ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Distributed Species Tracker: Final Report for ECE 445

Team #445

RYAN DAY
(rmday2@illinois.edu)
JONATHAN YUEN
(yuen9@illinois.edu)
MAX SHEPHERD
(maxes2@illinois.edu)

TA: Hanyin Shao
Professor: Victor Gruev

May 3, 2023

Abstract

This final report outlines the problem of invasive and endangered species and proposes a solution through the development of a network of camera-equipped nodes with GPS, infrared detection, and a routing subsystem to monitor their behavior. We describe the features, design, and cost of the project, as well as its implementation.

Contents

1 Introduction

1.1 Problem

Invasive species are non-native organisms that can cause significant harm to the environment, economy, and human health. These species can outcompete native species and disrupt the balance of ecosystems, leading to economic damage and even the death of native species. Removing invasive species is a challenging and resource-intensive task, with civilians often recruited to help monitor and locate the invading species. Methods for controlling invasive species include physical removal, chemical treatment, and biological control methods. Early detection and rapid response are key to preventing the spread and establishment of invasive species [1].

Endangered species are creatures that are on the brink of extinction. A lot of conservation efforts are made in order to restore the population of the species, including gathering the animals and breeding them in a controlled environment, as well as monitoring them via a tracking chip or satellite [2].

1.2 Solution

We propose a network of nodes that, once deployed in the wild, can capture images and process them to determine whether or not a species of interest has been in a certain area. The nodes will communicate with one another in order to compile a report of all of the places and times that an animal was seen. This can be an improvement on satellite imaging that is hindered by trees and overbrush and is also an improvement over the manual scouring of wilderness that is often used in the hunt of invasive and endangered species. The network, if deployed for long enough, can offer valuable data and present a comprehensive view of a species' behavior.

This semester, we aimed to provide a proof of concept for this idea by building a small set of these nodes and demonstrating their ability to recognize an animal and log its whereabouts in a way that is redundant.

In order to do this, we fit each node with a camera that can classify images. If the species being monitored is detected and classified, its location will be sent over the network of nodes via a networking subsystem. A power subsystem supplies and regulates power to the modules in each node. A sensor subsystem was designed to provide GPS data and infrared detection.

1.3 Visual Aid

Refer to Figure 1 below for the visual aid

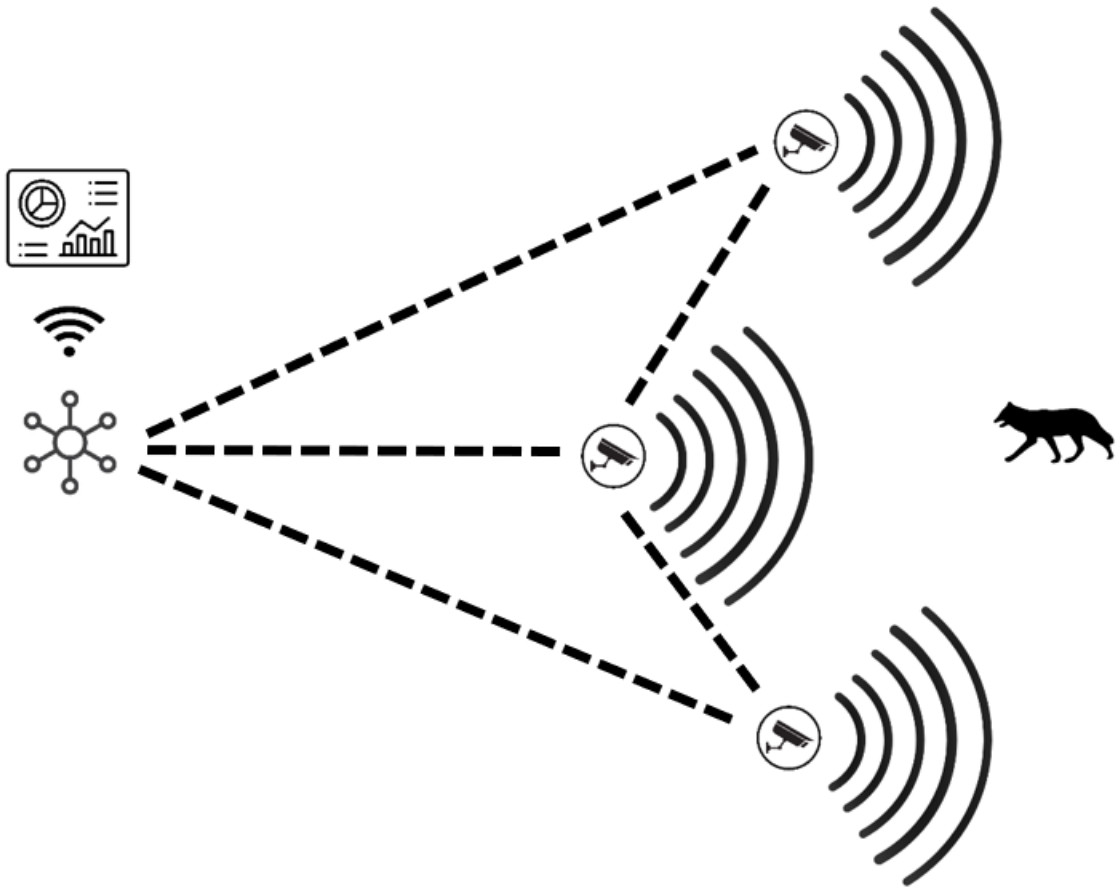

Figure 1: Visual Aid

1.4 High-Level Requirements

1. Data redundancy - We should be able to demonstrate that data gathered on any arbitrary node is reflected on the rest of the nodes in the network.
2. Detection accuracy - The system should be able to identify the presence of an animal with the infrared sensor and classify the animal we are monitoring with an accuracy of 70% or higher.
3. Power ability - The system should be able to power a node with a 3.7V LiPo battery and charge the battery via solar energy.

2 Design

2.1 Block Diagram

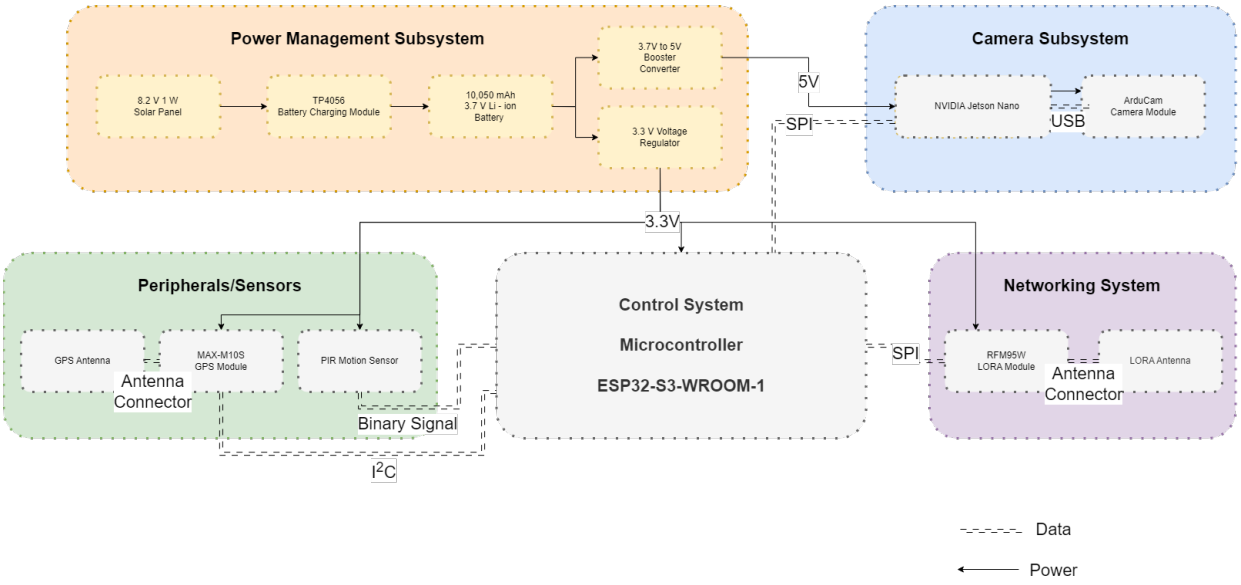

Figure 2: Block Diagram

This block diagram shown in Figure 2 consists of a power subsystem which charges a battery and regulates the power supply to the microcontroller (MCU) and the rest of the node. The networking subsystem sends and receives GPS and timestamp packets and replicates the data throughout the mesh network. The camera subsystem takes pictures of species and classifies them. The sensor subsystem gathers GPS data and detects the presence of wildlife with a passive InfraRed (PIR) sensor.

2.2 Networking Subsystem

2.2.1 Description

The networking subsystem establishes the network over which the nodes communicate. These nodes can replicate packets amongst themselves that contain information about the animal that was spotted and classified, their GPS location, a timestamp, and a numerical node ID. The networking subsystem includes an RFM95W LoRa transceiver to achieve the reliable transfer of relatively small byte packets over a long range. The LoRa module communicates with a microcontroller over SPI to receive the data that it should handle the transmission of and receives a 3.3V power supply from the power subsystem. A schematic of the networking subsystem can be found in Figure 3.

2.2.2 Design Decisions

Many design choices were made when creating the networking subsystem. Among the first things to consider was the communication method. We selected the LoRa (“Long

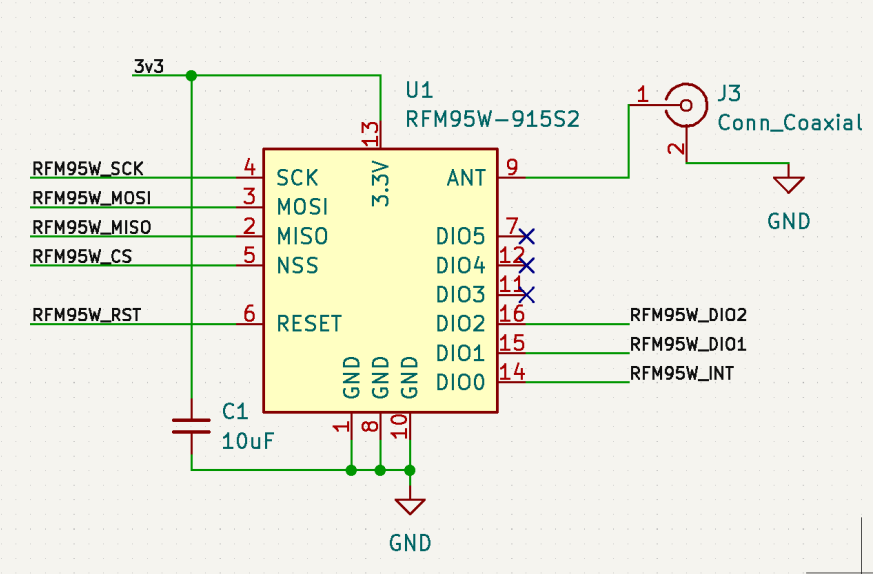

Figure 3: Networking Module Schematic

Range") protocol primarily for its low power consumption and long range capabilities. This fits our use case since nodes will be deployed and left at locations that are potentially sparse and unattended for extended periods of time and still need to be able to communicate reliably. With these long range capabilities comes a tradeoff of bit rate and acceptable packet size. However, we were willing to sacrifice transmission speed since that does not factor into our project's success and were able to make use of packets that were compact in size. On the PCB, in order to ensure that information coming to and from the antenna could be sent through the LoRa module reliably, we created a 50 ohm impedance-matched transmission line. In order to stay within two PCB layers we chose to use a coplanar waveguide with a ground plane whose thickness was calculated using the Kicad calculator.

The software component of the networking subsystem was the most complex and involved the majority of the key design elements and decisions. A Network class was written to initialize the RFM95W module, set the frequency, and customize the spreading factor, coding rate, and bandwidth. The frequency and bandwidth were set at 915MHz and 125kHz in order to be compatible with the selected LoRa module and antenna. The Network class also contained public functions that were written to create data packets, transmit and receive them, store them locally, and publish them to a server via WiFi.

In order to ensure that the process of transmitting and replicating data packets was robust in the face of scaling the size of the network or changing its topology, we used a meshing algorithm. When a node sends a message intended for a destination node, it consults a local routing table to see if a path to that node (either direct or through a path of previously discovered nodes) already exists. If a path does not exist, a route discovery broadcast is sent out and forwarded until the destination node or a node with a route to the destination node is found. The path to the destination node is passed back to the sending node, the routing table is updated, and the message is sent along that route. Ac-

knowledgement messages are sent back from the destination node to cover packet losses. In addition to this, LoRa uses forward-error correction by default – another reason LoRa was an attractive option for us.

Once a node compiles its own data packet to be transmitted or a data packet is received, we decided to store the packet in non-volatile memory so that in the worst case of the node failing before locally-gathered data could be replicated, the data can still be recovered. If the node is also deemed a "base node", the data will be published to a server via the WiFi module on the MCU. We encountered issues where the WiFi connection would fail or a connection attempt would timeout during the process of publishing data which would lead to an inaccurate representation of the sightings on the corresponding server. To address this, we created a vector that stores all received or gathered data. Whenever data is gathered or received, the entire vector is sent to be published and any data that is successfully published is removed from the vector.

2.2.3 Requirements and Verification

We were able to verify that each of the requirements for the networking subsystem were met. Details regarding the verification measures can be found in Table 1.

Table 1: Networking Subsystem Requirements and Verification

Requirements	Verifications
1) Each node must be able to successfully discover and connect with the network of Species Trackers.	When testing the networking modules, we powered on our LoRa modules and programmed an "introduction" message to be sent out once the LoRa module was initialized via the ESP32 microcontroller. These messages were then received on the other modules and printed out over USB serial to verify that they could all communicate.
2) Each node must be capable of sending the required data (GPS location, node number, timestamp) over the network or forwarding data to its neighbors to achieve data replication and redundancy.	Similar to the verification technique for the first requirement, data packets were compiled and sent out from nodes. Any received messages were printed out over serial to confirm that the correct packet was received and that no information was missing.

Table 1: Networking Subsystem Requirements and Verification (Continued)

Requirements	Verifications
3) The range of the communication between nodes should be at least 1 kilometer.	For this verification, we used two LoRa/MCU nodes - one as a receiver and one as a sender. One team member with a receiving node monitored any incoming messages via serial. Another team member then walked away with the other node, periodically sending messages. The person monitoring the receiver confirmed whether the message was properly received or not and logged the RSSI and SNR. We continued until the distance between nodes was too great for a message to be sent.

To ensure that our project would be reliable as distances between nodes scaled, especially up to a 1km benchmark, we performed a series of communication trials between pairs of LoRa modules with various combinations of spreading factors and coding rates. The results can be found in Table 2. We discovered that with a spreading factor of 12, we were able to send messages between nodes that were over 1 km apart even though there were significant obstructions in the line of sight between the nodes due to the curvature of the road on which they were placed and numerous other concrete and metal obstructions in the nodes' Fresnel zone. Details on the relevant tolerance analysis are in Appendix C.

Table 2: Max Distance Communication Trials

Spreading Factor	Coding Rate	Max Distance(m)	RSSI(dBm)	SNR(dB)
7	4/5	350	-127	-16
7	4/8	424	-121	-13
9	4/5	861	-137	-15
9	4/8	850	-139	-15
12	4/5	≥ 1000	-144	-20
12	4/8	≥ 1000	-141	-19

2.3 Power Management Subsystem

2.3.1 Description

The main function of this subsystem is to manage the charging and discharging of a 3.7V battery and provide a steady 3.3V power supply to the MCU, infrared sensor, LoRa module, and GPS module. Furthermore, the system is designed to use a boost converter circuit to step up the battery's 3.7V output to a 5V supply for the Jetson Nano. The subsystem can support a maximum current draw of 2.5A for a duration that scales with battery size. To charge the battery, the subsystem regulates the power generated by a solar cell through a battery charging module. A schematic of the battery management circuit is shown in Figure 4.

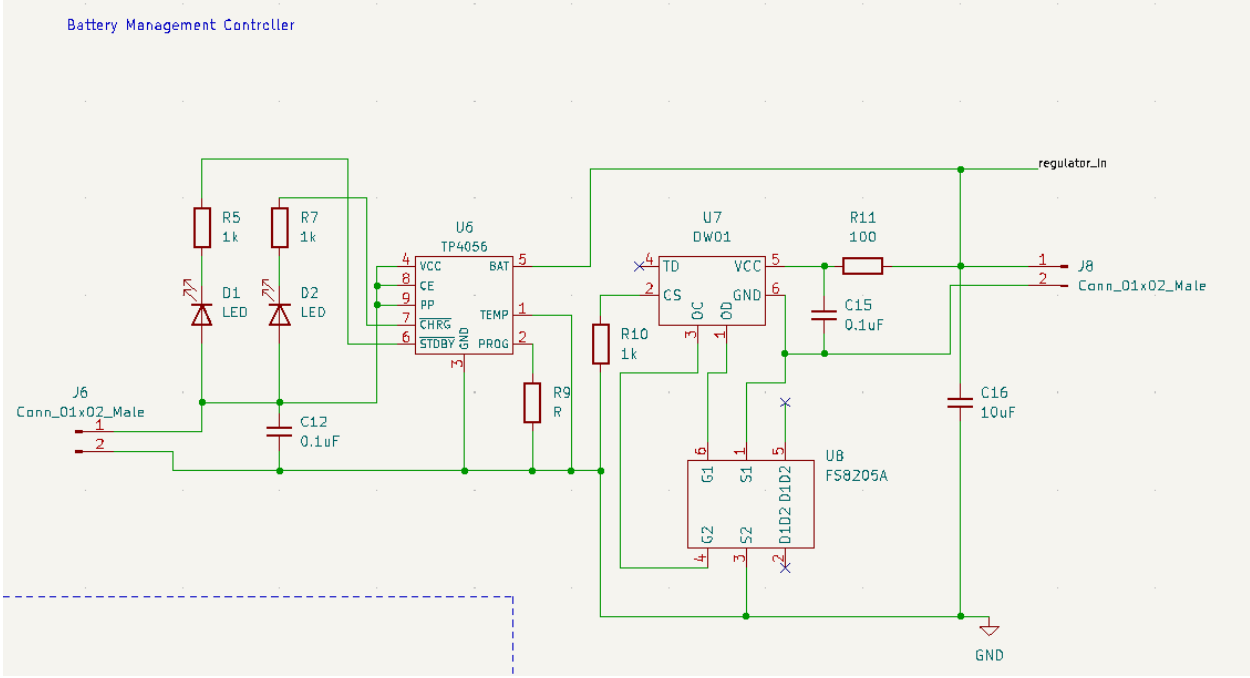

Figure 4: Battery Management Module Schematic

After a thorough analysis, we determined that a singular voltage regular would be inadequate in supplying sufficient current at 3.3V to our MCU, radio, and GPS. The regulator operates at 3.3V, but at 0.5A of current draw, it experiences a dropout voltage of 0.65V [3]. Given that the regulator is supplied with 3.7V and the system requires up to 0.8A of current draw at peak times, we thought we couldn't afford the dropout. To address this issue, we devised a solution that involves adding another voltage regulator to meet our current demands. The MCU then, can receive power from one regulator, while the other regulator can supply power to the LoRa module, GPS, and PIR sensor.

The 5V boost converter circuit was designed in reference to the Typical Application Circuit provided by the TPS61088 documentation [4]. A schematic is shown in Figure 5. The output voltage was determined by calculating the resistor values at the V_{out} PIN using the formula below.

$$R_1 = \frac{(V_{out} - V_{ref}) \times R_2}{V_{ref}} \text{ [4]}$$

With R_2 fixed at 56kΩ, R_1 was determined to be 176kΩ for an output voltage of 5V. Four 22uF capacitors were laid out at the output pin to minimize output voltage ripple to 100mV.

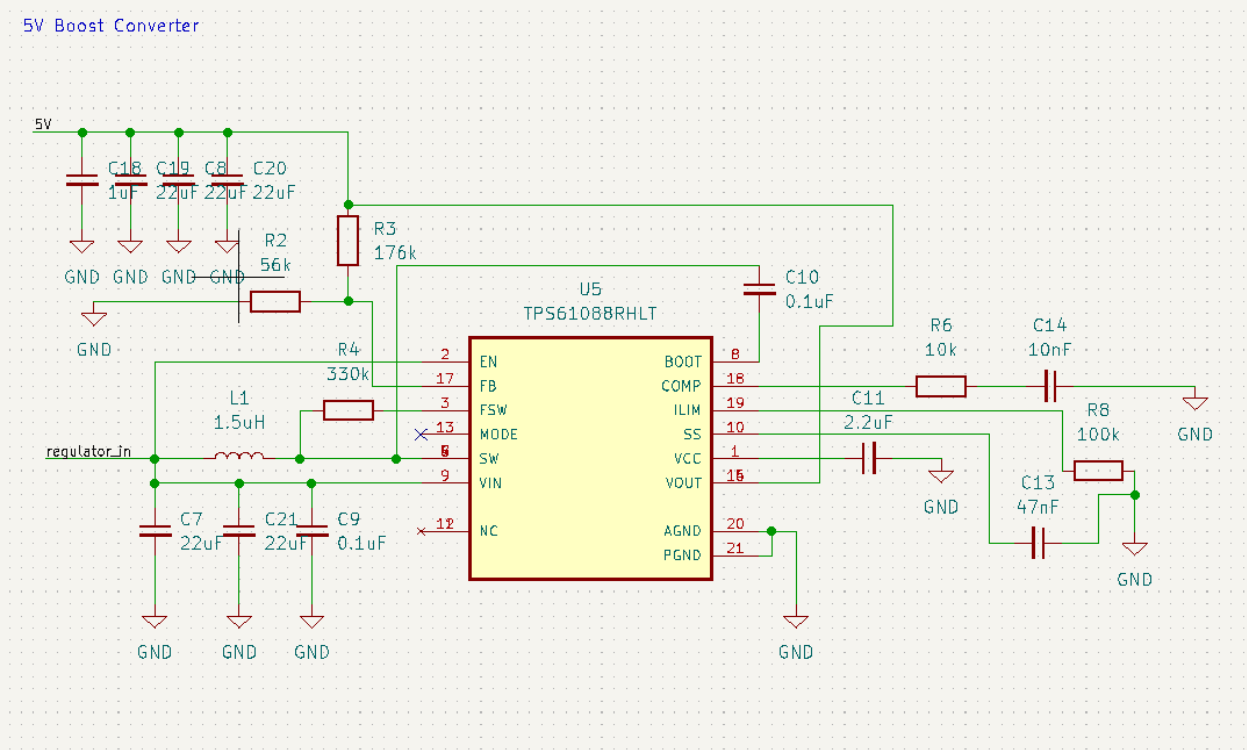

Figure 5: 5V Boost Converter Schematic

2.3.2 Design Decisions

Several design decisions were made when designing and implementing the power management subsystem. The first was the decision to implement two parallel 3.3V voltage regulators. We went with this approach to share the current draw of our system across two regulators to reduce the possible 0.65V voltage dropout [3] during peak current draw. We ended up never observing a case where this peak current draw and voltage dropout occurred so the second voltage regulator turned out to be a precautionary measure.

The second major design decision was the implementation of a solar panel charging circuit. Our nodes are intended to operate in the wild, where the nodes may be exposed to various weather conditions. For any surveillance equipment, it is not viable to need to retrieve and recharge the equipment frequently. Hence, it is imperative that our nodes last for a extended period of time. One solution was to simply use a high capacity battery and the battery would just be replaced if depleted. However, the addition of the Nvidia Jetson Nano added a layer of complexity that made that solution infeasible. Therefore, we decided to implement a solar panel charging circuit so when the nodes are deployed in the environment, the solar panel will charge the 3.7V lithium ion battery as the battery discharges to power the entire system. The TP4056 charging circuit was chosen for its thorough documentation and common use with solar panels. The TP4056 is a linear charger for single cell lithium ion batteries that supports under voltage lockout, automatic recharge, thermal regulation, and automatic charge termination [5]. While a switch mode maximum power point tracking (MPPT) charger may provide more efficient battery charging, we decided on the TP4056 to simplify our power management circuit.

According to the TP4056 datasheet, the charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached [5]. Table 3 displays the charge that is determined by the resistor values.

Table 3: Rprog Current Setting [5]

$R_{prog}(k\Omega)$	10	5	4	3	2	1.66	1.5	1.33	1.2
$I_{bat}(mA)$	130	250	300	400	580	690	780	900	1000

Next, we made the decision to use a 3.7V lithium ion battery due to its chargeability, lifespan, and size. 3.7V was also perfect as our microcontroller and other components required 3.3V. We initially wanted to use a 10,050mAh battery to ensure that the system can be powered for an extended period of time. However, the addition of the solar panel charging circuit and cost saving measures led us to reduce the battery capacity to 1,800 mAh.

The Nvidia Jetson Nano added a layer of complexity to the entire power system. It requires a steady supply of 5V and a maximum of 2.5A of current draw [6]. With the rest of our microcontroller and networking submodules requiring 3.3V, we decided to implement a boost converter circuit to reduce complexity with the battery and solar charging circuit.

2.3.3 Requirements and Verification

The power management system satisfies the following requirements:

1. The power subsystem must have the capacity to provide each node with a minimum of 4 hours of power in the absence of sunlight, and it should be able to maintain a longer lifespan with a consistent supply of sunlight.
2. The solar panel must be able to provide 4.2V to charge the 3.7V lithium ion battery.
3. The power subsystem must provide a stable 3.3 ± 0.1V power source, capable of supporting up to 0.5A of current draw from the MCU, infrared sensor, GPS module, and LoRa module.

These requirements were verified using the procedures described in Table 4.

The power management system fails to satisfy the following requirement:

1. The power subsystem must provide a stable 5.0 ± 0.1 V power source, capable of supporting up to 2A of current draw to the Nvidia Jetson Nano.

The boost converter circuit proved to be the biggest hurdle within the power management submodule. The TPS61088 was one of the only boost converter IC options that can operate within the necessary voltage and current range for our application. The TPS61088 has an extremely small footprint with extremely small connection pads spaced 0.29mm wide and 0.19mm wide respectively [4]. Despite using the PCB stencil, solder paste, and solder reflow oven, we were not able to get the boost converter circuit to operate. The connection pads also laid under the IC, making it impossible to attempt to fix solder bridging issues with solder flux. Due to our issues with the TPS61088, we were unable to implement and verify the functionality of the boost converter circuit.

We also tested the battery charging circuit. According to the complete charge cycle provided by the TP4056 datasheet in Figure 6, it takes approximately 1.25 hours to nearly fully charge a 1000mAh battery. [5] A simple way to estimate the charge time is Charge Time $= \frac{\text{Battery Capacity (Ah)}}{\text{Charge Current (A)}}$. Using the formula and estimating that battery charging will result in a

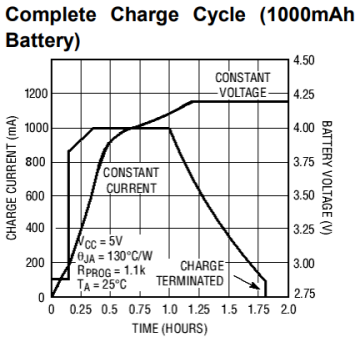

Figure 6: TP4056 Charge Cycle

maximum of 40% losses we can estimate the time to fully charge the 1800mAh 3.7V battery with a charging voltage of 4.2V and charging current of 1A to be 2.52 hours.

We found after three trials of charging an empty battery that the average charge time was 147 minutes. Comparing with the calculated charge time values using Percentage Error $= |\frac{v_a - v_e}{v_e}| \times 100\%$, the percentage error is 2.65%.

Table 4: Power Management Requirements and Verifications

Requirements	Verifications
1) The power subsystem must have the capacity to provide each node with a minimum of 4 hours of power in the absence of sunlight, and it should be able to maintain a longer lifespan with a consistent supply of sunlight.	To verify this requirement, we assembled a node with a complete power management subsystem and ensured that the solar panel was not being energized with light. We connected the output voltage pins of the 3.3V regulator to the rest of the connected node. After letting the battery drain for 4 hours, we verified that the voltage level at this pin was greater than 3.2V. If a 5V boost converter were to be integrated with the same battery source and a Jetson Nano load, its output voltage pin could also be probed and checked for a value of around 5V.
2) The solar panel must be able to provide 4.2V to charge the 3.7 V lithium ion battery.	We placed a jumper wire between the VCC (Pin 4) of the TP4056 IC and the rest of the circuit. We also probed the output of the solar panel to measure its voltage against ground. We ensured that the solar panel had access to ample lighting via an iPhone flashlight and verified a voltage level that consistently exceeded 4.2V. We also verified the bottom green LED of the charging circuit was lit up.
3) The power subsystem must provide a stable 3.3 ± 0.1 V power source, capable of supporting up to 0.5A of current draw to the micro controller, infrared sensor, GPS module, and LoRa module.	We assembled a node with a battery management system, and connected the output voltage pin of the regulator and the rest of the node. We verified that the node remained powered with 3.3V during outgoing/incoming networking messages which commands the highest current draws from the MCU and LoRa modules.

Table 4: Power Management Requirements and Verifications (Continued)

Requirements	Verifications
4) The power subsystem must provide a stable 5.0 ± 0.1 V power source, capable of supporting up to 2 A of current draw to the Nvidia Jetson Nano.	We can verify this requirement by assembling a node with a full power subsystem and connecting the boost converter to the Jetson. After several trials of processing images where current draw is highest, we can verify that the Jetson Nano is still being powered with greater than 4.9V.

2.4 Camera Subsystem

2.4.1 Description

The camera subsystem takes images of its surroundings once triggered by the infrared sensor. The image is then processed and run through an object detection model that determines whether or not the image contains the species of interest. If the model outputs a positive classification, the microcontroller compiles a packet of data to be sent to the networking submodule. The only components in the camera subsystem are a USB camera and an Nvidia Jetson embedded GPU computer which runs the deep learning model. Refer to Figure 8 in Section 2.6 to see how the camera subsystem connects to the MCU via the "Jetson" wires - there is no separate schematic for the camera subsystem.

2.4.2 Design Decisions

The first design decision that we faced for the camera subsystem was choosing which device would perform the image processing. Initially, we were going to only use a camera and the MCU to capture and process the image. However, this implementation would have required writing neural network code in C++ for a device that is non-native to PyTorch and Tensorflow, as well as implementing a driver for our camera. We chose to use the Nvidia Jetson Nano development board instead because we could use many tools that are fully developed such as Pytorch and device drivers. This device is also advantageous over the Raspberry Pi in it's neural network inference time due to its GPUs. This comes at the cost of a higher power consumption, which is addressed in Section 2.3.

The next design decision we made was choosing the communication protocol between the MCU and the Jetson. Initially, we planned on using SPI, but this was not possible as both the MCU and the Jetson are configured to operate in master mode and could not be altered to work as a slave. So, the final communication protocol that we went with was a single wire from MCU to Jetson to notify the Jetson to take a picture and two wires from Jetson to MCU to denote a positive classification and a negative classification. Therefore, one of the pins allocated for SPI on the Jetson and MCU went unused but no other part of the project was affected by this choice.

Another large design decision we made was how to implement a human classifier. Our initial idea was to download a pretrained convolutional neural network to do binary classification to determine if a human was in a picture or not. We were unable to find any such network so, instead, we chose to do classification using object detection. Object detection is the task of localizing and classifying all objects of interest in an image and the Jetson Nano has several powerful models to do this. We chose to use the Single Shot Detector network with a mobilenet backbone, a faster and less memory-consuming model which is ideal for our purposes.

2.4.3 Requirements and Verification

Details regarding the requirements we set for the camera module and how we verified them can be found in Table 5.

Table 5: Camera Subsystem Requirements and Verification

Requirements	Verifications
1) The deep learning model should have an accuracy of $> 80\%$ in the wild.	We ran 50 cases where a person was in sight of the camera and triggered the PIR sensor, and 50 cases where a person was not in front of the camera. Of the 50 positive trials, 44 detected a person in the image, and of the 50 negative trials, all 50 detected that there was no person in the image. This gave us a rough estimate of the accuracy of our model at 94%
2) The average time between the PIR sensor being triggered and the corresponding message being sent should be below 100 ms.	We grabbed timestamps at when the MCU was triggered by the PIR sensor and another timestamp when the resulting data packet was outbound from the LoRa module to measure the latency.

We were successful in verifying Requirement #1 in Table 5, however, we had to relax the second requirement set that calls for a 100ms detection-to-send latency. In order for the Jetson Nano to detect that the MCU is driving one of its GPIO pins high and then low in order to initiate an object detection sequence, the pin needs to be held low and high for an adequate period of time. For each device, we set this time to be 1 second in total. This, in addition to the overhead of running the object detection took roughly over 2 seconds. Results of ten trials in which the detection-to-send latency was measured can be found in Table 6.

Table 6: Detection-to-send Latency

Trial #	1	2	3	4	5	6	7	8	9	10
Latency (ms)	2014	2234	1988	2055	2304	2111	2008	2047	1976	2289

2.5 Sensor Subsystem

2.5.1 Description

This subsystem is responsible for gathering GPS data and hosting the infrared sensor. GPS data is to be sent to the MCU via I2C then processed and packaged into the data to be transmitted by the networking submodule when needed. The infrared sensor detects living creatures in the proximity of the node and triggers a photo to be taken by a camera. Figure 7 shows the schematic devised for the GPS module. Refer to Figure 8 in section 2.6 to see how the PIR sensor connects to the MCU via the "PIR-DATA" wire. There is no separate schematic for the PIR sensor.

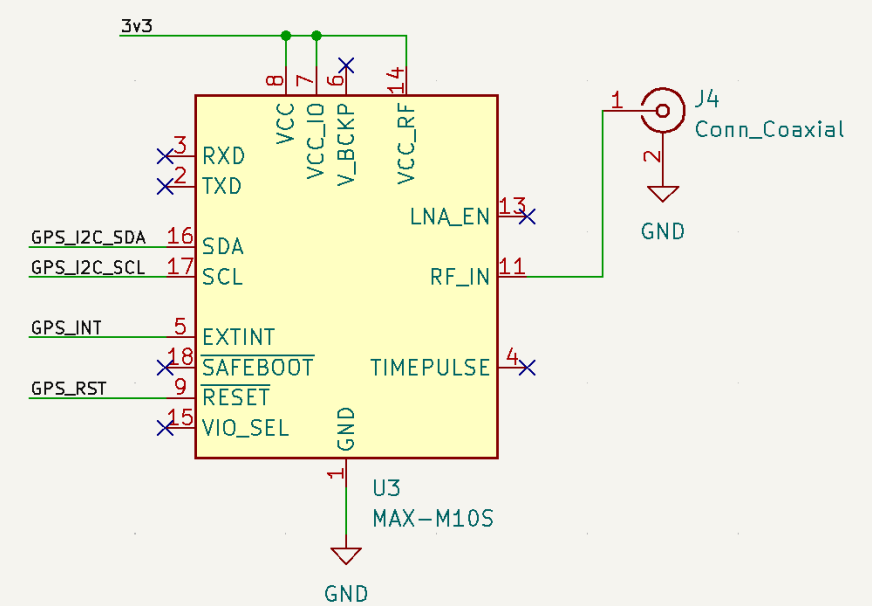

Figure 7: Max-M10S GNSS Schematic

2.5.2 Design Decisions

We opted to use cheap PIR sensors with very simple 3-pin interfaces to make communication with the MCU easy and to reduce cost. In order to get rid of false positives on the MCU pin connected to the PIR sensor's data line, a pull-down resistor was used to drive the digital logic level low by default.

A GPS sensor was originally included in our design to feed real-time latitude, longitude, and altitude data to our MCU. However, after researching GPS chips and their respective

breakout boards for testing, we concluded that both were too expensive to justify purchasing for this project. We also concluded that our use case does not necessarily lend itself towards using a GPS sensor since the nodes are meant to remain stationary once deployed - a GPS location can be flashed into memory and included in any outgoing data packets. In fact, this is what we elected to do for our testing and demonstrations.

2.5.3 Requirements and Verification

Details regarding the requirements and respective verification steps taken for the sensor subsystem can be found in Table 7. We were able to meet Requirement #2 but were unable to meet Requirement #1 since we decided against including a GPS module in our final design.

Table 7: Sensor Subsystem Requirements and Verification

Requirements	Verifications
1) The GPS module must provide a location that is within 10 meters of the node's true location.	We can verify this requirement by gathering unique GPS locations sent from the GPS module to the MCU. We can take note of the true GPS location and compare the values for all data points, verifying that they are within 10 meters.
2) The PIR sensor must be able to trigger the camera to take a photo if a living animal is within 5 feet (on a bigger budget, we would invest in sensors with much greater range).	We verified this requirement by testing the PIR sensor in isolation with the camera module. We walked past the PIR sensor at a distance of 5 feet and verified that object detection was prompted.

2.6 Microcontroller Subsystem

2.6.1 Description

The microcontroller hosts the software that manages the sending and receiving of messages throughout the network and handles communication with the infrared sensor, and Nvidia Jetson. A schematic of the ESP32-S3-WROOM-1 MCU and its boot and reset circuits can be found in Figures 8 and 9. The RC circuit in Figure 9 was taken from the ESP32's datasheet in order to provide a sufficient boot delay.

2.6.2 Design Decisions

The primary purpose of our ESP32-S3-WROOM-1 microcontroller is to run code that can receive and process messages in parallel with code that interfaces with the PIR sensor and Jetson before packaging and sending out data. For this, we made use of the dual core architecture of our MCU, running threads that each handle one of these two functionalities.

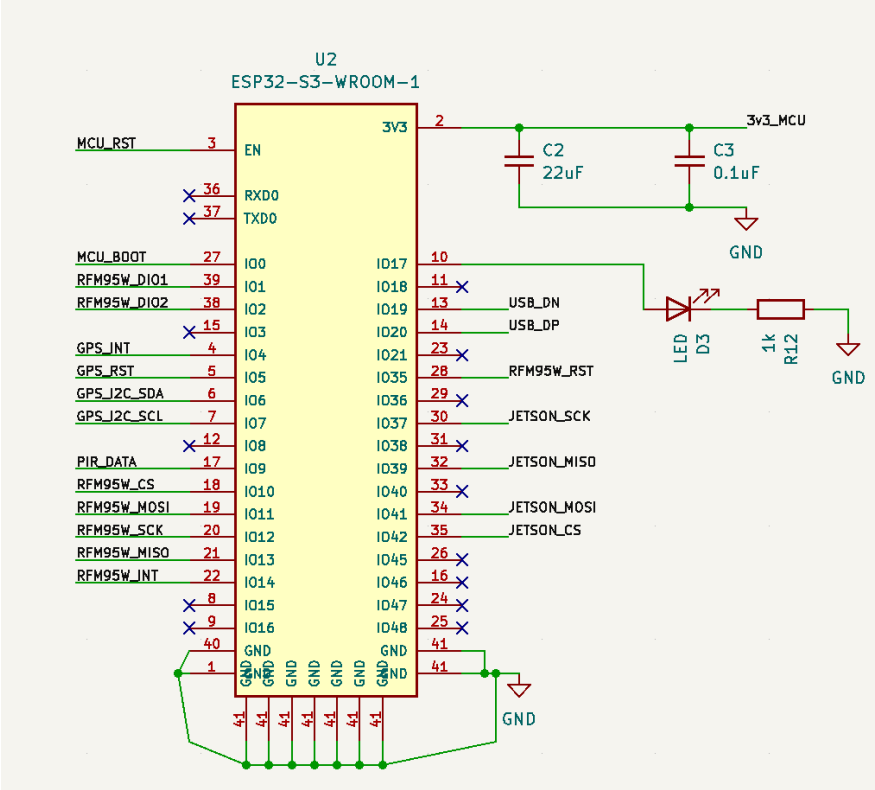

Figure 8: ESP32-S3-WROOM-1 Schematic

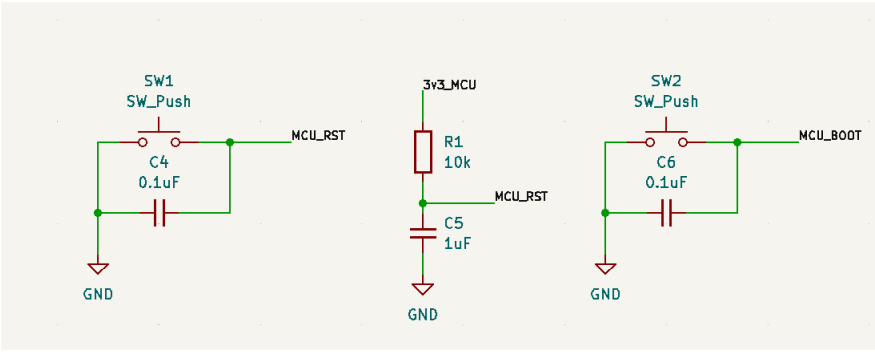

Figure 9: ESP32-S3-WROOM-1 Boot/Reset Schematic

We tested this software by triggering the sending of a message while an incoming message was being received to verify that our mutual exclusion primitives were employed correctly and that the flow of the software continued like normal. In order to handle redundant influxes of inputs from the PIR sensor we implemented a buffer zone in the software to give time for the Jetson to process images and communicate its results to the MCU. We also wrote a GPS class that includes functions for initializing a GPS module and requesting latitude, longitude, and altitude information via I2C but we never used it since we opted to scrub the GPS module from our project. More information about this can be found in section 2.5.

Our MCU is the most crucial component of our project which is why so much of our efforts went into attempting to flash the MCU once it was soldered onto our PCB. When prototyping with our development boards, we always made use of the built-in USB-OTG interface on the ESP32-S3-WROOM-1, so we designed our PCB so that we could flash the chip with a USB port directly to the data pins. We ran into connection issues that prevented our laptops from recognizing the MCU on their serial ports. We tried replacing the ESD protection chip on the USB datalines, bypassing the ESD protection chip entirely, and isolating the ground pin on the USB port all while confirming that the MCU was being properly powered with an oscilloscope. When all of this failed, we attempted to flash the chip through a USB-UART bridge but ran into issues regarding invalid packet headers. We attempted to debug this by implementing pull-up resistor circuits on the MCU-side RX/TX pins to ensure proper logic levels but were still unsuccessful. Finally, we attempted to use a JTAG programmer but still could not flash our MCU. Ultimately, we decided to finish our project with the MCU and its PIR and LoRa peripherals broken out on a breadboard.

2.6.3 Requirements and Verification

We were able to verify all requirements that we set for the microcontroller except for Requirement #2 in Table 8 that involves polling a GPS module since we decided against including a GPS module in our final design. Table 8 also contains the other details regarding the verification methods for each microcontroller requirement.

During verification of Requirement #3, we ran ten trials measuring the latency between the classification of an image and the sending of a message. The latency was very low and did not deviate from a 2-4 millisecond range. Had we included a GPS module in our design, we would have to wait for a response from the GPS before packaging and sending out a message meaning that our picture-to-send latency would have been higher and closer to the 1 second threshold we set.

Table 8: Microcontroller Requirements and Verification

Requirements	Verifications
1. The microcontroller should be able to run code that can parallelize the processing of messages coming in from the LoRa module and messages from the camera module regarding a specific sighting.	This was tested with the ESP32-S3-DevkitC-, the Jetson GPU, and the RFM95W breakouts by flashing software to the MCU and setting print statements and flashing an LED when a message comes in from the camera module, when a message is outbound via the transceiver, and when a message is inbound via the receiver. We pinned the receiver handling and the camera/sending handling threads to their own separate cores in the MCU to explicitly determine how these peripherals were interacting with the MCU.
2. The microcontroller should be able to poll the GPS module or request that it send updated GPS coordinates.	We can verify this requirement by flashing a program on our MCU that interacts with the GPS module by periodically logging the input from the GPS module and by sending it an interrupt indicating that a location should be sent. We can mark these events with serial print statements and send the received GPS information over serial as well in order to verify that updated GPS coordinates can be sent.
3. The microcontroller should be capable of executing the code that manages the LoRa module, the sensors, and communication with the Jetson such that the latency for gathering and processing data from the sensors does not exceed 1s. Therefore, when a picture is taken, the packet of data composed of GPS location and timestamp should be on the way to the LoRa module within 1 second.	To verify this requirement, we simply took timestamps at the point when the GPIO pin connected to the PIR sensor's data line went high, and compared it to the timestamp when the next outbound message from the RFM95W was sent.

3 Cost Analysis

3.1 Parts and Materials

We created three boards to showcase the networking abilities during the project demonstration. Refer to Table 11 Appendix A to see the PCB board components and their respective expenses. This list does not include the cost of shipping. The overall cost sums up to $913.50.

3.2 Labor Costs and Schedule

The Illini Success 2020-2021 Annual Report of the Grainger College of Engineering reports that the average annual starting salary of a graduate in Computer Engineering is $105,352 [7], which is equivalent to $52.68 per hour. A summary of these labor costs is in Table 9 and a schedule of this work can be found in Table 12 Appendix B.

Table 9: Labor Costs

Name	Weeks	Hours Per Week	Hourly Rate	Fudge Factor	Cost
Ryan	10	10	$52.68	2.5	$13,170
Jonathan	10	10	$52.68	2.5	$13,170
Max	10	10	$52.68	2.5	$13,170
Total Cost:	$39,510				

3.3 Total Cost

Table 10: Total Cost

Category	Estimated Cost
Labor	$39,510
Parts and Materials	$913.50
Development Resources	$103.50
Total Estimated Cost:	$40,527

The total estimated costs incurred over the course of our project are listed in Table 10.

4 Conclusion

4.1 Conclusion

We were successful in implementing a network of nodes that can detect and classify images with greater than 70% accuracy, replicate the relevant data amongst themselves for data redundancy, and publish the data to a server whose contents are viewable through a UI. We were able to implement a circuit on a PCB capable of powering every part of a node except for the image-processing Jetson Nano with a 3.7V LiPo battery and charging that battery with a solar panel. Therefore, we were able to meet all of our high level requirements. As discussed in Chapter 2 of our report, we did not integrate a GPS module with our project and were unable to implement a 5V boost converter circuit to power the Jetson Nano on a PCB. Also as discussed in Chapter 2, we were unable to integrate our MCU, LoRa module, and PIR sensor on a PCB due to issues flashing the MCU.

4.2 Future Work

In the future, we would like to implement a more robust failure detector for the networking subsystem so that nodes can publish data to a server about which nodes are no longer in service or have run out of battery. We would also like to consolidate the image processing efforts to the MCU so that we are not reliant on the separate, bulky hardware that comes with the Jetson Nano. In the future, we want to implement a more interactive UI that allows users to visualize on a map where the nodes are deployed. In our power subsystem, we would like to pivot towards using a switch-mode MPPT charging circuit instead of a linear charging circuit for more efficient power usage.

4.3 Ethics and Safety

Over the course of the development of our project, we made sure to work ethically and abide by the IEEE code of ethics. We prioritized collaborative and communicative work amongst our teammates to produce a viable project. For full details on Ethics and Safety, refer to Appendix D.

References

[1] Invasivespeciesinfo.gov. ""Control Mechanisms — National Invasive Species Information Center"." (2019), [Online]. Available: https://www.invasivespeciesinfo.gov/subject/control-mechanisms (visited on 02/09/2023).

[2] Y. E360. ""A New Way to Track Endangered Wildlife Populations from Space"." (2021), [Online]. Available: https://e360.yale.edu/digest/a-new-way-to-track-endangered-wildlife-populations-from-space (visited on 02/09/2023).

[3] M. T. Incorporated. ""TC1262 Features Package Type General Description"." (2009), [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21373C.pdf (visited on 04/29/2023).

[4] T. I. Incorporated. ""TPS61088 10-A Fully-Integrated Synchronous Boost Converter Typical Application Circuit TPS61088"." (2015), [Online]. Available: https://www.ti.com/lit/ds/symlink/tps61088.pdfts=1682882154337&ref_url=https%5C%253A%5C%252F%5C%252Fwww.google.com%5C%252F (visited on 04/29/2023).

[5] . N. T. P. A. Corp. ""TP4056 1A Standalone Linear Li-lon Battery Charger with Thermal Regulation in SOP-8"." (), [Online]. Available: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf (visited on 02/09/2023).

[6] N. CORPORATION. ""NVIDIA Jetson Nano System-on-Module Datasheet"." (2022), [Online]. Available: https://developer.download.nvidia.com/assets/embedded/secure/jetson/Nano/docs/JetsonNano_DataSheet_DS09366001v1.1.pdf?FU972qZV8qrBfnvdY2PMgPfEAQmCfL5Qm8o5oOTPpBh5rsVa14FJ0ORyKjZKuyswQlglfM-o-V7GzjKgmKlsN_kyLsZvTx47oKn1EVsfrpGsy_cEtcZMNEW7ySJytJOCYH3PYO9nbbthfWLfStUvy8gvSmQ==&t=eyJscyI6ImdzZW8iLCJsc2QiOiJodHRwczovL3d3dy5nb29nbGUuY29tLyJ9 (visited on 04/29/2023).

[7] B. of Trustees at the University of Illinois at Urbana-Champaign. ""Illini Success 2020-2021 Report"." (2021), [Online]. Available: https://uofi.app.box.com/s/aoply09y5kf6i36es8v3bl758n2lfl08 (visited on 04/29/2023).

[8] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2023).

Appendix A PCB Parts Itemization

Table 11: Parts Cost

Component	Manufacturer	Quantity	Unit Price	Total Price
RFM95W 1528-1667-ND	Adafruit	3	$19.50	$58.50
RFM95W-915S2	RF Solutions	3	$14.50	$43.50
ESP32-S3-DevKitC-1-N8R8	Espressif Systems	2	$15.00	$30.00
ESP32-S3-WROOM-1U-N16R8	Espressif Systems	3	$4.20	$12.60
ANT-916-CW-HW-SMA-ND	LINX Technologies	4	$9.89	$29.67
1.6mm Edge-Launch SMA Connector	Adafruit	3	$2.50	$7.50
USB micro connector	Adafruit	3	$2.95	$8.85
ESD protection USBLC6-2P6	STMicroelectronics	3	$1.00	$3.00
MAX-M10S MAX-M10S-00B	U-blox	3	$21.00	$63.00
PIR motion sensor	Adafuit	3	$9.95	$29.85
22uF capacitor GRM21BR60J226ME39L	Murata	3	$0.29	$0.87
1 uF capacitor TCKIX105CT	Cal-Chip Electronics, Inc.	3	$0.29	$0.87
0.1 uF capacitor T491A104K035AT	KEMET	9	$0.62	$5.58
10k Ohm resistor RT0603FRE0710KL	Yageo	3	$0.10	$0.30
Battery charging IC TP4056	Seeed Technology Co., Ltd	3	$4.90	$14.70
3.3V regulator 579-TC1264-3.3VDBTR	Microchip Technology / Atmel	3	$0.92	$2.76

Table 11: Parts Cost (Continued)

Component	Manufacturer	Quantity	Unit Price	Total Price
5V boost converter LM2585S-12/NOPB	National Semiconductor	3	$6.17	$18.51
Barrel connector 10-01935	Tensility International Corp	3	$3.23	$9.69
Adafruit Lithium Ion Polymer Battery 3.7V 10050mAh (10 Ah)	Adafruit	3	$29.95	$89.85
Solar panel 313070005	Seeed Technology Co., Ltd	3	$12.30	$36.90
Jetson Nano Developer Kit	Nvidia	3	$149.00	$447.00
Total Components Cost				$913.50

Appendix B Schedule

Table 12: Labor Schedule

Week	Tasks Completed
1 (Week of 1/30)	• Ryan: Worked on flashing the ESP32-S3 Devkit. • Jon: Worked on flashing the ESP32-S3 Devkit. • Max: Researched image classifiers.
2 (Week of 2/6)	• Ryan: Worked on project proposal. Researched drivers for the RFM95W LoRa module. • Jon: Worked on project proposal. Researched the power submodule. • Max: Worked on project proposal. Weighed pros and cons of using the MCU for image classification or an external computer.
3 (Week of 2/13)	• Ryan: Started working on RFM95W software to send messages between the nodes. • Jon: Further researched specific components for the power subsystem. Helped debug initial RFM95W development to get the transceiver working. • Max: Helped debug intial RFM95W development to get the transceiver working.
4 (Week of 2/20)	• Ryan: Worked on the design document. Continued development on LoRa communication. • Jon: Worked on the design document. Researched the GPS module and PIR sensor. • Max: Worked on the design document. Finalized what board will be used for image processing and what camera will be used.
5 (Week of 2/27)	• Ryan: Worked on powering and setting up the Jetson Nano. Designed and routed the PCB. • Jon: Designed and routed the PCB. • Max: Set up the Jetson Nano development environment and worked on a reliable powering method.
6 (Week of 3/6)	• Ryan: Tweaked PCB design and ordered. • Jon: Tweaked PCB design and ordered. • Max: Worked on model development for the Jetson.
7 (Week of 3/13)	Spring Break

Table 12: Labor Schedule (Continued)

8 (Week of 3/20)	• Ryan: Started development on the meshing protocol for the networking subsystem. Began working on a WiFi module. • Jon: Laid out tests and verification for the battery charging circuit. • Max: Continued Jetson development and model testing.
9 (Week of 3/27)	• Ryan: Soldered the first PCB boards. Tested and verified PIR sensor. Established communication between MCU and Jetson. • Jon: Soldered the first PCB boards. • Max: Established communication betweeen MCU and Jetson.
10 (Week of 4/3)	• Ryan: Worked on debugging issues with flashing the MCU. Continued mesh network development. • Jon: Worked on debugging issues with flashing the MCU. • Max: Continued Jetson/MCU development and ironed out communication between the two.
11 (Week of 4/10)	• Ryan: Tested and verified the PCB battery charging circuit. Ironed out the meshing protocol and tested. • Jon: Tested and verified the PCB battery charging circuit. • Max: Ran tests with the Jetson classifier and integrated with the MCU.
12 (Week of 4/17)	• Ryan: Practiced mock demos as a team. • Jon: Practiced mock demos as a team. • Max: Practiced mock demos as a team.
13 (Week of 4/24)	• Ryan: Worked on the final presentation and report. • Jon: Worked on the final presentation and report. • Max: Worked on the final presenation and report.

Appendix C LoRa Tolerance Analysis and Evaluation

During our tolerance analysis earlier in the semester, we determined that for our LoRa module, in order for a message to be received correctly, the received power (RP), represented in Equation 1, must be greater than -144 dBM.

$$\begin{aligned} \text{RP} &= \text{TP + TAG - TL - PL - ML + RAG - RL} \\ \text{TP} &= \text{Transmitted Power} \\ \text{TAG} &= \text{Transmitter Antenna Gain} \\ \text{TL} &= \text{Transmitter Losses (transmission line, connector)} \\ \text{PL} &= \text{Path Loss} \\ \text{ML} &= \text{Miscellaneous Loss} \\ \text{RAG} &= \text{Receiver Antenna Gain} \\ \text{RL} &= \text{Receiver Loss} \end{aligned} \tag{1}$$

The Transmitted Power for an RFM95W is +20 dBm and the gain of our antennas is 1.2 dBm. This means that (TL + PL + ML + RL $<$ 166 dB). We made sure to use impedance matched antennas and transmission lines which rendered RL and TL negligible, leaving a roughly 166 dB budget for path loss and miscellaneous losses. We calculated the free space loss over a 1km distance using Equation 2.

$$20 \log_{10} \left(\frac{4d}{\lambda} \right) \tag{2}$$

Substituting 1km for "d" and the wavelength of our 915Mhz signal of 33cm for λ, we can see that the minimum free space loss we faced was 91.6139 dB. Since our communication was successful, the remaining budget of roughly 75dB for losses due to other obstructions was sufficient for communication over 1km.

Appendix D Ethics and Safety

1. Our project is designed to interact with nature and stay outdoors for extended periods of time. It is important that the contents of our tracker nodes, including lithium ion batteries, are well-contained in order to avoid pollution or the harming of animals. In any further development, we will ensure that our batteries, PCBs, and LoRa modules are encapsulated safely in a container so that it can be deployed and then recovered in a way that leaves no trace on the environment in which it was stationed. By doing so, we are doing our best to comply with the IEEE code of ethics that calls for "ethical design and sustainable development practices" [8].

2. When using radio frequency transmission, there are a number of issues that can arise such as interference with other signals. When designing our networking module, we made sure to transmit at an unrestricted frequency and acting in full compliance with the FCC. Our project's use case is to be deployed in relatively remote areas, so the risk of interfering with outside signals is low.

3. When working on our project, we have and will continue to abide by the IEEE code of ethics [8] by being open to criticism of our work from our teammates, TA's, and professors. We organized set meeting times throughout the week designed to keep ourselves up to date on the progress made by our teammates and created a shared Gitlab project to host our code in an effort to maintain technical transparency. We emphasized the importance of being ready to pivot and embrace different design parameters and restrictions if our research turned up ethically-binding reasons to do so. Before starting work on our project, we completed lab safety tutorials and received nominal training in the areas of PCB design and soldering to make sure that we are undertaking only the tasks for which we are qualified. Finally, while working in a group, we treated each other fairly and respectfully, culturing an environment that welcomes the exchange of ideas and promotes productive, enjoyable work.

Appendix E PCB Design

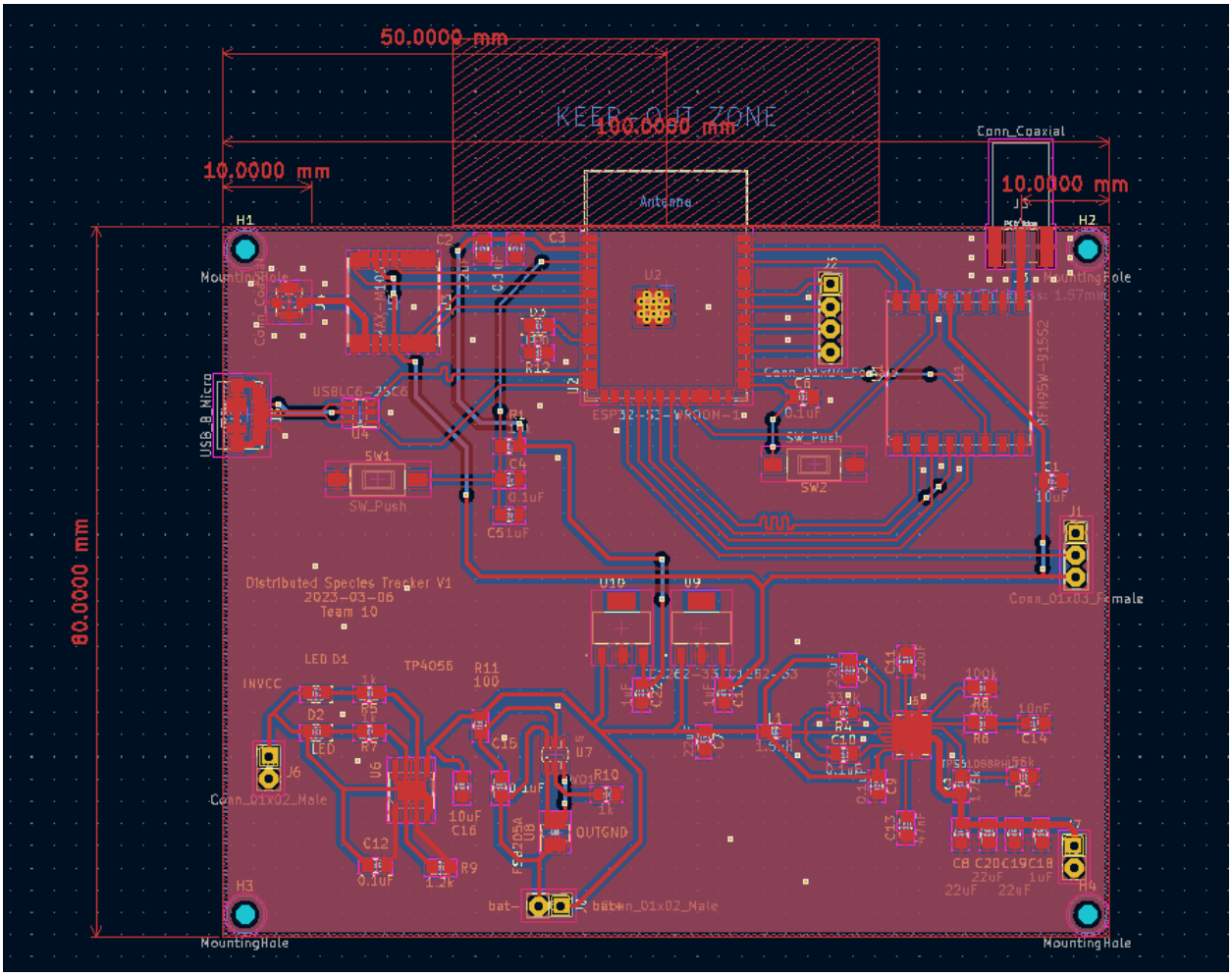

Figure 10: PCB Design

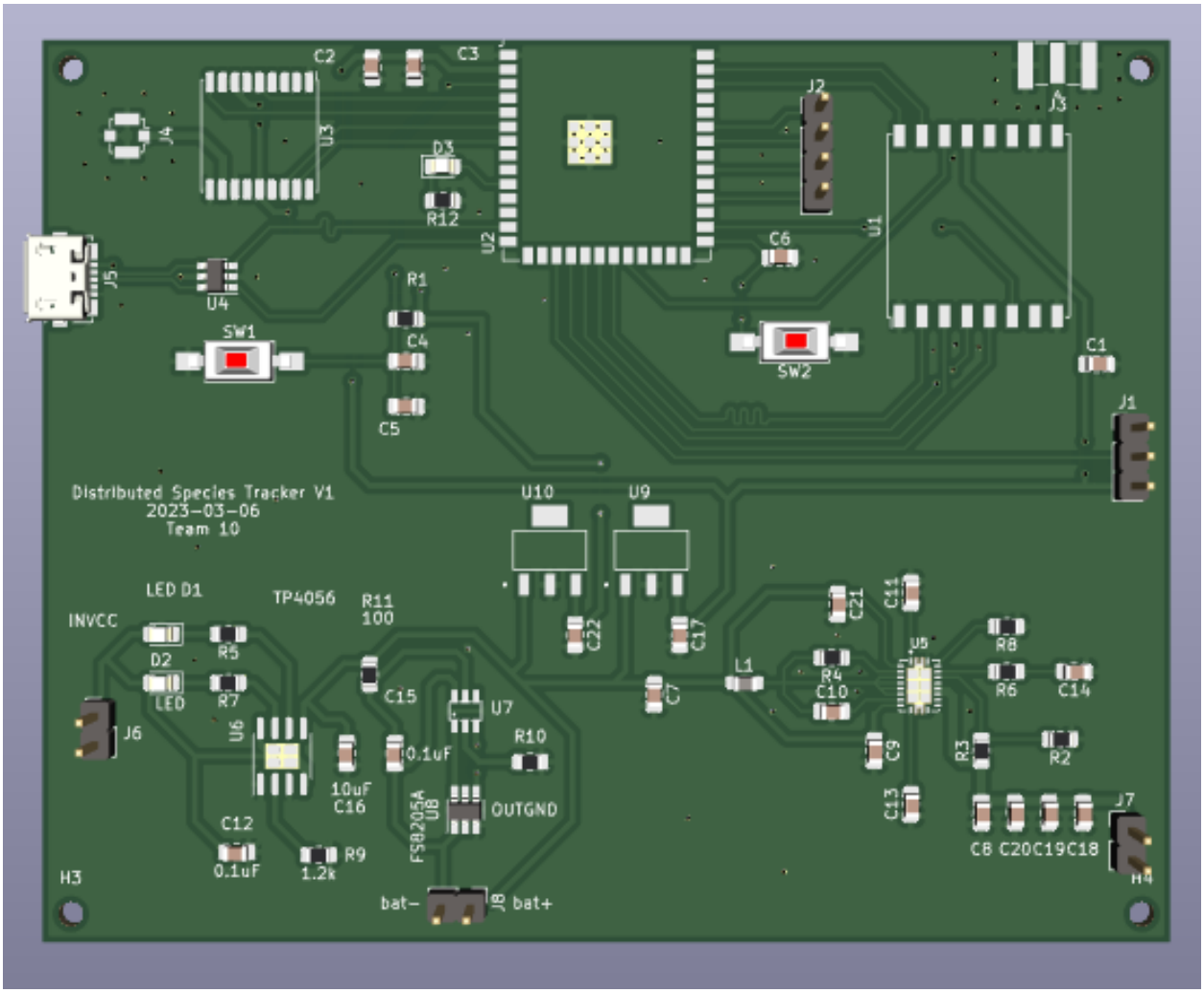

Figure 11: PCB Model

Appendix F PCB Schematic

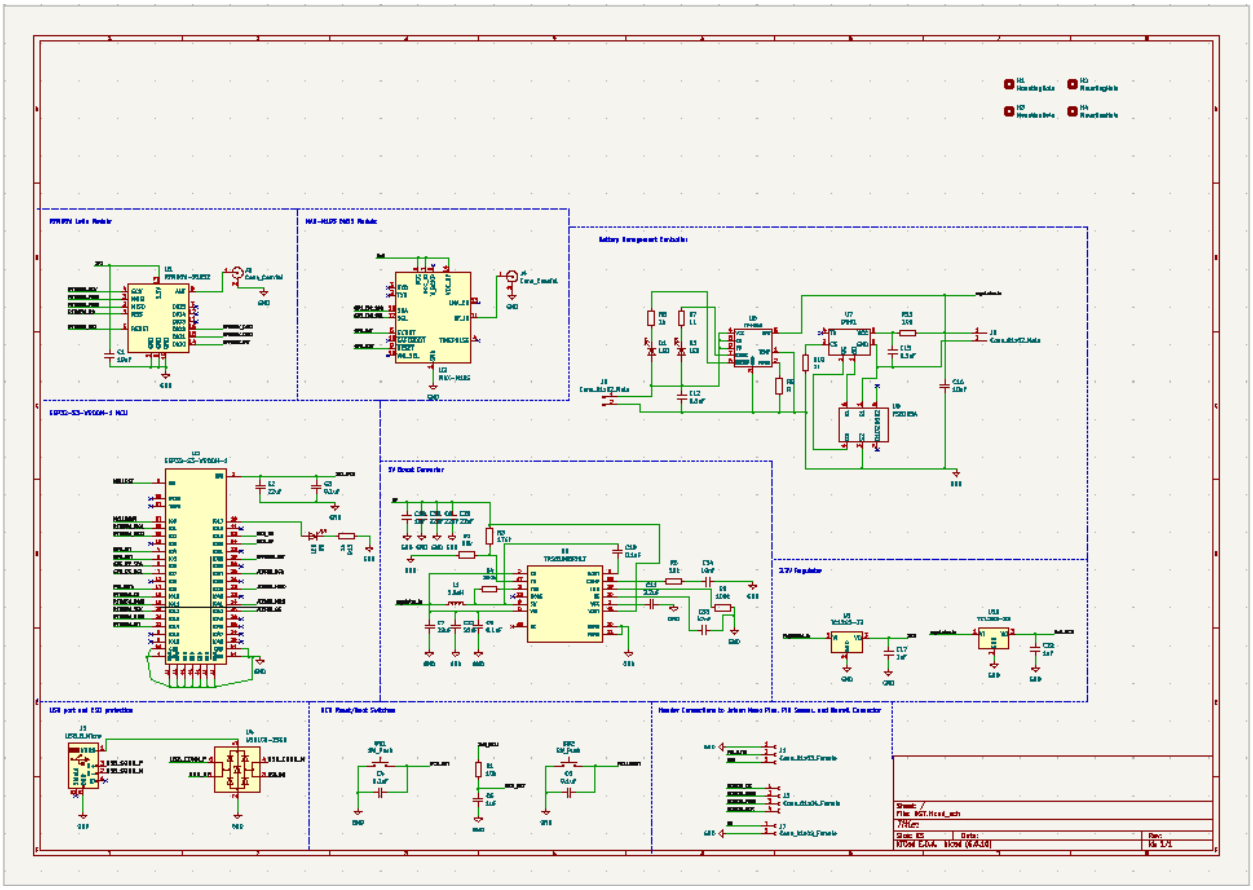

Figure 12: PCB Schematic