DuckWild Midpoint Report

December 10, 2025

Andrea Schuman

CPE 350

Team Name: DuckWild
Sponsored by: OWL Integrations

Michael Albert

Ben Garvin

Camille Leute

Luke Trusheim

1. Introduction

1.1 Project Overview

The context of this project is that OWL Integrations seeks to grow out of an R&D phase to make revenue from the OWL IoT board they will be releasing soon. To market the usage of mesh networks of radios, and specifically to market the ClusterDuck protocol, OWL seeks to have a wide variety of interesting projects listed on their website using many different development platforms.

Our team is designing a smart sensor using a Raspberry Pi board, IR camera, and motion sensor that can function fully off grid without internet or electricity access. The specific application of the sensor will be to track population trends and population density of many species of animals in a specific ecosystem. This sensor must function as the head node of a mesh network of ClusterDuck nodes, routing to a PapaDuck cloud portal. To do this, the sensor must be able to send and receive ClusterDuck protocol compatible packets over a LoRa standard radio using a SX1262 Waveshare board. Packets will contain the location and species ID of animals, as detected by a computer vision model running on the Raspberry Pi. Our sensor must be compatible with both existing DuckLink nodes created by OWL and DuckLink nodes developed on the Raspberry Pi by other capstone groups this year. The objective of this project is to demonstrate the unique capabilities and ease of use of the ClusterDuck protocol to Raspberry Pi developers. This will expand usage of the ClusterDuck protocol, corresponding to more sales of OWL Integrations' DuckLink and DuckLink Development Board products.

1.2 Client

DuckWild's sole client is OWL Integrations, a company that focuses on bringing connected systems into hard to reach environments. After the hurricanes in 2017 and 2018, IBM started the Call for Code to create disaster mitigation solutions. The idea that OWL came up with was to deploy clusters of "duck radios" to areas affected by disasters. These radios connect people in disaster-prone areas with emergency services so that they can request help. For this innovation, OWL won the Call for Code's Global Grand Prize, giving them the funding they needed to make their idea a reality. Now, their goal is to provide their customers with cost-effective network solutions for hard-to-reach geographical locations that lack stable communications infrastructure. The part we play with OWL is designing a head node using a Raspberry Pi 5 along with an IR camera and PID motion detector to create a smart sensor. This can relay information across a mesh using OWL's ClusterDuck Protocol (CDP) using the LoRa waveshare board. This eliminates the need for cellular connectivity in remote areas so that the state of the surrounding area can be communicated without needing a person to physically investigate.

1.3 Stakeholders

Stakeholder	Level of Support			Explanation of Level of Support		Influence Strategy Tactics To achieve or maintain needed level of support	
	Against	Neutral	Supportive	Positive or Supporting Items	Issues and Concerns	Strategy	Implementation (timeline, who)
OWL Integrations			✅	- Using their protocol to achieve something great - If our project is successful, it would promote their company and help them make money	- OWL Integrations promotes our project but it doesn't gain traction or provide guidance to potential adopters	- Creating a product that promotes their business - An implementation of the ClusterDuck protocol that acts as model and guide for potential adopters	- Regularly schedule meetings to discuss the project and ensure that we're meeting expectations.
Dr. Derickson			✅	- Our project's success may strengthen his relationship with OWL	- Our project's failure will harm his relationship with OWL	- Staying on schedule with the project - Creating a good product with his support - Asking questions	- Scheduling meetings with him when we need his support
Raspberry Pi Community		✅		- The Raspberry Pi community appreciates open source tools and example projects	- They have no reason (yet) to use ClusterDuck protocol for the Pi Projects	Engage with community via showcasing project and contributing code	Showcase project on GitHub and related community platforms (Hackaday)?
Meshtastic	✅			- Generally promoting mesh network technology could help their business too	- A direct competitor to OWL	- Watch them closely and analyze their movements in the network community	- Analyze applications to see if we can compete in any of their areas

Table 1: Stakeholder report

1.4 Framed Insights & Opportunities

Our client, OWL Integrations, needs to increase overall engagement with the ClusterDuck protocol in order to make their ClusterDuck-based solutions and development tools more attractive and reach a larger audience. To fulfill this need, OWL needs to establish a backlog of projects developed using various hardware devices so that engineers can learn about the advantages that ClusterDuck offers and the feasibility of implementing it into their projects and products. One such community is based around the Raspberry Pi single-board computing platform. The Raspberry Pi platform represents a particularly valuable opportunity for ClusterDuck integration, as it can be used in applications that require more computing power (such as AI-powered systems). The Raspberry Pi community includes a mix of hobbyist developers and professionals using the Raspberry Pi platform in their own products. While not an explicit "need," the Raspberry Pi community values open source development of both hardware and software (the Raspberry Pi SBCs primarily support the open-source Linux operating system).

Statement of Need

The wildlife monitoring application we have chosen could have real world benefits, and there is a large existing community of open source projects for conservation and animal rights work. Currently, remote wildlife monitoring requires not so remote means. Trail and wildlife cameras often need battery replacement, cellular connection, and have a lack of filtering or animal identification. With our smart sensor using solar power, CDP, and AI filtering we are able to tackle these issues. This should be an appealing application of the ClusterDuck protocol for marketing purposes, and may have real life benefits.

1.5 Project Goals & Objectives

The primary motivation behind our project is to demonstrate to the Raspberry Pi community that utilizing the ClusterDuck protocol in Raspberry Pi-based hardware projects is both easy and valuable. This motivation leads us to focus on several goals in particular:

1. Demonstrate the value of the ClusterDuck protocol.
2. Provide an example of how to use the ClusterDuck protocol with a Raspberry Pi.
3. Create a project that would generate sufficient community “buzz” to function as an advertisement for the ClusterDuck protocol and OWL Integrations.

To accomplish these goals, we decided on the wildlife population tracking application. This application of the ClusterDuck protocol…

1. Demonstrates the value of the ClusterDuck protocol and LoRa radio in off grid / remote environments where data transmission using cellular, WiFi, or satellite are either too expensive or not feasible.
2. Utilizes the processing power of the Raspberry Pi for a large computer vision model that is capable of identifying many different species of animals.
3. Appeals to both a general audience and the Raspberry Pi community of open source developers.

1.6 Project Outcomes & Deliverables

We aim to create a functioning Raspberry Pi smart sensor that can send data via the ClusterDuck Protocol. We hope to inspire the open source communities surrounding the Raspberry Pi SBC to use duck radios in their personal and professional projects by recording a video demonstrating the capabilities of our product, how to use it, and how to set it up. In order to provide an example for the Raspberry Pi community on how to use the ClusterDuck protocol with Raspberry Pi boards, we will open source the repositories containing the code for our specific application as well as the general integration of the ClusterDuck protocol.

Scheduling & Project Duration

- ☐ October 20: Specific use case determined & meet OWL CEO
- ☐ November 10: Set up Raspberry Pi with Linux, install AI libraries, and test sensor hardware.
- ☐ December 10: Functional prototype to demonstrate sending ClusterDuck compatible packets, accurate species detection, and power consumption.
- ☐ February 16: Testing of send and receive capabilities between Raspberry Pi head node and existing ClusterDuck nodes.
- ☐ February 24: Scaled power consumption for low power solar panel and small battery compatibility, demonstrated low and high power operating modes.
- ☐ March 1: Record demonstration video showcasing sensor functionality and ClusterDuck integration.

- [ ] March 13: End of project, complete integration between sensor and ClusterDuck and the product is tested in the real world.

1.7 Team Mission and Objectives

Our mission is to operate as a dependable, respectful, and responsive team, maintaining open communication and consistent progress throughout the quarter. Our objectives are to:

1. Showcase the value of the ClusterDuck Protocol through the smart sensor application using a Raspberry Pi, IR camera, motion detector, and LoRa radio connection.
2. Demonstrate successful communication between sensor nodes, DuckLink nodes, and a PapaDuck cloud gateway.
3. Deliver a polished and well-documented prototype that highlights the potential of the ClusterDuck ecosystem to both OWL Integrations and the broader community.

1.8 Team Membership and Roles

Camille Leute – *Project Manager*: Oversees project scheduling, ensures milestones are met, coordinates meetings, and maintains overall progress tracking.

Michael Albert – *Liaison*: Primary contact with OWL Integrations and Professor Derickson; responsible for communicating questions, updates, and feedback between the team and sponsors.

Ben Garvin – *Documentation and Reporting*: Compiles technical documentation, maintains the design log, and prepares interim and final project reports.

Luke Trusheim – *Financial Officer and Procurement*: Manages budgeting and ordering of materials such as the Raspberry Pi, sensors, IR camera, and spare parts.

1.9 Planning Information

There were two high risk aspects of this project solution. The first was (and still is) getting our Pi to be fully compatible with other duck nodes. This means our Pi must be able to receive and transmit messages that are compatible with the ClusterDuck protocol over LoRa radio signals. It is taking some time for our group to scan through the ClusterDuck Github repository and adapt signal sending over LoRa to send compatible packets. The second high risk aspect of our project was using an existing computer vision library to detect different kinds of animals. Fortunately, this aspect of the project has been mostly completed.

We minimized these risks by allocating a significant amount of our project time to working on each of these aspects of the design, and we divided into two groups to tackle each of these challenges. By dividing the project into two subsystems (one handling ClusterDuck communication and the other handling computer vision), we were able to make progress on multiple pieces of this project in parallel. This enabled us to produce separate prototypes for each subsystem by the end of the quarter, paving the way to integrate the subsystems and test a full prototype in the winter.

The first step on the ClusterDuck subsystem was to understand the state that the previous capstone team left this project in. Ultimately, we decided that the previous team's codebase was unmaintainable, as it made breaking changes to the upstream codebase. As such, we chose to focus on reimplementing ClusterDuck in Python to make it easier for us to develop and more approachable for the Raspberry Pi community. This implementation has reached the point where it can send and receive ClusterDuck formatted packets, so the next steps will be to test interoperability with existing ClusterDuck devices such as the QuackerBoard.

In order to parallelize work on this project, the AI subsystem development effort did not depend on the ClusterDuck subsystem in any way. Instead, we began our work by evaluating the currently available pre-trained animal recognition models to determine the best one for our use case. This included making sure that the models could properly run on a Raspberry Pi and can distinguish between a reasonable number of animal species, as well as running benchmarks on our own datasets. After we determined which model to use (SpeciesNet), we began implementing the animal detection system on the Raspberry Pi. This involved setting up a motion detection system that feeds into the animal recognition model. Our prototype showed this system running in real-time and coming up with predictions.

In the winter, our focus will shift to combining the subsystems together. After we have a complete prototype, we will begin looking for opportunities to test our prototype in the field. We will also look into ways to make our prototype more robust, such as by making it weather-proof and optimizing power usage. Once we have a sufficiently polished version of our product, we will begin marketing it to the Raspberry Pi community. This will include (at minimum) writing an article about the project and publishing our code on GitHub.

---

2. Background

2.1 Existing Solutions & Competitor Analysis

ClusterDuck exists alongside several other mesh-networking systems, each with different strengths. Meshtastic is the closest match (an open-source LoRa mesh) but focuses on basic messaging rather than structured sensor data. Zigbee, Thread, and Bluetooth Mesh work well indoors with many nearby devices, but they don't support the long-range, low-infrastructure needs of disaster or remote deployments. DigiMesh offers reliable routing but is proprietary and requires specialized hardware. ClusterDuck stands out because it combines long-range LoRa links, easy deployment, and an open-source design built for sensor networks in hard-to-reach environments, supporting the goals of our Raspberry Pi project.

2.2 Relevant Standards, Regulations, and Validation Procedures

Our system must follow the FCC Part 15 rules for unlicensed LoRa communication, which limits transmit power and ensures the radio operates legally and without causing interference. Software relies on basic standards like version control with Git, clear packet formatting, and simple testing for data consistency between nodes and gateways. To validate performance, we will run tests on measuring range, end-to-end packet delivery, and stressing the system to confirm that the mesh works reliably in real environments.

2.3 Technical Background

Our project builds on several existing technologies and ideas. Wildlife cameras already exist in many forms, but most consumer models only save images locally, requiring operators to physically retrieve the SD card. Some higher-end units use cellular connections for remote viewing, though these require paid data plans and reliable cell coverage. More advanced systems, such as BuckeyeCam and CuddeLink, use mesh networking so cameras can relay images to a central base station, but these products are costly and closed-source. To create a more accessible and flexible alternative, our system uses LoRa, a long-range, low-power radio technology suited for sending small amounts of data over several kilometers. LoRa serves as the foundation for the ClusterDuck Protocol. ClusterDuck allows devices to forward messages across multiple hops, enabling coverage in remote areas. It has been applied to agriculture, wildfire monitoring, and environmental research, making it a strong fit for a wildlife-tracking camera network.

2.4 Summary

Overall, the background shows that most existing wireless and mesh systems do not meet the long-range, low-infrastructure needs of a remote wildlife camera. This makes the ClusterDuck Protocol and LoRa a practical fit for our design. The standards and testing requirements also guide us toward using safe radio settings, clear data formats, and basic reliability checks. Together, these points shape our approach: build a low-power LoRa mesh that can send sensor data across multiple hops, works without cellular service, and stays reliable through straightforward testing and validation.

---

3. Formal Product Definition

3.1 Customer / Marketing Requirements

1. The product will demonstrate the capabilities of the ClusterDuck protocol to Raspberry Pi hobbyists.
2. The product will demonstrate that ClusterDuck protocol is easy to implement on Raspberry Pi.
3. The product will use a smart sensor to collect data.
4. The product will send data over a mesh network to a cloud server gateway.

3.2 User Experience Overview

The customer will install the WildDucks in various wilderness locations up to several kilometers apart. These WildDucks will be wirelessly daisy-chained, with one node on the chain having internet access. Each sensor will identify any animals it sees and relay that information over the mesh network to the base station. The data will then be viewable through a web app.

3.3 User Story / Use Case

Consider the case of a wildlife researcher who needs to track animal activity in a remote forest. Right now, they might rely on basic trail cameras that store images on SD cards and burn through batteries. To retrieve the data, someone has to hike out every few weeks, swap cards, replace batteries, all without even knowing if the camera actually captured something useful. Without any cell service, nothing can be sent back in real time.

With DuckWild, the same job can be done much more efficiently. The researcher installs several solar-powered WildDucks, each with a motion detector, IR camera, and a Raspberry Pi running the species-identification model. When an animal walks by, the device takes a picture, identifies the species, and sends a tiny packet across the ClusterDuck LoRa mesh network. Those packets hop from node to node until they reach a PapaDuck gateway, where the data is stored and viewed online. With this solution, there's no hiking out to cameras, no SD cards, and no cellular plans.

3.4 Engineering Requirements

Spec. Number	Parameter Description	Requirements or Target	Tolerance	Risk
1	Signal range	500 m	Min	Medium
2	Model accuracy	85%	Min	High

3	Single charge lifetime	18 hours	Min	High
4	Cost	$500 per node	Max	Low
5	Lifetime	2 years	Min	Low
6	Maximum power consumption	≤ 10 W	Max	Medium
7	Operating temperature	Reliable operation from 20ºF to 100ºF	Min	Medium
8	Weather resistance	Enclosure meets the equivalent of the IP54 standard	Min	Medium
9	Packet success rate	≥ 90% successful packet transmission (line-of-sight, 500 m)	Min	High
10	Maintenance interval	No maintenance for ≥ 6 months	Max	Medium

Table 2: Engineering requirements

3.5 Constraints

Technical

1. The product must use the ClusterDuck Protocol for sending packets across the mesh network.
2. The product must use the LoRa radio standard.
3. The product must be built on a Raspberry Pi platform.
4. The product must include a smart sensor that produces data to send over the mesh network.
5. The product must be compatible with DuckLink nodes and Raspberry Pi DuckLink nodes.
6. The system must operate off-grid using solar and battery power (no Wi-Fi or cellular).
7. All computation, including AI models, must run within Raspberry Pi hardware limits (CPU, memory, and power draw).

Business

1. The project must include a plan for maintenance and upkeep.
2. The product must be protected from theft and tampering.

3. The overall design must stay within a low-cost hardware budget suitable for scalable field deployments.

Society

1. The project must address privacy concerns for residents and nature enthusiasts by avoiding human-identifying data.
2. The system must follow ethical wildlife-monitoring practices (no harm or disturbance to animals or the environment).

Other

1. The project must be completed within the allotted time (final deadline: March 13).
2. The system must be weather-resistant to survive outdoor conditions.

---

4. Conceptual Design Report

4.1 Overview of Concept Exploration

For our first ideation session, we created a functional diagram of a UAV detection system (see Figure 2). The diagram included functionality for both detection and reporting of UAVs.

During our second ideation session, we used the BrainWriting technique to generate ideas. We started by each writing our own ideas on note cards for 5 minutes. After this round, we passed the notecards all the way around our table, giving each team member 2 minutes to write questions and build upon ideas. This process involved a lot of debating, and we started to think in detail about the goals of each idea and what implementation may look like. After we had our cards back, we spent about 30 minutes creating a list of our top ideas:

1. Crowd monitoring at events (like a music festival).
2. River/water pollution monitoring and tracking.
3. Wildlife monitoring (specific applications may be Butterflies or an ecologically sensitive area)
4. Road safety and traffic monitoring.
5. UAV detection.

During our third ideation session, we used the “yes and no” method to narrow our potential projects to three promising ideas: salmon counting, drone detection, and wildlife/poaching detection. Camille was an advocate of the wildlife poaching idea, but was specifically interested in the wildlife tracking aspect. Ben thought that the drone application would be the best idea because OWL Integrations expressed that this idea would best align with their business interests. Michael was not as interested in the drone application because of ethical concerns with the defense industry. Luke was most interested in addressing wildlife poaching and drone spotting. After discussing these options with OWL, we were encouraged to make this decision ourselves so that the application would be something we were passionate about.

4.2 Concept Sketches, Schematics & Block Diagrams

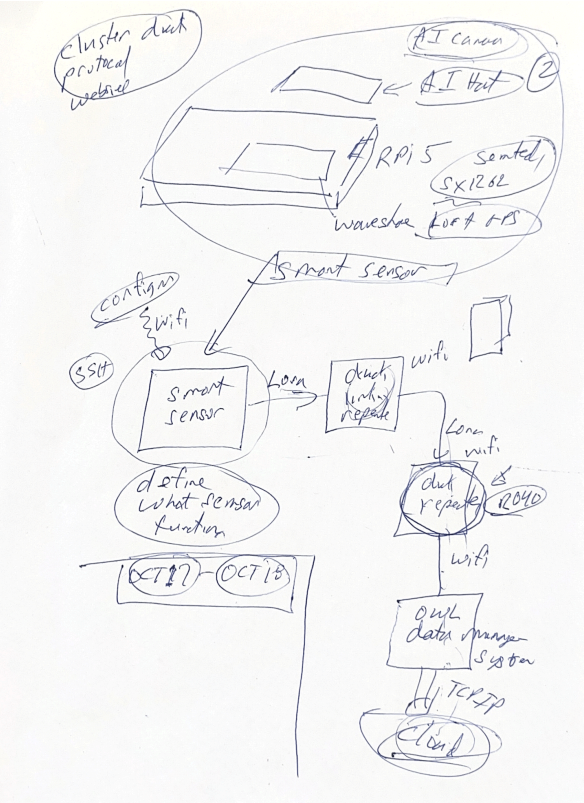

Figure 1: First concept rough sketch

Our initial discussions focused on system design rather than a particular application. This diagram, which was drawn for us by Prof. Derickson, showcases the high-level system architecture (involving a base station, repeaters, and a sensor running a local AI model) that we are still using.

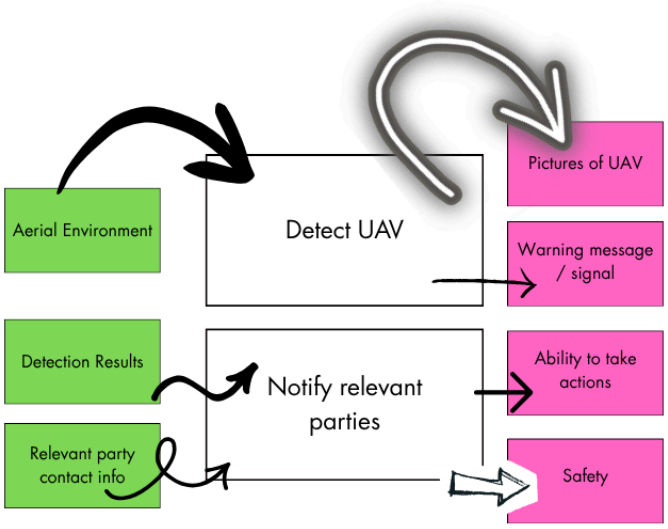

Figure 2: UAV detection alternative concept

4.3 Concept Selection Process

	Weight		Poaching Moni	Wildlife Habitat	Drone Spotting
Marketable to OWL's Clients	0.2		1.6	1.4	2
Publicity Potential to Pi Community	0.4		3.2	3.6	2.8
Cost	0.05		0.4	0.25	0.25
Helpful IRL	0.1		0.8	0.8	0.8
Ease of Development	0.2		0.6	1.4	1.6
Ease of Deployment	0.05		0.25	0.4	0.1
Sum			6.85	7.85	7.55

Table 3: Concept selection

In order to decide on a project, we evaluated three different concepts based on a number of different categories. We then assigned a weight to each category in order to calculate a final “score” for each concept. Our most important category was marketability to the Raspberry Pi community, as that was the stated goal for the project. On the other hand, cost was the least important, as the project is largely a proof-of-concept meant to demonstrate the capabilities of the ClusterDuck protocol rather than a product intended for mass production.

4.4 Selected Concept Description (Top Concept)

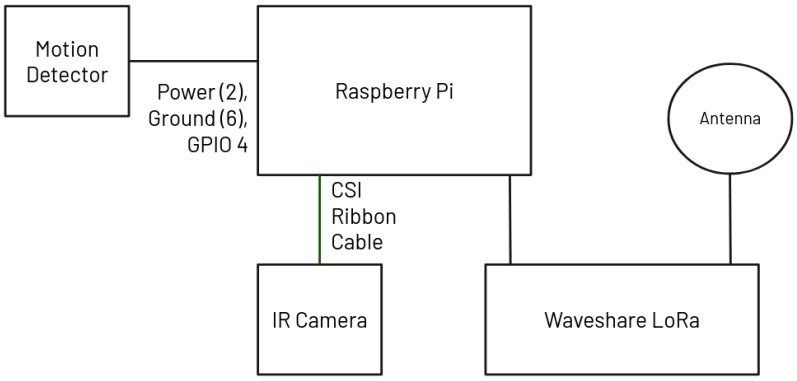

Figure 3: High-level system layout

Our product will use a motion detector, IR camera, and LoRa radio, all connected to a Raspberry Pi. When the motion detector is triggered, the Raspberry Pi will capture a photo using the IR camera, determine what species of animal the photo shows (if any), and relay that information using the LoRa radio and the ClusterDuck protocol.

Subsystems

Our project's functionality can be divided into two primary subsystems: ClusterDuck implementation and species detection.

The ClusterDuck subsystem is responsible for managing the communication with ClusterDuck nodes and the "main loop" of the device. By developing a generic "Duck" subsystem first rather than a complete "WildDuck," the ClusterDuck subsystem (including mesh networking features) can be tested entirely separately from the other parts of the device. This also makes it easier to adapt our codebase to use for other ClusterDuck projects.

The detection subsystem is responsible for detecting animals and classifying them. On the hardware side, it includes both the camera (which is attached to the Raspberry Pi through its CSI port) and a motion detector (which is attached to GPIO). The motion detector acts as a trigger for a Python script that will run the MegaDetect model (to find bounding boxes) followed by the SpeciesNet model (to identify the species of animal detected). In the final version of our project, the detection subsystem will be called within every iteration of the main loop of the ClusterDuck subsystem.

4.5 System-Level Documentation

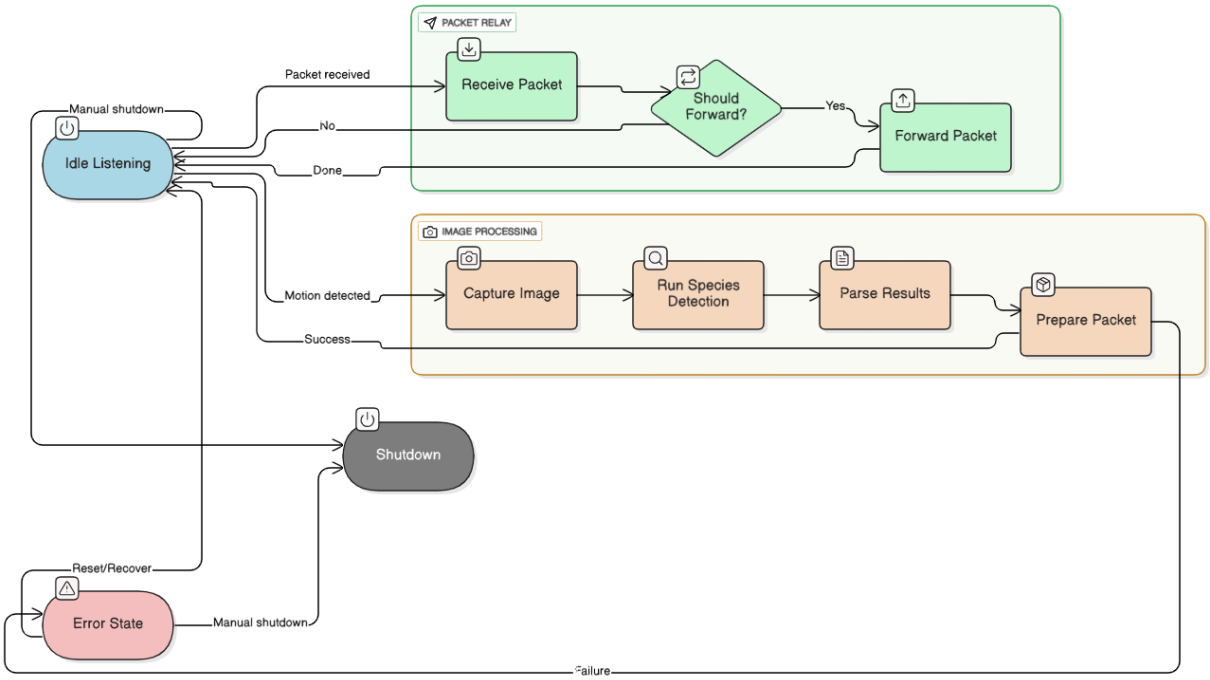

Figure 4: State machine flowchart

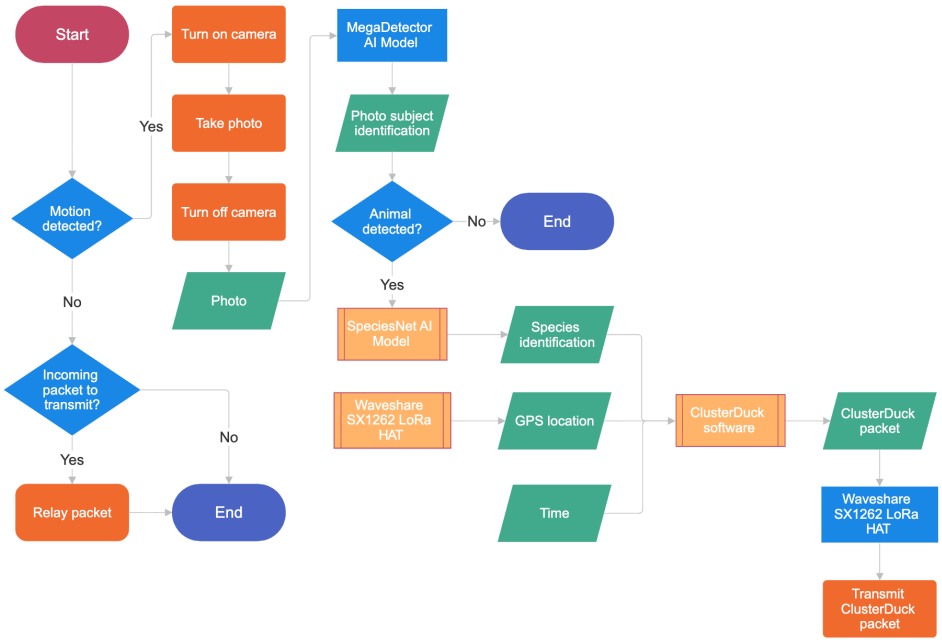

Figure 5: Full system flowchart

4.6 High-Level Software Design

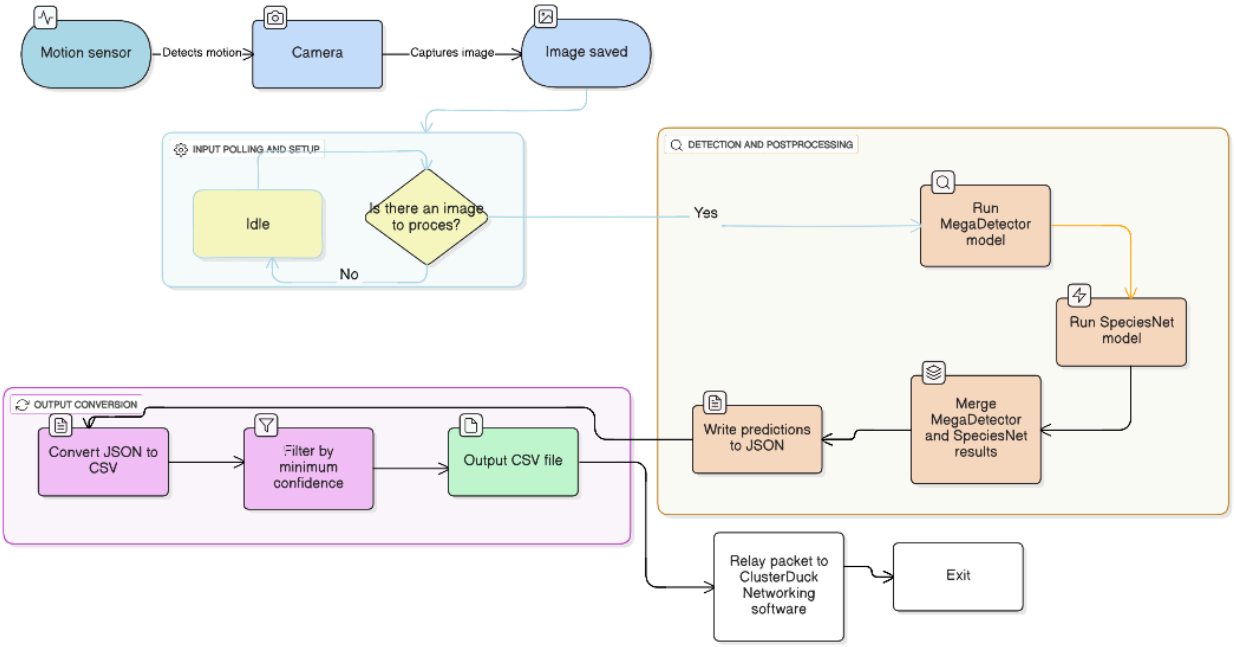

Figure 6: Computer vision / AI model pipeline software diagram

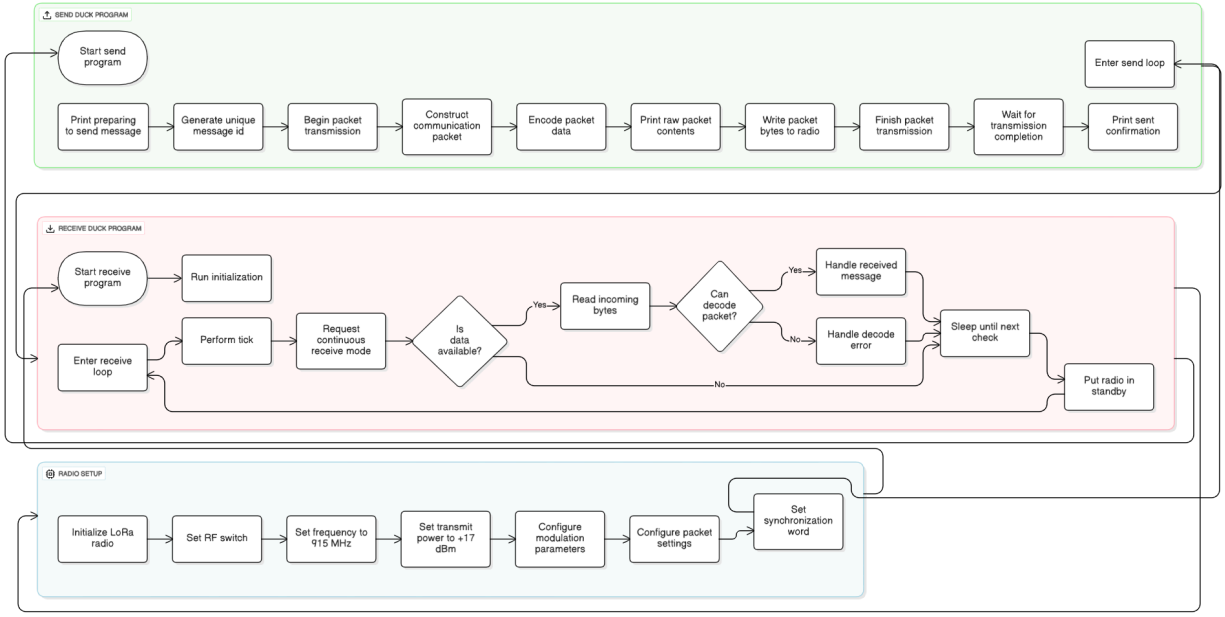

Figure 7: ClusterDuck and LoRa communication software flowchart

4.7 Mechanical Design

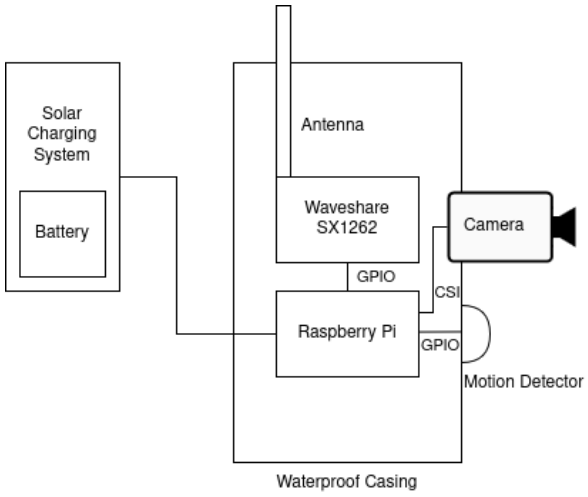

Figure 8: Hardware Diagram

The most important mechanical consideration is ensuring that the environment cannot harm our system and that our system cannot harm the environment. As for the environment affecting our system, we intend to use a custom 3D-printed waterproof case, as existing solutions won't properly accommodate both the camera and motion detector. We have also purchased weather-proof solar panels with a battery management system built-in, letting us focus on containing the Raspberry Pi itself.

4.8 Requirements Satisfaction Analysis

This concept easily meets the project goals of demonstrating the value of the ClusterDuck protocol, providing an example for hobbyists on how to implement ClusterDuck on a Raspberry Pi, and attracting members of the hobbyist community to ClusterDuck. This project demonstrates value by showing ClusterDuck being used for a relatively common application (wildlife cameras) without the drawbacks of existing techniques (e.g. the need to walk between cameras to see results or the need to pay for a cellular plan). The project will clearly demonstrate how to implement ClusterDuck on a Raspberry Pi as our subsystem architecture ensures that our codebase will be easy to reuse for different projects in the future. Finally, the project is very likely to generate hype in the Raspberry Pi community, as the application is easy to understand, has a "cool" factor (in that it showcases local wildlife), and is simple enough for hobbyists to implement on their own.

4.9 Quantitative Analysis & Justification

Based on our testing (see Section 6), our device should draw less than 10W of power at peak levels and ~2W when idle. It is unlikely that the camera will take photos more than one tenth of the time, so the average power draw will be around (2W * 90%) + (10W * 10%) = 2.8W. Assuming our 10W solar panel generates around 5W of power on average (10W assumes ideal conditions), that means the batteries are

charged at a rate of 2.2W. Over 8 hours, that means the batteries will accumulate 17.6Wh of power. Assuming an average drain of 2.8W, as noted previously, this results in approximately 6 hours of power after sundown. This currently does not meet our stated requirements, so we will have to compensate by adding more solar panels or reducing power draw.

4.10 Materials, Geometries & Manufacturing Considerations

We chose to use a 3D printed case, as that's what prior OWL projects, including Project TuiNet, have successfully used. As this project is unlikely to be mass-produced, cost is not an important factor in our component selection. For that reason, we chose to use primarily hobbyist-level components, as we didn't want complete solutions (which is what we're developing) or inexpensive bare-bones components (as those are difficult to work with). For example, as previously mentioned, we chose to use a solar panel that also included a weatherproof battery management system rather than purchasing those components individually.

4.11 Risk Analysis (Technical & Safety)

Risk	Cause	Effect	Category	Strategy	Likelihood	Severity	Risk
Capturing images of people without permission	Computer vision model mistakenly identifying a person as an animal	Privacy Concerns, Negative Publicity, Legal Action	Engineering Development, Social	Heavy computer vision model testing and multi-layers of model to verify that images are of animals before sending packets	Possible	Critical	High
Disrupting the natural environment	Bad radio/sensor placement	Disturbing the circle of life, negatively impacting ecosystems	Environmental	be mindful of the impact of the location of radio sensors	Possible	Critical	High
Sensors / network nodes stolen	Thieves, poor mounting / securing of nodes	Monetary loss, need to install another sensor / node, data security concerns	Resource, safety	Camouflage the sensors, mount them in discrete locations using robust mounts	Unlikely	Marginal	Medium
Wildfire	Overheating or battery management issues	Environmental damage, financial ruin, criminal consequences	Safety, environmental, social		Unlikely	Catastrophic	High
Toxic chemicals	Chemicals used in components (e.g. batteries)	Can leech into the environment	Safety, environmental	Weather proofing, fixed lifespan	Likely	Marginal	High

Table 4: Risk analysis

Disrupting the natural environment

Technology plays an increasingly vital role in wildlife conservation by allowing scientists to remotely monitor animal populations, track migration patterns, and identify environmental threats. The placement of monitoring equipment such as cameras, GPS collars, and motion sensors can disrupt natural behavior. Cameras or sensors placed too close to sensitive areas like dens, nests, or rub trees may stress animals, prompting them to avoid essential locations for feeding, mating, or raising their young [10]. A 2020 study published in Conservation Science and Practice found that motion-activated camera traps can alter species behavior, particularly among mammals that exhibit avoidance behavior toward unfamiliar objects or human scent [11]. Even when cameras are camouflaged, animals with strong senses—such as wolves or bears—may still detect them and respond defensively. This can distort behavioral data and create ethical concerns, as researchers may unintentionally harm the very species they aim to protect. To address this,

conservationists are experimenting with non-invasive monitoring methods, such as acoustic sensors, drones, and satellite imagery. To mitigate these effects, we plan on being strategic about our camera placement. Ensuring that they are not in areas super frequently trafficked by animals or near nests or dens will be crucial. We can also do this by camouflaging the camera so that creatures are not startled by them. Our project aims to help grow and cultivate wildlife, so we need to make sure that our sensors aren't doing the opposite of that.

Toxic chemicals

Modern electronics (including our wildlife camera) contain a variety of compounds that are uncommon or nonexistent in the natural environment. If left unattended for an extended period of time, our camera will inevitably degrade, potentially releasing these compounds. This poses a number of risks for both wildlife and people. According to a 2021 metastudy, "The toxic chemicals in e-waste can have a significant adverse impact on health of people living in exposed areas, particularly during sensitive windows of development such as pregnancy and childhood" [12]. These adverse effects can include "acute infections, morbidity, and even death." It's important to note that environmental contamination from e-waste isn't limited to scenarios in which it isn't properly disposed of; toxic leachates from e-waste can also contaminate the area around landfills.

Our project's approach to avoiding environmental contamination is two-fold. First, we will spend a significant amount of time on "weather-proofing" the device such that external factors (such as rain and wind) cannot cause the device to degrade faster than it should. Second, we will have a clearly defined "lifetime" for the device and take measures to either dismantle it ourselves after that period or arrange for someone else to do so. While we cannot guarantee that there will be no contamination whatsoever (due to unforeseen circumstances, leachates after the camera is disposed of, etc.), these measures mirror the strategies typically employed by typical wildlife cameras. While e-waste is a major unresolved issue, we can try to follow what best practices are available.

Wildfire

Wildfire is a serious risk when using electronics outdoors because overheating parts or poorly managed batteries can start fires in dry environments. If a node overheats or short-circuits, it can ignite nearby vegetation, especially in certain conditions that are hot or windy. This can lead to major environmental and financial damage, destroy equipment, and endanger lives. Wildfires caused by electronic faults have become more common as devices have spread into rural and forested areas, so preventing this risk is critical. Studies from the U.S. Department of Energy and Sandia National Laboratories show that equipment failure and extreme heat are leading causes of wildfire ignition, particularly where vegetation is dry and unmanaged [13].

To lower this risk, devices should be designed with strong thermal protection, automatic shut-off systems, and fire-resistant enclosures. Batteries must be high-quality and tested to prevent thermal runaway. Nodes should not be placed near dry brush and should be inspected regularly for heat damage. Temperature and humidity sensors can warn of unsafe conditions before a fire starts, and systems can be set to power down during high-risk weather. Following wildfire safety standards and monitoring environmental conditions can greatly reduce both the likelihood and impact of wildfires linked to our system.

Capturing Images of People Without Permission

Any AI model is known to make mistakes. This could come in the form of hallucinations in an LLM, or poor training data for detection in a computer vision model. This risk even comes through on wildlife cameras without AI integration, like cameras that operate on just a motion sensor. Wildlabs, a conservation technology network, conducted a survey of researchers who had deployed camera traps in ecological or conservation projects. More than 90% of the 235 respondents said that their cameras had taken images of people as well as wildlife [14]. This makes the risk of capturing images of humans highly likely if we don't do some preventative engineering. The consequences of taking human images without permission could be legal, although it is unlikely a human would ever find out we took their image. A more relevant consequence to our group is that this risk has negative effects on the publicity and marketing goals of our project. When we demonstrate our technology to OWL, and they inevitably share it on their website, including that the wildlife detectors use a layer in the computer vision model to block images where a human is wrongly identified as an animal from being sent across the network is a plus. So how will we mitigate this risk? First, we will heavily test the computer vision model and identify scenarios if/where a human is wrongly identified as an animal. Then, we can create a protective layer in between sending a packet across the network that filters out either human ID's or animal ID's that we would not expect in the sensor location.

4.12 Cost Estimate / B.O/M

Quantity	**Component**	**Per Unit Cost**	**Use**
4	Raspberry Pi 5	$85	Computation
4	Waveshare SX1262 LoRa Module	$25	Broadcast CDP Packets
2	PIR (motion) sensor	$10	Detect wildlife motion
2	RPi IR Camera Module	$23	Capture wildlife imagery
2	Solar System	$35	Power the DuckWild device and store power for night

Table 5: Budget justification and B.O.M.

5. Management Plan

5.1 Design Methodology

Our team follows an agile-inspired approach with structured milestones and iterative testing to ensure flexibility, consistent progress, and high-quality deliverables. We have broken the project into phases that align with our design process steps, using short work cycles and regular check-ins to reassess priorities and adjust timelines based on testing results and sponsor feedback. To track progress and maintain accountability, we use a Gantt chart to log weekly goals, task assignments, and completion status. Each week, we hold at least one group meeting to review progress, identify blocks, and redistribute tasks if need be. Between meetings, we communicated through group messaging and shared documents to ensure all members remain aligned.

5.2 Updated Timeline / Gantt Chart

GANTT CHART

Smartsheet Tip →

PROJECT TITLE	DuckWild	COMPANY NAME	OWL Integrations
PROJECT MANAGER	Camille Leute	DATE	10/22/25

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Integrating LoRa Board + Raspberry Pi**					
1.1	Set up Raspberry Pi	All	10/22/25	10/24/25		100
1.2	Set up meeting with prev capteam	Garvinator	10/22/25	10/29/25		50
1.2	Set up Waveshare board	All	10/24/25	10/31/25		100
1.3	Demonstrate simple LoRa transmitting on Pi	All	10/29/25	11/5/25		100
2	**Integrating Clusterduck + Rasp Pi**					
2.1	Meet with previous capteam		10/27/25	10/31/25		
2.2	Continue development/update existing ClusterDuck Pi codebase		11/3/25	11/21/25		50
2.3	Demonstrate ability to communicate with DuckNodes		12/1/25	12/10/25		100
2.4	Python integration/bindings					100
						100
3	**AI Wildlife Detection**					
3.1	Looking for/training model		11/3/25	11/5/25		100
3.2	Get michael camera cable	LT	10/27	10/31/25		100
3.2.1	Testing model on wildlife dataset		11/5	11/15/25		100
3.2.2	AI Hat and Camera hardware integration		11/12/25	11/22/25		NA
3.3	Real-time detection		12/1/25	12/8/25		NA
3.4	Motion Sensor and Camera Setup		12/1/25	12/8/25		100
4	**Project Performance / Monitoring**					
4.1	Meet with Bryan, show what we have so far		11/6/25	11/6/25		100
4.2	Final Presentation		11/19/25	11/19/25		100
4.3	Final Report		12/1/25	12/8/25		100
5	**Prototype Development and Testing**					
4.1	test computer vision side of prototype		12/1/25	12/5/25		100
4.2	test ClusterDuck side of prototype					100
4.3	test camera and motion sensor functionality		12/5/25	12/8/25		100
4.4	test power consumption		12/1/25	12/5/25		100

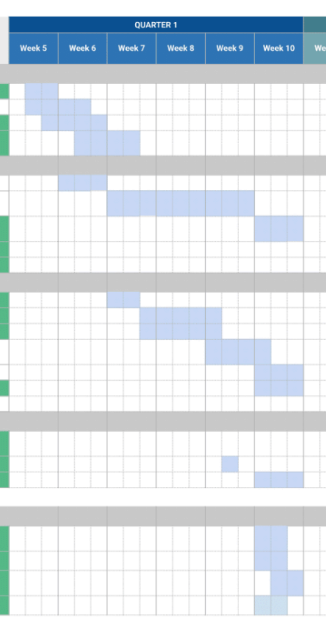

Table 6: Gantt Chart

5.3 Outstanding Tasks & Completion Plan

Our outstanding tasks to be completed for winter quarter are:

- ☐ Update Google Calendar with new class schedules
- ☐ Update Gantt chart with winter quarter tasks
- ☐ Further image post-processing
- ☐ Connect battery and solar panel
- ☐ Designing case and mounting technique
- ☐ Testing long range packet communication
- ☐ Testing in a controlled environment with physical animals
- ☐ Testing edge cases
- ☐ Testing in the wild
- ☐ Film informational demo
- ☐ Write blog post for ClusterDuck website

During the winter quarter, our focus will shift from subsystem prototyping to full integration, refinement, and field-readiness of the device. Building on the progress from fall, our remaining tasks are centered around unifying the ClusterDuck communication system with the computer vision pipeline, improving hardware robustness, and preparing the system for both controlled and real-world testing. We will continue to follow an Agile-inspired methodology, using iterative development cycles, weekly milestones, and frequent team check-ins to maintain progress and adapt to challenges as they arise. All tasks will be continuously tracked and updated on our shared Gantt chart and project management tools.

Hardware & System Integration Tasks

One of our first priorities will be completing the remaining hardware setup. This includes connecting the battery and solar panel to enable off-grid power, and finalizing a case design and mounting technique that protects the electronics while supporting long-term outdoor deployment. These components will undergo iterative prototyping and environmental stress testing to ensure they meet durability requirements. In parallel, we will update our Google Calendar with the winter course schedule and adjust our project Gantt chart to reflect the new timeline, ensuring that deadlines, testing phases, and integration milestones remain realistic and organized.

Software Refinement & Image Processing

On the software side, we will complete remaining post-processing work on collected images to improve recognition accuracy and optimize model performance on the Raspberry Pi. This phase will include refining motion-triggered inference, improving filtering of false positives, and preparing the model output for transmission through the ClusterDuck network. These refinements will be developed and tested in small, incremental sprints to maintain a steady feedback loop between detection accuracy and system performance.

Communication & Networking Validation

With both subsystems progressing, we will begin full-stack integration by validating long-range packet communication over the ClusterDuck mesh. Initial tests will be conducted in controlled environments to assess message reliability, latency, and error behavior. After confirming baseline performance, we will progress to more realistic scenarios by testing communication stability near physical animals and eventually deploying the prototype in limited field settings. Each testing stage will inform iterative adjustments to both software and hardware, following our Agile adaptation cycle.

Field Testing & Outreach Deliverables

Once the integrated prototype is stable, we will conduct edge-case testing—including low-light conditions, sensor occlusion, and intermittent network connectivity—to ensure system robustness. Successful tests will be followed by a small-scale “in the wild” deployment to gather real-world performance data. Toward the end of the quarter, we will produce project outreach materials, including filming an informational demo and drafting a blog post for the ClusterDuck website to document our process and results. These deliverables will support broader dissemination of our work and prepare the project for future teams or community contributors.

Throughout the winter quarter, our team will maintain the Agile framework adopted in the fall: weekly meetings, iterative development cycles, and regular Gantt chart updates to monitor progress and adjust priorities. This approach ensures transparency, accountability, and the flexibility required to manage the complex integration and testing stages ahead. This will prepare us to present a final product to our client, and present at the Capstone Design Expo.

---

6. Alpha Prototype

6.1 Prototype Description

This prototype is a demonstration of software functionality on the Raspberry Pi and Waveshare hardware. The software was implemented in Python, and handles both the computer vision functionality and network communication functionality of the DuckWild device.

The computer vision program automates script calls for the MegaDetector and SpeciesNet libraries, and formats the outputs into either a .csv file, or a ClusterDuck compatible packet with species ID, GPS location, and time. This program runs when an image has been captured by the camera and motion sensor and stored to the Pi drive. See section C of the appendices for more information about the Megadetector and SpeciesNet ensemble format.

Our ClusterDuck implementation is based on the headers and source file from the main ClusterDuck protocol codebase, but is dramatically simpler thanks to the higher-level nature and expansive standard library of the Python programming language. The prototype that we tested was able to serialize ClusterDuck packets and send them using the same LoRa parameters that other ClusterDuck devices use. As a result, it should be fully interoperable with all ClusterDuck devices.

6.2 Explanation of Prototype Selection

Due to delays with ordering products and shipments, our group does not yet have the solar panel, battery management system, batteries, motion sensor, or infrared camera to fully build and test a DuckWild device. The DuckWild software for this prototype was built such that the computer vision and network communication software can run separately, but the state machine is only fully operational for the computer vision part of the device. Full integration with the Waveshare hat and ClusterDuck protocol will be done early next quarter. Thus, the hardware selection for this prototype is just the Raspberry Pi 5, SX1262 LoRa Hat, motion sensor, and IR camera. We decided not to move forward with using the AI camera or Hailo AI Accelerator hat because real time computer vision detection was not a priority for the design. Running the AI camera and Hailo hat together with a standard detection model (similar to MegaDetector) consumed 4-5W of power constantly. During the day in ideal conditions, the solar panel is capable of providing 10W of power, but at night, the DuckWild would have to rely on a battery that was incapable of charging fully due to the high power draw during the day. Additionally, most of the time, there is no animal to detect, so real time detection has no discernable upside. Yet another downside is that even when running programs on the CPU, not on the Hailo or camera, the accelerators consume power. When running the detection pipeline, power usage was 6W-8W, and idle power usage was 2W. Taking off the Hailo and camera saves about 1W while running the pipeline script. Given this power usage, we have instead decided to use only the Raspberry Pi CPU for

computation. It takes approximately 20 seconds to run the detection pipeline on each image, consuming between 4-7 W of power while running. These 7W peaks only occur ⅛ of the time during the classification step of the pipeline. The rest of the time, the Pi can wait idly for a ClusterDuck packet or motion detection, running at 2W consumption.

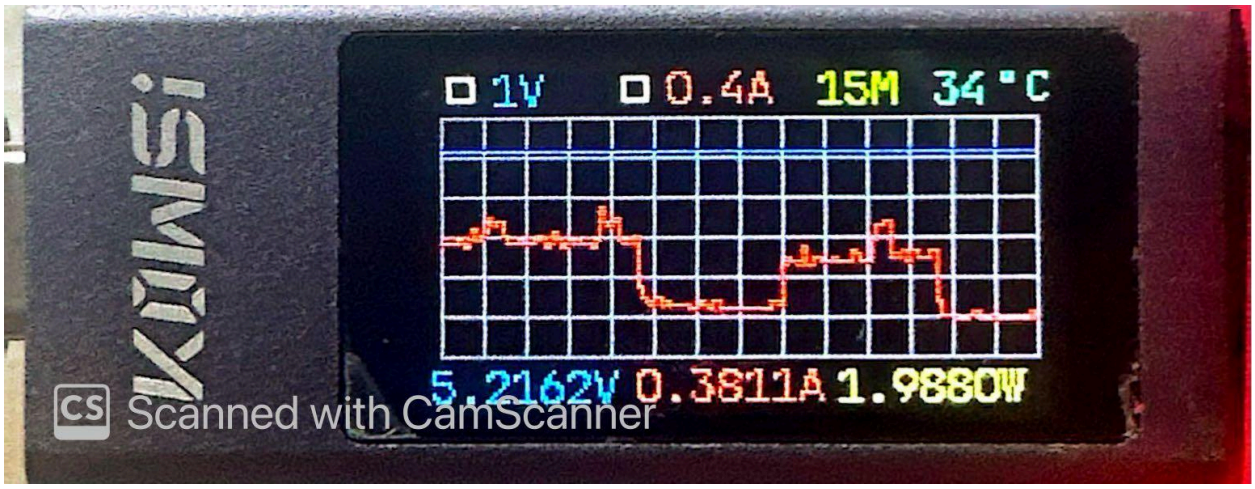

Figure 9: Power usage during and after detection

The first trace (running with the Hailo and AI camera attached) consumes between 6-8W. The second trace with just the waveshare board and infrared camera consumes between 4.5-7.5 W. Again, this level of power consumption only occurs when the script is running, which only occurs upon motion detection.

This allows the battery to charge significantly more during the day, and discharge significantly less power at night. This design choice also reduces the cost of the DuckWild by approximately $140.

As for the ClusterDuck protocol prototype, we believe that the ability to send and receive information on different devices is the most critical functionality we can implement at this stage. Ensuring that this functionality works properly is essential for implementing higher-level portions of the protocol, such as mesh networking, route information requests, and message acknowledgement.

6.3 Testing Conducted & Outcomes

A. Computer Vision Pipeline Testing

The computer vision python script was run on a dataset of 666 trailcam images of animals. The detection results, prediction confidence, and source file names were written to a csv file for evaluation. We also wrote a python script to compare detection results with the species ID of the input file. The results are as follows:

```
=== Evaluation Summary ===
Images root:           C:\Users\micha\Documents\Capstone\ClusterDuck_Images\DatasetSamples
CSV:                   C:\Users\micha\Downloads\predictions.ClusterDuck.normalized.csv
Evaluated rows:        652
Correct:               619
Incorrect:             33
Skipped (missing):     12
Skipped (ambiguous):   0
Accuracy:              0.9494
```

Figure 10: Computer Vision Accuracy Summary

```
Detector preprocess   : 100%|██████████| 666/666 [4:01:34<00:00, 21.76s/it]
Detector predict      : 100%|██████████| 666/666 [4:01:34<00:00, 21.76s/it]
Classifier preprocess : 100%|██████████| 666/666 [4:01:34<00:00, 21.76s/it]
Classifier predict    : 100%|██████████| 84/84 [4:01:34<00:00, 172.55s/it]
Geolocation           : 100%|██████████| 666/666 [4:01:34<00:00, 21.76s/it]
I1202 04:02:58.764096 140732218181376 utils.py:501] Saving predictions to `/home
/micha/DuckWild/species-pipeline/speciesnet_results.ClusterDuck.tmp.a04b71a0-565
1-48fc-8d8b-d23c2532c633.json`.
```

Figure 11: Detection results

The detection pipeline was 94.1% accurate at determining species of animal in the image set, and processed images at a rate of 21.76 seconds/image.
Average power consumption from this test is seen below:

Figure 12: Average power consumption during test

The 7.38 Watt peaks occur during (21/172) of the time, or 12.2% of the time on large datasets like the one in this test. The rest of the time, power consumption was approximately 5W.

Figure 13: 5W power consumption

This gives an average power consumption of (5W * (1- 0.122)) + (7.38W * 0.122) = 4.39 + 0.901 = **5.291W**

B. Cluster-Duck Network Communication Testing

Our Cluster-Duck communication testing focused on sending ClusterDuck packets from one Raspberry Pi + SX1262 Hat to another.

```
pi@benscapstonepi:~/DuckWild $ python3 send_example.py
Begin LoRa radio
Set frequency to 915 Mhz
Set TX power to +17 dBm
Set syncronize word to 0x1424
sending
1337 7331 2148481529 126 0 5 4117609678
b'\x00\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\x80\x0f9\xf9~\x00\x05\xf5m\xbc\xcehello there!'
sent
```

Figure 14: Transmitter LoRa Radio Setup (with correct parameters)

```
xbe~\x00\x05\xf5m\xbc\xcehello there!')
Received message 1088841406 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3h\xa9\x
e2\x05~\x00\x05\xf5m\xbc\xcehello there!')
Received message 1755963909 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\xcdA\x
1a\xb9~\x00\x05\xf5m\xbc\xcehello there!')
Received message 3443595961 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\xa6N8E
~\x00\x05\xf5m\xbc\xcehello there!')
Received message 2790144069 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\xecR\x
cf\xb3~\x00\x05\xf5m\xbc\xcehello there!')
Received message 3964850099 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\r\n#\x
cc~\x00\x05\xf5m\xbc\xcehello there!')
Received message 218776268 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3\x0b\x1
1\xb8\x0c~\x00\x05\xf5m\xbc\xcehello there!')
Received message 185710604 from 1337 : b'hello there!'
bytearray(b'\x00\x00\x00\x00\x00\x059\x00\x00\x00\x00\x00\x00\x1c\xa3R\xfb\x
03\x88~\x00\x05\xf5m\xbc\xcehello there!')
Received message 1592182152 from 1337 : b'hello there!'
```

Figure 15: Packets decoded by receiver radio

All portions of the ClusterDuck packet structure were accurately emulated, including the source and destination device UIDs, the message UID, the topic, the checksum, and the data field. These fields can also be properly decoded on the source device

---

7. Appendices

A. Bibliography / References

[1] BuckEye Cam, “BuckEye Cam,” 2025. Accessed: Nov. 09, 2025. [Online]. Available: https://www.buckeyecam.com/

[2] CuddeBack, “CuddeLink,” 2023. Accessed: Nov. 09, 2025. [Online]. Available: https://www.cuddeback.com/cuddelink

[3] J. Knapp, “What Is LoRa®?,” *Semtech.com*, Dec. 14, 2018. Accessed: Nov. 09, 2025. [Online]. Available: https://blog.semtech.com/what-is-lora

[4] Semtech, “FAQ,” *Semtech.com*, 2025. [Online]. Available: https://www.semtech.com/design-support/faq/faq-lorawan/P20

[5] ClusterDuck Protocol, “The ClusterDuck Protocol,” 2017. [Online]. Available: https://clusterduckprotocol.org/

[6] J. Wiles, “Using ClusterDuck Protocol to Monitor Vacuum and Temperature in Maple Syrup Production,” *Medium*, Mar. 28, 2023. Accessed: Nov. 09, 2025. [Online]. Available:

https://medium.com/clusterduck-protocol/using-clusterduck-protocol-to-monitor-vacuum-and-temperature-in-maple-syrup-production-3b1d0488499d

[7] E. Rodriguez, “Detecting Fires using the ClusterDuck Protocol,” *Medium*, Jun. 24, 2025. Accessed: Nov. 09, 2025. [Online]. Available: https://medium.com/clusterduck-protocol/fire-alert-system-43575163387d

[8] L. Kinajil-Moran, “Project TūīNet: Tools for Native Wildlife Conservation,” *Medium*, Jul. 31, 2025. Accessed: Nov. 09, 2025. [Online]. Available:

https://medium.com/project-owl/project-t%C5%AB%C4%ABnet-tools-for-native-wildlife-conservation-5551211158fe

[9] M. R. Villarim *et al.*, “An Evaluation of LoRa Communication Range in Urban and Forest Areas: A Case Study in Brazil and Portugal,” in *Proc. 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 827–832. [Online]. Available: https://api.semanticscholar.org/CorpusID:209458143

[10] A. Burton et al., “Wildlife camera trapping: Effects on animal behavior and recommendations for ethical use,” Wildlife Society Bulletin, vol. 44, no. 1, pp. 1–10, 2020.

[11] M. Meek et al., “Camera traps can alter the behavior of wildlife: A meta-analysis,” Conservation Science and Practice, vol. 2, no. 6, 2020.

[12] S. M. Parvez et al., “Health consequences of exposure to ewaste: an updated systematic review,” The Lancet Planetary Health, vol. 5, no. 12, pp. e905–e920, 2021, doi: https://doi.org/10.1016/S25425196(21)002631.

[13] Sandia National Laboratories, “Wildfire and Electric Grid Resilience,” U.S. Department of Energy, 2024. [Online]. Available: https://energy.sandia.gov/programs/electric-grid/wildfire-electric-grid-resilience

[14] R. Luque-Lora, B. Adams, and C. Sandbrook, “Camera traps designed for animals are now invading human privacy,” WILDLABS – The conservation technology network, 22 Nov. 2018. [Online]. Available: https://wildlabs.net/article/camera-traps-designed-animals-are-now-invading-human-privacy. [Accessed: 29-Oct-2025].

B. Team Development Goals

Individual Reflections

Camille Leute

After taking the ITP metrics test to evaluate how we work as a group, our results were communication at a 3.8, adapt at a 3.7, and relate at a 3.4. We can improve communication by utilizing our group chat more and acknowledging messages sent through text. We can also hold a briefing session at the end of classes to recognize the work we have completed so far, and what we want our next steps to be. We can improve our adaptability by being more capstone project oriented by setting more explicit goals for ourselves. As for relate, we can work on this by sharing how we feel about the project as we pass our milestones.

Michael Albert

From the ITP metrics evaluation, my lowest three teamwork traits were commitment, capabilities, and standards. To improve my ability to contribute and work with the team, I set these goals moving for the remainder of the capstone project.
Commitment Goal: I will make an effort to contribute alongside my teammates at scheduled team work sessions, and on my own when there is extra work to be done. Meeting this goal will result in a complete prototype prepared by the end of the quarter with parts of the software and hardware of the prototype where I had significant ownership and accountability.

Capabilities Goal: I will make an effort to work both on my own and with my team to develop a complete skill set required for implementing the computer vision ai model on our raspberry pi, and to handle power scaling and testing. This goal can be measured by my ability to help my teammates and by the prototype having functional and accurate computer vision capabilities with both a low and high power operation mode.
Standards Goal: I will make a conscious effort to motivate my team and help keep us on track during class and work sessions. This will be evaluated by whether or not our team is held accountable to meet regularly and complete a prototype.

Luke Trusheim

For communication, it seems that we did pretty well and could just use general improvements in our planning and role clarity. I think those can just be achieved by getting deeper into the project. For a specific action that I'd recommend, perhaps giving the team updates if you've accomplished anything with regards to the AI model or the ClusterDuck protocol.

For the adaptability area, it seems that we could use by far the most improvement in our task progress and pacing. I think since we filled out this survey, we've already done better because we're using class time to get work done and distractions are fewer. I think to further improve in this area, we could begin assigning specific tasks to people rather than just doing general work periods.

Unless I'm missing something, I think people might have misinterpreted the label for “lack of personal conflict.” I don't think we have much personal conflict in our group, and perhaps people are trying to say that our personal conflict was low, and so they rated this factor low. I would say we can improve the contribution conflict area by (as I stated earlier) assigning specific tasks to people so that our goals are actionable and measurable and people can clearly see that we're progressing towards our end objective.

Ben Garvin

Communicate: any time we have a goal that we intend to accomplish before our next dedicated work period, we should send each other at least one progress report before we meet.
Effective communication norms: we should strive for more professional communication during work periods (less humorous).
Role clarity: we should all try to accomplish at least one role-related goal in between our in-person meetings.
Planning: we should, by the end of every in-person meeting, update our Gantt chart.
Adapt: we should try to be open to implementation decisions that deviate from our original plan.
Workflow: (see communicate goal)

Task progress and pacing: (see planning goal)
Team resilience: we should dedicate some time to considering the positive impact our project could have on the world in order to increase morale.
Relate: we should use our decision making process (i.e. what we did in the Criteria & Project Selection assignment) more often, so that we can understand how our values affect the things each of us advocate for.
Contribution conflict: (see relate goal)
Lack of personal conflict: (see relate goal)
Trust: we should respond to each others' progress reports to reaffirm that we support their efforts.
Reflective communication and debrief: at the start of each work period, we should once again update each group member on our progress and explain the challenges that we've faced. This will give other group members the opportunity to offer suggestions or even reassign/reframe tasks.

Team Reflections / Actions

Planning
As a group, we are going to stay on top of planning meeting times where we can all work together in person. To do this we are going to update the gantt chart frequently and make a meeting calendar. A reflection of this action in place now is that we are taking time after each capstone class to work together and plan next steps. We have also been meeting as smaller working groups to complete sub-features of the prototype.

Communication
At the end of every capstone class or group meeting, we are going to set time aside to communicate about what is expected of each group member before the next meeting, and to share general project ideas. We also plan to communicate with the past capstone group, engineers at OWL, and Dr. Derickson moving forward next quarter as we continue working through the design and testing phase of this project. These people have all worked with the ClusterDuck protocol and mesh network technologies before, and successfully implementing the ClusterDuck protocol in python of Raspberry Pi will be the crux of this project.

Documentation
This project lends itself well to splitting up work, but to make sure that all group members understand the technology and are best able to contribute, it's essential that we document and share progress as we go. Next quarter, we will keep a running progress document for each team member so that we can check in on each other's work and answer questions we might have about parts of the project without needing to directly ask questions.

C. Species Detection Pipeline Json Output Example


```
{
   "filepath": "/home/micha/DuckWild/test1images/turkey.jpg",
   "country": "USA",
   "admin1_region": "CA",
   "classifications": {
    "classes": [

"94d8284e-6193-43b5-b80c-9029cf951675;aves;galliformes;phasianidae;meleagris;g
allopavo;wild turkey",

"83133617-8358-4910-82ee-4c23e40ba3dc;aves;galliformes;numididae;numida;meleag
ris;helmeted guineafowl",

"502958f0-482d-4a66-96d8-5018edd893ee;aves;galliformes;numididae;acryllium;vul
turinum;vulturine guineafowl",

"466b25f0-a916-432c-823e-394a69391328;aves;galliformes;phasianidae;meleagris;o
cellata;ocellated turkey",

"30015f2a-5598-461d-bc33-1eb2cef56f1a;aves;galliformes;numididae;;;guineafowl
family"
    ],
    "scores": [
     0.7459,
     0.1291,
     0.0594,
     0.0155,
     0.0086
    ]
   },
   "detections": [
    {
     "category": "1",
     "label": "animal",
     "conf": 0.7963,
     "bbox": [
      0.12,
      0.3552,
      0.4545,
      0.4754
     ]
    }
   ],
```

```
   "prediction":
"94d8284e-6193-43b5-b80c-9029cf951675;aves;galliformes;phasianidae;meleagris;g
allopavo;wild turkey",
   "prediction_score": 0.7459,
   "prediction_source": "classifier",
   "model_version": "4.0.1a"
  }
 ]
```