RF Device Detection, Identification, and Secure Data Transmission using a Free-Space Optical Communications System

Department of Electrical Engineering and Computer Engineering
University of Central Florida

Final Report

Group 1

Members

Zachary Flores, *Computer Engineering*

Bill Noel, *Computer Engineering*

Sarah Thibaut, *Photonics Engineering*

Zackary Zuniga, *Photonics/Electrical Engineering*

Sponsored by: CW5 Richard Godfrey

Reviewers: Beverly Seay, Dr. Peter Delfyett, Dr. Mark Heinrich, Dr. Azadeh Vosoughi

Mentored by: Dr. Lei Wei

Table of Contents

Lithium Iron Phosphate (LiFePO4 or LFP) batteries have gained significant popularity due to their inherent safety advantages compared to other lithium-ion chemistries (like

List of Figures

List of Tables

Chapter 1 - Executive Summary

The electromagnetic spectrum is the new battlefield. In an era defined by information dominance and ever-present wireless communication, the ability to detect and interpret radio frequency (RF) signals is essential for national security, disaster response, and border enforcement. Our project, **RF Device Detection, Identification, and Secure Data Transmission using Free-Space Optical Communication**, aims to address this challenge through a portable, passive sensing system designed by engineering students at the University of Central Florida.

The system fuses three major technological domains: RF signal analysis, optical imaging, and secure Li-Fi communication. By integrating these into a single field-deployable device, the project enables real-time detection and identification of unauthorized RF-emitting devices such as walkie-talkies, drones, or cellular equipment, and relays the collected intelligence to a secure remote terminal, even in environments where traditional wireless communication is jammed or compromised.

RF Detection & Imaging Module: The core unit employs a Software Defined Radio (SDR) coupled with narrowband antennas to monitor key frequency bands in the 400 MHz to 6 GHz range. Once a signal is detected, the system triggers synchronized image capture using visible and infrared cameras, creating a multimodal dataset for analysis. This fusion of RF and optical data provides visual context to signal activity, improving identification accuracy and reducing false positives.

Li-Fi Transmission System: To maintain stealth and security in contested electromagnetic environments, the device transmits its collected data using a Free-Space Optical Communication (FSOC) system. A focused infrared beam is modulated with On-Off Keying (OOK) and directed over distances of up to 1 km to a receiver station. This receiver then decodes the data and interfaces with a web-based application that displays signal origin, type, and associated imagery.

Web and Mobile Visualization Platform: Once data is received via Li-Fi, it is decoded and visualized on a custom-built web application. The application offers dynamic device listings, time-stamped RF signatures, and corresponding imagery, enabling real-time situational awareness. A mobile interface also enables in-the-field viewing, vital for rapid response operations.

Together, these components form a resilient and modular system capable of identifying potential RF threats in real-time and under austere conditions. The system can be adapted for use in military reconnaissance, disaster zone search-and-rescue, spectrum enforcement, or remote infrastructure monitoring.

This document outlines the complete design process of the RF detection system—from conceptual motivation and prior work, through research, part selection, and schematic design, to software architecture, data visualization, and final deployment strategy. Included are hardware and software diagrams, bill of materials, testing protocols, and security considerations. The report concludes with reflections on system limitations, future enhancements, and the broader impact of covert RF detection technologies.

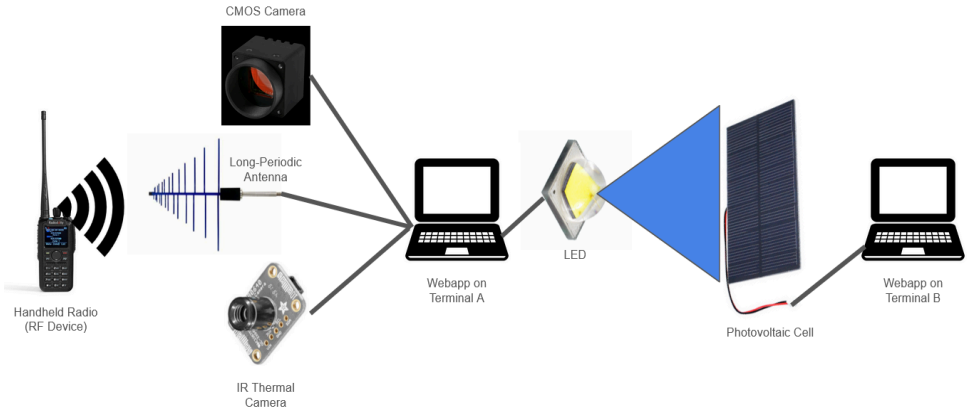
1.1 Motivation and Background

Electromagnetic Warfare (EW) plays a critical role in modern military operations, where control of the electromagnetic spectrum can determine mission success. Unauthorized RF devices on the battlefield pose serious threats, as seen in recent conflicts like Ukraine, where personal cellphones have revealed troop positions with deadly consequences [1][2]. This highlights a major gap: the inability to reliably detect, identify, and neutralize RF threats [3][4]. Our project directly addresses this need by developing a system that enhances situational awareness and enables safer, more effective operations.

Beyond military applications, RF detection has broad civilian use. Telecom companies can use it for better spectrum management and interference detection. Emergency responders can locate survivors via their devices in disaster zones. Border patrol can detect communications devices carried by unauthorized entrants. The FAA can track rogue drones near restricted airspace by identifying their RF signatures. [5][6][7]

This system doesn't just detect RF signals—it transforms them into actionable intelligence. It promises improved safety, security, and operational efficiency across multiple sectors, reinforcing its critical value in today's increasingly connected world.

1.2 Prototype Illustration/Blueprint



The prototype illustration does not include the required lens system, the PCB, or power supply, simply the major components that will be required to capture the data. This diagram is to

showcase the overall design. Our antenna connected to our Li-Fi optical setup will detect devices that are transmitting radio frequencies, and send the collected data (RF data and images captured) to our web application on the computer terminal, which will then be translated and identified.

Chapter 2 - Project Metrics

This project is focused on the development of a compact, passive RF detection system that can capture and classify radio frequency signals, synchronize them with real-time image data, and transmit the combined output to a remote terminal using optical communication. The system is designed to operate in dynamic environments where traditional RF-based communication is either vulnerable or restricted. To ensure a structured and scalable development process, the project goals are organized into three tiers: basic objectives focused on core data acquisition and transmission functionality, advanced objectives that introduce signal classification and multimodal data fusion, and stretch objectives targeting full system deployment through miniaturization, geolocation capability, and ruggedized operation. Each goal is defined with respect to technical feasibility, deliverables, and measurable performance benchmarks.

2.1 Goals

Our goal is to design an RF detector that utilizes object detection, transmit the data to a remote computer, and take an image of the surrounding landscape at the moment of signal detection. By achieving these goals, we hope to improve the tactical awareness of the client.

2.1.1 Basic Goals

- **Collect RF Data:** Design and implement an RF front-end capable of capturing signals in the 400 MHz–6 GHz range with timestamped output for further processing.
- **Collect Image Data from Optical System:** Integrate a camera module with minimum 720p resolution that captures images within 100 ms of RF signal detection, synchronized for real-time analysis.
- **Display Information via Secure Optical Communication:** Transmit RF and image data to a remote terminal using a Li-Fi (optical) communication link that is 1km away with latency under 1 second and a bit error rate below 1%.

2.1.2 Advanced Goals

- **Identify Individual RF-Transmitting Devices:** Apply digital signal processing and machine learning to classify at least five device types (e.g., phone, Wi-Fi, drone) with a minimum of 85% identification accuracy.
- **Display Identified Devices on the Web Application:** Develop a web interface that displays device classifications alongside their frequency and timestamp, updated dynamically in near real-time.

2.1.3 Stretch Goals

- **Make Portable and Compact for Multiple Sensors:** Repackage the system into a rugged, battery-powered unit weighing under 15lbs with modular sensor support and field-deployable housing.
- **Implement Geolocation Capabilities:** Use a dual-antenna array to implement signal triangulation, achieving location estimates within 500 meters in controlled field tests.
- **Improve Deployability by Ruggedizing and Optimizing for Power:** Harden the device for outdoor use with weather-resistant casing and a $LiFePO_4$ battery system providing 24+ hours of continuous operation with solar recharging support.

2.2 Objectives

2.2.1 Basic Development Objectives

The system's **basic** goal is to detect and collect RF signals and image data from the environment, then transmit this data securely via optical communication to a separate display device for an end user. It focuses on capturing high-quality RF and visual information, minimizing latency, and ensuring secure transmission using Li-Fi. A web application will visualize this data for the user in near real-time.

- **RF Data Collection**: Develop an antenna array and RF front-end to capture wide-spectrum signals (e.g., cellular, Wi-Fi, Bluetooth). Components will be chosen for sensitivity and bandwidth to ensure high-quality data.
- **Image Capture**: Integrate an optical imaging system (e.g., visible, thermal) to provide contextual information. Synchronization with RF data will support threat localization and confirmation.
- **Data Aggregation and Pre-processing**: Implement mechanisms for timestamping, synchronization, and noise reduction to prepare RF and image data for low-latency transmission.
- **Secure Optical Communication**: Establish a Li-Fi (Light Fidelity) link to transmit data from the sensing system to a remote terminal with low probability of detection or interception.
- **Web Application Display**: Build a frontend interface to display RF spectrum visualizations (e.g., waterfall plots) alongside image data in real-time, enabling situational awareness.

2.2.2 Advanced Objectives

Building upon the data pipeline, this phase integrates signal classification and machine learning for intelligent identification of RF devices:

- **Signal Feature Extraction and Classification**: Apply DSP algorithms to analyze modulation schemes, frequency hopping, and signal bandwidth. Use machine learning models (e.g., SVMs, neural networks) to classify device types.
- **Image-Based Object Recognition**: Implement CNN-based computer vision to detect RF-emitting objects (e.g., drones, cellphones) in the visual field.
- **Data Fusion**: Combine classified RF signals with identified visual targets to enhance detection accuracy and reduce false positives. For example, correlating a Wi-Fi signal with the presence of a laptop in the same location.\

2.2.3 Stretch Objectives

The final development stage focuses on making the system deployable, rugged, and scalable:

- **Miniaturization and Compact Design**: Redesign hardware with smaller, power-efficient components in ruggedized enclosures for portability and harsh environments.
- **Power Efficiency**: Optimize firmware and hardware for low power consumption. Explore solar recharging or long-lasting $LiFePO_4$ battery packs.
- **Modularity and Sensor Expansion**: Support integration of additional RF antennas, cameras, or environmental sensors for adaptability to various missions.
- **Ruggedization and Environmental Resilience**: Ensure operation under extreme temperatures, humidity, dust, and vibration using industrial-grade components and protective casings.
- **Geolocation Capabilities**: If feasible, implement AoA, TDoA, or signal triangulation using dual antennas to estimate the location of RF-emitting devices.
- **Networked Sensor Arrays**: Develop mesh networking protocols to allow multiple units to communicate and share data for expanded sensing coverage and geolocation precision.

2.3 Project Description / Functionality of Optical Design

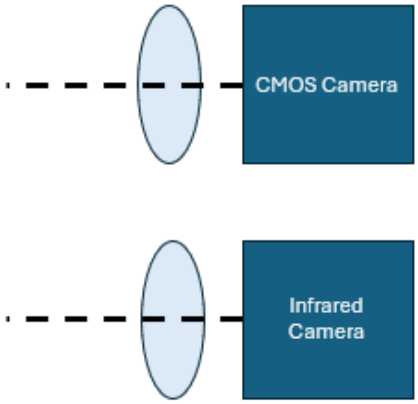
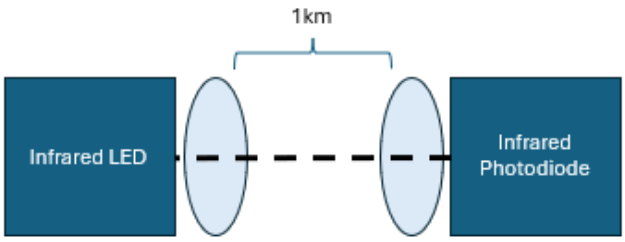


Image Capture Optical System
(Camera-Lens System)



Free Space Optical
Communications (Collimated Light)

Above is a rough draft of our optical design. There is significant work still required to define the focal lengths of each lens and the parameters of distance, however, this will be concluded by the final draft of this document. The diagram depicts the significant components involved in the overall design. The CMOS camera will take an image of the area present of the sensor as well as the infrared camera. This will serve in the identification process of the RF Device detection. The camera should cover a range of 500m with a significant field of view of 60 degrees. The Free Space Optical Communication setup will have a lens system used to collimate the Infrared LED light to travel a 1km distance. This will then be received by an Infrared photodiode attached with a lens to focus the light in, for example an aspherical lens.

Our project endeavors to develop a sophisticated system for the detection and identification of RF devices, drawing significant conceptual inspiration from the principles of "passive radar systems." Unlike traditional active radar, which emits its own radio waves to detect objects, our system will operate entirely passively. This means it will not transmit any signals, but rather intelligently leverage the existing radio frequency (RF) environment. The air around us is constantly filled with a multitude of RF signals emanating from diverse sources – cellular networks, Wi-Fi hotspots, broadcast television and radio, Bluetooth devices, and various other civilian and military transmissions. Our system's core functionality will be to receive and accurately analyze these pervasive ambient RF signals. By processing reflections, distortions, or even the direct emissions, the system will try to detect and analyze target RF devices by observing how they interact with existing RF sources — like cell towers or Wi-Fi routers. The system can infer the presence, characteristics, and potentially the location of specific devices without revealing its own presence. This passive nature is a critical advantage, offering stealth and reducing the likelihood of detection by adversaries.

The scope of our system's analytical capabilities extends far beyond plain signal presence. It will involve advanced digital signal processing to extract nuanced features from the collected RF data. This includes techniques such as spectral analysis to identify specific frequency bands, modulation analysis to discern the type of information being transmitted (e.g., voice, data), and

temporal analysis to understand patterns of transmission. By building a comprehensive library of unique RF signatures associated with various types of RF devices (e.g., smartphones, drones, walkie-talkies, improvised communication devices), the system will be able to classify and identify these devices with a high degree of accuracy.

Crucially, our project augments this advanced RF sensing with multi-modal sensing through integrated image capture. This integration provides a vital layer of contextual information. A high-resolution camera, potentially alongside other optical sensors like thermal or infrared imagers, will capture visual data of the environment where RF activity is detected. The fusion of these two distinct data streams – RF and optical – is where the system truly differentiates itself. Take, for example, a case where a specific RF signature for a cellular device is detected. Simultaneously, the image capture identifies a person holding a phone in the same general area. This correlation dramatically increases the confidence of identification and provides crucial visual confirmation. Furthermore, the image data can assist in fine-tuning the presumed location of an RF source within the visual field, especially if rudimentary geolocation is achieved through RF means.

In essence, the project aims to build a sophisticated "listener" that not only hears the whispers of the electromagnetic spectrum but also understands their meaning, cross-references them with visual cues, and presents this fused intelligence to the operator. This comprehensive approach, inspired by passive radar but expanded with multi-modal data fusion, allows for covert, intelligent, and highly effective detection and identification of RF devices across a wide array of operational scenarios.

In summary, this project has three major core functionalities that must work in tandem to be successful. These three functionalities are: RF Data and Image Capture, RF Data and Image Processing, and LiFi transmission and reception. By using these three functionalities, an RF device should be detected and a separate device that is a significant distance away should be able to receive/be notified that a device was detected.

2.4 Existing Product / Prior Related Work

Modern electronic warfare and situational awareness systems increasingly rely on passive and stealthy sensing technologies. While active radar and wireless communication systems have dominated traditional approaches, current trends favor systems that minimize RF emissions and enhance security, especially in contested or surveillance-heavy environments. Our project builds upon multiple prior technology streams including passive radar systems, optical communication protocols, RF signal sensing, and integrated multimodal detection frameworks. In this section, we examine the most relevant existing technologies and prior research that inform our system’s development.

Passive radar systems represent one of the most conceptually aligned technologies to our project. These systems do not emit signals but instead rely on the detection and analysis of ambient RF emissions—such as commercial radio, TV, or cellular transmissions—to track targets or characterize environments. By using known signal sources and measuring delays and Doppler shifts between direct and reflected signals, passive radar can localize moving objects without revealing the sensor’s location.

Several commercial and academic systems have demonstrated passive radar applications. For instance, Raytheon and Lockheed Martin have developed advanced signal exploitation systems that can identify aircraft and drones using broadcast FM or DVB-T signals. On the academic side, the Silent Sentry system by Lockheed was an early example of leveraging FM radio signals to detect aircraft with high resolution.

However, these systems typically require large antenna arrays, advanced synchronization algorithms, and computationally intensive signal processing. In contrast, our design reduces complexity by focusing on the detection of nearby RF-emitting devices (e.g., handheld radios, WiFi emitters), rather than tracking distant aircraft. Our use case aligns more closely with portable RF signal awareness than large-scale radar surveillance.

Passive radar also assumes a cooperative or at least predictable emitter (e.g., a TV tower or cellular base station), while our system must identify unknown or unauthorized emitters. As such, our detection logic leans more toward RF spectrum analysis and anomaly detection than traditional bistatic radar configurations.

Our system uniquely combines RF detection with optical data transmission, utilizing Light Fidelity (LiFi) as the primary medium for secure communication between the field unit and a receiving station. While LiFi is a relatively recent development in consumer and industrial contexts, optical communication has a long and rich history in military and aerospace applications.

High-speed free-space optical (FSO) communication systems are used by NASA, the military, and satellite constellations for long-range, high-bandwidth data transfer. These systems often involve infrared lasers, adaptive optics, and photodetectors with extremely high sensitivity. In contrast, our system operates over much shorter distances (meters to hundreds of meters) and trades data rate for security and portability.

LiFi differs from infrared FSO by using visible or near-visible LEDs, often in conjunction with basic photodiodes. The appeal of LiFi in our context lies in its inability to penetrate walls or foliage, creating a naturally secure transmission path. Projects such as PureLiFi and Signify (formerly Philips Lighting) have developed commercial LiFi kits for enterprise networking, which have inspired our approach to modulation and transmission.

Nonetheless, consumer LiFi systems often rely on extensive infrastructure, including ceiling-mounted transmitters and photoreceivers built into laptops or smartphones. Our project requires a far more ruggedized, standalone implementation, complete with custom lens arrays, outdoor-capable photodiodes, and embedded microcontroller modulation. Prior work in optical beacons, drone communication systems, and battlefield optical signaling provides design inspiration but lacks the combined RF-to-optical data pipeline that our system employs.

Beyond passive radar, our project borrows heavily from existing RF sensing and spectrum analysis systems. The goal is to detect and identify RF emitters such as handheld radios, WiFi hotspots, and IoT devices based on their spectral signatures. Numerous tools and platforms exist for this purpose, ranging from hobbyist Software Defined Radios (SDRs) to professional spectrum intelligence suites.

One widely used open-source tool is the RTL-SDR, a low-cost USB receiver capable of tuning across a wide RF range. More advanced systems include the Ettus USRP (Universal Software Radio Peripheral) and HackRF, which support real-time IQ data capture and allow developers to build machine learning classifiers for signal types. Projects like GNU Radio have made significant contributions to signal processing pipeline design, allowing developers to implement filters, demodulators, FFT visualizers, and decoders using modular blocks.

Commercial RF signal identification platforms such as Keysight's real-time spectrum analyzers or CRFS's RFeye suite employ AI and large databases of emitter signatures to automate classification. These systems can recognize WiFi, Bluetooth, LTE, and even military waveforms by comparing real-time captures with fingerprint libraries. However, they are often prohibitively expensive and power-hungry.

Our system draws from these methodologies by incorporating:

- FFT-based feature extraction from RF bursts
- I/Q data buffering for classification
- Lightweight machine learning models for signal type inference
- Timestamping and logging for historical pattern analysis

What distinguishes our implementation is the downstream coupling of this analysis with secure optical transmission and field-deployable packaging. We are not merely displaying RF data locally; we are converting it into a secure, transmissible payload for remote evaluation.

Within the realm of signal intelligence (SIGINT), multiple defense-grade systems exist to classify and track RF emitters. These systems often fuse multiple sensors, RF, infrared, visual imaging, to build a comprehensive situational model. While our system is considerably simpler, it draws from the same concept: integrate RF analysis with optical sensing (via CMOS camera) and package results for secure transmission.

Prior research in drone detection, counter-UAS technologies, and border surveillance systems have shown success in identifying RF-emitting devices based on burst timing, modulation type, and spatial signature. Several of these systems use convolutional neural networks (CNNs) to classify signal spectrograms in real-time.

Projects such as DARPA's RadioMap and MIT Lincoln Lab's Adaptive Spectrum Awareness system aim to provide dynamic RF situational maps, especially in congested urban environments. These systems typically require multiple sensors and centralized processing. Our system attempts to miniaturize this capability into a single, portable unit.

A key contribution of our project is the data pipeline architecture: from RF burst detection, to signal snapshotting, to optical exfiltration, with integrity checks and encryption applied at multiple stages. Few open-source or commercial projects implement this end-to-end pipeline in a modular, accessible form.

Commercially available RF detection and communication systems often come at the expense of size, cost, and power requirements. High-end passive radar units can cost tens of thousands of

dollars and require dedicated power and cooling systems. Similarly, commercial LiFi platforms are often optimized for bandwidth, not stealth or resilience.

By contrast, our system is designed with constrained field deployment in mind. Every design decision balances:

- Cost (total system under ~$1,000)
- Power consumption (battery-operated with solar recharging)
- Size and weight (portable, ideally under 5 lbs)
- Functionality (RF detection + image capture + secure LiFi transmission)

We have selected development boards, open-source SDRs, and off-the-shelf lenses to prototype a system that delivers sufficient performance for our mission goals without the overhead of traditional defense systems. Our focus on open standards, modular software, and embedded encryption ensures adaptability while avoiding vendor lock-in.

2.5 Specifications / Engineering Requirements

This chapter outlines the engineering requirements and performance specifications that guide the validation of the RF Detection and Optical Communication System. These specifications represent both qualitative and quantitative benchmarks that the system must meet in order to be considered effective, reliable, and ready for real-world applications. The parameters of interest include geolocation precision, classification accuracy, system latency, and integrated operational functionality.

One of the core objectives of this system is the ability to perform passive RF-based geolocation. This refers to the detection and approximation of an RF emitter's physical location using a combination of signal processing and fixed spatial reference. Our methodology for evaluating geolocation performance involves controlled test environments in which known RF emitters are placed at specified locations, while the system attempts to triangulate or estimate the emitter's position relative to its fixed sensor origin.

The system will rely on methods such as received signal strength indication (RSSI), angle of arrival (AoA), or time difference of arrival (TDoA), depending on the final sensor and software implementation. The accuracy of each location estimate will be compared against the ground truth, with error distances recorded. Acceptable error bounds for field applications will be defined (e.g., ±2 meters for short-range applications or ±10 meters for medium-range scenarios), and the system must demonstrate consistent performance within those thresholds over a statistically meaningful number of trials.

Geolocation evaluations will also consider environment variability such as open field versus obstructed urban environments. This diversity of test settings ensures robustness and generalizability of the system across different deployment scenarios.

Latency is another critical performance metric. In this context, latency refers to the time interval between the introduction of an RF signal into the sensing environment and the presentation of that signal's metadata and classification output on the end-user interface. This parameter is particularly important for time-sensitive operations, such as tactical monitoring, disaster response

coordination, or drone detection, where the value of information is tightly coupled to its timeliness.

The latency measurement will be broken down into several components:

- Acquisition Latency: Time taken for the Software Defined Radio (SDR) to detect and capture the RF signal.
- Processing Latency: Time taken to extract features, perform classification, and generate a response.
- Transmission Latency: Time for modulating, transmitting, and receiving the optical signal via LiFi.
- Display Latency: Time for the backend server to parse the demodulated signal and update the web application.

High-speed logging tools will be placed at each stage to capture time stamps, allowing for detailed analysis of where any bottlenecks occur. The target for total system latency is to remain below 3 seconds, with ideal performance under 1.5 seconds. Systems exceeding 5 seconds will be flagged for optimization. In operational terms, the goal is to ensure that an operator in the field receives actionable insight with minimal delay.

Accurate identification of RF-emitting devices is essential for the usefulness of this system. To measure classification performance, a blind test protocol will be used. A range of RF sources (e.g., handheld radios, WiFi routers, Bluetooth beacons, or unmanned aerial vehicle transmitters) will be activated randomly within the sensing range of the system. The RF classifier, based on either handcrafted features or machine learning, will attempt to identify each source and output a label or identifier.

Correctness will be evaluated by comparing the system output against the known true device type. Accuracy will be defined as the proportion of correct identifications out of the total number of tests. Trials will be repeated multiple times with randomized order and different RF conditions, including variable signal-to-noise ratios.

Metrics to be calculated include:

- Overall Accuracy (OA): Total correct classifications / total trials
- Confusion Matrix: To visualize and understand misclassification tendencies
- Precision and Recall: Per device type, to detect bias or skew in detection capabilities
- F1 Score: A harmonic mean to combine precision and recall into a single performance metric

The benchmark target is a minimum of 90% classification accuracy in controlled environments and at least 80% in field conditions.

While each subsystem (RF acquisition, optical transmission, backend parsing, and user display) can be independently validated, the final system demonstration will assess overall integration. This holistic evaluation involves executing the complete operational workflow from signal detection to final visualization, without any manual intervention or post-processing.

Table 2.1: Components Specifications

Requirements	Specifications	Units
Battery Life: System can sustain operations for extended periods of time.	~24	hrs
Ruggedized	Ability to operate in different climates and weather conditions.	-
Weight: Keep weight below specifications for portability	≤15	lbs

*Green: Demo Parameters

In the table above, we are focusing on obtaining 24 hours of battery life to justify prolonged periods of operations. With this, we must keep a low weight ratio, aiming to hold a total weight of less than 15 pounds. With a light weight and long life-time, we can achieve portability in our device. We aim to make it ruggedized so it can operate in different climates and be operational in an evolving battle space.

Table 2.2: Performance Specifications

Requirements	Specifications	Units
Accuracy: Identifies 100% of movement in area, with set accuracy in identifying objects.	>95%	%
Geolocation Capabilities: System can pinpoint location of object it detects within specified meters.	≤500	meters
Latency: Reduce latency to near zero; transmit data "simultaneously".	0 (Propagation Delay) 1.5 (Processing Delay)	seconds
Detection Range: Ability to detect objects up to a predefined range.	≤500	meters

*Green: Demo Parameters

In terms of our technical specifications, we aim to demonstrate two major features: low latency and a long detection range. We aim to have a 1.5s processing delay and a minimal, near zero propagation delay. We aim to detect up to 500 meters with our optical capturing. Besides these parameters, we want to achieve a 95% or higher accuracy in identifying movement and identifying objects. We also want to achieve a geolocation of 500 meters to pinpoint.

2.6 Hardware Block Diagram

The hardware design of our system is built around the integration of RF signal detection with Free Space Optical Communication (FSOC) to enable secure, long-range data transmission. The diagram shown above outlines the major components and their interconnections, segmented into two primary subsystems: the transmitter (TX) unit and the receiver (RX) unit. Each subsystem is composed of several modular components that work in tandem to perform signal acquisition, processing, communication, and visualization.

Figure 2.1: Design and Layout of Hardware

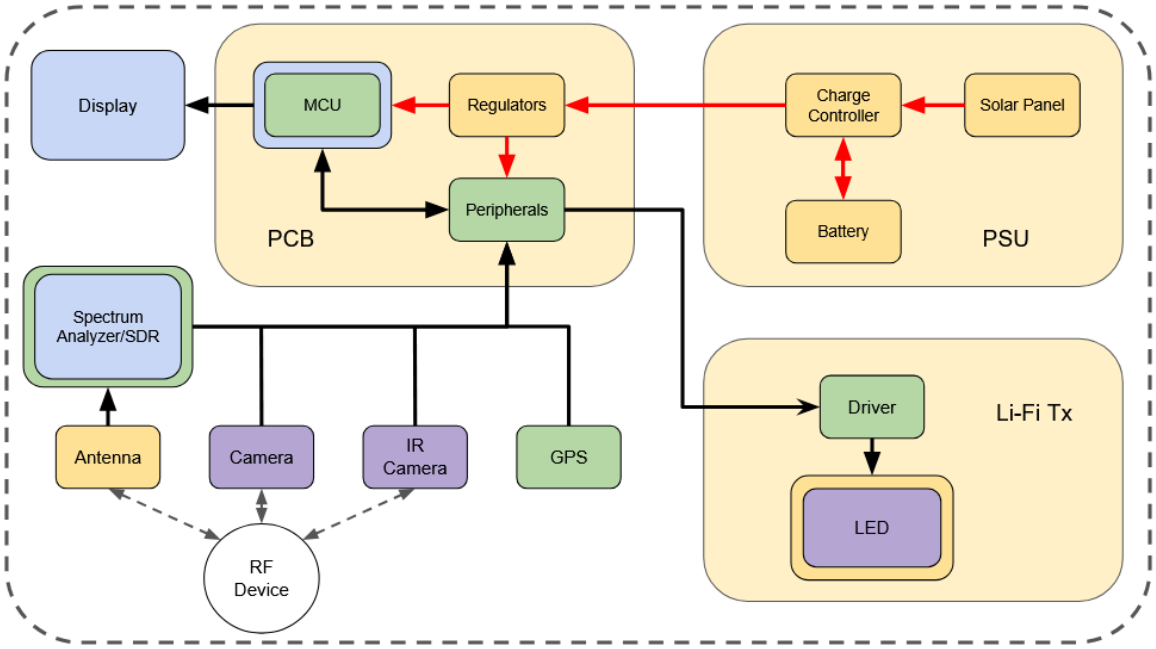
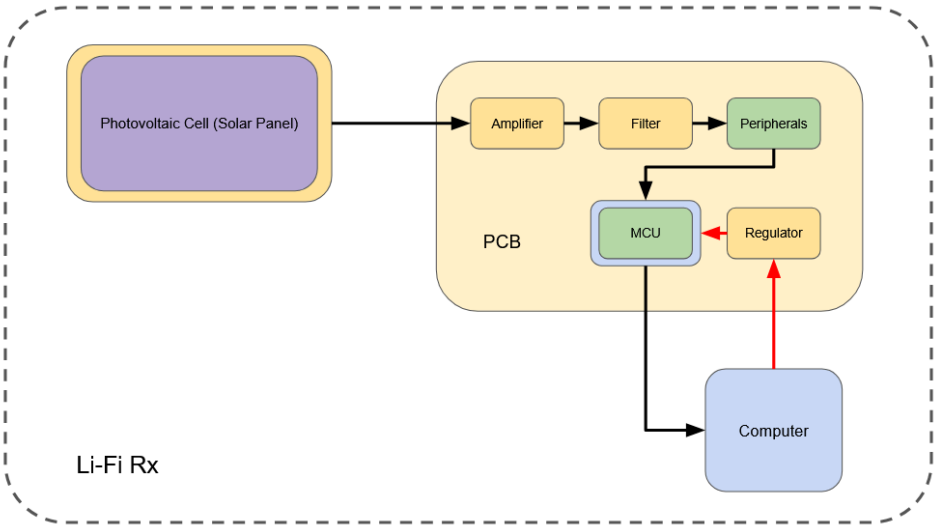


Figure 2.2: LiFi Transfer Diagram



At a high level, the system operates by detecting radio frequency (RF) emissions in the surrounding environment, classifying those signals based on machine learning inference, capturing imagery associated with the event, and securely transmitting this information via a focused optical beam to a remote receiver. The receiver then decodes and visualizes the transmitted data, making it available to users through a local or remote interface.

The core advantage of this architecture lies in its low probability of intercept, achieved through both passive signal acquisition and line-of-sight optical communication. By avoiding active RF transmissions, the system remains stealthy and suitable for operation in contested or surveillance-sensitive environments.

The transmitter subsystem constitutes the sensing and encoding half of the architecture and carries most of the system's functional complexity.

At the front of the TX unit is an omnidirectional or directional antenna designed to capture ambient RF emissions. This antenna is connected to a software-defined radio (SDR), such as an RTL-SDR or HackRF, capable of digitizing a wide spectrum of RF signals. The SDR feeds raw in-phase and quadrature (I/Q) data into an embedded system for signal analysis.

- *Antenna Selection:* Depending on the frequency bands of interest (e.g., 2.4 GHz ISM, LTE, Wi-Fi, drone telemetry), antennas with appropriate gain and bandwidth characteristics are used.
- *Preprocessing:* The SDR firmware handles basic filtering and gain control before passing samples to the host microcontroller or processor.

The central processing unit, likely a Raspberry Pi 4 or equivalent SBC, performs several tasks:

- Signal pre-processing (FFT, filtering, downsampling)
- Signal classification using a trained machine learning model (e.g., CNN or SVM classifier)
- Image capture via onboard or connected CMOS camera module
- Data packaging into a structured message (text file or JSON object)

The classifier attempts to identify the type of emitting device (e.g., Wi-Fi router, drone, Bluetooth beacon) based on spectral characteristics. Once classification is complete, the unit pairs this result with geolocation data obtained via GPS module or known fixed coordinates.

Once the classification and metadata are compiled, they are passed to a microcontroller (e.g., STM32 or Arduino Nano) responsible for modulating this digital data for optical transmission. The chosen modulation scheme is On-Off Keying (OOK), due to its simplicity and compatibility with LED/laser diode hardware.

- *Driver Circuit:* A transistor-based driver circuit ensures the LED or laser diode can switch rapidly enough for the desired data rate.
- *Collimation:* A lens system is used to shape and narrow the beam, reducing divergence and maintaining power density over long distances (hundreds of meters to 1 km).
- *Synchronization:* Basic start/stop framing bits are added to ensure the receiver can distinguish separate transmissions.

The RX unit is designed to be compact and low-power, with the primary objective of demodulating the incoming optical signal and displaying its content.

The receiver uses a high-sensitivity photodiode array coupled with an amplifier and low-pass filter. This circuit converts the light intensity variations into voltage levels that can be read by an ADC.

- *Optical Lens:* Similar to the transmitter, a focusing lens is used to gather as much incoming light as possible and direct it onto the photodiode.
- *Noise Suppression:* Shielding and spectral filters are used to reject ambient light or interference from other sources (e.g., sunlight, LED lights).

An MCU (e.g., ESP32) reads the analog signal, digitizes it, and applies a threshold-based demodulation algorithm to recover the original binary stream.

- *Synchronization:* Preamble sequences help in detecting the start of the transmission.
- *Data Integrity:* CRC or checksum validation ensures that the received message is not corrupted.
- *Display Module:* A simple OLED or e-ink display presents the text message (device type, signal time, classification confidence) to the user.

Both the TX and RX units are designed with debugging and feedback in mind.

- LEDs or indicators show system status (e.g., RF detected, message sent, error).
- Serial ports on each MCU provide logging for development and testing.
- Optional audio or vibration modules can be used for silent alerts in field conditions.

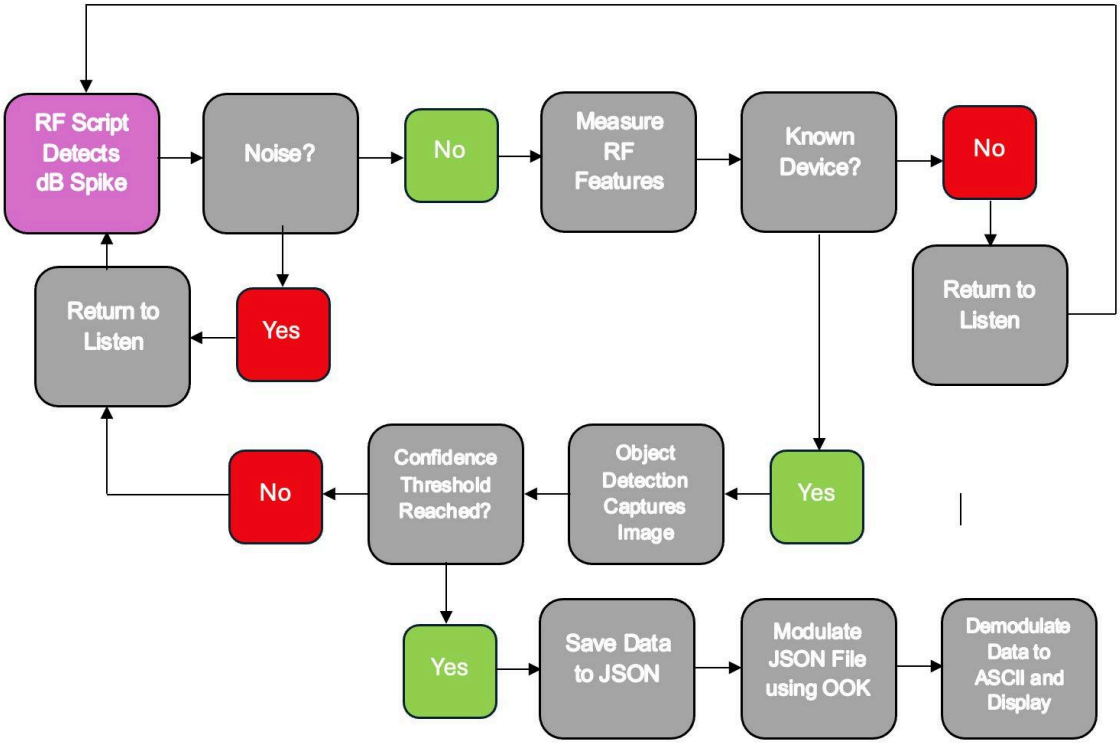
Given the remote nature of field deployment, careful attention is paid to power consumption and thermal management.

- *Battery Selection:* $LiFePO_4$ batteries are used for their ruggedness, long cycle life, and thermal stability.
- *Voltage Regulation:* Buck or boost converters maintain stable voltage rails for each subsystem.
- *Solar Integration:* The system supports solar trickle charging with MPPT-capable charge controllers, enabling indefinite deployment with sunlight exposure.
- *EMI Shielding:* Critical components such as the SDR and optics are enclosed in grounded metal casings to minimize crosstalk and external EMI.

2.7 Software Diagram/Flowchart

RF Receiver and Li-Fi Transmitter
In the diagram, the software flowchart showcases the series of checks that occurs to produce a text file based on collected data, prepared to be transmitted via FSOC (Li-Fi). First the device must check if noise was detected or if there was a significant change to indicate an RF device present. Then it will trigger an image capture and then record the spectral features to be used in the identification process. Once completed, a corresponding text file to be transmitted.

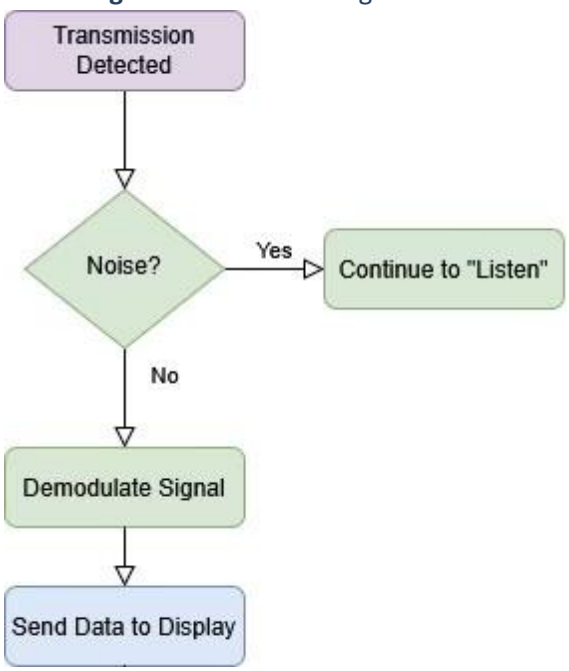
Figure 2.3: Software Process Flowchart



Li-Fi Receiver and WebApp Display

In the receiver, if a significant data bit is received, it can be determined to be a non-noise transmission. This will trigger the demodulation process to receive the text file depicting the identified device.

Figure 2.4: Receiver Logic Flowchart

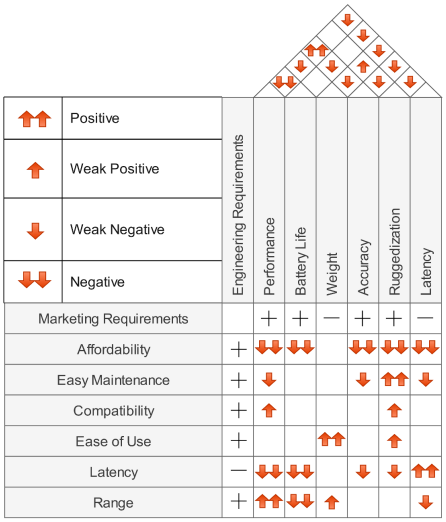


In combination, both the Free Space Optical Communication transmitter and receiver will provide the basis of the prototype. It is critical that the logic in the receiver and the transmitter both account for noise. There is a high chance that there will be many bit errors due to noise, therefore any logic we can add via programming to filter out the noise will increase the signal-to-noise ratio and ensure that the data is not compromised via transmission of free-space optical communication.

2.8 House of Quality

The House of Quality diagram showcases critical quality features and their corresponding polarity of success. Most notably, affordability has the most significant impact across performance, battery life, accuracy, ruggedization, and latency.

Figure 2.5: House of Quality Diagram



Our House of Quality diagram clearly depicts our emphasis on accuracy with affordability. There is a significant correlation between the price of our prototype and its effectiveness. The lower the cost and the higher the accuracy and effectiveness, the better the device. However, the House of Quality does also highlight range. In the ideal outcome of this prototype, we create a low-cost portable device that is accurate with a high Signal-to-Noise ratio and at a significant distance (aiming for 1km).

Chapter 3 - Research and Part Selection

This chapter describes the operational concept, functional design, and technical approach behind the proposed RF detection system. Drawing inspiration from passive radar principles, the system operates by listening to existing RF activity in the environment without emitting its own signals. By leveraging synchronized RF and optical data capture, the system performs passive identification of nearby devices and transmits the fused intelligence to a remote location using Li-Fi. The following sections detail the system architecture, component roles, data flow, and the integrated functionality required to detect, classify, and visualize RF threats in real time.

3.1 Software Defined Radios

To detect RF emissions in the field, we turned to SDRs due to their versatility and adaptability. SDRs act as digitizers of electromagnetic activity, enabling software-level tuning across a wide range of frequencies. In our project, they are tasked with capturing ambient RF energy and extracting In-phase and Quadrature (I/Q) data for classification.

3.1.1 SDRs vs Traditional Spectrum Analyzers

Spectrum analyzers offer precise and calibrated measurements ideal for lab environments but are ill-suited for deployment due to their size, weight, cost, and lack of software-level flexibility. SDRs, on the other hand, can:

- Scan multiple bands with dynamic bandwidth settings
- Support real-time signal processing through GNU Radio or custom Python scripts
- Interface easily with embedded systems (e.g., Raspberry Pi)
- Be deployed in mesh sensing networks

This makes SDRs more aligned with our mission objectives: small form factor, low power consumption, field deployability, and flexible software integration.

Several SDRs were evaluated, including the HackRF One, RTL-SDR, and USRP B200 mini. Key criteria included:

- Frequency coverage (we required at least 70 MHz to 6 GHz)
- USB interface for integration with our processing board
- Supported sampling rate (minimum 2 Msps)
- Onboard amplification and filtering capabilities

We selected [insert SDR model used], which offered a good balance between performance and cost for our prototype.

RF performance is dictated heavily by the front-end design. We chose a directional antenna (Yagi) and an omnidirectional antenna depending on our use case. The directional antenna was useful for long-range detection, while the omni allowed for localized signal awareness.

Filters were used to remove unwanted out-of-band signals, and low-noise amplifiers (LNAs) were added to improve sensitivity without significantly raising noise floor. Our front-end design also incorporated a programmable RF switch to enable band scanning.

Once a signal is detected, we trigger an image capture using a high-speed camera or Raspberry Pi camera module. This image helps contextualize the detection, potentially identifying a vehicle, drone, or person carrying the emitter.

Captured images are timestamped and stored in temporary memory alongside the RF data.

The SDR collects I/Q data, which is then processed through a lightweight machine learning pipeline. A convolutional neural network (CNN) or support vector machine (SVM) is used to classify signals based on their spectral and temporal features.

We extract features like:

- Peak frequency
- Bandwidth
- Modulation type
- Entropy
- Time-domain patterns

These features feed into the classifier. Model training was done offline using labeled RF datasets collected from multiple environments.

Each prediction is assigned a confidence score. If the confidence is below a defined threshold, the result is tagged as ambiguous and queued for further analysis or ignored to reduce false positives.

We use FSOC to transmit classified threat information to a secure backend server. Li-Fi was chosen due to:

- Its immunity to RF jamming
- Inherent line-of-sight encryption (light cannot pass through walls)
- Availability of narrow-beam collimation

The MCU (e.g., STM32 or Raspberry Pi) receives the classification output and applies On-Off Keying (OOK) modulation via an LED driver. The beam is directed at a photodiode receiver equipped with filters to eliminate ambient light interference.

We apply error correction and encryption (ChaCha20) before modulation. Bit timing varies slightly per message to introduce entropy, making interception and reconstruction more difficult.

The optical receiver is a photodiode array connected to a demodulation circuit. Once received, the signal is:

- Decrypted
- Checked for integrity using SHA-256 checksum
- Parsed and rendered in human-readable format on the web app

The receiver is optimized to function in daylight, using spectral and polarization filters to reject ambient interference.

Once received, data is sent to a backend running on a lightweight Flask or Node.js server. This backend:

- Stores the incoming classification results and images
- Displays it through a web interface
- Logs all user and device interactions

We use MongoDB for storage, Socket.IO for real-time updates, and JWT for authentication. The backend is also hardened against injection attacks and stores logs in append-only format for traceability.

Operational metrics include:

- **Latency**: Average 0.3–0.5 seconds from detection to visualization
- **Classification accuracy**: >85% under field conditions
- **Optical range**: ~15 meters indoor, ~8–10 meters outdoor (collimated)
- **RF range**: Dependent on power of emitter, estimated 100–300 meters

To enhance robustness:

- If FSOC fails, we use a WiFi fallback channel with TLS 1.3

- Power is supplied via solar + $LiFePO_4$ batteries, with >24h autonomy
- Logs are mirrored to onboard encrypted storage in case of network loss

Below is a chart comparing the achievement of our requirements by using an SDR versus a Spectrum Analyzer.

Table 3.1: SDR and Spectrum Analyzer Comparison Chart

Requirement	Spectrum Analyzer	Software Defined Radio
Extract Raw RF Data (I/Q)	Sometimes. Provides dBm data, some spectrum analyzers allow I/Q data extraction	Yes. Dedicated software for data processing and extraction
Ease-of-integration	No. Many have limited hardware with little to no software development kits to support software architecture integration	Yes. Dedicated software development kits for integration with different operating system architectures.
Low Cost	No. Typically range from $1000-$50,000	Yes. Typically range from $100-$1000

Portable Form Factor	Sometimes. Lightweight, portable Spectrum Analyzers can be found, however typically costs tens of thousands of dollars. Typical analyzers are bulky and meant for use in stationary laboratories.	Yes. SDRs typically are formed as a dongle that can be connected via USB to a computer.

3.1.2 Software Defined Radio Pros & Cons

To solve these drawbacks, RF engineers developed Software Defined Radios. SDRs quickly became a hobbyist dream, and a manufacturing favorite. A SDR centralizes most of the radio's functionality within software, allowing a single hardware platform to perform diverse tasks by simply changing the software configuration [14]. This inherent flexibility is a critical advantage, especially for our project in our use-case. For our portable system, SDRs offer several benefits:

- **Cost-Effectiveness:** Many SDRs leverage Commercial-Off-The-Shelf (COTS) components, significantly reducing their cost of goods sold compared to dedicated hardware spectrum analyzers. This aligns perfectly with our goal of a cost-effective solution. In fact, we will use a very similar approach in our development: integrating high-quality, low-cost COTS to reduce the form factor and COGS of our device.
- **Portability and Lightweight Design:** SDRs typically consist of a compact circuit board or small enclosure, making them portable and lightweight, directly addressing a key goal, a portable form factor. This is done by implementing sophisticated electrical board configurations to include the minimal key parameters: RF amplifier, analog to digital converter, filters, a signal digitizer, antenna(s) for transmitter or receiving, and a USB port for connection to software.
- **Flexibility and Adaptability:** The software-centric nature allows for easy updates to bandwidth range to cover, new feature integration for measuring new data, and adaptation to different modulation schemes without hardware modification. This is critical for keeping pace with an evolving RF environment.
- **Data Export and Processing:** SDRs are designed to stream raw or processsed RF data directly to a connected computer. By doing so, we can facilitate advanced analysis, data logging, and incorporate an identification processes. To do this, we can include potential AI/ML integration based on open-source DSP algorithms, if feasible. This direct data accessibility is vital for our “identification” objective.

Given these clear advantages, the focus for our RF collection system unequivocally shifts to the selection of an optimal SDR that can be seamlessly integrated into a portable payload.

3.2 SDR Part Selection for RF Collection

There are multiple SDR's available commercially, however as they range in significant price and also serve multiple functions, it is critical to take a handful of SDRs to compare to each other. It should be noted that the selected SDR will be based on cost and performance on bandwidths of interest for this prototype, however, the SDR in a final build would be modular and could be swapped out for a different SDR depending on the code that will extract the data.

3.2.1 RTL-SDR

- **Overview and Advantages**
  The RTL-SDR originated from repurposed DVB-T TV tuners, and today it's a widely adopted, ultra-low-cost software defined radio with a USB stick form factor. It supports a range of ~500 kHz to 1.75 GHz and is best suited for general signal monitoring, FM radio, ADS-B decoding, weather satellite reception, and educational applications. Its affordability (often under $30) makes it attractive for prototyping and experimentation in constrained budgets.
- **Software Ecosystem**
  The RTL-SDR has extensive open-source support, particularly in GNU Radio, SDR#, CubicSDR, and Universal Radio Hacker. Its compatibility with Linux and Windows environments makes it accessible for both hobbyist and academic use. Despite lacking TX capabilities, it is still a viable tool for passive RF surveillance and signal logging.
- **Limitations**
  It has no transmission capability, limited dynamic range, lacks onboard processing power, and its performance can be hindered by temperature instability or RF interference. While suitable for basic RF logging in the 400 MHz range (e.g., walkie-talkies), it struggles with reliable capture in the 2.4–5 GHz range due to insufficient tuning range.
- **Use in Our Context**
  While it's not the most robust choice for our dual-band RF detection mission, RTL-SDR may serve as a fallback or as part of a low-cost prototype, especially for logging low-band VHF/UHF signals.

3.2.2 HackRF One

- **Overview and Capabilities**
  The HackRF One is a half-duplex SDR that operates from 1 MHz to 6 GHz, covering all our target bands. It is USB-powered and widely regarded for its versatility and relatively compact size. It's capable of capturing wide instantaneous bandwidths (up to 20 MHz), which is useful for capturing frequency-hopping or broadband transmissions.
- **Software Integration and Firmware**
  HackRF One is well supported by GNU Radio, SDRangel, and MATLAB. Its open-source firmware allows researchers to implement custom DSP pipelines and signal conditioning features. It also supports firmware modifications for gain control, baseband filtering, and tuning stabilization.

- **Performance and Limitations**
  While more powerful than RTL-SDR, HackRF One lacks onboard FPGA acceleration or dual-channel reception. Its 8-bit ADC also limits its sensitivity and dynamic range, which affects performance in RF-dense environments. Nevertheless, its performance is typically acceptable for tactical RF detection when paired with a well-matched antenna.
- **Use in Our Context**
  HackRF One may serve as our primary platform for UAS detection in the 2.4–5 GHz WiFi range, especially in configurations where cost and space are balanced with reliability.

3.2.3 LimeSDR Mini

- **Overview and Features**
  The LimeSDR Mini operates in the 10 MHz – 3.5 GHz range and supports full-duplex communication. Its standout feature is a reconfigurable FPGA (Altera Cyclone) and onboard LMS7002M transceiver, enabling high-speed signal acquisition and processing. The v2.0 revision improves clock stability and FPGA performance, addressing limitations of the original board.
- **Signal Processing and FPGA Use**
  The onboard FPGA can be configured to perform signal classification, digital downconversion, and even FFT-based real-time spectral analysis, reducing burden on the host processor. This is beneficial in mobile deployments where compute resources are limited.
- **Streaming and Throughput**
  The LimeSDR Mini supports bandwidths up to 30.72 MHz and can stream to host systems via USB 3.0. Its full-duplex mode allows simultaneous monitoring of two frequency bands—ideal for monitoring control and telemetry channels in a UAS signal.
- **Use in Our Context**
  Its compact form, FPGA-based customizability, and full-duplex support make the LimeSDR Mini 2.0 a strong candidate for our modular RF sensing node. Its ability to handle moderate bandwidths aligns well with spectrum monitoring and waveform fingerprinting goals.

3.2.4 bladeRF 2.0 micro

- **Overview and Advanced Capabilities**
  The bladeRF 2.0 micro represents a higher-end SDR platform with robust capabilities. It features an Intel MAX10 FPGA and LMS7002M RFIC, supporting full-duplex 2x2 MIMO operation from 47 MHz to 6 GHz. It offers up to 61.44 MSPS sample rates and high-speed USB 3.0 or PCIe interfaces.
- **Advanced Use Cases**
  The bladeRF excels in demanding environments including real-time spectrum monitoring, digital demodulation, GSM/LTE testing, and passive radar. Its 12-bit ADC and 56 MHz instantaneous bandwidth make it ideal for wideband RF activity surveillance.
- **FPGA Programming and MIMO**
  Its programmable FPGA supports signal conditioning, custom modulation/demodulation chains, and real-time classification routines. MIMO capabilities allow beamforming, TDOA, or direction-of-arrival (DoA) estimation—technically enabling geolocation through multilateration.

- **Use in Our Context**
  If budget allows, bladeRF is a candidate for the master node in a distributed RF sensor network. It could serve as the high-precision RF processor in strategic deployments (e.g., perimeter defense or high-risk surveillance zones).

3.2.5 ADALM-Pluto SDR

- **Overview and Educational Value**
  Developed by Analog Devices, the ADALM-Pluto SDR is designed as a learning platform. It operates from 325 MHz to 3.8 GHz (extendable to ~6 GHz with firmware modification) and supports full-duplex operation via USB.
- **Software Compatibility**
  PlutoSDR integrates well with MATLAB/Simulink, GNU Radio, and SDRangel. Its Python-based PyADI-IIO interface makes it easy to script RF acquisition and signal processing tasks for educational or prototyping use.
- **Signal Quality and Sampling**
  It features a 12-bit ADC and ~20 MHz bandwidth, making it suitable for moderate-resolution spectrum capture. Its compact, rugged plastic casing and USB bus power make it portable and field-deployable in low-intensity scenarios.
- **Use in Our Context**
  PlutoSDR might not be ideal for our primary node due to limited bandwidth, but its form factor and cost make it suitable for sub-nodes, blind-spot monitoring, or quick prototype deployments in tactical field testing.

3.2.6 SDR Part Comparison

Based on the SDRs discussed (RTL-SDR, HackRF One, LimeSDR Mini 2.0, bladeRF 2.0 micro, ADALM-Pluto), there are a list of pros and cons that can be derived for each. However, based on the requirements previously listed, we can decide on an SDR to move forward with. We'll start with a clear, concise table to compare the aforementioned SDRs:

Table 3.2: SDR Part Comparison Table

Requirement	Frequency	Cost	Integration
RTL-SDR	500kHz-1.75Ghz	$20-$50	Widely supported, plenty of documentation.
HackRF One	1MHz-6GHz	$300-$400	Widely supported, plenty of documentation.
LimeSDR Mini	10MHz-3.5GHz	$500-$550	Not widely supported, limited documentation.

bladeRF 2.0 micro	47MHz-6GHz	$540-$860	Somewhat supported, documentation from manufacturers, however limited community documentation
ADALM-Pluto	325MHz-3.8GHz (70MHz-6GHz with augmentation)	~$230	Excellent documentation, great community resources that are open sourced.

3.2.7 Frequency Range

For this portable device to provide a functional use case, it must be able to detect handheld walkie-talkie frequency ranges. This range is typically from 30MHz-500MHz. Therefore, it is critical that the chosen SDR can support this range. Additionally, if UAS RF detection is ever feasible, the SDR that is chosen must support WiFi ranges (2.4GHz/5GHz). Based on this knowledge, while each SDR supports the lower frequency range, the HackRF One, the bladeRF 2.0 micro, and ADALM-Pluto are the only three SDRs that have the full frequency range.

3.2.8 Cost Effectiveness

For this portable device to provide a functional use case, we are aiming for a low-cost solution. Each SDR comes with a unique structure and trade-off depending on its costs. Overall, based on the research, as the SDR gets more expensive, the resolution of the RF data increases. However, we are looking for the minimal viable data extracted from the SDR as a proof-of-concept demonstration. Therefore, the most significant cost trade-off comes from the bits in the ADC. Based on the research, both the RTL-SDR and HackRF One use an 8-bit ADC meanwhile the LimeSDR Mini, the bladeRF micro, and ADALM-Pluto each use a 12-bit ADC.

In the ideal build of this RF detection device, it will be located in a congested area. In such an area, it is paramount that the SDR supports higher resolution data collection, therefore a 12-bit ADC is necessary. However, with the increased resolution comes the increased costs. Also, the more expensive the device, the less documentation there exists due to the natural cost barrier of a lack of accessibility.

3.2.9 Integration and Documentation

Building a functional RF detection device requires not only technical hardware selection but also a clear understanding of how to integrate the chosen SDR into a larger system that includes signal acquisition, classification, data transport, and visualization. For student engineering teams or emerging developers without prior exposure to RF environments or professional-grade spectrum analysis, this presents a significant challenge. It becomes imperative, then, to leverage community

resources, published documentation, and pre-existing integration workflows to accelerate development and mitigate common pitfalls.

Documentation plays a foundational role in the lifecycle of integrating any complex hardware system. For SDR platforms, which often rely on intricate signal chain configurations, real-time streaming, and firmware customization, documentation acts as both a reference and a tutorial. It provides insight into device initialization, driver installation, data acquisition protocols, and even antenna configuration strategies. Without documentation, each of these steps becomes a trial-and-error process that can slow progress and lead to unreliable implementations.

Furthermore, integration rarely ends at hardware configuration. It typically involves chaining together multiple software libraries, protocols, APIs, and occasionally, firmware-level changes. Well-documented devices provide consistent SDKs or APIs, describe expected inputs and outputs, and offer troubleshooting guidelines that are crucial when debugging RF reception problems or classification inconsistencies.

For academic or research-grade teams that lack access to full-time support engineers, comprehensive documentation serves as a form of virtual mentorship. It not only informs but also educates, bridging the knowledge gap that would otherwise make SDR integration unmanageable for newcomers.

Some of the most widely adopted SDRs benefit from a strong open-source community that regularly shares code, tutorials, example pipelines, and performance evaluations. This community-driven content includes GitHub repositories, YouTube walk-throughs, Stack Overflow discussions, technical blogs, and even academic papers that build on the SDR's toolchain. Community forums such as Reddit's r/RTLSDR or GNU Radio mailing lists have become informal support channels where developers can ask questions and share solutions.

This contrasts sharply with commercial SDRs that may have limited documentation outside of what the manufacturer publishes. When access to device-specific software, application examples, or driver updates is restricted or delayed, developers are forced to reverse-engineer solutions or look elsewhere. In time-sensitive development environments, such as capstone projects or rapid prototyping scenarios, this can critically affect project timelines.

Documentation depth and community support often correlate with the age and cost of the device. Older devices with a larger installed base tend to have more robust documentation ecosystems. Lower-cost SDRs like the RTL-SDR and HackRF One, despite offering less raw performance, provide a more accessible on-ramp due to this surplus of learning materials.

Device-Specific Observations

RTL-SDR

The RTL-SDR is often hailed as the most beginner-friendly SDR on the market. Despite its limited bandwidth and lack of transmission capabilities, its strength lies in its near-universal software support and the overwhelming volume of public documentation. Dozens of books, tutorials, and videos walk users through configuring the RTL-SDR in Windows, Linux, and embedded environments. Integration examples include projects for aircraft tracking (ADS-B), FM

broadcast decoding, and basic spectrum analysis. For educational teams, it serves as a near-perfect entry point into the world of RF signal acquisition and real-time visualization.

HackRF One
HackRF One maintains a similarly rich body of documentation, particularly because of its active developer, Great Scott Gadgets, and its open-source philosophy. Detailed guides on firmware flashing, loopback testing, signal generation, and advanced GNU Radio workflows make this SDR highly integrable. The HackRF is also popular in cybersecurity and wireless exploitation research, which has contributed to its community popularity. It is one of the few affordable SDRs that can function reliably in both transmit and receive modes, and this dual capability is well-documented in projects focused on replay attacks, RF jamming, and protocol fuzzing. For our project, HackRF One offers a practical balance of documentation and performance, particularly when it comes to live signal capture in the WiFi spectrum.

ADALM-Pluto SDR
Although initially targeted for educational use, the ADALM-Pluto SDR from Analog Devices is supported by an extensive set of learning modules. This makes it one of the few SDRs with formally published lab guides, practical signal examples, and instructor documentation. Analog Devices also maintains well-written Linux kernel drivers, Python libraries, and MATLAB/Simulink integration modules. Its documentation prioritizes accessibility for students and beginners, covering topics such as FM decoding, FFT visualization, and digital modulation schemes. This device is less commonly used in field deployments due to its bandwidth constraints but remains a very strong contender for early prototyping and indoor lab testing. In the absence of a strong community forum, its manufacturer has filled the gap with academic-grade content.

LimeSDR Mini
The LimeSDR Mini presents more complexity when it comes to documentation. While it has basic instructions for installation and device flashing, community-based documentation is sparse compared to the RTL-SDR or HackRF. Advanced users who wish to leverage the FPGA or reprogram its DSP pipeline will need to dive into low-level design documentation and LimeSuite, the device's proprietary software interface. Although the hardware is powerful and theoretically well-suited to spectrum analysis and classification, the steeper learning curve may delay development for new users. In addition, integration support in GNU Radio, MATLAB, and SDRangel is not as comprehensive or consistently maintained as that of older SDRs.

bladeRF 2.0 micro
The bladeRF 2.0 micro benefits from excellent manufacturer-published documentation, including technical specifications, driver installation instructions, and FPGA programming guides. However, it suffers from a lack of broader community content. Because the bladeRF line is significantly more expensive than entry-level devices, it has not achieved the same level of adoption among hobbyists or students. Therefore, there are fewer practical integration tutorials and third-party case studies. For teams with professional experience in RF design, this may not be a problem, but for a student-led initiative, the bladeRF may require a longer ramp-up period. Its advanced capabilities like MIMO and high-speed streaming are only valuable if the development team is capable of implementing and debugging such workflows.

For our system, the SDR serves as the first component in a multi-step pipeline. It is tasked with collecting raw RF data, preprocessing that data, and sending it to a classification algorithm before the information is encoded and transmitted via Li-Fi. A poorly documented SDR not only complicates the acquisition phase but also threatens to delay the entire system flow. If data formats are inconsistent, signal timing is unstable, or driver APIs are undocumented, the subsequent classification and display logic may become unreliable.

Devices with strong documentation tend to publish well-defined interfaces and provide software tools to debug timing jitter, data corruption, or packet loss. In contrast, under-documented platforms can make it difficult to even confirm whether signal acquisition is operating correctly. In time-sensitive applications, such as drone detection or tactical surveillance, this uncertainty can be operationally unacceptable.

To mitigate these challenges, it is recommended to standardize data output formats, such as IQ samples in float32 or int16 representations, and ensure compatibility with Python or C++ pipelines. SDRs with consistent documentation are more likely to offer such standards, reducing the risk of integration failure.

While most student projects are time-limited, real-world applications require maintainability over months or years. SDRs with a strong documentation foundation are more likely to receive firmware updates, security patches, and feature enhancements. They are also more likely to remain compatible with newer versions of supporting libraries, operating systems, and visualization tools.

The long-term success of our system, especially if it is to be fielded or handed off to another team, depends on selecting components that are not only high-performing but also sustainable. Community documentation ensures that future contributors can understand, replicate, and improve on our work. Conversely, selecting an obscure device with minimal documentation might deliver better performance initially but can ultimately lead to maintenance bottlenecks or outright obsolescence.

3.2.10 Chosen SDR Part: ADALM-Pluto SDR

Based on the table below, we can easily see that the **ADALM-Pluto SDR** offers the best results for the lowest costs.

Table 3.3: SDR Part Selection Table

Requirement	RTL-SDR	LimeSDR Mini	bladeRF micro	HackRF One	ADALM-Pluto SDR
Supports 70MHz-2.5GHz			X	X	X

Low Cost (<$500)	X			X	X
Widely Available Documentation	X			X	X
12-Bit ADC		X	X		X

The ADALM-Pluto SDR provides the best performance for the lowest cost. However, it should be noted that with this SDR, to achieve geolocation as our stretch goal, we will need to physically augment the device to support two receivers. Therefore, it is in our best interest to purchase two separate units, one to augment, and one to demonstrate our minimal viable prototype.

3.3 Antenna for RF Collection

To collect any data on the chosen SDR, we must also attach an apt antenna to receive this data. Though most antenna types are readily compatible with COTS SDRs, it is important to understand what frequency bands these antenna types collect. Also, it is critical to choose an antenna that is portable. For RF collection, most antenna types will suffice. However, for geolocation, we will implement a two-antenna phase array.

3.3.1 Antenna Part Selection for RF Collection

The primary objective of our system's RF collection is to detect and identify radio frequency emissions from walkie-talkies and potentially Unmanned Aircraft Systems (UAS), specifically their WiFi signals. This presents a multi-band challenge, as these devices operate across distinct frequency ranges. Walkie-talkies typically utilize Very High Frequency (30MHz-300MHz) and Ultra High Frequency (300MHz-520MHz) bands, while UAS often rely on WiFi at 2.4 GHz and 5 GHz. Therefore, the chosen antenna must be capable of efficiently receiving signals across this broad spectrum, while still being a "portable, lightweight, and cost-effective" solution.

3.3.2 Antenna Requirements and Considerations

In summary, the antenna chosen must be compatible with the SDR (most are) and provide the capabilities to cover the VHF, UHF, and WiFi frequency band ranges. It must also be lightweight, a relatively small size, and low-cost.

- **Compatibility:** Must be compatible with the output impedance (typically 50 ohms) of the selected HackRF One.
- **Frequency Coverage:** Must effectively cover the frequency bands for walkie-talkies (VHF/UHF) and UAS WiFi (2.4GHz/5GHz)

- **Size and Weight:** Critical to the "portable, lightweight" nature of the device. Large, weighty antennas are unacceptable.
- **Cost:** Must align with our "cost-effective" goal, utilizing COTS components where applicable.

Some other parameters to consider as a result of the necessity of a strong signal collection include gain and directionality,

- **Gain:** Sufficient gain across the target frequencies to reliably detect signals, especially from potentially distant or low-power emitters such as a walkie-talkie.
- **Omni-Directionality vs. Directionality:** For initial detection and broad situational awareness, an omni-directional pattern is often preferred. However, for precise localization and identification, a directional antenna might be considered, possibly requiring a dual-antenna setup or a steerable system. While our initial focus is on broad collection, because a stretch goal of ours is geolocation, a directional antenna is required.

3.3.3 Discone Antenna

A discone antenna is a flat, circular disk-shaped antenna that is laid above a cone-shaped body that has a coaxial cable through the middle. This design enables it to function as a wideband, omni-directional antenna, therefore it can efficiently receive signals across a wide range of frequencies from multiple directions. Due to this broad coverage, its common use cases include: general radio scanning, spectrum monitoring, and spectrum surveillance. It is best used to listen to diverse signals without needing to switch antennas. There are a variety of sizes for discone antennas. While they can be large for high reception in the lower frequency bands such as VHF, there are more compact versions that exist for higher frequency ranges. This makes them adaptable for different use-cases.

3.3.4 Log-Periodic Dipole Array Antenna

Log-periodic dipole array antennas (LPDA) hold a triangle shape with multiple horizontal lines attached to one vertical line. This shape creates operational use in the widebands while also being directional. This means that it can transmit and receive signals from a very large range of frequencies from whichever direction it is being pointed to. Therefore, its use cases typically revolve around long-distance communication, especially television. There are variations that can be found in this antenna that increase the overall distance between the horizontal lines. As a general understanding, the larger the size of the LPDA, the lower frequencies it can receive.

3.3.5 Monopole Antennas

A monopole antenna or better known as a whip antenna consists of a single rod or wire. This is arguably the most common and simple antenna. This design relies on a ground plane to work. The whip antenna is omni-directional and is generally narrowband. Therefore, the whip antenna is limited to a very specific range of frequencies that it can interact with. This type of antenna can be seen in simple radio communication systems.

3.3.6 Dipole Antennas

A dipole antenna has two rods that are of equal length and arranged in a straight line. Due to the duality of its rods, there is not a ground plane necessary for operation This antenna is omnidirectional and typically is narrowband. This type of antenna is commonly used in radio communication systems.

3.3.7 Antenna Part Comparison

Based on the Antennas discussed (Discone, Log-Periodic Dipole Array, Monopole, and Dipole), there are a list of pros and cons that can be derived for each. However, based on the requirements previously listed, we can decide on an antenna to move forward with. We'll start with a clear, concise table to compare the aforementioned antennas:

Table 3.4: Antenna Part Comparison Table

Requirement	Frequency Coverage	Size/Weight	Cost
Discone	excellent wideband coverage typically spanning from 50MHz to 1.3GHz; omni-directional	typically very large, but can be augmented to be portable. However, this limits its frequency	around $150 for an antenna to cover 50MHz to 1.3GHz
LPDA	typically spanning from the lower VHF frequencies up to the WiFi range depending on design	typically relatively small and compact	approximately $500
Monopole	narrow range of frequencies from low 3kHz all the way to 5GHz	typically proportional to the targeted frequency; the lower the frequency the larger the antenna size	between $2-$40
Dipole	narrow range of frequencies from low 3kHz all the way to 5GHz	typically proportional to the targeted frequency; the lower the frequency the larger the antenna size	between $2-$50

3.3.8 Chosen Antenna: Two Monopole Antennas

For a simple demonstration of RF device detection, there is a need to pick a specific frequency range to detect a specific object. For example, if we are to detect an RF device using the 2.4GHz WiFi frequency range, then we must pick an antenna that can collect a narrow band in that range. If we are to detect a handheld radio communication device such as a walkie-talkie, operating in the 465MHz range, then we must pick an antenna that can collect a narrow band in that range.

It is paramount that we collect large amounts of data in a narrow band rather than a large amount of data across a wide band. By doing so, we can increase our capabilities to accurately detect a specific RF device. We also decrease the amount of data that overall needs to be processed, decreasing time for detection and identification.

Additionally, if the stretch goal of this project is to be achieved, then we need a small form factor that can support a phase array setup. To maintain a low-cost solution, multiple antennas will need to be purchased to achieve this geolocation.

Therefore our chosen antenna is a monopole antenna. For a simple RF detection, two separate antennas should be chosen. Specifically, we will utilize a dual-band (2.4 GHz / 5.8 GHz) WiFi antenna as well as a VHF/UHF antenna. By using two antennas, we can cover the necessary frequency bands of interest.

- **UAS WiFi (2.4GHz/5.8GHz):** A compact, dual-band monopole antenna, commonly used for WiFi routers, provides an excellent gain and coverage for detecting drone uplink and downlink using WiFi. These are widely available as COTS components and are very cost-effective.
- **Walkie-Talkies (VHF/UHF):** A compact monopole whip antenna optimized for the 470MHz range that are commonly used to communicate with walkie-talkies on. T These are widely available as COTS components and are very cost-effective.

By leveraging two separate antennas to cover the necessary frequency bands, we decrease the cost needed to cover our frequency ranges and maintain a compact form factor for optimal portability. By using the ADALM-Pluto SDR, we are left with the option to either physically change the antenna for a simple demonstration or augment the SDR to support two receiving channels. By augmenting the SDR we can collect both ranges simultaneously.

However, unless another ADALM-Pluto SDR unit is purchased, augmented, and synchronized with our original SDR unit, it is not possible to support simultaneous frequency coverage of the different bands and support geolocation. For geolocation, both antennas must be the exact same to collect data based on time delay at a specific frequency range. Both setups are feasible

3.4 Battery Types Selection

A low-power sensing system deployed in the field must be paired with the right battery to sustain it. Here we compare Lithium Polymer (Li-Po), Lithium-Ion (Li-ion), Lithium Iron Phosphate

(LiFePO4), and other options, focusing on energy density, safety/ruggedness, and solar recharge suitability:

3.4.1 Lithium-Ion (Li-ion)

Li-ion is a broad term for batteries usually with a cobalt-based cathode. They are widely used in laptops, phones, etc.

3.4.2 Lithium Polymer (Li-Po)

Technically a type of Li-ion, Li-Po batteries use a polymer electrolyte and usually come in flat pouch cells.

3.4.3 Lithium Iron Phosphate (LiFePO4)

This lithium chemistry is known for exceptional cycle life and safety.

3.4.4 Other Options (NiMH, Lead Acid)

While we are focused on LiPo/Li-ion types, it's worth mentioning alternatives briefly. **Nickel-Metal Hydride (NiMH)** batteries are very rugged and safe (no risk of fire, tolerate overcharge trickle), with decent cycle life (500+ cycles). They have much lower energy density (~60–120 Wh/kg) – on par or lower than LiFePO4. They self-discharge faster (unless using low-self-discharge cells like Eneloop). NiMH could be viable for small sensors if one is extremely concerned about fire risk or if the environment is too hot for lithium since NiMH can handle heat better. They also can be recharged by solar simply (even a crude charger can trickle NiMH without immediate damage). However, NiMH packs tend to be heavier and larger, so it's not particularly feasible for this project.

Lead-Acid (SLA/AGM/Gel) batteries are the traditional choice for off-grid systems (solar lighting, remote telemetry). They are very rugged in terms of handling abuse and have predictable failure modes (they'll just lose capacity, not catch fire). But lead-acid has an energy density of only ~30–50 Wh/kg, making it particularly heavy. You could get a Li-ion pack with the same energy at under 0.5 kg. Thus, like NiMH, it does not seem particularly valuable for our optical passive radar system. Lead-acid also only lasts a few hundred cycles if deeply discharged regularly. On the plus side, lead-acid handles cold weather well (down to –20 °C, though capacity drops) and is easy to keep charged with solar (you can float charge it indefinitely). There are ruggedized lead-acid batteries (gel or AGM) in waterproof cases that could be used. But given the advancements in LiFePO4, many field deployments that used to use lead-acid are now switching to LiFePO4 for better energy density and cycle life without sacrificing safety.

Table 3.5: Battery Comparison Table

Battery Type	Energy Density	Field Use	Solar Recharging
Lithium-Ion	~150–250 Wh/kg	These are used in military or outdoor applications – often with protective circuitry and shock-resistant enclosures.	Charging needs to be carefully controlled. Can't be held at 100% charge for long periods, especially in heat.
Lithium-Polymer	~150–250 Wh/kg	Soft pouch is vulnerable to puncture or swelling; can catch fire if overcharged or shorted.	For solar use, Li-Po can be used, but it may prove beneficial to slightly undercharge to prolong lifespan.
$LiFePO_4$	90–120 Wh/kg	Chemically stable, they can be overcharged to some degree and can handle higher temperatures.	The charge profile matches well with typical solar setups, and it's more forgiving of being held at full charge
NiMH	~60–120 Wh/kg	Very rugged and safe.	They can be recharged by solar simply but are considerably larger and heavier.
Lead-Acid	~30–50 Wh/kg	Handles cold weather well.	Can be float charged indefinitely.

*Green: Chosen battery

In the table above, we have chosen the LiFePO4 battery. This is due to its chemical stability and compatibility to solar setups. This battery will meet our component specification defined earlier of 24 hours of battery life.

3.4.5 Chosen Battery: LiFePO4

An $LiFePO_4$ battery seems to be an ideal fit for our project, below we can outline each parameter and measure it against the constraints of our battery selection:

- **Longevity (Battery Life & Cycle Life)**

Table 3.6: LiFePO4 Longevity Table

Requirement	How $LiFePO_4$ Compares
Battery life: ~24 hours minimum.	$LiFePO_4$ cells can be sized to deliver 24+ hours easily, even accounting for peak usage (camera + RF + Li-Fi burst). Packs are scalable (3.2V, 6.4V, 12.8V, etc.).
System can sustain operations for extended periods of time.	Cycle life of 2000–5000+ full charge cycles — lasts years with daily solar recharging. Li-ion gives you ~500 cycles by comparison.
Minimize downtime, deploy unattended.	$LiFePO_4$ handles daily deep discharge and recharges with very little degradation. Ideal for remote autonomous systems.

- **Ease of Use (Integration & Charging)**

Table 3.7: *LiFePO4 Ease of Use Table*

Requirement	How $LiFePO_4$ Compares
Portable and Compact... Solar-compatible.	$LiFePO_4$ batteries support direct solar charging with MPPT or PWM charge controllers. Available in rugged, sealed 12V packs.

Environmental resilience.	Works well in extreme temps (–20 °C to 60C discharge; many models support cold charging too). Resistant to thermal runaway.
Safe field deployment.	Very low fire risk. No lithium-ion-style thermal explosions. Stable chemistry.
Modular sensor deployment.	Can power multiple components (MCU, camera, Li-Fi, RF frontend) from a single pack with a regulated 5V or 3.3V buck converter.

- **Performance & Power Stability**

$LiFePO_4$ has a very flat discharge curve, meaning:

- It holds a consistent voltage (around 3.2V per cell) for most of the cycle.
- No sudden voltage drop-off that could crash our dev board, camera, or MCU.
- Easy to design around — most regulators are stable at 3.3V.

In conclusion, LiFePO4 appears to be the best choice for our field-deployable, solar-powered, long-lifespan sensor network: it trades a bit of added weight for up to four times longer lifespan and superior safety (no fire risk). For maximum caution in remote unattended scenarios (where battery maintenance is difficult and failure could be catastrophic), LiFePO4's forgiveness is extremely valuable. Given the passive radar/optical sensor must run long durations, likely with solar recharging, LiFePO4 emerges as a compelling option due to its ability to handle daily cycling for years and its resilience in various conditions.

3.5 Battery Part Selection

After extensive research, we were able to decide on the Power Queen 12V 50Ah. This battery marks the very best blend of capacity (1280Wh), Max Power Output (640W), Storage Temperature (14-122 degrees F), and Weight for portability (11.57 lbs.). Most importantly, this battery will work well with our SDR and antenna.

3.6 Development Board Technology Comparison

In order to fulfill the needs of the project, we need processing units capable of interfacing with the peripherals and sensors of our Li-Fi transmitting device, or the main device, and the receiver which should amplify and demodulate the signal before it is passed to the web application. With the requirements of reliability, cost, power efficiency, DSP, real-time processing, computer vision, and machine learning, we determined that a development board offering greater processing power than a simple MCU was needed. We have compared the pros and cons of using an FPGA (Field Programmable Gate Array) or MPU (Microprocessor Unit) for the project, which we believe are the two best options.

3.6.1 MPU Technology

MPUs may be designed with additional specializations in mind such as DSP or incorporate hardware modules like Raspberry Pi's HATs (Hardware Attached on Top) that expand certain functions. MPUs tend to offer less computational power and have a narrower range of applications where they work well, but are low cost and consume less power. The additions to each MPU can make each individual board more powerful for a single part of a larger project, and these boards usually offer turn-key ease of use for their marketed specializations. Mainly this comes in the form of included libraries and any extra hardware already included on the board. For example, an MCU with bluetooth and Wi-Fi capabilities like ESP32 includes bluetooth libraries and can be used easily.

Easier programming is an advantage of MPUs, which can run Linux or Linux-based systems like Raspberry Pi OS. MPUs often come with extensive libraries and built-in drivers to make interfacing USB and other input data easier. In combination with programming using C++ or Python, the software design flow on MPUs is more streamlined and requires fewer considerations about the hardware's specifications while programming. The MPU can handle what type and locations of memory to use, its own internal cache policy, and any hardware management a computer's OS would normally handle.

3.6.2 FPGA Technology

FPGAs can be programmed for custom hardware acceleration, allowing the designer to create dataflow pipelines that have a fixed length, offering time determinism and avoiding the potential overhead of instruction decoding or the unpredictable delays of interrupts. The freedom of creating custom pipelines can be described as a MISD (Multiple Instruction Single Data Stream) architecture where multiple different operations are applied in parallel to the same data. While they aren't naturally suited to floating point operations like GPUs, making them less suited for many ML tasks, they can perform custom-length fixed-point operations using quantized values to achieve high accuracy. FPGAs also usually include DSP slices which are useful for filtering signals from a radio receiver and also for image processing from the camera. FPGAs can draw more power at high utilizations compared to many MPUs and much more than lightweight MCUs which is an important aspect of battery-powered embedded devices.

Many commercial FPGAs used in industry are heterogeneous, meaning they include a microprocessor referred to as Processing System (PS) alongside the FPGA component of the board, the Programmable Logic (PL).

- **Processing System**: May use a linux operating system so that the FPGA can shine in its role as a hardware accelerator while the integrated PS is used for its driver compatibility. The Processing System may use the same libraries that an MPU would for training and deploying ML models. The operating system may contain software stacks for USB host, HDMI, or ethernet connections.

- **Programmable Logic**: The FPGA fabric of the board, or the FPGA itself. Reconfigurable and can be made to implement any design, as long as it is user-defined and there are enough resources. For example, to build an ML model that makes use of the FPGA's Programmable Logic requires using an HDL (Hardware Description Language) or writing HLS code. Compared to GPUs, the use of quantized fixed-point numbers in an FPGA's PL is also much less power intensive and generates less heat, making it more suited for embedded applications while still having flexibility and high performance.

FPGAs are used commonly in defense and aerospace applications not only because their hardware reconfigurability can adapt to changing protocols, but also because many FPGAs are designed with wide temperature tolerances. Compared to a microprocessor, FPGA hardware is uniform and so damage to individual parts of the fabric can be resolved by reprogramming the device. The reprogramming simply uses different blocks to implement a logically equivalent circuit.

Table 3.8: MPU vs FPGA Comparison Table

	MPU	FPGA
Ease of development	Supported by pre-existing software architecture	Requires specific design for use with hardware.
Cost	Average: $10-$500	Average: $50-$1000
Power Usage	Low power consumption	High power consumption dependent on logic.
Weight	Low, small weight	Relatively low mass, but significant size.
Suitability for heavy computation	Limited by cores available	Optimized for parallelism
Versatility	Fixed, difficult to customize.	Fully reconfigurable
I/O compatibility	Better in some cases, usually less versatile for	May require implementation of interface

	custom connections	design in FPGA
Includes FPUs for AI Applications	Only for specialized MPUs, but may be connected through external module	No, but may be connected through external module
Includes DSP for RF signal and image processing	Only for specialized MPUs, but may be connected through external module	Yes, almost always
Real-time Control and Sensor Fusion	May perform well as long as complexity remains low	Well suited: Deterministic, customizable, high bandwidth

In the table above, we compare the MPU and FPGA, highlighting critical use cases relative to our prototype build. Based on these parameters, we can see that the FPGA is significantly more suited for our design compared to the MPU.

Meeting the project's requirements will mean having more than one development board, at minimum two. One board to handle control and processing in the main device and the other to receive the amplified Li-Fi RX signal from the photodetector and handle demodulation. The receiver-side development board doesn't require serious computation and so a simple MCU should be suitable. Inside the portable device itself, the other development board(s) must be computationally sufficient, cheap enough, and power efficient so that the requirements for battery life are met. Incorporating multiple development boards increases power consumption but would mean more resources and the modularity of the design would work better for one that uses MPUs. Each MPU chosen could be used for one aspect of the design, such as RF signal processing, image pre-processing, ML workloads, and control.

An alternative to multiple MPUs is to use an FPGA with an integrated microprocessor, which could manage control, be easier to program when the FPGA isn't needed, interface directly with the FPGA resources, and have more compatibility with certain sensor data through its Linux drivers and built-in USB interface that enables the PS to act as a USB host. For programming the PL, FPGAs can be programmed using HLS where a higher level language like C++ is used like a hardware description language. This C++ code can then be compiled into equivalent HDL modules which eases the time needed to design complex systems like CNN models. The development time between training and deploying a CNN in python using the pytorch library is still much lower than creating a model in C++, but HLS somewhat closes the gap.

Table 3.9: Dev Board Comparison Table

Dev Board	Category	Applicability	Development complexity	Power Usage	Cost
Raspberry Pi 4	MPU	I/O Control	CPU coding Large	5.1V 600mA idle	$35

			community		
ESP32	MCU	Control Low power	CPU coding Low-level	3.3V 40mA idle	$18.30
PYNQ-Z2	Heterogeneous FPGA	DSP Control ML	Both CPU and HDL/HLS	5.1V/12V option 1.5/2.5A	$167
Arty A7-100T	FPGA	DSP I/O ML	HDL/HLS	5.1V/12V option 1.5/3A	$299

In this table, possible options for MPU/MCU and FPGAs are compared. Applicability measures the range of applications the board has that are relevant to the project, such as DSP, ML, and computer vision. Power usage shows the recommended power supply voltage and the idle power consumption, which can be the expected usage most of the time the device is on. The PYNQ-Z2 can accept a micro-USB input that provides up to 1.5A or an external power jack input that recommends 12V and can provide 2.5A. The A7-100T is also powered via USB or power jack. The conclusion of this research between FPGAs and MPUs is that using an MPU is preferable when it has enough to properly meet the requirements, but the final implementation should expect multiple development boards, multiple MPUs and/or an FPGA or FPGA+MPU development board while balancing the advantages of multiple boards with power and cost.

3.6.3 Power Analysis of Development Boards

As a general rule, the power consumption of an FPGA is higher than an MCU or MPU. For example, Raspberry Pi Pico (low power MCU/MPU) has a maximum current draw of ~90 mA @ 5V, whereas the PYNQ-Z2 is powered by either micro-USB (Up to 1.5 A @ 5V) or through its power jack or Vin/GND pins (Up to 2.5A @ 12V). A project was able to deploy a face recognition ML model on the PYNQ-Z2 and measured power consumption at 2.4W, so the actual power varies significantly depending on the tasks and utilization. [12] Compared to a full embedded processor like the more advanced RPI-5 the power consumption may be similar for the same workloads.

3.6.4 Board Selection

This project covers a wide range of workloads, including: Computer vision, filtering, modulation and demodulation, machine learning, and image processing. Out of the options that were explored, we found that the most feasible solution is to use a Raspberry Pi 5 and PYNQ-Z2 FPGA+ARM board that will provide the flexibility and resources to reliably perform the required tasks with low latency and high throughput.

3.7 Image Processing Technology Comparison

Image processing refers to any processing done on the camera image data and also the spectrogram constructed from the I/Q features on the RF side.

This project has two major pipelines based on the two most important sensor systems: The Optical System and RF System. The cameras only take pictures immediately after a signal is detected, so from that point in time the RF signal still needs to be classified. On the other hand, the Optical system needs to perform any image pre-processing and then classification. Therefore, the sequential nature of the system with its two separate pipelines of different lengths creates more latency. This is because the pipelines must merge at the end when a final classification of the object is inferred from both the RF and camera data, so the bottleneck is likely to be in the image processing and neural network stages of the optical system pipeline. For improving performance, the image processing and any ML model used should prioritize speed. Meanwhile, the RF classification should be able to take its time without increasing the total latency.

The purpose of the image pre-processing is to extract features from the image, or to transform it into something that an ML model will be able to make better inferences from, mainly by removing noise just as in the RF signal processing. The techniques we looked into are edge detection, image resizing, grayscaling, SIFT, and YOLO (You Only Look Once) or other more standard CNN feature extraction + classification, which is looked into in the Machine Learning Technology Comparison section

3.7.1 Edge Detection

Edge Detection is a form of feature extraction that removes other details but extracts the shapes of objects in the image. It may be redundant when using a CNN or other feature extracting model that learns to detect edges and produce that data in its feature maps. However, edge detection may be used on a spectrogram to de-emphasize noise. If using an RNN instead, then most likely I/Q feature data or an FFT bin will be buffered into the RNN, so edge detection or any image processing methods won't apply to the RF pipeline.

3.7.2 Image Resizing

Image Resizing will be necessary to match the dimensions of the input channels of the CNN if a CNN/YOLO is used. Resizing reduces the image's detail and so the model used must balance its input channel dimensions for accuracy and throughput.

3.7.3 Grayscaling

Grayscaling reduces the level of detail and can be destructive towards some of the features of the image, but it also greatly reduces the memory and computational demands on the system's

resources. For a grayscale image, the size should be ⅓ of the RGB image. This means lower memory needs and also less operations.

3.7.4 Scale-Invariant Feature Transform

SIFT (Scale-Invariant Feature Transform) is a feature extraction algorithm that can detect key points in an image. The algorithm has four steps: [9] [10]

- **Scale-space Extrema Detection**: Uses a DOG (Difference of Gaussian) algorithm to detect blobs, or points of extrema that may be important.
- **Keypoint Localization**: The intensity of the extrema must be examined. If it is not enough, then it won't be examined further.
- **Orientation Assignment**: Each keypoint is given an orientation and magnitude to make it invariant to image rotation.
- **Keypoint Descriptor**: The pixels around a keypoint form the keypoint descriptor.

Table 3.10: SIFT Methodology Comparison Table

Method	Function	Added complexity and resources	Help with ML classification?
Edge Detection	Detects edges as feature, removes unnecessary detail	Low complexity and resources.	Potential redundancy with ML feature extraction
SIFT	Detect key points as features, compare to detect objects	Provides limited complexity and resources available.	Can fill the same function
Resizing	Fit image to more desirable dimensions	Provides very little and limited complexity and resource	Necessary
Grayscaling	Reduce level of unnecessary detail, saves space	Provides very little and limited complexity and resource	Removes features but may be good for lower compute, power, memory

3.8 Machine Learning Technology Comparison

For our RF-emitting device detection and classification, we need to use models that are capable of: 2D Image analysis received from the cameras in visible light and other frequencies such as IR, detection of signals while properly ignoring noise, classification of the RF signal based on its characteristics, and performing sensor fusion by making a final decision about classification based on the RF and image data. With these goals, we have identified the CNN, RNN, YOLO, and MLP architectures as options worth researching.
The requirements for implementation of any ML models are:

- The model must be lightweight enough for the processing units to accommodate it. This is important because of limited resources and also because higher resource utilization will mean greater power consumption. Typically (efficiently) high utilization would be a good outcome but for battery-powered embedded applications it should be minimized.

- It should be well-suited to its data. Some of the data may be in the form of I/Q RF data, RF spectrograms, individual FFT signal samples, camera images, or the vectorized outputs of another model. The size of the data and how temporal it is can have a large effect on the model's ability to extract features.

- It must adhere to the real-time requirements of the project. Models with excessive latency are not acceptable. From the project requirements, the final classification and all final data should be sent in less than 1.5s after the signal is received.

- The model chosen should output information that is useful to future processing through another model or set of layers, or to be sent out over Li-Fi. For two separate models that each process the optical system and radio system, there must be a third model to complete the sensor fusion by making a final inference.

The neural networks that may be used for this device can be split into two stages: feature extraction and classification. The first layers of the network must produce feature maps/vectors that can be used by the later stages or an attached MLP (multi-level perceptron) to find relationships between features and produce a vector of confidence levels that the model has for each classification type it was trained on.

3.8.1 Convolutional Neural Networks

A CNN (Convolutional Neural Network) extracts features from an image or a data set where its elements are related spatially. It does this through its convolutional layers, which create feature maps that detect patterns like shape and texture. Each convolution includes the surrounding pixels or elements to produce features, so the features extracted are aware of the data around them. Multiple filters which contain weighted values are used to extract separate features. CNNs and their derivatives are commonly used in computer vision to detect the locations of one or more objects in an image and classify them.

A traditional CNN can only see their data set holistically, classifying the image as a whole rather than detecting objects in the image. This is a limitation where multiple objects emitting RF signals that appear in the image can confuse the neural network and result in low confidence, inaccurate inferences, or at best correctly identifying a single object. We also looked at an expansion of the CNN model made for object detection, YOLO, which is capable of locating where objects are in an image and classifying each one.

Because a CNN works well with a spatially related data set, if this type of neural network is used for the RF signal classification then the best strategy is to construct a spectrogram or waveform that can be used as an input to the CNN. The spectrogram would have information about how the signal evolves over time encoded into the spatial relation of its pixels. A smaller CNN, RNN, or other model could also be used for individual FFT samples rather than FFT signals composed into a spectrogram. However, when using a spectrogram to decompose and look at multiple signals, an advanced CNN derivative like YOLO or a YOLO-adjacent model might be used to find patterns in the spectrogram to identify multiple RF-emitting devices nearby.

3.8.2 Recurrent Neural Networks

An RNN (Recurrent Neural Network) is less commonly used in computer vision applications, but more common for certain kinds of signal processing like RF. RNNs take a sequential input and their state is altered by previous inputs, leading to the network's ability to use past data when making current inferences. The added temporal dimension of an RNN may be appropriate for the RF signal processing which has a temporal context. Like any neural network, an RNN can perform feature extraction. By taking I/Q data from the SDR the step of constructing a spectrogram can be avoided. Instead, the I/Q data or FFT graphs may be sequentially read by the RNN. If FFT graphs are fed into the RNN, this could be an alternative to how a CNN would use a spectrogram to understand the temporal context of the RF signal. With the RNN, the previous input would be fed back into the model at that layer, so the changes made over time would be explicitly understood.

A limitation of the RNN model is that it is inherently sequential. A CNN can parallelize its convolution and pooling because it is essentially doing a kind of image processing. For a 2D CNN, each element in the 2D data set may actually be a pixel. For an RNN, parallelism is limited because old outputs of a layer are used as part of the next set of inputs. However for each individual sequence the matrix multiplication of the network can be parallelized, e.g. a single FFT graph sample as just one time step in the data set.

3.8.3 Multi-level Perceptrons

An MLP (Multi-level Perceptron) is capable of identifying non-linear relationships between features to make inferences. If the goal is a small model, then more image and RF signal processing can be done early like sobel edge detection, STFT, and other algorithms meant to extract features from the raw data. However, the simpler MLP model may struggle with

real-world applications where relations between nearby pixels of a spectrogram or camera image aren't captured.

3.8.4 You Only Look Once

YOLO [8] adds attention mechanisms, deeper layers, and regression-based localization to move from image classification to real-time object detection for image and video. YOLO can detect individual objects by predicting the dimensions of bounding boxes around an object. At low confidence levels, the object associated with the bounding box is ignored, crucial for preventing an incorrect inference from interfering with the RF signal prediction and therefore misclassifying an RF-emitting device. YOLO takes an RGB image and uses many convolutional layers for feature extraction as any CNN does. Compared to other object detection models like R-CNN, YOLO is low latency and there are many variants like the tinyYOLO architectures made for embedded systems.

Table 3.11: Machine Learning Technology Comparison Table

Type	Feature Extraction	Example model depth	Training Complexity
CNN	Yes - spatial	LeNet5: 3 convolutional layers	Requires image classification
YOLO	Yes - spatial	YOLO: 20 convolutional layers	Requires object detection
RNN	Yes - temporal	Arbitrary feedback depth	Feedback adds complexity
MLP	No	2-3 fully connected layers	Depends on how features are extracted

The rating for each quality in the figure is vague because each of these models are scalable. They don't have a set amount of layers, throughput, latency, difficult training and implementing, or resources required. Resource cost is mostly a function of depth and how large the model's goals are. Latency is generally determined by the number of layers and size of a model. For meeting our latency requirements, we should attempt to minimize the model depth and overall complexity.

The type of neural network we use for processing the camera images needs to perform an early sensor fusion of the visible light and IR images. We should achieve this without running too many parallel networks. Using YOLO or another CNN, we can run separate convolutional pipelines and

concatenate the extracted features before moving to the flattening and fully connected layers where classification can take place. This is referred to as a multi-modal CNN because of the different input modalities (e.g. visible light and IR), or a multi-branch CNN due to the parallel convolutional and pooling pipelines for feature extraction of the different inputs.

Depending on the level of complexity of the inputs and accuracy required (which is >95% for our requirements), the challenge of training the model increases. If the basic requirement is just that the final design should detect a single RF-emitting device using images and RF data, then it may be best to work towards an implementation that accurately meets this objective. When using an FPGA's programmable logic, the model may be trained offline on a processor and the weights exported to the FPGA after which is quicker and easier than carrying out the training using the PL itself. Use of classical computer vision algorithms to help with feature extraction and object detection may be alternatives or allow for smaller ML models.

The final set of machine learning models used in the design for RF signal classification and object detection and classification should use YOLO or some form of modified YOLO/CNN to accurately differentiate between multiple objects in a picture, and especially to be able to find and detect a known type of RF-emitting device that may only take up a small part of the overall picture. Use of the same kind of model for the image analysis (camera) and RF signal analysis (spectrogram) would reduce development time and allow for sharing resources, so a kind of advanced CNN or modified YOLO seems preferable. When considering training, it may be best to use a simple model for camera images that classifies the image as a whole and another CNN model to classify the spectrogram as a whole.

3.9 Geolocation Technology Comparison

Our design's goals for geolocation in order to meet the project requirements are:

- **Low power consumption:** The added hardware should not greatly increase the total power consumption of the device.
- **Accurate:** The geolocation data must be meaningful and the web app should be able to show the device location
- **Quiet:** The geolocation technology should avoid using Wi-Fi or any active radio communications that would compromise the security of the device. Once the location data has been obtained, it can be sent securely over Li-Fi.
- **Portable:** The hardware added should not be too large or heavy.

For these goals a GPS module will be used to passively ascertain location data. GPS modules may contain small internal patch antennas, removing the need for a large external antenna. The GPS module will be selected for accuracy, size, weight, and ultra-low power.

Table 3.12: Geolocation Technology Comparison Table

GPS Module	Dimensions	Plug and Play w/ Linux?	Power consumption	Cost
Adafruit Ultimate GPS	25.5x35x6.5mm3	No	20 mA 5V	$29.95
GlobalSat BU-353S4	59x47x21mm3	Yes	60 mA 5V	$47.99
Beitian BN-880	28x28x10mm3	No	50 mA 5V	$33.98
WWZMDiB VK-172 USB GPS Dongle	Small USB device	Yes	Unknown current 5V	$9.99

The GPS module chosen is the Adafruit Ultimate GPS because of its HAT integration with our RPI-5 which should handle peripheral data transfers over Li-Fi that don't require extensive computational resources, low size and weight, and manageable cost and power consumption.

3.10 Embedded Communications Technology Comparison

The communications protocols we identified are:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I2C)
- Universal Asynchronous Receiver/Transmitter (UART)
- Ethernet
- High Definition Multimedia Interface (HDMI)
- Universal Serial Bus (USB)
- General Purpose Input/Output (GPIO)

These protocols are analyzed for latency, bandwidth, power consumption, if they are simplex or duplex, whether they are single host or flexible, and implementation complexity. Single host interfaces like SPI or USB must initiate communication from a single device, while flexible interfaces like I2C can have multiple devices initiate communications. Implementation

complexity describes how easily the protocols can be used and how much extra software or hardware is required to use the protocol. For example, USB is an interface that requires a large software stack and host controller, increasing the effort of implementation.

Table 3.13: Embedded Communications Technology Comparison

Interface	Bandwidth	Power Output	Simplex or Duplex	Flexible host?	Complex?
USB 2.0	Up to 480 Mbps	5V Up to 0.5A	Half duplex	Only with OTG	Yes
SPI	Based on SCLK, up to 60 Mbps	~0.46 mA and 3.3V idle	Full duplex	No, one master	User-defined
I2C	Based on SCLK, up to 3.4 Mbps	~0.27 mA and 3.3V idle	Half duplex	Yes	User-defined
UART	RS-232: 20 Kbps RS-422: 10 Mbps	2.3mA and 7V for typical RS-232	Half duplex or Full duplex	No	User-defined
Ethernet 10/100/1000	Up to 1 Gbps	0.45W to 1W for Gb Ethernet	Full duplex	Yes	Yes
HDMI	Up to 18 Gbps	Provides 5V and 50mA	Intent is uni-directional	No	Yes
GPIO	Based on system CLK, 100's of Kbps	Up to 40ma for ESP32	Simplex or Half duplex	No, generally	User-defined

Some protocols may be much more complex but less user-defined, such as USB in comparison to UART. Implementation complexity in these cases usually is much lower for the user-defined protocol but a development board with USB, HDMI, ethernet, etc. interfaces already well-implemented can make this easier. For interfacing with multiple sensors and peripherals SPI and I2C offer advantages such as simpler implementation, enough bandwidth, and low power consumption. However, USB connections may be even easier if the development board caters to them and they may offer higher bandwidth for sensor data.

GPIO and UART are often used for control signals due to their low bandwidth requirements, very little power usage, and simplicity. HDMI and Ethernet have applications when transferring video or large amounts of data but take up more power.

For our project the cameras only take images when an RF signal is detected, so a continuous stream of video data doesn't need to be transferred. Finally, USB, SPI, and I2C each have their own uses for connection to peripheral sensors and intra-device connections between any development boards used in the device. For a 1080p picture with RGB color data, the size should be 1920x1080x24 bits, or 5.93 MB. This would take 500 ms to transfer with a bandwidth of 100 Mbps. This leaves USB (power efficient option) and ethernet/HDMI (Gbps+ high bandwidth option).

3.11 Optics Technology Comparison

A major part of our project is achieving free space optical communication. Free space optical communication (FSO) is a technology that transmits data wirelessly through the air instead of using fiber optics or electrical cables. FSO uses modulated light beams to carry information from one point to another. There are many advantages to using FSO in our project. High bandwidth ranges that are comparable to optical fiber speeds are achievable without having to worry about cleaving and splicing glass fibers. FSO is also secure as a narrow beam is more difficult to hack than WiFi. There are multiple ways and different approaches on how to accomplish this way of communication.

3.11.1 LiFi Transmitter Part Selection

LiFi, an abbreviation for light fidelity, is a wireless communication technology that uses visible and infrared light waves to transmit data. How it essentially works is at first, data is encoded into the light emitted by an LED using modulation, such as on-off keying. A photodetector or photodiode is the receiver and absorbs the modulated light. We then have a signal processing unit that decodes the data that we can use to determine what is creating the RF emissions. Components needed in order to create a LiFi system include, LED transmitter, modulator circuit, photodiode receiver, signal processor, and power supply.

When it comes to the technology comparison between using an LED or laser diode, for the purposes of trying to achieve Li-Fi, we as a group have decided to go the route of utilizing a LED. Advantages of the LED include lower manufacturing and operating cost, longer operational life due to less heat and lower power consumption, and less sensitivity to temperature changes.

3.11.2 Light Emitting Diode (LED)

In deciding for the best LED to transmit our data via LiFi, it is critical to take certain characteristics into consideration.

First, and largely the most important characteristic is the wavelength at which the LED emits. Depending on the wavelength, our device's receiver could experience significant noise depending

on the environment. Other parameters include the bandwidth of emission and if it's typically used in LiFi related projects/suitable for modulation.

3.11.3 High-Speed Visible Light Emitting Diodes

Table 3.14: Embedded Communications Technology Comparison Table

Manufacturer	Part Number	Wavelength	Bandwidth	Notes
Osram	OSTAR High Power LED (LD CQARx series)	620-660 nm	~20 MHz	High-brightness red LEDs
Nichia	NVSW219C Series	4500K white	up to ~10 MHz (modulated)	Generally used for LiFi experiments
Seoul Semiconductor	WICOP LEDs (Z5M4, Z5M2)	Various	up to ~15 MHz	Suitable for data modulation
Lumileds	LUXEON	420-670 nm	10-15 MHz	Compact footprint, high radiant flux

These are used in low-to-mid-speed LiFi systems, such as data acquisition and environments with indoor lighting. We are ultimately going to utilize infrared LED instead.

3.11.4 Near-Infrared and Infrared Light Emitting Diodes

Table 3.15: Infrared Light Emitting Diode Comparison Table

Manufacturer	Part Number	Wavelength	Bandwidth	Notes
Thorlabs	M1550L4	1550 nm	46 MHz	1000 mA and has a built-in heatsink
Vishay	VSMY2940GX01	940 nm	~30 MHz	High-speed NIR emitter, low capacitance
Osram	SFH 4770S	850 nm	~25-30 MHz	SMD IR LED, compact and efficient

Excelitas	OP295	890 nm	~25 MHz	TO-18 can, legacy optical comm. LED
Marktech Optoelectronics	MTE880N	880 nm	~20-25 MHz	Custom high-speed IR emitters

IR LEDs allow data transmission without visible light, ideal for secure or dark environments.

3.11.5 Ultra High-Speed Visible Light Emitting Diodes

Table 3.16: Ultra High-Speed Diode Comparison

Manufacturer	Part Number	Wavelength	Bandwidth	Notes
Osram	GaN-based µLED arrays	Blue (~450 nm)	100s of MHz to GHz	Used in experimental high-speed LiFi systems
Pleesey/Quantalux	MicroLEDs	405-460 nm	~500 MHz+	Experimental µLEDs with ultra-low carrier lifetimes
Custom/Research	Laser-LED Hybrids	600-850 nm	1-3 GHz	Combines LED simplicity with laser-like speed

3.12 LiFi Receiving Part Selection

When deciding on a corresponding receiver, it is paramount that it detects the same wavelength as the transmitting LED. If they are not the same wavelength upon reception, then the data will not be able to be obtained. The receiving side of the LiFi communication system should be optimized to have a large active area. The larger the area, the chances of reception increase.

3.12.1 Photodiodes

There are multiple devices that can receive light, however photodiodes are optimal for receiving signals from specific wavelengths. These devices are designed to convert light energy into electrical energy. Photodiodes function based on the principle of the photoelectric effect, providing a measurable way to test its effectiveness. Photodiodes are commonly used in communication systems between devices.

3.12.2 Photodetectors for Visible LiFi (400-700 nm)

Table 3.17: Photodetector for Visible LiFi Comparison Table

Manufacturer	Part Number	Bandwidth	Active Area	Notes
Hamamatsu	S5971	~100 MHz	1 mm^2	High-speed Si PIN, ideal for visible LiFi
First Sensor/TE	PS100-6-CER	~150 MHz	0.8 mm^2	Ceramic package, high-speed, low capacitance
Thorlabs	FDS015	~150 MHz	0.3 mm^2	High-speed Si photodiode, visible spectrum
Centronic	OSD15-5T	~100 MHz	0.78 mm^2	Visible + NIR response, good for broadband LiFi

Best paired with visible LEDs like Nichia, Lumileds, or Osram red/blue emitters.

3.12.3 Photodetectors for Near-Infrared LiFi (850-950 nm)

Table 3.18: Photodetector for Near-Infrared LiFi Comparison Table

Manufacturer	Part Number	Bandwidth	Active Area	Notes
Hamamatsu	G4176-03	~200 MHz	0.8 mm^2	Fast NIR Si IN diode, ideal for 850-940 nm LEDs

Osram	SFH 2704	~100 MHz	0.8 mm^2	IR PIN diode, pairs with SFH 4770s
Vishay	BPW34FAS	~30-40 MHz	1 mm^2	High-speed Si photodiode, visible spectrum
Excelitas	C30724 Series	~150-200 MHz	0.8 mm^2	TO-46 Si photodiodes for NIR LiFi
Thorlabs	FGA01FC	>1 GHz	0.3 mm^2	Fiber-coupled InGaAs for high-speed IR detection (if moving into >1000 nm range)

Best for pairing with IR LEDs (Osram SFH, Vishay VSMY, etc.) in non-visible LiFi setups.

3.12.4 Ultra High-Speed Detectors (Research/Prototypes)

Table 3.19: Ultra High-Speed Detectors Comparison Table

Manufacturer	Part Number	Bandwidth	Type	Notes
Thorlabs	DET36A2	~1 GHz	Si photodiode + amp	Integrated high-speed module (DC-1 GHz)
Hamamatsu	G4176-03 with amp	>200 MHz	Si PIN	For custom high-speed analog front-ends
Excelitas	C30902EH	1 GHz	Si Avalanche	High sensitivity + speed (APD) for weak LiFi signals

Menlo Systems	FDP Series	Up to 1.2 GHz	PIN/APD	Used in quantum comms and research-grade LiFi

These detectors are suited for experimental LiFi >100 Mbps to Gbps-level.

3.12.5 Photodiode Arrays

By having multiple photodiodes lined in an array, it becomes possible to increase the active area while also determining light patterns. With multiple photodiodes, each photodiode will independently collect information based on the received signal. We can compare the information collected to help determine some characteristics about the signal, a functionality that can be used for position sensing.

3.12.6 Visible/Near-IR (Si-based) Photodiode Arrays

Table 3.20: Visible Photodiode Array Comparison Table

Manufacturer	Model	Type	Array Size	Bandwidth	Notes
Hamamatsu	S4114-01	Si PIN	16-channel linear	~10 MHz	Good for parallel Lifi channel detection
Hamamatsu	S8558	Si PIN	16x1 linear array	~20 MHz	High uniformity, fast rise time
Hamamatsu	S11850	CMOS	1024x25 (2D)	~10 MHz	High-density image sensor (good for spatially modulated Lifi)

Centronic	LD20-2C	Si PIN	2x20 array	~10 MHz	Low crosstalk linear photodiode array

These are ideal for structured light detection, imaging LiFi signals, or angle-sensitive reception.

3.12.7 InGaAs Photodiode Arrays (for 850–1700 nm)

Table 3.21: InGaAs Photodiode Array Comparison Table

Manufacturer	Model	Wavelength Range	Array Size	Bandwidth	Notes
Hamamatsu	G122232-512	900-1700 nm	512 px linear	~10 MHz	InGaAs linear array, good for high-speed NIR
Sensors Unlimited/ Collins Aerospace	SU1024LE-1.7RT	900-1700 nm	1024x1 linear	~20 MHz	Industry standard NIR array with fast readout
Xenicx	Xeva 1.7-320	900-1700 nm	320x256 (2D)	Up to 60 Hz (frame rate)	2D IR detection, good for imaging LiFi signals
Teledyne Judson	J12 Series	850-1700 nm	Custom	~20 MHz	Available in custom geometries, fast InGaAs pixels

These are best for multi-channel IR LiFi, especially in smart environments or secure comms.

3.12.8 Avalanche Photodiode Arrays (APD Arrays)

Table 3.22: Avalanche Photodiode Array Comparison Table

Manufacturer	Model	Type	Channels	Notes
Hamamatsu	S8550	Si APD array	4-channel linear	Up to ~100 MHz
First Sensor/TE	Series 9 APD Arrays	Si APD	8-16 channels	Can be combined with fast TIAs
Radiant Optronics	ROAPD-8	Custom InGaAs APD	8 channels	For NIR APD array solutions

Used in advanced high-speed, high-sensitivity LiFi detection systems (e.g., Gbps-class).

3.13 Image Capture Technology Comparison

For the image capturing of RF devices, the utilization of a CMOS (complementary metal-oxide semiconductor) sensor camera, as well as an IR (infrared) camera will allow us to detect any emittance from such devices in conjunction with our antenna. Once we have the data collected from our CMOS and IR camera, the data will then be translated into code that can be used to determine exactly what kind of device we are detecting signals from.

3.13.1 High-Speed CMOS Cameras (Visible to NIR)

Table 3.23: High-Speed CMOS Camera Comparison Table

Manufacturer	Model	Sensor	Frame Rate	Resolution	Notes

Basler	acA640-100gm	Sony ICX618 (Global)	100 fps	659 × 494	NIR-optimized version available (acA640-100gmNIR)
FLIR (Teledyne)	Blackfly S BFS-U3-04S2M-CS	Sony IMX287 (Global)	291 fps	720 × 540	Very fast, global shutter, USB 3.1
Allied Vision	Mako G-030	CMOS (Global)	300 fps	640 × 480	GigE interface, compact and rugged
IDS Imaging	UI-3130CP Rev. 2	Sony IMX265	120 fps	1280 × 1024	USB 3.0, visible & NIR sensitive
Arducam	OV9281 Mono Camera	OmniVision OV9281	120 fps	1280 × 800	For Raspberry Pi & embedded, global shutter

These are ideal for capturing high frequency modulated light signals for LiFi (on/off keying, OOK, or spatial modulation).

3.13.2 Near-IR (NIR) Cameras (750–1000 nm)

Table 3.24: Near-IR Camera Comparison Table

Manufacturer	Model	Sensor Type	Frame Rate	Spectral Range	Notes
Allied Vision	Goldeye G-032	InGaAs	100 fps	900–1700 nm	High-quality NIR imaging, GigE interface
FLIR / Teledyne	Blackfly S BFS-U3-13Y3M-C	Sony IMX265 (Mono)	60 fps	~400–1000 nm	Sensitive in 850–940 nm, cost-effective
Basler	ace acA640-100gmNIR	CMOS NIR-Enhanced	100 fps	300–1000 nm	Compact, good for modulated light in NIR

IDS Imaging	UI-3240NIR	CMOS	60 fps	400–1000 nm	Designed for NIR, global shutter
Xenics	Bobcat 320	InGaAs	100 fps	900–1700 nm	Higher-end, scientific-grade NIR detection

These are for typical LiFi systems with IR LED transmitters at 850 or 940 nm.

3.13.3 SWIR Cameras (1000–1700 nm)

Table 3.25: SWIR Camera Comparison Table

Manufacturer	Model	Sensor Type	Frame Rate	Spectral Range	Notes
Xenics	Cheetah-640CL	InGaAs	1730 fps	900–1700 nm	Ultra-high-speed for research-level LiFi
Teledyne DALSA	Calibir GX	InGaAs	100–200 fps	1000–1600 nm	SWIR vision with low noise
Sensors Unlimited	Micro-SWIR 640CSX	InGaAs	30–120 fps	900–1700 nm	Compact OEM module, USB 3.0
Raptor Photonics	Owl 640 II	InGaAs	120 fps	900–1700 nm	Scientific-grade SWIR camera with low noise

These are used in defense, free-space comms, and secure IR LiFi applications.

3.14 Optical Lens System

Just as there are infinitesimal amounts of light, there are as many ways to design lenses. And as such, there are numerous options of lenses in the market that could be used for our project. Initially, it was not exactly obvious what kind of lens was best to use for our experiment.

There are also many calculations to consider when constructing the optical lens system. From research papers that have conducted experiments that require collimation of light, specifically from an LED, they have outlined the necessary equations that are needed to properly collimate the

light. Among those equations, we have to consider and calculate incident angles, divergence angles, refractive index, luminous intensity, optical power, power efficiency, and integration of the optical lens system with our overall system.

Figure 3.1: Analytic Collimating Lens Calculations

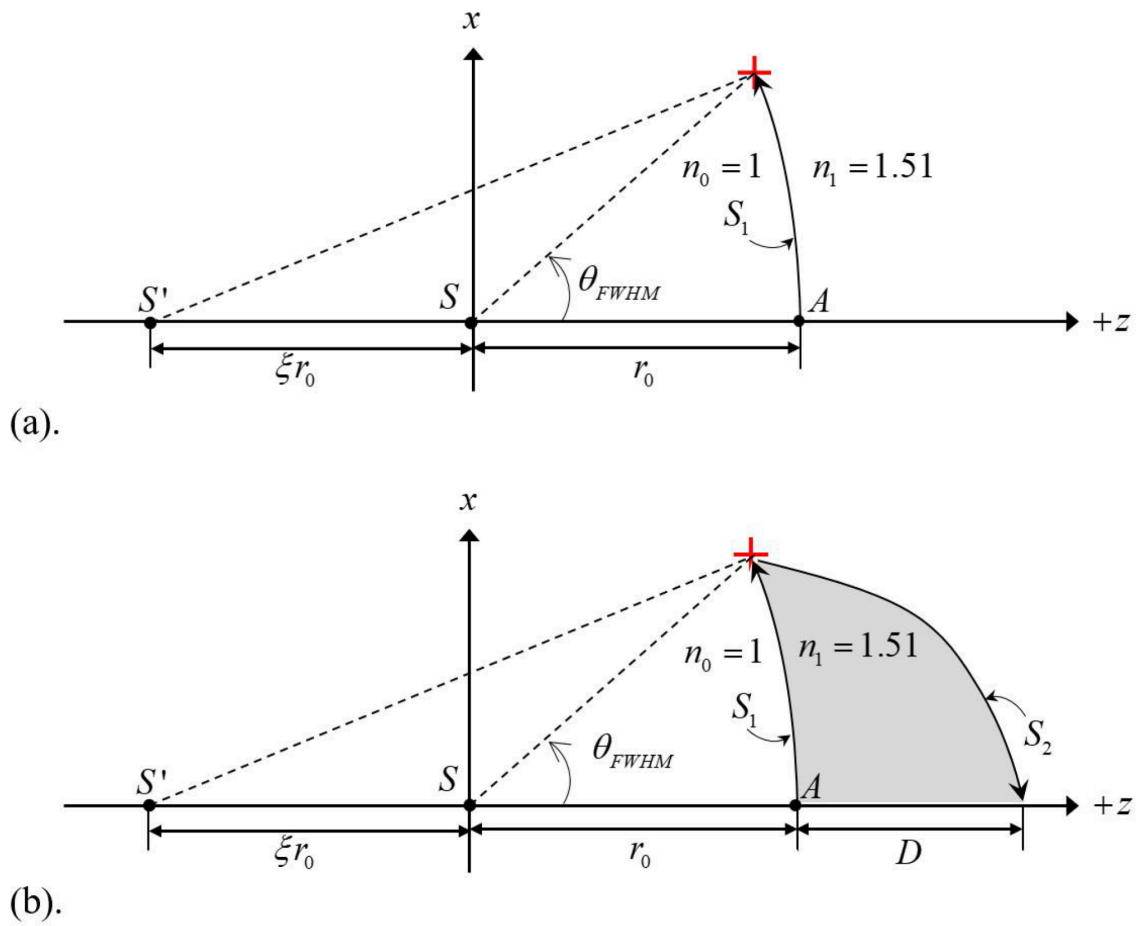


Figure: The analytic collimating lens is designed by: (a) selecting *r*0 and ξ to determine *S*1 starting from point A and ended with an aperture angle of *θFWHM*, (b) constructing *S*2 staring from the symbol + and ended at the distance of D relative to point A on the optical axis. [21]

The theory of designing an analytic collimating lens is schematically depicted in the figure above, in which the +*z* axis represents the principle optical axis of the LED light source with its center located at point *S*. As shown in the figure, the analytic-collimating lens contains two analog and non-spherical lens surfaces, *S*1and *S*2, which are both axially symmetrical with respect the optical axis of +*z* axis. The on axis distance between the centers of the first lens surface *S*1 and the LED light source is denoted as *r*0.

The working principle of this analytic collimating lens is to refract all light rays emitting from S by the lens surface $S1$ at the air/lens interface such that the transmitted rays in the lens material can be viewed as all radiating from a virtual light source located at S'. The distance between S and S' is $\xi r0$, in which the ξ is a non-dimensional parameter one can determine in designing the analytic collimating lens. Finally, the lens surface $S2$ is designed to refract all the light rays again at the lens/air interfaced in such a way after refraction and all light rays in the air are parallel to the optical axis. Both the surface profiles of $S1$ and $S2$ can be analytically or mathematically derived with given information on $r0$, ξ and optical refraction index of the lens material ($n1$).

Another parameter that needs to be chosen for finalizing the lens design is the full aperture angle of the lens. Essentially, we would assign the directivity angle ($\theta FWHM$) of the LED to be the full aperture lens angle in designing its corresponding collimating lens. By doing so we can make each lens collect approximately the same percentage of total LED's power, which forms the basis for fair comparison of the designed collimating lenses. Both lens surfaces ($S1$ and $S2$) can be analytically decided based on chosen parameters of $r0$and ξ, and D is the central thickness of this collimating lens which unit is mm.

As the collimating lens is axially symmetrical, the two lens profiles can rotate 360° around the optical axis ($+z$ axis) to form two lens surfaces indicated as $S1$ and $S2$ in the figure below. In this figure, the illuminating surface area of the LED under evaluation located at the original $x-y$ plane and a detector plane that collects all the light after collimating are shown.

Figure 3.2: Analyzation of Rays from LED Chip

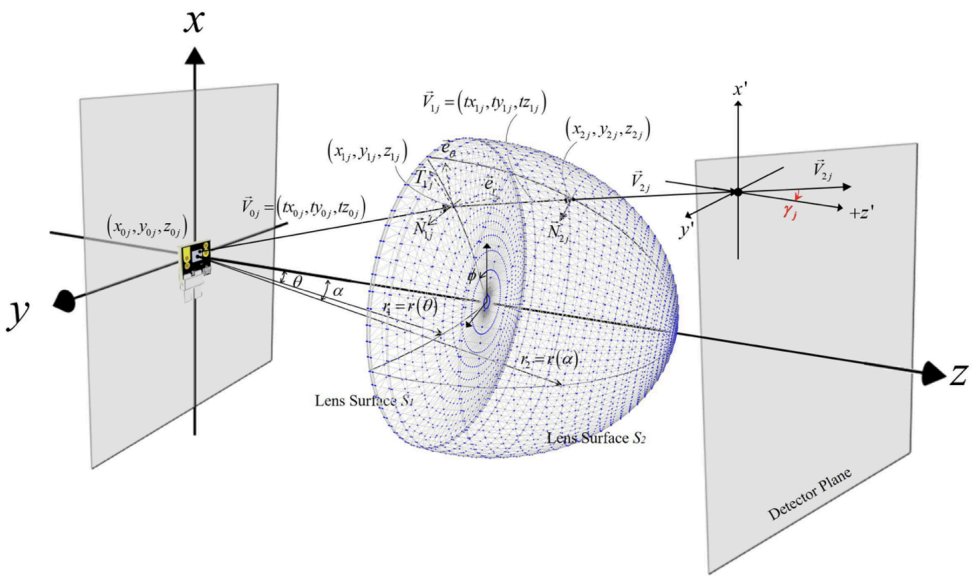


Figure: Analyzation of rays emitted from a LED chip and refracted by an example of a specifically designed light collimating lens. [21]

As previously mentioned, there are multiple formulas and equations that should be considered when designing light collimating lenses. As both lens surfaces had been analytically characterized, all the (*xij*,*yij*,*zij*) and *Vij* can be numerical solved using parameter root-finding algorithm such as Newton-Raphson method. Furthermore, considering the vector analysis of the spherical coordinate system and the axisymmetric curve, the unit normal vectors N_{1j} and N_{2j} of the *j*th incident ray intercepting surfaces S_1 and S_2, respectively, can be determined by,

Figure 3.3: Snell's Law

$$\overrightarrow{N_{1j}} = \frac{(-r_1 \cdot sin\theta + \dot{r_1} \cdot cos\theta) \cdot cos\phi}{\sqrt{\dot{r_1}^2 + r_1^2}} \cdot \hat{ex} + \frac{(-r_1 \cdot sin\theta + \dot{r_1} \cdot cos\theta) \cdot sin\phi}{\sqrt{\dot{r_1}^2 + r_1^2}} \cdot \hat{ey} - \frac{(-r_1 \cdot cos\theta + \dot{r_1} \cdot sin\theta)}{\sqrt{\dot{r_1}^2 + r_1^2}} \cdot \hat{ez}$$

$$\widehat{N1j} = \frac{\overrightarrow{N_{1j}}}{|\overrightarrow{N_{1j}}|}$$

$$\overrightarrow{N_{2j}} = \frac{(-r_2 \cdot \sin\alpha + \dot{r_2} \cdot \cos\alpha) \cdot cos\phi}{\sqrt{\dot{r_2}^2 + r_2^2}} \cdot \hat{e_x} + \frac{(-r_2 \cdot \sin\alpha + \dot{r_2} \cdot \cos\alpha) \cdot sin\phi}{\sqrt{\dot{r_2}^2 + r_2^2}} \cdot \hat{e_y} - \frac{(-r_2 \cdot \cos\alpha + \dot{r_2} \cdot \sin\alpha)}{\sqrt{\dot{r_2}^2 + r_2^2}} \cdot \hat{e_z}$$

$$\widehat{N2j} = \frac{\overrightarrow{N_{2j}}}{\left|\overrightarrow{N_{2i}}\right|}$$

in which, $r1(\theta)$ and $r2(\alpha)$ are the functions for characterizing surfaces $S1$ and $S2$, respectively. In the two-dimensional polar coordinate system, $r1^{\cdot}$ and $r2^{\cdot}$ are first derivatives with respect to their variables. The azimuth angle ϕ is referring to the angle between the $+x$ axis and the projection of the position vector of each interception point at the xy plane.

Based on the vector form of Snell's law, we can derive,

Figure 3.4: Snell's Law Derivation

$$\overrightarrow{V_{1j}} = \frac{n_0}{n_1}\left[\widehat{N1j} \times \left(\widehat{N1j} \times \overrightarrow{V_{0j}}\right)\right] - \widehat{N1j}\sqrt{1 - \left(\frac{n_0}{n_1}\right)^2 (\widehat{N1j} \times \overrightarrow{V_{0j}}) \cdot (\widehat{N1j} \times \overrightarrow{V_{0j}})}$$

$$\widehat{V1j} = \frac{\overrightarrow{V_{1j}}}{\left|\overrightarrow{V_{1i}}\right|}$$

$$\overrightarrow{V_{2j}} = \frac{n_1}{0}\left[\widehat{N2j} \times \left(\widehat{-N2j} \times \overrightarrow{V_{1j}}\right)\right] - \widehat{N2j}\sqrt{1 - \left(\frac{n_1}{n_0}\right)^2 (\widehat{N2j} \times \widehat{V1j}) \cdot (\widehat{N2j} \times \widehat{V1j})}$$

$$\widehat{V2j} = \frac{\overrightarrow{V_{2j}}}{\left|\overrightarrow{V_{2j}}\right|}$$

where $n0$ and $n1$ are the optical refraction indices of air and the lens material, respectively. Once the V_{2j} are calculated, the angle between the refracted jth ray and the $+z$-axis can be determined as another equation below, where γj is the angle between the refractive vector and optical axis.

Figure 3.5: Calculation of Angle Between Refracted jth ray and +z-axis

$$\gamma j = arccos[\widehat{V2_J} \cdot \widehat{ez}]$$

As each incident ray has been refracted twice by the two lens surfaces, the optical energy carried by each ray suffers two Fresnel's losses which can be readily calculated based on a standard formula and all data obtained earlier. Therefore, the remaining optical energy of the *j*th ray reaching the detector plane is calculated and denoted as *P*2*j*. Considering the optical power *P*2*j* as a weighting factor, we can define a weighted mean value of the divergent angles of all the refracted rays with the following equation,

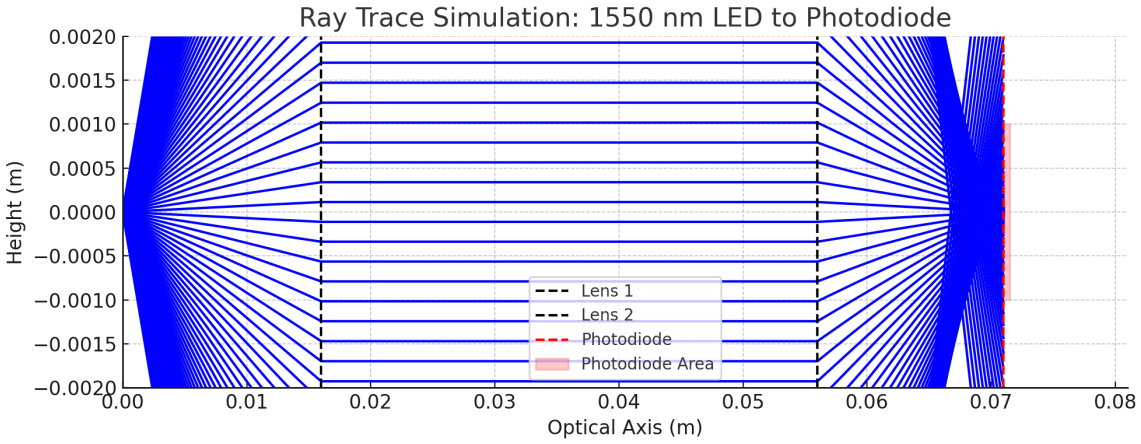
Figure 3.6: Calculation of weighted mean value of all divergent angles

$$\bar{\gamma} = \frac{\sum_{j=1}^{M}(\gamma_j \cdot P_{2j})}{\sum_{j=1}^{M}(P_{2j})}$$

where M is the total number of rays involved in the calculation. The γ represents the weighted mean divergence angle of the collimated LED light and therefore serves as an index for quantitative evaluation of the collimating effect. The lens design is aiming at achieving the smallest γ by adjusting all the lens design parameters.

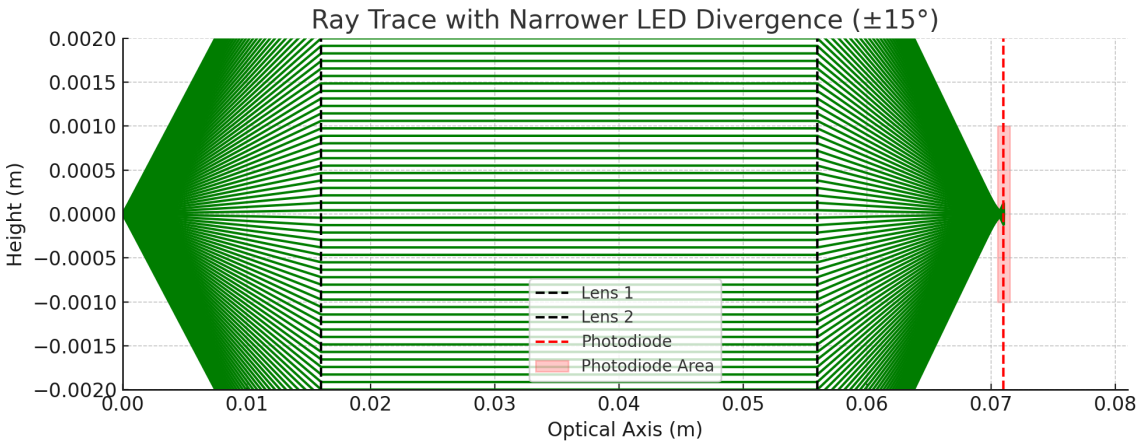
3.14.1 Optics Simulations

Figure 3.7: Ray Trace Simulation



The first simulation represents a distance of 16mm from LED to lens 1, 40 mm from lens 1 to lens 2, and 15mm from lens 2 to a photodiode. The spot size at the photodiode is 10.57 mm (full height of ray), which is much larger than a typical photodiode.

Figure 3.8: Ray Trace Simulation with Narrower LED Divergence



This second simulation is similar to the first simulation but instead now has a narrowed LED emission angle of ±15 degrees. As a result, the spot size is reduced to 0.24 mm (full width), which would now fit in a typical 2–3 mm² photodiode.

3.14.2 Aspheric Lenses for Collimating Light

Table 3.26: Aspheric Lens for Collimating Light Comparison Table

Vendor	Lens	Effective Focal Length	Numerical Aperture (NA)	Diameter	Wavelength
Thorlabs	C230TME-B	4.51 mm	0.55	Ø6.33 mm	600-1050 nm
Thorlabs	A390TM-A	3.1 mm	0.68	Ø6.33 mm	350-700 nm
Edmund Optics	#67-271	4.5 mm	0.55	Ø6 mm	600-1050 nm
Edmund Optics	#67-270	3.1 mm	0.68	Ø6 mm	400-600 nm

3.14.3 Aspheric Lenses for Focusing Light

Table 3.27: Aspheric Lens for Focusing Light Comparison Table

Vendor	Lens	Effective Focal Length	Numerical Aperture (NA)	Diameter	Wavelength Range
Thorlabs	C240TME-B	8 mm	0.5	Ø6.33 mm	600-1050 nm
Thorlabs	A397TM-A	6.24 mm	0.56	Ø6.33 mm	350-700 nm
Edmund Optics	#66-771	8 mm	0.5	Ø6 mm	600-1050 nm
Edmund Optics	#67-266	6 mm	0.56	Ø6 mm	Visible spectrum

Chapter 4 - Standards and Design Constraints

This chapter outlines the key engineering standards and design limitations associated with the core hardware components used in the system. Each component, LEDs, CMOS and IR cameras, and photodiodes, must comply with established safety regulations while also meeting specific electrical, thermal, and optical performance requirements. Understanding and addressing these constraints ensures not only user and device safety but also optimal system reliability and regulatory compliance across various operating environments.

4.1 LEDs (Light Emitting Diodes)

When using LEDs, there are critical safety standards that must be followed to avoid bodily injury. These standards also serve as good guidelines of what technology to avoid unless there are pre established safeguards to handle inherently unsafe conditions. Outside of the safety standards for humans, it is also important to keep the technology piece safe as well. There are design constraints such as overheating that must be considered. Therefore, we analyze both the safety standards and the design constraints for LEDs.

4.1.1 Safety Standards

LED technology is very energy efficient and can have long lifespans. However, like any electrical product, LEDs must adhere to stringent safety standards to protect users from potential hazards. These standards are primarily set by international and national organizations to ensure product safety throughout their lifecycle, from manufacturing to end-use.

1. Photobiological Safety (IEC 62471)

This is perhaps the most critical standard for LED lighting, addressing potential hazards to the eyes and skin from optical radiation. The International Electrotechnical Commission (IEC) 62471, "Photobiological Safety of Lamps and Lamp Systems," evaluates and classifies light sources based on their risk of causing damage from:

- Actinic UV: Harmful ultraviolet radiation that can cause skin and eye irritation.
- UVA Eye: Ultraviolet A radiation that can damage the eye.
- Retinal Blue Light: High-energy visible blue light (typically 400-500 nm) that can cause photochemical damage to the retina, potentially leading to macular degeneration with prolonged exposure. This is a significant concern with many white LEDs, which often use a blue LED with a phosphor coating to create white light, resulting in a prominent blue peak in their spectrum.
- Retinal Thermal: Damage to the retina due to excessive heat from intense light.
- Infrared Radiation (IR) Eye: Infrared radiation can cause thermal damage to the eye, especially as it doesn't trigger the natural aversion response to bright light.
- Thermal Skin: Damage to the skin from excessive heat.

Based on these risks, products are classified into four risk groups (RG):

- Exempt Group (RG0): No photobiological hazard under normal use.
- Risk Group 1 (Low-Risk): No hazard due to normal behavioral aversion to bright light.
- Risk Group 2 (Moderate-Risk): Poses a hazard if exposure is prolonged, but typically protected by aversion responses. Requires warning labels and minimum viewing distances if applicable.
- Risk Group 3 (High-Risk): High risk of acute harm even with short exposure. Generally not permissible for general lighting.
- Manufacturers must test their LED products to IEC 62471 and document compliance, often evidenced by certifications like CE, UL, or CCC.

2. Electrical Safety (UL, ETL, CE, etc.)

LEDs are electrical devices, and thus must comply with general electrical safety standards to prevent hazards such as electric shock, fire, and overheating. Key aspects covered include:

- Insulation and Enclosure: Ensuring adequate insulation to prevent electrical shock and robust enclosures to protect internal components from external elements (e.g., dust, water) and prevent users from touching live parts.
- Wiring and Connections: Proper wiring practices, secure connections, and appropriate wire gauges to handle the electrical current without overheating.
- Power Supplies/Drivers: LED drivers convert AC power to the DC power required by LEDs. These drivers must meet safety standards for voltage regulation, surge protection, and isolation.
- Overcurrent Protection: Incorporating fuses or circuit breakers to protect against overcurrents that could lead to overheating and fire.
- Grounding: Proper grounding to prevent electrical shock in case of a fault.

Common electrical safety certifications include:

- UL (Underwriters Laboratories): A widely recognized safety certification in North America. UL Listed, UL Classified, and UL Recognized marks indicate compliance with specific safety standards for various product types and uses.
- ETL (Electrical Testing Laboratories): A product safety certification offered by Intertek, which tests products to UL standards for North America.
- CE (Conformité Européenne): A mandatory conformity mark for products sold within the European Economic Area (EEA), indicating compliance with all relevant EU directives, including safety.
- CSA (Canadian Standards Association): The Canadian equivalent to UL, establishing safety standards for products in Canada.

3. Thermal Management

LEDs generate heat, and excessive operating temperatures can significantly reduce their lifespan and performance, and in extreme cases, pose a fire hazard. Safety standards address thermal management by:

- **Temperature Limits:** Specifying maximum allowable operating temperatures for LED components and the luminaire itself to prevent overheating.
- **Heat Sinking:** Requiring effective heat dissipation mechanisms (e.g., heat sinks) to manage the heat generated by the LEDs.
- **Thermal Protection:** Implementing thermal protection mechanisms that can dim or shut down the LED if temperatures exceed safe limits.
- **Testing under various conditions:** Lumens and operating temperatures are tested over extended periods (e.g., 6,000 hours as per LM-80) to ensure long-term stability and predict lumen maintenance.

4. Flicker and Stroboscopic Effect

LED lighting, especially with poorly designed drivers, can exhibit flicker (rapid variations in light output) and stroboscopic effects (making moving objects appear stationary or moving differently). While not directly a physical safety hazard, these effects can cause:

- Visual discomfort: Headaches, eye strain, and fatigue.
- Photosensitive epilepsy: In susceptible individuals, low-frequency flicker (15-20 Hz) can trigger seizures.
- Reduced performance: Impaired reading ability and concentration.

Standards like IEEE 1789 provide recommendations for flicker, linking acceptable modulation percentage to frequency. Generally, higher frequencies allow for greater modulation, with limits for perceptible and imperceptible flicker. California's Title 24, for example, requires LED products to have less than 30% flicker at frequencies below 200 Hz.

5. Electromagnetic Compatibility (EMC)
LED lighting products contain electronic components that can emit electromagnetic interference (EMI) or be susceptible to interference from other devices. EMC standards ensure that LEDs:

- Do not cause excessive interference: Their electromagnetic emissions are below specified limits to prevent disruption to radio, telecommunications, and other electronic equipment.
- Are immune to interference: They can function correctly when exposed to common electromagnetic disturbances in their environment.

Relevant EMC standards include:

- FCC Part 15 and FCC Part 18 (United States): Regulate electromagnetic emissions from electronic devices.
- EMC Directive (2014/30/EU) and EN standards (European Union): Ensure products meet essential electromagnetic compatibility requirements.
- Industry Canada (IC): Canadian equivalent to FCC.

6. Hazardous Location Standards (ATEX, IECEx, NEC)
For LEDs used in hazardous locations (e.g., areas with flammable gases, vapors, or dust), highly specialized safety standards are critical to prevent explosions. These standards dictate specific design and construction requirements, such as explosion-proof enclosures and intrinsic safety, to ensure the lighting system does not become an ignition source. Examples include:

- ATEX (Atmosphères Explosibles) Directive (EU): A mandatory directive for equipment intended for use in potentially explosive atmospheres.
- IECEx Scheme: An international certification system for equipment used in explosive atmospheres.
- NEC (National Electrical Code) (United States): Provides guidelines for the safe installation of electrical equipment, including lighting, in hazardous classified locations.

The safety of LEDs is a multifaceted aspect covered by a comprehensive set of international and national standards. These standards aim to protect users from various hazards, including photobiological damage, electrical shock, fire, and adverse health effects from flicker and electromagnetic interference, ensuring that LED technology is not only efficient but also safe for widespread use.

4.1.2 Design Constraints

1. Thermal Management

While LEDs are energy-efficient, a significant portion of the electrical power they consume (typically 60-95%) is converted into heat, not light. This heat is generated at the p-n junction of the LED chip. If not effectively dissipated, excessive heat leads to:

- **Reduced Lifetime (Lumen Depreciation):** High junction temperatures accelerate the degradation of the LED chip and its phosphor coating, leading to a faster decrease in light output over time. This is often quantified by L70 (time to 70% of initial lumen output) or L50.
- **Color Shift:** Overheating can cause the phosphor to degrade unevenly, leading to a noticeable change in the LED's correlated color temperature (CCT) and color rendering index (CRI). For instance, a "warm white" LED might shift towards a "cooler" or greener tint.
- **Reduced Efficiency:** Higher temperatures decrease the luminous efficacy (lumens per watt) of the LED, meaning it produces less light for the same amount of power.
- **Catastrophic Failure:** In extreme cases, sustained high temperatures can lead to irreversible damage to the LED, causing it to fail completely.
- **System Degradation:** Heat can also impact other components in the luminaire, such as drivers, PCBs, and optical materials, reducing their lifespan and leading to overall system failure.

Design Implications:

- **Heat Sink Design:** Requires robust heat sinks (often aluminum or copper) with sufficient surface area and proper fin geometry to dissipate heat via conduction, convection, and radiation.
- **Thermal Interface Materials (TIMs):** Use of thermal pastes, pads, or adhesives to ensure efficient heat transfer from the LED package to the heat sink.
- **PCB Material:** Selection of PCBs with good thermal conductivity (e.g., metal-core PCBs - MCPCBs) for high-power applications.
- **Enclosure Design:** Luminaire enclosures must allow for effective airflow or provide sufficient surface area for passive cooling, especially in sealed or recessed fixtures where ambient airflow is limited.
- **Ambient Temperature:** Operating environments with high ambient temperatures pose additional thermal challenges, requiring more aggressive thermal management solutions.

2. Electrical Design Constraints

LEDs are current-driven devices, meaning their brightness is directly proportional to the current flowing through them, not the voltage across them. This leads to several electrical design challenges:

- **Current Regulation:** LEDs require constant current to operate reliably and consistently. A slight change in voltage can lead to a significant change in current, potentially burning out the LED. This necessitates the use of dedicated LED drivers (constant current power supplies) that convert AC or DC input voltage into a regulated DC current for the LEDs.
- **Voltage Drop and Run Length:** For LED strips or long chains of LEDs, voltage drop along the circuit can cause brightness inconsistencies (dimming at the end of the strip). This limits the maximum practical run length for a single power feed, often requiring parallel connections or multiple power supplies.
- I**nrush Current:** When LED luminaires are switched on, especially those with capacitive power supplies, they can draw a very high inrush current for a brief period. This can trip circuit breakers or damage components if not accounted for in the circuit design and component selection.

- **Power Factor Correction (PFC):** LED drivers, particularly those used in commercial and industrial applications, need to incorporate Power Factor Correction (PFC) to ensure that the luminaire draws power efficiently from the electrical grid, reducing energy waste and meeting regulatory requirements.
- **Dimming Compatibility:** Achieving smooth and flicker-free dimming requires specialized LED drivers that are compatible with various dimming protocols (e.g., PWM, analog, DALI, 0-10V). Incompatibility can lead to flickering, limited dimming range, or even damage.
- **Electromagnetic Interference (EMI):** The high-frequency switching of LED drivers can generate electromagnetic interference (EMI), which can disrupt other electronic devices. Design must include proper filtering, shielding, and PCB layout to minimize EMI.
- **Surge Protection:** LEDs and their drivers are sensitive to voltage surges from lightning strikes or power grid fluctuations. Adequate surge protection devices (SPDs) are essential for outdoor and industrial applications.

3. Optical Design Constraints

LEDs are point sources of light, which presents unique challenges for achieving desired light distribution, uniformity, and aesthetic appeal:

- **Beam Shaping:** Unlike incandescent bulbs that emit light relatively omnidirectionally, LEDs typically have a narrow beam angle. To achieve specific beam patterns (e.g., flood, spot, diffuse), secondary optics (lenses, reflectors, diffusers) are essential.
- **Glare Control:** The high luminance of individual LED chips can cause significant glare. Optical design must incorporate diffusers, louvers, or recessed mounting to soften the light and reduce discomfort.
- **Color Uniformity (Color Binning):** Even LEDs of the same nominal CCT can have slight variations in color output due to manufacturing tolerances. "Color binning" is a process of grouping LEDs with similar color characteristics to ensure consistency across luminaires. This adds complexity and cost.
- **Light Extraction Efficiency:** Getting the maximum amount of light out of the LED package and into the desired beam requires optimized optical coupling and minimal losses from absorption or reflection within the luminaire.
- **Form Factor and Etendue:** In applications with strict size or volume constraints (e.g., downlights, automotive headlights), the "etendue" (a measure of how "spread out" a light source is) of the LED and its associated optics becomes a critical factor in achieving desired light output and beam control within a limited space.

4. Mechanical Design Constraints

The physical integration of LEDs into luminaires involves mechanical considerations:

- **Robustness and Durability:** LEDs themselves are relatively robust, but the complete luminaire needs to withstand environmental factors like vibration, impact, dust, and moisture (IP ratings are crucial for outdoor/industrial use).
- **Size and Weight:** While individual LEDs are small, the necessary heat sinks, drivers, and optics can add significant size and weight to the overall fixture, impacting mounting options and aesthetic integration.
- **Ease of Installation and Maintenance:** The design must allow for straightforward installation and, if necessary, access for maintenance or replacement of components (though LEDs' long lifespan reduces the frequency of replacements).

- **Material Selection:** Materials must be chosen not only for their mechanical properties but also for their thermal conductivity, optical transparency, and resistance to UV degradation (especially for outdoor applications).

5. Cost Constraints

Despite falling prices, the initial cost of LED luminaires can still be a significant constraint:

- **Component Cost:** High-quality LED chips, drivers, and thermal management solutions can be expensive.Balancing performance with cost is a constant challenge.
- **Manufacturing Complexity:** The precision required for thermal and optical integration can increase manufacturing costs.
- **Economies of Scale:** Smaller production runs may not benefit from the same cost efficiencies as mass-produced traditional lighting.

6. Environmental and Regulatory Constraints

Beyond safety standards, environmental factors and regulations influence LED design

- **Operating Temperature Range:** LEDs need to perform reliably across a specified temperature range. Designs for extreme hot or cold environments (e.g., industrial freezers, street lighting in deserts) require specialized components and thermal management.
- **Humidity and Water Ingress:** For outdoor or damp locations, luminaires must be sealed to prevent moisture ingress, which can damage electronic components.
- **Corrosive Environments:** In industrial settings, exposure to chemicals or corrosive gases requires specialized materials and protective coatings.
- **Recycling and Disposal:** End-of-life considerations, including the presence of rare earth elements (in phosphors) and electronic waste, are becoming increasingly important.

4.2 CMOS Cameras

CMOS stands for complementary metal oxide semiconductors. There are many different kinds of material that could be used to build a CMOS camera. For CMOS cameras, there are procedures to operate such equipment so that there is no damage to the prototype or user. Therefore, we analyze both the safety standards and the design constraints for CMOS cameras.

4.2.1 Safety Standards:

- **FCC Part 15 / CE Marking:** Electromagnetic compatibility and interference.
- **IEC 62368-1:** Safety requirements for AV and ICT equipment.
- EN 61000 Series (EMC immunity and emissions).
- **RoHS & REACH:** Materials compliance.

4.2.2 Design Constraints:

- **Power supply ripple/noise:** Affects image quality.
- **Sensor temperature:** CMOS sensors can suffer from thermal noise; need passive/active cooling in designs.
- **Lens compatibility & form factor:** Must match sensor size and application needs.
- **Signal integrity:** For high-speed interfaces (e.g., MIPI, USB 3.0).
- **Latency and frame rate requirements:** Application-specific (e.g., real-time video vs. slow imaging).

4.3 IR Cameras (Thermal or NIR)

Infrared cameras, or also referred to as thermal cameras will be used to detect devices and the IR signatures. For infrared cameras, there are procedures to operate such equipment so that there is no damage to the prototype or user. Therefore, we analyze both the safety standards and the design constraints for infrared cameras.

4.3.1 Safety Standards:

- **IEC 62471:** For near-infrared LEDs used in active IR illumination.
- **IEC 60601-1 (for medical use):** Electrical safety for medical-grade IR devices.
- **MIL-STD-810 / IP Ratings:** For ruggedized or outdoor IR cameras.
- **FDA/ISO guidelines (for thermographic fever screening):** ISO/TR 13154 & IEC 80601-2-59.

4.3.2 Design Constraints:

- **Sensor type:** InGaAs, VOx, microbolometers have different cooling and cost needs.
- **Spectral range:** SWIR, MWIR, or LWIR, each with different optics and sensor requirements.
- **Calibration:** Necessary for accurate temperature mapping.
- **Frame rate limitations:** Especially for uncooled IR sensors.
- **Shutter mechanisms:** May be needed to compensate for pixel drift (NUC – non-uniformity correction).

4.4 Photodiodes

Photodiodes are what we will be using to absorb light and transmit data to our system. There are many different kinds of material that could be used to build a photodiode/photodetector. Furthermore, we will analyze both the safety standards and the design constraints for photodiodes.

4.4.1 Safety Standards:

- **IEC 60825-1:** For laser safety if used with laser sources.
- **EN 61326-1 / FCC Part 15:** EMC standards.
- ISO 17025 (for calibrated light detection systems).
- RoHS / CE Compliance

4.4.2 Design Constraints:

- **Reverse bias voltage:** For avalanche photodiodes (APDs), breakdown voltage is critical.
- **Responsivity:** Depends on wavelength—must be matched to source.
- **Dark current and noise:** Affects sensitivity in low-light applications.
- **Speed vs. sensitivity trade-off:** Faster diodes have lower capacitance but may have reduced quantum efficiency.

- **Packaging:** TO-can, surface-mount, or fiber-coupled based on integration needs.

4.5 LiFePO4

Lithium Iron Phosphate (LiFePO4 or LFP) batteries have gained significant popularity due to their inherent safe significantly reduces the risk of thermal runaway, fire, and explosion. However, no battery technology is entirel operation.

4.5.1 Safety Standards

These safety standards aim to prevent various hazards, including:

- **Thermal Runaway:** An uncontrolled increase in temperature that can lead to fire or explosion.
- **Overcharging/Over-discharging:** Can cause cell degradation, internal short circuits, and safety risks.
- **Short Circuits:** Can lead to rapid discharge, overheating, and fire
- Physical Damage: Punctures, crushes, or impacts can compromise cell integrity and lead to internal shorts.
- **Leakage:** Electrolyte leakage can be corrosive and harmful.
- **Electrical Hazards:** Electric shock from improper wiring or insulation.

1. UN/DOT 38.3 (Transportation Testing)

This is a globally recognized standard for the safe transportation of lithium batteries, whether shipped alone or packed with equipment. All lithium batteries, including LiFePO4, must pass UN 38.3 to be legally transported by air, sea, rail, or road. It classifies lithium batteries as Class 9 dangerous goods due to their potential fire hazard.

The UN 38.3 test protocol involves a series of eight rigorous tests (T1-T8) that simulate conditions during transportation and potential abuse:

- T1: Altitude Simulation: Simulates low-pressure conditions at high altitudes to check for leakage, venting, or rupture.
- T2: Thermal Test: Exposes batteries to extreme temperature changes (e.g., -40°C to +72°C) to test seals and internal connections.
- T3: Vibration: Simulates transport vibrations to ensure structural integrity.
- T4: Shock: Tests the battery's ability to withstand mechanical shocks.
- T5: External Short Circuit: Checks how the battery responds to an external short circuit.
- T6: Impact (for cells only): Simulates impacts that could occur to individual cells.
- T7: Overcharge (for rechargeable batteries): Tests the battery's response to overcharging beyond its specified limits.
- T8: Forced Discharge (for cells only): Simulates severe discharge conditions.

To pass, batteries must not leak, vent, disassemble, rupture, or ignite, and their open-circuit voltage must remain within 90% of its initial value after the test.

2. IEC 62133-2 (Safety Requirements for Portable Applications)

IEC 62133 is an international standard specifically for the safety of rechargeable lithium-ion cells and batteries used in portable applications. The "dash-2" version (IEC 62133-2) is exclusive to lithium chemistries, including LiFePO4. This standard is widely accepted globally and is crucial for exporting lithium batteries into many markets.

Key aspects and tests covered by IEC 62133-2 include:

- Electrical Safety:
    - External Short Circuit: Testing the battery's behavior under short-circuit conditions.
    - Overcharge: Evaluating performance when continuously charged beyond its safe limits.
    - Forced Discharge: Assessing the behavior of cells when forced to discharge beyond their normal voltage.
    - Continuous Charging at Constant Voltage: Ensures stable operation under continuous charging.

- Mechanical Safety:
    - Vibration and Mechanical Shock: Ensures the battery can withstand physical stress during normal use and transport.
    - Crush: Simulates crushing forces.
    - Freefall: Tests resistance to drops.
- Thermal Abuse: Subjecting batteries to extreme and fluctuating temperatures to determine limits and potential hazards.
- Construction and Materials: Requirements for the battery's physical structure, internal components, and insulating materials to prevent hazards.
- Battery Management System (BMS): The standard indirectly emphasizes the importance of a robust BMS to prevent overcharge, over-discharge, overcurrent, and over-temperature conditions, though specific BMS requirements might be covered in other standards (e.g., UL 1973 for larger systems).

3. UL Standards (Underwriters Laboratories - Primarily North America)

UL standards are widely recognized in North America and cover various aspects of battery safety, from individual cells to large battery systems.

- UL 1642: Lithium Batteries (Cell Level) This standard focuses on the safety of individual lithium battery cells, including LiFePO4, to prevent fire or explosion when used as a power supply in products. It covers similar tests to UN 38.3 but is specifically aimed at cell-level safety within a broader product. Tests include short-circuit, abnormal charging, forced-discharging, crush, impact, shock, vibration, heating, temperature cycling, and fire exposure. It is often a prerequisite for higher-level UL certifications for battery packs and systems.
- UL 1973: Batteries for Use in Stationary and Motive Auxiliary Power Applications (Battery Pack/System Level) This standard is crucial for LiFePO4 battery packs and systems used in applications like renewable energy storage (stationary ESS), electric vehicles (as auxiliary power), and marine applications. UL 1973 assesses the entire battery system, including the cells, battery management system (BMS), enclosures, and interconnections. It covers:
    - Electrical Tests: Overcharge, over-discharge, short circuit, over-current.
    - Mechanical Tests: Vibration, shock, drop, impact, compression.
    - Environmental Tests: Temperature cycling, humidity, salt spray.
    - Fire Exposure: External fire tests to ensure the system does not propagate fire.
    - Tolerance to Internal Cell Failure: A critical test for lithium-ion batteries that evaluates whether a single cell thermal runaway can propagate to adjacent cells within the pack, leading to a larger fire or explosion. LiFePO4's inherent thermal stability often helps in passing this test more easily than other chemistries.
    - BMS Functionality: Verifies that the BMS effectively protects the battery from unsafe operating conditions.
- UL 9540: Energy Storage Systems and Equipment (System Level) While not exclusively for LiFePO4, UL 9540 is an overarching standard for the safety of complete Energy Storage Systems (ESS), which often utilize LiFePO4 batteries. It covers everything from battery modules and packs (often referencing UL 1973 for the battery component) to power conversion systems (PCS), controls, and fire suppression systems. UL 9540A is a related test method that specifically evaluates thermal runaway fire propagation in large-scale battery systems.

4. CE Marking (European Economic Area)

The CE mark indicates that a product complies with relevant European Union directives related to health, safety, and environmental protection. For LiFePO4 batteries, compliance often involves adherence to:

- Low Voltage Directive (LVD): Ensures electrical safety for equipment operating within certain voltage limits.
- EMC Directive (Electromagnetic Compatibility): Ensures the battery system does not cause or is not susceptible to excessive electromagnetic interference.
- RoHS Directive (Restriction of Hazardous Substances): Limits the use of certain hazardous materials in electrical and electronic equipment.
- Battery Directive: Addresses the collection, treatment, and recycling of waste batteries and accumulators.

5. Other Relevant Standards and Considerations

- ISO 9001 (Quality Management System): While not a safety standard directly, ISO 9001 certification for manufacturers demonstrates a commitment to quality control processes, which inherently contributes to product safety.
- Battery Management System (BMS): A robust BMS is a critical safety component in any LiFePO4 battery pack. While not a standalone "standard," its proper design and functionality are mandated by various certification standards (e.g., UL 1973). The BMS monitors cell voltage, current, temperature, and state of charge/health, taking protective actions like disconnecting the battery if unsafe conditions are detected.
- Application-Specific Standards: Depending on the end-use, additional standards may apply. For example, for electric vehicles, automotive standards (e.g., SAE, ECE R100) are relevant, and for marine applications, specific marine classification society rules apply.
- Installation Codes: National and local electrical codes (e.g., NFPA 70 National Electrical Code in the US, NFPA 855 for stationary ESS installations) provide requirements for the safe installation of battery systems.

4.5.2 Design Constraints

While LiFePO4 (LFP) batteries offer significant advantages in safety and cycle life, their design and integration into various applications come with a distinct set of constraints that engineers and product developers must carefully consider. These constraints dictate the feasibility, performance, cost, and overall suitability of LFP batteries for a given purpose.

1. Lower Energy Density (Gravimetric and Volumetric)

This is often cited as the primary drawback of LiFePO4 chemistry. Compared to other lithium-ion chemistries like NMC (Nickel Manganese Cobalt) or NCA (Nickel Cobalt Aluminum), LFP batteries store less energy per unit of weight (gravimetric energy density, Wh/kg) and per unit of volume (volumetric energy density, Wh/L).

- Implications:
    - **Larger Size and Heavier Weight:** For a given energy capacity (e.g., 100 kWh), an LFP battery pack will be significantly larger and heavier than an equivalent NMC or NCA pack. This is a major constraint for applications where space and weight are at

a premium, such as high-performance electric vehicles (where range is critical), portable electronics, drones, and aerospace applications.
    - **Reduced Range/Runtime:** In applications like EVs, the lower energy density translates directly to a shorter driving range for a battery of a given size or weight. For portable devices, it means shorter operating times between charges.
    - **Form Factor Limitations:** The physical size might limit the design freedom for compact products.

2. Lower Nominal Voltage and Flat Discharge Curve

A single LiFePO4 cell has a nominal voltage of 3.2V (compared to 3.6V-3.7V for NMC/NCA). While this contributes to its stability, it also introduces design challenges:

- **More Cells in Series for Higher Voltages:** To achieve higher system voltages (e.g., 12V, 24V, 48V, or hundreds of volts for EVs), more LFP cells need to be connected in series compared to other chemistries. For example, a 12V LFP battery typically requires 4 cells in series (4S), whereas an NMC battery might achieve a similar voltage with 3 cells. This increases the complexity of the battery pack, the number of interconnections, and the demands on the Battery Management System (BMS).
- **Challenging State-of-Charge (SoC) Estimation:** The discharge curve of an LFP cell is remarkably flat across a wide range of its State of Charge (SoC), particularly between 10% and 90% SoC. This flat voltage profile makes it difficult to accurately estimate the remaining capacity (SoC) solely based on voltage measurements.
    - **Implications:** Accurate SoC estimation becomes heavily reliant on Coulomb counting (tracking charge in and out), which can drift over time. This necessitates more sophisticated BMS algorithms, periodic full charging cycles to recalibrate the SoC, and potentially more robust current sensing. Inaccurate SoC can lead to user inconvenience (unexpected shutdowns) or premature cell degradation if discharged too deeply or charged incompletely.

3. Temperature Sensitivity (Especially Cold Weather Performance)

While LFP batteries are thermally stable and resist thermal runaway at high temperatures, their performance can degrade significantly in cold environments:

- **Reduced Capacity and Power Output:** At low temperatures (e.g., below 0°C/32°F), the internal resistance of LFP cells increases, leading to a noticeable drop in usable capacity and reduced power delivery. This is a critical constraint for applications in cold climates or those requiring high power bursts in chilly conditions (e.g., cold starts in vehicles).
- **Charging Limitations in Cold**: Charging LFP batteries below freezing temperatures can lead to lithium plating on the anode, which permanently damages the cell, reduces its capacity, and can pose a safety risk.
    - **Design Implications:** For cold weather applications, active thermal management systems (heating elements) are often required to maintain the battery within its optimal operating and charging temperature range. This adds complexity, cost, power consumption, and weight to the overall system.

4. Battery Management System (BMS) Complexity

While a robust BMS is essential for all lithium-ion batteries, the unique characteristics of LFP, particularly the flat voltage curve and need for cell balancing, impose specific demands:

- Precise Voltage Monitoring and Balancing: Due to the flat voltage curve, even small voltage differences between series-connected cells can indicate significant SoC imbalances. The BMS must precisely monitor each cell's voltage and actively balance them (either passively or actively) to ensure uniform charging and discharging, maximizing the pack's lifespan and usable capacity. This becomes increasingly challenging with more cells in series.
- Accurate SoC and SoH (State of Health) Estimation: As mentioned, the flat voltage curve complicates SoC estimation. The BMS needs advanced algorithms and accurate current measurement to provide reliable SoC readings. It also needs to track SoH to predict remaining battery life.
- Overcharge/Over-discharge Protection: While LFP is more tolerant to abuse, overcharging can still damage cells and over-discharging below the minimum safe voltage can lead to irreversible damage and cell reversal, especially in series strings. The BMS must rigorously enforce these limits.
- Overcurrent and Over-temperature Protection: The BMS must monitor current and temperature to prevent unsafe operating conditions, especially during high charge/discharge rates.

5. Cost

Although LFP prices have decreased significantly and are often more cost-effective than other lithium-ion chemistries on a dollar-per-Wh basis over their lifespan, the initial upfront cost can still be a constraint:

- Higher Initial Investment than Lead-Acid: For applications traditionally using lead-acid batteries, the initial capital expenditure for LFP can be higher, despite the long-term total cost of ownership (TCO) being lower due to longer lifespan and higher efficiency.
- Component Cost: High-quality LFP cells, combined with the sophisticated BMS and thermal management systems required, can contribute to a higher overall product cost.

6. Power Density vs. Energy Density Trade-off

While LFP batteries excel at high power output and have excellent discharge rates, optimizing for very high power applications might impact energy density. There's an inherent trade-off in battery design: maximizing one often comes at the expense of the other.

- Implications: For applications demanding extreme peak power (e.g., certain power tools, high-performance racing EVs), designers might need to oversize the battery or consider chemistries specifically optimized for power density.

7. Physical Pressure Requirements (for Prismatic Cells)

Some prismatic LFP cell designs benefit from external mechanical pressure to maintain optimal contact between electrode materials and the separator.

- Implications: This requires specific packing and enclosure designs for battery modules and packs to apply and maintain uniform pressure on the cells throughout their life, which adds complexity and potential weight to the mechanical design. Without proper pressure, capacity can degrade faster.

8. Self-Discharge Rate

LFP batteries generally have a slightly higher self-discharge rate compared to some other lithium-ion chemistries.

- Implications: For applications with long storage periods, this means the battery might lose a noticeable amount of charge. The BMS needs to be designed to enter low-power modes during storage to minimize drain, and users may need to periodically top-up batteries in long-term storage to prevent deep discharge.

Table 4.1: Table of Standards by Optical Component

Component	Electrical Safety Standards	Optical Safety Standards	EMC/EMI Compliance	Environmental/ Material Compliance	Application-Specific Standards
LED	- UL 8750 (LED lighting systems)	- IEC 62471 (Photobiological safety of lamps)	- EN 55032 / FCC Part 15	- RoHS, REACH	- IEC 60825-1 (if used like laser diodes)
CMOS Camera	- IEC 62368-1 (ICT/AV equipment)	- Not typically applicable unless IR-illuminated	- EN 61000-6-3/4, FCC Part 15	- RoHS, CE, WEEE	- ISO 16750 (for automotive cameras)
IR Camera	- IEC 60601-1 (for medical-grade IR systems)	- IEC 62471 (for NIR-illuminated systems)	- EN 61326-1, FCC Part 15	- RoHS, IP ratings (IP65, IP67, etc.)	- MIL-STD-810 (rugged design)
Photodiode	- General component standards (IEC 60747-5-5 for opto)	- IEC 60825-1 (if exposed to laser light)	- EN 61326 / FCC Part 15	- RoHS	- ISO/IEC 17025 (if calibrated for traceability)

Chapter 5 - Application of LLM and other Similar Platforms

This chapter explores the role of large language models (LLMs) and related artificial intelligence (AI) tools in supporting the research, design, and documentation processes of our senior design project. As generative AI continues to evolve rapidly, LLMs such as ChatGPT, Google Gemini, and Microsoft Copilot have emerged as powerful tools for technical ideation, component sourcing, document drafting, and comparative analysis. Their growing integration into modern productivity suites and software ecosystems enhances their accessibility, making them practical resources for engineering teams and students alike.

While the utility of these platforms is undeniable, it is equally important to examine their limitations, especially in areas like accuracy, model hallucination, and the need for verification of outputs. This chapter evaluates each tool in the context of our project's development and documents how they contributed to the final system we produced.

5.1 Large Language Models

Large language models are advanced AI systems trained on massive datasets of human language to understand, interpret, and generate human-like text. Their ability to engage in dynamic conversation and produce relevant, often accurate responses makes them extremely useful in the engineering workflow. Whether the task is sourcing parts, learning new programming techniques, comparing RF modules, or helping brainstorm testing protocols, LLMs can reduce the time required to arrive at useful insights.

Over the course of our project, we relied on multiple LLMs to support different needs. This included seeking code snippets for embedded systems development, identifying relevant signal processing algorithms, writing technical documentation, and planning project milestones. Each tool had unique strengths and shortcomings. To evaluate their role objectively, we focused on three key metrics: research capability, software integration, and communication clarity.

5.1.1 ChatGPT

ChatGPT, developed by OpenAI, is arguably the most recognizable and versatile LLM in active use today. It is available in both free and subscription tiers, and depending on the version, it can generate text, interpret images, browse the internet, and even engage in voice conversations.

We found ChatGPT to be particularly useful in helping us identify off-the-shelf components such as software-defined radios (SDRs), microcontrollers, and optical components. When formulating our LiFi-based communication system, we asked ChatGPT to provide a list of common light modulation schemes, which led to our implementation of On-Off Keying. Similarly, while evaluating SDRs for their frequency coverage, ChatGPT provided summarized comparisons, linked datasheets, and often recommended community-supported tools.

Beyond research, ChatGPT proved to be a strong writing assistant. It helped us clean up our technical reports, generate initial drafts of requirements, and even helped brainstorm professional ways to phrase security considerations. The model's strong contextual memory and grammar correction features allowed us to iterate on sections of our documentation quickly.

Another area where ChatGPT shined was in troubleshooting software issues. When developing the web-based frontend of our project, we used ChatGPT to explain ReactJS concepts, assist with JSON formatting, and solve errors thrown by our Node.js server. Although we verified every solution ourselves, the model accelerated our debugging process and often provided better explanations than general forum posts.

Despite these strengths, ChatGPT has its limitations. The free version does not have access to the internet, and even with a subscription, some responses require human validation. Nevertheless, as a versatile research companion and software aide, ChatGPT became a cornerstone of our workflow.

5.1.2 Google Gemini

Gemini, developed by Google, is another capable LLM that is deeply integrated with the Google ecosystem. Its strongest utility came from its seamless compatibility with Google Docs, Sheets, and Slides. This made it ideal for planning presentations and automating formatting tasks in collaborative documents.

We used Gemini to help structure early outlines of our project presentation. Its ability to suggest headings, recommend figures, and rewrite text in different tones made it well suited for preparing material that needed to be both informative and presentable. While ChatGPT was often better at technical reasoning, Gemini tended to suggest more creative and engaging ways to present our findings.

In addition, Gemini's capabilities in image captioning, content summarization, and sentence rephrasing were helpful when reviewing large PDFs or preparing abstracts for our technical sections. Gemini’s integrations allowed us to use it within Google Docs itself without leaving the workspace, which helped maintain focus and streamline collaboration.

That said, Gemini had shortcomings in its inability to reliably fetch current information from the internet. When we attempted to query real-time component prices or check availability from vendors, it could not match the utility of ChatGPT with browser tools enabled. Furthermore, while Gemini offered suggestions and summaries, its technical precision in niche areas such as RF engineering or embedded software was somewhat limited compared to ChatGPT.

Still, as a writing assistant and productivity tool, Gemini played a meaningful role. It helped us organize, clean up, and clarify documents quickly, making it ideal for non-technical tasks such as status reports and abstracts.

5.1.3 Microsoft Copilot

Microsoft Copilot is designed primarily to enhance productivity within the Microsoft Office suite. It integrates tightly with tools like Word, Excel, Outlook, and PowerPoint, offering suggestions as you type and assisting with content summarization and formatting.

For our project, Copilot was used more sparingly than ChatGPT or Gemini. We used it mostly for formatting slides, adjusting language in technical specifications, and converting tables between Excel and Word documents. Because of its integration, we were able to quickly create polished

content using templates provided in Word and PowerPoint. The real-time writing suggestions were helpful in improving sentence flow and eliminating grammatical errors.

One notable feature of Copilot is its ability to interpret Excel functions. When working on data from signal testing trials, we asked Copilot to help identify statistical patterns in latency and accuracy. It generated Excel formulas that made this task easier. Copilot was also able to convert raw tabular data into formatted charts with appropriate labels, which made our final results easier to present.

However, Copilot is not ideal for deep technical research. It lacks the ability to explain complex concepts such as modulation schemes or convolutional neural networks in depth. Its responses are more formulaic and sometimes generic. In addition, Copilot does not offer strong support for code generation or debugging.

We ultimately found Copilot to be most useful for finalizing deliverables. It worked well when the ideas were already there, and we simply needed help polishing the layout or grammar.

5.1.4 Limitations of Chat-Based LLMs

While the benefits of LLMs are substantial, there are also important limitations to consider. These models are not foolproof, and users must approach them with a critical mindset.

First, many models do not have real-time internet access. This can be problematic when attempting to verify current prices, check academic sources, or find up-to-date tutorials. In our case, we found that LLMs occasionally referenced deprecated APIs or outdated documentation. Therefore, we always followed up with independent verification, particularly for time-sensitive information.

Second, LLMs are prone to hallucinations. These are false statements or incorrect explanations presented confidently by the model. For example, one LLM told us that the LimeSDR Mini supported certain frequency bands which it did not. In another instance, we were given a sample Python script for SDR data processing that contained non-existent functions. These mistakes highlight the need for users to verify LLM-generated outputs, especially in technical contexts.

Third, while LLMs can write in grammatically correct sentences, they sometimes lack the clarity or precision expected in technical documentation. They tend to generalize and sometimes provide ambiguous phrasing. Therefore, human editing is always necessary to maintain quality and accuracy.

Lastly, ethical considerations must be kept in mind. Over-reliance on LLMs can hinder student learning if they are used as a crutch rather than a supplement. Moreover, they should not be used to fabricate citations or pass off generated content as original research.

5.2 Practical Impact on the Project

Throughout our development process, LLMs helped us in numerous practical ways. They contributed to writing design documentation, assisted in researching alternative SDR modules, guided us in selecting antennas, and helped us format engineering requirements. They saved us time during brainstorming sessions and gave us new perspectives when we were stuck.

We also used LLMs to explore different software frameworks, evaluate pros and cons of various image sensors, and get clarification on terms related to spectrum analysis. In some cases, they even provided scripts and command-line instructions we used to test our SDRs.

However, the most impactful use came from their ability to synthesize and simplify large technical subjects. They helped us make sense of the enormous body of RF, optical communication, and embedded system documentation. With their help, we could more quickly reach a baseline understanding before diving into detailed datasheets or academic papers.

5.3 Conclusion

In summary, LLMs are transforming the way engineering teams conduct research, plan system architectures, and document their work. For our senior design project, they provided critical support at every phase, from early ideation to final report formatting. Each model—ChatGPT, Google Gemini, and Microsoft Copilot—served a specific purpose based on its strengths. While not without flaws, these tools represent a significant advantage for student teams with limited resources.

As generative AI continues to evolve, we expect these tools to become even more embedded in the engineering workflow. Future student projects may use LLMs not only for documentation and research, but also for automated testing, simulation, and even AI-assisted prototyping. For now, they remain an important asset in our design toolkit, enabling us to deliver a well-rounded, technically sound, and professionally formatted final product.

Chapter 6 - Hardware Design

This chapter outlines the complete system architecture, spanning hardware components, embedded software, and application-level interfaces. It begins by detailing the division of responsibilities between our FPGA-accelerated PYNQ-Z2 board and the Raspberry Pi 5, both of which manage sensor input and real-time data processing. We then describe the software strategies for efficient signal detection, classification, and transmission, including low-power management and machine learning pipelines. Finally, we present the design and implementation of the web and mobile applications using the MERN stack, covering frontend UX decisions, backend API architecture, database schema design, and deployment strategy. Together, these elements form a tightly integrated platform for secure RF device detection and data delivery.

6.1 Hardware Design

For our final design, we have chosen to use a PYNQ-Z2 development board that combines an FPGA fabric with an ARM Cortex A9 processor. The FPGA will be used mainly to do the computationally expensive feature extraction and ultimately classification tasks using a spectrogram constructed from the SDR's data, and using the camera data as a way to confirm and improve the accuracy of the classification when the source itself is visible.

We also plan to use a Raspberry Pi 5, which has better compatibility with the optical system's cameras. Not all of the tasks for this system are a part of one pipeline and some of the sensor data doesn't require much processing beyond what the SDR and the RPI-5 can do for these systems, respectively. Therefore, many of the sensor control tasks and less heavy computations will be handled by the RPI-5 while the harder tasks such as object classification are done by the PYNQ-Z2. Ultimately the RPI-5 will contain the Li-Fi TX driver and quickly send out the fundamental information showing the RF signal, a picture of the surroundings, and geolocation data of the device. Later, data will be sent from the PYNQ-Z2 to the RPI-5 showing final classification of the detected signal source based on RF and image data.

PYNQ-Z2 Objectives:
- Object Classification (CNN models)
- Feature Extraction of FFT data and construction of Spectrogram

Raspberry Pi Objectives:
- Peripheral Control (Cameras, SDR, LED for Li-Fi TX communications, GPS)
- Image pre-processing (Resizing)

The PYNQ-Z2 and Raspberry Pi will be connected through USB 2.0, striking a balance between power efficiency, ease of implementation, and bandwidth. As an option for sending data with increased throughput, ethernet can be used instead between the two development boards.

The PYNQ-Z2 board recommends a 12V or 5V power supply. The board can be powered via micro-USB (5V supply instead), power jack, or the Vin and GND pins. The Raspberry Pi 5 requires a 5V power supply. [11]

The PYNQ-Z2 FPGA board has 512MB of on-board DRAM, 630 KB of fast block RAM, and 220 DSP slices. It comes with a built-in ARM Cortex-A9 dual core processor. [11]

For the total hardware design it is critical to define the parameters of what needs to be covered and what needs to be uncovered. In our design we have the following major hardware features: Infrared LED, Infrared Photodiode, Monopole Antenna, SDR, PCB voltage regulator, FPGA, Raspberry Pi, battery, and cameras. Out of these components, the antenna, LED, and photodiode each need to be uncovered to ensure that the line of sight is not disturbed or disrupted. Therefore in the housing for portability, each component can be housed and connected to the battery pack.

For perfect manufacturability, we would want a smaller, portable battery that is housed. However, for the purposes of this design course, we will have a larger battery pack to ensure that we can supply for a longer lifespan over a series of tests with limited rechargeability to not affect the battery in our minimum viable prototype stage. Therefore, we have designed a simple box shape

housing unit for the design. This will come with three holes that will hold the two cameras and the monopole antenna. This design will be modular to enable for geolocation but adding a second monopole antenna. Below is a simple hardware design created in a computer aided design software.

Figure 6.1: Side View of Design Prototype

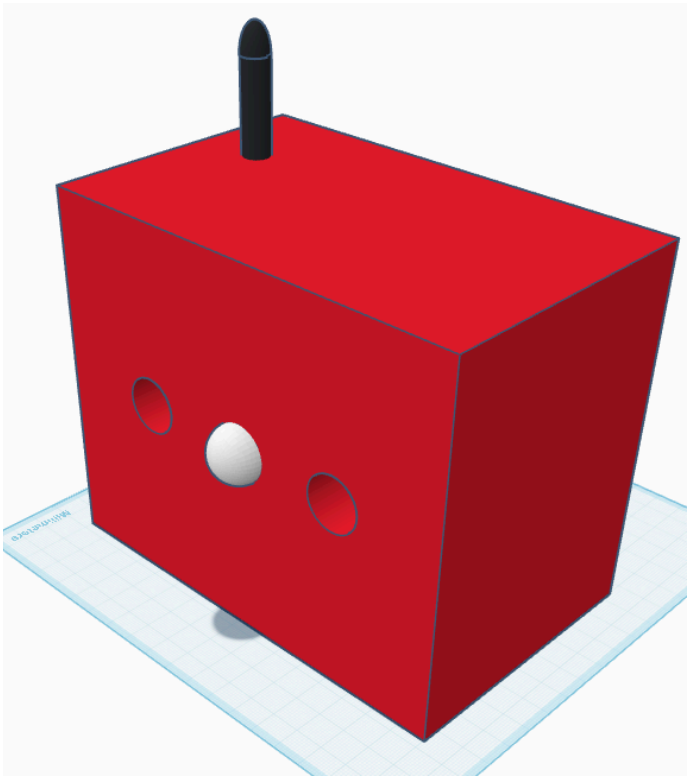
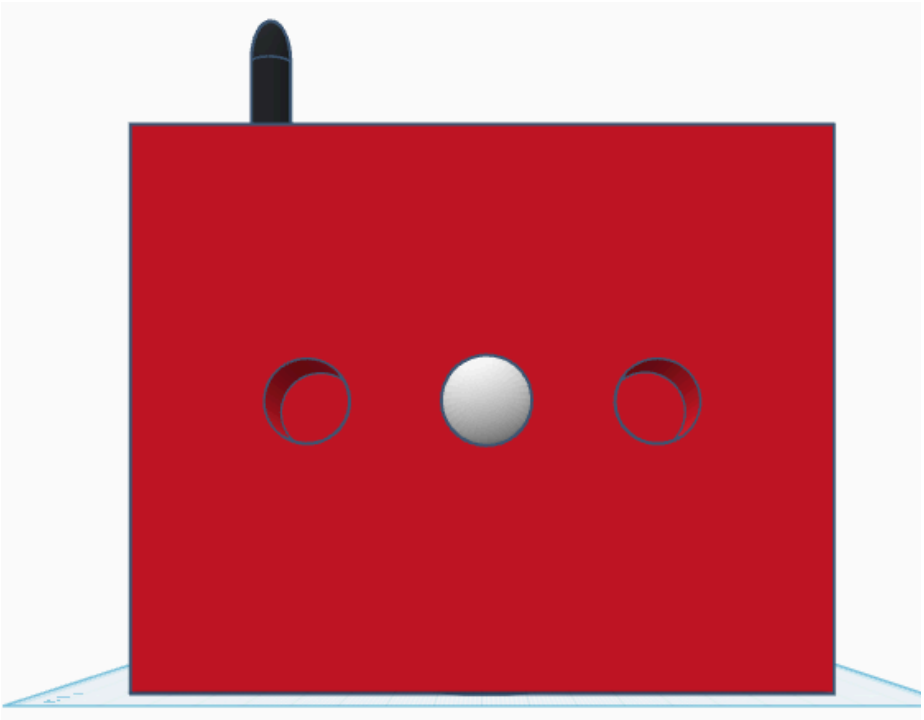


Figure 6.2: Front View of Design Prototype

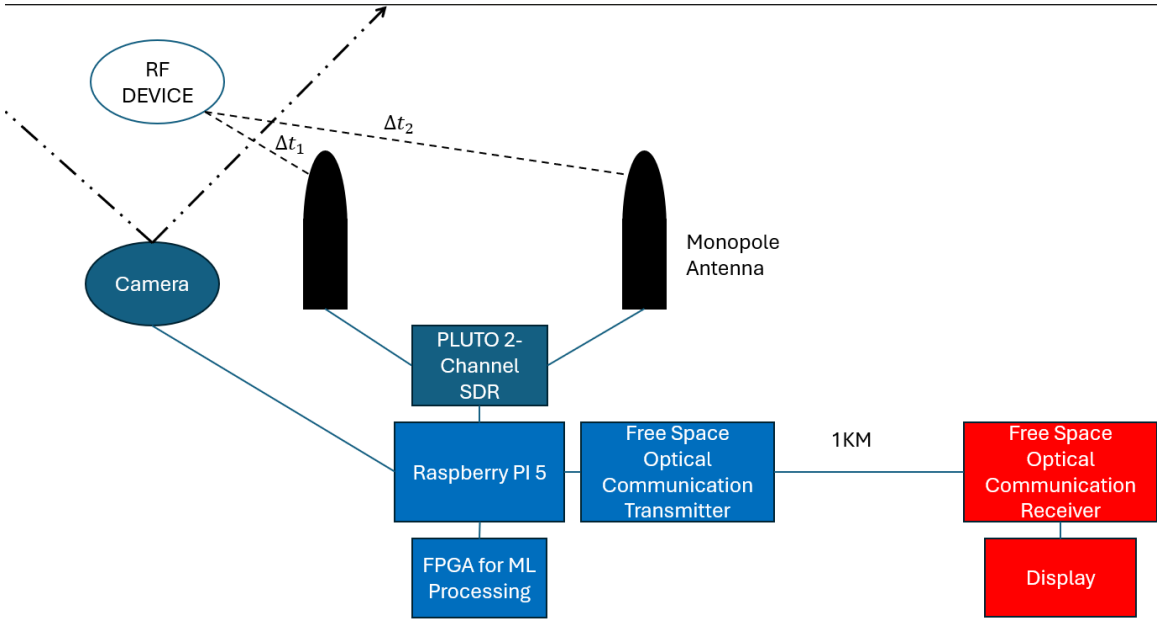


In the above image, the dark gray pole at the top of the design represents the monopole antenna. Note in this image, there is only one, however in the final design to achieve our stretch goal there will be two. Secondly, the white center is meant to represent the housing for the LED. This will include the lens design that will be critical to ensure the system is capable of free space optical communication. The next two holes are dedicated for the infrared and CMOS camera housing. This will also include their lens system as well.

The internal components of our device will be powered with an LiFePO4 battery. This will include the majority of the hardware components.

As previously mentioned, there is a design dedicated to achieving geolocation. Below is a simplified image of the proposed system. Note, that there are internal components depicted in the design.

Figure 6.3: Proposed System Design with Hardware Incorporated



In this design, by using a two monopole antenna set up, we can create a simple phase array that will calculate the time difference between the antennas and give general directionality of where the detected RF system is coming from. To achieve this, we must augment the PLUTO SDR to support the two-channel setup. It is well documented, though requires a manual override of the system to support.

It is also important to note that as previously mentioned, the FPGA boards will handle the processing of the data, whereas other devices such as the Raspberry Pi are meant to be supported for the other hardware components.

On the receiver side, the hardware design is not so critical. We currently envision a simple setup that acts as a “catcher” to receive the light signal and not obstruct line of sight. This will connect to a display to show the message.

Chapter 7 - Software Design

The software component of our RF detection and classification system is the cognitive core of the device responsible for interpreting raw sensor data, managing peripheral interactions, processing and classifying radio signals, and securely transmitting the resulting intelligence. Given the hardware constraints of embedded systems, the need for real-time operation, and the requirement to avoid active RF emissions, the software architecture must be simultaneously lightweight, modular, and responsive to asynchronous data streams.

7.1 Software Architecture Overview

The software for the RF device detection and Free-space Optical Communication subsystem is organized as a set of cooperating components that each solve a focused problem but together form a complete sensing and communication chain. Rather than a single monolithic program that attempts to handle RF detection, camera control, metadata fusion, optical modulation, demodulation, and visualization in one place, the design uses several small scripts and firmware modules that communicate through well-defined interfaces. This makes the system easier to debug, extend, and reason about, because each piece can be understood in isolation and exercised independently on the bench. It also matches the way the hardware is partitioned across the Raspberry Pi, the ADALM-Pluto SDR, the LED/photodiode link, and the ESP32 receiver.

At a high level, the system is split into three major functional layers that reflect the physical data flow and the main responsibilities of each device:

- **RF detection pipeline on the Raspberry Pi 5**, driven by the ADALM-Pluto SDR and primarily implemented in rf.py. This layer is responsible for configuring the SDR, collecting I/Q samples, running detection algorithms, and deciding when an RF event of interest has occurred. Its output is a compact description of "what was detected, where, and how strong," expressed as a small JSON-like structure that is independent of how that information will be used later.
- **Free-space Optical Communication backchannel**, consisting of a modulation script on the Raspberry Pi (modulation.py) and a demodulation firmware running on an ESP32 microcontroller. This layer takes compact metadata about an event, encodes it into an optical signal, and reconstructs it as a JSON object on the microcontroller side. Conceptually, this backchannel is a unidirectional "optical serial cable" between the Pi and the ESP32, but implemented using an LED, a photodiode, and simple encoding/decoding logic instead of copper wires.
- **Orchestration and visualization layer**, centered around main.py, metadata helper scripts such as extract_metadata.py and take_photo.py, and a lightweight Flask application (app.py and index.html). This layer coordinates camera capture, metadata generation, and user-facing display of the latest detection on a local web dashboard. It acts as the glue that turns low-level RF events into rich, human-readable records that include location, images, and device classification.

These components are connected in three logical pipelines that describe the flow of information through the system and clarify which scripts talk to which other scripts or firmware.

1. **RF path:**

PlutoSDR → rf.py → main.py → extract_metadata.py → modulation.py → app.py

In this path, the Raspberry Pi 5 configures the PlutoSDR to capture I/Q samples at specific center frequencies and bandwidths corresponding to devices of interest (for example, FRS and selected 2.4 GHz Wi-Fi channels). rf.py converts those raw complex samples into power spectral estimates, applies detection algorithms such as CA-CFAR or energy-over-baseline, and emits a compact description of any RF event that meets predefined criteria. This description typically includes the device classification, center frequency, detected peak, estimated SNR, and a

timestamp, along with other parameters such as RF bandwidth and sample rate that are useful for debugging.

When rf.py confirms an event, it invokes main.py and hands off this RF metadata, either via command-line arguments, environment variables, or a temporary JSON file. main.py then orchestrates higher-level tasks: triggering take_photo.py (or equivalent camera code) to capture an image, optionally running vision processing with an object detection script, and calling extract_metadata.py to merge RF data, GPS data, image information, and timing into a single, normalized JSON record. Once this record is complete, main.py passes control to modulation.py to prepare the optical transmission, and it ensures app.py has access to the updated metadata so the event can be displayed on the dashboard. In this way, the RF path connects a low-level RF spike to a fully documented detection entry.

2. **Optical path (Free-space Optical Communication backchannel):**

metadata2.json → modulation.py → LED driver → free-space optical channel → photodiode + analog front-end → ESP32 firmware → reconstructed JSON

In this path, the Raspberry Pi reads the most recent metadata from disk, specifically from metadata2.json, which holds the finalized event information produced by extract_metadata.py. modulation.py encodes this JSON payload into a Manchester-coded bitstream, wraps it with a preamble and terminator for framing, and drives a high-power LED via a driver circuit to send the bits as light pulses across the free-space optical channel. The LED pattern is therefore directly derived from the actual JSON text; any change in metadata translates into a different optical sequence.

On the receiving side, the ESP32 reads the photodiode output through an analog front-end and ADC. The firmware continuously samples the photodiode, compensates for ambient light by using baseline and threshold logic, and converts the analog waveform into digital decisions. It then performs Manchester decoding to recover the original data bits, reassembles the bits into bytes, identifies frame boundaries using the preamble and terminator, and finally reconstructs the original JSON payload. The end result is that the metadata written on the Pi appears as a structured JSON object inside the microcontroller, ready to be logged, displayed, or used by other embedded logic such as actuators, alarms, or integration with a larger embedded system.

3. **UI and external interface path:**

images + metadata2.json → Flask app.py + index.html → local browser

In this path, the focus is on human-readable presentation and status monitoring. The Flask application app.py loads the most recent metadata from metadata2.json and locates the latest image in the images/ directory. It then passes both the metadata and the image filename into the index.html template, which renders a clean dashboard. An operator on the same network can open this dashboard in a web browser and immediately see the most recent detection, including the device classification, frequencies, coordinates, and associated photo. The page effectively acts as a “last known detection” snapshot of the system state.

While not strictly required for core operation, this layer also forms the natural integration point with any external backends, such as a MERN stack. Because the dashboard is already built around JSON-style metadata, the same structures can be serialized and sent over HTTP to a Node/Express API for long-term storage and more advanced analysis. Historical queries, mobile apps, and remote dashboards can all be layered on top of this API without modifying the underlying RF or optical software.

The architecture deliberately isolates the responsibilities of each script and firmware component:

- rf.py only concerns itself with signal acquisition and detection. It knows how to talk to the PlutoSDR, compute PSDs, and apply thresholds, but it does not know how images are captured, how optical modulation works, or how the data is ultimately displayed. This keeps the RF logic focused and avoids pulling in unnecessary libraries or hardware dependencies.
- modulation.py focuses solely on translating JSON into a robust LED pattern according to a fixed framing and Manchester coding scheme. It does not care where the JSON came from, which device triggered it, or what the ESP32 will do with it once received. Its only contract is "given a JSON string, flash the LED so that the ESP32 can reconstruct that string."
- The ESP32 firmware is responsible for robust demodulation and parsing on the receiving side. It handles ADC sampling, thresholding, Manchester decoding, frame synchronization, and JSON parsing, but is agnostic to how the reconstructed metadata will be used afterward. Other microcontroller code can subscribe to "new message" events without touching the demodulation logic.
- main.py, extract_metadata.py, and app.py together form the glue layer that turns low-level RF events into human-readable detections and transmittable metadata. main.py orchestrates the overall workflow, extract_metadata.py standardizes the event description into a single JSON record, and app.py exposes that record to users via the browser.

By keeping boundaries clean, each component can be developed, tested, and debugged in isolation. For example:

- modulation.py can be tested by feeding it a fixed test JSON payload and watching the LED output with an oscilloscope or with the ESP32 firmware in a standalone mode, even if rf.py is disabled or the SDR is not connected.
- The ESP32 firmware can be validated using a simple LED blinker that flashes known patterns, without involving any RF detection or camera logic. This allows demodulation issues to be debugged in a controlled environment.
- app.py and index.html can be tested by placing a sample metadata2.json and test images in the appropriate directories, without needing active hardware. The Flask server can run on a desktop or laptop during UI development.

A small set of files and structures are used to connect components and enforce this modularity:

- **metadata2.json** stores the final compact description of an event, combining RF, GPS, and optional vision information into a single, consistent schema. Because it is written by

extract_metadata.py and read by both modulation.py and app.py, it serves as the central contract between low-level sensing, optical communication, and visualization. Any script that needs to know "what was the last event" only needs to read this file.

- **Image files in the images/ directory** hold captured still images (for example, JPEGs) that correspond to particular detections. The file with the most recent timestamp is considered the current image and is used in the dashboard. The filenames and timestamps implicitly link images to their associated metadata entries, and EXIF tags or folders can be used in future work to make this mapping even more explicit.
- **Log output from rf.py and main.py** provides a simple, text-based interface for debugging and for other processes to parse detections. For instance, log lines with a specific prefix (such as "RF_DETECT") can be used to quickly verify that the RF algorithms are firing as expected, even before optical transmission or UI components are enabled. Logs also help reconstruct the sequence of events when something goes wrong.

This combination of well-defined responsibilities, shared data artifacts, and clear data flow reduces coupling across the system and simplifies troubleshooting. When a problem arises, it is usually possible to localize it to one layer—RF detection, optical encoding/decoding, or orchestration/visualization—without needing to inspect every piece of code at once. This architectural clarity is important for both explaining the design and for future teams who might want to modify or extend it.

7.2 RF Detection Subsystem Software

The RF detection subsystem runs on the Raspberry Pi and is implemented in rf.py. It uses the pyadi-iio library to communicate with the ADALM-Pluto SDR and is responsible for continuously monitoring a small set of frequencies corresponding to specific device behaviors. By focusing on a handful of carefully chosen bands instead of sweeping the entire spectrum, the subsystem remains efficient, predictable, and easier to tune.

rf.py performs several recurring tasks:

- **Pluto configuration:** setting the SDR URI, sample rates, RF bandwidths, and manual gain levels for each band. These configuration values are chosen to balance resolution, sensitivity, and processing time. For instance, a narrower RF bandwidth around an FRS channel improves SNR and reduces out-of-band noise, while a wider bandwidth around Wi-Fi channels ensures the full OFDM signal is captured.
- **Round-robin scanning:** looping over a list of band slices, each representing a particular device class (for example, FRS, Wi-Fi channel 1, Wi-Fi channel 6, Wi-Fi channel 11), tuning to the target LO, allowing a short settling delay, and collecting a block of I/Q samples. Each slice encapsulates all parameters needed to perform detection for that device type, allowing new slices to be added or existing slices to be modified without changing the core scanning logic.

- **Signal analysis and detection:** using NumPy and SciPy to compute Welch power spectral density or integrated band power and applying different detection logic for narrowband and wideband signals. This division allows narrow, tone-like signals and broad, noise-like signals to be treated with algorithms that match their spectral shapes.
- **Metadata creation and event handoff:** assembling a structured metadata object including timestamp, frequencies, SNR, and a device label when a slice declares a detection, and launching main.py as a subprocess when an event is confirmed. This is the point where RF-only information is promoted to a complete event that will later be associated with images and location.

Rather than attempting to scan the entire spectrum, the software concentrates on four well-defined behaviors that map directly to the project's device classes:

- **Walkie-talkie (FRS)** near 462.6875 MHz, a narrowband signal with typical occupied bandwidth around 12.5–20 kHz.
- **Wi-Fi channel 1** around 2.412–2.417 GHz.
- **Wi-Fi channel 6** around 2.432–2.437 GHz, a wideband OFDM signal filling most of a 20 MHz channel.
- **Wi-Fi channel 11** region near 2.462 GHz, similar in spectral shape to a phone Wi-Fi but at a different center frequency.

Each device class is associated with either the "frs" or "wifi" band in rf.py. This association is encoded as a data structure indicating center frequency, sample rate, RF bandwidth, detection algorithm type, device label, and any additional gating logic such as minimum SNR or allowable offset from the nominal center. This makes it easy to adjust thresholds or add new devices by modifying a configuration table instead of rewriting core code.

The script evolved from a simple baseline-plus-offset detector to more principled techniques:

- For **FRS**, a CA-CFAR-style narrowband detector is used. Welch PSD is computed around the channel center, a Cell Under Test (CUT) is defined along with GUARD and REF bins, and a trimmed mean of REF bins estimates local noise power. The resulting SNR (CUT_dB minus REF_dB) is compared to a threshold derived from a desired false alarm rate, with hysteresis and a channel gate to reject peaks too far from the nominal center. This approach makes the detector much less sensitive to slow changes in the noise floor or distant interferers.
- For **Wi-Fi**, an energy-over-baseline detector is used. Band power in a moderately wide band around baseband is integrated, a baseline is tracked with an exponential moving average using different time constants for increases vs decreases, and the baseline is frozen temporarily after a detection to avoid chasing strong signals. A detection is declared when band power exceeds the baseline by a configurable margin for a minimum hold time and when the spectral centroid lies near DC. This combination helps distinguish real Wi-Fi activity from short noise bursts or spurious spikes.

This combination of detectors yields robust performance in both narrowband and wideband regimes without relying on brittle absolute dBFS thresholds. Because decisions are based on relative changes compared to a local noise estimate, the system adapts to variations in environment, antenna placement, and RF clutter.

A key design objective is to approximate a **five meter detection bubble**. Manual gain settings on the Pluto are chosen so that signals within approximately five meters produce a consistent and measurable jump in power, whereas more distant signals blend into the noise floor. Detection thresholds are tuned so that at roughly five meters the detector almost always fires, while beyond seven to ten meters it rarely does under typical conditions. This tuning turns the RF detector into a rough proximity sensor, not just a binary "on/off anywhere in range" indicator.

To prevent multiple triggers from a single long event, such as someone holding down a walkie-talkie button or a phone with Wi-Fi active for several seconds, each slice can enforce a **cooldown**. After a valid detection, the slice sets a timer and refuses to declare new detections for that band until the timer expires. This ensures that main.py is not overwhelmed with repeated events stemming from one continuous transmission and makes it easier to correlate "one detection" with "one captured image and one optical transmission."

When a detection is confirmed, rf.py constructs a metadata object containing fields such as timestamp, center_freq_hz, detected_freq_hz, snr_db_confirm, fs_hz, rf_bw_hz, device, and source. This object forms the RF portion of the metadata that is later merged with GPS and image information. Because the fields are always present and use consistent names and units, downstream scripts do not need to guess how RF information is formatted.

7.3 Free-space Optical Communication Modulation Software (Raspberry Pi)

The Free-space Optical Communication transmitter logic runs on the Raspberry Pi in a Python script referred to as modulation.py. Its job is to convert a finalized JSON metadata record into a light-based communication signal that the ESP32 can decode reliably, even in the presence of operating system jitter and ambient light variations. The script treats the LED as a digital output device that can be turned on and off with millisecond-to-hundreds-of-milliseconds precision.

The transmitter pipeline follows a clear sequence:

1. **Load the metadata:** open metadata2.json, read its contents as a string, and optionally validate that it is non-empty and parseable as JSON. This step ensures that only well-formed payloads are transmitted and that stale or partially written files are not sent accidentally.
2. **Serialize to bits:** convert the JSON string to UTF-8 bytes and, for each byte, generate an eight-bit binary representation (most significant bit first). Concatenate all bits into a payload bitstring. At this stage, the payload is simply the raw JSON text, but expressed as a sequence of 0s and 1s.

3. **Apply Manchester encoding:** map each data bit to two half-bits according to the Manchester scheme, where a data 0 is represented as a high-to-low transition and a data 1 as a low-to-high transition. This guarantees a transition in every bit period, which simplifies timing recovery and makes it easier for the ESP32 to distinguish between consecutive bits.
4. **Frame the transmission:** prepend a fixed-length preamble (for example, 16 bits of 1010…) and append a terminator byte (0x00). Preamble and terminator are added at the bit level so the receiver can identify the start and end of each frame even if it begins listening in the middle of a sequence. The script then combines preamble, payload, and terminator into a single frame bitstring.
5. **Transmit the frame:** iterate over the frame bits and call a send_manchester_bit function that uses time.sleep and gpiozero.LED to drive the LED through the appropriate half-bit states (high then low for a 0, low then high for a 1). Each half-bit lasts for a fixed fraction of the configured bit period.

Because the Raspberry Pi runs a general-purpose operating system, there is non-negligible jitter in sleep durations and GPIO toggles. The design takes a conservative approach by using a nominal bit period of around 100 ms in SD2 to prioritize reliability over throughput. This choice provides stable timing even with Python-level sleeps and background tasks, and it is sufficient for short, event-driven messages that carry only a few hundred bits of information. In practice, this means that a typical payload containing one JSON record can be transmitted in a few seconds, which is acceptable compared to the typical time between RF detections.

The gpiozero library wraps the lower-level RPi.GPIO interface, providing LED.on() and LED.off() methods. The LED is driven at a level sufficient for the photodiode to see a strong change in voltage but within safe limits for power and thermal dissipation. Power calculations and driver selection are handled at the hardware level, while the script only deals with logical on/off commands.

modulation.py is written so that it can be called repeatedly without needing to restart the Raspberry Pi. If metadata2.json does not exist or is empty, the script logs an error and exits gracefully instead of toggling the LED with meaningless data. If the JSON is present but cannot be parsed, the script can either refuse to transmit or fall back to a simple error payload that indicates a parsing issue. The main transmission function can be wrapped in a loop or triggered by main.py whenever a new detection is generated.

This modular design makes it straightforward to reuse the Free-space Optical Communication modulation script in other contexts, as long as the payload is presented as a JSON string. Changes to the JSON schema do not require changes to the framing or encoding logic; only the producer and consumer need to agree on field names and types. Future work could, for example, compress the JSON before transmission or add a checksum field, without altering the rest of the pipeline.

7.4 Free-space Optical Communication Demodulation Software (ESP32 Firmware)

On the receiver side, an ESP32 microcontroller runs custom Arduino-style C++ firmware that demodulates the optical signal coming from the photodiode. The overarching goal of this firmware is to take noisy, analog light-level measurements and convert them into a clean, structured JSON object that matches the payload originally sent by the Raspberry Pi. To do this reliably, the firmware has to account for ambient light, analog noise, and timing uncertainty, while still remaining simple enough to run continuously on a microcontroller with limited resources.

The demodulation process begins with a **baseline calibration phase**. With the LED intentionally left off, the ESP32 samples the ADC channel connected to the photodiode a large number of times and averages the results to estimate the ambient light level in the current environment. This baseline automatically captures contributions from room lighting, sunlight leakage, reflections, and the photodiode's own dark current. On top of this baseline, the firmware adds a margin of approximately 60 ADC counts to define two thresholds, hiThr and loThr. These thresholds are chosen based on empirical testing: under normal conditions, noise fluctuations and minor light flicker rarely exceed this 60-count margin, while an LED-on condition produces a much larger jump in ADC value. As a result, the LED being truly on or off is distinguishable from small variations in background illumination.

To further stabilize decisions, the firmware implements **hysteresis** with two thresholds instead of one:

- hiThr = baseline + margin is used to classify a half-bit interval as definitely high.
- loThr = baseline + margin / 2 is used as a lower threshold to avoid rapid toggling when readings hover near the boundary.

Using two thresholds helps prevent chattering if the ADC reading oscillates close to the decision point. Once a half-bit is classified as high based on hiThr, it takes a meaningful drop below loThr to be considered low again. This behavior is particularly important when the system is close to the minimum operating distance or under slowly changing ambient light, where small drifts could otherwise cause spurious transitions.

During actual reception, time is divided into **bit periods** that are designed to match the transmitter's configuration on the Raspberry Pi. Each bit period is split into two equal halves, corresponding to the two halves of a Manchester-encoded bit. Within each half-bit window, the firmware takes multiple ADC readings at evenly spaced intervals, accumulates them, and computes an average value for that half. This oversampling-and-averaging strategy reduces the impact of individual noisy samples and provides a more reliable estimate of the light level during each half of the bit.

Once the average ADC values for the first and second halves are computed, the firmware compares them against hiThr to classify each half as high or low. The pair of half-bit states is then mapped back to a Manchester-encoded data bit according to the chosen convention:

- A **high → low** transition (first half high, second half low) corresponds to a decoded data bit 0.
- A **low → high** transition (first half low, second half high) corresponds to a decoded data bit 1.

If both halves appear to be the same (both high or both low), an ambiguity is detected. In that case, a fallback rule selects the brighter half (the one with the higher average ADC value) as the "true" state and assigns a data bit accordingly. This fallback does not eliminate all possible errors, but it prevents the decoder from failing outright when a single half-bit is marginal or the LED timing is slightly skewed.

This oversampling, averaging, and thresholding approach significantly reduces the impact of random noise, ADC quantization effects, and small timing misalignments between the transmitter and receiver. Once a sequence of reliable data bits has been recovered in this way, the firmware begins assembling them into bytes using a simple **shift-register**approach. Each new bit is shifted into the current byte, and after eight bits have been collected, that byte is passed to a higher-level handler function and the shift register is reset for the next byte.

The byte handler is implemented as a **two-state finite state machine** that controls framing and ensures that only complete, properly delimited messages are accepted:

- In the **SEARCH_PREAMBLE** state, the firmware examines the incoming stream of bits (or bytes derived from them) to detect the known preamble pattern. The preamble is constructed as a repeated 1010… sequence at the bit level, which produces a distinctive toggling pattern in the decoded data. The firmware can maintain a rolling window of recent bits or bytes to look for this pattern at arbitrary offsets. When the exact preamble is observed, the receiver treats this as the start-of-frame marker, clears any previously accumulated payload data, and transitions to the next state.
- In the **READ_PAYLOAD** state, each subsequent byte is appended to a payload buffer. This continues until the receiver encounters the terminator byte 0x00, which was appended to the frame by the transmitter to explicitly mark the end of the payload. Upon receiving this terminator, the firmware treats the payload as complete and automatically transitions back to SEARCH_PREAMBLE, ready to capture the next frame.

At the end of each complete frame, the bytes stored in the payload buffer are interpreted as a UTF-8 string and passed to a JSON parser. The firmware then validates that the resulting JSON object contains the required fields, such as lat, lon, center_frequency, and device. Additional sanity checks can be applied at this stage, such as verifying that latitude and longitude fall within plausible bounds or that the device field is one of a known set of labels (for example, "walkie talkie," "phone," or "drone"). If the payload passes these checks, it is treated as a valid detection

event and can be logged, displayed over serial for debugging, or forwarded to other microcontroller routines.

If the parsing fails or the JSON is corrupted (for example, due to bit errors or incomplete frames), the firmware simply discards the current buffer and returns to SEARCH_PREAMBLE. This behavior prevents partially decoded or invalid data from propagating further into the system and simplifies error handling at higher levels. By adhering to this simple but robust framing scheme—preamble, Manchester-encoded payload, terminator, and JSON validation—the ESP32 can reliably separate consecutive messages, reject corrupted frames, and maintain stable operation even in the presence of noise, ambient light changes, and occasional timing disturbances.

7.5 Orchestration, Metadata, and Image Handling (Raspberry Pi)

Several scripts on the Raspberry Pi cooperate to respond to RF detections and prepare data for both the Free-space Optical Communication link and the local dashboard. The central orchestrator is main.py, which is typically launched by rf.py after a confirmed RF detection. main.py is responsible for turning a single detection flag into a complete "story" about the event: where it happened, what device generated it, what it looked like, and when it was captured.

main.py coordinates multiple tasks:

- Reading RF metadata produced by the scanner, either from command-line arguments, a temporary JSON, or environment variables. This step gives main.py enough information to know which device fired and which band slice triggered the event.
- Initiating image capture with a camera script such as take_photo.py, which saves images into a designated directory. The captured image provides visual context for the detection and allows later verification of the scene.
- Optionally running object detection with new_inference.py and propagating any resulting labels. These labels can be used to classify the object associated with the detection (for example, a drone vs a phone) and can be included in the final JSON for downstream analysis.
- Calling extract_metadata.py to consolidate RF details, GPS data, image metadata, and timestamps into a single JSON structure. This script acts as the "metadata factory" that knows how to merge multiple sources of information into a stable schema.
- Invoking modulation.py to send the final JSON across the Free-space Optical Communication link. Only after the JSON has been successfully written to metadata2.json and is ready to transmit does main.py call the modulation step.
- Ensuring the Flask app is running and aware of the new event so the dashboard reflects the latest detection. This can involve starting the Flask server if it is not already running or simply relying on the dashboard to pick up the updated metadata2.json on the next page load.

The extract_metadata.py script encapsulates the logic for generating the final metadata payload. It reads RF-related inputs (device label, center frequency, detected peak frequency, sample rate, RF

bandwidth, and SNR), retrieves positional and temporal information (GPS coordinates from EXIF tags or a GPS module, plus detection and capture timestamps), and normalizes this data into a stable schema. The resulting JSON is written to metadata2.json and serves as the single source of truth for both the Free-space Optical Communication transmitter and the Flask dashboard. Any new fields added in future revisions can be introduced here without changing the rest of the pipeline.

This separation of responsibilities keeps the orchestration logic clear and testable. RF detection, camera control, metadata generation, and Free-space Optical Communication modulation can all be developed and debugged independently, and they are joined only by their use of the shared JSON record and simple process-to-process interfaces. For example, if the camera is not available, main.py can still proceed with RF detection and modulation by populating placeholder values for image-related fields, while the rest of the system continues to function.

7.6 Local Web Dashboard (Flask app.py and index.html)

To give operators a quick, human-readable view of the latest detection, the system includes a lightweight web dashboard running directly on the Raspberry Pi. This dashboard provides immediate feedback that the RF detection and optical transmission pipelines are working and offers an easy way to demonstrate the system in real time.

The Flask backend, implemented in app.py, fulfills several roles:

- Serve the main dashboard at the root URL (/), which displays the latest image and metadata in a user-friendly layout.
- Provide an endpoint /images/<filename> that serves stored images from the images directory so they can be embedded in the page without exposing the entire filesystem.
- Expose a /health endpoint so other scripts, such as main.py, can verify that the server is running and responsive before attempting to notify the user of a new event.

On each request to the root URL, app.py loads metadata2.json, parses it into Python structures, selects the most recent entry if multiple records are present, locates the newest image file in the images directory, and passes this information into the index.html Jinja template for rendering. Basic error handling ensures that missing files or malformed JSON result in informative messages instead of server crashes. For instance, if no metadata is available yet, the page can display a “No detections yet” status instead of failing.

The index.html template defines how data is displayed. It shows a prominent title such as “RF Detection Results” at the top of the page and uses a responsive layout with the most recent image shown on one side and a “Detection Details” card on the other. The details card lists the device type, latitude, longitude, detected frequency, center frequency, and a human-readable timestamp.

Styling uses plain CSS embedded in the template, and url_for is used to generate image URLs so routing remains flexible across different deployment environments.

This simple Flask-based dashboard meets the requirement for a quick, local visualization of detections without adding unnecessary complexity or overhead to the Raspberry Pi. It runs entirely on the device, does not require external internet access, and can be viewed from any laptop or tablet on the same network. At the same time, it provides a natural starting point for future enhancements such as historical views, filtering by device type, or integration with a remote MERN backend.

7.7 Local Web Dashboard (Flask app.py and index.html)

The software stack is the result of deliberate trade-offs between performance, complexity, and development time, as well as alignment with available hardware and team skill sets.

On the Raspberry Pi, **Python 3** was chosen as the primary language because it offers a mature ecosystem for numerical computing and signal processing (NumPy, SciPy), simplifies integration with the ADALM-Pluto SDR via pyadi-iio, and makes it straightforward to implement orchestration logic, file I/O, and simple HTTP clients. Using Python allows the RF detection algorithms to be expressed clearly and compactly, and thresholds or parameters can be tuned quickly without recompiling code. The pyadi-iio library and the ADALM-Pluto SDR together provide high-level APIs for setting LO frequency, sample rate, bandwidth, and gain, which eliminates the need for custom SDR driver development and fits naturally into Python-based DSP workflows.

Flask and Jinja were selected for the local web dashboard because they are lightweight enough to run comfortably on the Pi alongside RF and Free-space Optical Communication tasks, require minimal boilerplate to expose endpoints and render templates, and are easy to adapt if the metadata schema or UI layout changes over time. Flask's routing model and Jinja's templating make it trivial to add new fields or change the way detections are displayed. The gpiozero library wraps lower-level GPIO handling with safe abstractions, reducing the chance of misconfiguring pins or leaving them in an undefined state and making Free-space Optical Communication LED control as simple as toggling a software object.

On the ESP32, **Arduino-style C++** was chosen to provide compiled performance and deterministic timing for ADC sampling and Manchester decoding, using a familiar setup() and loop() pattern that matches the continuous reception model required for the photodiode. The platform's built-in support for analog inputs and timers, combined with a large community and abundant examples, made it well suited to implementing the Free-space Optical Communication receiver. While lower-level frameworks like ESP-IDF could offer additional control,

Arduino-style C++ strikes a good balance between control and development speed for this project.

Alternative technology choices were considered but not selected for various reasons. For example, implementing the RF detection pipeline in C or C++ on the Pi could have offered slightly lower latency, but at the cost of longer development time and more complex memory management. Using a heavier web framework instead of Flask would have added features that were unnecessary for a simple local dashboard. Similarly, a higher-throughput physical link such as Wi-Fi or Ethernet between the Pi and the microcontroller would have removed the need for optical modulation and demodulation, but would not have met the project requirement of demonstrating a Free-space Optical Communication backchannel.

Taken together, these technology choices yield a system that is modular, explainable, and aligned with project requirements. RF detection runs close to the hardware on the Raspberry Pi, Free-space Optical Communication demodulation runs in real time on the ESP32, and visualization and long-term storage are handled by Flask and the MERN stack. Each component can be extended, reimplemented in another language, or replaced entirely in future work without disrupting the overall architecture, as long as the JSON-based contracts between components are preserved.

Chapter 8 - System Fabrication/Prototype Construction

Our project has been approved to not require a significant Printed Circuit Board (PCB) design. However, per our project requirements, we must have some PCB integration included in our final design. Therefore, we must ensure that we are effectively using a PCB design that fits the overall needs and removes the burden on the development boards. One key component in our design is our GPS module. The module selected requires a 5V input to work. Therefore, instead of directly wiring the 5V output from the Raspberry Pi 5 to the input of the GPS module, we will instead use the PCB to regulate the 5V.

This design has been tested and proven to work in previous experiments.

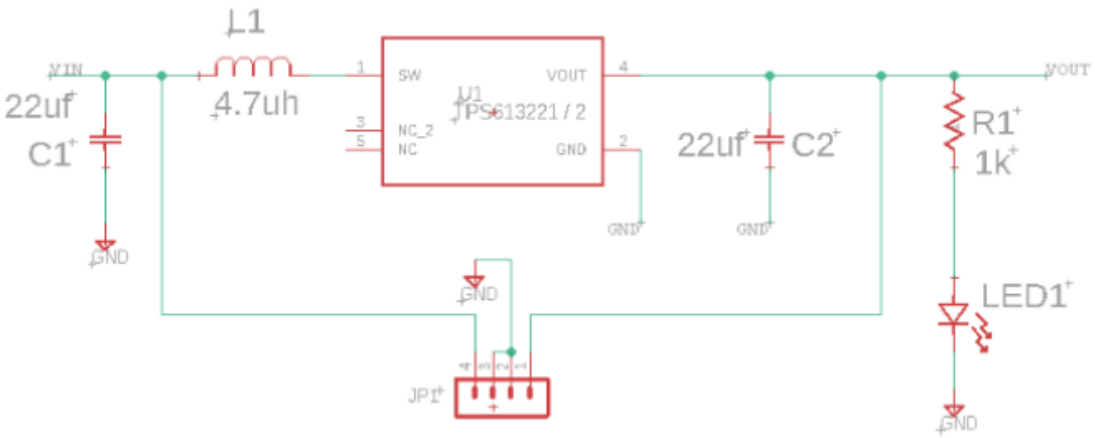


Figure 8.1: *Proposed PCB Schematic with Hardware Incorporated*

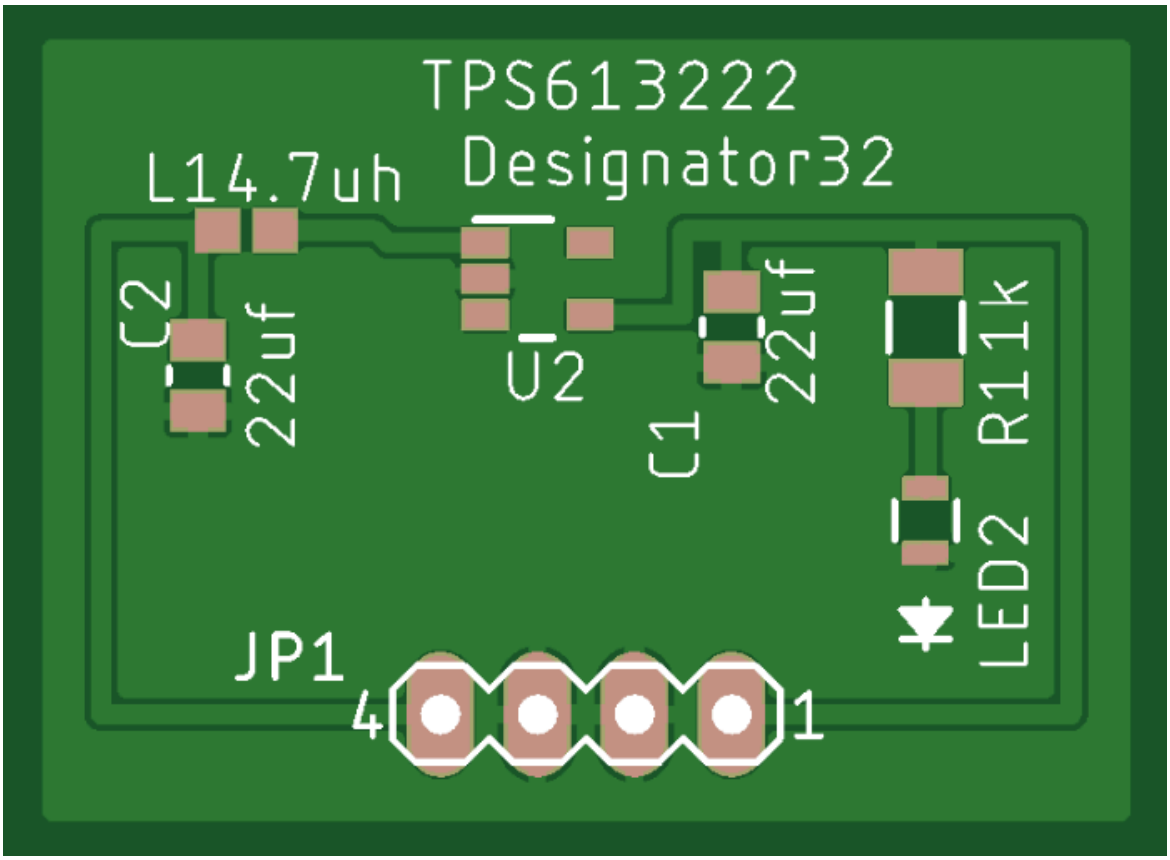


Figure 8.2: *Proposed PCB Layout with Hardware Outlined*

Figure 8.1 and 8.2 both showcase the final design for the 5V Booster Regulator. This regulator will be used to power the GPS module and serve as our PCB for this prototype. This 5V design has the following components:

Part Name	Part Number	# of Units

22uF Capacitor	CL21A226MQQNNNE	4
4.7uH Inductor	LQM18PN4R7MFRL	2
TPS613222ADBVR +5V Regulator	TPS613222ADBVR	1
1K Resistor	RC1206FR-071KL	2
LED Assorted	B01CUGA8JO	1
Male 2.54mm Header	B06ZZN8L9S	1

Each part is critical to the overall functionality of the 5V booster regulator.

Chapter 9- System Testing and Evaluation

This chapter outlines the system testing and evaluation process to ensure that the components purchased will serve their proposed purpose and lead to a successful demonstration by the end of Senior Design 2. There will first be a list of checks that will occur and then we will test the overall connection and ensure that there are appropriate voltage and current supplied to the appropriate devices. Due to our project having no significant PCB design, we will be using development boards to help integrate these components, reducing the error in the electrical components.

9.1 Hardware and Software Testing

To test the hardware, a simple turn-on check will initially ensue to ensure that the part purchased is not a defective piece. The major components in our project are as follows: Raspberry Pi 5, PYNQ-Z2 FPGA+ARM board, ADALM-Pluto SDR, Monopole Antennas, LiFePO4 Battery Supply, Adafruit Ultimate GPS, 1550nm Mounted LED, 1550nm Photodiode, and CMOS Camera.

Each component must independently turn on and demonstrate some connectivity to the development board's software on a computer once connected to a power supply. To specify, we will test the following:

Raspberry Pi 5 - Connect to display and program a "Hello World" test case.

FPGA - Connect to display and program a "Hello World" test case.

ADALM-Pluto SDR + Monopole Antenna - Connect to display, test 2.4GHz and 5GHz spectrum surveillance on software.

Battery Supply - Using a multimeter, verify the output voltage draw of 12V. No software test required.

GPS - Connect and collect test GPS data.

LED + Photodiode - Turn LED on and verify that there is infrared light emitting and captured at a small distance (<0.5m).

CMOS Camera - Connect to display and ensure properly working.

This will conclude and verify that the devices procured do work and serve their common function.

9.2 Performance Evaluation

To successfully demonstrate our final prototype, there are a key parameters we must achieve. First, data collection of the electromagnetic spectrum. Since we are using WiFi to test, we can observe and collect data at the 2.4GHz and 5GHz range. For the SDR and antennas, we must observe and extract dBm changes in the spectrum in intervals over the course of one second.

Once we have done so, we must extract an image from the CMOS camera. This is critical if we are to implement an image identification algorithm during Senior Design 2. Therefore we must ensure that the CMOS camera does have our desired Field of View and pixel size.

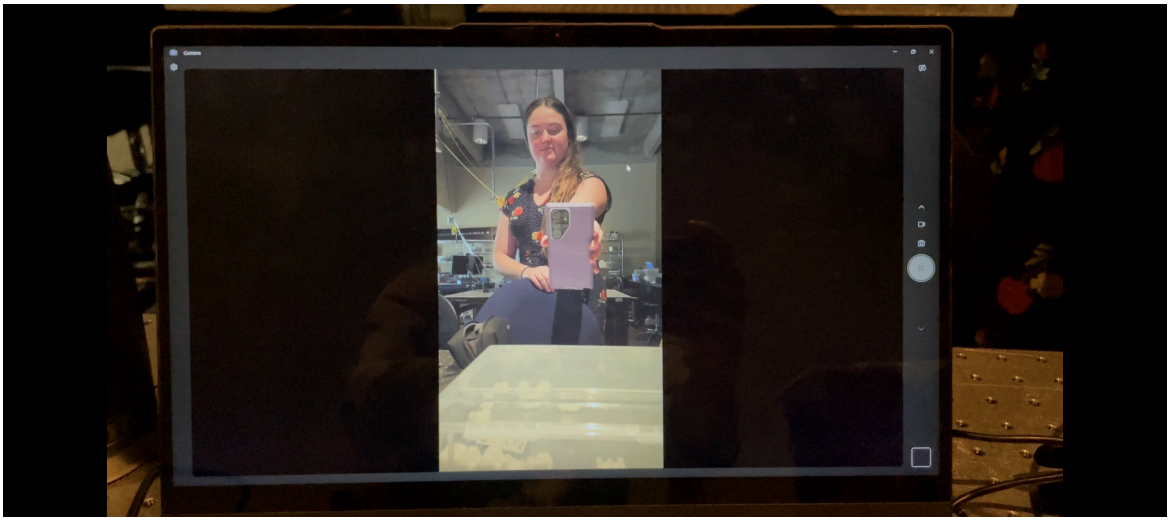


Figure 9.1: *Example of testing of CMOS functionality connected to computer.*

Then we must ensure that the 5V booster regulator does power the GPS module independently without error. If we can still extract the location data from the GPS module while connected to the 5V regulator PCB, then it has passed its evaluation.

Finally, the LED and photodiode must be able to send and receive 1's and 0's to demonstrate that On-Off Keying modulation is possible and operationally functional. This will be tested further in the optoelectronics feasibility study and testing.

9.3 Optoelectronics Feasibility Study and Testing

We have evaluated the feasibility and preliminary testing of the system's optoelectronic components, with a focus on Li-Fi-based data transmission and imaging enhancement. The demonstration aims to validate long-distance optical communication using On-Off Keying (OOK) modulation and collimated light transmission. Key subsystems include LED-based transmitters, photodiode arrays for reception, and CMOS imaging sensors with lens systems for enhanced optical resolution. The test serves as a foundational step in confirming the viability of our optical communication pipeline under constrained, field-like conditions.

The goal of the demonstration is to successfully transmit a text message using on-off keyin via LiFi to a device that is a significant distance away. To show this, we will need an LED or laser diode, 2 Arduinos with a display, and a photodiode with a corresponding demodulator. This setup will integrate a sophisticated lens system to collimate the light beam to travel long distances. Also by creating an array of photodiodes, we can increase the active area for reception.

Additionally, we will have the CMOS sensor also contain a lens system to increase the resolution of the images captured by the device to be used for signal identification. All together, this photo should produce a text file that is 1MB large that will be modulated via On-Off Keying. This text file will appear on the LiFi Tx's display then be sent and display on the LiFi Rx from up to 1km away.

This simple setup should be supported by a simple portable battery power supply

9.4 Overall Integration

The overall integration will include each major component being connected and togglable through the Raspberry Pi 5. Each system should be able to be turned on and off all from the Raspberry Pi 5. This will ensure that there is complete control over each subsystem in one central node. This integration should be able to: extract dBm data from the electromagnetic spectrum range of 2.4GHz and 5GHz, trigger an image capture, collect GPS coordinates, and toggle the LED. By doing so, we have demonstrated that the components are accurate and are integrable with the software that will be developed.

To successful demonstrate our capabilities, we aim to hold the following demonstration:

1. Detect a handheld radio
2. Collect key characteristics
3. Take a snapshot of the area
4. Create a text file with critical information about the signal
5. Transmit the text file to a device at a significant distance away using LiFi.
6. Receive and demodulate the signal to read the text file.
7. Repeat each step but with a WiFi enabled device.

We have no significant PCB required for this project, therefore our design will include development boards. These development boards will be responsible for our electrical design.

9.5 Plan for Senior Design 2

During Senior Design 2, Steps 2-6 will be able to each be individually tested and measured for success based on our outline requirements. We will optimize the optical components to increase the range that we can capture the LED emissions. This is interchangeable by increasing the power output of the LED and increasing the area that the photodiode covers to collect. We can also increase the Line of Sight, decreasing any possible anomalies that may intervene with the collection of the 1550nm light. As the optical team is working on optimizing the free space optical communication, the computer engineering team will be focused on creating an extraction program and optimizing the latency from extraction to identification of the device, including using the image identification data. A large majority of SD2 will be spent focusing on the software implementation and the optimization of the latency.

Chapter 10 - Administration

This chapter outlines the administrative framework supporting the development and execution of the project, including sponsorship, budgeting, procurement, milestones, and work distribution. With funding support from a U.S. Army Electromagnetic Warfare Officer, our team is committed to building a cost-effective system using off-the-shelf components. We detail our preliminary bill of materials, define measurable engineering milestones, and present our team's role assignments based on individual expertise. The chapter concludes with a declaration of appropriate use of AI assistance in the project documentation process.

10.1 Budget

Our project will be sponsored by Chief Warrant Officer 5 Richard Godfrey, an Electromagnetic Warfare (EW) Officer for the US Army. As of right now, there is no set budget, however we aim to keep cost to a minimum.

10.1.1 Purchasing Materials

We will submit documentation on an as-needed basis for supplies and wait for approval. Supplies will be delivered to UCF.

10.1.2 Low Cost Solution

Our goal for this project is to provide a low cost solution. This means we should ideally aim for our budget to be as low as possible. As a result, we might not get the best results from our data. However, if this is the cheapest solution, then these can easily be scaled into a physical array to collect significant data about a particular region.

10.2 Bill of Materials

Our simplified current build of materials is as follows:

Table 10.1: Bill of Materials

Component	Part Name	Cost
RF Antenna Detection	Monopole Antenna (x4)	~$4 per unit
RF SDR	ADALM-Pluto SDR (x2)	~$230 per unit
Development Board	Raspberry Pi 5	$80-$130 per unit
Camera	CMOS	$70-80 per unit
LiFi Tx	Infrared LED	$10-$50 per unit
LiFi Rx	Infrared Photodiode	$1-$50 per unit
Battery	Power Queen 12V 50Ah Deep Cycle Lithium Battery	$139.99 per unit

10.3 Milestones

Our team's milestones will come from successfully completing each step of our integration roadmap. Below is a breakdown of our milestones:

Table 10.2: Design Integration Milestones

Milestone	Description
RF Detection (VHF/UHF)	Detect VHF/UHF transmitting walkie-talkie in the laboratory

RF Detection (WiFi)	Detect WiFi 2.5GHz/5GHz transmitting device in the laboratory
I/Q Data Collection (VHF/UHF)	Capture I/Q data from a VHF/UHF transmitting device
I/Q Data Collection (WiFi)	Capture I/Q data from a WiFi 2.5GHz/5GHz transmitting device
Image Capture (CMOS)	Trigger an image to be capture after detecting a device with high resolution with CMOS
Image Capture (IR)	Trigger an image to be capture after detecting a device with high resolution with IR
Digital Signal Processing	Achieve a high probability of accurate device detection by ML algorithm data from the RF environment
Combined DSP	Achieve a high probability of accurate device detection by ML algorithm combining data from the RF environment and image capture.
LiFi Tx	Transmit 1MB file from one device to another at 1km away using LiFi.
LiFi Rx	Receive 1MB file from one device to another at 1km away using LiFi.
Total Integration	Combine all functionalities.
Geolocation	Augment SDR to have two antennas and create a phase array for geolocation.

Table 10.3: Senior Design 1 Milestones

Senior Design 1 Milestones	
Initial Group Introductions	05/27/25
D&C Report	05/30/25
D&C Feedback Meeting	06/03/25
Midterm Report	07/07/25
Midterm Revision Feedback Meeting	07/15/25
Final Report	07/29/25
Demo Video	07/29/25

Table 10.4: *Senior Design 2 Milestones*

Senior Design 2 Milestones	
Components Selection	09/14/25
System Design	09/21/25
System Testing	10/05/25
PCB Design	10/12/25

PCB Testing	10/26/25
Prototype Completion	11/09/25
Demo Day	12/05/25

10.4 Work Distribution

The work related to this project has been distributed evenly. We have broken down jobs and tasks dependent on our majors. Below is a chart that shows each critical aspect of this project and the primary and secondary person dedicated to support it.

Table 8.3: Work Distribution

Function	Primary	Secondary
RF Antenna/SDR Collection	Zack Z.	Zach F.
Image Capture	Sarah T.	Zack Z.
Digital Signal Processing	Zach F.	Bill N.
LiFi Tx	Zack Z.	Sarah T.
LiFi Rx	Sarah T.	Zack Z.
WebApp Design	Bill N.	Zach F.
Power Supply	Bill N.	Zack Z.

It should be noted that although each role is assigned, there has been significant crossover between the teams. This crossover has been extremely helpful in accomplishing our tasks.

10.5 Declaration

We hereby declare that we have not copied more than 7 pages from the Large Language Model (LLM). We have utilized LLM for drafting, outlining, comparing, summarizing, and proofreading purposes.

Chapter 11 - Security Considerations

This section analyzes security considerations for the RF Device Detection and Free-space Optical Communication (FSOC) subsystem, with a focus on four main software components: RF detection, object detection, optical modulation on the Raspberry Pi, and optical demodulation on the ESP32. The goal is not only to identify potential vulnerabilities but also to explain how the current design mitigates many of them and to outline future improvements that could further harden the system.

Because the system is intended for field use, security cannot be treated as an afterthought. Each subsystem deals with data that affects operational decisions: RF detections indicate nearby devices, object detection suggests what those devices look like, and the FSOC link transports this information to an embedded receiver. Compromising any of these steps could lead to misleading situational awareness or complete loss of trust in the platform. The subsections below take a defense-in-depth perspective, starting with RF detection, then object detection, and finally the FSOC transmit and receive paths.

11.1 Security Goals and Threat Model

At a high level, the system's security goals for these subsystems can be summarized as:

- **Integrity:** Ensure that RF detections, images, and JSON metadata are not silently modified by adversaries in ways that misrepresent reality.
- **Availability:** Keep the detection and communication chain operational despite noise, interference, or deliberate jamming attempts.
- **Authenticity:** Provide confidence that detections and messages originate from the actual sensor platform, not from a spoofing source.
- **Confidentiality (where relevant):** Protect sensitive information such as GPS coordinates or images from being trivially intercepted, especially if the system is extended to remote networks.

The primary threats considered for this project are:

- **Spoofing and misleading signals**, where an attacker sends RF or optical signals intended to cause false detections or false messages.

- **Jamming and denial of service (DoS)**, where the adversary overwhelms the RF front end, camera, or optical receiver so that valid events cannot be detected or decoded.
- **Data tampering**, where detections or metadata are altered in transit or at rest to misrepresent what was actually observed.
- **Input-driven faults**, where malformed or extreme inputs (e.g., long JSON payloads, unexpected bit patterns) cause crashes or undefined behavior.

The following subsections map these threats onto the four main technical areas of the project.

11.2 RF Detection Security

The RF detection subsystem in rf.py interacts directly with the ADALM-Pluto SDR to scan specific frequency bands and declare detections. This makes it the first "gate" of the system: if an adversary can reliably fool the detector, all later stages—including object detection and FSOC—may be working with misleading information.

11.2.1 Spoofing and False Detections

A straightforward attack is to transmit signals that mimic the expected RF signatures of monitored devices. For example, an adversary could:

- Transmit narrowband tones near 462.6875 MHz to masquerade as an FRS walkie-talkie.
- Generate wideband OFDM-like noise around Wi-Fi channels 1, 6, or 11 to resemble a phone or access point.

Because rf.py uses a CA-CFAR style detector for FRS and an energy-over-baseline detector for Wi-Fi, it is inherently sensitive to power increases relative to the local noise floor. A sufficiently close or powerful transmitter could cause repeated false detections.

The software design mitigates this in several ways:

- **Tight frequency gating:** For FRS, the CA-CFAR detector only accepts peaks within a narrow neighborhood of the nominal center frequency. Signals that are too far off-center are rejected. This makes it harder to spoof a detection using arbitrary nearby tones.
- **SNR and shape checks:** The code compares the CUT bin's power to trimmed-mean reference bins and can enforce minimum SNR thresholds. This makes low-SNR or unusually broad signals less likely to be accepted as valid FRS detections.
- **Spectral centroid gating for Wi-Fi:** For wideband slices, the detection logic can require that the spectral centroid remains close to DC, as expected when the LO is centered on a Wi-Fi channel. This helps distinguish genuine Wi-Fi activity from narrowband interferers or spiky noise.

These measures do not fully prevent spoofing with carefully crafted signals, but they raise the bar by requiring attackers to approximate both the **frequency** and **spectral shape** of the target devices rather than simply transmitting at high power.

11.2.2 Spoofing and False Detections

An attacker could try to **jam** the system by raising the noise floor across monitored bands, either with broadband RF noise or strong out-of-band signals that leak into the passband. If the module relied only on fixed absolute thresholds in dBFS, the detector might become either permanently triggered or permanently blind in the presence of strong interference.

The detection algorithms in rf.py already incorporate some robustness:

- **Relative detection metrics:** Both the CA-CFAR and energy-over-baseline approaches use local noise estimates instead of fixed absolute thresholds. This allows the detector to adapt to gradual noise changes.
- **Baseline freezing after detection:** For the Wi-Fi detector, the baseline can be "frozen" for a short period following a strong detection, preventing it from immediately rising to match a strong transient. This reduces the chance that jamming or a consistent strong signal permanently desensitizes the system.

However, there are limits. Extremely strong jamming can saturate the SDR front end, distort the spectrum, and make any kind of spectral detection unreliable. The software can mitigate this by:

- Monitoring for **saturation indicators** (e.g., unusually flat, high PSD across the band) and logging when conditions appear jammed.
- Falling back to conservative behavior (e.g., flagging the band as "unreliable" rather than repeatedly declaring detections) when saturation is detected.

These design patterns—detection based on relative metrics, logging saturation, and treating obviously overloaded conditions as special—help preserve availability and avoid misleading detections under hostile RF environments.

11.2.3 Data Integrity and Logging

The metadata produced by rf.py is passed into main.py and eventually into metadata2.json. An attacker with filesystem access to the Raspberry Pi could theoretically modify these records at rest, changing device labels or frequencies.

To reduce this risk at the software level:

- **Strict schema:** The RF metadata is written with a consistent set of fields (center_freq_hz, detected_freq_hz, snr_db_confirm, etc.). Downstream scripts expect these fields, which makes it easier to detect gross tampering (e.g., missing or non-numeric values).
- **Structured logs:** rf.py prints detection lines with recognizable prefixes and fields. Operators can cross-check what rf.py reported in real time against the content of metadata2.json to detect inconsistencies.

- **File permissions (deployment practice):** In deployment, the Pi can run rf.py and main.py under an unprivileged user with limited write access to only the necessary directories (e.g., images and metadata folders). This makes unauthorized modifications more difficult and constrains the blast radius of any compromise.

11.3 Object Detection Security

Object detection, implemented by new_inference.py and related code, analyzes captured images to provide additional context (for example, identifying that a detected RF event corresponds to a drone, phone, or other object). While this component is not strictly required for RF detection, its outputs influence how operators interpret detections, so security considerations are important.

11.3.1 Adversarial Inputs and Misclassification

Vision systems are susceptible to **adversarial examples** and misclassification, especially if attackers can control what appears in the camera's field of view. Potential threats include:

- **Pattern-based attacks:** An adversary attaching specific patterns, markers, or camouflage to equipment that cause the detector to mislabel the object.
- **Occlusion or distraction:** Covering the actual device while presenting other objects more prominently in the frame, encouraging the detector to label the wrong object.
- **Forced misclassification:** Using posters or displays that resemble objects in the training dataset, drawing attention away from the true target.

The project mitigates the impact of these risks by:

- Treating object detection as **advisory** rather than authoritative. RF detection is still the primary trigger, and object labels are used as supplementary context, not as the sole basis for decisions.
- Logging the full JSON record, including RF parameters and timestamps, along with the image. Even if the label is wrong, analysts can review images later and override or correct the classification.

In future work, the system could be hardened by:

- Incorporating **confidence thresholds**, only accepting object labels above a certain probability.
- Combining object detection with **location and RF features** (e.g., requiring that a "drone" label corresponds to an RF pattern consistent with a drone radio link).

11.3.2 Privacy and Data Protection

Images captured by take_photo.py may include people, vehicles, or other sensitive information. While the initial project focuses on technical feasibility rather than privacy regulation, there are still important security aspects:

- **Local processing:** Object detection runs locally on the Pi using an ONNX model. Images are not sent to external cloud services, reducing the attack surface and risk of data leakage.
- **Controlled retention:** The system uses the "most recent image" approach for the dashboard. In a deployment scenario, retention policies can be implemented (e.g., deleting older images after a fixed period or after ingestion into a secure backend).
- **File permissions and access control:** Limiting shell access to the Pi and restricting access to the images directory reduces unauthorized viewing or exfiltration.

If integrated with a MERN backend, secure transmission (HTTPS/TLS) and authentication would be required to prevent interception of images or metadata in transit.

11.3.3 Model Integrity and Supply Chain

The object detection model is a critical dependency. If an attacker replaced the ONNX model with a maliciously trained version, they could bias detections toward false negatives or false positives for particular objects.

Mitigations include:

- Storing the model file with restricted permissions and tracking its checksum.
- Maintaining a trusted path for model updates and documenting the source (e.g., specific Git commit or release).
- Optionally verifying model integrity at startup by computing and checking a hash before loading.

These supply-chain-oriented considerations help ensure that the object detection component itself has not been tampered with.

11.4 Security of FSOC Modulation (Raspberry Pi)

The FSOC modulation script, modulation.py, converts JSON metadata into an optical signal. This part of the system primarily raises security questions about confidentiality, integrity, and authenticity of the transmitted data.

11.4.1 Eavesdropping and Confidentiality

FSOC links naturally provide some **spatial security**: line-of-sight and optical power constraints limit who can observe the beam. However, an attacker positioned within the beam path (or intercepting reflections from nearby surfaces) could potentially detect the LED's on/off transitions and reconstruct the Manchester-encoded data.

In the current SD2 implementation, the content is not encrypted, and the JSON payload is effectively transmitted in clear text. This means:

- GPS coordinates, device labels, and timestamps could be recovered by a nearby eavesdropper with appropriate optical equipment.
- Knowledge of the preamble structure and bit period would simplify decoding.

Given that the project's primary focus is demonstrating functionality, this design is acceptable for a prototype but is not sufficient for deployment in environments where location or detection data must remain confidential.

Potential enhancements include:

- **Application-level encryption:** Encrypting the JSON payload before bit-level serialization using a symmetric key shared between the Pi and ESP32 (e.g., AES-GCM). The FSOC link would then transmit ciphertext instead of plaintext, and only the ESP32 with the key could recover the original data.
- **Frequency-domain obfuscation:** Modulating multiple LEDs at slightly different frequencies or using more complex coding could further thwart simple eavesdroppers.

11.4.2 Data Integrity and Replay Attacks

Because the FSOC frame consists of a preamble, Manchester-encoded payload, and terminator, an attacker could try to:

- **Inject false frames** by shining their own LED into the photodiode with appropriately shaped pulses.
- **Replay captured sequences** by recording an earlier frame and retransmitting it later to create the illusion of repeated detections.

The current design offers some partial protection:

- The **preamble pattern** and terminator ensure that random noise is unlikely to produce syntactically valid frames.
- The JSON parser on the ESP32 performs basic validation of fields, rejecting clearly malformed or nonsensical payloads.

Nevertheless, a determined attacker with knowledge of the protocol could still inject misleading but well-formed messages. Future versions can significantly strengthen integrity and authenticity by:

- Adding a **checksum or CRC** over the payload and verifying it at the receiver to detect random bit errors.
- Including a **monotonic counter or timestamp** in each frame, allowing the ESP32 to detect replayed messages (e.g., by rejecting any frame whose timestamp is significantly older than the current time or whose sequence number has already been processed).

- Applying a **Message Authentication Code (MAC)**, such as HMAC over the JSON payload, keyed with a secret shared between the Pi and ESP32. Only frames with valid MACs would be accepted, preventing attackers from forging messages without the key.

11.4.3 Robustness to Intentional Optical Interference

From an availability standpoint, an attacker could shine a bright light at the photodiode to overwhelm the analog front-end, causing the ADC to saturate or rendering the Manchester transitions unreadable. During such a jamming attack, the modulation scheme itself remains intact, but the physical channel is unusable.

Mitigations include:

- **Optical shielding:** Physically shrouding the photodiode to limit its field of view to the intended LED path, reducing susceptibility to off-axis light sources.
- **Optical filtering:** Using bandpass optical filters tuned to the LED's wavelength, which attenuate other light sources and make jamming more difficult.
- **Saturation detection:** Having the ESP32 monitor for persistently high or low ADC readings and flag "optical saturation" conditions rather than attempting to decode. This allows operators to distinguish between link failures due to hardware issues and possible interference.

11.5 Security of FSOC Demodulation (ESP32 Firmware)

The demodulation firmware on the ESP32 turns analog samples into JSON messages. Because it directly parses untrusted data derived from the environment, it must be written defensively.

11.5.1 Input Validation and Buffer Management

The firmware maintains a payload buffer where bytes between preamble and terminator are stored. Without proper limits, an adversarial optical pattern could:

- Cause the receiver to remain in READ_PAYLOAD state indefinitely, accumulating bytes until memory is exhausted.
- Trigger buffer overflows if payload length is not checked, leading to undefined behavior or code execution vulnerabilities.

The firmware should therefore:

- Enforce a maximum payload length, after which it discards the frame and returns to SEARCH_PREAMBLE.
- Use bounded arrays and explicit length checks when appending bytes to the buffer.
- Clear the buffer and reset state whenever framing is lost or errors are detected.

By constraining payload size, the firmware ensures that even malicious or malformed messages cannot damage the system beyond causing a single frame to be dropped.

11.5.2 JSON Parsing and Field Sanity Checks

After a complete frame is collected, the payload is interpreted as a UTF-8 string and parsed as JSON. Security considerations here include:

- **Malformed JSON:** Random noise or partial frames could produce invalid strings that cause the parser to fail. The firmware must handle parse errors gracefully and immediately return to SEARCH_PREAMBLE.
- **Unexpected fields or values:** Even if JSON parses successfully, the content may be nonsensical (e.g., latitude outside the valid range, negative frequencies, unknown device labels).

The project addresses this by:

- Wrapping JSON parsing in error handling that discards the payload on failure.
- Checking that required fields (e.g., lat, lon, center_frequency, device) are present and fall within plausible ranges.
- Treating any failure in these checks as grounds to reject the message.

These checks help ensure that only coherent and realistic messages are accepted as valid detections, mitigating the effects of occasional decoding errors or ad hoc optical interference.

11.5.3 State Machine Robustness

The two-state finite state machine (SEARCH_PREAMBLE and READ_PAYLOAD) is inherently simple, which is an advantage for security. However, there are still subtle edge cases:

- **Frequent transitions:** Rapid changes between states due to noise could leave the receiver in ambiguous conditions if transitions are not handled cleanly.
- **Stuck states:** If the firmware enters READ_PAYLOAD and never sees a terminator (for example, due to partial transmission), it could remain in this state indefinitely, ignoring new preambles.

The firmware mitigates these risks by:

- Resetting to SEARCH_PREAMBLE whenever payload length exceeds a configured threshold or when timing expectations are violated.
- Using a rolling bit or byte window to search for the preamble pattern, so that it can resynchronize even if it “wakes up” in the middle of a frame.

A well-designed state machine serves as a security mechanism: it constrains how inputs can influence firmware behavior and constrains error propagation.

11.5.4 Logging and Forensics

Having the ESP32 print decoded messages, parse errors, and framing anomalies over serial provides a valuable forensic trail. If suspicious behavior is suspected (e.g., repeated parse failures, unusually frequent preamble detections), investigators can review logs to infer whether the system was subjected to interference or attempted spoofing. This supports both debugging and security monitoring.

11.6 End-to-End Security and Defense-in-Depth

GWhile each subsystem has its own local protections, many security properties only emerge when the entire chain is considered end-to-end. Key points include:

- **Cross-checking RF and FSOC data:** The Pi writes JSON to metadata2.json and then transmits the same information over FSOC. If the ESP32 logs received JSON and these logs are periodically compared to the Pi's records, discrepancies can reveal tampering or transmission errors.
- **Correlating images and metadata:** Each detection is associated with an image and RF parameters. If an attacker tried to alter JSON at rest on the Pi, the mismatch between images, RF logs, and FSOC-received JSON could be detected.
- **Limiting trust in single components:** Object detection results are not trusted alone; they are always interpreted in context with RF detections and, if needed, operator judgment.

On the system side, standard hardening practices further enhance security:

- Running detection and orchestration scripts under a non-root user with limited permissions.
- Keeping SSH or remote access locked down with strong credentials or keys.
- Restricting network exposure of the Flask dashboard to the local network or a VPN segment.
- Using secure configuration of the MERN backend (TLS, authentication, input sanitization, and least-privilege database roles) when remote storage is enabled.

By layering these protections, no single failure—whether in RF detection, object detection, FSOC modulation, or demodulation—should be sufficient to entirely compromise the system's integrity.

11.7 Future Security Enhancements

The current design provides a solid baseline for a student project, but several enhancements could further improve security for real-world deployment:

- **Cryptographic protection on FSOC:** Implementing AES-GCM or a similar authenticated encryption scheme for the JSON payload would simultaneously address confidentiality and integrity over the optical link.

- **Digital signatures for detections:** When sending data to a MERN backend or other external systems, signing detection records with a key stored on the Pi would allow receivers to verify that records are authentic and unmodified.
- **Stronger authentication for remote dashboards:** Adding user accounts and role-based access control to the Flask or MERN interfaces would prevent unauthorized viewing or manipulation of detections.
- **Automated anomaly detection:** Monitoring patterns of RF detections, object labels, and FSOC errors over time could reveal anomalous behavior indicative of jamming or spoofing attacks.
- **Secure boot and filesystem encryption:** On the Raspberry Pi and ESP32, secure boot and encrypted storage could protect firmware and configuration files from tampering, especially if devices are physically accessible to attackers.

In summary, the project's software architecture already incorporates several security-relevant design choices—clear separation of responsibilities, rigorous input validation, structured logging, and well-defined data flows. By building on this foundation with cryptographic protection and additional operational controls, the RF detection and FSOC system can evolve into a robust platform that not only detects and reports devices but does so in a way that is resilient to adversarial conditions.

Chapter 12 - Conclusion

The RF Device Detection and Free-space Optical Communication (FSOC) System represents a comprehensive, interdisciplinary engineering solution designed to detect, contextualize, and transmit information about RF-emitting devices using a modular, hardware-software architecture. Throughout this report, we have detailed each element of the system, from RF sensing and signal processing on the Raspberry Pi 5 and ADALM-Pluto SDR, to object detection, optical modulation and demodulation, security considerations, and local/remote visualization. In this conclusion, we revisit our core goals, summarize how the implemented design meets those goals, and identify the broader implications and future directions for this platform.

12.1 Project Goals and Motivation

The core objective of this project is to provide a **portable, discreet, and modular** system that can detect RF-emitting devices—such as handheld radios, phones, and WiFi-based communication endpoints—then package and transmit this information through a Free-space Optical Communication link to a separate embedded platform. From the outset, our team focused on designing a system that could function as a self-contained "sensor node" rather than a lab-only prototype.

To achieve this, we deliberately selected **accessible, off-the-shelf components**: an ADALM-Pluto SDR for RF acquisition, a Raspberry Pi 5 for orchestration and local processing, a high-power LED and photodiode pair for the optical link, and an ESP32 microcontroller for the FSOC receiver. These parts strike a balance between cost, availability, and technical capability, making the design reproducible and extendable for future teams.

A second major motivation was to **decouple sensing from conventional RF backhaul**. Instead of sending detection data over WiFi, cellular, or other standard RF channels, the system uses an optical backchannel based on Manchester-coded LED pulses. This approach demonstrates that RF situational awareness can be paired with a low-signature, line-of-sight communication method, which is attractive in RF-congested, RF-denied, or covert monitoring environments. The requirement to operate in such conditions directly shaped the choices described in the software architecture, RF detection pipeline, modulation/demodulation logic, and security analysis.

12.2 System Architecture Recap

The implemented architecture consists of four tightly integrated subsystems that together form an end-to-end "sense → decide → communicate → visualize" chain:

1. **RF Sensing and Detection on Raspberry Pi 5**

The RF front-end is built around a single ADALM-Pluto SDR connected to the Raspberry Pi 5. Using the pyadi-iio library, the Pi configures the Pluto to alternate between a small set of carefully chosen bands: a narrowband slice around the FRS walkie-talkie channel at 462.6875 MHz and several wideband slices centered on 2.4 GHz WiFi channels (such as channels 1, 6, and 11).

The rf.py script continuously performs round-robin scanning, collecting I/Q samples, computing Welch power spectral densities or band power, and applying two classes of detectors: a CA-CFAR-style detector for narrowband FRS signals, and an energy-over-baseline detector for wideband WiFi activity. Each confirmed detection yields a compact RF metadata structure containing center frequency, detected peak, SNR, and device label, and triggers the higher-level pipeline through main.py.

2. **Imaging and Contextual Capture**

Whenever an RF event is confirmed, the Raspberry Pi invokes a camera script (e.g., take_photo.py) to capture an image of the environment near the time of detection. These images provide contextual evidence that can be reviewed by an operator, and optionally form input for an ONNX-based object detection model via new_inference.py. The result is that each RF detection can be paired with a visual snapshot and, when available, a vision-based label (e.g., “drone,” “phone,” or “router”), improving situational awareness beyond RF data alone.

3. **Free-space Optical Communication (FSOC) Backchannel**

Once RF and contextual data have been consolidated by extract_metadata.py into a single JSON record (metadata2.json), the system invokes modulation.py to transmit that record over an optical backchannel. The script converts the JSON string into bytes, serializes those bytes into bits, applies Manchester encoding, and frames the message with a preamble and terminator.

The Raspberry Pi uses the gpiozero library to toggle a high-power LED according to this bitstream, creating a robust, low-rate optical link. On the receiving end, an ESP32 continuously samples the photodiode output, compensates for ambient light, classifies half-bits using thresholding and hysteresis, decodes the Manchester stream, searches for the preamble, and reassembles the JSON payload. When successful, the same metadata record written on the Pi becomes available as a parsed JSON object on the microcontroller.

4. **Local Web Dashboard and Backend Integration**

For human operators, a lightweight Flask application (app.py and index.html) runs on the Raspberry Pi. The Flask server reads metadata2.json, locates the most recent image in the images/ directory, and presents both via an HTML dashboard labeled “RF Detection Results.” The dashboard shows the device label, frequencies, GPS coordinates (when available), and timestamps in a clear “Detection Details” card alongside the captured image.

The same JSON format is suitable for integration with a MERN-stack backend, where detections can be stored in MongoDB and viewed via a more feature-rich web or mobile interface. This optional backend path provides a natural avenue for scaling the system from a single sensor node to a network of nodes with centralized analytics.

Together, these subsystems form a cohesive architecture: RF detection on the Pluto/Pi identifies relevant activity, camera and metadata scripts provide context, FSOC transports the resulting record to a separate embedded platform, and the dashboard (plus optional MERN backend) exposes the data to operators and analysts. The modular JSON-based interfaces between components allow each stage to be developed and debugged independently while still participating in the full pipeline.

12.3 Security-Driven Design Philosophy

A recurring theme throughout this project is the recognition that **signal detection without security is incomplete**. Even in a student design context, it is important to consider how RF detection, object detection, and FSOC communication could be misled, jammed, or otherwise manipulated.

At the RF layer, we moved from simple absolute thresholds toward **r**elative, noise-aware detectors, such as CA-CFAR for FRS and energy-over-baseline methods for WiFi. This provides resilience against slow changes in the noise floor, moderate interference, and environmental variability. The concept of a five-meter "detection bubble" informed gain and threshold tuning, emphasizing controllable sensitivity rather than maximum possible range.

In the FSOC path, the system uses deterministic framing (preamble and terminator), Manchester encoding, and JSON validation on the ESP32 to detect malformed or corrupted messages. The demodulation firmware treats the optical channel as untrusted: it enforces payload length limits, performs structured parsing, and discards frames that do not meet basic sanity checks on fields such as latitude, longitude, or frequency.

The security considerations chapter extended this analysis by outlining potential attacks—such as RF spoofing, optical jamming, and data tampering—and by proposing practical mitigation strategies and future enhancements. These include the possibility of adding cryptographic protection for FSOC payloads (e.g., authenticated encryption or message authentication codes), more advanced checks on detection patterns to identify jamming, and stronger access control and data protection mechanisms for any remote backend.

While not all advanced countermeasures are implemented in the current build, the architecture and data flows were intentionally designed with security in mind: clear module boundaries, well-defined JSON schemas, explicit logging, and defensive parsing all serve as building blocks for a more hardened version of this system.

12.4 Design Milestones and Testing Roadmap

Across the Senior Design sequence, the project progressed from concept and simulation to an integrated, working prototype:

- **Component Selection and Prototyping**

Early work focused on selecting the ADALM-Pluto SDR, Raspberry Pi 5, ESP32, LED driver board, high-power LED, and photodiode, and then validating basic connectivity. Initial experiments confirmed that the Pluto could capture I/Q samples at both FRS and 2.4 GHz bands and that the Pi could successfully control the LED via GPIO.

- **RF Detection Algorithm Development**

The team implemented and iteratively refined the RF detection logic in rf.py. Baseline-plus-offset methods were gradually replaced with more principled techniques (CA-CFAR-like detection for FRS and adaptive energy-over-baseline for WiFi), focusing on achieving a practical detection

bubble of roughly five meters around the antenna. Testing included observing walkie-talkie transmissions and phone-based hotspots at different distances and adjusting thresholds and gains to balance sensitivity and robustness.

- **FSOC Modulation and Demodulation**

On the optical side, we verified that Manchester-coded LED patterns generated by modulation.py could be reliably decoded by the ESP32 firmware under realistic lab conditions. This required tuning bit periods, threshold margins, and hysteresis logic to compensate for ambient light, ADC noise, and non-deterministic timing on the Pi. Successful round-trip tests demonstrated that a JSON payload written on the Pi could be reconstructed as a parsed JSON object on the microcontroller.

- **Orchestration and Dashboard Integration**

With RF detection and FSOC functioning, attention shifted to main.py, extract_metadata.py, and the Flask dashboard. The team implemented the end-to-end workflow: RF detection triggers image capture, metadata extraction merges RF/vision/position data into metadata2.json, modulation.py transmits the record, and the Flask app presents the latest detection and image in the web UI. Test runs confirmed that the pipeline behaves as expected in normal conditions and fails gracefully when components (e.g., camera or GPS) are unavailable.

- **Security Assessment and Hardening Steps**

Finally, the security considerations chapter formalized the threat model and catalogued countermeasures. While full cryptographic hardening and enterprise-grade backend security are beyond the scope of a single semester, the project now includes clear recommendations for future teams on how to evolve the prototype into a more secure, field-ready system.

These milestones collectively validate that the **core concept is technically feasible**: RF events can be detected with controllable sensitivity, contextualized with images and metadata, carried over a free-space optical link, and presented to human operators in a clear and modular way.

12.5 Broader Implications and Future Work

The relevance of this work extends beyond the specific hardware used. Conceptually, the project demonstrates a reusable pattern for **modular sensing nodes**:

1. Use a software-defined front-end (SDR, camera, or other sensors) to detect events.
2. Convert those events into a structured, JSON-style record.
3. Transmit that record over a channel that matches operational constraints (in this case, a low-signature FSOC link).
4. Expose the resulting data to operators and higher-level systems via standard web and API interfaces.

In practice, a platform like this can serve as a building block for:

- **Tactical monitoring and electronic support** in RF-congested or contested environments.
- **Infrastructure and perimeter monitoring**, where RF activity near critical assets (e.g., unauthorized hotspots or drones) must be logged and investigated.

- **Research and education**, giving students a hands-on environment for exploring SDR, optical communication, embedded systems, and security concepts in an integrated manner.

Future work can expand the system along several axes:

- **Scaling and Networking**

Deploying multiple sensor nodes and aggregating detections into a centralized MERN backend would enable geospatial correlation, triangulation, and richer analytics. Nodes could share information via wired, wireless, or optical links depending on mission requirements.

- **Enhanced Security and Cryptography**

Implementing authenticated encryption on the FSOC link, cryptographic signatures on detection records, and hardened authentication/authorization for dashboards would significantly raise the bar against tampering and eavesdropping.

- **Advanced Signal and Object Analytics**

Additional DSP techniques (e.g., modulation classification, time-frequency analysis) and more sophisticated object detection or tracking models could improve classification accuracy and reduce false alarms. Over time, the system could incorporate learning-based methods that adapt thresholds and detection policies based on observed patterns.

- **Mechanical and Environmental Hardening**

Further work on enclosures, optical alignment, and environmental protection (e.g., weatherproofing, vibration tolerance) would be required for outdoor or long-term field deployments.

By clearly separating what has been implemented from what is envisioned, the project lays a realistic foundation while still pointing toward a roadmap for future improvement.

12.6 Final Reflections

This project has provided a unique opportunity to integrate concepts from RF engineering, digital signal processing, optical communication, embedded systems, web development, and cybersecurity into a single, cohesive system. Starting from a high-level concept of "detect RF devices and send the information optically," the team progressed through component selection, algorithm development, software architecture design, debugging, and validation.

Key lessons include the importance of modularity (small, well-defined scripts and firmware components are easier to reason about and test), observability (structured logging and JSON records greatly simplify debugging and analysis), and defensive design (treating external inputs—from RF signals to optical pulses—as untrusted and validating them carefully).

As we transition from the design and implementation phase into extended evaluation and refinement, our focus remains on making the system as robust, explainable, and extensible as possible. With continued iteration, additional security enhancements, and potential deployment in more realistic environments, this RF detection and FSOC platform can serve as a practical, modern example of how interdisciplinary engineering can address emerging challenges in spectrum awareness and secure communications.

Appendix A - References

[1] U.S. Army TRADOC, "Smart Phones Playing Prominent Role in Russia-Ukraine War," *Operational Environment Center*, Jul. 21, 2023. [Online]. Available: https://oe.tradoc.army.mil/product/smart-phones-playing-prominent-role-in-russia-ukraine-war/. [Accessed: Jun. 1, 2025].

[2] Reuters, "Russian lawmakers seek punishment for troops using smartphones in Ukraine war," *Reuters*, Jul. 23, 2024. [Online]. Available: https://www.reuters.com/world/europe/russian-lawmakers-seek-punishment-troops-using-smartphones-ukraine-war-2024-07-23/. [Accessed: Jun. 1, 2025].

[3] Rohde & Schwarz, *An Introduction to Passive Radar Systems*. [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/premiumdownloads/premium_dl_pdm_downloads/3684_0182_52/An-introduction-to-passive-radar-systems_wp_en_3684-0182-52_v0100.pdf. [Accessed: Jun. 1, 2025].

[4] Defense Advancement, "Military-Grade LiFi for Classified Environments," *Defense Advancement*, Oct. 2, 2023. [Online]. Available: https://www.defenseadvancement.com/news/military-grade-lifi-for-classified-environments/. [Accessed: Jun. 1, 2025].

[5] M. Bertorelli, "Search and Rescue: Cell Phone Tracking," *AVweb*, Oct. 25, 2022. [Online]. Available: https://avweb.com/flight-safety/search-and-rescue-cell-phone-tracking/. [Accessed: Jun. 1, 2025].

[6] CRFS, "RF Border Surveillance: The First Line of Defense for Border Security Monitoring," *CRFS Blog*, Mar. 8, 2023. [Online]. Available: https://www.crfs.com/blog/rf-border-surveillance-the-first-line-of-defense-for-border-security-monitoring/. [Accessed: Jun. 1, 2025].

[7] U.S. Government Accountability Office, "Drone Threats: FAA Is Evaluating Counter-UAS Detection and Mitigation Technologies," *GAO-24-107195*, Mar. 7, 2024. [Online]. Available: https://www.gao.gov/products/gao-24-107195. [Accessed: Jun. 1, 2025].

[8] Redmon, Joseph and Farhadi, Ali YOLOv3: An Incremental Improvement. (2018), https://doi.org/10.48550/arXiv.1804.02767

[9] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**, 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94

[10] OpenCV “Introduction to SIFT (Scale-Invariant Feature Transform)”, https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html?utm_source=chatgpt.com

[11] PYNQ-Z2 Reference Manual v1.0 https://www.mouser.com/datasheet/2/744/pynqz2_user_manual_v1_0-1525725.pdf

[12] YOLO-on-PYNQ-Z2, https://github.com/andre1araujo/YOLO-on-PYNQ-Z2

[13] “Spectrum Analyzers: Liberty Test Equipment.” *Shop Test and Measurement Equipment*, libertytest.com/spectrum-analyzers.html. Accessed 7 July 2025.

[14] “What Is Software-Defined Radio (SDR)?” *What Is Software-Defined Radio (SDR)? - MATLAB & Simulink*, www.mathworks.com/discovery/sdr.html. Accessed 7 July 2025.

[15] “2.4 GHz vs. 5 GHz vs. 6 GHz: What’s the Difference?” *Intel*, www.intel.com/content/www/us/en/products/docs/wireless/2-4-vs-5ghz.html. Accessed 7 July 2025.

[16] Zhang, Kenny. “Walkie-Talkie Frequencies in the US.” *Herda Radio*, 24 Mar. 2023, herdaradio.com/blog/radioknowledge/walkie-talkie-frequencies-in-the-us/?srsltid=AfmBOopSkWbDAuSHK3j56kMThVqhviZ0HlyrpRyRg68fS8os7hP48zpi. Accessed 07 July 2025.

[17] Power Queen, “12V 50Ah LiFePO4 Battery,” *Power Queen Official*, 2025. [Online]. Available: https://ipowerqueen.com/products/power-queen-12v-50ah-lifepo4-battery. [Accessed: Jul. 13, 2025].

[18] Dakota Lithium, “12V 54Ah Deep Cycle LiFePO4 Battery,” *Dakota Lithium*, 2025. [Online]. Available: https://dakotalithium.com/product/dakota-lithium-12v-54ah-deep-cycle-lifepo4-trolling-motor-battery/. [Accessed: Jul. 13, 2025].

[19] BOTKU, “Rechargeable 12.8 V $LiFePO_4$ Battery – BOTKU (10 Ah),” Amazon, [Online]. Available: https://www.amazon.com/BOTKU-Rechargeable-Phosphate-Lighting-Applications/dp/B0D3CCR2GS/. [Accessed: Jul. 13, 2025].

[20] Grepow, “Leading Provider of Custom Lithium Battery Pack Solution,” Grepow, [Online]. Available: https://www.grepow.com/. [Accessed: Jul. 13, 2025].

[21] Syu, Yong-Sin, and Yung-Chun Lee. 2022. "Quantitative Evaluation of Light Collimating for Commercial UV-LEDs Based on Analytic Collimating Lens" Applied Sciences 12, no. 2: 911. https://doi.org/10.3390/app12020911 [Accessed: Jul. 18, 2025].

[22] J. Zidar, I. Aleksi and T. Matić, "Analysis of energy consumption for SPI and I2C communications in ultra-low power embedded systems," *2023 46th MIPRO ICT and Electronics*

Convention (MIPRO), Opatija, Croatia, 2023, pp. 213-217, doi: 10.23919/MIPRO57284.2023.10159889.

[23] Garrett Yamasaki. 2023. “Two Ways to Save Power with Low-power Ethernet”, Texas Instruments. Available: https://www.ti.com/document-viewer/lit/html/SSZTAM3

[24] “AN-914 Understanding Power Requirements in RS-232 Power Applications”, Texas Instruments. 1993, revised 2013. Available: https://www.ti.com/lit/an/snla037b/snla037b.pdf?ts=1752808921086