

Smart Predator Detection and Fence Monitoring System



Prepared by Group 18:

Imaan Shaik - SHKIMA004

Safiya Mia - MXXSAF002

Triss Naidoo - NDXTRI021

Prepared for:

EEE4113F

Department of Electrical Engineering

University of Cape Town

May 25, 2025

Declaration

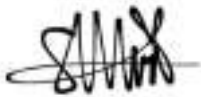
1. We know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. We have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is our own work.
4. We have not allowed, and will not allow, anyone to copy our work with the intention of passing it off as their own work or part thereof.



May 25, 2025

Imaan Shaik


Date



May 25, 2025

Safiya Mia

Date



May 25, 2025

Triss Naidoo

Date

Contents

Contents	iii
List of Figures	vii
List of Tables	ix
Abbreviations	xi
1 Introduction	1
1.1 Problem Statement	1
1.2 Problem Analysis	2
1.3 System Requirements	2
1.4 Scope & Limitations	2
1.5 Report Outline	2
2 Problem Analysis	3
2.1 D-school Activities	3
2.2 Design Choices	3
2.3 Subsystem Description	4
2.3.1 Predator Detection and Housing - Imaan	4
2.3.2 Camera Imaging and Predator Identification - Triss	4
2.3.3 Front-end - Safiya	4
2.4 Problem Statement	4
3 Literature Review	5
3.1 Introduction	5
3.2 Overview of Existing Predator Deterrent Strategies	5
3.2.1 Physical Barriers	5
3.2.2 Sensory-Based Deterrents	6
3.2.3 Chemical deterrents	7
3.2.4 Projectile Deterrents	7
3.2.5 Guardian Dogs	8
3.2.6 Automated Detection & Response Systems	8
3.3 Effectiveness of These Strategies in the Context of Honey Badgers & Penguin Colonies	8
3.3.1 Strengths & Weaknesses of Physical Barriers	8

3.3.2	Strengths & Weaknesses of Sensory Deterrents	9
3.3.3	Strengths & Weaknesses of Chemical Deterrents	9
3.3.4	Strengths & Weaknesses of Projectile Deterrents	10
3.3.5	Strengths & Weaknesses of Guardian Dogs	10
3.3.6	Feasibility of AI-Based Systems in Remote Areas	10
3.4	Artificial Neural Network (ANN)s	10
3.4.1	Convolutional Neural Network (CNN)s	11
3.4.2	Densely Connected Convolutional Network (DenseNet)	11
3.4.3	Vision Transformers (ViTs)	11
3.4.4	Multi-Layer Perceptron (MLP)s	12
3.4.5	Hybrid Models	12
3.5	Dataset Augmentation	12
3.5.1	Simple Transformations	12
3.5.2	Colour Adjustments	12
3.5.3	Introducing Noise	12
3.6	Data Pre-processing	12
3.6.1	De-noising	12
3.6.2	Normalization	13
3.7	Conclusions from the Literature	13
4	Predator Detection Subsystem By: Imaan Shaik - SHKIMA004	14
4.1	Predator Detection Subsystem Overview	14
4.2	Requirements and Specifications	14
4.2.1	User requirements	14
4.2.2	Functional Requirements	15
4.3	Design Rationale	15
4.3.1	Sensor Selection and Justification	15
4.3.2	Microcontroller Selection	15
4.3.3	Power Regulation	16
4.4	Predator Confirmation Logic	16
4.4.1	Primary Detection – PIR Sensor	16
4.4.2	False Trigger Mitigation	16
4.4.3	Secondary Detection – Laser Break-Beam	17
4.4.4	Camera Trigger Communication Protocol	17
4.5	Implementation and Testing	17
4.5.1	Detection Logic	17
4.5.2	Performance of PIR and Laser Modules	18
4.5.3	Designing a laser receiver circuit	18
4.5.4	Hardware Filtering the Laser Rx Module	18
4.6	Acceptance Testing, Results and Analysis	19
4.7	Design Trade-offs and Final Outcome	20
4.8	Recommendations	20
5	Housing Subsystem	21

5.1	Housing Subsystem Introduction	21
5.2	Requirements and Specifications	21
5.2.1	User Requirements	21
5.2.2	Functional Requirements	22
5.3	Design choices	22
5.3.1	Material Selection	22
5.3.2	Internal Cooling Techniques	23
5.4	Submodule Design and Implementation	23
5.4.1	Compartmentalization and Ensuring Component Function	23
5.4.2	Thermal Monitoring and Passive Cooling	24
5.5	Acceptance Testing, Results and Analysis	26
5.6	Conclusion	27
5.7	Deliverables	27
6	Front-end Subsystem	28
6.1	Introduction	28
6.2	Scope and Limitations	28
6.2.1	Scope	28
6.2.2	Limitations	28
6.3	Requirement Analysis	29
6.3.1	Non-functional Requirements, Specifications, and Acceptance Tests	29
6.3.2	Functional Requirements, Specifications, and Acceptance Tests	29
6.4	Back-End Design and Theory	30
6.4.1	Selection of Processor and Hosting Platform	30
6.4.2	Implementation	31
6.4.3	Exposing the Flask Sever to the Internet	33
6.5	Front-End Design and Theory	34
6.5.1	Styling	34
6.6	Implementing User Interactivity	35
6.6.1	Login Prompt and Access Control	35
6.6.2	Download Functionality	36
6.6.3	Chart Rendering with <code>Chart.js</code>	36
6.6.4	Filtering Mechanisms on Log and Analytics Pages	37
6.7	Results and Acceptance Testing	38
6.8	Conclusion	38
7	Imaging Subsystem	39
7.1	Subsystem introduction	39
7.2	Requirements Analysis	39
7.2.1	User Requirements	39
7.2.2	Functional Requirements	40
7.2.3	Specifications	40
7.2.4	Acceptance Testing Procedures	41
7.2.5	Traceability Matrix	41

7.3	Design choices	42
7.3.1	Hardware Design	42
7.3.2	Buck Converter	42
7.3.3	IR LEDs	43
7.3.4	Software Design	43
7.4	Implementation	43
7.4.1	Hardware Design	43
7.4.2	Subsystem Triggering	44
7.4.3	IR LED triggering	45
7.4.4	Camera image and video recording	45
7.4.5	45
7.5	Testing	45
7.6	Conclusion	45
8	Conclusions	46
	Bibliography	47
A	Appendix	49
A.1	GA Criteria - Imaan Shaik	49
A.1.1	Predator Detection and Housing Subsystems	49
A.2	GitHub Repo	49
A.3	Predator Detection System	51
A.4	Designing a laser receiver circuit to replace the module	52
A.5	Testing camera trigger implementation	53
A.6	Hardware filtering the Laser Rx Module	53
A.7	Housing System	54
A.7.1	Enclosure Fabrication	54
A.7.2	Thermal Management System	55
A.7.3	Testing and Subsystem Integration	56
B	Front-End Subsystem	58
B.0.1	GA Criteria	58
B.0.2	Pictures from Testing	59

List of Figures

4.1	KiCad schematic of the designed laser receiver circuit	18
4.2	Serial monitor output showing improved signal stability and clean trigger detection after RC filtering	19
5.1	CAD renderings of the enclosure system showing internal component arrangement, external laser alignment, and heatsink placement.	25
6.1	Branding elements used in the front-end interface design	34
6.2	Index page showing the password prompt	35
6.3	Download button successfully triggering a media file download, confirming correct route setup and file access from the user interface.	36
6.4	Screenshots of the analytics page showcasing two key visualisations used to interpret detection data. Both views include interactive filters for species and time range selection.	37
7.1	IR LED Layout	44
7.2	RPi Zero 2 W GPIO Configuration	45
A.1	Breadboard prototype of the receiver circuit with phototransistor, LDR, and LM393 comparator and LM324 op-amp.	52
A.2	Full system integration test with ESP32, Raspberry Pi, Pi camera, PIR, laser transmitter and receiver modules	53
A.3	ESP32 with PIR and Laser filtering circuits soldered on a veroboard.	53
A.4	Breadboard test of low pass filtered laser module	53
A.5	Scalability concept for the laser detection subsystem. Multiple laser beams can be emitted from a single enclosure (reflected off mirrors) to detect motion at various heights and positions. This concept aims to improve detection reliability against agile predators like honey badgers.	54
A.6	Design and physical build stages of the Perspex enclosure.	54
A.7	Passive heatsink cooling system embedded in rear wall.	55
A.8	Subsystem testing and functional verification.	56

A.9	Scalability of the enclosure system along the perimeter fence. Photos provided by BirdLife Stakeholder slides. These images depict the different possible employments of the system, showing its adaptability. It would likely be deployed at a height near the ground to detect honeybadgers. Top-left: Real deployment at De Hoop Nature Reserve. Top-right: Detection of honey badger at night. Bottom-left: Linear alignment of multiple enclosures using a laser break-beam. Bottom-right: Close-up view of enclosure pair with laser beam.	57
B.1	Responsive design of the dashboard interface, showing key system stats and recent detections. The layout adapts seamlessly between mobile and desktop views.	59
B.2	Screenshots of the Animal Log interface displaying detection records, metadata, and download options.	59
B.5	Snapshot of the manually entered data within detection.db used for testing	60
B.3	Error message displayed when an incorrect password is entered on the index page/login screen.	60
B.4	CPU usage snapshot from the top command while the system is idle.	60
B.6	User feedback on interface responsiveness, clarity, and usability.	60

List of Tables

4.1	Traceability Matrix: User Requirements, Specifications, and Acceptance Criteria . . .	14
4.2	Traceability Matrix: Functional Requirements, Specifications, and Acceptance Criteria	15
4.3	Comparison of Microcontroller Platforms	16
4.4	Infrared Sensor Comparison	16
4.5	UART vs I ² C for Camera Triggering [1]	17
4.6	Observed receiver performance before and after RC low-pass filter implementation . .	19
4.7	Acceptance Test Procedures (ATPs) for the Sensing Subsystem	20
5.1	Traceability Matrix: User Requirements, Specifications and Acceptance Criteria	21
5.2	Traceability Matrix: Functional Requirements, Specifications and Acceptance Criteria	22
5.3	Material Comparison for Enclosure Design [2]	22
5.4	Critical Dimensions to account for in Enclosure Design	24
5.5	Acceptance Test Procedures (ATPs) for the Housing Subsystem	27
6.1	Non-functional Requirements, Specification, and Acceptance Test	29
6.2	Functional Requirements, Specification, and Acceptance Test	30
6.3	Comparison of Processor Options for Hosting the Front-End Subsystem	30
6.4	Comparison of Web Frameworks for the Front-End Server	31
6.5	Comparison of Storage and Database Options for Detection Data	32
6.6	Acceptance Tests with Results	38
7.1	Imaging Subsystem User Requirements	39
7.2	Imaging Subsystem Functional Requirements	40
7.3	Imaging Subsection Specifications	40
7.4	Acceptance Test Procedures for Imaging Subsystem	41
7.5	Traceability Matrix for Imaging Subsystem	41
7.6	Microcontroller Comparison	42
7.7	DC-DC Converters Comparison	42
7.8	Camera Module Comparison	42
7.9	IR LED Comparison	43
7.10	Acceptance Test Procedures for Imaging Subsystem	45
A.1	GA Criteria summary for SHKIMA004 (Imaan Shaik) — Predator Detection and Housing	49
A.2	Bill of Materials	50
A.3	Comparison of Different Motion Sensors: PIR, Ultrasonic, and Radar [3]	51

A.4	PIR Signal Conditioning Techniques Implemented in Software	51
A.5	Comparison of Predator Detection Strategies: PIR, Laser, and Combined	52
A.6	UART pin mapping between ESP32 and Raspberry Pi	52
B.1	GA Criteria Mapping for Front-End Subsystem (Safiya Mia)	58

Abbreviations

Chapter 1

Introduction

This report explores the intersection of engineering and conservation in a real-world setting: the protection of endangered African penguins at the De Hoop Nature Reserve in South Africa. These penguins, already facing population decline, are under increasing threat from terrestrial predators—most notably the resilient and relentless honey badger.

BirdLife South Africa, represented by stakeholder Christina, has highlighted the inadequacies of current protective measures. While an electric fence encircles the colony, it is compromised by poor ground coverage and structural vulnerabilities. The electrification is limited to the upper section, leaving the base exposed to burrowing predators. Reinforcements and overlays have been applied in critical areas, and motion-activated cameras have been installed to monitor activity [4]. Despite these efforts, predators continue to infiltrate the area.

Traditional deterrents such as flashing lights, alarms, or scent-based repellents offer only short-term relief. Predators like honey badgers rapidly adapt and learn to ignore static systems that pose no real threat. This calls for a smarter, more adaptable approach—one that not only detects but actively responds to intrusion attempts, deterring predators without causing harm to the ecosystem.

The proposed solution is a modular, scalable predator detection and deterrent system that leverages sensors, software filtering, and non-lethal countermeasures to identify and repel threats. It is designed to be ecologically safe, intelligent enough to handle predator habituation, and robust enough for deployment in harsh outdoor conditions. This project demonstrates how thoughtful engineering can address urgent environmental challenges while respecting the delicate balance of natural ecosystems.

1.1 Problem Statement

Predators such as honey badgers continue to breach the colony perimeter and prey on endangered African penguins. Despite existing fencing, vulnerabilities remain unaddressed. A system is needed to monitor the fence in real-time, detect intrusions, and provide stakeholders with actionable data on predator type, frequency, location, and fence condition.

1.2 Problem Analysis

1.3 System Requirements

1.4 Scope & Limitations

1.5 Report Outline

Chapter 2

Problem Analysis

2.1 D-school Activities

The D-School program consisted of weekly four hour sessions held over four weeks, introducing the core principles of design thinking. These sessions allowed project members to collaborate through various design-oriented exercises that fostered empathy, creativity, and structured problem-solving.

The first session focused on understanding potential stakeholders. Students applied design thinking techniques to uncover key information about stakeholder challenges and created stakeholder maps. Interviews with prospective stakeholders were conducted to gain deeper insight into their needs and pain points.

In the second session, students developed a problem statement tailored to a chosen stakeholder. This process involved asking critical questions such as: "Who are our stakeholders?", "What is their goal?", and "Why do they need to achieve it?" Based on these reflections, each group formulated a focused problem statement. This session also included group-based brainstorming, during which teams proposed multiple design ideas, selected a final concept, and assigned individual responsibilities for each subsystem.

The third session introduced the structure, methodology, and expectations of the literature review. Each group member was tasked with writing a short review and collaboratively presenting their findings.

The final session expanded on the literature review and provided a comprehensive overview of the project deliverables. Each group formally presented their literature reviews to course staff and peers for feedback and evaluation.

2.2 Design Choices

The initial stages of the Design School involved brainstorming a wide range of ideas that students believed could address the stakeholders' challenges. However, only after progressing through the D-School process, which included engaging with stakeholders and conducting a literature review, were students able to make informed design decisions. This process deepened their understanding of the problem and helped align their solutions with the stakeholders' key requirements, while reinforcing

core principles of design thinking.

2.3 Subsystem Description

2.3.1 Predator Detection and Housing - Imaan

This subsystem is responsible for providing reliable detections of predators and triggering capture subsystem, in order for images or videos to be captured. The subsystem also designs and implements a durable, environmentally friendly enclosure for all subsystems, to promote a durable and scalable design.

2.3.2 Camera Imaging and Predator Identification - Triss

This subsystem is primarily responsible for capturing and storing images or videos of the predators. It receives a trigger from the detection system, and performs machine learning to identify the type of predator captures. It also facilitates data transferal to the front-end system.

2.3.3 Front-end - Safiya

This subsystem provides conservationists with a remote-accessible interface for viewing data captured by the camera subsystem. It displays images, videos, and metadata, and includes analytics to help understand predator behaviour along the fence.

2.4 Problem Statement

Predators such as honey badgers continue to breach the colony perimeter and prey on endangered African penguins. Despite existing fencing, vulnerabilities remain unaddressed. A system is needed to monitor the fence in real-time, detect intrusions, and provide stakeholders with actionable data on predator type, frequency, location, and fence condition.

Chapter 3

Literature Review

3.1 Introduction

Predator management in wildlife conservation is an ongoing challenge, particularly when the goal is to protect vulnerable species, such as the African penguin. In the context of remote conservation areas, the presence of various predators poses a significant threat to the survival of African penguin colonies. One such critical breeding ground is the De Hoop Nature Reserve, a protected coastal area that serves as an important habitat for the species.

Among the most prominent threats are species such as baboons, leopards, caracals, and honey badgers. Each of these predators exhibits distinct behaviors that require targeted deterrent strategies to mitigate their impact. However, of these various threats, the honey badger is by far the most difficult to deter. Known for its incredible resilience and its tenacious nature, the honey badger is able to persist in the face of barriers that might deter other species.

In addressing the challenge of protecting penguins from predators, particularly honey badgers, in remote conservation areas, several non-lethal deterrent systems have been explored in the literature. Effective deterrents must not only protect the penguin species but also avoid harming non-target wildlife and the surrounding ecosystem.

3.2 Overview of Existing Predator Deterrent Strategies

The methods explored for mitigating predator damage can be broadly categorised into physical barriers, sensory-based deterrents, chemical deterrents, projectile deterrents, guardian dogs, and automated detection and response systems. A critical part of these studies is understanding the behaviour of the targeted predators and selecting appropriate deterrents that minimise harm to both the predators and the environment.

3.2.1 Physical Barriers

Physical barriers such as fences and enclosures are commonly used to prevent predator access to vulnerable species.

Poole et al. [5] demonstrate the effectiveness of an electric strained-wire fence to exclude badgers (*Meles meles*) from foraging areas, including crops like maize. Their study reveals that electric fences reduce crop damage by as much as 95%. While the fence design was successful in preventing badger incursions, it required frequent maintenance to maintain its effectiveness, as vegetation and weather conditions could interfere with its electrical output.

Furthermore, Poole et al. [5] points out that the fence design must be adaptable to different environmental conditions, such as the challenges presented by larger, more resilient predators like the honey badger (*Mellivora capensis*).

In contrast, Chwalibog [6] focuses on the difficulties of using physical barriers to manage honey badgers in South Africa, highlighting the challenges that arise from their digging and climbing behaviours. These traits make honey badgers more difficult to contain using traditional physical barriers. Nevertheless, Chwalibog [6] suggests that while electric fences can be effective for other wildlife, the adaptability and persistence of honey badgers often necessitate the use of additional measures, such as raised structures or reinforced barriers, to prevent access.

The use of reinforced structures to deter honey badgers has also been explored by Johnson [7], who examined the use of various physical deterrents, including metal sheeting and wire, to prevent honey badgers from raiding beehives in Kenya. In his study, Johnson [7] observed that although these methods provided some degree of protection, they were not always foolproof. The honey badgers' remarkable physical resilience allowed them to bypass many of these barriers, particularly when the deterrents were improperly installed. For example, the metal sheeting proved effective only if the posts were securely fixed and the barriers were continuous, but any slight misalignment or gaps allowed the honey badgers to manoeuvre around the obstacles. Similarly, Johnson [7] noted that while reinforced beehive fences reduced the success rate of honey badger attacks significantly, there was still a considerable rate of damage when barriers were not properly maintained.

3.2.2 Sensory-Based Deterrents

Sensory-based deterrents use stimuli such as sound, light, and motion to scare away predators by disrupting their normal behaviour. Johnson [7] investigated the impact of motion-activated lights and cone baffles in reducing honey badger predation on beehives. His findings indicated that introducing visual deterrents such as motion-activated lights reduced the rate of hive absconding from 77.1% to 11.1%. Johnson posited that the fear of detection was a significant motivator for the honey badger's avoidance behaviour, which aligned with other studies showing that predators avoid areas where they feel threatened. However, he also noted that the efficacy of these deterrents was compromised over time as the animals habituated to the stimuli.

Clouse [8] expanded on the idea of sensory deterrents by testing multimodal scarecrows, which use a combination of flashing lights and randomised sounds to deter nocturnal pests such as bushpigs and porcupines. His study demonstrated that this combination of stimuli significantly reduced feeding time, with bushpigs halving their foraging duration after activation. Additionally, the scarecrow triggered more running behaviour, indicating that the animals perceived the stimuli as a threat. Clouse's findings suggest that combining multiple sensory cues enhances deterrence, particularly for species that might habituate to a single type of stimulus, as seen in Johnson's work with honey badgers.

According to Mishra et al. [9], IoT-based animal deterrents have been developed to leverage real-time environmental data, such as motion detection, to activate visual or auditory signals. These devices can adapt to changes in an animal's behaviour, improving long-term deterrence. For example, the application of ultrasonic sounds tailored to specific animals, like honey badgers, has demonstrated varying levels of success. However, ultrasonic deterrents have shown limited effectiveness against mammals, as many species quickly become desensitised to the sound [10].

Gilsdorf et al. [10] examined the effectiveness of frightening devices incorporating bioacoustic and visual cues, such as strobe lights, reflective tape, and distress calls. Their study found that these deterrents were most effective when used in an integrated approach, reducing the frequency of predator visits. Motion-activated lights have been shown to be particularly effective in deterring nocturnal predators, including honey badgers, as they disrupt normal behaviour patterns by altering environmental lighting. However, ultrasonic devices, which emit high-frequency sounds designed to repel specific animals, have demonstrated inconsistent results. Gilsdorf et al. [10] noted that many mammals, including foxes and raccoons, quickly became desensitised to ultrasonic deterrents, significantly reducing their long-term efficacy.

Similarly, traditional deterrents failed to prevent predation in cases where predator motivation was high. For instance, studies on predation management for Little Penguins found that ultrasonic deterrents did not significantly reduce fox attacks, suggesting that such devices may be ineffective in high-risk situations [11]. Furthermore, while bioacoustic deterrents—such as alarm or distress calls—can provide a targeted response, they require a precise understanding of animal communication to ensure that the signals elicit the intended behavioural reaction [10].

Kommasani et al. [12] explored the integration of motion sensors and cameras into deterrent systems, allowing for more targeted responses. When an animal is detected, the system can activate a sequence of deterrents, such as triggering lights, sounds, or sending alerts to conservationists. This aligns with the broader trend of dynamic deterrents, which are crucial for addressing adaptive predators that become accustomed to static stimuli over time.

3.2.3 Chemical deterrents

Sometimes chemicals are used in order to deter animals from entering certain locations or engaging in certain behaviours. Capsicum-based sprays, ammonia, and other chemicals can be used to create significant discomfort such that an animal does not enter an area. Capsicum might also be sprayed onto crops/objects that one does not want animals to eat, or at the animal to scare it off.

Chemical deterrents can also be used in the application of aversive conditioning. In this process, animals are taught to associate the undesired behaviour with negative tastes or odours. Predator urine may be used to elicit fear in animals in an area, scaring them away, and bittering agents make certain crops/objects taste unpleasant so that animals do not wish to eat them [13].

3.2.4 Projectile Deterrents

There are various types of non-lethal bullets, such as rubber or plastic bullets. These can be fired at certain animals with the intent of causing pain/shock but not injuring the animal. In Martin et al. [13], projectile deterrents were shown to be effective against bears.

3.2.5 Guardian Dogs

Farmers often use guardian dogs to protect their cattle, sheep, chickens, etc. from predators. In Australia, at Middle Island (near Warrnambool), a successful attempt was made to reduce predation on a penguin colony by using guardian dogs.

3.2.6 Automated Detection & Response Systems

Automated detection systems that integrate IoT and artificial intelligence (AI) have emerged as powerful tools for wildlife management. These systems are able to continuously monitor areas for predator activity and automatically trigger deterrent responses when needed.

Moreover, Passive Infrared (PIR) sensors and GSM-based alert systems, as explored by Kommasani et al. [12] have been tested to detect animal movement and trigger deterrent responses. Such systems often integrate multiple deterrent methods, including sound and light, for a multi-layered approach to predator management.

The integration of machine learning algorithms in these automated detection systems has greatly improved their accuracy and efficiency in real-time. Mishra et al. [9] discuss a system that uses AI-powered cameras and sensors to detect and classify animals. This technology can identify specific species, ensuring that the correct deterrent is deployed. Furthermore, the use of real-time communication and alerts enables farmers or conservationists to respond quickly to potential threats.

3.3 Effectiveness of These Strategies in the Context of Honey Badgers & Penguin Colonies

While various deterrents have been explored, their effectiveness in the context of honey badger predation on penguin colonies is subject to several challenges. This section critically evaluates the strengths and weaknesses of these approaches, considering honey badger behavior and the environmental conditions of remote conservation sites.

3.3.1 Strengths & Weaknesses of Physical Barriers

The use of physical barriers such as electric fences has shown promise in deterring predators, including badgers.

Poole et al. [5] demonstrates that the four-strand electric fence design proves to be highly effective in deterring European badgers, with studies reporting a 95% reduction in crop damage. However, this method may not be entirely suitable for honey badgers, whose digging and climbing abilities make them more difficult to exclude from protected areas.

As Chwalibog [6] points out, honey badgers' behavior complicates the application of such barriers, especially in areas where they can easily dig or climb over the fence. Similarly, raised structures that work well in beekeeping [6] are not directly applicable to penguins, as their colonies are ground-based, requiring alternative adaptations.

3.3. Effectiveness of These Strategies in the Context of Honey Badgers & Penguin Colonies

The addition of reinforced structures or underground barriers may be required to increase the effectiveness of physical barriers, but these adaptations come with additional costs and challenges.

Johnson’s study on honey badger predation [7] demonstrated that while physical barriers such as metal sheeting and raised hive platforms provided some protection, honey badgers’ remarkable adaptability meant that these measures were often circumvented if they were not carefully designed and maintained. In contrast, Clouse’s [8] work with nocturnal pests in South Africa suggests that while physical barriers can play a role in deterring crop raiders, they must be combined with other deterrent methods, particularly when dealing with highly adaptive species like honey badgers.

3.3.2 Strengths & Weaknesses of Sensory Deterrents

Sensory deterrents provide an alternative to physical barriers by utilizing sound, light, and motion to repel predators. Motion-activated scarecrows [8] have successfully reduced foraging behaviour in species such as bushpigs and genets. The multimodal scarecrow [8], which combines flashing lights and randomised sounds, prompted more running behaviour in bushpigs, indicating a fear-based response to the stimuli. Johnson’s [7] findings indicated that motion-activated lights were particularly effective at reducing honey badger predation on beehives by creating a fear of detection.

Despite the promise of sensory-based deterrents, both Johnson [7] and Clouse [8] note the risk of habituation. While sensory deterrents can be highly effective in the short term, their long-term success depends on the predator’s ability to adapt to the changing stimuli. Both studies recommend that these deterrents be used in conjunction with other methods to maintain their effectiveness over time. Additionally, both researchers emphasize the importance of monitoring the responses of the target species to ensure that the deterrents are working as intended and not inadvertently causing negative impacts on non-target species.

According to Gilsdorf et al. [10], animals quickly adapt to repeated stimuli unless the deterrents are modified over time. This presents a significant challenge for long-term deterrence strategies.

Ultrasonic deterrents [10], often marketed as a non-invasive method, may lose their efficacy, especially for highly adaptable species like honey badgers.

3.3.3 Strengths & Weaknesses of Chemical Deterrents

Landa et al. (1998) (as cited in [13]) found that aversive conditioning was effective at deterring wolverines from attacking sheep that had pungent chemicals attached to them via ear tags or collars. However, when these chemical tags were applied to all sheep in the herd, it was no longer effective. The aversive conditioning was only effective in the presence of alternate food sources. Several other sources cited in [13] also show this type of chemical deterrent to be ineffective at reducing predation by coyotes. Capsicum-based sprays were shown to be effective at deterring bears in several different scenarios in [13]. The article concludes that these methods are dependent on the specific predators and circumstances, and produce variable results.

3.3.4 Strengths & Weaknesses of Projectile Deterrents

Projectile deterrents were shown to be effective against bears. However, the weaknesses are numerous when considered in the context of the De Hoop Penguin Colony.

Firstly, the predators that are present near this colony are not bears, but leopards, caracals, baboons and honey badgers. As stated in Martin et al. (2000) [13], smaller animals may be killed or injured by projectile deterrents. Secondly, these projectile deterrents were fired manually by a human holding a gun. In the remote location of the De Hoop Nature Reserve, it is impractical to have humans constantly monitor and police the fence.

3.3.5 Strengths & Weaknesses of Guardian Dogs

Guardian Dogs are routinely used in the context of farming, protecting livestock from predators. However their usage in protecting wild animals is less studied. In Middle Island, Australia, guardian dogs proved effective in protecting penguins from predators.

One limitation of applying this success to the context of De Hoop Nature Reserve, is that the Middle Island colony is on an island, making it harder for predators to reach it compared to the mainland colony at De Hoop Nature Reserve.

3.3.6 Feasibility of AI-Based Systems in Remote Areas

The primary advantage of these systems is their ability to adapt to the behaviour of predators. While traditional deterrents often require manual intervention, AI-powered systems [9] can adjust based on the animal's previous interactions with the deterrents. For example, if a predator like a honey badger begins to show signs of habituation to a specific deterrent, the system can automatically alter its strategy to maintain its effectiveness.

However, as noted by both Mishra et al. [9] and Kommasani et al. [12], the feasibility of these systems in remote conservation areas is still limited by connectivity and power constraints.

PIR sensors and camera-based tracking [12] provide valuable monitoring capabilities but require durable, weather-resistant hardware.

While solar-powered systems [9] provide an energy-efficient solution, they require careful integration with low-power devices to ensure reliable operation in areas without access to the electrical grid. Furthermore, these systems need to be weather-resistant and capable of operating under harsh environmental conditions, which presents additional challenges in coastal and remote regions.

These systems can benefit from continuous adaptation based on predator movement and behaviour, which could be particularly useful in environments where honey badgers and other predators pose a recurring threat.

3.4 ANNs

In order to improve the security of the fence, it is necessary to detect and classify incidents of breaches or near-breaches of the fence, in order to facilitate remedial measures or responses to these incidents.

To this end, it is useful to classify what animals are seen at the fence. ANNs are used widely in classification tasks today, and are especially capable in image recognition tasks such as the animal identification task required for this report. Below, different architectures, as well as some factors affecting ANN performance are outlined.

3.4.1 CNNs

CNNs were developed for image recognition tasks, and are particularly suited for extracting features such as edges or faces in images. They are composed of convolutional layers, in which a layer typically has many smaller windows that are convolved with (slid across) the image (or previous layer) in order to extract different features. CNNs typically consist of:

- **Convolutional Layers:** These are described above, and are the main feature extraction mechanism of CNNs.
- **Pooling Layers:** Used to downsample the image in order to reduce the number of neurons required in subsequent layers, as well as helping to prevent overfitting.
- **Fully Connected Layers:** Used on the output of the network in order to produce a guess.

While CNNs are effective at image classification, they can be improved upon by adding to them.

3.4.2 DenseNet

DenseNet connects each layer to every previous layer. This is useful because it can prevent information from being lost (especially as the number of layers grows). Variants include:

- DenseNet-121: Contains 121 layers, used in image classification.
- DenseNet-169: 169 layers, better performance than DenseNet-121.
- DenseNet-201: 201 layers, more computationally demanding than the above two versions.
- DenseNet-264: 264 layers. Even more computationally demanding, but useful for high resolution images.

3.4.3 Vision Transformers (ViTs)

Departing from CNNs, Vision Transformer (ViT)s use self-attention mechanisms instead of convolution. These models work more like text-based generative AI since it makes use of transformers. It works by token-izing the image (breaking it up into chunks) and then processing each chunk. It has better ability to recognise global patterns in an image.

- Self-Attention Mechanism: Determines relevance between different image regions.
- Positional Encoding: Preserves spatial information lost in patch processing.
- Transformer Layers: Improve interpretability by capturing long-range dependencies.

ViTs outperform CNNs in large-scale datasets. However, the datasets to be used here are not very large.

3.4.4 MLPs

In a **MLP**, each neuron in a layer is connected to every neuron of the previous layer. They are not directly suited for image classification, but are sometimes used at the output stage of a larger network.

3.4.5 Hybrid Models

Hybrid models combine **CNNs**, **ViTs**, and **MLPs** to leverage the strengths of different architectures. An example is seen in [?], where **DenseNet**, **ViT**, and Global Average Pooling layers are combined to classify lung disease.

3.5 Dataset Augmentation

Dataset augmentation is used to help prevent overfitting, as well as to increase the size of the training dataset by artificially creating more data. This is done in a few ways:

3.5.1 Simple Transformations

- **Rotation**: Simulate different viewing angles by rotating the images in the dataset randomly.
- **Cropping**: Remove parts of the image.
- **Flipping**: Horizontally or vertically flipping images to increase variation.

3.5.2 Colour Adjustments

Adjusting the RGB channels of the image.

3.5.3 Introducing Noise

Artificially injecting noise helps a network ignore noise in real images. Some methods include:

- Gaussian noise.
- Motion blur simulation.
- Sharpening or blurring.
- Random patch obscuring or removing.

3.6 Data Pre-processing

While data augmentation includes *adding* noise into training data, it is also useful to *remove* noise from real data to improve classification accuracy. Some methods include:

3.6.1 De-noising

Some examples of de-noising are:

- Gaussian smoothing (reduces Gaussian noise).

- Morphological transformations (erosion, dilation, opening, closing).

3.6.2 Normalization

Normalization helps correct for lighting differences by doing the following:

- Scaling pixel values to fit from 0 to 1.
- Histogram equalization (where the range of pixel values are used to adjust the image, ensuring maximum contrast).

3.7 Conclusions from the Literature

The review of existing predator deterrents has shown that many conventional methods are unsuitable for long-term application in protecting the penguin colony that resides in De Hoop nature reserve. Visual and acoustic deterrents, such as predator-activated scarecrows, fail over time due to habituation [8, 13]. Similarly, aversive conditioning (chemical deterrents) has been effective only when alternative, untreated food sources are available, making it unreliable for remote conservation settings [13]. Projectile repellents, which rely on human intervention, are impractical for autonomous wildlife protection [13].

After ruling out these approaches, two potential solutions remain: guard dogs and physical barriers. While guard dogs have successfully protected penguin colonies in Australia [11], they fall outside the scope of this project due to their lack of an engineering component—unless robotic alternatives are considered, which would be costly and impractical. Instead, this report will focus on enhancing physical barriers, particularly improving the existing electric fence, to ensure greater reliability and prevent future honey badger intrusions, such as the one that resulted in the loss of 11 penguins in the De Hoop colony [14].

Chapter 4

Predator Detection Subsystem By: Imaan Shaik - SHKIMA004

4.1 Predator Detection Subsystem Overview

The primary objective of this subsystem is to detect the presence of predators near the fence and reliably trigger the Imaging subsystem to initiate recording and computer vision processing. Since any nearby movement is considered relevant, the sensing logic must function consistently across varying weather and lighting conditions. Signal filtering (both hardware and software) is employed to minimize false triggers from sunlight, wind, and ambient heat. Detection reliability is guided by the most challenging threat: the honey badger, known for its size and burrowing behavior.

4.2 Requirements and Specifications

4.2.1 User requirements

Requirement (SRE)		Specification (SP)		Acceptance Test (AT)	
SRE-1	The system must not disturb the target predators (i.e honey badgers) nor the non-target species, particularly penguins.	SP-1	Use passive sensing and silent sensing methods	AT-1	Test that end design and placement to confirm no visual or auditory harm.
SRE-2	The system must consume low power	SP-2	Total power consumption of the sensing subsystem must not exceed 2W (of the 12W system budget).	AT-2	Measure average current draw under typical operating conditions. The sensing system must turn off when no motion nearby.

Table 4.1: Traceability Matrix: User Requirements, Specifications, and Acceptance Criteria

4.2.2 Functional Requirements

System Requirement (SRE)		Specification (SP)		Acceptance Test (AT)	
SRE-3	The system must reliably detect the presence of predators near the fence line.	SP-3	Implement a verification mechanism that confirms detected motion originates from an actual threat.	AT-3	Conduct controlled walk-through tests at varying distances. Confirm detection occurs only within the specified detection zone.
SRE-4	The system must detect different predator types, including nocturnal species such as honey badgers.	SP-4	Detection sensors must operate effectively in both daylight and nighttime conditions.	AT-4	Test system under sunlight and complete darkness. Confirm consistent detection and ambient light rejection in both cases.
SRE-5	The system must trigger the camera subsystem upon valid detection.	SP-5	Use a reliable communication protocol (e.g., UART) to send trigger signals to the Raspberry Pi-based camera module.	AT-5	Log serial transmission and validate trigger reception using the Pi's system logs or camera activation records.
SRE-6	The design must be modular and easily replicable for large-scale deployment.	SP-6	All components must be low-cost, off-the-shelf, and easy to assemble using standard tools.	AT-6	Independently assemble a second unit using the provided wiring diagram and BOM. Confirm identical performance.
SRE-7	The system must filter out false detections caused by environmental noise.	SP-7	Apply hardware (RC filtering) and/or software (timing debounce, thresholding) noise rejection techniques.	AT-7	Compare raw sensor signals against filtered outputs under dynamic lighting. Confirm reduction in false triggers.
SRE-8	Detection must occur before the predator makes direct contact with the fence.	SP-8	The detection range must cover a minimum of 1 meter from the fence outward.	AT-8	Place test objects at varying distances. Verify system reliably detects motion at or beyond the 1m threshold.
SRE-9	The system must trigger the camera with minimal latency after confirmed detection.	SP-9	Ensure total response time from detection to camera trigger is less than 1 second.	AT-9	Measure and log the delay between motion detection and GPIO signal to the Pi. Confirm sub-1s timing.

Table 4.2: Traceability Matrix: Functional Requirements, Specifications, and Acceptance Criteria

4.3 Design Rationale

4.3.1 Sensor Selection and Justification

Various motion sensors were evaluated (Table A.3). [Passive Infrared \(PIR\)](#) sensors were selected for their low power draw, simplicity, and ability to detect warm-bodied animals [3]. Although an analog PIR system was originally planned, component availability shifted the design to a digital HC-SR501 module. This offered reliable detection with adjustable delay and sensitivity controls.

4.3.2 Microcontroller Selection

A Raspberry Pi was not used as its OS-based architecture was excessive for this subsystem's lightweight logic. Instead, the [Espressif Systems 32-bit Microcontroller \(ESP32\)](#)-WROOM-32 offered the best balance of processing power, GPIO availability, and cost. Its integrated Wi-Fi and Bluetooth add future scalability. With appropriate enclosure protection, the ESP32 is also robust enough for long-term use in harsh outdoor environments. A comparison of platforms is provided in Table 7.6, where the ESP32 outperformed STM32 and Arduino in nearly all metrics.

Aspect	ESP32	STM32	Arduino Uno
Power Consumption	Highly efficient in deep sleep modes; suitable for battery-powered use	Low-power modes available but less optimized than ESP32	Relatively higher consumption due to lack of advanced power modes
Modularity	Extensive: GPIO, Analog-to-Digital Converter (ADC) , DAC, I2C, SPI, PWM	Highly modular with rich peripheral sets	Modular via shields; limited built-in flexibility
Scalability	Ideal for small to medium scale systems	Scalable across wide performance tiers	suitable for basic applications
Environmental Robustness	Reliable with a proper enclosure; suitable for outdoor use	Excellent for harsh environments	Not designed for harsh conditions
Wireless Communication	Integrated Wireless Fidelity (Wi-Fi) and Bluetooth	Requires external wireless modules	Requires external modules for Wi-Fi/Bluetooth
Processor Architecture	32-bit Dual-core Xtensa LX6	32-bit ARM Cortex	8-bit AVR ATmega32
Clock Speed	Up to 240 MHz	Typically 72 MHz	16 MHz
Connectivity	Wi-Fi and Bluetooth (built-in)	No built-in wireless connectivity	No built-in wireless connectivity
Operating Voltage	3.3 V	2.0 – 3.6 V	5 V

Table 4.3: Comparison of Microcontroller Platforms

4.3.3 Power Regulation

Power is supplied across all subsystems from a 12 V 1 A source. The [buck converter](#) module selected by the Camera Subsystem was shared to fulfill the requirements of this subsystem. The sensing system uses a 5 V output from the buck regulator, which is sufficient for the ESP32 Dev Kit and all peripheral components.

4.4 Predator Confirmation Logic

4.4.1 Primary Detection – PIR Sensor

While building an analogue PIR sensor was the original plan for this subsystem, a digital PIR module was selected due unavailability of the analog PIR component ordered.

Two PIR sensors were considered, and the HC-SR501 was selected for this system due to its broader detection angle, longer range and configurable delay. Table 4.4 compares key specs.

Property	HC-SR501	HC-SR505
Range	3m–7m	3
Time Delay	3s – 5min	8s – 10.4s
Detection Angle	110°	<100°
Trigger Mode	Single / Repeated	Repeated

Table 4.4: Infrared Sensor Comparison

4.4.2 False Trigger Mitigation

To reduce the occurrence of false triggers due to environmental noise, the PIR sensor output undergoes software-based filtering on the ESP32, as discussed in [A.4](#).

To reduce false detections from the HC-SR501 PIR sensor, software-based filtering was applied:

- **Debouncing:** Ignores brief noise spikes $< 100\text{ ms}$
- **Minimum Duration:** Accepts motion only if HIGH for $> 300\text{ ms}$

- **Pulse Stretching:** Ensures valid detections are merged into a single trigger
- **Cooldown:** Waits before accepting new triggers to avoid repeats

These techniques add no hardware complexity while improving accuracy under unstable lighting and windy conditions. Although the software-based approach introduces slight computational overhead, the ESP32's dual-core architecture ensures responsive performance.

4.4.3 Secondary Detection – Laser Break-Beam

The PIR sensor was chosen to detect general motion near the fence, with filtering applied to improve trigger accuracy. However, it cannot confirm whether the detected motion is an actual threat. To address this, a secondary verification method: a laser break-beam, was added. This module confirms an intrusion has entered a critical zone near the fence, indicating a real threat. This layered approach mimicks techniques commonly used in security alarm systems, enabling high-confidence triggering.

The HW-493 laser module (5 mW, 650 nm) and its matching receiver module were selected. The laser system outputs a digital **HIGH** when the beam is detected and **LOW** when interrupted, enabling straightforward digital line-of-sight detection.

Another consideration was an IR break beam module, as utilized in remote control systems. This module was not selected due to its limited range, high susceptibility to ambient infrared interference and lower reliability in outdoor environments compared to the laser based system.

Due to the system's power constraint, the laser is not powered continuously. Instead, it is connected through a 1 k Ω resistor to the base of a 2N2222A NPN transistor, allowing the ESP32 to pulse the laser via GPIO only when needed (as seen in A.3). This reduces unnecessary power draw while preserving reliable detection. An analysis of detecting using a PIR alone vs in conjunction with a break beam was done in table A.5

4.4.4 Camera Trigger Communication Protocol

To trigger the Raspberry Pi camera from the ESP32 upon predator detection, UART was chosen over I²C due to its simplicity, low overhead, and native support on both devices. This communication (connected as mentioned in A.6), sends a discrete command from the ESP32 to the Pi using a two wire TX–RX connection with a shared ground.

Table 4.5: UART vs I²C for Camera Triggering [1]

Protocol	Information
UART	Simple to implement; supports direct string transmission; reliable for short-distance asynchronous signaling; no clock or addressing needed.
I ² C	Requires Pi to be configured as slave; more setup complexity and not ideal for one-way, event-based signaling.

4.5 Implementation and Testing

4.5.1 Detection Logic

The logic is as follows:

- Confirm PIR motion after filtering
- Pulse the laser ON for 10 s
- If beam is broken for ≥ 2 s, trigger camera for 15 s
- Enter cooldown before rearming

4.5.2 Performance of PIR and Laser Modules

During testing, the PIR sensor demonstrated reliable performance under all conditions. The module's delay and sensitivity adjustment hardware dials worked effectively when combined with ESP32-based software filtering.

In contrast, the laser receiver module exhibited noise issues, particularly under ambient indoor and daylight conditions. Despite software filtering, it remained prone to false readings during the day, despite performed reliably in the evening. The receiver module lacks a formal datasheet, but it is likely based on a photodiode, which is highly sensitive and may be responding to general ambient light instead of the focused laser beam.

4.5.3 Designing a laser receiver circuit

To improve upon the performance laser receiver module, a receiver circuit was designed from various components, such as a phototransistor, LM324 op-amp, LDR, Potentiometer and Lm393 comparator. This design aimed to mitigate ambient light interference, reduce false positives, and provide a reliable break-beam detection to the ESP32.

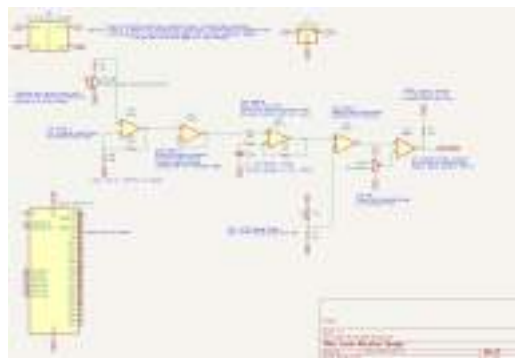


Figure 4.1: KiCad schematic of the designed laser receiver circuit

This circuit was employed [A.1](#) but ultimately failed due to problems debugging the circuit and insufficient tests were conducted to get this circuit to work. The circuit was able to filter analog noise but did not provide an adequate digital trigger output from the comparator. The schematic design was included for future improvement.

4.5.4 Hardware Filtering the Laser Rx Module

Due to not getting the laser receiver circuit to work, hardware filtering and constant debugging of the receiver module was done to try and get working functionality of the break-beam system.

An RC low-pass filter (comprised of $1\text{k}\Omega$ and 100nF) was added to the laser receiver output [A.4](#) to mitigate ambient noise. This simple hardware filtering improved signal stability and reduced false triggers in daylight.

The filter has a time constant of:

$$\tau = RC = (1\text{ k}\Omega)(100\text{ nF}) = 0.1\text{ ms}$$

This promotes a fast response time, allowing the receiver to register beam breaks more quickly. Without filtering, brief hand motions or ambient flickering caused false beam break detections. After implementing the RC filter, the system responded only to deliberate, sustained beam interruptions which is consistent with expected logic [A.2](#).

Test Condition	Before RC Filter	After RC Filter
Brief hand flick ($< 1\text{ s}$)	Triggered camera (false positive)	No trigger
Ambient flicker (indoor light)	Triggered camera (false positive)	No trigger
Deliberate beam break ($> 2\text{ s}$)	Triggered correctly	Triggered correctly
Stability under idle conditions	Unstable readings (flickering)	Stable and consistent

Table 4.6: Observed receiver performance before and after RC low-pass filter implementation

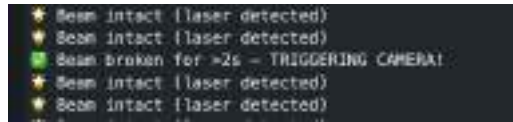


Figure 4.2: Serial monitor output showing improved signal stability and clean trigger detection after RC filtering

This hardware filtering iteration significantly improved the receiver module's output stability. The low pass filter ensures stable logic levels under both daylight and indoor conditions, without excessive response delay.

4.6 Acceptance Testing, Results and Analysis

See Appendix [A.3](#) for additional sensing photos and testing documentation.

AT-ID	Test Description	Test Procedure	Expected Result	Actual Result	P/F
AT-01	Visual disturbance test for predators and penguins	Inspect final subsystem setup for lights, sounds, or movements. Review placement on outer fence.	Laser beam is visible at night but only to predator side of fence.	Not tested in the field, but tests at night confirm the visibility of laser beams to predators. May deter them (hopefully) or lure them so we can adequately alert their presence to stakeholders	N/A
AT-02	Power consumption measurement	Measure average current draw using multimeter under normal detection cycles.	$< 2\text{ W}$ average power draw under test conditions.	Measured 1.5W under full operation (ESP32 + sensors).	P
AT-03	Motion detection distance test	Walk through detection area at different distances. Observe detection status.	Detect motion $\geq 1\text{ m}$ from fence consistently.	Consistently detected movement at 1.5m+ range with HC-SR501. Range limits at 6m.	P
AT-04	Night/day detection performance	Test under both direct sunlight and no light conditions (indoors).	Detect valid motion in both scenarios.	Day and night tested. Detection remained reliable (ambient light filtered).	P
AT-05	Trigger signal delivery to camera	Simulate detection event and observe Pi camera log or LED trigger.	Trigger received within 1s of detection.	UART signal sent and camera triggered in 300ms (as seen in log).	P
AT-06	System replicability check	Simulate second unit using same design. Perform same detection tests.	Identical performance with modular components.	The sensors and lasers are able to be duplicated and scaled along the fence. Unsure of how connection will be made but likely through advancing the project to one main Pi to control many slave devices. The laser and PIR system is highly scalable and many lasers can be employed on each enclosure for more robust and wider range of traps. A.5	P
AT-07	False trigger reduction test	Expose to flickering lights, moving shadows, and wind simulation.	False triggers significantly reduced via filtering.	Software + RC filtering prevented flicker and breeze triggers. 4.6	P
AT-08	Trigger latency test	Track the time from a simulated motion event to the GPIO output. Monitor via serial logs.	$< 1\text{ s}$ latency from detection to trigger.	Measured average delay: $\sim 350\text{ ms}$	P

Table 4.7: Acceptance Test Procedures (ATPs) for the Sensing Subsystem

4.7 Design Trade-offs and Final Outcome

An analog laser receiver circuit was developed (Figure 4.1), but due to debugging challenges and time constraints, it was not deployed. Software filtering was preferred for its simplicity and reliability. The final system, combining a digital PIR and laser break-beam, achieved all goals and is scalable for full perimeter deployment.

The dual verification strategy significantly reduces false triggers and ensures reliable detection of threats like honey badgers. The ESP32 controller, with software filtering and power efficient design, enables robust outdoor scalability.

4.8 Recommendations

Future iterations should prioritize:

- Refining the analog laser receiver circuit with more test iterations and debugging
- Adding a 650 nm narrow-band optical filter to the receiver for better noise rejection [\[15\]](#) [\[16\]](#)
- Exploring digital output phototransistor modules with built-in modulation filtering

These additions would enhance detection reliability, especially under fluctuating daylight conditions.

Chapter 5

Housing Subsystem

By: Imaan Shaik - SHKIMA004

5.1 Housing Subsystem Introduction

This subsystem details the design and implementation of a robust enclosure to house all integrated components of the predator detection and fence monitoring system.

The enclosure is designed to operate reliably in the harsh coastal conditions of the De Hoop Nature Reserve, where it is exposed to salt-laden air, wind, sand, rain, and seasonal temperature extremes ranging from 7°C to 25°C [17].

The design prioritizes environmental protection, ensuring that the internal components are shielded from the elements. The enclosure also facilitates optimal placement and unobstructed function of critical components, while remaining non-invasive to local wildlife.

5.2 Requirements and Specifications

5.2.1 User Requirements

The following user-driven requirements were defined in collaboration with the primary stakeholder:

Requirement (HRE)		Specifications (SP)		Acceptance Test (AT)	
HRE-1	The system must withstand harsh coastal conditions, including salt, wind, and rain and be durable for long term deployment.	SP-1	The enclosure must be made of strong, weather-resistant materials, with a suitable IP rating.	AT-1	Ensure that no water or dust can enter the enclosure through testing.
HRE-2	The enclosure must not disrupt the natural behavior or habitat of penguins and other species.	SP-2	The design must ensure minimal visibility and interference with the penguins, focusing only on predator detection.	AT-2	The enclosure is positioned on the outer fence side, away from the penguin colony.
HRE-3	The design must be scalable and enable deployment across the full fence perimeter.	SP-3	The enclosure must support modular installation and support a replicable design.	AT-3	Test for scalability
HRE-4	The housing must be cost-effective and suitable for mass deployment.	SP-4	Use off-the-shelf components and low-cost materials where possible.	AT-4	Test for budget compliance

Table 5.1: Traceability Matrix: User Requirements, Specifications and Acceptance Criteria

5.2.2 Functional Requirements

These requirements ensure the enclosure supports the sensing system and camera system and withstands field conditions:

Functional Requirement (HRE)		Specifications (SP)		Acceptance Test (AT)	
HRE-5	The enclosure must support the components doing the detection and monitoring of predators.	SP-5	The internal supports must ensure line-of-sight alignment and unobstructed operation of all components (camera, PIR and laser)	AT-5	Sensor and camera functionality validated through alignment and detection testing.
		SP-6	The enclosure must be compartmentalized with the correct dimensions of all subsystems.	AT-6	Test that all components fit in their designed compartments.
HRE-6	Internal components must be cooled effectively without allowing salt, dust, or insect ingress.	SP-7	The enclosure must be fully sealed with no holes that enable insects or elements to come in, especially salt from the air which could corrode the internal components	AT-7	Temperature measurements of the Pi after cooling to confirm safe operation.
				AT-8	Enclosure sealing tested for air tightness
HRE-7	Enclosure must allow easy access to components for maintenance and debugging	SP-8	The components must be placed in a spread out and spacious manner for easy access		

Table 5.2: Traceability Matrix: Functional Requirements, Specifications and Acceptance Criteria

5.3 Design choices

The technical details and justification for the design choices are covered in detail in the following subsections.

5.3.1 Material Selection

The choice of enclosure materials is heavily influenced by the harsh environmental conditions at the deployment site and potential interactions with surrounding wildlife. Material selection must balance strength, cost, manufacturability, environmental impact, and resistance to weathering to ensure long-term durability and safe operation.

Table 5.3 illustrates the comparison between potential materials based on the factors listed.

Property	Sheet Metal	Plywood	PLA Plastic	Perspex (Acrylic)	ABS Plastic
Cost	High	Low	Moderate	Moderate	Moderate
Construction	Difficult (needs tools)	Moderate (manual tools but inconsistent)	Easy (3D print)	Easy (laser cut)	Moderate (requires molding or print)
Strength	Very High	Moderate	Low to Moderate	Moderate	Moderate to High
Environmental Impact	High (requires coatings)	High (biodegradable, rots)	Moderate (non-biodegradable)	Moderate	High
Weather Tolerance	Very High (if coated)	Low (absorbs moisture, degrades)	Low (warps and cracks)	Moderate (needs sealing)	High
UV Resistance	High	Low	Low (degrades in sun)	High	High
Water Resistance	High	Very Low (requires sealing)	Low	High (when sealed)	High
Suitability for Coastal Use	Poor (rust)	Poor	Low	Moderate	High

Table 5.3: Material Comparison for Enclosure Design [2]

Based on the comparison in Table 5.3, Perspex (acrylic) was selected as the enclosure material due to its balance of durability, ease of fabrication, and moderate environmental resistance. While not the most robust option for long-term coastal deployment, its relatively low cost and ability to achieve adequate water resistance make it well-suited for scalable installation. With an IP rating potential

of approximately IP54–IP65 [18] [19], Perspex meets the environmental protection requirements expected at the deployment site.

For full-scale deployment in harsher coastal environments, ABS Plastic would be the preferred material due to its superior durability, higher IP rating potential (up to IP66) [19], and long-term resistance to corrosion and UV degradation. However, due to project constraints, Perspex was chosen as a practical and cost-effective compromise.

5.3.2 Internal Cooling Techniques

Cooling requirements were primarily driven by internal heat build-up, as external temperatures at the site are quite moderate. To ensure reliable operation, internal components must remain within safe thermal limits to avoid overheating and potential damage.

After evaluating the available options, passive cooling using an integrated [heatsink](#) (as part of the enclosure) was selected as the most suitable approach. This decision was based on the following considerations:

- The enclosure is made from Perspex, which is non-conductive and unsuitable for ventilation in a coastal environment exposed to salt, sand, and humidity.
- Air vents or filters would compromise the enclosure’s seal, increasing the risk of corrosion and system failure due to dust or salt ingress.
- Active cooling using a fan was ruled out due to its power demand, moving parts, and vulnerability to environmental conditions, especially under the strict 12V, 1A power constraint.
- Passive heatsinks effectively dissipate heat from the Raspberry Pi and other components through conduction, maintaining thermal stability without requiring airflow or compromising enclosure integrity.

This method keeps the system sealed, energy-efficient, and reliable, providing the best trade-off between thermal regulation, environmental protection, and long-term operational robustness.

5.4 Submodule Design and Implementation

5.4.1 Compartmentalization and Ensuring Component Function

To maintain modularity and simplify future deployment, components were spaced with clear routing and support structures. Table 5.4 lists component sizes used in CAD modeling. The main enclosure was oversized slightly to improve passive cooling and fit all electronics. Receiver and transmitter housings were 3D printed and shielded from sunlight to reduce interference.

Placement	Component	Dimensions (L × W × H) in mm
1	HC-SR501 PIR Sensor	25 × 25 × 16
2	Voltage Regulator	20 × 20 × 5
3	Sensor Circuit Veroboard	72 × 54
4	Camera Circuit Veroboard	50 × 50
5	Laser Receiver	6 × 6
6	Laser Transmitter	0.6mm Diameter
7	Raspberry Pi Zero 2 W	65 × 30 × 16
8	Heatsink	65 × 38 × 26

Table 5.4: Critical Dimensions to account for in Enclosure Design

To meet the functional requirements of the laser break-beam system, the design was implemented using a two-enclosure setup.

An early design concept aimed to house both the transmitter and receiver within a single enclosure using a mirror to reflect the beam path. However, this approach was discarded due to environmental risks such as beam misalignment from rain, reflection interference from sunlight, and potential disruption by animals.

The final configuration places all system components, including the laser receiver module, inside the main enclosure, while the laser transmitter is housed separately in a secondary enclosure. These two units are connected via a pair of wires. This separation improves reliability by preserving line-of-sight and minimizing ambient interference.

The main enclosure was laser-cut from Perspex and designed with internal compartments to improve accessibility during maintenance. It was intentionally oversized to increase internal volume and surface area, enhancing passive heat dissipation of the components.

The internal component supports were 3D-printed using PLA plastic, which was selected for its ease of prototyping. Weather resistance was not a critical concern since these parts were housed inside the sealed Perspex enclosure. The transmitter housing was also 3D printed, as its main purpose was to demonstrate break-beam functionality rather than endure extended outdoor use.

To reduce false negatives caused by ambient light, especially direct sunlight, the laser transmitter and receiver were physically shielded. The receiver was fitted with a 6 x 6 mm square shaped extended shade for the photodiode, and the transmitter was housed in a square casing with an extended hole panel to focus the beam. Both enclosures were 3D printed using PLA.

5.4.2 Thermal Monitoring and Passive Cooling

After analyzing the power dissipation of the sensing subsystem, none of the components were found to generate significant heat to require active cooling. Adequate spacing within the enclosure was deemed sufficient for thermal management.

Following a consultation with the camera subsystem, the only internal heat source of concern was identified as the Raspberry Pi Zero 2 W.

The Raspberry Pi features a built-in temperature sensor that allows real-time thermal monitoring. The system operates safely as long as the temperature remains below 75°C, providing a buffer below the

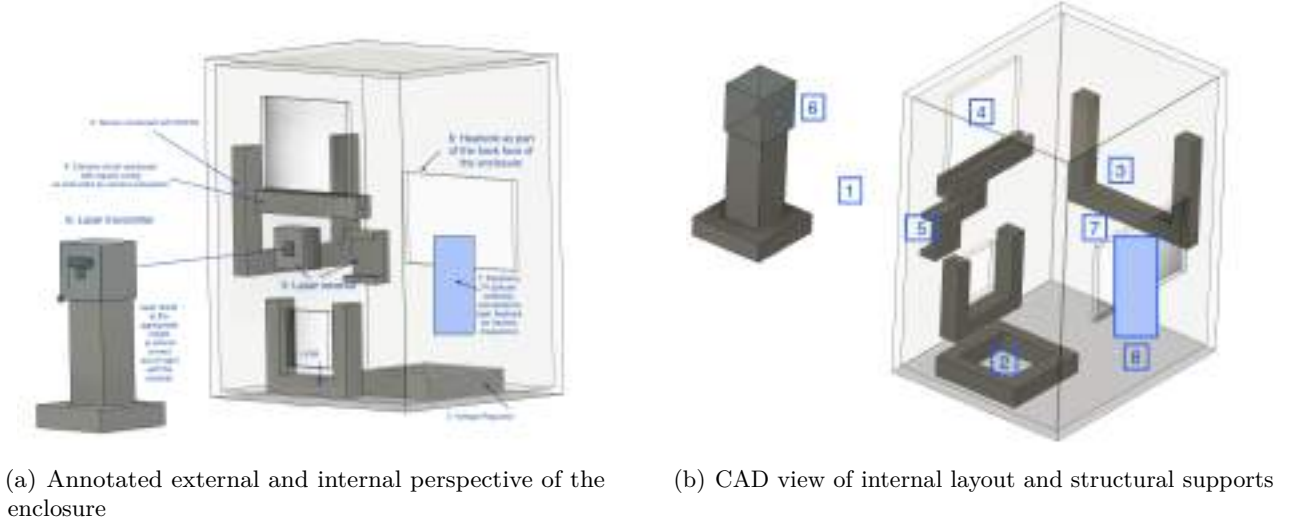


Figure 5.1: CAD renderings of the enclosure system showing internal component arrangement, external laser alignment, and heatsink placement.

thermal throttling threshold of 85°C [20] [21] [22]. Tests were conducted under prolonged operation, and the cooling system successfully maintained the Raspberry Pi's temperature well within safe limits.

To ensure reliable thermal performance without compromising the enclosure's environmental seal, a custom passive cooling system was designed. As the enclosure is constructed from Perspex, a fully sealed conduction-based heatsink solution was implemented.

A small aluminum heatsink (thermal resistance of $25\text{--}30^{\circ}\text{C/W}$) [23] was mounted directly onto the Raspberry Pi's CPU from the thermal adhesive sticker on the back. A second heatsink's aluminium fins were interlocked on top of the Pi's heatsink, with thermal paste applied between the fins to ensure conduction and increase the surface area. This stack created a flat top interface, which was then thermally adhered to a larger external heatsink embedded into the rear face of the enclosure. The main heatsink, made of anodized aluminum, is 37.5 mm in length and rated at 3.25 K/W [24], is designed to extrude out of the rear face of the enclosure, to dissipate heat without air flow coming in. It was attached using high-temperature, electrically insulating silicone paste/ gasket to ensure a thermally conductive yet weather-sealed interface with the Perspex enclosure (A.7).

The rate of temperature rise through a heatsink is governed by its thermal resistance [25] [26] [27] :

$$\Delta T = \theta \cdot P$$

Where:

- ΔT is the temperature rise above ambient,
- θ is the thermal resistance of the heatsink (in $^{\circ}\text{C/W}$),
- P is the power dissipated (in W).

Assuming a worst-case estimate of $P = 2.5\text{ W}$ (typical for the Raspberry Pi Zero 2 W under load [28])

) and using the main heatsink's thermal resistance of $\theta = 3.25 \text{ K/W}$:

$$\Delta T = 3.25 \times 2.5 = 8.125^\circ\text{C}$$

Assuming an ambient temperature of 25°C , the estimated CPU temperature is:

$$T_{\text{CPU}} = 25^\circ\text{C} + 8.125^\circ\text{C} = 33.13^\circ\text{C}$$

Since the heatsink protrudes from the rear face of the enclosure and is externally accessible, it must be earthed [29] [30] to ensure user safety and suppress electromagnetic interference (EMI) [31], which is critical given the high-frequency signals processed by the ESP32 and Raspberry Pi. Although it is thermally bonded using an electrically insulating RTV silicone paste and not connected to the circuit ground, its exposed metal surface can act as a radiating structure. Earthing the heatsink ensures that any buildup of displacement current or interference is safely redirected to earth, maintaining electromagnetic compatibility and minimizing risk of electrical shock. A rivet or bolt made of conductive material (like copper) can be installed to provide an electrical and mechanical connection for a wire to earth the heatsink. ¹.

5.5 Acceptance Testing, Results and Analysis

See Appendix A.7 for additional enclosure photos and testing documentation.

¹This was unfortunately not implemented due to limited time and issues securing a hole in the heatsink, however, future iterations will implement this

AT-ID	Test Description	Test Procedure	Expected Result	Actual Result	P/F
AT-01	Ensure that no water or dust can enter the enclosure through testing.	Expose the enclosure to water from the shower to test water resistance; place in a dusty environment.	No water or dust inside the enclosure.	Enclosure remained sealed under splash conditions and no dust passed the seals.	P
ATP-02	The enclosure is positioned on the outer fence side, away from the penguin colony.	Upon deployment, ensure the system faces the predator side of the fence.	Successful implementation	Unable to go to site and perform test however the system was modeled to be deployed in that manner	N/A
ATP-03	Scalability and Mounting Test	Mount enclosure on representative wire fence.	Materials used and design is modular and replicable	The materials used are widely available for mass deployment and the enclosure design is simple and capable of being replicated and placed at any height or location along the perimeter (optimally at the height near the ground to sense a badger) A.9	P
ATP-04	Budget Compliance	Check total Bill of Materials.	Enclosure cost <R500.	Cost confirmed at <R500 (see A.2)	P
ATP-05	Component Functionality and Alignment Test	Insert PIR, camera, laser modules into designed compartments. Check mechanical fit, alignment, and functional performance when tested.	All components fit securely and align correctly.	All modules fit flush within the supports made; line-of-sight confirmed. A.8	P
ATP-06	Subsystem Compartment Test	Measure internal cut-outs and compare with component specs. Test for wiggle/play.	Correct spacing and dimensions	Dimensions matched; components did not shift during testing.	P(A.6)
ATP-07	Passive Cooling Performance (A.7(b))	Stress test Raspberry Pi for 20 minutes and log CPU temperature seen on laptop.	CPU remains below 75°C. (A.7(a))	Peak temperature recorded: 58.3°C.	P
ATP-08	Airtightness and Vent Prevention	Inspect enclosure for unsealed openings. Blow hairdryer and listen for any air gaps.	Fully sealed enclosure; no holes for element ingress.	Enclosure laser cut with finger-joints, perspex glued each side and silicone-insulated heatsink as part of the enclosure; no openings or air flow inwards after test.	P(A.8)

Table 5.5: Acceptance Test Procedures (ATPs) for the Housing Subsystem

5.6 Conclusion

This housing subsystem met all major requirements: environmental protection, cooling, modularity, and alignment of critical components. Although full field deployment wasn't tested, the prototype validated enclosure feasibility for harsh conditions. Key improvements for future versions include proper heatsink earthing and ABS material substitution.

5.7 Deliverables

The GA Table for the Predator Detection and Housing subsystems can be found in Appendix [A.1](#). The Bill of Materials can also be found in Appendix [A.2](#)

Chapter 6

Front-end Subsystem

Author: Safiya Mia - MXXSAF002

6.1 Introduction

The front-end subsystem provides conservationists, like Christina, with a remote-accessible interface for viewing data captured by the camera subsystem. It displays images, videos, and metadata, and includes analytics to help understand predator behaviour along the fence. This chapter outlines its design process, beginning with specifications based on user requirements from Christina, followed by acceptance test procedures. It then discusses the critical design decision made and presents the final implementation, which is evaluated against the test criteria.

6.2 Scope and Limitations

6.2.1 Scope

The front-end subsystem is responsible for providing a user-friendly web interface that allows researchers to access and interact with data collected by the camera system. Its primary functions include displaying detection images and videos, presenting metadata such as timestamps and species classification done by computer vision in the camera subsystem, and providing summary analytics on animal activity. The interface also enables users to download media files for offline analysis and to filter the animal logs and graphs, according to preference.

6.2.2 Limitations

The main limitation of the front-end subsystem is its reliance on a single, fixed camera position for data collection. As a result, the information presented through the interface is limited to one viewpoint along the fence, which may not fully represent broader animal activity in the area. This constraint reduces the system's spatial awareness and could lead to missed detections outside the camera's field of view. It also limits the scope and richness of the analytics generated, as visual trends and detection patterns are drawn from only one location. The resulting charts, while functional, may not reflect the full extent of nocturnal movement or species distribution across the reserve. While future iterations

could incorporate multiple camera nodes to enhance spatial coverage and improve analytical depth, this version is designed to demonstrate functionality using one static source.

6.3 Requirement Analysis

To ensure a robust and well-tested front-end subsystem, this section outlines a detailed traceability matrix connecting user needs, technical specifications, and their corresponding acceptance tests. The tables below are structured to separately show the requirements, derived specifications and the acceptance tests for both non-functional and functional aspects of the subsystem.

6.3.1 Non-functional Requirements, Specifications, and Acceptance Tests

In this chapter, non-functional refers to qualities that enhance the subsystem’s performance or usability without altering its core functionality.

Requirement		Derived Specification		Acceptance Test	
ID	Description	ID	Description	ID	Pass Criteria
NFR-01	The front-end must be user-friendly.	NFSP-01	The system must maintain a consistent visual style using a limited set of colours and fonts.	NFAT-01	The interface uses a custom colour palette of less than 5 colours, 1 font style, and not more than 10 buttons/inputs (excluding download buttons).
		NFSP-02	The layout must be clear and include sections and headings.	NFAT-02	Headings and sections must be clearly visible and verified by peer feedback.
		NFSP-03	The system must display correctly on different devices and screen sizes.	NFAT-03	UI must adjust without distortion on phones, tablets, and laptops.
NFR-02	There should be no recurring costs.	NFSP-04	All tools, libraries, and services must be open-source or freely licensed.	NFAT-04	All components verified to be free or open-source (e.g., Bootstrap, Flask, Chart.js).

Table 6.1: Non-functional Requirements, Specification, and Acceptance Test

6.3.2 Functional Requirements, Specifications, and Acceptance Tests

Requirement		Derived Specification		Acceptance Test	
ID	Description	ID	Description	ID	Pass Criteria
FR-01	The user must be able to access data remotely through a browser	FSP-01	The system must host a Flask-based web server accessible via local or internet IP.	FAT-01	UI loads on different devices (e.g., phone, laptop) within 5 seconds.
FR-02	The user must be able to view detection images, videos, and metadata	FSP-02	The interface must display detection entries with correct timestamps and species classification.	FAT-02	Manually inserted test detections appear with the correct media and metadata.
FR-03	The user must be able to download image and video files.	FSP-03	Each detection entry must have a download button that allows media to be saved locally.	FAT-03	Clicking the download button saves the correct file with the appropriate filename.
FR-04	The user must be able to view detection trends and patterns over time.	FSP-04	The system must generate pie/line charts summarising detection activity using Chart.js.	FAT-04	Bar and pie charts render correctly and match known test data distributions.
FR-05	The user must be able to see new detections without restarting the system.	FSP-05	Refreshing the page must fetch and display the latest logged detections from the Pi storage.	FAT-05	A new detection added to the back-end appears in the UI without restarting the server.
FR-06	The subsystem must restrict access using a password-protected login page.	FSP-06	A password screen must be displayed on first visit. Access is only granted after validating the password (securely stored or hashed).	FAT-06	Correct password grants access. Incorrect password shows error.

ID	Description	ID	Description	ID	Pass Criteria
FR-07	The subsystem must be lightweight and low-power for shared or battery-powered use.	FSP-07	No background polling is used; media is only loaded on demand. Flask runs with minimal overhead to conserve CPU and memory.	FAT-07	System remains under 10% CPU usage and low memory load during idle periods; no background services run unless triggered by user actions.
FR-08	The user must be able to filter and sort detection data on the UI.	FSP-08	The interface must provide dropdown filters and sorting options for species and time range. Filtered results must be fetched using Flask route parameters.	FAT-08	Species and time range selections correctly update the displayed results without reloading the full page or breaking functionality.

Table 6.2: Functional Requirements, Specification, and Acceptance Test

6.4 Back-End Design and Theory

6.4.1 Selection of Processor and Hosting Platform

Multiple options were considered and compared in terms of cost, power consumption, processing capability, integration complexity, and suitability for long-term deployment in remote environments (see Table 6.3). The ESP32 was an appealing option due to its extremely low power draw and cost. However, it lacks a full operating system, making it poorly suited for serving a rich, file-heavy user interface or managing large video/image files generated by the camera subsystem. Implementing a responsive UI, handling Flask routes, and serving embedded media from a file system would have required significant workarounds or entirely external infrastructure, adding complexity and potential points of failure. Laptops, while offering robust computing power and familiar development environments, are unsuitable for continuous operation in remote, field-based settings due to their size, power demands, and lack of environmental durability. The Raspberry Pi Zero 2 W was ultimately chosen for its unique ability to bridge these gaps. It is compact, affordable, and powerful enough to run a lightweight Flask server while also handling file serving, UI hosting, and analytics generation. Most importantly, it is shared with the camera subsystem, meaning no additional processor is required. This not only reduces system cost, but also eliminates the need for real-time data transmission between separate boards (a task that would have introduced new failure modes, delays, and additional development time).




Descriptor:	RaspberryPi Zero 2 W	ESP32	Laptop/PC
Component:			
Cost	Low (R330)	Very low (R100)	Very high (R5000+)
Power Consumption	Low	Very low	High
Ability to Run Web Server	Yes (Flask, static files)	Limited (micro web server only)	Yes (all stacks)
Media File Handling	Yes (can store and serve files)	No (not suitable for video/images)	Yes
Wi-Fi Support	Built-in	Built-in	Yes (but consumes more power)
Ease of Integration with Overall System	Excellent (shared with camera subsystem, no need for separate data transmission)	Poor (requires separate board and custom transfer protocol)	Good for dev, impractical for field deployment

Table 6.3: Comparison of Processor Options for Hosting the Front-End Subsystem

6.4.2 Implementation

While the Raspberry Pi Zero 2 W served as the actual host for the front-end server, all development and testing of the backend scripts and Flask routes were done using a laptop. This was achieved by using SSH tools such as PuTTY and WinSCP to remotely access the Pi's terminal and make changes directly to the files stored on the device. This setup allowed for rapid testing, debugging, and deployment without needing to physically connect a screen or keyboard to the Pi, making development more efficient and flexible.

Web Server Framework Selection

Framework	Language	Resource Usage
Flask	Python	Very lightweight; minimal memory and CPU usage. Suitable for small, embedded systems.
Django	Python	Heavy; includes built-in ORM, admin interface, and templating engine. Higher RAM and CPU requirements.
Node.js	JavaScript	Moderate to high resource usage depending on module complexity. Runs continuously as an event loop.

Table 6.4: Comparison of Web Frameworks for the Front-End Server

Flask was chosen as the front-end server framework due to its extremely low resource requirements, which aligned well with the project's low power goals. Unlike Django, which includes many unnecessary features like an ORM and admin interface, Flask is simple, fast to deploy, and ideal for serving static files and basic APIs. JavaScript-based options like Node.js was also considered, but they introduce more complexity and generally consume more memory and processing power, which would be unnecessary on the shared Raspberry Pi Zero 2 W. Since Flask runs natively in Python and integrates easily with the rest of the system, it helped keep the server lightweight and efficient, contributing to the overall low-power design.

The Flask server is launched using the following minimal setup:

```

1 from flask import Flask
2 app = Flask(__name__)
3
4 if __name__ == '__main__':
5     app.run(host='0.0.0.0', port=8000)

```

This snippet shows how the application initialises and begins listening for requests. The server is explicitly bound to `0.0.0.0` to allow access from other devices on the network.

Communication Protocol

The front-end user interface communicates with the Flask backend using standard HTTP requests. The built-in Flask routing system was used to define route handlers that return either full HTML pages or JSON data depending on the purpose of the route.

For example, the main interface is loaded by sending a simple GET request to the root URL:

```

1 @app.route('/')
2 def index():
3     return render_template('index.html')

```

This route returns the `index.html` page from the `templates/` folder, which serves as the initial password screen. The same folder also includes other interface pages such as `dashboard.html`, `analytics.html`, and `animal_log.html`, which are only accessible after successful login. When a user opens the web page in a browser, this root route is automatically triggered by the browser's GET request, presenting the user with the authentication prompt before granting access to the rest of the interface.

Data shown on a page, such as detection logs, timestamps, and chart data, is dynamically fetched from the SQLite database during each page request. There is no background polling; instead, the user manually refreshes the page or navigates between tabs, which causes a new Flask route to execute. This ensures that resource usage remains low.

New detections only appear on the front-end if they have first been added to the SQLite database. The camera subsystem stores its output in three folders: `images`, `videos`, and `metadata`, and then triggers the script `import_metadata.py`, which parses new metadata files and inserts entries into `detections.db`. Once this process has occurred, any user who refreshes the page will see the updated results immediately.

Database and Storage Strategy

To support the analytics and animal logs pages and enable persistence of detection events, the front-end subsystem relies on a local file-based storage structure combined with a lightweight SQLite database.

Storage Option	Advantages	Disadvantages
SQLite	<ul style="list-style-type: none"> - Structured and queryable - Lightweight and self-contained - Fast local performance 	<ul style="list-style-type: none"> - Limited to one write operation at a time - Requires schema and basic logic
JSON + Folder System	<ul style="list-style-type: none"> - Simple and human-readable - Easily generated by Python 	<ul style="list-style-type: none"> - No built-in query support - Slower for analytics without aggregation step
CSV / TXT Files	<ul style="list-style-type: none"> - Very simple and portable - No libraries required 	<ul style="list-style-type: none"> - Poor support for relationships or structure - Difficult to filter/sort without loading everything into memory
Firebase / Cloud DB	<ul style="list-style-type: none"> - Real-time sync across devices - Built-in web access and dashboards 	<ul style="list-style-type: none"> - Requires constant internet connection - Potential recurring cost
MySQL	<ul style="list-style-type: none"> - Powerful relational features - Ideal for multi-user environments 	<ul style="list-style-type: none"> - Too heavy for Raspberry Pi - More complex setup and runtime requirements

Table 6.5: Comparison of Storage and Database Options for Detection Data

After comparing all the options, SQLite was chosen because it provided the right balance between structure and simplicity. JSON and CSV files were easy to generate, but they become limiting when needing to sort or filter detection data for the analytics and log pages. Firebase initially seemed like a viable option due to its ease of use and built-in data syncing features. However, the free tier includes only 5GB of storage, with additional usage requiring a paid plan. Since the Raspberry Pi was equipped with a 64GB SD card, a local storage solution offered significantly more capacity at no extra cost.

SQLite is lightweight, worked well with Flask, and didn't require an internet connection or constant setup. It allows everything to be kept on the Pi and still have fast, reliable access to the data.

Detection data generated by the camera subsystem is stored directly on the Raspberry Pi's 64GB SD card in the following structure:

- `static/data/images/` – still frames (JPEGs) extracted from detection videos.
- `static/data/videos/` – short detection clips in MP4 format.
- `static/data/metadata/` – JSON files containing structured metadata for each detection (e.g., timestamp, predicted species, model confidence score).

Each detection has a uniquely indexed filename (e.g., `detection007.jpg`, `detection007.mp4`, `detection007.json`) that ties its media and metadata together. The JSON metadata folder acts as the system's 'source of truth'. A preprocessing script called `import_metadata.py` scans this folder, parses each file, and inserts the relevant fields into the SQLite database. This enables efficient chart generation, filtering, and sorting on the front-end, while maintaining modularity between the camera and interface subsystems. All image, video, metadata, and database files are stored on the Pi's local 64GB SD card. Given that each video is short and each metadata file is only a few kilobytes, this configuration can support thousands of detections before storage becomes a concern. This makes the system robust for remote or unattended deployments where long-term autonomy is required.

6.4.3 Exposing the Flask Sever to the Internet

For the user interface to be accessible remotely, from any device on any network, the Flask server running on the Raspberry Pi needed to be made available over the internet. This requirement posed a challenge, as the Pi typically runs behind a local router in a private IP range and does not have a public IP address.

At first, I explored free tunneling services that could expose the Flask server without requiring changes to router settings or port forwarding:

- **LocalTunnel:** This was the simplest to set up and worked well initially. However, the major limitation was that the randomly generated public URL changed every time the tunnel restarted, which made it impractical for a stable demo or public access.
- **Ngrok:** This service worked reliably and supported custom subdomains, but those features were locked behind a paid plan. The free tier had limited session time and bandwidth, and the automatic disconnection made it unsuitable for a continuously running UI.

After evaluating multiple options, I chose to use **Cloudflare Tunnel (cloudflared)** in combination with a custom domain name purchased through Namecheap. This setup provided the persistent and branded URL at a very low cost. Unlike Ngrok's free tier, Cloudflare tunnels remain active as long as the 'cloudflared' process is running, making it more reliable.

To implement this, the domain name `faunawatch.site` was registered using Namecheap and updated its DNS settings to use Cloudflare's nameservers. On the Raspberry Pi, I installed `cloudflared` and authenticated it to my Cloudflare account. I configured a Cloudflare Tunnel to forward the custom domain to the locally running Flask server (e.g., `localhost:8000`). The tunnel is started automatically on boot, ensuring that the UI remains accessible even after a reboot.

6.5 Front-End Design and Theory

6.5.1 Styling

The interface was styled using **Bootstrap 5**, linked via CDN for simplicity:

```
1 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
```

Bootstrap handled most of the layout, card structure, buttons, and responsive grid design. A small amount of custom CSS was added to ensure better compatibility across different screen sizes, particularly to adjust spacing and scaling for mobile users.

Logo Design and System Name



Figure 6.1: Branding elements used in the front-end interface design

The Fauna Watch logo (*Figure 6.1(a)*) features a minimalist penguin silhouette within a shield, paired with clean, bold typography, which symbolises the system’s overarching mission to protect penguins in their natural habitat. The penguin represents the focal wildlife of the project, while the shield conveys protection and non-invasive monitoring. The colour palette complements the coastal palette derived from the deployment environment, creating brand cohesion.

The name **FaunaWatch** was chosen to reflect the system’s primary goal of monitoring and protecting wildlife activity. The term *fauna* refers to animal life, while *watch* conveys the idea of vigilant observation, visual awareness, and conservation, which are all central to the system’s function at the reserve fence line.

Background Image and Colour Palette

The background image of De Hoop Nature Reserve was chosen to visually immerse users in the exact environment where the system operates. Since the UI displays real camera footage of animals detected in this landscape, using an authentic image helps create a stronger contextual connection and reinforces the purpose of the system, protecting wildlife in this specific, rugged coastal habitat.

The palette in *Figure 6.1(b)* captures key natural elements: vegetation (Soya Bean), rock formations (Sisal), ocean tones (Danube), and sky haze (Tower Gray), to maintain visual harmony and contextual relevance in the front-end design.

6.6 Implementing User Interactivity

6.6.1 Login Prompt and Access Control

To prevent unauthorised access to the detection dashboard, a simple password screen was implemented as the first step when loading the interface. The root route (/) serves the `index.html` page, which contains a basic password form. Upon submission, the password is compared against a stored hash or environment variable, and if correct, the user is redirected to the main dashboard.

This method provides basic security suitable for a field-deployed system with limited external exposure, without requiring a full authentication system.

```

1 @app.route('/', methods=['GET', 'POST'])
2 def login():
3     if request.method == 'POST':
4         if request.form['password'] == 'your_password_here':
5             return redirect('/dashboard')
6         else:
7             return render_template('index.html', error=True)
8     return render_template('index.html')
```

Listing 6.1: Simple login check in Flask

This approach keeps the system lightweight and offline-capable, while still ensuring that only approved users can view sensitive data.



Figure 6.2: Index page showing the password prompt

6.6.2 Download Functionality

To allow stakeholders to review or archive detection data, I implemented download functionality for both images and videos directly from the user interface. This was particularly useful for researchers who might want to manually verify detections or include media in offline reports.

The download feature is enabled by defining a route in `server.py` that uses Flask's built-in `send_from_directory()` function. When a user clicks a *Download* button on the UI, it triggers a GET request to this route with the filename as a parameter.

In the HTML template, each detection entry includes a download button that links to this route. For example, to allow downloading a video file:

```
1 <a href="/download/videos/detection007.mp4" class="btn btn-sm btn-outline-primary">Download</a>
```

This setup ensures a clean separation between the front-end and back-end, while keeping the process secure and intuitive for users.



Figure 6.3: Download button successfully triggering a media file download, confirming correct route setup and file access from the user interface.

6.6.3 Chart Rendering with `Chart.js`

Three types of charts were implemented using the Chart.js library to provide useful visual insights into detection activity:

- **Animal Frequency Pie Chart** – Shows the proportion of detections per species, helping researchers quickly identify which animals are most frequently observed.
- **Detections Over Time Line Chart** – Displays how detection activity changes over time, revealing trends such as spikes in movement or quiet periods.
- **Time-of-Day Activity Heatmap** – Highlights when each animal is most active across a 24-hour period, offering insight into nocturnal and diurnal behaviour patterns.

All charts are embedded using `<canvas>` elements in the HTML and styled with Bootstrap card components.

```
1 <canvas id="activityChart"></canvas>
```

Data is fetched from the Flask backend, passed into the page using Jinja's `tojson` filter, and rendered by Chart.js on the client side.

```
1 labels: {{ pie_labels | tojson }},
2 data: {{ pie_data | tojson }}
```

These analytics update dynamically based on time range filters selected by the user, and they provide immediate, interpretable summaries of activity without requiring access to raw logs.

6.6.4 Filtering Mechanisms on Log and Analytics Pages

Both the Animal Log and Analytics pages include dropdown filters to make the interface more interactive. On the log page, users can filter by species and sort by timestamp. On the analytics page, users can select a time range (e.g., last 24 hours, 7 days, or 30 days) to update the charts.

Filters are handled through simple GET requests, and the Flask backend adjusts database queries based on the selected parameters:

```
1 species = request.args.get('species')
2 range_option = request.args.get('range', '7d')
```



(a) Analytics page displaying detection activity over time using a line chart

(b) Analytics page showing the animal frequency pie chart

Figure 6.4: Screenshots of the analytics page showcasing two key visualisations used to interpret detection data. Both views include interactive filters for species and time range selection.

6.7 Results and Acceptance Testing

To keep the design process as modular as possible, the front-end was initially developed and tested using fake detection data. This approach allowed the interface and database functionality to be implemented independently of the sensing subsystem. Simulated image, video, and metadata files were manually created and stored in the appropriate folder structure on the Raspberry Pi. A set of test JSON files was also generated to mimic real detection events, which were then loaded into the SQLite database using the `import_metadata.py` script. This made it possible to test core features such as filtering, chart rendering, and download functionality before real data from the camera subsystem was available.

ID	Test Description	Test Result
NFAT-01	The interface uses a custom colour palette of less than 5 colours, 1 font style, and less than 10 buttons/inputs (excluding download buttons).	PASS: The system styling is clear and readable and uses 4 colours. There are 2 buttons for the user on the dashboard page, 2 filter/sort inputs on the animal log and analytics pages. (<i>Figures 6.1(b)</i> , <i>B.1</i> , <i>6.4</i> , <i>B.2</i>)
NFAT-02	Headings and sections must be clearly visible and verified by user feedback.	PASS: The website uses labelled cards to differentiate sections on a page. (<i>Figures B.1</i> , <i>6.4</i> , and <i>B.2</i>). User's rated the layout of the UI a 4.14/5, which is a passing grade. (<i>Figure B</i>)
NFAT-03	UI must adjust without distortion on phones, tablets, and laptops.	PASS: The system responds to changing screen size. (<i>Figure 6.2</i> and <i>Figure B.1</i>). The system is accessible on any portable device/PC.
NFAT-04	All components verified to be free or open-source	FAIL: The total recurring cost is \$0.98 (or R17,49) for per year, to maintain the personalised domain name required for remote access via Cloudflare Tunnel. This minimal but nonzero recurring cost must be acknowledged.
FAT-01	UI loads on different devices (e.g., phone, laptop) within 5 seconds.	PASS: Manual Testing showed the UI loaded in under 5 seconds on both a mobile device and a laptop, in different locations across the country. No delays or timeouts were observed.
FAT-02	Manually inserted test detections appear with the correct media and metadata.	PASS: The animal in (<i>Figure B.2</i>) matches the manually uploaded detection metadata. This is confirmed by comparing it to the manually inserted data recorded in the <i>B</i>
FAT-03	Clicking the download button saves the correct file with the appropriate filename.	PASS: The user is able to download media from the site and it saves with the appropriate filename. (<i>Figure 6.3</i>).
FAT-04	Line and pie charts render correctly and match the known manually uploaded detection data distributions.	PASS: See <i>Figure 6.4(b)</i> and <i>6.4(a)</i> . This is confirmed by comparing the data plots to the manually inserted data recorded in the <i>B</i>
FAT-05	A new detection added to the back-end appears in the UI without restarting the server.	PASS: A new image, video, and JSON file were added to the metadata folder and then loaded into the database using the <code>import_metadata.py</code> script. The entry appeared on the Animal Log page and updated the charts immediately after refreshing the page. No server restart was needed.
FAT-06	Correct password grants access. Incorrect password shows error.	PASS: System grants access (redirects user to dashboard route) only when the correct password is entered and displays an appropriate error message when incorrect (<i>Figure B.3</i>).
FAT-07	System remains under 10% CPU usage and low memory load during idle periods; no background services run unless triggered by user actions.	PASS: System idled with only Flask server active; CPU usage remained below 1% and memory usage stable. No background processes observed beyond system and server tasks. (<i>Figure B.4</i>)

Table 6.6: Acceptance Tests with Results

6.8 Conclusion

The front-end subsystem successfully meets all defined functional and non-functional requirements. Hosted on the Raspberry Pi Zero 2 W, it provides password-protected access to a responsive web interface for viewing, filtering, and downloading detection data. Analytics are clearly visualised using dynamic charts, and all components were tested and validated using both simulated and real data. The only test that failed was a non-functional one, due to the small annual cost of maintaining the custom domain for remote access.

Chapter 7

Imaging Subsystem

By: Triss Naidoo - NDXTRI021

7.1 Subsystem introduction

To fulfil our goal of enhancing the security of the fence, it is necessary to detect the location and nature of each breach (or near-breach) that occurs. In the previous subsystem, a PIR and laser break-beam system was employed to trigger the Imaging subsystem when a predator was detected.

In the imaging subsystem, a Pi-Cam is used to capture an image followed by a short video each time the PIR detects movement. In order to operate both during the day and at night, IR LEDs are used to illuminate the scene. Each time the Imaging subsystem captures data, it then classifies the image using an [ANN](#).

7.2 Requirements Analysis

7.2.1 User Requirements

The user requirements covered by this section are:

Req ID	Description
UR01	The system should be able to function on the 12 VDC, 1A supply that is available.
UR02	The system should be able to capture imagery and video of animals near the fence.
UR03	The system should be able to do this regardless of the time of day.
UR04	Imagery of incidents at the fence must be classified by animal type.

Table 7.1: Imaging Subsystem User Requirements

7.2.2 Functional Requirements

Req. ID	Description
FR01	System must contain a buck converter to convert the 12 VDC to the voltage required by the microcontroller.
FR02	The microcontroller must have a low power draw to keep system power below 12 W.
FR03	System must have a microcontroller capable of video processing and running a neural network.
FR04	The microcontroller must interface with a camera that can operate at night.
FR05	LEDs are required to give illumination for the camera to operate at night.
FR06	LEDs must be driven by MOSFETs to prevent overloading of microcontroller GPIO.
FR07	Neural Network must be able to classify animals typically found in the region.
FR08	Neural Network must be able to operate on differing light conditions and scale/position of animals depending on distance and still classify correctly.

Table 7.2: Imaging Subsystem Functional Requirements

7.2.3 Specifications

Spec. ID	Description
SP01	Buck converter must handle 12 VDC input to 5 VDC 2.4 A output.
SP02	Microcontroller must use less than 1.5 A to provide power overhead for other components.
SP03	Microcontroller must handle FHD 30 fps video recording and running an ANN that can process an FHD image.
SP04	Camera module must use an interfacing method supported by microcontroller.
SP05	Camera sensor must be sensitive to IR light at 940 nm.
SP06	Infrared LEDs must illuminate scene at $> 0.1 \text{ W/m}^2$
SP07	IR LEDs must have current limiting resistors that keep $I < I_{max}$
SP08	MOSFETs must have $V_{GS_{th}} < 3.3 \text{ V}$ due to GPIO voltage
SP09	MOSFETs must have $I_{DS} > 2I_{max_{LED}}$
SP10	Neural Network classes must include: honey badger, mongoose, caracal, genet, baboon.
SP11	Neural Network accuracy must be > 0.9 on training data
SP12	Neural Network accuracy must be > 0.7 on real data

Table 7.3: Imaging Subsection Specifications

7.2.4 Acceptance Testing Procedures

ATP ID	Test Procedure	Criteria
ATP01	Connect buck converter to 12 VDC input with current limit $I_{max_{source}} > 1 A$ and measure V_{out} with multimeter	$V_{out} = 5 V \pm 5 \%$
ATP02	Load output of buck converter with 2Ω resistance.	$I_{max_{converter}}$ not triggered.
ATP03	Power microcontroller and measure current draw using an ammeter	$I < 1.5 A$
ATP04	Run a full HD (FHD) 30 fps video recording test then execute an artificial neural network (ANN) model.	Recording successfully saved, network successfully run.
ATP05	Verify camera output on microcontroller	Image successfully saved.
ATP06	Shine 940 nm IR source at camera and observe whether visible in video preview.	IR visible on camera preview
ATP07	Check IR LED radiance on datasheet and calculate total radiance of IR LED array.	radiance $> 0.1 W/m^2$
ATP08	Measure current through IR LEDs using ammeter.	$I < I_{max}$.
ATP09	Apply 3.3V to MOSFET gate. at drain connect 100Ω resistor into 12 V, connect source to ground. Measure V_{DS} with voltmeter.	$V_{DS} < 12 V$
ATP10	Same setup as ATP09. Measure I_D using ammeter.	$I_D > 100 mA$
ATP11	Dataset must include classes: honey badger, mongoose, caracal, genet, and baboon.	All classes included in dataset.
ATP12	Neural Network performs $>90\%$ accuracy on training data	$Acc > 0.9$
ATP13	Find 10 images of each animal. Display image on laptop screen. Point camera at screen and take image. Classify using neural network.	$Acc > 0.7$

Table 7.4: Acceptance Test Procedures for Imaging Subsystem

7.2.5 Traceability Matrix

UR	FR	SP	ATP
UR01	FR01	SP01	ATP01
UR01	FR01	SP01	ATP02
UR02	FR02	SP02	ATP03
UR02/04	FR03	SP03	ATP04
UR02	FR04	SP04	ATP05
UR03	FR04	SP05	ATP06
UR03	FR05	SP06	ATP07
UR03	FR06	SP07	ATP08
UR03	FR06	SP08	ATP09
UR03	FR06	SP09	ATP10
UR04	FR07	SP10	ATP11
UR04	FR07	SP11	ATP12
UR04	FR08	SP12	ATP13

Table 7.5: Traceability Matrix for Imaging Subsystem

7.3 Design choices

7.3.1 Hardware Design

Microcontrollers

Microcontroller	Current Draw	1080p 30fps?	Neural Network Capable
ESP32	240 mA	No	Maybe
STM32F4	160 mA	No	Maybe
RPi Zero 2 W	0.5 A	Yes	Yes
RPi 4	3 A	Yes	Yes

Table 7.6: Microcontroller Comparison

Based on the above, the Raspberry Pi Zero 2 W was chosen as the micro-controller for this system. This microcontroller in addition to meeting both the power requirements and the processing speed requirements, is also able to run PiOS, a linux OS. This makes programming it potentially easier than a microcontroller such as the ESP32 or STM32F4, since code is written directly on it rather than having to be flashed from a separate device each time.

7.3.2 Buck Converter

Name	Price (R)	Available?	I_{\max} (A)
HKD DC/DC 12V TO 3.3V/5V/12V	13	No	0.8
BMT DC/DC BUCK ADJ 5-80V 0.8A LG	35	No	0.8
BMT ADJ DC/DC MODULE 3A+DISPLAY	42	No	3
BMT DC/DC BUCK/LIPO CHG 1.25-36V	55	Yes	5
BMT DC/DC ADJ BUCK MOD 1.2-35V	75	Yes	9

Table 7.7: DC-DC Converters Comparison

The BMT DC/DC BUCK/LIPO CHG 1.25-36V converter was chosen because it supports up to 5 A (greater than the 2.4A required). It was also in stock with suppliers, unlike the BMT ADJ DC/DC MODULE 3A+DISPLAY which is 13 rand cheaper. Compared to the BMT DC/DC ADJ BUCK MOD 1.2-35V, the BMT DC/DC BUCK/LIPO CHG 1.25-36V is cheaper, and the additional current allowed by the BMT DC/DC ADJ BUCK MOD 1.2-35V is not necessary or beneficial for our purposes. It maintains adjustable voltage capability, allowing flexibility in system design.

Camera

Cam Module	Price (ZAR)	Resolution	Available?	IR Filter?
Sony IMX708	816.50	4608x2592	Yes	No
OmniVision OV5647	135.00	2592x1944	Yes	Yes ¹
OmniVision OV7670-V2	101.20	640x480	Yes	Yes

Table 7.8: Camera Module Comparison

The OmniVision OV5647 was selected over other options for many reasons. It is far more cost-effective (R135) than the Sony IMX708 (R816.50). Its resolution (2592x1944) is more than sufficient for recording FHD as defined in the requirements. And its lack of an IR filter was a welcome surprise for night vision purposes.

7.3.3 IR LEDs

Name	λ_P (nm)	Radiance (mW/sr)	Viewing Angle ($^\circ$)	I_{max} (mA)
L-34SF4C	880	20	50	50
L-34F3C	940	20	50	50
L-53F3C	940	30	30	50
KM-4457F3C	940	3	150	50

Table 7.9: IR LED Comparison

The IR LED with the highest radiance was the L-53F3C, with 30 mW/sr. This is a result of its narrow viewing angle. Due to low power requirements, this is an acceptable trade off in order to keep power consumption low while not using too many LEDs.

Since there is a 12 VDC power supply, the LEDs were arranged in 4 groups of 5 series LEDs, to reduce the number and size of resistors required. 4x 150 Ω resistors were used in this design. Further, 2 sets of LEDs were powered by each MOSFET, to reduce the number of MOSFETs needed.

7.3.4 Software Design

Training Data

It is difficult to obtain a dataset containing all the animals from the list of predators, with enough samples per class. For this reason, it is necessary to combine datasets to cover all the required classes.

Network Architecture

Dataset Augmentation

Image Pre-Processing

7.4 Implementation

7.4.1 Hardware Design

Pi Connectivity

Only a few GPIO pins on the Pi, as well as the microSD slot, and the 22 pin ribbon connector were needed—most GPIO pins were not needed, nor were the micro-USB or mini-HDMI connectors. Because of this, it was possible to only use GPIO pins on one row, leaving the other side unused. This allowed the Pi to lay parallel to the breakout board used for connectivity from it, saving space inside the case.

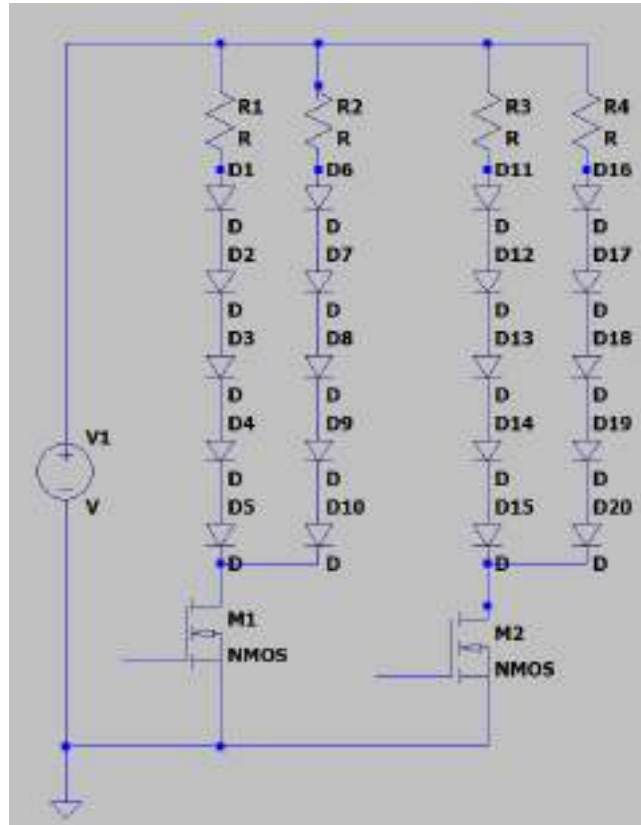


Figure 7.1: IR LED Layout

Buck Converter

The buck converter has 4 connections: a positive input and GND, and a positive output and GND. To ensure the converter was correctly configured to output 12 V, a power supply was set to 12 V and connected to the input of the buck converter. The voltage control potentiometer was then turned until a multimeter read 5 V on the output. This was also verified with an oscilloscope which was used to ensure ripple was acceptable when loaded with a 330Ω resistor (a smaller resistor was not used due to resistor power limits).

The outputs of the converter were then connected to the 5V and GND lines of the Pi.

IR LED layout

The IR LEDs were arranged in 4 groups, each consisting of 5 IR LEDs and a 150Ω resistor in series. These 4 groups were then connected as shown in Figure 7.1. This allowed fewer MOSFETs to be used, since while still keeping 2 different "zones" in the event that a MOSFET failed (and by that measure, 4 different parallel branches in the event that an LED failed).

Combined Hardware Design

7.4.2 Subsystem Triggering

As detailed in the Predator Detection subsystem, the camera is triggered by a UART RX line. UART is referred to as "idle high" meaning that when no data is being sent, the line is held high. On the

Raspberry Pi Zero 2 W, GPIO 15 allows UART RX. However since this signal is very simple, it does not actually need to be configured as a UART RX line.

In order to trigger the Imaging subsystem, GPIO 15 is configured as a Button, as seen in 7.2. This method includes an internal pull-up resistor, meaning the button is idle high.

```
from gpiozero import Button, DigitalOutputDevice
from picamera2 import PiCam

# Initialize GPIO for trigger and IR LED driver MOSFETs
trigger = Button(15)
led1 = DigitalOutputDevice(11)
led2 = DigitalOutputDevice(16)
```

Figure 7.2: RPi Zero 2 W GPIO Configuration

7.4.3 IR LED triggering

7.4.4 Camera image and video recording

7.4.5

7.5 Testing

The acceptance tests outlined in Table 7.10, were carried out, producing the following results. As can be seen, all hardware related tests were a pass. However, the neural network and machine vision tasks had several failed acceptance tests.

Test ID	Pass Criteria	Test Result
ATP01	$V_{out} = 5 V \pm 5 \%$	Pass - Chosen buck Converter has adjustable output voltage.
ATP02	$I_{max_{converter}}$ not triggered.	Pass - Chosen buck converter has adjustable output current, max 5A
ATP03	$I < 1.5 A$	Pass. Highest reading under load was $I = 0.72 A$
ATP04	Recording successfully saved, network successfully run.	Pass. Recording saved. CNN network runs fine.
ATP05	Image successfully saved.	Pass. Image saved.
ATP06	IR visible on camera preview.	Pass. IR visible to camera.
ATP07	Radiance $> 0.1 W/m^2$.	Pass. However, max distance = 6 m
ATP08	$I < I_{max}$.	Pass. $I = 37 mA$, $I_{max} = 50mA$
ATP09	$V_{DS} << 12 V$.	Pass.
ATP10	$I_D > 100 mA$.	Pass.
ATP11	All classes included in dataset.	Fail. Only honey badger was tested
ATP12	$Acc > 0.9$ on training data.	Fail. Accuracy = 0.8
ATP13	$Acc > 0.7$ on real-world classification.	Due to camera configuration issues, no real-world classification was attempted

Table 7.10: Acceptance Test Procedures for Imaging Subsystem

7.6 Conclusion

Chapter 8

Conclusions

In conclusion, the project prototype demonstrated strong performance and validated key concepts, while also revealing areas for improvement that will be addressed in future iterations.

The detection/sensing subsystem, which combines a digital PIR sensor and a laser break-beam, successfully met all design objectives and offers a scalable, low-power solution for perimeter protection. While the analog receiver circuit showed potential, it was not finalized due to time constraints. Future development should focus on refining the analog design.

The housing subsystem met all key requirements, including environmental protection, passive cooling, modularity, and precise component alignment. While full field deployment was not conducted, the prototype confirmed the enclosure's suitability for harsh conditions. Future improvements should focus on proper heatsink grounding and evaluating ABS as a more durable and weather-resistant enclosure material.

The front-end subsystem, comprising a Flask-based web server and responsive Bootstrap interface, successfully fulfilled all functional and aesthetic design requirements. It enables real-time display of detection data, model confidence, and media playback while maintaining a clean, mobile-friendly user experience. If the project budget increases, a key recommendation would be to migrate to an online database solution such as Firebase to support scalability and remote access. Additionally, deploying more camera subsystems would allow for richer, more insightful analytics on the front end, enhancing situational awareness and overall system effectiveness.

Bibliography

- [1] Zachariah Peterson, “UART vs. SPI vs. I2C: Routing & Layout Guidelines,” 2020. Accessed: 25 May 2025.
- [2] Advanced Plastiform, Inc., “Using ABS vs Acrylic in Plastic Manufacturing,” 2025. Accessed: 25 May 2025.
- [3] Mammoth Security Inc., “Types of Motion Sensors,” 2023. Accessed: 25 May 2025.
- [4] Mail & Guardian, “Killing spree: Honey badger wipes out 11 endangered penguins at Cape reserve,” 2024. Accessed: 6 May 2025.
- [5] D. Poole, I. McKillop, G. Western, P. Hancocks, and J. Packer, “Effectiveness of an electric fence to reduce badger (meles meles) damage to field crops,” *Crop Protection*, vol. 21, pp. 409–417, 06 2002.
- [6] A. Chwalibog, “The honey badger in south africa: Biology and conservation,” *International International Journal of Avian & Wildlife Biology*, vol. 2, 07 2017.
- [7] A. Johnson, *The Effects of Tactile and Visual Deterrents on Honey Badger Predation of Beehives*. PhD thesis, 02 2019.
- [8] M. Clouse, *Effects of an Electronic, Motion-Activated Scarecrow on Foraging Time in Nocturnal Vertebrate Crop Pets in South Africa*. PhD thesis, 2016.
- [9] A. Mishra and K. K. Yadav, “Smart animal repelling device: Utilizing iot and ai for effective anti-adaptive harmful animal deterrence,” *Bio Web of Conferences*, vol. 82, pp. 05014–05014, 01 2024.
- [10] J. M. Gilsdorf, S. E. Hygnstrom, and K. C. VerCauteren, “Use of frightening devices in wildlife damage management,” *Integrated Pest Management Reviews*, vol. 7, pp. 29–45, 2002.
- [11] K. King, R. Wallis, A. Wallis, A. Peucker, and D. Williams, “Successful protection against canid predation on little penguins (eudyptula minor) in australia using maremma guardian dogs: ‘the warrnambool method’,” 2015.
- [12] R. Reddy Kommasani, K. Babu Malla, S. Vasam, T. Bhukya, and A. Karlapudy, “Solar based fencing and animal detection system for agricultural fields,” 2019.

- [13] M. E. Smith, J. D. C. Linnell, J. Odden, and J. E. Swenson, “Review of methods to reduce livestock depredation ii. aversive conditioning, deterrents and repellents,” *Acta Agriculturae Scandinavica, Section A - Animal Science*, vol. 50, pp. 304–315, 12 2000.
- [14] B. S. Africa, “Honey badger wreaks havoc at de hoop’s new african penguin colony,” 04 2024.
- [15] ScienceDirect, “Optical Filter - an overview,” 2025. Accessed: 25 May 2025.
- [16] ChineseLens, “What Are Optical Filters and Their Types,” 2025. Accessed: 25 May 2025.
- [17] Siyabona Africa, “De Hoop Nature Reserve Travel Guide,” 2025. Accessed: 8 May 2025.
- [18] Rittal UK, “IP or Design for Outdoor Enclosures: Sometimes Less Really Can Be More,” 2023. Accessed: 25 May 2025.
- [19] Stockwell Elastomerics, “IP Ratings for Sealed Enclosures,” 2023. Accessed: 25 May 2025.
- [20] Arrow Electronics, “Raspberry Pi 4 Cooling Solutions Comparison,” 2024. Accessed: 25 May 2025.
- [21] element14 Community, “Raspberry Pi 3 Cooling: Heat Sink Ideas,” 2024. Accessed: 25 May 2025.
- [22] Raspberry Pi Forums, “To prevent overheating, all Raspberry Pi models will throttle at a temp of 85°C,” 2025. Accessed: 25 May 2025.
- [23] RaspberryPi.dk, “Heatsink for Raspberry Pi,” 2025. Accessed: 25 May 2025.
- [24] Communica South Africa, “SK63-37,5SA Extruded Heatsink for TO-3,” 2025. Accessed: 25 May 2025.
- [25] Hypex Electronics BV, “AN: Heatsinking Requirements of Class D Amplifiers,” 2016. Accessed: 25 May 2025.
- [26] Texas Instruments, “Understanding Thermal Dissipation and Design of a Heatsink,” 2011. Accessed: 25 May 2025.
- [27] Analog Devices, “MT-093 Tutorial: Thermal Design Basics,” 2009. Accessed: 25 May 2025.
- [28] IoT Insider, “Considering the Energy Consumption of a Raspberry Pi,” 2024. Accessed: 25 May 2025.
- [29] EEPowers Forum, “Heat Sink Questions,” 2025. Accessed: 25 May 2025.
- [30] Dawson, J. F. and Marvin, A. C. and Porter, S. J. and Nothofer, A. and Will, J. E. and Hopkins, S., “The Effect of Grounding on Radiated Emissions from Heatsinks,” in *IEEE International Symposium on Electromagnetic Compatibility (EMC)*, (Montreal, Canada), pp. 1248–1252, IEEE, 2001. Accessed: 25 May 2025.
- [31] EEVblog Electronics Community Forum, “Heatsink for multiple devices, floating, grounded or connected?,” 2021. Accessed: 25 May 2025.

Appendix A

Appendix

A.1 GA Criteria - Imaan Shaik

A.1.1 Predator Detection and Housing Subsystems

Student Number: SHKIMA004	Where met
GA3: Engineering Design	Successfully designed and implemented the predator detection system for reliable triggering of the imaging subsystem. Also designed the enclosure to house and cool other components, ensuring durability, environmental protection, and scalability. See Sections 4 and 5.
GA7: Sustainability and Impact of Engineering Activity	Participated in D-School to develop a safe and eco-friendly system. Designed a non-invasive enclosure and detection setup that avoids harming surrounding wildlife.
GA8: Individual, Team and Multi-Disciplinary Working	Actively participated in all D-School and team activities. Coordinated with camera and UI subsystem owners to ensure proper detection triggering. See MS Teams for collaboration records.
GA10: Engineering Professionalism	Attended all D-School sessions, booked lecturer consultations, and met all deadlines including final report and presentation.

Table A.1: GA Criteria summary for SHKIMA004 (Imaan Shaik) — Predator Detection and Housing

A.2 GitHub Repo

link to repo: <https://github.com/Imaanshaik/EEE4113F-Group-18-2025/tree/main>

Table A.2: Bill of Materials

Subsystem	Part	Supplier	Price (ZAR)
Triss	1x Raspberry Pi Zero 2 W	Pishop	329.90
Triss	1x HKD RASPBERRY PI CAMERA MOD 5MP	Communica	135.00
Triss	20x L-53F3C Round Infrared Emitting Diode	Communica	72.40
Triss	1x BMT DC/DC BUCK/LIPO CHG 1.25-36V	Communica	55.00
Triss	2x VN0300L	Communica	22.32
Triss	0.5x Experimental Board Strip Grid 100x300mm	Communica	40.00
Triss	0.5x HKD RIBBON JUMPER 40W M/F 15CM	Communica	11.50
Triss	2x 711041 OPEN FRAME HEADER 2,54MM 90DEG DIL 40 WAY (2 ROW X 20)	Communica	16.56
Triss	2x 711400 SOCKET DIL 40WAY R?A PCB 2,54MM	Communica	25.58
Imaan	PIR sensor PCB HC-SR501	Communica	25.00
Imaan	HKD LASER TRANSMITTER MOD KY-008	Communica	13.00
Imaan	HKD LASER RECEIVE MOD FOR KY-008	Communica	25.00
Imaan	HKD DC/DC 12V TO 3.3V/5V/12V	Communica	13.00
Imaan	ESP32-Wroom_32	Communica	115.00
Imaan	Perspex for enclosure	UCT	70.00
Imaan	Perspex glue	Communica	14.00
Imaan	Heat sink for Pi x2	Communica	30.00
Imaan	Heat sink for box	Communica	59.20
Imaan	heatsink thermal paste	Communica	28.00
Imaan	high-temp silicone gasket sealant	takealot	90.00
Imaan	3D printing supports	UCT	50.00
Safiya	64 GB SD card	Incredible Connection	129.00
Safiya	Domain name registration	Namecheap	17.49
Total Cost			1386.95

A.3 Predator Detection System

Sensor Type	Detection Principle	Advantages	Limitations
PIR (Passive Infrared)	Detects changes in infrared radiation from warm bodies (e.g., animals)	<ul style="list-style-type: none"> • Low power • Cost-effective • Excellent for detecting humans or animals 	<ul style="list-style-type: none"> • Affected by temperature shifts and sunlight • Requires line-of-sight
Ultrasonic	Emits and receives high-frequency sound waves to detect movement via reflection timing	<ul style="list-style-type: none"> • Detects motion in 3D space • Works in total darkness 	<ul style="list-style-type: none"> • Susceptible to wind, soft objects, and ambient noise • Higher false positive rate outdoors
Microwave Radar	Emits microwave signals and detects motion via Doppler shift	<ul style="list-style-type: none"> • Detects through thin materials (e.g., plastic) • Reliable in fog, dark, or obscured areas 	<ul style="list-style-type: none"> • Can be too sensitive (detects through walls) • Prone to interference if unshielded

Table A.3: Comparison of Different Motion Sensors: PIR, Ultrasonic, and Radar [3]

Filtering Technique	Description and Purpose
Debouncing	Eliminates brief signal spikes or noise by ignoring inputs shorter than a defined duration which may result from wind or electrical transients.
Pulse Stretching	Extends a valid HIGH signal for a fixed window, ensuring that rapid flickers are interpreted as a single detection event.
Minimum Active Duration	Requires the signal to remain HIGH for a minimum time before confirming detection, reducing sensitivity to quick, irrelevant fluctuations.
Cooldown Timer	Introduces a delay after a valid detection during which further triggers are ignored. This avoids redundant triggers from continued motion or drifting objects.

Table A.4: PIR Signal Conditioning Techniques Implemented in Software

A.4. Designing a laser receiver circuit to replace the module

Factor	PIR Only	Laser Only	PIR + Laser
False Positive Rate	High – triggered by heat, wind,	Moderate – affected by insects, dust	Low – requires both sensors to be triggered, reducing noise
Night-Time Reliability	Reliable – detects warm bodies	Reliable – laser beam unaffected by lighting	Highly reliable under low-light conditions
Daylight Reliability	Can be affected by sunlight and ambient heat	Red laser may scatter in direct sunlight	High – combined logic mitigates individual sensor weaknesses
Security Accuracy	Moderate – prone to false alarms	Moderate – may miss brief crossings	High – dual validation improves detection confidence
Power Consumption	Low – digital output, no emissions	Low – continuous current draw (30 mA)	Efficient – laser pulsed, camera triggered only on valid events
Suitability for Predator Detection	May trigger on non-threats (wind, penguins)	May miss fast-moving threats	Highly suitable; detects confirmed threats crossing beam path

Table A.5: Comparison of Predator Detection Strategies: PIR, Laser, and Combined

Table A.6: UART pin mapping between ESP32 and Raspberry Pi

ESP32 Pin	Raspberry Pi GPIO	Function
GPIO17 (TX)	GPIO15 (RX)	UART data (ESP → Pi)
GND	GND	Common ground reference

A.4 Designing a laser receiver circuit to replace the module



Figure A.1: Breadboard prototype of the receiver circuit with phototransistor, LDR, and LM393 comparator and LM324 op-amp.

A.5 Testing camera trigger implementation

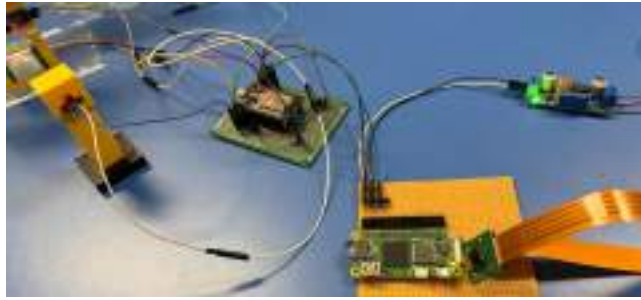


Figure A.2: Full system integration test with ESP32, Raspberry Pi, Pi camera, PIR, laser transmitter and receiver modules

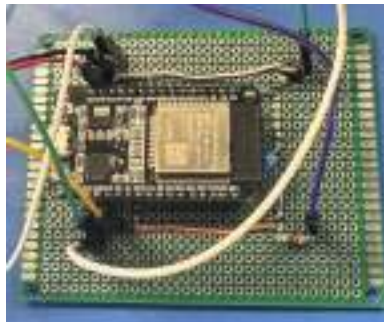


Figure A.3: ESP32 with PIR and Laser filtering circuits soldered on a veroboard.

A.6 Hardware filtering the Laser Rx Module

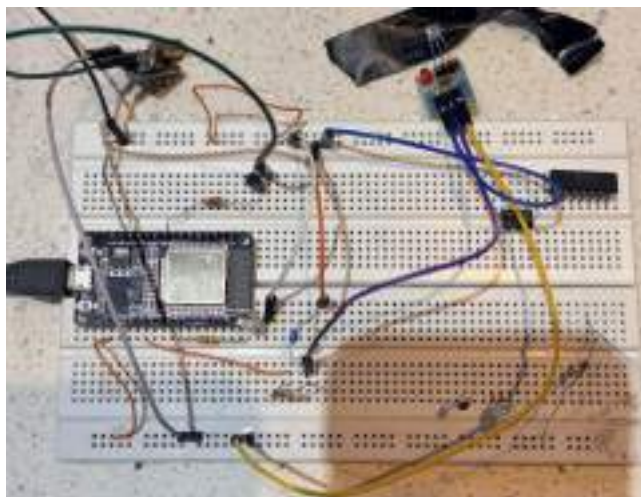


Figure A.4: Breadboard test of low pass filtered laser module



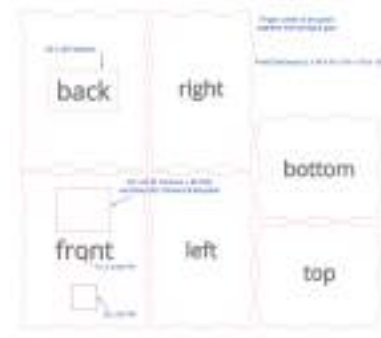
Figure A.5: Scalability concept for the laser detection subsystem. Multiple laser beams can be emitted from a single enclosure (reflected off mirrors) to detect motion at various heights and positions. This concept aims to improve detection reliability against agile predators like honey badgers.

A.7 Housing System

A.7.1 Enclosure Fabrication



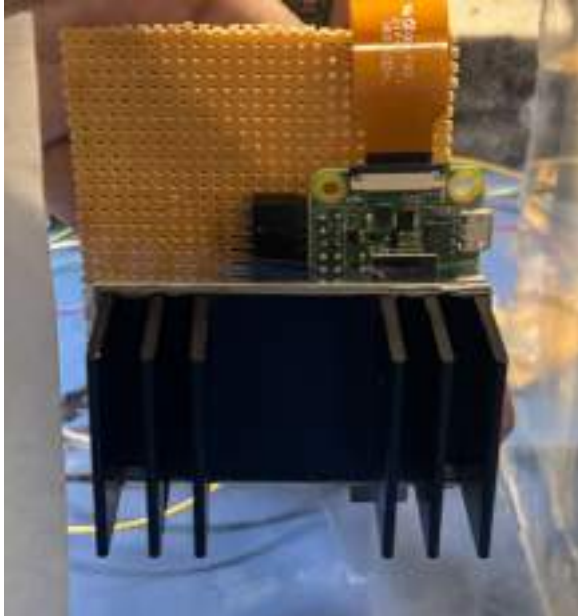
(a) Laser-cut Perspex enclosure before assembly



(b) Laser cutting layout with measured slots for all sensors

Figure A.6: Design and physical build stages of the Perspex enclosure.

A.7.2 Thermal Management System



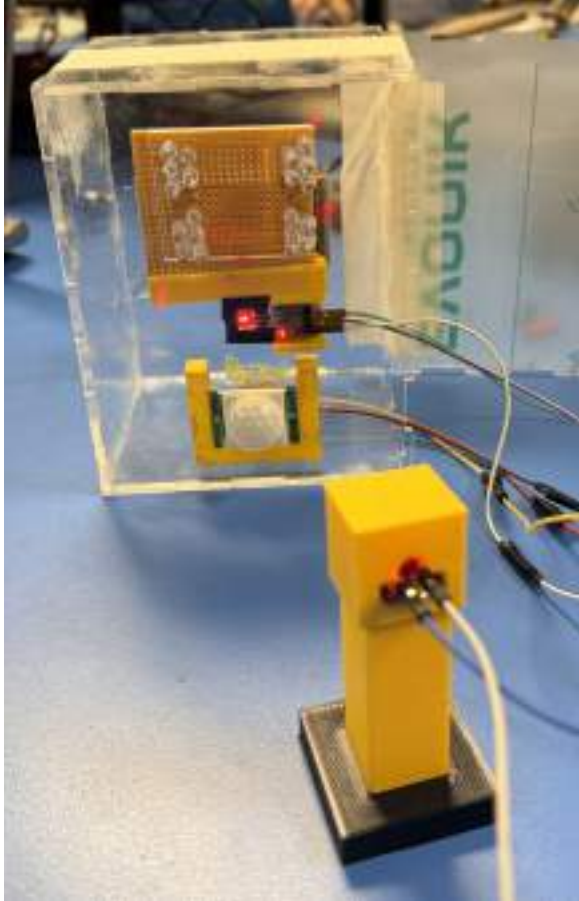
(a) Raspberry Pi mounted onto dual heatsink stack



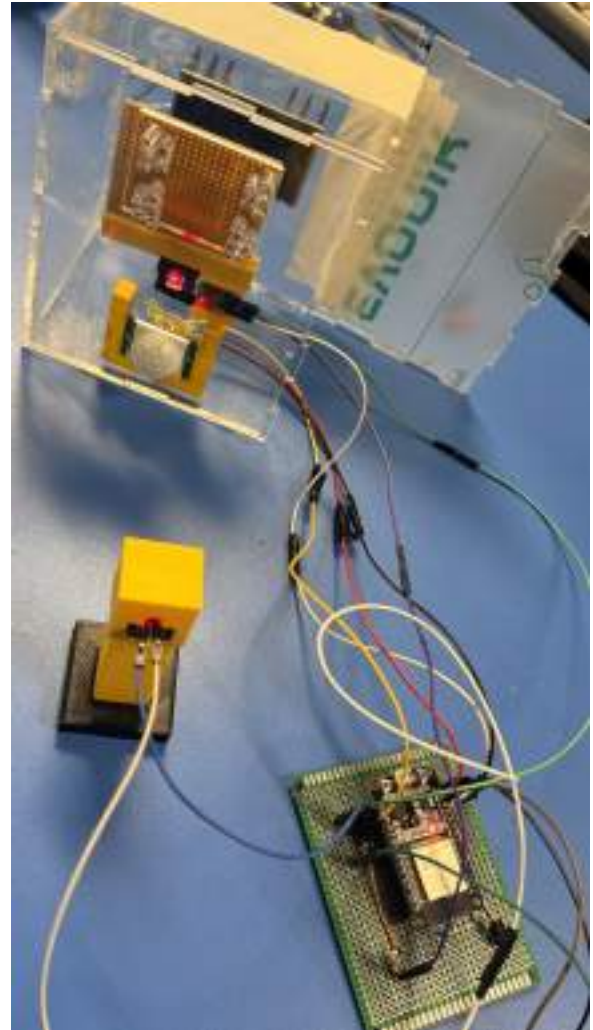
(b) Interlocked heatsinks attached to enclosure for internal cooling

Figure A.7: Passive heatsink cooling system embedded in rear wall.

A.7.3 Testing and Subsystem Integration



(a) Laser alignment testing between TX and RX units



(b) Fully wired enclosure with IR camera and PIR. Camera was not yet active, but dimensions were verified.

Figure A.8: Subsystem testing and functional verification.



Figure A.9: Scalability of the enclosure system along the perimeter fence. Photos provided by BirdLife Stakeholder slides. These images depict the different possible employments of the system, showing its adaptability. It would likely be deployed at a height near the ground to detect honeybadgers. Top-left: Real deployment at De Hoop Nature Reserve. Top-right: Detection of honey badger at night. Bottom-left: Linear alignment of multiple enclosures using a laser break-beam. Bottom-right: Close-up view of enclosure pair with laser beam.

Appendix B

Front-End Subsystem

B.0.1 GA Criteria

Student Number: MXXSAF002	Where Met
GA3: Engineering Design	Designed and implemented the full front-end user subsystem, including UI layout, page routing, and data visualisation. See Sections 6.4 , 6.5 and 6.6 . Participated in all d-school activities.
GA7: Sustainability and Impact of Engineering Activity	Selected lightweight tools like Flask and SQLite to reduce resource usage on embedded hardware. Developed an offline-capable system using a Raspberry Pi Zero 2 W with a 64GB SD card to ensure long-term operation in remote wildlife areas without cloud dependencies. See Section 6.4 . D-school sustainability considerations were also explored.
GA8: Individual, Team and Multi-Disciplinary Working	Collaborated closely with the sensing and camera team to agree on shared data formats and file structures. Integrated the interface with their simulated output to allow parallel development. Contributed to meetings, file sharing, and report consistency across subsystems. Communicated actively in the Teams channel and in-person meetings.
GA10: Engineering Professionalism	Attended all group meetings, consultations, and design reviews. Completed the final report, participated in system integration, and successfully presented the subsystem during evaluation. All deadlines met with tested and documented features. Communication was professional throughout the project.

Table B.1: GA Criteria Mapping for Front-End Subsystem (Safiya Mia)

B.0.2 Pictures from Testing



(a) Dashboard on a mobile screen



(b) Full dashboard layout on desktop

Figure B.1: Responsive design of the dashboard interface, showing key system stats and recent detections. The layout adapts seamlessly between mobile and desktop views.



(a) Animal Log on a mobile screen



(b) Animal Log table view on Mobile

Figure B.2: Screenshots of the Animal Log interface displaying detection records, metadata, and download options.

ID	Timestamp	Animal Name	Latitude	Image Path	Video Path
1	2021-05-01 00:00:00	Whisper	0.00	download01.jpg	video01.mp4
2	2021-05-01 00:00:01	Whisper	0.01	download02.jpg	video02.mp4
3	2021-05-01 00:00:02	Whisper	0.02	download03.jpg	video03.mp4
4	2021-05-01 00:00:03	Ugg	0.03	download04.jpg	video04.mp4
5	2021-05-01 00:00:04	Ugg	0.04	download05.jpg	video05.mp4
6	2021-05-01 00:00:05	Ugg	0.05	download06.jpg	video06.mp4
7	2021-05-01 00:00:06	Heavy Breeze	0.06	download07.jpg	video07.mp4
8	2021-05-01 00:00:07	Heavy Breeze	0.07	download08.jpg	video08.mp4
9	2021-05-01 00:00:08	Whisper	0.08	download09.jpg	video09.mp4
10	2021-05-01 00:00:09	Ugg	0.09	download10.jpg	video10.mp4
11	2021-05-01 00:00:10	Whisper	0.10	download11.jpg	video11.mp4
12	2021-05-01 00:00:11	Heavy Breeze	0.11	download12.jpg	video12.mp4
13	2021-05-01 00:00:12	Ugg	0.12	download13.jpg	video13.mp4
14	2021-05-01 00:00:13	Whisper	0.13	download14.jpg	video14.mp4
15	2021-05-01 00:00:14	Heavy Breeze	0.14	download15.jpg	video15.mp4
16	2021-05-01 00:00:15	Ugg	0.15	download16.jpg	video16.mp4
17	2021-05-01 00:00:16	Whisper	0.16	download17.jpg	video17.mp4
18	2021-05-01 00:00:17	Heavy Breeze	0.17	download18.jpg	video18.mp4
19	2021-05-01 00:00:18	Ugg	0.18	download19.jpg	video19.mp4
20	2021-05-01 00:00:19	Heavy Breeze	0.19	download20.jpg	video20.mp4

Figure B.5: Snapshot of the manually entered data within detection.db used for testing

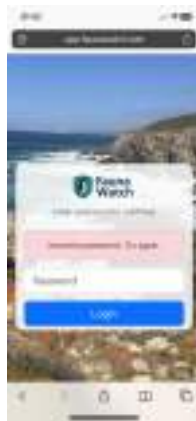


Figure B.3: Error message displayed when an incorrect password is entered on the index page/login screen.



Figure B.4: CPU usage snapshot from the top command while the system is idle.



(a) Survey feedback indicating that the format of the displayed data and media was clear and understandable, with an average rating of 4.36 from 14 users. (b) User responses rating the layout and design of the interface as intuitive and user-friendly, averaging 4.14 stars across all submissions.



(c) User survey results showing high responsiveness of the interface when interacting with features such as downloading media or applying filters. The average rating was 4.64 out of 5.

Figure B.6: User feedback on interface responsiveness, clarity, and usability.