenna.php.

“Log Periodic Dipole Array Antenna.” *Www.antenna-Theory.com*, www.antenna-theory.com/antennas/wideband/log-periodic-dipole.php.

“Planar Log-Periodic Antenna (AX-37A) - Aventas.” *Aventas*, 10 Apr. 2025, aventasinc.com/product/planar-log-periodic-antenna-8/. Accessed 21 Apr. 2025.

“RSA306B Spectrum Analyzer Datasheet.” *Tek.com*, 2015, www.tek.com/en/datasheet/spectrum-analyzer-0. Accessed 21 Apr. 2025.

“The Yagi-Uda Antenna - Yagi Antennas.” *Www.antenna-Theory.com*, www.antenna-theory.com/antennas/travelling/yagi.php.