Penguin Patrol Alert System

An Alerting Solution for Penguin Protection at De Hoop

Prepared by:
Ethan Faraday, Aaron Isserow, and Emanuele Vichi

Prepared for:
EEE4113F
Department of Electrical Engineering
University of Cape Town

May 25, 2025

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I 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 my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

May 25, 2025

Ethan Faraday, Aaron Isserow, Emanuele Vichi

Date

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Context

African penguins are a seriously endangered species. Their numbers are rapidly declining at alarming rates, which calls for additional protocols to be put in place to help prevent further loss. One major factor contributing to their decline is the threat posed by land-based predators. These predators are a serious risk to conservation efforts, especially when trying to establish new colonies like the one at De Hoop Nature Reserve. Although various protective measures such as fencing have been implemented, they have not always produced reliable results. These methods may work temporarily, but they have proven to be insufficient on their own.

Christina Hagen, the Pamela Isdell Fellow of Penguin Conservation at BirdLife South Africa, is currently leading efforts to revive the African penguin population. Since transitioning into this role in 2015, she has been responsible for identifying potential new colony sites, working with key stakeholders, and investigating techniques that can help re-establish stable breeding grounds.

1.2 Problem Statement

The current system for protecting the penguin colony at De Hoop uses fencing along with CCTV cameras that detect motion when a predator, such as a honey badger, approaches. While these fences do offer some protection, they are passive and can fail at any point. The real issue is that although the cameras can send a notification, there's no clear system in place for what should happen next. Human surveillance is limited and conservation staff cannot be on-site at all times to intervene and deter these predators. Our project aims to take motion detection data and turn it into an automated response, such as triggering deterrents, to actively help protect the penguins.

1.3 Benefits of the Solution

This solution aims to reduce predator activity near the fence, with the hope of setting a precedent that discourages predators from approaching the area. It does not rely on constant human intervention, but still allows for accurate records and increased awareness. It's also a scalable solution that can be applied to multiple sites supporting other penguin populations. In essence, the system is designed to improve the survival chances of the penguins and help the colony grow and repopulate over time.

1.4 GitHub Link

https://github.com/AaronIsserow/penguin-patrol-alert-system2.git

Chapter 2

Problem Analysis

2.1 D-School Activities

As part of the D-School phase of the course, we worked in teams of six to complete a small design project. The task was to build a "dream house" using arts and crafts materials. The focus was not on the end product but rather on teamwork and the process of design.

We started by discussing ideas as a full group before splitting into smaller teams to build different parts of the house. When we tried to bring everything together, we quickly saw the importance of planning ahead and making sure everyone was on the same page. Miscommunication led to a final product that did not meet the specifications we set out for it.

It was clear how important it was to test the product in the early stages of development. Some parts did not work as expected and we had to adapt quickly. This taught us how valuable early prototyping and iteration are in any design process. Overall, the activity highlighted the importance of communication, planning and thinking about the end-user throughout a project.

2.2 Design Choices Explored and Final Design Selection

Our group focused on developing a system to protect penguins from predators like honey badgers. We explored a wide range of ideas, starting with physical solutions like improved fencing, patrol dogs and moats. These were ruled out early on because they were not practical or reliable.

We then looked at technology-based options, including drones and noise sensors, but these also had limitations. The most promising approach combined motion detection with a deterrent system.

We considered several deterrents—sound, water, paintballs and electric shocks. Most were eliminated due to safety risks or ethical concerns. In the end, we decided that a motion-triggered system using safe but effective deterrents would offer the best solution. It could respond quickly to predators while still keeping the penguins and their environment safe.

2.3 Subsystem Overview

The sensing and object detection subsystem acquires environmental signals through image processing techniques and converts them into actionable tracking data. The mechanical design and actuation control subsystem converts tracking data into a precise physical orientation of the deterrent mechanism, while also protecting the electronics from the environment. The user interface subsystem delivers real-time system operation feedback and interactive controls to users in order to facilitate monitoring and response.

Chapter 3

Literature Review

3.1 Background on African Penguin conservation

The African penguin population has declined by approximately 65% since 1989. In 2019, the estimated population was 17,700 breeding pairs [1].

In the 20th century, the African penguin population was estimated to be between 1.5 and 3 million. Since then, multiple factors have contributed to the decline in population size, including overexploitation, habitat loss, predation in land-based colonies, and climate change [2].

3.1.1 Predation as a threat

Predation is a major threat to the African penguin species, particularly severe for mainland colonies says Coudert et al. state: 'Caracals have been recorded engaging in repeated supernumerary predation events in both the Simon's Town and Betty's Bay (mainland) penguin colonies, while leopards have similarly attacked penguins in Betty's Bay and the nascent colony in the De Hoop Nature Reserve' [3]. Nattrass and O'Riain argue that without direct human intervention, it is expected that the threat the predators pose to their prey's populations will continue to exist and potentially increase [4]; Rhoda extends this argument to the penguin colonies [1]. It is clear from these authors (Rhoda, Nattrass and O'Riain) that preventative measures need to be implemented to stop the declining population size of the African penguins.

However, it is also important to consider the effects of depredation on the ecosystem. Due to anthropogenic impacts, large predators are particularly vulnerable [5]. Ordiz argues that anthropogenic impacts, in addition to being physically larger than their resource competitors, mean that large predator species invariably have large home ranges and relatively low densities. These factors risk a greater secondary extinction [1][6]. Since apex predators play an important role in maintaining the balance of the ecosystem by limiting the populations of prey and competing predators [5], it is important to consider the effect of predator deterrents on the predators themselves as, 'the suppression or removal of apex predators may also lead to proliferation of alien or invasive species, potentially driving secondary pest problems for agriculture and fisheries, which may, in turn, threaten vulnerable prey species' [1]. Certain vulnerable prey species, with low growth rates, inhabiting particularly exposed habitats, such as the African Penguin, may experience local distinction [1]. For this reason, it is important to consider anti-predation techniques which do not harm the populations of the predators, but rather deter them from the vulnerable prey species.

3.2 Monitoring

Monitoring is crucial in effective wildlife conservation. It allows for continuous data collection of predator behaviour and movement patterns. Monitoring systems are key to deploying effective deterrents to protect against predation. Monitoring can be split into surveillance and detection, the former focusing on the study of animal movements and behaviours and the latter focusing on the accurate identification of animal positions. This section explores various surveillance and detection methods employed in wildlife conservation, ranging from human surveillance to the inclusion of modern technologies in more advanced monitoring systems.

3.2.1 Human Surveillance

Traditional predator management often relies on human surveillance. While this approach can be effective in certain settings, it presents several challenges. The presence of humans in conservation areas has been shown to influence wildlife behavior, often leading to shifts in habitat use and activity patterns. Lewis et al. observed that 'some species (e.g., black bear, coyote, and mule deer) altered their activity patterns on recreation trails to be more active at night' as a direct response to human presence [7]. This suggests that continuous monitoring by patrollers may unintentionally disturb wildlife, potentially impacting their natural behaviors and ecological interactions.

A key responsibility of human monitors is to patrol conservation areas to prevent illegal activities like poaching. This usually involves ranger teams regularly moving through wildlife zones to check for any signs of activity. This on-the-ground approach has proven to be quite effective. For example, in Rwanda's Nyungwe National Park, ten years of patrol data showed that increasing patrols in high-risk areas led to a noticeable decrease in poaching indicators, such as snares and illegal camps [8]. The rangers not only found and removed these threats, but their presence alone made it harder and riskier for poachers to operate, which helped reduce illegal activity overall.

In many rural communities, especially those bordering wildlife reserves, local people actively guard farms and livestock to protect them from predators and crop-raiding animals. By staying in watchtowers or patrolling fields with flashlights and making noise, they can prevent surprise invasions by elephants or other herbivores. Studies have found that such human guarding, especially when combined with simple deterrents, is one of the most effective strategies to keep crop-raiders at bay [9].

3.2.2 Motion-Activated Cameras

Motion-activated cameras are widely used in wildlife monitoring to record animal presence, behavior, and movement patterns. These devices are equipped with sensors that trigger image or video capture when movement is detected. Their ability to function autonomously for extended periods allows researchers to obtain continuous data over different times of the day and seasons, reducing the need for direct human observation [10].

Kays et al. describe camera traps as 'an appropriate technique for animal monitoring' due to their ability to 'require low labor, yield robust data, and be non-invasive when photographs are captured using invisible IR flashes' [11]. Their research, which involved a year-long case study at Barro Colorado Island (BCI), Panama, highlighted the effectiveness of motion-sensitive cameras in monitoring medium- and large-sized terrestrial animals. They further emphasize that camera traps provide 'a unique and unbiased picture of environmental dynamics' by recording the movement of wildlife without directly interfering with their behavior [11].

One major advantage of using camera traps is that they provide concrete evidence of which predators are responsible for raiding nests, for example. In some cases, predators that were only suspected before have now been clearly identified through camera footage[12]. These cameras have also recorded unexpected events—like surprising predator appearances or unusual behaviour—that would be difficult or even impossible to observe in person[12]. However, a downside to this method is the massive amount of data it can generate. As more cameras are added to the network, researchers can quickly face data management challenges, with thousands of images that need to be reviewed and analysed [10].

3.2.3 Infrared Sensors

Infrared (IR) sensors play a crucial role in wildlife monitoring, especially in areas where visibility is poor due to low light or dense vegetation. These sensors work by detecting the heat emitted by animals, allowing researchers to identify species even in total darkness. This makes IR technology particularly useful for studying nocturnal predators, as it captures movement without disrupting natural animal behaviour[10].

IR-enabled camera traps are often combined with motion sensors to improve detection accuracy. According to Kays et al., many of these systems use infrared flashes to capture black-and-white images at night and colour photos during the day[11]. This dual-mode setup allows for round-the-clock monitoring and helps researchers gather consistent data.

However, IR sensors aren't perfect. They can sometimes be triggered by non-animal sources like warm air currents or sunlit leaves moving in the wind, which leads to false alarms and lots of empty images[13]. To avoid missing important activity, some researchers switch to time-lapse mode, where images are taken at regular intervals regardless of motion. While this guarantees image capture, it also results in an overwhelming number of photos to sort through[13].

3.2.4 AI and IoT

The implementation of Artificial Intelligence (AI) in the monitoring and detection of wildlife animals is becoming more widely used. This is due to the limitations of traditional approaches to animal deterrence, which include effectiveness and ecological sustainability. Mishra and Yadav mention that this has led to a change in the process of finding creative solutions, which focuses on the Internet of Things (IoT) and AI. Animal deterrent systems which utilize both these concepts in tandem have shown a transformative change in methodology [14].

Mishra and Yadav also mention that one of the main problems is the adaptability of animals to already existing traditional deterrents, such as fear tactics or physical obstacles, which reduces their long-term effectiveness. The addition of IoT and AI to animal deterrence systems will enable continuous monitoring and real-time decision-making, thus making it able to adapt to the evolving behaviours of wild animals [14]. Simla et al. presented the Agricultural Intrusion Detection (AID) system, which makes use of IoT with image processing and deep learning techniques to allow for continuous, real-time intrusion detection with lightweight communication protocols. The experimental results from the study achieved a 94% accuracy in intrusion detection [15]. Balakrishna et al. proposed a Crop Protection System (CPS) model which emphasizes the use of machine learning algorithms such as Region-based Convolutional Neural Networks (R-CNN) and Single Shot Detection (SSD) to accurately detect animals from images. The model uses a Raspberry Pi to run the machine algorithm which is interfaced to an ESP8266 WiFi module, reiterating the importance of IoT in animal monitoring. The model achieved a 92% accuracy in animal detection [16]. Lastly, Adami et al. created the Embedded Edge-AI-based Intelligent Animal Repelling System (EEAIRS), which is based on IoT platforms and evaluated various edge computing devices running real-time animal detection. The system achieved an accuracy of 87% in detecting animals [17].

3.2.5 UAV Animal Deterrents

Studies have evaluated the use of Unmanned Aerial Vehicles (UAVs) in animal detection and crop protection. King et al. suggested a concept for UAVs for 'bio-herding' of animal groups. Their idea proposes a pair of UAVs, where one performs the 'surveillance' of the animals and another herds them. The paper primarily focuses on the framework, rather than experimental results. It emphasises the potential for UAV use in monitoring animal movements [18].

Breck et al. highlight that animal deterrents can be enhanced by the use of UAVs. These can be programmed to detect predetermined movements, at set times and routes. Adaptive movements are also discussed in the form of UAVs being able to react to animal movements directly. While the study examines various robotic systems, it outlines the idea of predetermined and adaptive movement patterns relevant only to UAV operations. The study provides no experimental data regarding UAV use in animal deterrence [19]. A similar paper by Pitla et al. (2020) discusses ground and aerial robots for crop production. UAVs equipped with cameras and sensors can be used to detect and monitor wildlife, allowing them to detect and/or intervene

predator intrusions. Pitla et al. also introduced the idea of thermal sensors for detecting animals in visually crowded environments. The paper provides only a comprehensive overview of UAV applications, specific experimental results of UAV predator deterrence are not detailed [20].

The performance of UAVs in predator deterrence is evaluated in the paper by Weston et al. They observed that birds had the tendency to escape when in the vicinity of drones. They found that the escape response was 14.6% higher when the drone (UAV) was at a lower altitude ($\approx 4\ m$) when compared to higher altitudes ($\approx 10\ m$), with 88.4% of overflights resulting in escape responses. The paper also highlighted that a longer take-off distance reduced the likelihood of an escape response (optimal distance beyond 120 m). The authors also mentioned that the distance between the drone and the wildlife should be minimized to reduce noise [21]. Similarly, Schroeder et al. also found that higher altitudes and lower speeds in drones reduces disturbances. Their study focused on the monitoring of guanacos (camelid species of South America). They also reported that large groups of guanacos tend to react regardless of altitude and speed [22].

3.3 Predator Deterrent Systems

In addition to monitoring, active intervention is often required to reduce predator access to vulnerable species. Predator deterrent systems are designed to either physically prevent entry or discourage approach through sensory means. These methods are broadly classified into two categories: exclusion deterrents, which rely on physical barriers, and light, sound and odour deterrents, which affect an animal's perception or response to the environment.

3.3.1 Exclusion Deterrents

Fencing

Fencing is a passive predator mitigation technique that involves 'constructing a physical barrier between human-resources and predators,' states Shivik [23]. Shivik argues that these exclusionary devices can be as simple as a wired fence or more complex, such as an electrified metallic fence, but due to maintenance and construction costs, they are generally more effective as small night-time enclosures [23]. Robel et al. mention that 'night penning' is effective in minimising losses to predators [24]; however, due to the nature of the African penguin, it is unlikely for this to work, as it would be difficult to ensure the penguins return to a designated pen at night, and penguins may be difficult to herd into a pen.

Rhoda also mentions fencing as a preventative method for caracals accessing mainland penguin colonies. Rhoda concluded that of various fencing techniques (Metal fence, Electrified fence, Mesh fence, Combination Fence, and Virtual Fence), the combination fence was the most successful proposed barrier solution to the caracal-penguin conflict (Figure A.1) [1]. Rhoda details a combination fence as a fence 'made of various materials such as wooden posts, trawl netting, an angled overhand, electric strands, and anchovy netting on the ground.' The strategy of using a combination fence to protect the penguins at the mainland colony of Simon's Town excludes and potentially endangers non-target species, mentions Rhoda when discussing the limitations of this strategy [1]. Exclusion techniques such as fences may not provide a foolproof barrier against all predator species [25]. Some predators, such as coyotes, are known to be able to climb over or dig under fences [25]. Small predators, such as honey badgers, are exceptional at finding or creating ways to get through or under fences [26]. Additionally, maintaining natural wildlife passages is essential in maintaining balance in the ecosystem [27].

Moats as Barriers

One of the most critical factors that makes moats effective barriers is their physical structure. Research suggests that 'Moats and similar human-made features are hardly comparable with natural ecosystems. The steep, almost vertical banks, artificial channels, depth differences, and many other features had a crucial

impact on the sediment spatial composition and, hence, also on moat functioning' [28]. This indicates that moats naturally limit movement across their boundaries, making them a potential tool for restricting predator access in conservation settings.

Beyond simply acting as barriers, moats can also influence the surrounding habitat in ways that benefit prey species. Human-added materials in the moats—such as wood and other objects—have been found to contribute to ecological enhancement. As Antczak-Orlewska et al. note, "anthropogenic wood and other artefacts and ecofacts in the bottom can serve as an additional habitat" [28]. This indicates that moats have the potential to create microhabitats that offer shelter and resources for smaller species.

Animals such as honey badgers are good swimmers, which makes moats a less effective option as a barrier [29].

3.3.2 Sensory Deterrents

Light Deterrents

Light-based deterrents include the use of visual stimuli to deter animals. Ohrens et al. presented research on the use of flashing lights to deter puma attacks on livestock. The authors of the paper inferred that the flashing lights were successful in deterring pumas from attacking llamas and alpacas [30].

Another approach is studied by Laguna et al., who proposed light-emitting devices for deterring red deer from certain pre-established areas of study. The light devices consisted of changing patterns of light based on intensity and frequency. The experimental results showed a 48.96% reduction in deer when the deterrents were activated; no results were available on which combination of intensity and frequency was the most effective [31].

Habituation is the primary limitation for all types of sensory deterrents [32][33]. Birds, known for their intelligence, often become unbothered by a variety of different visual deterrents, such as predator decoys, reflective materials, and lasers [34]. The initial fear response wanes unless the visual cues are associated with an actual threat.

Sound Deterrents

These methods use auditory signals to deter animals; examples include ultrasounds, natural sounds, and artificial sound playbacks. Bomford & O'Brien studied the effect of sonic deterrents. They reported that the novelty of the sound and the biological reference to the animal being studied are key factors to consider when applying sonic deterrents [35]. Biedenweg et al. experimented on the response of western grey kangaroos to natural sounds, such as foot stomps and raven calls, and artificial sounds like a fake snake hiss or bullwhip crack. The authors utilised a Principal Component Analysis to classify the animal response. They concluded that an 'alertness' response and an 'escape' response accounted for 36% of the variance, meaning that they were the most dominant behaviours [33].

Another experiment by Belant et al. focused on the use of electronic frightening devices such as the Yard Gate (an electronic pest repellent that uses ultrasonic sound waves) and the Usonic Sentry (a motion-activated ultrasonic device) to deter white deer from certain areas. The authors state: 'It was concluded that the electronic frightening devices tested were generally ineffective in deterring white-tailed deer from preferred feeding areas' [36, p. 107]. Auditory deterrents, such as ultrasonic devices, have been shown to be largely ineffective due to the continual playback. Additionally, these devices must operate at frequencies that fall within the animal's optimal hearing range [33].

Odour Deterrents

Odour deterrents use chemical repellents that certain animals may be sensitive to. These repellents can be used to deter animals from certain areas. Baker et al. used chemical repellents to deter European badgers.

The study concluded that badgers preferred untreated baits, followed by cinnamamide and capsaicin (in no particular order), and lastly ziram. 'This study provides proof of concept that ziram has clear potential for reducing badger feeding damage through conditioned taste aversion to an odour,' states Baker et al. [37, p. 921]. The effect of the odour of apex predators on mesopredators (medium-sized carnivorous or omnivorous animals) was studied to check how their behaviour would adjust in an outdoor area. The stoat was used as the mesopredator, and the apex predator odours used were those of the cat, ferret, and wild dog. The study concluded that the apex predator odour was an attracting factor rather than a deterring factor [38]. Odour deterrents which function on novelty and irritation, are also susceptible to habituation unless the stimulus is sufficiently unpleasant [39].

The common limitation of these sensory deterrents is the animals' ability to learn and adapt which renders these static sensory deterrent methods less effective as time goes on. To counteract this, deterrent strategies must incorporate continuous variation or rotation of sensory deterrent methods in order to maintain efficacy [32].

3.4 Limitations of these Deterrent Systems

While innovations in automated deterrent systems represent significant promise for enhancing wildlife conservation, a comprehensive understanding of their inherent limitations is crucial to ensure effective, ethical, and sustainable implementation.

3.4.1 Habituation of Wildlife

The primary challenge with the implementation of any predator deterrent system is the potential for wildlife to habituate to the deterrent stimuli over extended periods of time [40]. Habituation is defined as a form of learning where an animal gradually reduces its response to a repeated stimulus that is neither threatening nor beneficial [40]. This process diminishes the effectiveness and success of various deterrents, including, but not limited to auditory, visual, and chemical cues [32] (such as specific odours).

For instance, Goodyear et al. show that elephants, which are highly intelligent animals with well-developed sensory systems, will habituate to auditory deterrents like the sound of banging pots and pans [40], which have traditionally been used to deter them from farmlands. Even artificial auditory deterrents like ultrasonic devices have proved largely ineffective due to the continuous signal playback that facilitates habituation [33].

3.4.2 Resource Constraints

Resource constraints are a crucial factor in wildlife conservation as a whole, and more specifically, they are crucial in implementing advanced technology-based predator deterrent systems. Limitations on resources can be due to operational requirements, technical expertise and initial implementation costs [41].

For instance, automated monitoring systems often require data storage solutions, software updates, maintenance, and reliable energy solutions. This is highlighted by Mbithi in his work in light deterrent methods for lions in Kenya [42].

3.4.3 Ethical and Ecological Considerations

The implementation of predator deterrent systems raises ethical and ecological considerations that need to be carefully addressed to ensure responsible conservation practices. Ethical concerns involve the welfare of both the target predator and the non-target species being affected by deterrent measures, potentially disrupting local ecosystems.

While lethal control methods have been employed in the past, they are criticised for being an inhumane and indiscriminate practice, often leading to the death or capture of non-target species alongside the intended predator [41].

Even non-lethal deterrent methods, such as the ones explored in this review, may pose ethical concerns. Electrified fences, for instance, are often considered humane but can still inflict pain and aversive sensations upon contact. Trapping may cause stress and potential injury to both the target and non-target species [43].

It is clear that although the deterrent method may seem ethical and take into account the surrounding ecology, it is important to consider potential unexpected consequences of the deterrent method.

3.5 Conclusion

This literature review has explored a wide range of predator deterrent systems and monitoring techniques relevant to the conservation of the African penguin. The core conservation issue is the dramatic population decline of African penguins. The main causes of this decline include predation by land animals, human activity, habitat loss, and climate change, emphasising the immediate need for deterrents. It is crucial that these deterrents do not harm the existing ecosystem (for example, removing apex predators could cause an ecological imbalance).

Monitoring methods were examined and organised according to their approach, beginning with active methods such as human surveillance, followed by passive techniques like camera traps, infrared sensors, and AI- or IoT-based systems. Each method presents its own advantages and drawbacks. Human surveillance is highly effective but can disturb wildlife and requires human interference. Camera traps are useful for long-term data collection but generate large volumes of footage. Infrared sensors are most effective at night but may trigger false alarms due to environmental factors. AI and IoT technologies offer accuracy, adaptability, and strong potential for future monitoring as these fields rapidly develop, but these methods require extensive research, large effective learning data sets and expensive computational software and implementation hardware. Additionally, the use of UAVs (drones) show promise in both surveillance and increasing the efficacy of predator deterrence. Their monitoring effectiveness varies based on operating conditions such as altitude and speed.

Various deterrent systems are used alongside monitoring to provide the necessary action component of a broader conservation strategy. Fencing, while effective for certain animals, is subject to wear and can become expensive or time-consuming to maintain. Moats act as natural barriers and may also support other species. Robotic deterrents offer adaptability and unpredictability compared to more static solutions. Sensory deterrents, including light, sound, and odour, may work initially but can lose effectiveness as animals become habituated.

Limitations present significant challenges to the success of any system. Animals often adapt to deterrents, diminishing their effectiveness over time. Resource constraints—including high costs, maintenance requirements, and the need for technical expertise—can impact feasibility. Ethical and ecological concerns must also be addressed to prevent harm to non-target species, reduce animal stress, and avoid unintended disruptions to ecosystems.

Despite these challenges, there is considerable promise in automated and adaptive systems that integrate robotics, AI, and IoT for both monitoring and deterrence of animals and predators.

Any solution must be ethical, sustainable, cost-effective, and considerate of the entire ecosystem. Ongoing research, interdisciplinary collaboration, and real-world testing are essential for the continued advancement of conservation.

Chapter 4

Sensing and Motion Detection- VCHEMA001

4.1 Introduction to Sensing

The sensing subsystem is responsible for the motion detection of honey badgers in the protected penguin enclosure. Its function is to provide a clear way of identifying the location of the moving target, serving as the primary interface between the world and our system. By continuously monitoring its surroundings, the subsystem enables autonomous operation of the entire turret system.

The sensing system will comprise of the following: the sensor, the processing platform and the detection approach. The system should be capable of detecting moving objects in real-time, with sufficient accuracy and responsiveness to initiate further system processes.

4.1.1 Scope and Limitations

The scope of the sensing subsystem is limited to the detection of motion. It is not intended to perform object classification, or complex environmental mapping. The goal is to detect the presence of movement robustly enough to activate subsequent system processes. The system is originally designed for the detection of honey badgers in a protected enclosure for penguins, the environment is assumed to be peaceful with any motion detection being associated only to potential threats.

Potential limitations include sensitivity to lighting variations, background noise (e.g., wind-blown objects), and false positives due to environmental motion. Furthermore, computational constraints may limit the complexity of processing algorithms that can be employed. These challenges will need to be considered during the selection of hardware and development of detection software.

4.2 Requirements and Specifications Analysis

Table 4.1: Requirements of the sensing subsystem.

ID	Description
SENSE01R	Detect any significant motion.
SENSE02R	Provide continuous real-time motion detection.
SENSE03R	Operate autonomously with minimal remote human interference.
SENSE04R	Output location of motion detected to trigger further system processes.
SENSE05R	Provide live monitoring capabilities.
ENV01R	Operate in all lighting conditions.
ENV02R	Minimize false positives from potential environmental noise.
PWR01R	The subsystem should operate off of a DC power supply.
BUS01R	The total budget for the entire project is R1500. The sensing subsystem budget, which includes the processing platform and sensing apparatus, should not exceed R1200. This is subject to change depending on other subsystem's monetary requirements.

Table 4.2: Specifications of the sensing subsystem.

ID	Description
SENSE01S	Provide a single rectangular bounding box identifying where motion has been detected.
SENSE02S	The subsystem should update the motion detection state at a frequency of at least 10 Hz (i.e. $\leq$ 100 ms per check).
SENSE03S	The subsystem shall initialise a connection to the frontend via WiFi at system boot and allow for full remote control of the sensing subsystem (i.e. start, stop, view live feed).
SENSE04S	The subsystem shall output the (x, y) pixel coordinate of the centre of the rectangular bounding box.
SENSE05S	The subsystem shall have live feed streaming capabilities of sensor output with $\leq$ 0.5 s latency and a minimum of 15 FPS.
ENV01S	The subsystem shall operate in lighting conditions between 0.1 lux (nighttime conditions) and $10,000$ lux (sunlight).
ENV02S	The subsystem shall not detect minor motion (leaves, birds, etc.) and shall have a recalibration to reset any unwanted environmental changes.
PWR01S	The subsystem shall operate using a 12 V, 1 A DC power supply, with considerations of other subsystem's potential power consumptions (i.e. servo motors, deterrent trigger, etc.).

The requirements of the subsystem are chosen based on what it is expected to achieve and how its performance can be qualitatively assessed. These requirements can be seen in Table 4.1.
From these requirements, more technical specifications are derived which are able to describe more quantitatively how the subsystem will achieve the previously defined requirements. The table of specifications is Table 4.2.

4.2.1 Acceptance Testing Procedures

A summary of the testing procedures detailed in section 4.5 is given in Table 4.3.

Table 4.3: Acceptance Test Procedures

Test ID	Description
AT01	Verify motion detection triggers bounding boxes and updates Supabase with correct timestamps upon detecting movement in the camera frame.
AT02	Confirm motion state update frequency meets or exceeds 10 Hz by logging timestamps over continuous motion and analyzing inter-update intervals.
AT03	Test autonomous remote operation by verifying Raspberry Pi boot, WiFi connection, frontend accessibility, and timely response to Start, Stop, and Live View commands.
AT04	Validate accuracy of detected motion coordinates by comparing generated bounding box size to a realistic bounding box around the detected object.
AT05	Measure video feed latency and framerate by simultaneously recording live feed and external stopwatch, then calculating delay and frame count over a sample period.
AT06	Assess motion detection functionality across different lighting conditions: direct sunlight, indoor lighting, and low/no light using NoIR camera.
AT07	Evaluate environmental noise rejection by monitoring false positives during minimal movement over 5 minutes and after recalibration of detection parameters.
AT08	Confirm power stability by running the system continuously under a 12 V, 1 A supply, monitoring current draw, system operation, and absence of emergency shutdowns.

4.2.2 Traceability Matrix

Table 4.4: Requirements Traceability Matrix

#	Requirement	Specification	Acceptance Test ID
1	SENSE01R	SENSE01S	AT01
2	SENSE02R	SENSE02S	AT02
3	SENSE03R	SENSE03S	AT03
4	SENSE04R	SENSE04S	AT04
5	SENSE05R	SENSE05S	AT05
6	ENV01R	ENV01S	AT06
7	ENV02R	ENV02S	AT07
8	PWR01R	PWR01S	AT08

4.3 Design Decisions

4.3.1 Sensor Selection

In the Literature Review (chapter 3), the following monitoring methods were examined as ways of detecting wildlife: Human Surveillance, Motion-Activated Cameras, Infrared Sensors, AI and IoT and UAVs. Out of these five choices, the Human Surveillance will be removed due to the requirement of autonomous operation with minimal human interference (SENSE03R) and so will UAVs, due to the cost of purchasing a drone exceeding the required budget (BUS01R). AI and IoT are not considered a sensor choice as they do not provide a way to analyze the surroundings, it is rather a detection algorithm used in tandem with one of these sensor selections in order to facilitate the detection of moving objects. To consider other options, Microphones and Thermal Cameras are also compared to Cameras and IR sensors. The criterion stem directly from the requirements and specifications listed previously in Table 4.1 and Table 4.2. The comparison table of the listed detection methods can be seen below in Table 4.5.

Based on the information from Table 4.5, the advantages and disadvantages and the ability of each sensor type to satisfy the requirements were compared and tabulated in Table 4.6.

From Table 4.6, it is evident that thermal cameras are not a viable option for a low-budget project. Despite their ability to provide precise heat map location of targets and excellent performance in low-visibility environments, their high cost, significant power consumption, and the requirement for specialised hardware and processing make them impractical for this application.

Table 4.5: Comparison of Detection Technologies[44][45]

Criterion	Cameras	IR Sensors	Microphones	Thermal Cameras
Detection Approach	Visual motion (frame differencing/direct pixel comparison)	Heat/motion via infrared intensity difference	Acoustic pattern or threshold detection	Heat signatures mapped visually
Location Detection Capability	Precise pixel coordinates (bounding box and centre coordinates)	Approximate zone (sensor field of view)	Directional if microphone array used; limited precision	Precise heat map location
Environmental Robustness	Sensitive to lighting/weather	Robust to lighting; affected by heat sources	Sensitive to wind and ambient noise	Detects heat signatures in total darkness, fog, smoke and through some visual obstructions
Power Consumption	Medium to high depending on resolution/FPS	Very low	Low to medium	Medium to high depending on resolution
Real-time detection delay	Moderate depending on resolution/FPS	Very low (instantaneous)	Low	Moderate to High depending on resolution and processing complexity
Cost (ZAR)	$\geq R$ 200, increases based on resolution	$\leq R$ 50	$\geq R$ 100	$\geq R$ 1000
Data Size	High (video, frames)	Very low (binary)	Medium (waveforms, frequency data)	High (thermal maps)
Integration Complexity	Medium (required computer vision libraries)	Very low (GPIO-level)	Medium (signal filtering + mic amplification)	High (specialised hardware/-software to analyse thermal maps)
Nighttime Performance	Poor unless IR filter removed	Excellent	Not affected	Excellent
Processing Requirements	High (image processing)	Minimal	Moderate (FFT or ML on signals)	Moderate to high
Maintenance	Medium (lens cleaning, housing)	No maintenance required	Medium (wind/weatherproofing)	Medium (lens calibration, shielding)
Live monitoring capabilities	Yes (video streaming available)	None (event-driven only)	Yes (sound streaming available)	Yes (live thermal mapping available)

Table 4.6: Advantages and Disadvantages of Detection Technologies Based on Requirements

Technology	Advantages	Disadvantages
Cameras	- Meets live monitoring requirement (video streaming) - Provides precise location (pixel-based) - Image processing capabilities with computer vision libraries (OpenCV available for free) - Real-time detection possible with low to moderate delay if lower resolution used - Medium cost and widely available - Little maintenance required (lens cleaning if exposed)	- Fails environmental robustness requirement (sensitive to light/weather) - Medium to high power usage, depending on resolution/FPS - High processing and data requirements
IR Sensors	- Meets low power requirement (very low consumption) - Excellent environmental robustness (not light-dependent) - Real-time detection with no delay - Low cost and easy to integrate - No maintenance required	- Fails location precision requirement (only zone-based) - No live monitoring (binary output only)
Microphones	- Meets live monitoring (audio stream) - Real-time detection with almost no delay - Moderate power usage - Not affected by lighting and environmental changes - Low data size	- Limited location precision (without array setup) - Poor environmental robustness (affected by wind/noise) - Moderate processing needs (FFT/filtering), not ideal for ultra-low systems
Thermal Cameras	- Meets live monitoring (thermal stream) - Provides precise location detection (heat map) - Excellent nighttime and environmental robustness	- High cost (not suitable for budget-constrained systems) - High data size and processing needs for high-resolution thermal maps - High processing complexity leads to long delays in real-time detection - Moderate to high power usage (fails strict low-power requirement) - High integration complexity (requires specialised hardware/software)

Microphones offer a low-cost, low-power solution; however, their susceptibility to ambient noise and their inability to provide accurate information on the location of the target render them unsuitable for this project.

IR sensors are highly affordable, energy-efficient,robust, with minimal integration complexity and have no delay which makes them suitable for real-time systems. Their lack of precise location output and the absence of live feed capabilities makes them unsuitable for a system that demands accurate tracking and monitoring.

Cameras are capable of meeting the live monitoring requirement through continuous video streaming. They can provide accurate location data of a moving target via pixel-based mapping and offer minimal delay in real-time processing when operating at lower resolutions. Additionally, a wide range of cameras is available, covering various power consumption levels and price points. Their primary disadvantage lies in their sensitivity to environmental changes and lighting conditions. Despite this limitation, cameras will be employed in this project due to their ability to provide pixel-perfect location accuracy and live video

stream. The effects of environmental changes will be addressed, as far as possible, through software-based compensation techniques in the Subsystem Architecture Section, section 4.4.

4.3.2 Processing Platform and Camera Selection

A camera based approach was chosen for this subsystem design. The next step is to choose a processing platform which will be able to run all required processing steps while also being able to run off of the 12 V, 1 A power supply and provide motor control capabilities through GPIO/PWM pins. It must be able to support a camera and take in images as inputs, process them frame by frame, create a bounding box around the target, send the centre of the bounding box to the motors, stream live video footage and connect to the frontend via WiFi. Traditional micro-controllers are not suitable for computer vision projects such as this one. A more suitable choice is to consider Single Board Computers (SBCs) which run off of their own Operating System and are able to do high-level computing and image processing at relatively fast speeds. Another feature to consider is the choice of camera that will be used. This is relevant because some cameras have compatibilities with certain processing platforms. The overall choice of processing platform and camera will be done together to maximise the requirements met. The following processing platforms were analysed based on popular use in this context and availability: Raspberry Pi 3B+, Raspberry Pi Zero 2 W and ESP32-CAM. The latter being a micro-controller with access to WiFi and a built-in camera, which is why it is considered for this application. These can be seen in Table 4.7. The cameras chosen for the subsystem are: Raspberry Pi Camera Module V2, USB Webcam (WINX DO Simple Full HD 1080P Webcam) and OV2640 (included with ESP32-CAM). Their features are compared in Table 4.8.

Table 4.7: Comparison of Processing Platforms

Criterion	Raspberry Pi 3B+	Raspberry Pi Zero 2 W	ESP32-CAM
CPU	Broadcom BCM2837B0, Quad-core ARM Cortex-A53 @ 1.4GHz	Broadcom BCM2710A1, Quad-core ARM Cortex-A53 @ 1.0GHz	Dual-core Xtensa LX6 @ 240MHz (600 DMIPS)
RAM	1GB LPDDR2	512MB LPDDR2	520KB SRAM + 2MB PSRAM
Operating System	Linux (Raspberry Pi OS, Ubuntu Lite, etc.)	Linux (Raspberry Pi OS Lite)	Embedded FreeRTOS / Arduino SDK
Connectivity	Dual-band Wi-Fi (2.4GHz & 5GHz), Bluetooth 4.2, Ethernet, 4×USB, GPIO	Wi-Fi (2.4GHz), Bluetooth 4.2, micro-USB OTG, GPIO	Wi-Fi (STA/AP), microSD, UART (via USB adapter), GPIO
Camera Interface	CSI (supports official Pi Camera), USB webcams	CSI-2 (supports official Pi Camera), USB OTG	Dedicated support for OV2640 integrated camera connector
Video / Graphics	1080p30 H.264 encode/decode, OpenGL ES 2.0, Composite video	1080p30 H.264 encode/decode, OpenGL ES 1.1/2.0, Composite out	MJPEG streaming at lower resolutions, JPEG output, no GPU
Power Consumption	5V @ 2.5A (typical)	5V @ 2.5A (typical)	3.3V @ 160–240mA (active)
Image Processing Capability	High — supports OpenCV, ML models, Python & C++ pipelines	Moderate — OpenCV supported, limited by RAM/CPU	Very low — no OpenCV, uses lightweight image manipulation
Streaming Capability	Full MJPEG, RTSP, Flask/WebRTC support	Lightweight MJPEG/RTSP streaming	Native MJPEG streaming at VGA/SVGA resolutions
Live Detection Compatibility	Excellent — robust motion detection, tracking, filtering possible	Good — basic motion detection via OpenCV feasible	Minimal — relies on pixel difference thresholding or custom logic
Expandability	Excellent — GPIO, USB, HATs, CSI, DSI, audio/video out	Moderate — GPIO, CSI, HDMI, limited USB expansion	Limited — basic GPIO, UART, SPI/I2C/PWM interfaces only
Ease of Integration	Very High — extensive documentation and package support	High — well-documented, slight performance tuning needed	Low — embedded development, C++/Arduino SDK required
Cost (ZAR)	R758.90	R332.90	R149.90

Table 4.8: Comparison of Compatible Camera Modules

Criterion	Raspberry Pi Camera V2	USB Webcam (WINX DO Simple Full HD 1080P Webcam)	OV2640 (ESP32-CAM)
Interface Type	CSI (15-pin ribbon; Pi Zero requires specific cable)	USB 2.0 (UVC)	Parallel camera interface (native to ESP32)
Resolution and Video Modes	8MP (3280×2464); Video: 1080p30, 720p60, 480p90	1080p (1920×1080) @30FPS	UXGA (1600×1200@15fps), SVGA@30fps, CIF@60fps
Field of View / Optics	Fixed-focus lens; no autofocus; 62.2° horizontal FOV	67° FOV; fixed-focus (50 cm –)	Narrow FOV; fixed lens with moderate sharpness
Nighttime Performance	Poor (V2), Good with NoIR + IR lighting	Poor (no IR support)	Poor (no native IR support, low sensitivity)
Power Consumption	250–300 mW typical	>500 mW (includes microphone and LED indicator)	125–140 mW active, 600 A standby
Form Factor and Weight	25×23×9 mm; 3g	Integrated clip mount; 360° rotation; bulkier	Compact board; small integrated form factor
Operating System Compatibility	Raspberry Pi OS (official support)	Cross-platform: Windows, MacOS, Linux, Android (driver-free)	Compatible with Arduino IDE, ESP-IDF
Cost (ZAR)	R299.90 (standard or NoIR)	R349.00	Included in ESP32-CAM cost
Board Compatibility	Native on Raspberry Pi 3B+, Pi Zero 2 W (with adapter cable - R42.90)	USB on Pi 3B+, Pi Zero 2 W (with USB to micro-USB cable R34.90)	ESP32-CAM only, pre-built

Based on the information from Table 4.7 and Table 4.8, the advantages and disadvantages and the ability of each processing platfor and camera to satisfy the requirements were compared and tabulated in Table 4.9 and Table 4.10 respectively.

Table 4.9: Advantages and Disadvantages of Processing Platforms

Processing Platform	**Advantages**	**Disadvantages**
Raspberry Pi 3B+	- High processing power (quad-core 1.4 GHz) - Full Linux OS with extensive software support - Native CSI camera interface with high bandwidth - Multiple USB ports for peripherals and expansion - Supports Wi-Fi - Strong community and documentation	- Higher power consumption (5V @ 2.5A) - Larger size and weight - Higher cost relative to other options
Raspberry Pi Zero 2 W	- Compact size and low cost - Quad-core ARM CPU suitable for moderate processing - Supports CSI camera modules and USB devices with suitable adapter cables - Supports Wi-Fi - Full Linux OS with decent software support	- Higher power consumption (5V @ 2.5A) - Limited RAM (512 MB) restricts heavy multitasking - Limited USB connectivity (single micro-USB port) - Lower processing power than Pi 3B+
ESP32-CAM	- Very low power consumption - Integrated 2MP camera included - Small form factor, cost effective - Supports Wi-Fi streaming and simple image processing	- Very limited RAM and CPU power (240 MHz dual-core) - No full operating system; limited software stack - Low image resolution and processing capability - Limited expandability and peripheral support

Table 4.10: Advantages and Disadvantages of Camera Modules

Camera Module	**Advantages**	**Disadvantages**
Raspberry Pi Camera V2	- High resolution (8MP stills, 1080p30 video) - Native CSI interface with Raspberry Pi - Low power consumption (250–300 mW) - Small, lightweight form factor - Official support and drivers in Raspberry Pi OS - Supports multiple video modes and resolutions	- Limited low-light performance without NoIR and IR illumination - Fixed focus lens limits close-range sharpness
USB Webcam (WINX DO Simple Full HD)	- Plug-and-play via USB interface - Built-in microphone and privacy shutter - Universal OS compatibility (Windows, Linux, MacOS) - Good image quality at 1080p resolution - Wide 67-degree field of view	- Higher power consumption (>500 mW) - Larger size and cable management required - Depends on USB bandwidth, may burden processor - Lower location precision due to compression and optics
OV2640	- Very low power consumption (140 mW active) - Integrated camera with minimal hardware required - Supports multiple image formats and compression - Small form factor for embedded applications - Cost included with that of ESP32-CAM	- Limited max resolution (1600×1200) - Poor low-light sensitivity and dynamic range - Limited image quality vs dedicated cameras

Based on the information tabulated in Table 4.9 and Table 4.10, the ESP32-CAM and in turn the OV2640 camera module were discarded due to their limited computational power, which is needed for real-time image processing applications such as this one. The USB Webcam (WINX DO Simple Full HD) is also discarded due to the larger size making it difficult to position safely in an open environment, and the higher power consumption. The camera of choice is therefore the Raspberry Pi Camera V2 NoIR (for nighttime capabilities).

This leads to the choice between the Raspberry Pi 3B+ and Zero 2 W for the processing platforms. Both processors run on the same Operating System, have the same camera support and have the same power consumption. The only meaningful difference is the cost to processing power tradeoff between the two. This choice is purely based on personal preference as it comes down to how complex the software implementation needs to be and how easy it is to interface with each of the processors. The processor chosen for this particular project is the Raspberry Pi 3B+. The reason behind this being the higher processing power, allowing for a faster, more accurate output without a lot of heavy optimisations in the software. Another reason is the availability of multiple USB ports, allowing for a keyboard and mouse to be connected for an easier interface with the system through the official desktop. The main disadvantage is the extremely high cost, which makes up most of the given budget. This has been taken into consideration together with the other subsystems and has been deemed viable for our implementation.

4.3.3 Detection Algorithm

The main approaches involve classical computer vision (through OpenCV or Scikit-image) or more complex approaches such as Machine Learning. Machine Learning models have become more used due to their increased accuracy in detecting objects, with the added functionality of being trained to classify objects. This project would benefit from a ML model trained on an extensive honey badger dataset but due to limited computational power it would not be able to perform in real-time. The more lightweight approach of computer vision through image processing is taken. The following are the most common motion detection algorithms:[46][47]

Frame Differencing

Frame differencing involves subtracting each frame from the previous one to detect pixel changes indicative of motion. It is best for fast and distinct motion but limited by noise sensitivity and lack of memory of static objects.

Figure 4.1: Output mask from frame differencing

Background Subtraction

This approach uses a reference background image, which is continuously updated, to isolate moving foreground objects. It is accurate for fixed cameras but requires adaptation to lighting and background changes.

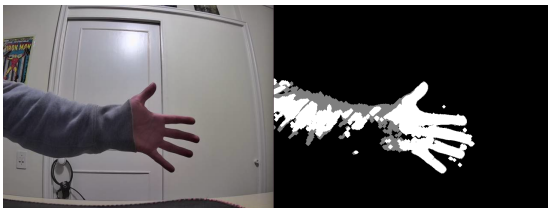


Figure 4.2: Output mask from background subtraction

Optical Flow

Optical flow tracks pixel displacements between frames, generating a vector field of motion. It is well-suited for detecting slow or complex movement, though it's computationally demanding and more difficult to implement.

Figure 4.3: Output mask from optical flow

Based on Table 4.11 and the figures showing the key differences in the motion binary masks created, Figure 4.1 for frame difference, Figure 4.2 for background subtraction and Figure 4.3 for optical flow, optical flow will not be used in this project. It is more computationally intensive and more sensitive to lighting variations. The frame differencing is the simplest and fastest approach, while background subtraction is more robust for different lighting conditions.

Table 4.11: Comparison of Motion Detection Methods

Method	Principle	Advantages	Disadvantages
Frame Differencing	Pixel-wise subtraction of consecutive frames	- Simple and fast - Low computational cost - Suitable for real-time	- Sensitive to noise - Poor for slow movement - Requires threshold tuning
Background Subtraction	Compare current frame to a background model	- Robust to noise - Handles stationary objects - Better detection accuracy	- Requires background update logic - Sensitive to illumination/shadow - Not ideal for moving cameras
Optical Flow	Estimate pixel motion vectors based on intensity gradients	- Detailed motion estimation - Works with slow/smooth motion - Good with partial occlusions	- Computationally intensive - Sensitive to lighting variation - Harder to tune for real-time

For this project, a variation of frame referencing will be used. Rather than taking the difference between the current frame and the previous frame, a stationary frame is initialised on boot and the absolute difference between the two is taken. This will make up for the poor performance of slow moving targets of basic frame difference as any object that was not there originally, even if stationary, will be detected as a target. Due

to the lighting in the environment changing over time, a new background frame will be taken at regular intervals to recalibrate the motion detection to the newly lit environment.

4.4 Subsystem Architecture and Integration

This section is involved in highlighting the actual design of the sensing subsystem, starting from powering the Raspberry Pi 3B+ processor(referenced as RPi3 from now on), to the structure and implementation of the motion detection code on the RPi3, to how the sensing subsystem integrates with the actuation control and the frontend UI.

4.4.1 Hardware Overview

The RPi3 requires 5 V, 2.5 A in order to function. The specifications state that the subsystem must be powered by a 12 V, 1 A DC power supply (PWR01S). In order to step down the voltage, a buck converter will be used. Since a buck converter is a robust component which can be bought as a single unit, there was little to no design choice in which one to purchase, it needed to be able to step 12 V down to 5 V. The buck converter that was purchased was the DC Buck Step-Down Power Module, 3A/24V, MP2315. A picture of the buck converter and its pin-outs can be seen in Figure 4.4. It is able to take in 4.5 V − 24 V, and can supply a maximum current of 3 A.

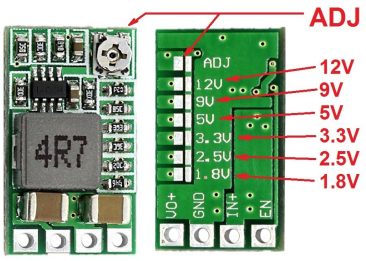


Figure 4.4: MP2315 Buck Converter with labeled input/output pins

A high level interface diagram of the buck converter with the DC supply and the RPi3 is provided in Figure 4.5, it also includes all other hardware components connected to the RPi3 for the sensing subsystem. The monitor, keyboard and mouse are optional components used only for the use of the RPi3 with the Raspberry Pi OS desktop.

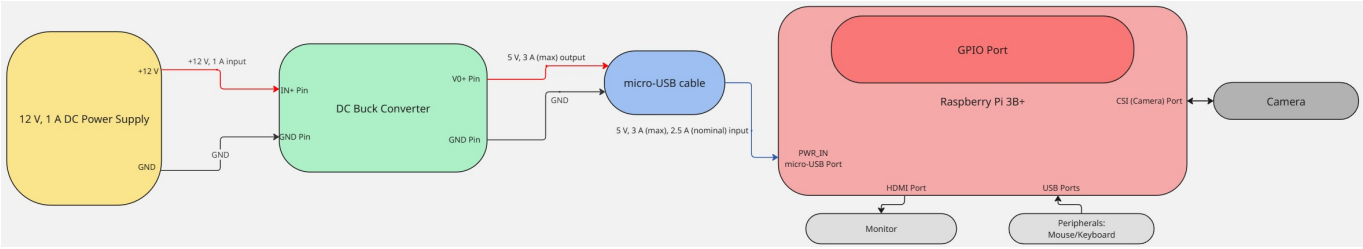


Figure 4.5: High-level interfacing diagram of the all the hardware used when using the Raspberry Pi 3B+ for the sensing subsystem.

4.4.2 Software Structure

The software implemented in the sensing subsystem is intended to perform the tasks numbered below. All tasks except for 6. will be explained in this subsection as it forms part of the basic motion detection algorithm. Task 6. is the interfacing and integration with the other subsystems and will be spoken about in the following subsections, subsection 4.4.3 and subsection 4.4.4. All of the image processing was done through the open source Python library OpenCV, due to continuous community support and helpful guides available.

1. **Initialise System Components**: set up camera and load any models and libraries.
2. **Continuously Capture Video Frame when Activated**: real-time video feed.
3. **Set up Reference Background Frame**: analyse background and acquire a stable reference background frame.
4. **Detect Motion Using Frame Differencing and Generate Detection Mask**: identifies all pixels that have moved with a binary mask.

5. **Find Bounding Box of Motion Pixels**: find the contour and highlight it on the video output.
6. **Trigger Response**: send update flags to frontend and start the actuation control (Subsystem Integration).
7. **Handle Recalibration of Background Frame Passively**: reset the background frame periodically to account for environmental changes.

Motion Detection Loop

The subsystem is designed to continuously process a video feed to detect motion using an adapted frame differencing approach with passive recalibration. The motion detection logic and continuous loop can be seen in the following Flowchart Diagram, Figure 4.6:

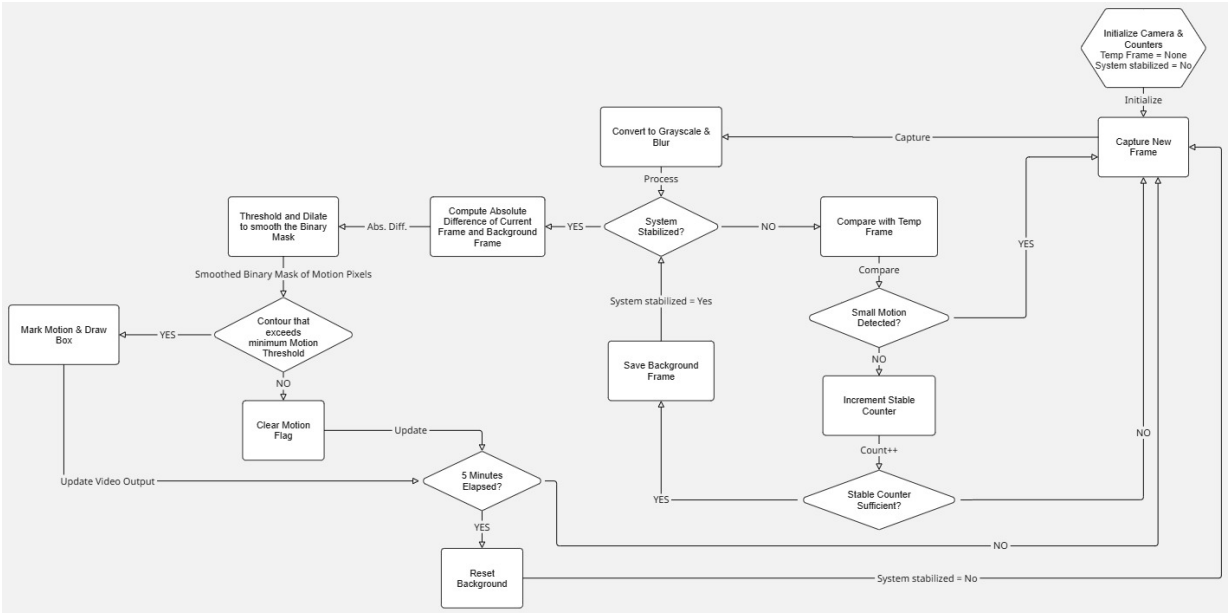


Figure 4.6: High-level Flowchart Diagram of the Motion Detection Logic Loop and Stable Background Frame Calibration.

I will not go through the code implementation, but I will highlight some of the processes happening in the flowchart. Firstly, motion detection is done through frame differencing, this consists of converting the current frame to grayscale, performing a Gaussian Blur (OpenCV function) and performing an absolute difference between it and the background frame, also grayscale and blurred (provided stabilisation has occurred). This will result in a grayscale mask of all the pixels that have changed from the reference background image and their relative intensities, representing potential motion. This mask is then filtered using a binary threshold to isolate only the pixels that have the highest intensity as the lower intensities are most likely noise. Finally a morphological dilation is performed to suppress additional noise ad bridge small gaps there may be, the kernel used is the standard 3x3 pixel element. Examples of these operations performed in sequence is provided in Figure 4.7.

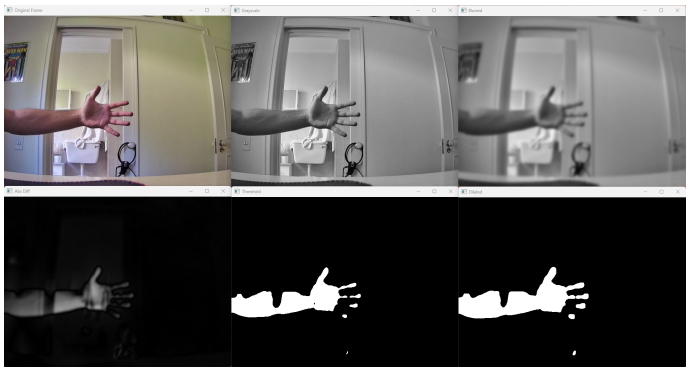


Figure 4.7: Examples of the motion detection image processing sequence: 1. Original Frame, 2. Grayscale Frame, 3. Gaussian Blur, 4. Absolute Differencing, 5. Binary Thresholding, 6. Morphological Dilation.

Finding the Bounding Box

The next feature that is worth noting is the contour finding algorithm. OpenCV has a findContours() function which is able to detect the outer boundaries of shapes. The bounding rectangle of the contours can be found using the OpenCV function boundingRect(). This will create bounding rectangles for all contours that the findContours() function finds in the binary mask. This results in multiple targets which would not be suitable for this application. A get_best_contour() function was defined that retrieves all external contours and selects the one with the largest area. This is based on the assumption that the largest contour will always correspond to the primary moving object in the frame. There is a minimum motion threshold which defines the smallest contour area that is allowed for the algorithm to recognise it as motion. This filters out small environmental changes and noise. A figure comparing the bounding boxes generated with the basic algorithm and the get_best_contour() function are seen in Figure 4.8.

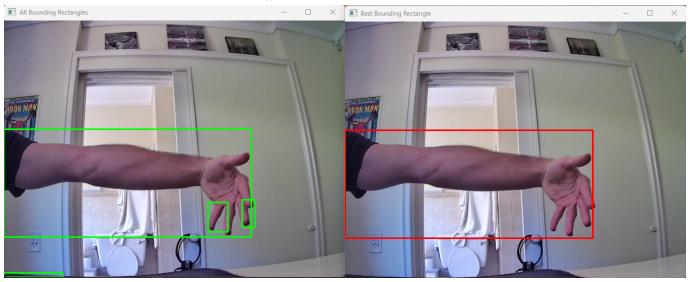


Figure 4.8: Comparison of bounding boxes generated by basic algorithm vs the get_best_contour() function for moving objects in the frame.

4.4.3 Actuation Control Integration

After a bounding rectangle is found in the motion detection loop, the subsystem will provide this to the actuation control algorithm to move the motors to point to the direct location of the target. The algorithm for controlling the actuators is talked about in the Mechanical and, Actuation Control Design, chapter 5. This section will speak about what the sensing subsection provides and how the motors are physically connected to the RPi3 processing board.

The motion detection algorithm provides an outer bounding box defined in pixel coordinates, the actuators need a specific coordinate (x, y) in order to move to that location. The assumption is taken that the most probably location of the object is in the centre of the bounding box, thus that is given as the parameter into the move_servos() function. The wiring configuration for connecting the hardware of the Actuation Control subsystem can be found in the figure provided in that section, Figure 5.19.

4.4.4 frontend UI Integration

The interface with the frontend UI consists of two key components: a Flask web server acting as the REST API gateway for external control, and a motion detection python script which runs locally on the RPi3. The interface is handled via the Flask subprocess control, allowing the server to start and stop the system remotely. Additionally, the frontend makes use of Supabase Database as a real-time backend to log the motion status, and system flags.

A Flask server is a lightweight Python-based web server used to create web APIs. It was chosen because it is simple to set up, runs efficiently on the Raspberry Pi, and allows the frontend to communicate with the backend using HTTP requests. It integrates well with Python scripts (like motion detection and actuation) and supports easy interaction with Supabase and the UI. The table below explains what each of the processes do and how they interact with one another, Table 4.12.

Subsystem	Functionality	Interaction Description
Motion Detection	Captures video feed, processes frames to detect motion using frame differencing, thresholding, and dilation.	- On detecting motion, triggers a flag and stores metadata (e.g., timestamp) relevant to the motion event. - Logs motion events to Supabase for historical tracking and alerting. - Provides a live video stream accessible through the '/video_feed' HTTP endpoint.
Flask Controller	Provides an interface to start/stop the system and monitor its status via HTTP endpoints.	- Acts as the bridge between the frontend and backend. - Exposes '/start' and '/stop' endpoints to control the motion detection script execution. - Updates Supabase with the system 'status' ('True' or 'False') on command and at shutdown.
Frontend UI	A web-based dashboard that displays system status, live video, and allows user control.	- Sends HTTP POST requests to Flask controller via '/start' and '/stop' to activate or deactivate the detection system. - Queries '/status' endpoint to poll the current operational state. - Consumes the live video stream from the '/video_feed' endpoint to display real-time camera feed.
Supabase Database	Cloud-hosted database for storing status and detection logs.	- Holds the 'status' flag indicating system monitoring activity, updated by Flask controller on startup, shutdown, and frontend commands. - Stores motion detection event logs sent from the motion detection subsystem for historical tracking and alerts.

Table 4.12: Overview of Subsystem Interactions and Architecture Integration with the frontend and UI

Lastly, the only other thing that needs to be addressed is the logic behind when the sensor subsystem logs a motion event. The following flowchart explains the logic which which alerts are logged in the database, Figure 4.9.

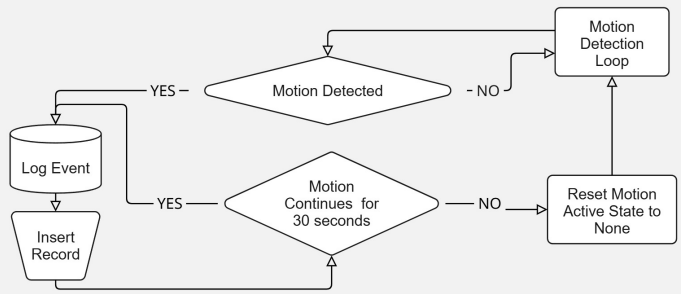


Figure 4.9: Flowchart Diagram of Logic with which the Motion Detection System Logs Motion Active Events to the Supabase Database.

4.4.5 Failure Management

Failure Aspect	Description
Supabase Connectivity Errors	- All database interactions, including inserts and updates, are wrapped in try-except blocks to catch exceptions related to network failures, API timeouts, or invalid credentials. - On failure, error messages are logged locally or printed to console to aid debugging without halting the system. - The system retries or skips updates gracefully, ensuring non-blocking operation despite transient connectivity issues.
Process Control Reliability	- Before launching the motion detection subprocess, the code checks if an existing instance is running to prevent spawning multiple conflicting processes, which could cause resource exhaustion or inconsistent states. - On stopping detection, the subprocess termination is verified to ensure proper release of system resources and avoid zombie processes. - Process state management uses flags and status checks to maintain synchronization between the main thread and subprocesses.
Camera Initialization and Stability	- The system establishes connection to the pigpio daemon for PWM control; failure to connect raises exceptions immediately, preventing further execution and alerting the user to the missing daemon. - A background frame stabilization phase captures multiple frames to compute a reliable background model, which mitigates false positive detections caused by camera noise, lighting changes, or transient movements. - The frame difference threshold and minimum contour areas are calibrated dynamically or predefined to balance sensitivity and false alarms.
Thread Safety and Race Conditions	- Access to shared data structures such as the frame buffer, motion detection flags, and global state variables are protected with thread locks (mutexes) to prevent simultaneous read/write conflicts. - This ensures data consistency when multiple threads, e.g., camera capture, motion detection, and logging, operate concurrently. - Locks are carefully applied only where necessary to minimize performance bottlenecks and avoid deadlocks.
Graceful Shutdown and Cleanup	- The system registers cleanup functions using Python's 'atexit' module and signal handlers to capture termination events such as interrupts or exceptions. - On shutdown, active flags in Supabase are updated (e.g., 'perimeter_active' set to 'False'), guaranteeing the database reflects the true system status and prevents stale data. - Hardware states like servo enable pins and GPIO outputs are safely reset to avoid leaving actuators energized or in undefined states.
Motion Event Logging Resilience	- Motion event data including timestamps and flags are sent to Supabase with error handling to catch API or network failures. - Failure to log does not disrupt the main detection loop; instead, errors are logged locally, ensuring continuous system operation. - This design supports robustness in environments with intermittent connectivity or backend outages.

Table 4.13: Failure Management Strategies and Error Handling in the System

4.5 Acceptance Testing

4.5.1 Tests

Table 4.14: Acceptance Tests for the Sensing Subsystem

Test ID	Description	Testing Procedure	Pass/Fail Criteria
AT01	Verify motion detection functionality	1. Connect Raspberry Pi to power and boot the system. 2. Confirm the motion detection service has started (via logs or 'systemctl status'). 3. Walk laterally across the frame at a steady pace. 4. Observe the live stream and check for bounding box appearance.	**Pass:** - Bounding box appears within 1 s of motion. - Supabase updates 'motion_detected' flag to 'True'. - Timestamp field is written with current time. **Fail:** - No bounding box appears. - Supabase does not reflect the event or shows incorrect timestamps.
AT02	Test motion state update frequency (10 Hz)	1. Modify code to log timestamp of every motion check cycle. 2. Trigger motion continuously by moving hand in front of the camera. 3. Let the system run for at least 5 s and export logs. 4. Compute time differences between consecutive entries.	**Pass:** - More than 10 cycles in a second. **Fail:** - Fewer than 10 cycles in a second.
AT03	Verify autonomous remote operation	1. Boot Raspberry Pi with working firmware. 2. Confirm WiFi connection established (check 'ifconfig' or frontend heartbeat). 3. Open frontend on a separate device and test the control buttons. 4. Ensure 'Start', 'Stop', and 'Live View' functionalities respond appropriately.	**Pass:** - Frontend connects within 30 s of boot. - All remote commands execute with 2 s delay. - Live stream launches and reflects real-time video. **Fail:** - Connection fails, commands don't respond, or live feed is inaccessible.
AT04	Validate accuracy of motion coordinate output	1. Walk into frame. 2. Observe bounding box generated. 3. Find realistic bounding box by taking an accurate bounding box around your own body in the frame. 4. Compare area of generated bounding box to the realistic bounding box	**Pass:** - Generated bounding box area occupies >50% than realistic bounding box. **Fail:** - Generated bounding box area occupies <50% than realistic bounding box.
AT05	Evaluate latency and framerate of video feed	1. Record the live feed and an external stopwatch simultaneously. 2. Trigger fast movement (e.g., waving hand) in front of the camera. 3. Playback the footage side-by-side and measure delay. 4. Measure how long it takes for the video to display 20 new frames.	**Pass:** - Measured latency 0.5 s. - 75 frames appear in 5 s (15 FPS). **Fail:** - Delay exceeds 0.5 s or frame count <75.
AT06	Test functionality in varied lighting conditions	1. Perform motion tests under two conditions: direct sunlight, and nighttime. 2. In each case, walk slowly through the frame. 3. Observe bounding box detection.	**Pass:** - Detection occurs in all scenarios. - Bounding box appears. **Fail:** - Detection fails in any lighting condition. - No bounding box, or incorrect bounding box.
AT07	Ensure environmental noise is minimised	1. Perform a very minimal movement in the frame (small object) repeatedly throughout a 5 minute period. 2. Check if bounding box is created and if alert was triggered at frontend. 3. Wait for recalibration and perform test again.	**Pass:** - No false positives over the 5 minute interval prior to recalibration. - No motion detected after recalibration in same environment. **Fail:** - Small movement recognised as motion, OR. - Recalibration has no impact.
AT08	Confirm stable operation under power load	1. Connect Subsystem to 12 V 1 A DC supply. 3. Stream video and run motion detection. 4. Monitor DC power supply ammeter reading and check Overload emergency light.	**Pass:** - System runs continuously with no reboots. - Motion detection and streaming uninterrupted. - Drawn current ≤1 A. **Fail:** - System crashes, overheats, or freezes. - Overload emergency light flickers or turns on.

4.5.2 Analysis of Testing

Table 4.15: Acceptance Test Data, Results, and Pass/Fail Status

Test	Data Collected	Analysis / Results	Pass
AT01	- 5 trials bounding box time: less than 0.1 s - Supabase flag updated within 0.5 s - Accurate timestamps logged	- Latency <1 s met - Updates consistent and accurate - Functionality verified	Yes
AT02	- Reference Figure is Figure 4.10a - Logged from 17:51:53.24 to 17:52:10.57 - Corresponds to 17.33 s - Total number of cycles = 520	- Frequency = 30.01 Hz - Equal to FPS of video feed (expected for very low processing times)	Yes
AT03	- Boot-connection avg: 8.9 s (3 runs) - Command delays: Start 0.5 s, Stop 0.5 s, Live View 1.2 s	- Connection <30 s - Commands <2 s delay	Yes
AT04	- Reference Figure is Figure 4.10b - Area of Realistic Bounding Box (red) ≈ 163x344 pixels - Area of Generated Bounding Box (green) ≈ 146x267 pixels	- The percentage coverage of the generated bounding box to the realistic bounding box is 69.52% - This meets acceptance test pass requirements, but it could be improved further	Yes
AT05	- Latency: 0.5 s avg - 20 frames took approximately 0.67 s	- Latency <0.5 s - Frame rate ≈ 30 FPS, therefore >15	Yes
AT06	- Reference Figure is Figure 4.10c - Detection in daylight - No motion detected at nighttime	- No detection occurs in nighttime scenario - No Bounding Box appears	No

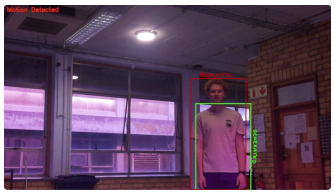
Continued on next page

Table 4.15 – *Continued from previous page*

Test	Data Collected	Analysis / Results	Pass
AT07	- Reference picture is Figure 4.10d - 5 min idle with small object oscillating: 0 false positives detected - Zero false positives post recalibration	- No detections of small movements (motion threshold is correct) - After recalibration background is reset to no motion	Yes
AT08	- 5 min runtime at 12 V, 1 A - Current draw 0.46 - 0.65 A under max load - No crashes or overload	- Stable operation with no reboots - No overload and current drawn < 1 A	Yes

2025-05-25T17:51:58.170459	FALSE
2025-05-25T17:51:58.185756	FALSE
2025-05-25T17:51:58.202267	FALSE
2025-05-25T17:51:58.264981	FALSE
2025-05-25T17:51:58.280184	FALSE
2025-05-25T17:51:58.296897	FALSE
2025-05-25T17:51:58.330155	TRUE
2025-05-25T17:51:58.361240	TRUE
2025-05-25T17:51:58.424864	TRUE
2025-05-25T17:51:58.474844	TRUE
2025-05-25T17:51:58.494098	TRUE
2025-05-25T17:51:58.538285	TRUE
2025-05-25T17:51:58.557979	TRUE
2025-05-25T17:51:58.570680	TRUE
2025-05-25T17:51:58.602776	TRUE
2025-05-25T17:51:58.665009	TRUE
2025-05-25T17:51:58.682721	TRUE

(a) Table of logs for every Update Cycle during Motion Detection for AT02



(b) Picture of Generated Bounding Box (green) and Realistic Bounding Box (red) for AT04

(c) Live Video Feed of Detection Algorithm during Nighttime with target placed on tree for AT06

(d) Picture of stable background with small oscillating object in top right corner for AT07

Figure 4.10: Acceptance Testing images to Accompany Data Collected and Justify Results

4.6 Subsystem Conclusion

This section documents the design process, architecture, and testing of the sensing subsystem. The subsystem's purpose is to identify the location of a moving target, log the motion event, and trigger actuation control. It was required to detect significant motion in the camera frame while rejecting environmental noise, working within lighting, background, and computational constraints.

The final solution uses a Raspberry Pi 3B+ and a Pi Camera Module V2 NoIR, processing video frames with an adapted Frame Differencing algorithm. A bounding box is drawn around the detected object, streamed to the frontend, with its centre used as the target coordinate for the Actuation Control subsystem. The entire system enables real-time motion detection. Integration with other subsystems was seamless: Actuation Control used the bounding box centre to aim motors, and the frontend—connected via a Flask server on the Pi—could remotely start, stop, and view the feed. Motion events were logged to Supabase, updating the UI.

Despite meeting key requirements, testing revealed some specifications were not fully met:

- Motion detection latency remained under 1 second, meeting real-time performance requirements.
- System maintained 30 FPS, confirming low processing overhead and real-time capability.
- All frontend commands (start, stop, live view) responded in under 2 seconds.
- The generated bounding box covered 69.5% of the actual object which is above the required amount.
- Motion detection failed in low-light/nighttime conditions, violating robustness expectations.
- No false positives were triggered by small or oscillating background objects after recalibration.
- System remained stable under full load, with current draw well below 1 A and no overheating or crashes.

4.6.1 Subsystem Recommendations

To address the failed acceptance test (AT06), several improvements are suggested. For the nighttime detection failure (AT06), it is recommended to design a simple infrared (IR) LED circuit to illuminate the camera's field of view. Since the Raspberry Pi Camera Module V2 NoIR already supports night vision, this enhancement would enable proper motion detection in low-light environments. The power requirements of the IR LED circuit would fit into the DC power supply of 12 V and 1 A.

Additionally to increase bounding box accuracy, the current frame differencing approach can be improved by tuning the binary threshold. Alternatively, more robust detection algorithms could be considered, including background subtraction methods like MOG2, optical flow techniques, or lightweight deep learning-based object detectors such as YOLO or MobileNet-SSD, provided computational resources permit.

Chapter 5

Mechanical and, Actuation Control Design - FRDETH004

5.1 Introduction

This section focuses specifically on the mechanical design and actuation control of the design. This subsystem is the physical backbone that links perception to action. The problem involved designing a solution that was able to receive sensor information from the Raspberry Pi (Chapter 4), and dynamically orient the deterrent method towards the 'predators' so that the detected objects (predators) can be accurately tracked and deterred.

The accurate target acquisition involves converting 2D image coordinates into precise pan/tilt angles for the corresponding motors. Without a reliable mechanical mount, any accurate target acquisition would prove to be useless due to physical disturbances, and any control algorithm would struggle with the constant output disturbances. Additionally, the mechanical mount and housing are designed to be deployed in the De Hoop Nature Reserve, and so environmental factors need to be considered.

5.1.1 Scope and Limitations of subsystem

In this chapter, we present the design, implementation, and testing of the mechanical and actuation control subsystem. This module converts 2D image coordinates into precise pan–tilt motions and includes a weather-resistant housing and mounting assembly, ensuring the system fully satisfies its performance requirements.

This chapter does not cover detailed modal fatigue or analysis, dynamic load testing in environmental conditions such as wind/rain however, basic analysis was performed for design purposes. It only briefly mentions external mounting. Budget and time constraints thoroughly limited the design and implementation of this project. The design is further limited in that it must be easy to install with low footprint, and limited environmental effects. As this build represents an early prototype, the enclosure was only fabricated in PLA and subjected to preliminary testing procedures. Full IP-rated weather-proofing methods could not be implemented or formally tested within the project timeline. Consequently, field validation under real coastal conditions remains outstanding and will be addressed in the next development iteration.

5.2 Requirements and Specifications

Christina Hagen has underscored the urgent need for a targeted intervention to protect the penguin colony at De Hoop Nature Reserve from predatory threats. In response, the requirements listed in table Table 5.1 were established specifically for this mechanical and actuation control subsystem. These criteria were derived not only from the project's core functional objectives—accurate, responsive pan-tilt tracking—but also from the demanding environmental and structural conditions of the reserve.

5.2.1 Requirements

The requirements for the mechanical and actuation control subsystem are listed in table 5.1.

ID	Description
R1	Two-axis rotation of deterrent.
R2	Targeting is centered on the object.
R3	Supports mass deterrent
R4	Stiffness and rigidity under maximum torque.
R5	Weather resistant.
R6	Easy to service and maintain condition.
R7	Housing should fit all electronic components.
R8	Easy to install and move around (low footprint, no permanent damage where installed).

Table 5.1: Requirements

5.2.2 Specifications

The specifications are refined from the requirements listed in Table 5.1 and are shown in Table 5.2

ID	Description
S1	Pan and tilt rotations each with 120° range.
S2	Steady-state error $\leq \pm 2^\circ$.
S3	Supports mass $\gtrapprox$ 500g.
S4	Deflection $\leq$ 0.5mm at 0.5N m per axis.
S5	Weather rating of materials $\geq$IP65.
S6	Modular sub-assemblies, quick release mounts.
S7	Housing should be $\approx 150 \times 100 \times 50$mm

Table 5.2: Specifications

5.2.3 Acceptance Testing Procedures

A summary of testing procedures are detailed in Table 5.3

Test ID	Description	Procedure	Pass/Fail Criteria
AT01	Rotation Range	Command pan and tilt motors to their maximum and minimum duty cycles. Measure the actual angle with a protractor.	Pass if both axes achieve $\geq 90^\circ$ rotations.
AT02	Pointing Accuracy	Place a static target at the center of the frame, and at each corner of the frame. Command the system to track each target, record actual servo angles.	Pass if steady-state error is $\leq 2^\circ$ for all tested positions.
AT03	Load Support	Mount a 500g test weight on the platform. Run a full pan/tilt sweep. Observe motion for any signs of stalling, strain, or inaccurate tracking.	Pass if servos move smoothly through the full range without stalls, strain or inaccuracies.
AT04	Structural Integrity	Apply a known 0.5N m torque at the end of the pan/tilt gimbal. Measure resulting deflection via laser displacement sensor.	Pass if the maximum deflection $\leq$ 0.5mm under the applied torque.
AT05	Weather Resistance	Expose the sealed housing to environmental conditions by: (1) placing the housing in dust and moving it around; (2) spraying a water-jet ($12.5\text{L}\,\text{min}^{-1}$ at 30kPa).	Pass if no dust or water enters the enclosure.
AT06	Modular Housing	Have someone remove and reinstall the assembly. Time the operation and verify correct alignment and positioning	Pass if done in $\leq$ 5 minutes without the use of tools, and full operational functionality restored
AT07	Housing Fit	Install all electronics into the enclosure. Measure clearances and external dimensions.	Pass if all components fit with $\geq$ 10mm clearance and housing is $\approx 150 \times 100 \times 50$mm.

Table 5.3: Acceptance Test Procedures

5.2.4 Traceability Analysis

Table 5.4 highlights the links between the requirements (Table 5.1), the specifications (Table 5.2) and the testing procedures (Table 5.3).

#	Requirement	Specification	ATP
1	R1	S1	AT01
2	R2	S2	AT02
3	R3	S3	AT03
4	R4	S4	AT04
5	R5	S5	AT05
6	R6	S6	AT06
7	R7 and R8	S7	AT07

Table 5.4: Traceability Matrix for all requirements, specifications and acceptance test procedures.

5.3 Subsystem Design

This section is broken up into the design of the mechanical module, which includes the design of the housing and the pan and tilt gimbal; and into the design and development of the actuation control.

5.3.1 Design Decisions

Actuator Selection

In choosing the drive elements for the pan-tilt gimbal, three motor types were evaluated against key criteria: torque, positional accuracy, control complexity, weight, cost and environmental robustness.

Type	Pros	Cons
DC Motor	Offers high continuous torque and a wide speed range.	Requires a separate H-bridge driver and rotary encoder; closed-loop position control must be implemented in software; larger footprint with higher current draw; more complex wiring and tuning.
Stepper Motor	Provides inherent open-loop positional control via discrete steps, good low-speed torque, a simple driver interface, and available Python library support.	Can lose steps under high load or vibration; requires microstepping drivers for smooth motion; typically has high power consumption.
Micro Servo Motor	Features an integrated gearbox with closed-loop feedback, a simple PWM interface, compact and lightweight form factor, Python library support, adequate torque and resolution, and low integration complexity.	Limited to a fixed angular range (±90°); constrained by the manufacturer's internal control loop parameters; potential for PWM-induced jitter causing small oscillations.

Table 5.5: Comparison of different drive elements

Based on the comparison in Table 5.5, we selected the micro-servo motor to drive the pan–tilt gimbal. Its built-in gearbox and closed-loop controller minimize integration overhead, and its compact size and low weight meet our mechanical constraints. As the three motor types fall into the same price bracket, cost did not influence our decision. The MG90S Micro-Servo Motor was chosen since its metal gears mean that it can withstand higher loads and resist wear far better than plastic-geared alternatives (such as the more common SG90S), resulting in reduced backlash, longer service life, and more reliable, precise positioning under continuous or high-torque operation.

Housing Material Selection

In selecting the enclosure material for our pan–tilt gimbal and housing, we balanced mechanical stiffness, environmental resistance, fabrication complexity, and per-unit cost. Table 5.6 summarises each material's advantages, drawbacks, and approximate material-plus-processing cost. Anodised 6061-T6 aluminum would

Material	Pros	Cons	Est. Cost (USD/ZAR)
UV-stable ASA (3D-print filament)	Inherently UV- and weather-resistant, very low weight, rapid prototyping on desktop FDM, and excellent chemical resistance.	Lower stiffness, requiring thicker walls or ribs; layer lines may absorb moisture over time; limited thermal range ($\sim 20^\circ C$ to $+80^\circ C$).	$\approx$ \$7($\approx R126$)
Polycarbonate (PC)	High impact resistance, good UV and weather resistance with UV-stabilized grades, and compatibility with CNC machining or injection molding.	More expensive than ASA; moderate stiffness requiring reinforcement; may yellow under prolonged UV exposure without coating.	$\approx$ \$30($\approx R540$)
6061-T6 Aluminum (anodized/powder-coated)	Excellent strength-to-weight ratio, very stiff under load (deflection $< 0.5mm$ at $0.5N\mathring{u}m$), good thermal conduction for heat dissipation, and proven corrosion resistance.	Requires CNC or metal-working tools; higher material cost compared to plastics; potential for galvanic corrosion when mated with dissimilar metals.	$\approx$ \$50($\approx R900$)
Fiberglass-Reinforced Polymer (FRP)	Outstanding stiffness-to-weight ratio, inherently waterproof and corrosion-proof, low thermal expansion, and ability to mold complex shapes.	Requires hand-layup or molding tooling; higher material and labor cost; health and safety concerns during fabrication.	$\approx$ \$100($\approx R1800$)
316 Stainless Steel	Best-in-class corrosion resistance in marine environments; very durable and wear-proof.	Very heavy, increasing actuator loads; highest material and machining cost; poor thermal conductivity that may trap heat.	$\approx$ \$150($\approx R2700$)

Table 5.6: Comparison of candidate enclosure materials

be chosen for the enclosure because it delivers the best balance of stiffness, weather resistance, and cost for our seaside deployment. In testing, its high strength-to-weight ratio keeps deflection well under 0.5 mm at 0.5 N · m, and the anodized finish provides proven protection against salt spray, UV exposure, and temperature extremes. At roughly \$50($\approx R900$) per unit, it costs more than printed plastics but remains far more durable and machinable than FRP or stainless steel, making it the optimal choice for a long-lasting, precision housing.

The final prototype will be developed through 3D printing using PLA filament. The benefit of this is the allowance for cheap and fast prototyping, which allows for various iterations of models to be developed and tested.

5.3.2 Final Configuration

The final configuration provides a complete mechanical and actuation-control subsystem, focusing on CAD geometry, assembly interfaces and actuation integration.

Mechanical Enclosure

According to S1 and S3 in Table 5.2, the pan-tilt gimbal is required to enable 120° of rotation while supporting a mass of $\gtrapprox$ 500g. For this, a simple two-axis design was developed:

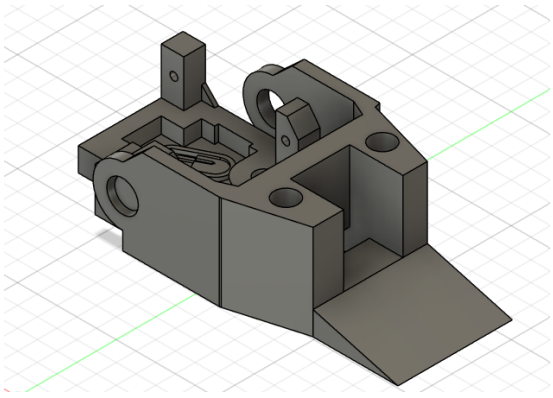


Figure 5.1: Two Axis Pan-Tilt Design

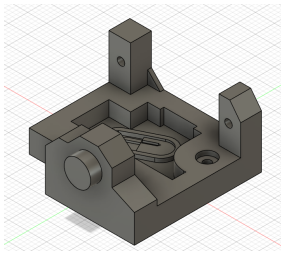


Figure 5.2: Isolated Pan Axis Design

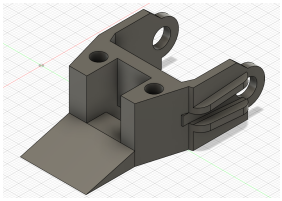


Figure 5.3: Isolated Tilt Axis Design

The pan assembly is founded on a sturdy base plate with a precisely machined opening that seats the servo's output gear flush, eliminating backlash and dampening unwanted vibrations for smooth, repeatable rotation. A central pocket aligns the tilt axis concentrically with the pan bearing, minimising kinematic offsets and maximising tracking accuracy (see S2,Table 5.2). Two vertical towers secure the tilt servo, keeping its output shaft orthogonal to the pan plane. A mounting wall provides a rigid interface for the tilt subassembly, ensuring precise alignment under load. All critical dimensions are shown in Figure 5.5.

The tilt stage fastens directly to the pan via matching bores, guaranteeing concentric motion between axes. A recessed servo mounting pocket increases lever arm for larger payloads, improving torque capacity and load balancing (see S3,Table 5.2). Triangular gussets tie the vertical arms back into the module base, stiffening the joint against bending and torsion in accordance with the stiffness requirement (see S4,Table 5.2). A built-in ramp and flanking boss holes accommodate interchangeable deterrent mounts. Prototype testing used a laser pointer for rapid validation. Detailed drawings of the pan and tilt gimbal are shown below (Figure 5.4,Figure 5.5) for all dimensions:

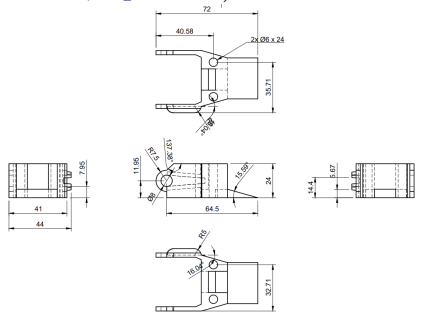


Figure 5.4: Mechanical Drawing of tilt design

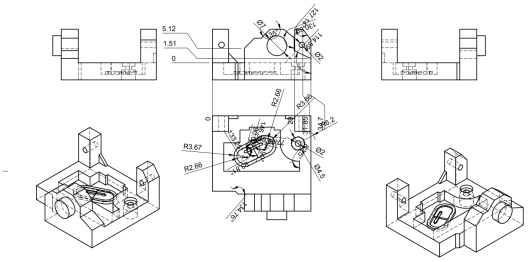


Figure 5.5: Mechanical Drawing of pan design

For further clarity, these are shown in subsection A.2.1. With the pan-tilt gimbal designed, the focus shifts to the protective enclosure that houses all electronics, ensures IP66 weatherproofing and connects the gimbal to the rest of the system. The enclosure comprises a rigid base case, and a removable lid, connected via a hinge and a snap-lid. The case and lid are engineered for structural strength, reliable sealing and straightforward assembly.

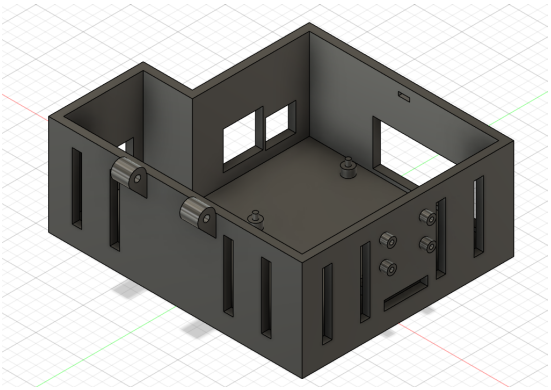


Figure 5.6: Isometric View of case design

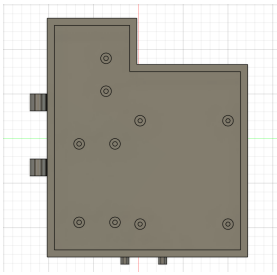


Figure 5.7: Top view of case design

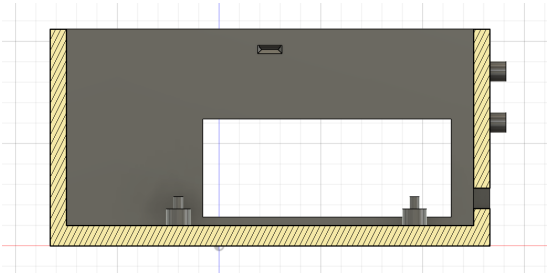


Figure 5.8: Side View Sectional Analysis of case design

The enclosure base is molded with robust walls and a reinforced floor to create a rigid shell for all control electronics. Integrated standoffs ensure precise board placement, while paired hinge knuckles and a matching retention pin lock the lid without external fasteners. Exterior bosses align and secure the camera module, and a discreet cable channel maintains sealing integrity by routing all wiring internally. Strategically placed cutouts grant service access to power and I/O ports and provide passive airflow for temperature control. All critical tolerances and layout details are specified in Figure 5.15 Planned refinements will replace open vents with IP-rated cable glands, over-mold the camera feed for complete weatherproofing, and add a modular clip interface for fence mounting.

The lid design completes the hinge of the base case (Figure 5.6); it also features internal walls to prevent slipping and movement that may occur during high-speed movement of the pan and tilt gimbal (Figure 5.1). Figure 5.9 shows the pan servo's internal mounting, with a wiring slit and a drill hole for flush mounting. The case lid dimensions are specified in Figure 5.16.

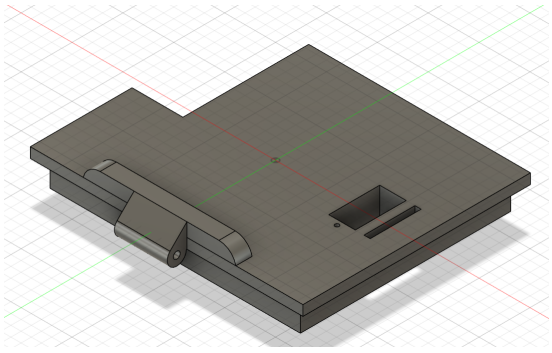


Figure 5.9: Isometric View of the lid design

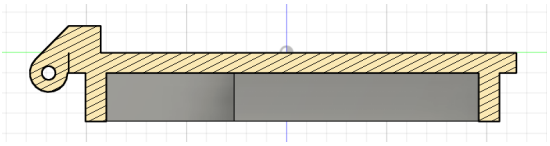


Figure 5.10: Side View Sectional Analysis of lid design

The mechanical drawings of both the case and lid are shown below (Figure 5.15,Figure 5.16):

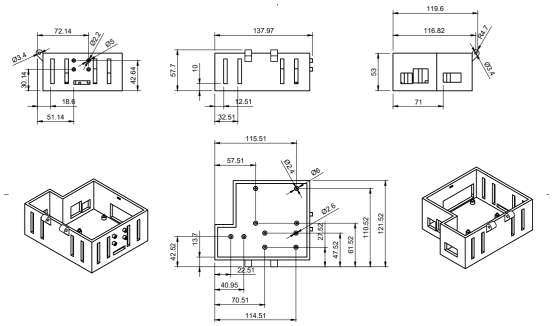


Figure 5.11: Mechanical Drawing of case design

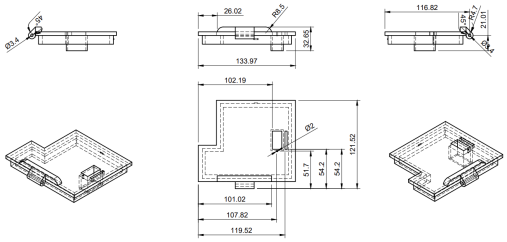


Figure 5.12: Mechanical Drawing of lid design

With all the components designed, the final Mechanical Enclosure is shown below (Figure 5.13):

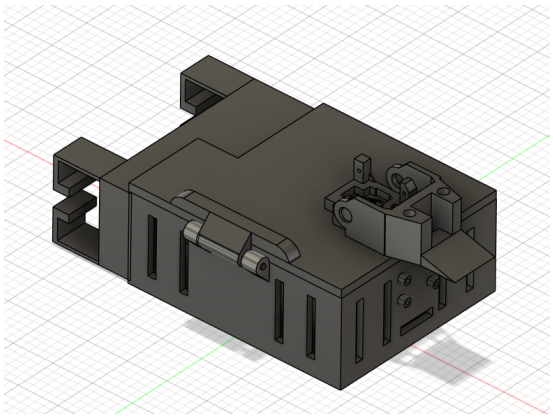


Figure 5.13: Final Combined Mechanical Enclosure

Figure 5.13 can be exploded to view all components and how they connect:

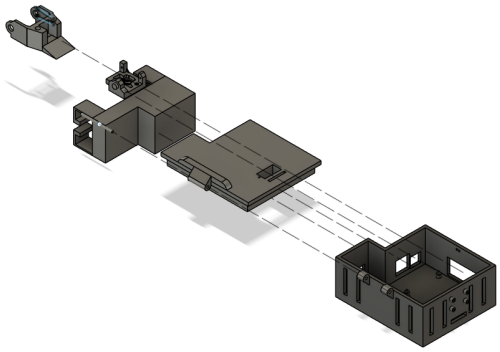


Figure 5.14: Exploded isometric view of Figure 5.13

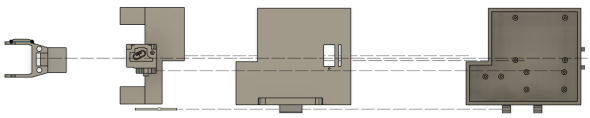


Figure 5.15: Exploded top view of Figure 5.13.

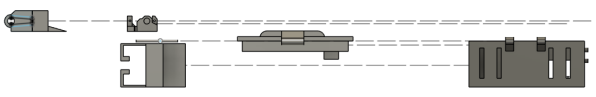


Figure 5.16: Exploded view of Figure 5.13.

Actuation Control

Since the control loop relies on 2D object centroids extracted from video frames, it does not make sense for external sensors such as rotary encoders, IMUs, ToF sensors, or laser displacement sensors to be involved. The sensor chosen in chapter 4 is suitable since it acquires the video frames of the object as well as the current position of the predator deterrent (laser-pointer for the prototype), which is converted to coordinates, and then the required angles for the servo motors in the microcontroller. A detailed feedback loop is shown in Figure 5.17:

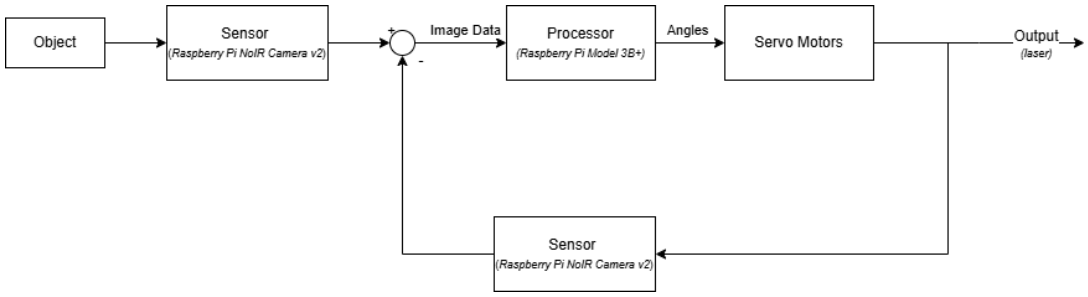


Figure 5.17: Control Loop

The Raspberry Pi NoIR Camera v2 captures each video frame and extracts pixel coordinates for the moving object's centroid and the laser dot. The processor converts their pixel difference into angular errors using the lens's calibrated field-of-view, then feeds those errors into two PID controllers that compute corrective angle offsets. Those offsets are translated into PWM signals to update the servos in real time, creating a continuous feedback loop that compensates for disturbances and maintains precise laser alignment on the target.

Despite the clear benefits of laser-dot feedback for closed-loop control, implementing this was not possible. The Raspberry Pi struggled to outline both the object's centroid and the laser spot. By using an NoIR camera, the sensor is sensitive to both visible red (the laser) and ambient near-IR (sunlight, LEDs incandescent lamps). The extra IR noise raises the background level, and so the laser's red is no longer distinct and is mistaken for background by the Raspberry Pi.

In order to overcome this for future development, one can add an optical band-pass filter ($\approx$ 450nm over the lens, which blocks most IR noise, and only passes the laser's light. Alternatively one could simply not use a NoIR camera module, however since honey badgers are nocturnal (*chapter* 3), this does not make sense in the context of the problem.

Since the laser-dot feedback closed-loop control was unable to be implemented, the prototype operates in open-loop mode, using only the object's centroid to command pan and tilt angles. This is acceptable, since the mechanical enclosure was designed such that the axis of rotation, and laser pointer, lie directly above the camera module's center.

5.3.3 Failure Management

Figure 5.18 shows the failure management techniques implemented:

Name	Description
Servo Motor Stall and over-rotation	Stall may occur under excessive torque or rapid, large movements. To prevent this, software constraints were introduced which prevents the servo motor from making continuous large movements. Additionally, maximum angle thresholds were produced in order to limit over-rotation.
Loss of target	Due to lighting changes or software errors, occasionally targets may be lost which may result in erratic motion. To prevent this a 'park' routine was implemented which maintains the servos previous position, eventually resetting to the rest position if no new motion is detected.

Figure 5.18: Table showing failure management techniques.

In order to enhance system robustness in the future, it's recommended that a separate dedicated servo motor power regulator be implemented with built-in overcurrent and undervoltage lockout, with a capacitor to overcome brief brown-outs.

In terms of weather-proofing failure management, an internal humidity or moisture sensor would be implemented in future developments. This will alert or log any ingress which would prompt maintenance before corrosion or water leaks can damage the components and system. This is not implemented in this prototype.

5.3.4 System Integration and Interfacing

Figure 5.19 shows the electrical integration of this subsystem with the rest of the system:

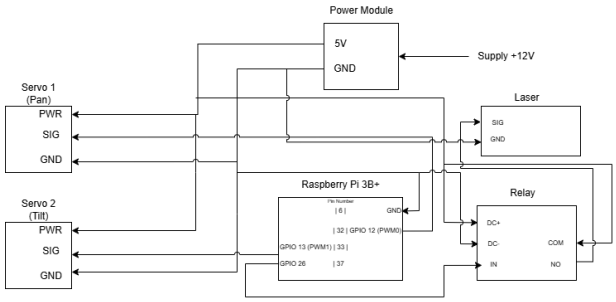


Figure 5.19: System Interfacing Diagram showing the how the actuation system integrates with the rest of the system

Software Interfacing

In order to control the servo motors, the signal wires were connected to PWM0 (GPIO12)and PWM1 (GPIO12) as shown in Figure 5.19. Using the 'pigpio' library, initialising the GPIO pins was simple:

```
import pigpio
pi = pigpio.pi()
# Servo Setup
PAN_GPIO = 12   # GPIO for pan servo
TILT_GPIO = 13  # GPIO for tilt servo
# Laser Setup
LASER_GPIO = 26  #GPIO for relay module, connected to the laser output
```

Listing 5.1: Initialising GPIO

The servo motors, and laser pointer (connected through the relay module), were controlled simple by using the function `pi.set_servo_pulsewidth(GPIO,PULSE_WIDTH)`. Figure 5.20 shows the functional and data flow of moving the servo motors:

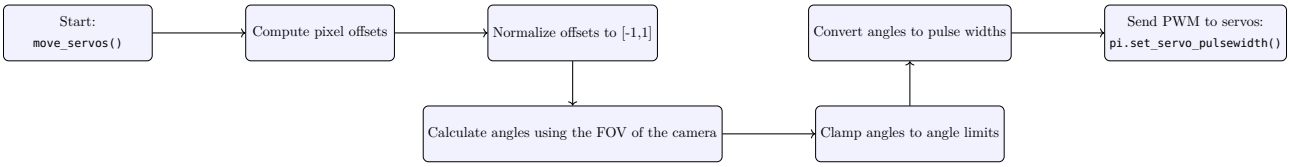


Figure 5.20: Flow diagram of the `move_servos` routine

The laser module is triggered, when an object is detected, by simply setting GPIO26 high, which in turn switches the relay. The relay was used in order to maximise the brightness of the laser, and prototype how other predator deterrents would be implemented.

Mechanical Mounting

As shown in Figure 5.6, the electrical system integrates to the mechanical system through the use of mounting holes. This allows for a flush connection, stabilising all electrical components to prevent movement, and the tangling of wires. Additionally, in Figure 5.9 the mounting for the pan servo motor is clearly indicated, also allowing for a flush connection with drill holes to secure the servo motors to prevent movement. The pan-tilt gimbal in Figure 5.1 shows how the tilt servo connects to both the pan and tilt components.

5.3.5 Acceptance Testing

Test ID	Description	Result
AT01	Rotation Range	Pass
AT02	Pointing Accuracy	Pass
AT03	Load Support	Pass
AT04	Structural Integrity	Pass
AT05	Weather Resistance	Pass in simulation
AT06	Modularity	Pass
AT07	Housing fit	Pass

AT01
The servo motors were commanded to perform a full sweep while connected to the pan/tilt gimbal. The gimbal allowed for full 120° motion. Results can be seen in Figure 5.21 and Figure 5.22. AT01 is therefore considered a pass.

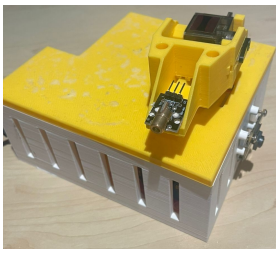


Figure 5.21: Maximum pan provided by the gimbal

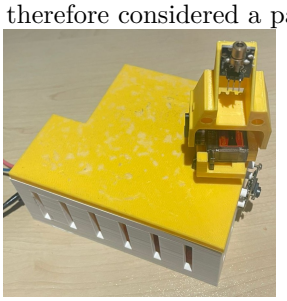


Figure 5.22: Maximum tilt provided by the gimbal.

AT02
Once the system was fully integrated, static targets were placed at the center, and each corner of the camera's frame. The laser's position was tracked, and the resulting error was found. For the center of the frame, the object was perfectly tracked. For the corner frames, there were slight offsets (within 2°) from the center in both the x and y axes. The tracking of a static object is shown in figure Figure 5.23. The green cross marks the center coordinates, the bright red dot is the laser (where the servo motors are essentially pointing). AT02 is therefore considered a pass.

This figure clearly shows how that despite the lack of negative feedback control, the accuracy is still high through the open-loop mapping from pixels to servo angles (duty cycles). The difference between the object's centroid, and the laser position is due to the physical offsets between the centroid of the camera, and the laser pointer.

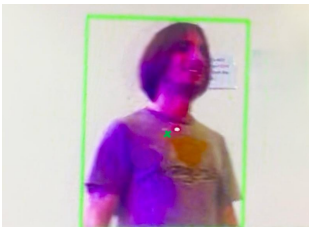


Figure 5.23: Accuracy of tracking

AT03
A 500g test weight was placed on the tilt gimbal, despite the added weight, the servo motors were able to function normally, while operating under increased torque. This is due to the internal controller inside each servo motor which regulates speed and torque. It is noted that the servo motors exhibited very slight strains, but not enough that affect it's continued operation. AT03 is therefore a pass.

AT04
A 0.5N m torque was applied around the z, x, and y axes in order to test the structural integrity of the pan and tilt gimbal. The measured deflection was $0.3mm$. AT04 is therefore a pass.

AT05
AT05 was only able to be tested through simulation due to the final design being a prototype. In simulation, with the correct fasteners, enclosures for the camera modules, and other weather proofing techniques described in section 5.3.2. AT05 was tested by performing jet impact test in autodesk CFD. This involved setting up a water-jet to simulate wind driven rain. The flow rate was $12.5\text{L}\,\text{min}^{-1}$ at 30kPa as per the testing procedure. This validated that the enclosure is likely to hold the desired IP66 rating, however it isn't a guarantee. Real testing procedures would need to be performed when the enclosure is out of prototype phase.

AT06 & AT07
The design was designed to be modular and fit all the required electronic components. Assembly and disassembly was timed to be 4 minutes, and all components fit within the desired clearances. Both AT06 and AT07 passed. Figure 5.24 and Figure 5.25 show the components fit, along with the modularity of the design.

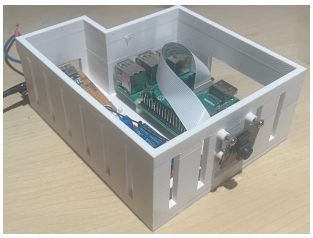


Figure 5.24: Isometric view of the prototype case, showing the fit and clearance of all electronics.

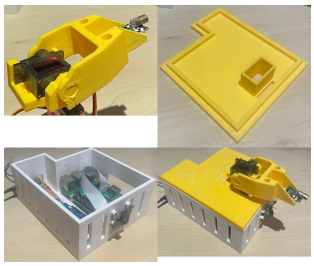


Figure 5.25: Consecutive image showing the modularity of the design

5.3.6 Recommendations

Several enhancements can improve the robustness, accuracy and maintainability of the mechanical and actuation-control subsystem. It's recommended that in future iterations, a narrow-band optical filter be incorporated in order to tune the laser wavelength to restore contrast in the NoIR camera's feedback loop, and eliminate the IR-induced noise. Additionally, hardware level current monitoring should be incorporated on each servo-rail, in order to detect stall and overload conditions. A separated voltage power supply (such as another step-down power module) should be included in order to prevent brown-outs in the Raspberry Pi. The integrated module should be designed such that the tilt gimble is able to tilt to $-90°$, for a situation in which the predator walks below the system. Additionally, attaching the camera module to the rotation gimbal would prove useful for a larger field of view however, a different detection algorithm would need to be developed. These are considerations for future development.

5.3.7 Conclusion

The mechanical and actuation-control subsystem delivers a reliable two-axis pan-tilt pointing system with $120°$ range in both axes. The system has sub $2°$ steady-state accuracy with rapid response times, fully satisfying the functional and performance requirements. The anodised 6061-T6 aluminum material chosen for the housing, along with the precision-fit servo interfaces provided stiffness and rigidity needed for dynamic target tracking. The enclosure design, ensures weather resistance and serviceability in remote coastal environments. Although laser-dot feedback is deferred by processing constraints, the core hardware and control architecture provide a solid framework for field deployment. With the recommended additions mentioned in subsection 5.3.6, the subsystem can evolve into a robust, high precision deterrent platform with sustained operation despite demanding environmental conditions. Its important to note that although this design provides a complete solution, further implementation and testing is required for the weather proofing requirements due to the prototype nature of this design.

Chapter 6

User Interface, User Experience and Frontend Implementation - ISSAAR001

6.1 Introduction

This section focuses specifically on the user interface (UI), user experience (UX) and frontend implementation developed for wildlife conservation personnel. Its purpose is to allow users to observe detections of honey badgers near the fencing that surrounds the African penguins at De Hoop Nature Reserve. The subsystem displays real-time alerts when a predator is detected and can also show a visual of the predator. It is designed to be reliable, easy to use and quick to interpret. The engineering problem is how to transform raw predator detection data into clear, timely and actionable visual alerts for conservation personnel operating in remote and time-sensitive environments.

The UI plays a key role as the final step in the detection and tracking pipeline, converting backend data into actionable awareness for human users. Even with accurate sensing and tracking, poor or delayed communication can result in ineffective intervention. As such, this subsystem acts as the bridge between automated processes and human response, directly impacting the success of the penguin protection efforts.

6.1.1 Scope and Limitations

The scope of this subsystem is limited to presenting data that is already captured and processed by the sensing and tracking subsystems. It does not influence how detections are made but focuses on how that information is delivered and interpreted by users.

This version of the interface is designed to work for a Raspberry Pi-based deployment at a single colony site (De Hoop) and is not yet integrated with deterrent activation or multi-site management. Additionally, due to the time constraints of the project, the system assumes network availability and does not handle offline integration modes.

6.2 Requirements Analysis

After listening to a presentation from Christina Hagen about the devastating honey badger attack at the De Hoop penguin colony, where 11 endangered African penguins were killed in a single incident, it became clear that additional and more effective measures were urgently required. This event revealed a critical gap in the existing protection system, particularly in alerting conservationists early enough to respond. The goal of this subsystem is to deliver reliable real-time alerts and predator status updates through a simple and intuitive interface. The following section outlines the requirements, design specifications and testing procedures developed to help prevent such incidents from occurring again.

6.2.1 Requirements

The requirements for the UI subsystem are described in Table 6.1.

Table 6.1: Requirements of the UI subsystem.

Requirement ID	Description
FR01	The system must display an alert when a predator is detected.
FR02	The interface must update with minimal delay.
FR03	The interface must be easy to understand and usable by non-technical users.
FR04	The system must include a manual test button to simulate detections.
FR05	A live video feed must be available to visually confirm detections.
FR06	The system must clearly indicate a detection using a visual notification.
FR07	The system must show when the system is online and operating correctly.

6.2.2 Specifications

The specifications, refined from the requirements in Table 6.1, are shown in Table 6.2.

Table 6.2: Specifications of the UI subsystem.

Specification ID	Description
SP01	A clear and obvious popup alert should appear when a detection is logged. This must include an accompanying audio alert that is clearly noticeable.
SP02	New detections that are captured in a database must be reflected on the interface within 3 seconds of that detection.
SP03	The UI must include large, clearly labelled components without technical jargon and should be usable without prior training.
SP04	A “Simulate Detection” button must trigger the same UI alert logic as a real database detection.
SP05	The interface must stream a live camera feed at a minimum of 10 FPS. The resolution should be at least 640×480 to ensure users can clearly see what is detected even in low light conditions.
SP06	A toast notification system must be used as a background alert update.
SP07	The UI must show live updates as to whether the system is online using a green status indicator and offline using a red status indicator.

6.2.3 Testing Procedures

A summary of the testing procedures for the UI subsystem is provided in Table 6.3.

Table 6.3: Acceptance tests for the UI subsystem.

Test ID	Description	Testing Procedure	Pass/Fail Criteria
AT01	Simulated detection triggers alert	Click the ‘Simulate Detection’ button on the UI. Observe whether the entry is captured in the database and an alert appears on the UI.	A visual alert must appear within 3 seconds.
AT02	Real detection triggers alert	Trigger a detection from the sensing subsystem. This can be performed by walking in front of the Raspberry Pi’s camera module. Observe UI for response.	An alert must be displayed on the UI within 3 seconds.
AT03	Notification triggered	Trigger either a simulated or real detection.	A notification should appear on the screen of the dashboard with a sound alert within 2 seconds of a detection.
AT04	Live video feed is visible	Turn the system on by clicking the ‘Turn system on’ button and wait for the UI to successfully connect to the Pi (wait 15 seconds). Click on the view camera button in the UI.	The video feed must load and stream at a minimum of 10 FPS at 480p resolution.
AT05	UI clarity and simplicity	Present the interface to a user unfamiliar with the system. Ask them to describe what is happening.	The user must correctly identify that a predator was detected without needing further explanation.
AT06	System status indicator	Disconnect the Raspberry Pi from the internet or shut down the detection backend. Observe the UI’s status indicator. Reconnect and repeat.	A red indicator must appear when the system is offline and a green one when it is back online.

6.2.4 Traceability Analysis

To show how the requirements, specifications and testing procedures are linked, the traceability matrix is presented in Table 6.4.

Table 6.4: Requirements Traceability Matrix

#	Requirement(s)	Specification(s)	Acceptance Test(s)
1	FR01	SP01	AT01, AT02
2	FR02	SP02	AT02
3	FR03	SP03	AT05
4	FR04	SP04	AT01
5	FR05	SP05	AT06
6	FR06	SP06	AT04
7	FR07	SP07	AT03

6.3 Design

6.3.1 System Overview

The main goal of this design subsection of the project was to keep the user interface simple and easily understandable. Since, theoretically, the system is supposed to run on a Raspberry Pi in the field, the design also needed to be fast and able to load reliably. It was important that any detection be immediately visible without the user needing to dig for it, so all important elements are placed at the top of the interface so that they can not be missed. The system was also designed to be modular so that different components (like the video stream or alert logic) could be tested or adjusted independently before testing the UI with the sensing and tracking subsystems.

6.3.2 Technology Stack

Framework Selection

The choice of frontend framework was the first major consideration in this design subsection. Two options were considered: using a basic HTML/CSS/JavaScript setup, or opting for a more advanced framework like those listed in Table 6.5. The latter was chosen due to the additional features and advantages these frameworks offer.

Each option in Table 6.5 came with its own set of advantages and disadvantages. Angular, developed by Google, is known for offering a wide range of built-in features straight out of the box. React, developed by Meta, offers fewer features by default, but allows greater flexibility in how the project is structured and which modules are used. Vue, created by a former Google developer, is considered the easiest to learn and can be used effectively even without prior experience in HTML or JavaScript.

Table 6.5: Comparison of common UI frameworks

Category	**React**	**Angular**	**Vue**
Structure	Library focused on UI components	Full framework with built-in tools	Lightweight framework with optional features
Language	JavaScript or TypeScript	TypeScript	JavaScript or TypeScript
Learning Curve	Moderate	Difficult	Easiest
Performance	Fast and efficient	Heavier for small apps	Fast and lightweight
Community Support	Very large	Large	Medium but growing

Although all three options are widely used in the software development community, React was chosen for its flexibility in project structure, ease of handling dynamic UI logic and strong ecosystem of open-source resources. Its large community, with countless tutorials and pre-built libraries, was the final deciding factor.

Build Tool

To support development with React, Vite was selected as the build tool due to its simplicity and speed. It serves two key purposes: serving code locally during development and bundling all assets for production deployment.

UI Libraries and Styling

In addition to React and Vite, several supporting libraries and tools were used to help with the development process and ensure a well presented UI. Radix UI was used to provide the basic building blocks for interface elements like buttons and menus, ensuring they behave correctly and accessibly across different browsers. On top of that, shadcn/ui was used to provide pre-styled, accessible components built on Radix UI, while Tailwind CSS handled the styling across the interface. Tailwind CSS makes it quick and consistent to apply styling directly within each component, without writing long separate CSS files.

Backend Integration

For handling the backend and database, Supabase was chosen. It offers tools to manage data, user logins and real-time updates, all of which integrate easily with React.

6.3.3 Component Design

Alert Display System

When a detection is captured by the Pi, a new record is created in the Supabase detections table. This contains the location, timestamp, action taken and an 'acknowledged' flag set to false. A similar process occurs if an admin manually simulates a detection event using the 'Simulate Detection' form in the UI, where they can select a location (Perimeter A-D) and action taken (No action/Laser deployed) from dropdown menus before clicking the 'Simulate Detection' button. This form is implemented through the AddAlertForm component, which calls the addDetection function to insert a new detection into Supabase.

On the frontend, the system uses a custom React hook called useDetections that regularly polls the Supabase database to retrieve both recent and unacknowledged detections. When a new, unacknowledged detection is found, it triggers two visual elements: a prominent popup alert that appears in the center of the screen with a loud alarm sound and a status indicator at the top of the dashboard that changes from a green 'All Clear' to a red 'Honey Badger Detected!' message. The modal is implemented through the DetectionModal component, which displays the detection details and provides acknowledgment options.

For current alerts, authorized users (admins and field agents) see two buttons at the bottom of the card: 'Acknowledge' to clear that specific alert and 'Acknowledge All' to clear all current alerts. These cards are implemented through the AlertCard component, which handles the visual presentation and user interactions.

For a broader view of detection history, users can click the 'View Alert History' button in the dashboard, which opens a modal containing a table of past detections. The table shows timestamps, locations, actions taken and status of acknowledgments. At the top of this modal, users can click an 'Export CSV' button to download the latest 50 detections as a spreadsheet file. This functionality is implemented through the AlertHistoryDialog and AlertHistory components, which work together to display the data and handle the CSV export process.

The system maintains synchronization between the frontend and backend through regular polling of the Supabase database. When a user acknowledges an alert, they receive a green toast notification confirming the action and the alert card immediately updates to show the acknowledged state. This is handled by the acknowledgeDetection and acknowledgeAllDetections functions in the detectionService, which update the database and trigger UI updates.

This integrated approach gives users the ability to leave the web application throughout the day and come back at the end of the day to see activity throughout the day, with precise records easily available without needing to access the backend database. The combination of real-time alerts, historical records and export capabilities creates a comprehensive alert management system that serves both immediate response needs and long-term record-keeping requirements.

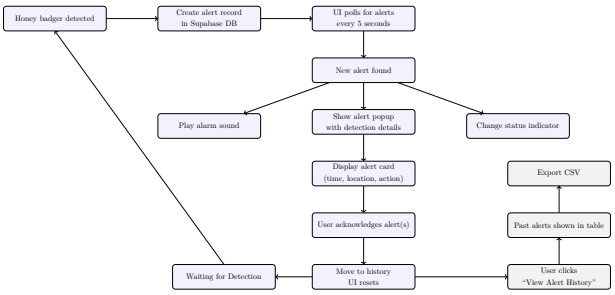


Figure 6.1: Final alert lifecycle: clean loopback and compact history panel

Video Feed and Pi Control Integration

The video feed integration in the system is focused around the CameraFeed component, which displays the live camera stream from the Pi on the dashboard. When a user clicks the 'View Live Camera' button, the CameraFeed component opens a modal dialog containing an image element that connects to the Pi's camera feed endpoint. The camera dialog's open/close state is managed within the CameraFeed component, but it can also be triggered from other parts of the application, such as the DetectionModal, by dispatching the open-camera-dialog event. This ensures that users can access the video feed from any context in the application, whether they're viewing an alert or simply monitoring the area.

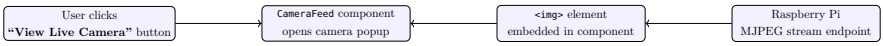


Figure 6.2: Flow of video feed integration from UI to Pi stream via `CameraFeed`

The PiControl component provides controls to start or stop the camera and detection process on the Pi by sending HTTP requests to the Pi's backend flask controller. During the startup process, a loading bar is shown to the user and the 'View Live Camera' button is temporarily disabled to ensure the system is ready before the video feed is accessed. This loading bar was implemented as a safety measure since the Pi takes around 10 seconds to be properly set up from when the Pi is started on the UI. The integration is further supported by the camera and controller URLs and utility hooks for managing state and side effects. Together, these components and supporting files ensure a seamless and responsive experience for viewing and controlling the live camera feed within the application.

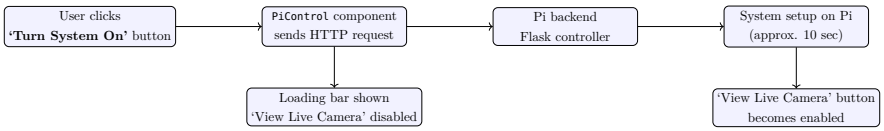


Figure 6.3: Flow of system startup and camera readiness using the `PiControl` component

Notification System

As soon as the detection is captured, it triggers two key notification mechanisms. The first is done using a toast notification system to provide immediate feedback to the users. A toast notification system was used since it able to display small messages at the bottom of the screen like a quick and temporary notification system. The toast system is implemented using a custom hook (use-toast) and is triggered in several scenarios throughout the app. For example, when a detection is successfully added, the addDetection function in detectionService.ts calls the toast function to display a message such as "Alert: Honey Badger Detected!" along with the location and time of the event. Similarly, toast notifications are used to inform the user of actions, such as acknowledging an alert.

In addition to the toast notification system, an audible notification system is also used. This is implemented by running a useEffect hook when the DetectionModal is opened and thereafter the hook creates a new Audio object that is pointing to an alarm sound that was added to the public folder in the project. The sound continues until the user acknowledges the alert or closes the modal.

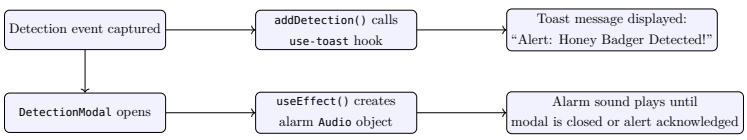


Figure 6.4: Flow of visual and audible notifications triggered by a detection

Perimeter status

The perimeter status is stored in the perimeters table in the database. Each record includes zone and a status where TRUE represents the system being online and FALSE when the system is offline. The system being offline occurs when the Pi is not connected to power, if the UI and Pi are not connected to the same WIFI or finally if the Pi's IP address has changed and has not been updated on the UI side. The UI uses a custom React hook called usePerimterStatus that fetches the current status of all perimeters from the database. The hook contains a function called updatePerimeterStatus that can be called to update a perimeter's status in the database. This function is used in components like PiControl, which checks the Pi's status and updates Perimeter A accordingly. The InfoBox component uses the usePerimeterStatus hook to get the current status of all perimeters. It displays each perimeter's status as a colored status indicator (green for online, red for offline) along with a label (e.g., "Perimeter A: Online/Offline").

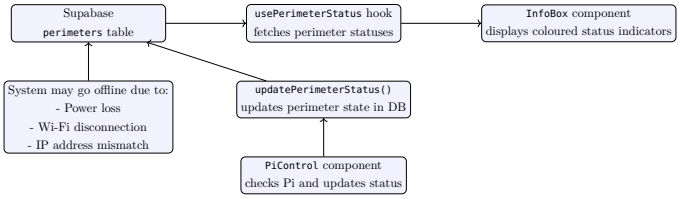


Figure 6.5: Flow of perimeter status tracking and display in the system

Chatbot

A Chatbot component is displayed on the dashboard that provides a chat interface where users can type questions related to penguins, honey badgers, or wildlife safety. It is a fun component to visually improve the UI but also give users a way to interact with the system without needing to leave the UI. When a message is sent, the Chatbot component makes an HTTP request to the OpenAI API. The Chatbot will look for key words in the prompt which have been preset so that users can not ask questions not relating to the project.

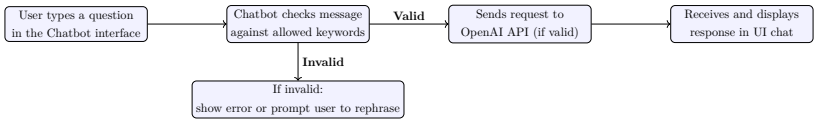


Figure 6.6: Flow of user interaction with the Chatbot component

6.3.4 User Interface Display

The UI consists of two main pages: a login page, where users authenticate themselves before accessing the system and a dashboard page, which displays all the main features such as alerts, status indicators, the chatbot and live camera feeds.

Login Page

The login system is built using shadcn UI components, which allow for clean, accessible and responsive design. The login page, shown to the right in Figure 6.7, includes user registration, input validation and integration with the backend via Supabase.

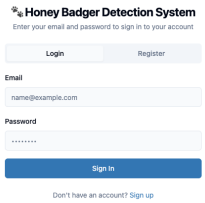


Figure 6.7: Login page

Once on the login page, users have two options: they can either register for a new account or log in if they already have one. When registering, the system creates an account using Supabase authentication and automatically assigns the user a default 'viewer' role. After completing registration, users must verify their email before being allowed to log in.

The overall functionality of the login system follows the flow shown in Figure 6.8.

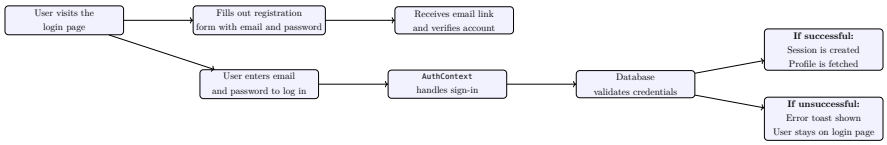


Figure 6.8: Login and registration process flow

Dashboard

Once the user has logged in, they are directed to the dashboard page. There is a banner which includes important information like the name of the the system, the systems time and the user information placed which is placed at the top of the dashboard as seen in Figure 6.9.

Figure 6.9: Banner displaying the name of the system, the time and the current user

Users can log in and log out by clicking on their profile name, as shown in Figure 6.10. The profile status is displayed directly beneath the user's name, indicating their current role and access level within the app. This gives users different privileges based on their assigned role.

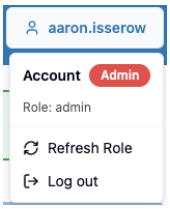


Figure 6.10: Profile status

As admin, this user had full control of all capabilities in the app. At the lowest level, a viewer, is able to access the camera and see live detections. They are however unable to acknowledge alerts or set a simulated detection. Field agents on the other hand have a bit more access such that they can also acknowledge alerts but cannot simulate detections since that was solely used for testing of the system by the admin (myself).

The dashboard is divided into three columns as seen in Figure 6.11 with a alert banner displayed at the top of the dashboard. The dashboard displays a green "All Clear" status indicator, showing that no predators are currently detected. This visual is large and cannot be missed due to to its size and green colour.

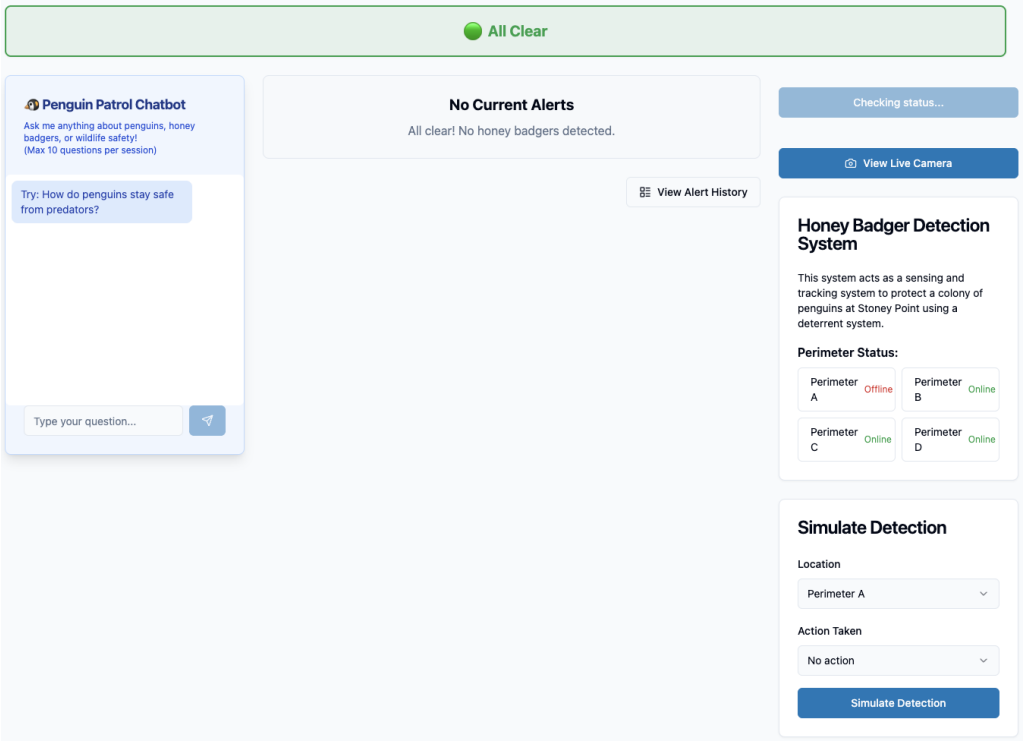


Figure 6.11: Dashboard of the system

In the first column is the Chatbot. This visual is designed to stay the same size with additional chat prompts. This was implemented using a function to wrap the chat in a container with a maximum fixed height. An example prompt is displayed in the Chatbot's display which cycles through a couple examples such as 'Ask me about penguin safety!' and 'Try: How do penguins stay safe from predators?' as seen in Figure 6.11.

The middle column has the alert visual display. It lists the most recent unacknowledged detections if there are any. It also has a 'View Alert History' button that when clicked shows the a visual as seen in Figure 6.12.

In this display, users can view the most recent five alerts and, more importantly, export a CSV file containing the most recent 50 detections. This feature provides users with a log of all recorded alerts, which can reflect detection activity over the course of a day, week, or even a month, depending on the activity around the perimeter.

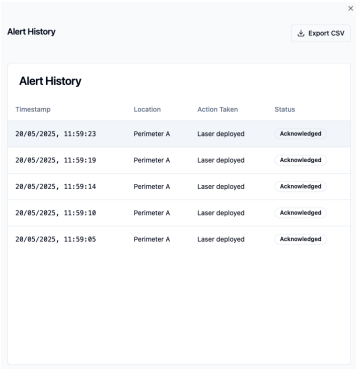


Figure 6.12: History of alerts

The rightmost column (Figure 6.11) contains several key components. At the top is a button used to turn the Camera/Detection system on the Pi. This enables the UI to remotely activate the system, eliminating the need to configure the Pi manually. This is a powerful feature, as it gives the UI complete control over the system's operation.

Below this, the "View Live Camera" button allows the user to access the live video feed from the Pi. Beneath the camera button is the information box and status indicator. While the current system is only connected to Perimeter A, the UI design demonstrates how it can scale to support multiple perimeter zones, making it adaptable to larger protected areas if needed.

The final component in this column is the "Simulate Detection" button, which is used during testing to simulate detection events and verify UI responses.

The status indicator immediately changes from green ("All Clear") to red ('Honey Badger Detected') as seen in Figure 6.13, signaling that a motion has been detected.

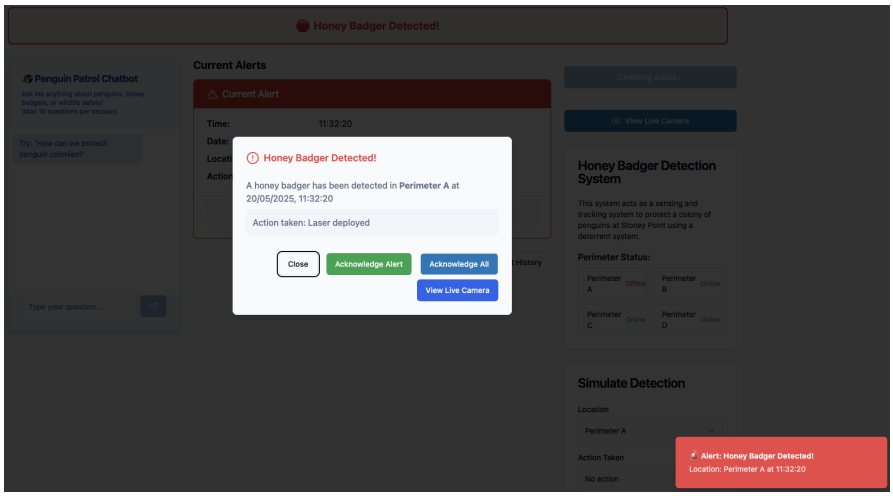
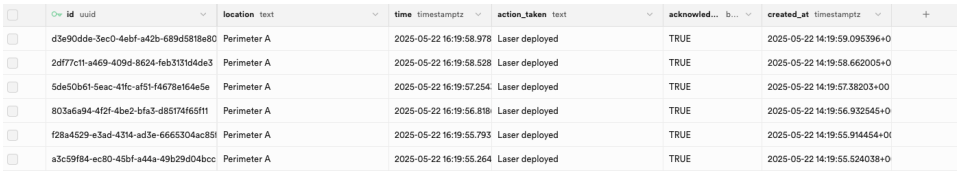


Figure 6.13: Detection alert dashboard

The alert is fetched from the database as seen in Figure 6.14.



id uuid	location text	time timestamptz	action_taken text	acknowled... b...	created_at timestamptz
d3e90dde-3ec0-4ebf-a42b-689d5818e80	Perimeter A	2025-05-22 16:19:58.978	Laser deployed	TRUE	2025-05-22 14:19:59.095396+0
2df77c11-a469-409d-8624-feb3131d4de3	Perimeter A	2025-05-22 16:19:58.528	Laser deployed	TRUE	2025-05-22 14:19:58.662005+0
5de50b61-5eac-41fc-af51-f4678e164e5e	Perimeter A	2025-05-22 16:19:57.254	Laser deployed	TRUE	2025-05-22 14:19:57.38203+00
803a6a94-4f2f-4be2-bfa3-d85174f65f11	Perimeter A	2025-05-22 16:19:56.818	Laser deployed	TRUE	2025-05-22 14:19:56.932545+0
f28a4529-e3ad-4314-ad3e-6665304ac85	Perimeter A	2025-05-22 16:19:55.793	Laser deployed	TRUE	2025-05-22 14:19:55.914454+0
a3c59f84-ec80-45bf-a44a-49b29d04bcc	Perimeter A	2025-05-22 16:19:55.264	Laser deployed	TRUE	2025-05-22 14:19:55.524038+0

Figure 6.14: Detections captured in the Supabase database

The toast notification is displayed at the bottom of the screen and a popup alert is shown with the detection information like the location, time and date and action taken. In this view, there are four options that a user can take. The first is to simply close the alert and return to the dashboard. There are two options to acknowledge the alert. The first is to acknowledge this single alert and the second is to acknowledge all alerts that have not been acknowledged yet. This 'Acknowledge All' button was only added after integration with the sensing and control systems since detections can sometimes occur multiple times if the camera loses and regains visual of a predator in a short time frame, thus alerting the UI twice or even multiple more times of the same event.

6.3.5 Failure Management

Several techniques were implemented to improve the system's reliability and manage failures that could arise during use. These are summarised in Table 6.6.

Table 6.6: Failure management techniques implemented

Name	Description
Simulate Detection Button	A button was added to manually trigger a test detection. This allows the system to be tested in isolation before integrating other subsystems, helping identify early bugs related to detection, database fetches and alert rendering.
Network Monitoring via Inspect	The browser's Developer Tools (Inspect → Network) were used to monitor requests between the UI and Supabase. This helped identify delays and camera stream issues by observing endpoint response times and behaviour.
Camera Loading Bar	A loading bar was introduced to delay the 'View Live Camera' button. This ensures the Pi has fully started before the camera feed is requested. Previously, early clicks caused the component to fail entirely, requiring a system restart. This approach prevents premature access and improves UX.

These techniques provided both development time and user-facing preventive measures. The simulate detection feature allowed debugging in isolation, reducing integration risk. Network analysis tools were useful for tracing performance issues and validating Supabase interactions. Lastly, the loading bar resolved a critical bug where the camera feed failed if triggered too early, ensuring users only interact with the system once it's stable.

6.4 Acceptance Testing

Table 6.7: Acceptance tests for the UI subsystem.

Test ID	Description	Result
AT01	Simulated detection triggers alert	Pass
AT02	Real detection triggers alert	Pass
AT03	Notification deployed	Pass
AT04	Live video feed is visible	Pass
AT05	UI clarity and simplicity	Pass
AT06	System status indicator	Pass

AT01

When the 'Simulate Detection' button was clicked, an entry was correctly logged in the database and the UI displayed a visual alert within 3 seconds. This confirmed that the frotnend and backend components of the system are communicating properly in a simulated context and the system is ready for implementation with the two other systems. As seen in Figure 6.15 the detection log was recorded correctly in the database

within 250ms and was displayed on the UI. The audio alert is also set off after 38ms after the detection was recorded.

detections	201	fetch	@supabase_supabase	0.4 kB	225 ms
profiles?select=id%2Crole&id=eq.835e...	200	fetch	@supabase_supabase	0.6 kB	6.29 s
detections?select=*&acknowledged=eq....	200	fetch	@supabase_supabase	0.9 kB	215 ms
detections?select=*&acknowledged=eq....	200	fetch	@supabase_supabase	0.9 kB	222 ms
profiles?select=id%2Crole&id=eq.835e...	200	fetch	@supabase_supabase	0.6 kB	6.28 s
detections?select=*&acknowledged=eq....	200	fetch	@supabase_supabase	0.7 kB	244 ms
profiles?select=id%2Crole&id=eq.835e...	200	fetch	@supabase_supabase	0.6 kB	6.29 s
detections?select=*&acknowledged=eq....	200	fetch	@supabase_supabase	0.7 kB	234 ms
alarm.mp3	206	media	Other	78.6 kB	38 ms

Figure 6.15: Inspection log of simulated detection

AT02

A real detection was triggered by walking in front of the Raspberry Pi's camera. The UI responded by displaying a visual alert within 3 seconds, meeting the expected response time. This confirmed that the system's full functionality is working properly with respect to the most important feature of alerting users when there is a successful detection. Similarly to the simulated detection scenario, Figure 6.16 shows the polling of the real detection coming thought on the inspection logs which clearly shows that the test was passed.

detections	201	fetch	@supabase_supal	0.1 kB	243 ms
detections?select=*&acknowle...	200	fetch	@supabase_supal	0.5 kB	277 ms
profiles?select=id%2Crole&id...	200	fetch	@supabase_supal	0.1 kB	6.35 s
detections?select=*&acknowle...	200	fetch	@supabase_supal	0.2 kB	417 ms
detections?select=*&acknowle...	200	fetch	@supabase_supal	0.5 kB	322 ms
alarm.mp3	304	media	Other	0.1 kB	10 ms

Figure 6.16: Inspection log of real detection

AT03

Following both simulated and real detections, a visual notification appeared on the dashboard with a sound alert. The notification was triggered within 2 seconds of detection, as required. This confirmed that users are immediately made aware of predator activity through both visual and sound alerts. This test was confirmed in both Figures 6.13 where the alert popup appeared and in 6.15 and 6.16 above where the audio was played.

AT04

After turning on the system and allowing it to connect to the Raspberry Pi, the camera feed was successfully displayed on the UI. In Figure 4.10b in the sensing subsystem, the camera feed was successfully displayed on the UI with a bounding box to track motion. The stream met the required resolution requirements since it used a 960 by 540 resolution. This confirmed that users can reliably monitor the area in real time through the interface in low data remote areas (high latency and slow speeds).

AT05

The interface was presented to a user with no prior experience or explanation. This test was performed on friends and family in which the response was highly positive in how easy the UI was to use. Upon triggering a detection, the user was able to correctly identify that a predator alert had occurred based on the visual cues displayed on the dashboard. This confirmed that the UI is intuitive and easy to understand, even for non-technical users.

AT06

When the Raspberry Pi was taken offline as seen in Figure 6.17, the UI correctly displayed a red status indicator for Perimeter A. Upon reconnection, the indicator changed to green as seen in Figure 6.18, confirming that the UI accurately reflects the system's current state. This functionality ensures that users are always

aware of whether the system is operational or if field agents may need to physically assess the systems for bugs.

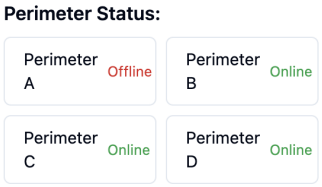


Figure 6.17: The system with a status of offline

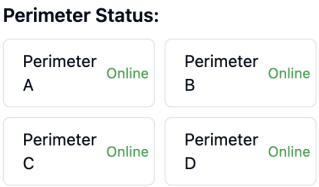


Figure 6.18: The system with a status of online

6.5 Conclusion

The UI subsystem effectively fulfilled its role as the bridge between sensing and tracking and user awareness. It provides conservation staff with a responsive, easy-to-use interface that enables live video access, real-time alerts and system status feedback. Despite challenges such as Pi network instability and timing mismatches across subsystems, the design ensured reliable performance by favouring polling-based updates and routing most communication through Supabase. This architecture allowed for stable integration and flexible development under time constraints, while still delivering a complete alerting solution.

6.5.1 Recommendations

To improve performance and expand functionality, the following enhancements are recommended:

- **Replace polling with real-time database listeners:** While a polling method was simple and reliable, using Supabase's real-time subscription features would reduce latency and improve responsiveness in alert delivery.

- **Enable remote access without local Wi-Fi:** Currently, the Pi and UI must be on the same network. This limits field use. A simple near-term solution would be to use a tunnelling service like ngrok, which allows secure, remote access to the Pi's endpoints. For a longer-term and more scalable approach, a cloud-based proxy server (e.g. hosted on AWS or Google Cloud) could securely mediate between the Pi and the UI, enabling global accessibility without exposing the Pi directly to the internet.

These improvements would significantly enhance the robustness, scalability and usability of the system in real-world conservation environments.

Chapter 7

Conclusions and Recommendations

7.1 Conclusion

This project successfully developed and integrated a modular motion-detection turret system aimed at supporting wildlife conservation efforts. The system consists of three core subsystems—Sensing, Mechanical and Actuation Control, and UI/Frontend—each of which was individually designed, tested, and validated before final integration. The result is a functioning prototype that achieves real-time motion tracking, live video streaming, and remote control capabilities under constrained power, processing, and environmental conditions.

The sensing subsystem delivered reliable motion detection under daylight conditions, providing positional data for targeting and logging events to the database. Despite meeting most specifications, robustness under low-light conditions needs further improvement. The mechanical and actuation control subsystem achieved precise targeting and rapid motor response, with robust mechanical integrity suitable for deployment. The UI subsystem provided a user-friendly interface that allowed for real-time interaction and system monitoring, facilitating seamless operation in the field.

While the project met its core objectives, several limitations were identified. Nighttime detection failure and an improved bounding vox detection are areas where hardware and software enhancements could significantly improve performance. Moreover, power and networking limitations constrained broader deployment potential.

7.2 Recommendations

- **Nighttime Capability:** Integrate an IR LED circuit to leverage the Pi Camera NoIR's native low-light sensitivity and enable robust nighttime detection.
- **Detection Accuracy:** Tune the frame differencing parameters or replace the current method with more advanced techniques such as background subtraction (MOG2), optical flow, or lightweight neural networks like YOLO or MobileNet-SSD.
- **Optical Filtering:** Implement a narrow-band optical filter to mitigate IR noise introduced by the laser in the visual feedback loop.
- **Power Management:** Add isolated voltage rails and hardware-level current monitoring to protect critical components and prevent brown-outs during high load.
- **User Accessibility:** Improve frontend responsiveness by adopting real-time database listeners and enable remote access through secure tunneling or cloud-based proxies.
- **Field Readiness:** Extend testing for weather resistance and structural durability to ensure long-term field deployment viability.

In conclusion, the project demonstrates the feasibility of a low-cost, modular, and networked motion-tracking turret system for conservation applications. The current prototype lays a solid foundation for future improvements, providing a flexible and extensible platform that can be adapted for broader field use with minimal modifications.

Appendix A

Appendix

A.1 Literature Review

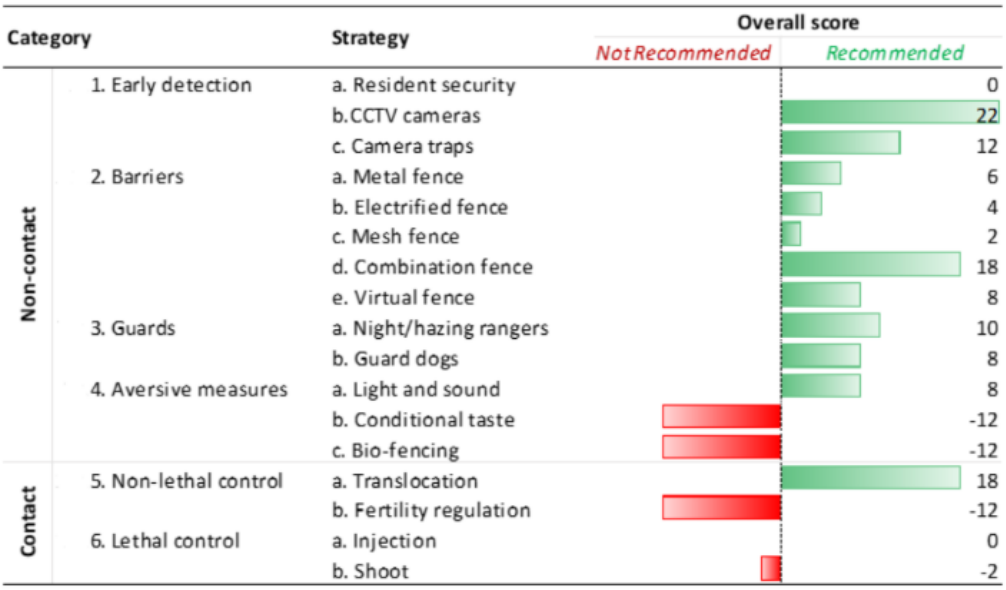


Figure A.1: A summary of overall standardized evaluation scores for primary mitigation strategies [1]

A.2 Mechanical and Actuation-Control Subsystem

A.2.1 Mechanical Drawings of components

Enlarged mechanical drawings of components for readability:

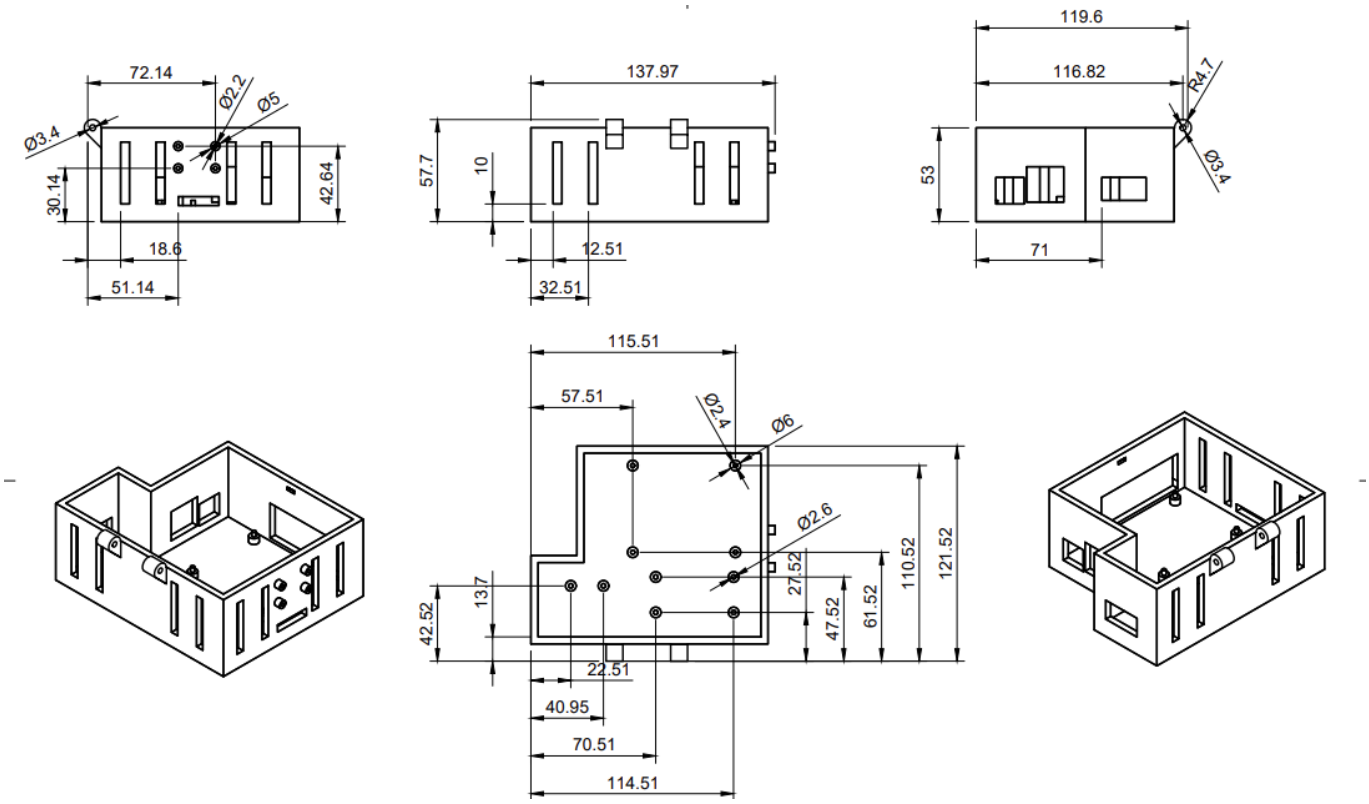


Figure A.2: Mechanical drawing of case design

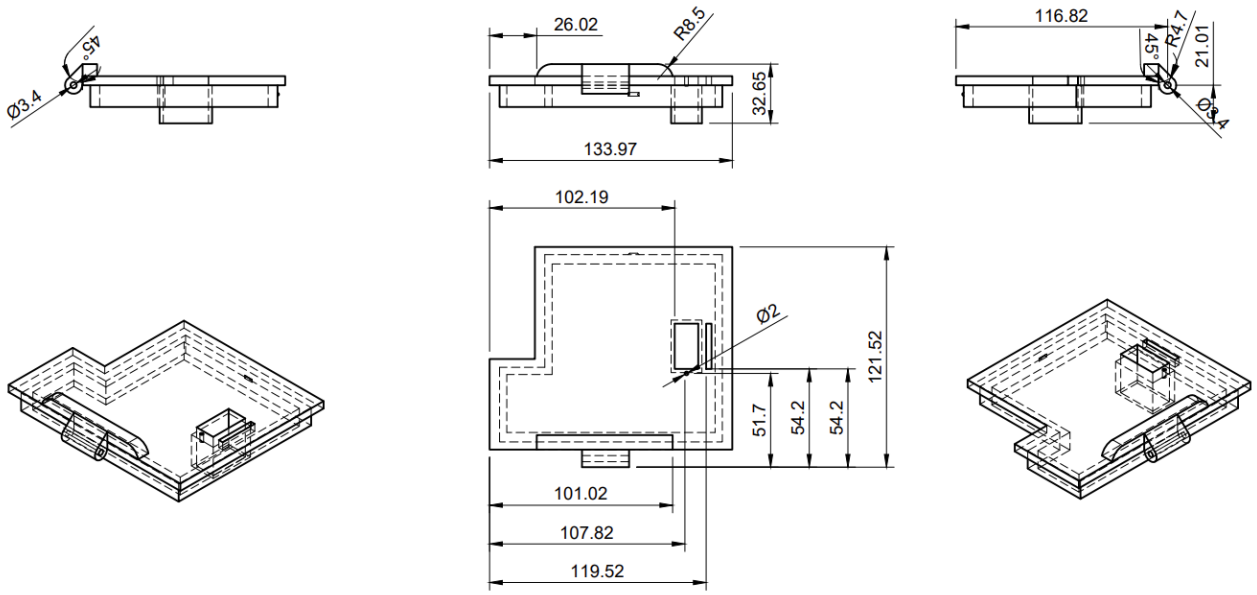


Figure A.3: Mechanical drawing of lid design

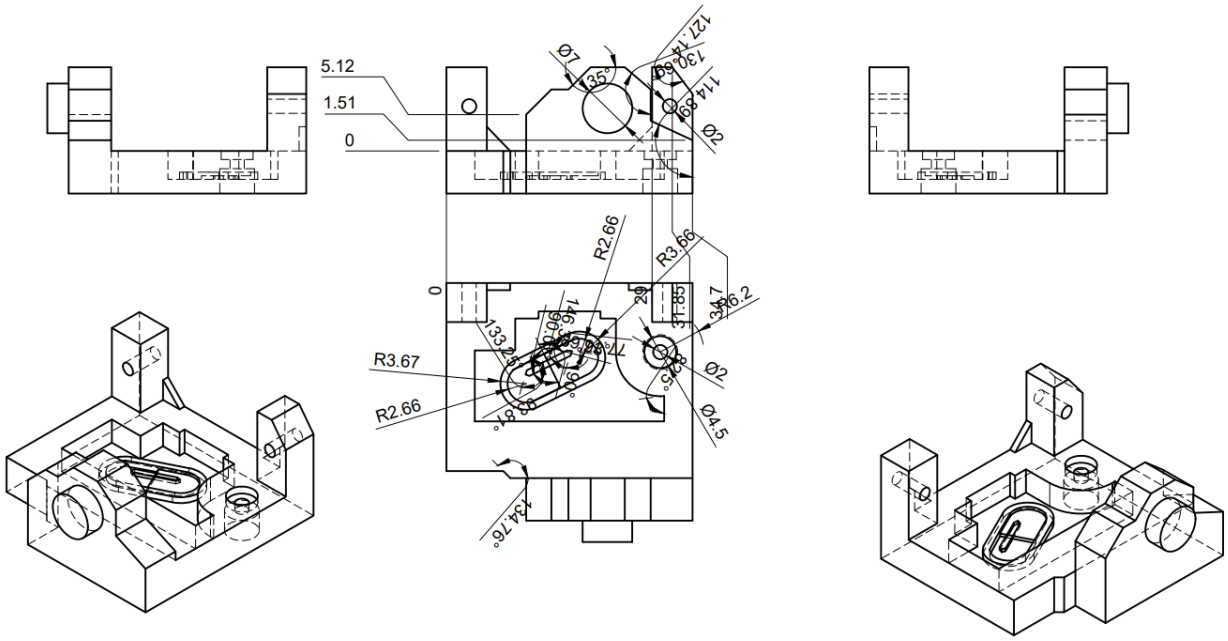


Figure A.4: Mechanical drawing of pan gimbal design

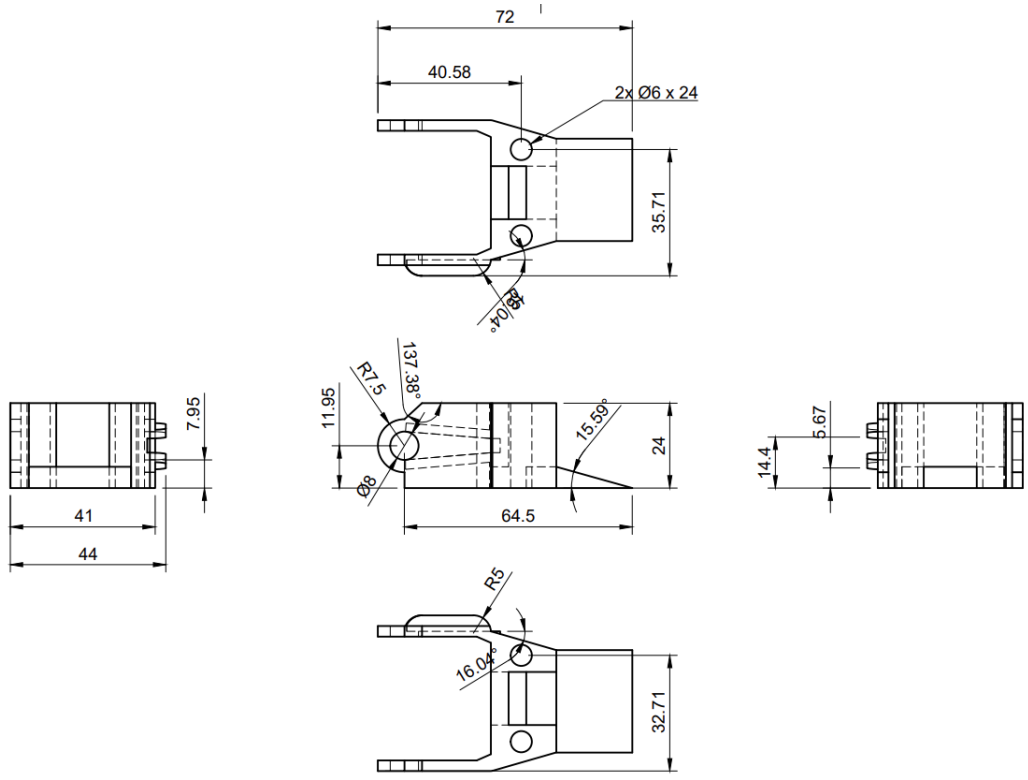


Figure A.5: Mechanical drawing of tilt gimbal design

Appendix B

Bill Of Materials

Table B.1: Bill of Materials (BOM)

Item	**Price (ZAR)**	**Link**
Raspberry Pi 3 Model B plus	758.90	Link
MG90S Micro Servo, Metal Gear x2	150.00	Link
Raspberry Pi 'NoIR' Infrared Camera Board V2	299.90	Link
HKV HS-TFC1-16GB+ADPT	56.01	Link
HKD LASER TRANSMITTER MOD KY-008	13.00	Link
5V 1 Channel Level Trigger Optocoupler Relay Module	25.76	Link
DC Buck Step-Down Power Module, 3A/24V, MP2315	60.35	Link
TOTAL	**1363.92**	

Appendix C

Evidence of GAs Met

Table C.1

Table C.1: Graduate Attributes (GAs) Evidence Table

Subsystem Title	Student Number	Where Met
UI, UX, and Frontend Implementation	ISSAAR001	GA 3: Section 6.3 (Design), pages 34–40 GA 7: Section 6.1 (Introduction), page 32; D-School, page 2 GA 8: Section 6, pages 32–42, collaboration with sensing and control subsystems; Work on introduction and literature review, pages 1 and 3-9 GA 10: Final report submission and report documentation and implementation detail (Sections 6.1–6.5, pages 32–42)
Mechanical Design and Actuation Control	FRDETH004	GA 3: Section 5.3.1 (Design Decisions), pages 23-24. This compares actuator selection, enclosure material trade-offs and justifies final choices with technical criteria. Section 5.3.2 (Final Configuration), pages 24-27 provides detailed CAD drawings and descriptions of the model. GA 7: Section 5.2.1 (Requirements), page 23 certain requirements detail the model design and it's need to be of low impact to the environment. Section 5.3.1 (Design Decisions) page 23 notes that PLA prototypes are recyclable and biodegradable. GA 8: Section 5.3.1 & 5.3.4 pages 23& 28. Collaboration between the sensing and UI subsystems via mechanical-electronics interfaces showing independent work and cross-team integration. GA 10: Section 5.2.4 (Traceability Analysis) page 23. Mapping each requirement to specification to acceptance test shows clear documentation and design process work. Section 5.3.5 (Acceptance Testing) pages 29-31. Results of acceptance test procedures shows validation and reporting of both successful and failed tests.
Sensing and Motion Detection	VCHEMA001	GA 3: Section 4.3 (Design Decisions), pages 11-16: compares motion detection approaches, processing platforms, cameras and camera based detection algorithms, all choices were made through thorough research and justifiable reason. Section 4.4 (Subsystem Architecture and Integration), pages 16-19: provides detailed explanations to how the subsystem was designed and how the overall architecture works, together with how the subsystem integrates with the rest of the system. GA 7: Section 4.2 (Requirements), page 10: requirements considered energy efficiency. Section 4.3.1 (Sensor Selection), pages 11-12: design choice does not involve invasive sensing such as no audio and thermal disruption. Section 4.6.1 (Subsystem Recommendations), page 21: recommended use of IR LEDs to provide vision during nighttime rather than employing bright lights. GA 8: Section 4.4.3 & 4.4.4 (Subsystem Integration), pages 18-19: successful integration with the other two subsystems demonstrates effective teamwork and interactions across both mechanical and software disciplines. GA 10: Section 4.2 (Requirements and Specifications Analysis), pages 10-11: clear requirements/specifications were defined. Section 4.5 (Acceptance Testing), pages 20-21: thorough testing was performed and the results were validated and reported professionally, failures were transparently reported. Section 4.6 (Subsystem Conclusion), page 21: conclusion provides an honest and accountable evaluation of the system performance.

This section summarizes how the work presented in this report has fully addressed the specified Graduate Attributes (GAs) through the design, analysis, and teamwork activities.

B.1 Evidence of GAs Met – Emanuele Vichi (VCHEMA001)

GA 3: Engineering Design

I designed the sensing subsystem, which was made up of choosing a motion detection approach, a processing platform and designing the algorithm. The requirements and specifications were defined at the beginning such that they met the expectations we set for the system based on the problem statement. Design decisions were taken to choose components and approaches that would maximise the requirements met. The next step was to then provide the full architecture of the system and how it integrated with the rest of the subsystems. The final step was to test and evaluate the performance of the designed subsystem based on the previously defined requirements.

GA 7: Sustainability and Impact of Engineering Activity

The design considered power efficiency by using a Raspberry Pi 3B+ and a NoIR camera, operated at 12V 1A. Environmental impact was addressed by selecting non-invasive detection methods and proposing IR lighting for low-light scenarios to preserve nocturnal animal behaviour and ecosystem balance. The subsystem is small and does not make use of thermal or audio sensors to detect motion, making it non-invasive.

GA 8: Individual, Team and Multidisciplinary Working

As the sole developer of the sensing subsystem, I was responsible for its architecture, implementation, and integration. Effective collaboration with teammates was necessary for aligning the sensing outputs with actuation and frontend requirements. Interfacing with mechanical and software disciplines showed successful multidisciplinary engagement. Regular meetups were arranged between all teammates to integrate the system successfully.

GA 10: Engineering Professionalism

All deliverables were handed in on time, and all classroom activities were attended, engaged regularly with instructors to receive feedback and communicated through appropriate messaging channels. Throughout the project, professional standards were upheld in documentation, version control, and communication. All results were rigorously tested and honestly reported, including failures and limitations.

B.1 Evidence of GAs Met – Ethan Faraday (FRDETH004)

GA 3: Engineering Design

I designed the mechanical and actuation control subsystem. This involved defining the user requirements, translating them into technical specifications and developing them into ATPs to test the range, accuracy and structural integrity under load. I designed all CAD models, and developed control simulations which guided my design iterations. The design process of: identifying the problem and it's requirements; preliminary design; detailed design; prototyping; as well as a detailed analysis was performed.

GA 7: Sustainability and Impact of Engineering Activity

The proposed design involves no direct harm and interference directed towards wildlife, since the design is operated in a remote context. Additionally, the small non-invasive footprint of the enclosure I designed means that there is little environmental impact. The machining process for the material used can be harmful, for this reason, prototypes were made using PLA filament, which is recyclable and industrially biodegradable. The predator deterrent selected poses no excessive harm to to the wildlife or their habitat.

GA 8: Individual, Team and Multidisciplinary Working

I worked independently on the mechanical and actuation control subsystem while collaborating with team members to meet full system requirements. Regular design updates, shared models, and clear communication ensured that each member's design requirements and constraints were understood and respected by the others. This involved the integration and interfacing through the mechanical design to ensure all electronic components were safely stored, as well as designing the interfacing between the servo motor control and the object detection algorithms. This also involved working as a team to troubleshoot errors that occurred while interfacing between subsystems. Additionally, This subsystem spanned multiple disciplines, including mechanical design—creating CAD models in AutoCAD, conducting stress analyses, validating weatherproofing, and selecting appropriate materials—as well as electrical and software engineering, covering actuator control algorithms, wiring design and routing, and embedded software development.

GA 10: Engineering Professionalism

I met every project milestone for my subsystem deliverables, attended all team and classroom sessions (except for a few with an excused absence), responded promptly to instructor feedback and to team member requests. I managed my portion of the version-controlled repository, and ensured that my sections were submitted on time and fully reviewed. This consistent professional conduct resulted in the successful completion of the mechanical and actuation-control subsystem.

B.1 Evidence of GAs Met – Aaron Isserow (ISSAAR001)

GA 3: Engineering Design
I designed and implemented the user interface and alert system. This involved building a frontend that displays predator detections, camera feeds, and status indicators in a way that is reliable, responsive, and easy to interpret.

GA 7: Sustainability and Impact of Engineering Activity
The UI contributes to wildlife conservation by enabling faster human response to predator threats, helping improve the survival chances of endangered penguins.

GA 8: Individual, Team and Multidisciplinary Working
I worked independently on the UI subsystem while collaborating with teammates to integrate the interface with the sensing and control components.

GA 10: Engineering Professionalism
I followed best practices in code structure, testing, and documentation, and made design decisions based on real-world constraints like network delays and Pi limitations.

Bibliography

[1] L. K. Rhoda, "The threat of terrestrial predators to mainland penguin colonies: searching for sustainable solutions," 2022.

[2] R. B. Sherley, R. J. M. Crawford, A. D. de Blocq, B. M. Dyer, D. Geldenhuys, C. Hagen, J. Kemper, A. B. Makhado, L. Pichegru, D. Tom, L. Upfold, J. Visagie, L. J. Waller, and H. Winker, "The conservation status and population decline of the african penguin deconstructed in space and time," *Ecology and evolution.*, vol. 10, no. 15, 2020-08.

[3] Y. Ropert-Coudert, A. Chiaradia, D. Ainley, A. Barbosa, P. D. Boersma, R. Brasso, M. Dewar, U. Ellenberg, P. García-Borboroglu, L. Emmerson *et al.*, "Happy feet in a hostile world? the future of penguins depends on proactive management of current and expected threats," *Frontiers in Marine Science*, vol. 6, p. 248, 2019.

[4] N. Nattrass, M. Drouilly, and M. J. O'riain, "Learning from science and history about black-backed jackals canis mesomelas and their conflict with sheep farmers in south africa," *Mammal Review*, vol. 50, no. 1, pp. 101–111, 2020.

[5] A. A. Ordiz Fernandez, R. Bischof, and J. Swenson, "Saving large carnivores, but losing the apex predator?" 2013.

[6] F. Courchamp, M. Langlais, and G. Sugihara, "Cats protecting birds: modelling the mesopredator release effect," *Journal of Animal Ecology*, vol. 68, no. 2, pp. 282–292, 1999.

[7] J. S. Lewis, S. Spaulding, H. Swanson, W. Keeley, A. R. Gramza, S. VandeWoude, and K. R. Crooks, "Human activity influences wildlife populations and activity patterns: implications for spatial and temporal refuges," *Ecosphere*, vol. 12, no. 5, p. e03487, 2021. [Online]. Available: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1002/ecs2.3487

[8] J. F. Moore, F. Mulindahabi, M. K. Masozera, J. D. Nichols, J. E. Hines, E. Turikunkiko, and M. K. Oli, "Are ranger patrols effective in reducing poaching-related threats within protected areas?" *Journal of Applied Ecology*, vol. 55, no. 1, pp. 99–107, 2018.

[9] C. A. Mackenzie and P. Ahabyona, "Elephants in the garden: Financial and social costs of crop raiding," *Ecological Economics*, vol. 75, pp. 72–82, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921800911005349

[10] M. Becker, D. J. Huggard, M. Dickie, C. Warbington, J. Schieck, E. Herdman, R. Serrouya, and S. Boutin, "Applying and testing a novel method to estimate animal density from motion-triggered cameras," *Ecosphere*, vol. 13, no. 4, p. e4005. [Online]. Available: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1002/ecs2.4005

[11] R. Kays, S. Tilak, B. Kranstauber, P. A. Jansen, C. Carbone, M. J. Rowcliffe, T. Fountain, J. Eggert, and Z. He, "Monitoring wild animal communities with arrays of motion sensitive camera traps," *arXiv preprint arXiv:1009.5718*, 2010.

[12] W. A. Cox, F. Thompson III, and J. Reidy, "The effects of temperature on nest predation by mammals, birds, and snakes," *The Auk*, vol. 130, no. 4, pp. 784–790, 2013.

[13] S. Hamel, S. T. Killengreen, J.-A. Henden, N. E. Eide, L. Roed-Eriksen, R. A. Ims, and N. G. Yoccoz, "Towards good practice guidance in using camera-traps in ecology: influence of sampling design on validity of ecological inferences," *Methods in Ecology and Evolution*, vol. 4, no. 2, pp. 105–113, 2013.

[14] 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, p. 05014, 2024.

[15] A. J. Simla, R. Chakravarthi, and L. M. Leo, "Agricultural intrusion detection (aid) based on the internet of things and deep learning with the enhanced lightweight m2m protocol," *Soft Computing*, pp. 1–12, 2023.

[16] K. Balakrishna, F. Mohammed, C. Ullas, C. Hema, and S. Sonakshi, "Application of iot and machine learning in crop protection against animal intrusion," *Global Transitions Proceedings*, vol. 2, pp. 169–174, 2021.

[17] D. Adami, M. O. Ojo, and S. Giordano, "Design, development and evaluation of an intelligent animal repelling system for crop protection based on embedded edge-ai," *IEEE Access*, vol. 9, pp. 132 125–132 139, 2021.

[18] A. J. King, S. J. Portugal, D. Strömbom, R. P. Mann, J. A. Carrillo, D. Kalise, G. de Croon, H. Barnett, P. Scerri, R. Groß *et al.*, "Biologically inspired herding of animal groups by robots," *Methods in Ecology and Evolution*, vol. 14, no. 2, pp. 478–486, 2023.

[19] S. W. Breck, J. T. Schultz, D. Prause, C. Krebs, A. J. Giordano, and B. Boots, "Integrating robotics into wildlife conservation: testing improvements to predator deterrents through movement," *PeerJ*, vol. 11, p. e15491, 2023.

[20] S. Pitla, S. Bajwa, S. Bhusal, T. Brumm, T. M. Brown-Brandl, D. R. Buckmaster, I. Condotta, J. Fulton, T. J. Janzen, M. Karkee *et al.*, "Ground and aerial robots for agricultural production: Opportunities and challenges," *Biological Systems Engineering: Papers and Publications*, vol. 727, 2020.

[21] M. A. Weston, C. O'Brien, K. N. Kostoglou, and M. R. Symonds, "Escape responses of terrestrial and aquatic birds to drones: Towards a code of practice to minimize disturbance," *Journal of Applied Ecology*, vol. 57, no. 4, pp. 777–785, 2020.

[22] N. M. Schroeder, A. Panebianco, R. Gonzalez Musso, and P. Carmanchahi, "An experimental approach to evaluate the potential of drones in terrestrial mammal research: a gregarious ungulate as a study model," *Royal Society Open Science*, vol. 7, no. 1, p. 191482, 2020.

[23] J. A. Shivik, "Non-lethal alternatives for predation management," *Sheep Goat Research Journal*, vol. 19, pp. 64–72, 2004.

[24] R. J. Robel, A. D. Dayton, F. R. Henderson, R. L. Meduna, and C. W. Spaeth, "Relationships between husbandry methods and sheep losses to canine predators," *The Journal of Wildlife Management*, pp. 894–911, 1981.

[25] A. O'Brien and J. Craig, "Fencing options for predator control," Ontario Ministry of Agriculture, Food and Rural Affairs (OMAFRA), June 2020. [Online]. Available: https://www.ontariosheep.org/media/soqfaeww/fencing-options-for-predator-control_omafra.pdf

[26] R. Hering, M. Hauptfleisch, S. Kramer-Schadt, J. Stiegler, and N. Blaum, "Effects of fences and fence gaps on the movement behavior of three southern african antelope species," *Frontiers in Conservation Science*, vol. 3, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fcosc.2022.959423

[27] D. A. Wade, "The use of fences for predator damage control," 1982. [Online]. Available: https://escholarship.org/uc/item/0jd0m3zg

[28] O. Antczak-Orlewska, D. Okupny, A. Kruk, R. I. Bailey, M. Płóciennik, J. Sikora, M. Krąpiec, and P. Kittel, "The spatial and temporal reconstruction of a medieval moat ecosystem," *Scientific Reports*, vol. 12, no. 1, p. 20679, 2022.

[29] G. V. Kollias and J. Fernandez-Moran, "Mustelidae," *Fowler's Zoo and Wild Animal Medicine, Volume 8*, p. 476, 2014.

[30] O. Ohrens, C. Bonacic, and A. Treves, "Non-lethal defense of livestock against predators: flashing lights deter puma attacks in chile," *Frontiers in Ecology and the Environment*, vol. 17, no. 1, pp. 32–36, 2019.

[31] E. Laguna, P. Palencia, A. J. Carpio, J. Mateos-Aparicio, C. Herraiz, C. Notario, J. Vicente, V. Montoro, and P. Acevedo, "Evaluation of a combined and portable light-ultrasound device with which to deter red deer," *European Journal of Wildlife Research*, vol. 68, no. 1, pp. 1–11, 2022.

[32] H. K. Ober and A. Kane, "How to use deterrents to stop damage caused by nuisance wildlife in your yard," 2024. [Online]. Available: https://edis.ifas.ufl.edu/publication/UW371

[33] T. A. Biedenweg, M. H. Parsons, P. A. Fleming, and D. T. Blumstein, "Sounds scary? lack of habituation following the presentation of novel sounds," *PLoS One*, vol. 6, 2011. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC3022648/

[34] S. Lukas, L. Clark, A. Davis, D. Sanchez, and L. Brewer, "Nonlethal bird deterrent strategies," 2020.

[35] M. Bomford and P. H. O'Brien, "Sonic deterrents in animal damage control: A review of device tests and effectiveness," *Wildlife Society Bulletin (1973-2006)*, pp. 411–422, 1990.

[36] J. L. Belant, T. W. Seamans, and L. A. Tyson, "Evaluation of electronic frightening devices as white-tailed deer deterrents," *Proceedings of the Vertebrate Pest Conference*, vol. 18, pp. 151–155, 1998.

[37] S. E. Baker, S. A. Ellwood, R. Watkins, and D. W. Macdonald, "Non-lethal control of wildlife: using chemical repellents as feeding deterrents for the european badger meles meles," *Journal of Applied Ecology*, vol. 42, no. 5, pp. 921–931, 2005.

[38] P. M. Garvey, A. S. Glen, and R. P. Pech, "Dominant predator odour triggers caution and eavesdropping behaviour in a mammalian mesopredator," *Behavioral Ecology and Sociobiology*, vol. 70, pp. 481–492, 2016.

[39] J. A. Shivik, A. Treves, and P. Callahan, "Nonlethal techniques for managing predation: Primary and secondary repellents," *Conservation Biology*, vol. 17, no. 6, pp. 1531–1537, 2003.

[40] S. E. Goodyear, "Habituation to auditory stimuli by captive african elephants (loxodonta africana)," 2015.

[41] M. Cook-Hines, "The fallacy of lethal predator-control," 2024. [Online]. Available: https://edspace.american.edu/atrium/portfolio-item/cook-hines-madison-the-fallacy-of-lethal-predator-control/

[42] J. Tukai and ENCOSH, "Predator deterrent light systems," 2021. [Online]. Available: https://encosh.org/simplified_initiatives/predator-deterrent-light-systems/

[43] "Banning predator control on national wildlife refuges," Humane Society Veterinary Medical Association (HSVMA). [Online]. Available: https://www.hsvma.org/predatorcontrol_nationalwildliferefuges

[44] A. Vigren. (2024, Apr.) Multiple methods for motion detection. Accessed: 2025-05-24. [Online]. Available: https://newsroom.axis.com/blog/motion-detection

[45] Avigilon. (2024, April) Infrared vs thermal cameras: Key differences & considerations. Accessed: 2025-05-24. [Online]. Available: https://www.avigilon.com/blog/infrared-vs-thermal-cameras

[46] I. Berrios. (2023, Oct.) Introduction to motion detection: Part 1 — frame differencing. Accessed: 2025-05-24. [Online]. Available: https://medium.com/@itberrios6/introduction-to-motion-detection-part-1-e031b0bb9bb2

[47] HeyCoach. (2025, January) Motion detection algorithms in c++. Accessed: 2025-05-24. [Online]. Available: https://blog.heycoach.in/motion-detection-algorithms-in-c/