The Smart Bird Feeder

EEL 4915 | Senior Design II | December 7, Fall 2021 | Group 7

Paul Amoruso
Computer Engineering
University of Central Florida

John Hauff
Computer Engineering
University of Central Florida

Nikki Marrow
Electrical Engineering
University of Central Florida

Matthew Wilkinson
Computer Engineering
University of Central Florida

Table Of Contents

1. Executive Summary

As people try to reconnect with nature, many people find watching and interacting with wild, free-spirited birds as a form of relaxing entertainment. However, where there are birds, there are often squirrels. Squirrels are constantly scavenging for food, and the number of squirrels that populate areas that humans inhabit grows each day. Thus, squirrels are often considered one of the top pests in the backyards, gardens, and any places where food and trees exist. This leads us into the issue we have aimed to fix. Due to the similar diets birds and squirrels have, squirrels will often steal bird food that is given by humans to entice bird activity. Thus, because of the skittish nature of birds, squirrels will often scare off the wild birds by approaching and snatching most if not all of the food before the birds are able to. In response to this, our team's goal for this project was to provide a method of feeding wild birds without pesky squirrels. People have been inventing bird feeders that try to prevent squirrels from doing this for a long time. However, they often have relatively annoying constraints on the end users by their required shape, design, or placement to be remotely effective. On the other hand, several bird feeders have been invented that incorporate modern-day technology to improve the traditional bird feeder. Moreover, some popular modern bird feeders have made attempts at creating devices that can recognize bird species with an attached camera. Features like notifications, uploaded pictures, and videos, have allowed the bird watchers to be anywhere in the world and still get to watch their backyard birds. However, nearly all of these bird detection feeders lack the ability to defend against squirrels. This presented the opportunity for our group to innovate, using the inspiration from these modern bird feeders, and create a smart bird feeder that not only detects birds for us to look at, but also defends against pesky squirrels from eating all the food and scaring off the visiting birds. However, in our endeavor to develop a Smart Bird Feeder that is able to do all that we aim for it to do, we were faced with a couple constraints. While we address these constraints along with issues that arose later, it is important that throughout our project we keep in mind that a limited amount of time of around seven months to build and test this product prior to presentation is important to note, and that on top of that, we were highly reliant on the animals (birds & squirrels) to interact with the bird feeder to test and document the effectiveness of the product.

The Smart Bird Feeder has been designed with the intention of allowing bird enthusiasts to stay up to date with their feathery friends, while defending birds from squirrels, allowing them to enjoy the bird feeder to its full potential. To realize this idea, our smart bird feeder contains a trained machine learning (ML) model performing object detection from a purchased ML development board. This development board is connected to a camera and a custom designed printed circuit board (PCB) which itself is connected to a speaker to play sound to help scare off invading squirrels, an ultrasonic sensor to keep track of amount of feed left, and a servo motor to close a hatch door which provides a physical barrier between squirrels and the feed. These components also interface with a microcontroller located on the custom PCB. It is important to note that the

utilization of sounds to scare away squirrels is not a one hundred percent effective method, as it is based upon the assumption that the squirrel is not experienced enough to realize that the sound is coming from the speaker and that there is no potential harm that comes from it. However, since squirrels are often underestimated, we suspected that the sound would not be one hundred percent effective, thus, this led us to implement a physical barrier known as the hatch, that closes when squirrels visit, so that we can get to as close as possible to one hundred percent effective. The idea for the project was to have the ML development board detect between bird species and squirrels, then either close the hatch and play a noise through the speaker when a squirrel arrives or take a picture if the required percent accuracy of the bird species is reached. This picture will then be sent to the mobile application through bluetooth or Wi-Fi. The user will then be able to open the app and enjoy a live stream of the camera's view. Our smart bird feeder device also automatically captures a still image of the detected bird and stores the image for easy viewing from the user application, along with a label on the image indicating the species of the bird(s) contained within the image.

The following report documents the design process of The Smart Bird Feeder. It will start off with the motivations, goals, and objectives for this project. It will then proceed to describe the details of the requirement specifications and how our team designed our smart bird feeder device to adhere to these requirements, such as choosing to partially use low power modes in our device for the purpose of limiting battery consumption. Chapter 3, known as the technology investigation chapter, will contain the decisions made for each of the individual parts of the system and the steps that lead to those decisions. Subsequently, the report will dive deep into the design constraints that deal with such things like power consumption and ease of use, and standards set by such organizations as IEEE, that influence the making of our project. This document then moves on to design details that go into discussing and providing visual depictions of such things as the hardware, software, interfaces, and schematic of the overall system. Then, once the design section is adequately discussed, the report will go into the overall integration of the custom PCB into our smart bird feeder device, the design of the PCB, and system testing performed on our smart bird feeder device. Afterwards, the paper will finish off with an administration section followed by an overall conclusion.

2. Project Description

In this section, we dive deeper into the project from the initial inspiration to a basic design of what we had thought the application may look like. In other words, the intention for this section is so that the project may be further explained and introduced, in a short and to-the-point manner that is intended to be relatively easy to comprehend and explain our intentions moving forward in the project. Thus, this section starts off with a brief introduction and motivation to shed light on how the project came to be. Then we move on to describe our group goals and objectives before diving deep into the requirements specifications that help describe requirements and implementations for such things as power, hardware, mobile application, and more. Following this, we begin providing visual representations of some aspects of the project, starting with the House of Quality to help distinguish and relate our project's engineering requirements and marketing specifications. Then, a software block diagram and hardware block diagram are provided to show various aspects that will go into the project, and who is leading the team on each individual aspect of this project. In subsection 2.8, there is a simple drawn sketch of the most basic features of the mobile application to describe a very brief overview and representation of the important pages of our user application, such as the login and main page of the application. Then, finally, subsection 2.9 outlines the main steps to operate the smart bird feeder device.

2.1 Introduction

The purpose of this project is to design and develop a solution for anyone who enjoys feeding backyard birds, without the hassle of squirrels eating the food set out for birds in their backyard. This project will incorporate designs for many technical features. This includes a computer vision machine learning model designed to identify bird species and squirrels, and a method of preventing squirrels from staying at the feeder which involves using a speaker and automatic door. In addition to the features mentioned, the bird feeder will incorporate novel features such as image captures of the visiting birds provided in a mobile application. This will let the user easily view the pictures, and also check the amount of food left in the feeder.

2.2 Motivation

The idea to create a smart bird feeder stems from one of our team members' hobbies, which we have found extends to a large market that’s rife for innovation. There is a large majority of the population that enjoys bird feeding; zoos often have attractions dedicated to the task, it’s commonly seen in media, and most of the time, bird feeding carries with it a positive societal connotation. The idea of creating a personal bird feeding station right in a backyard is a great way to enjoy the hobby in the comfort of one's home. However, as with many hobbies, in order to maximize the potential benefits of a bird feeder, one must devote time and resources into the hobby.

Our team found, as many others have, that it is challenging to devote enough time towards the hobby to feel properly rewarded by it. A person could spend an indefinite amount of time waiting and watching the feeder while nothing happens, which is time that could be spent on another hobby or performing some other more essential task. Due to these time constraints on a user, they are likely to miss the birds feeding from time to time. Our team has set out to solve this issue by creating a bird feeder that provides the level of satisfaction that the user would be rewarded with if they always had time to watch the feeder. The team started by considering how to create this dynamic and found that implementing a streaming and photo system would ensure the user always had access to the bird feeding moments.

This idea to extend the typical bird feeder’s potential began to grow the more the group investigated current bird feeding systems. The ability to track the species of the bird feeding is a key feature that spoke to our group, as tracking the different types of birds visiting the feeder is an exciting part of the hobby. This feature would also open other avenues in scalability for us or others to pursue in the future, as a notifications system tied to a user application could be extended in many ways, such as only notifying the user when the type of bird they’re interested in is feeding.

A group member also noticed a significant problem when trying to feed birds in the past. Squirrels acted as a threat to the hobby, constantly stealing the food and scaring the birds away. Our group hypothesized that by using the machine learning species detection technology, we could equip the bird feeder with a technology to detect unwanted animals, such as squirrels, preying on the bird feed, and a chance to maximize the potential of the bird feeder. This detection system could provide us with the potential to create a bird feeder which would only allow birds to eat the bird feed. With this, our motivations began to align with this train of thought. If our group could find a way to deter these predators, then the user would not need to concern themselves with other animals feeding from their bird feeder and could focus on enjoying their bird watching activities.

We noticed that other bird feeders on the market were not considering all of these aspects at once. Overall, our group holds a strong motivation to automate and maximize the potential of the bird feeding and bird watching process and consider our product to be rife with potential based on the current marketplace, giving the user the choice to take an active or more passive role in their bird watching hobby.

2.3 Goals and Objectives

Our group’s motivation to maximize the potential of the bird feeder in such a way where the user needs to devote as little time as possible to actively bird watching themselves, while still achieving maximum results, led us to develop several concrete goals and objectives to achieve this potential. The primary consideration was how to ensure autonomous functionality with the bird feeder. To do this, the bird feeder is equipped with a device that uses machine learning with object detection to be capable of determining the species of bird that's currently feeding, as well as any squirrels that may appear to unwantedly prey on the bird feed. The process should be automatic, in which case The Smart Bird Feeder uses HTTP communication over WiFi to send signals to an end-user's application that has ties to the Smart Bird Feeder. This will alert the user to any feeding event, allowing them to obtain immediate satisfaction without having to watch the bird feeder all the time. The camera also provides a user who can't view the event directly with high quality images that are stored on the bird feeder application. The bird feeder application should be seamless and efficient, providing the user with easy access to any of the streams or image captures that have taken place.

In order to save time and the hassle of checking everyday to see if the bird food has run out, the product will alert the user to when the feed is low. By incorporating an ultrasonic sensor and connecting it to the processor, the processor can then send a signal to the application when the bird feed has passed a set threshold. The same thought process can be incorporated by setting a threshold for the battery. The user should receive a notification upon a low battery alert so they're aware when they have to charge the device. If added onto the system as a stretch goal, given that our team would like to limit the input time from the user, a solar panel should provide strong support for the battery to make the battery last as long as possible between charges.

To actively deter squirrels from the feeder, which will promote more bird feeding time, the device will be equipped with a hatch door that will close the access to the food when it detects a squirrel. However, given that the squirrels' presence alone may deter birds, the feeder will also have an alarm that emits a sound that will cause the feeding predators to flee from the feeder. This will ensure the squirrel should have no reason to stick around near the feeder.

Our design goals ensure that we will bring to life a smart bird feeder that is full of innovation and has the potential to find a proper area in the marketplace, all while protecting itself against predators, providing beautiful up close images of visiting birds, and keeping the user informed with the status of the feeder. With this project, we are given the opportunity to satisfy every bird watcher's wish of having a sanctuary for local birds to eat in peace and provide customers a way to experience the process.

2.4 Requirements Specifications

This section will outline the requirements that are highlighted above in a more specific and verifiable manner. By writing the requirements in clear sentences, the group will have a clearer and more organized path to take going forward. This section will also be consulted throughout the design process to ensure we remain focused on completing the objectives outlined below.

2.4.1 General

- A container houses the bird feed, which is protected by a motor-controlled door, and gravity brings the bird feed into the feeding holes at the bottom of the bird-feeder.
- Bird feed is always visible to the birds, even when covered from the squirrels.
- The electronics are housed in a sturdy casing in order to protect from the elements.
- A bird perch adjacent to the feeding holes provides a spot for image capture.

2.4.2 Hardware

- Connected components:
  - A machine learning acceleration device with camera and WiFi module attached.
  - A custom PCB with a motor and speaker attached.
  - Battery bank to power all components.
- A servo, powered by the PCB, is connected to a hatch to rotate and limits unwanted animals' access to feeding zones.
- A camera is placed at a proper vantage point to observe the feeding platform.
- An alarm sound system deters unwanted visitors.
- An ultrasonic sensor provides information on low food level based on the range between the ultrasonic sensor and the bird feed.
- A motion sensor detects movement to activate the machine learning processor.

2.4.3 App and UI Integration

- The mobile application has an easy-to-use user interface, connecting the user with various feeder technologies.
- Notifications
  - Notifications for low feed level.
  - Notifications for birds active at the feeder.
  - Notifications for new images added to the memories gallery.
- Streaming (stretch goal)
  - Access a stream of the bird feeding in real time.
- Memories
  - Save a picture of the bird feeding as a memory.
  - Sort memories by date or bird species.

2.4.4 Machine Learning Bird Species Detection

- Object detection is suspended and inactive until movement is detected at the feeder.
- A machine learning (ML) model is fed camera input and identifies bird species or other feeder predators.
- The ML model identifies squirrel predators, and communicates with the PCB to signal the feeding mode or defense mode.
- The ML dev kit sends a photo to the PCB to transfer to the application.
    - The photo will be the picture with the highest percentage of species confirmation.

Table 1 - Chosen bird species for recognition and their class labels

Bird Species	Object Detection Class Label
American Crow	american-crow
Blue Jay	blue-jay
Blue-gray Gnatcatcher	blue-gray-gnatcatcher
Carolina Wren	carolina-wren
Common Grackle	common-grackle
Downy Woodpecker	downy-woodpecker
Gray Catbird	gray-catbird
Green-cheeked Parakeet	green-cheeked-parakeet
Mourning Dove	mourning-dove
Northern Cardinal	cardinal
Northern Mockingbird	northern-mockingbird
Palm Warbler	palm-warbler
Pileated Woodpecker	pileated-woodpecker
Red-bellied Woodpecker	red-bellied-woodpecker
Tufted Titmouse	tufted-titmouse
Yellow-rumped Warbler	yellow-rumped-warbler

The amount of time granted to our team for the development of our smart bird feeder device put a heavy potential constraint on our device's object detection capabilities. Given that the amount of time required to train a machine learning (ML) model tends to increase as the number of objects the model is trained to classify increases, our team had to carefully consider which bird species to train our ML model to detect. Our team's choice was based on enabling our smart bird feeder to detect and capture images of the most commonly found birds in the backyards of Florida (according to online sources [23]), given the time available to develop this device. Table 1 shows the bird species that we chose to train our smart bird feeder to detect, along with the species labels that the ML model uses when it classifies the object detected in the live camera feed. These bird species are the fifteen most common bird species found in Florida backyards and are the bird species that our team required our smart bird feeder device to recognize, capture images of, and feed.

In selecting our bird choice, we thought it to be logical to factor in the location in which the project was being developed into our decision. Therefore, our team decided that detecting the most commonly found bird species in Florida backyards was the most natural choice. Our team is certainly considering expansion past this list of bird species, however, if time permits. Our team will also make efforts to ensure that the machine learning model is scalable, allowing for growth and adaptation of the ML model to incorporate detection of additional bird species or other objects if so desired. The team is also prepared in the event a bird is not recognized, as the machine learning model will still be able to sense whether or not the creature is a bird of an unknown species. Under these circumstances, the device will tell the user that this bird is of an unknown species to the device, and more updates will be coming to identify more species. The functionality of the bird feeder will still provide users with a livestream, a picture, and deterrent against squirrels.

2.4.5 WiFi Connectivity

- The Jetson Nano connects to WiFi
    - Using *IEEE 802.11* standard.
    - Uses antennae to boost WiFi signal range.

2.4.6 Power

- Powered by a rechargeable battery.
- Battery is able to be charged while in use.

2.5 House of Quality

Tables 2.1 and 3.1 describe the numerous engineering requirements which are relevant to this project with various relevant marketing requirements. The three columns of these tables share an engineering requirement, a marketing requirement that relates to the adjacent engineering requirement, and a justification for the engineering requirement's existence in this project. Highlighted in Tables 2.1 and 3.1 are the three project requirements that our team plans to demonstrate with our device in Senior Design 2. The first requirement is for the machine learning (ML) model of The Smart Bird Feeder to have at least a 70% confidence level that an object is either a bird or a squirrel. This accuracy level should be indicated along with a bounding box on the live video feed from the smart feeder's camera. The second requirement for our smart bird feeder which we plan to demonstrate is the speed at which the hatch that will shield the bird feed from predatory squirrels will shut itself whenever there is a squirrel detected nearby, where the camera on the smart bird feeder device will be the primary component for detecting nearby squirrels. The third requirement that we will demonstrate is that a given user will have access to at least 1 GB of storage space for their bird images within their smart bird feeder application.

Below this table, the described requirements are then related to one another in Figure 1, a product planning matrix known as The House of Quality (HOQ). This figure is a tool designed to help groups like ours in establishing a visual representation between the engineering requirements mentioned and marketing specifications. We set our marketing requirements based on the wants and needs that most potential customers in the market for bird feeders would have. It is important to note that many of the engineering requirements are derived from the marketing requirements initially set in place by these common wants and needs. Obviously, a common requirement is to have a low maintenance system. Thus, two of the engineering requirements deal with the longevity and charge speed of the power supply.

We can see how the House of Quality displays the relationship between marketing and engineering, by looking at the correlation between cost and battery life. Customers will often tend to desire a low cost item, however, increasing the battery life will increase the cost with it, thus it is evident that where cost and battery life intersect, there is a strong negative correlation. It is important to also note that while consumer desires play an important role in the requirements, the engineering requirements that list the needs of our system (or any system) must also be from the viewpoint of the developer/engineers of different engineering aspects. This can easily be seen in how cost, ease-of-installation, and battery life affect one another. While smaller dimensions correlate to lower battery life, we must also ensure that we have enough power to supply to the electronic devices, and meet our minimum baseline requirement.

Table 2 - Marketing and engineering requirements

Marketing Requirements	Engineering Requirements	Justification
3	1. Accuracy of bird species and squirrel detection must exceed 90% capture images.	Customer expectations based on similar products are high.
1	2. The dimensions should not exceed 8" x 8" x 14".	The entire system must be hung or mounted and remain stable without falling.
5	3. The system should have enough power to run for at least 2 hours.	The system should have enough battery life to be used practically.
5	4. Duration of external battery recharge should not exceed 6 hours.	The system recharge time should not exceed a night cycle and should be ready by the next day or sooner.
7	5. The speaker should have an output range of 60-95 dB.	Speaker sound intensity should be at a safe but effective range for deterring squirrels.
7	6. Hatch door closing time should not exceed 3 seconds.	The door should close quickly enough to prevent squirrels' access to feed
4	7. Users should receive a notification of a new bird memory in 10 seconds or less.	Long delays in notifying users of birds at the feeder would diminish the quality of the bird-spotting experience.

Table 3 - Marketing and engineering requirements

Marketing Requirements
1. The device should have a compact and easy-to-install design.
2. The device should have a low cost.
3. The object detection for birds and squirrels should be reliable.
4. Images should be of high quality.
5. The device should be low-maintenance.
6. The device should be durable against weather and wildlife.
7. The device should not be harmful to its environment or any wildlife.

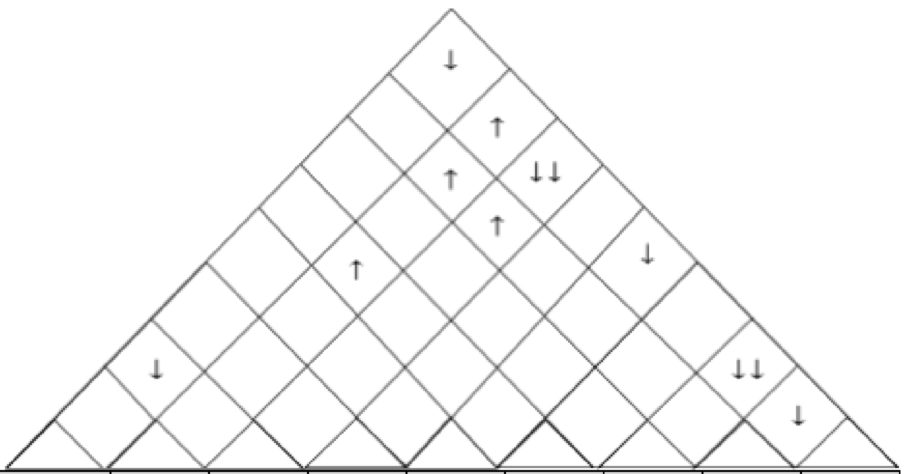
Table 2.1 - Marketing and engineering requirements *(continued)*

Marketing Requirements	Engineering Requirements	Justification
3	8. Accuracy of bird species and squirrel detection must exceed 90% capture images.	Customer expectations based on similar products are high.
1	9. The dimensions should not exceed 8" x 8" x 14".	The entire system must be hung or mounted and remain stable without falling.
5	10. The system should have enough power to run for at least 2 hours.	The system should have enough battery life to be used practically.
5	11. Duration of external battery recharge should not exceed 6 hours.	The system recharge time should not exceed a night cycle and should be ready by the next day or sooner.
7	12. The speaker should have an output range of 60-95 dB.	Speaker sound intensity should be at a safe but effective range for deterring squirrels.
7	13. Hatch door closing time should not exceed 3 seconds.	The door should close quickly enough to prevent squirrels' access to feed
4	14. Users should receive a notification of a new bird memory in 10 seconds or less.	Long delays in notifying users of birds at the feeder would diminish the quality of the bird-spotting experience.

Table 3.1 - Marketing and engineering requirements *(continued)*

Marketing Requirements

8. The device should have a compact and easy-to-install design.
9. The device should have a low cost.
10. The object detection for birds and squirrels should be reliable.
11. Images should be of high quality.
12. The device should be low-maintenance.
13. The device should be durable against weather and wildlife.
14. The device should not be harmful to its environment or any wildlife.



		Bird/Squirrel Recognition Accuracy	Dimensions	Battery Life	External Battery Recharge Time	Speaker Sound Frequency	Door Mechanism Speed	Quantity of Storage Per User in App	Weight	Cost
		+	-	+	-	+	+	+	-	-
Installation Ease	+		↑		↑↑				↑↑	
Cost	-		↑	↓↓	↓↓	↓	↓	↓↓	↓	↑↑
High Performance	+	↑		↓↓			↑↑			↓
High Image Quality	+	↑		↓				↑↑		↓↓
Low-maintenance	-		↑	↑↑	↑↑	↑↑			↑↑	↑↑
Durability	+			↓		↓	↓		↑	↓↓
Safety of Wildlife and Environment	+	↑	↑			↓↓	↑		↑	↓
Targets for Engineering Requirements		> 70%	≤ 8" x 8" x 14"	≥ 20000 mAh	≤ 6 hours	60 - 90 dB	≤ 3 seconds	≥ 1 GB	≤ 15 lbs	≤ $300

Legend	
+	Increasing this requirement increases the desirability of the device.
-	Decreasing this requirement increases the desirability of the device.
↑	Positive correlation.
↑↑	Strong positive correlation.
↓	Negative correlation.
↓↓	Strong negative correlation.

Figure 1 - House of Quality

2.5.SD2 House of Quality

In Senior Design 2, the group changed the engineering requirements to be more relevant and testable. We found that a few of the engineering requirements listed before were not backed up by reasoning and lacked clear meaning.

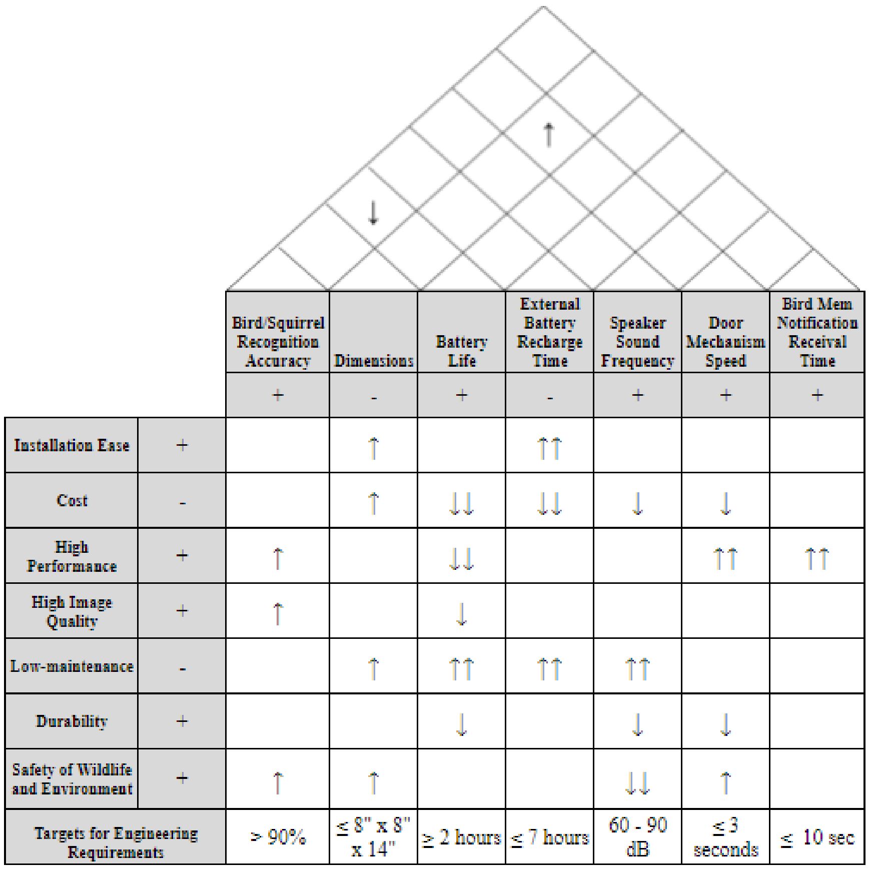
For instance, we noticed that requirement 3, *"Life of the power bank should be at or exceeding 20000 mAh"*, was attempting to ensure our device could run for a certain amount of time, but as we didn't yet fully know the current consumption, it was rather arbitrary. This requirement now reads "The system should have enough power to run for a certain amount of time" with a specification of at least 8 hours. This clarification is more aligned with our goals. The accuracy regarding bird species detection in requirement 1 was raised from 70% to 90% as further testing ensured we could aim for a higher percentage. Requirement 7 reads *"Users may store a minimum of 1 GB of bird images on their app"* but we found this number to be arbitrary and not very demonstrable. Instead this requirement was shifted to focus on the speed in which a user is notified of a bird memory, *"User will be notified of a new bird memory within a certain time period"*.

Table 2.SD2 - Updated engineering requirements

Engineering Requirement	Specification
Accuracy of bird species and squirrel detection must exceed a certain confidence level to capture images.	≥ 90%
The dimensions should not exceed the size of a common bird feeder.	≤ (8" x 8" x 14")
The system should have enough power to run for a certain amount of time.	≥ 2 hours
Duration of external battery recharge should not exceed a certain time.	≤ 7 hours
The alarm speaker should have a certain output range.	60-95 dB
Hatch door closing time should close within a certain amount of time.	≤ 3 seconds
Users will be notified of a new bird memory within a certain time period.	≤ 10 seconds

Some of the requirements were seen as excess and cut entirely. Requirement 8 and 9 read "The system should not exceed a weight of 15 pounds." and "The cost of all materials should not exceed $300 (not including replacement materials)." The weight requirement was arbitrary and unlike the size of the bird feeder requirement, wasn't based on the common size of bird feeders. The cost of the feeder was raised higher than the threshold listed in SD1 as products were added to meet functionality requirements, and is no longer as competitive as we had hoped, so raising it does not seem appropriate as there isn't as much reasoning to justify its existence. This is a design reality the team has learned a lesson from.

The new engineering requirements are listed in Table 2.SD2, with demonstrable requirements highlighted. An updated House of Quality is included in Figure 1.SD2.



		Bird/Squirrel Recognition Accuracy	Dimensions	Battery Life	External Battery Recharge Time	Speaker Sound Frequency	Door Mechanism Speed	Bird Mem Notification Receival Time
		+	-	+	-	+	+	+
Installation Ease	+		↑		↑↑			
Cost	-		↑	↓↓	↓↓	↓	↓	
High Performance	+	↑		↓↓			↑↑	↑↑
High Image Quality	+	↑		↓				
Low-maintenance	-		↑	↑↑	↑↑	↑↑		
Durability	+			↓		↓	↓	
Safety of Wildlife and Environment	+	↑	↑			↓↓	↑	
Targets for Engineering Requirements		> 90%	≤ 8" x 8" x 14"	≥ 2 hours	≤ 7 hours	60 - 90 dB	≤ 3 seconds	≤ 10 sec

Legend	
+	Increasing this requirement increases the desirability of the device.
-	Decreasing this requirement increases the desirability of the device.
↑	Positive correlation.
↑↑	Strong positive correlation.
↓	Negative correlation.
↓↓	Strong negative correlation.

Figure 1.SD2 - Updated House of Quality

2.6 Software Block Diagram

Figure 2 shows a block diagram developed and designed by our group to effectively provide a visual description of the software aspects that will make up the project, and how it all integrates with each other to create a complete system. With this said, it is clearly evident that the software we are working on is divided into three integral parts. Considering these three parts of the software block diagram, only the mobile application sector will contain the necessary user interface to our consumers. This diagram displays how the main computation that creates the backbone of our project is under the AI Dev Kit software block. Due to our project's hardware and software demands and the skillset of our group members, we organized ourselves into certain blocks based on each team member's various strengths and their confidence in a particular area. Therefore, Matthew Wilkinson pioneered the mobile app development and design, John Hauff led the team on object detection, and Paul Amoruso was in charge of additional code on the PCB and Jetson Nano. Taking a look at the custom PCB Software block, it is relevant to note that Nikki Marrow has only one block under her in the software diagram. This is due to her limited experience in software, and increased workload in the hardware space. Unlike the next diagram on hardware that has statuses on each block, the software diagram only contains statuses on the three main software parts.

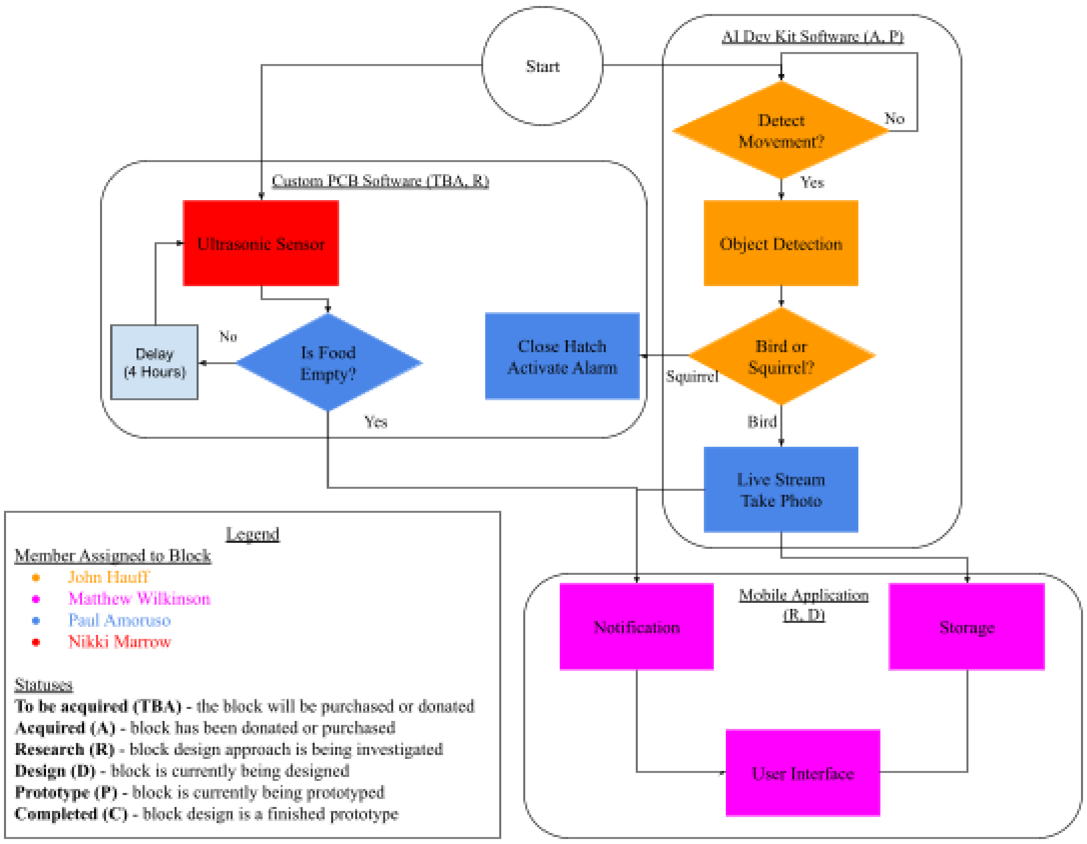

Start
AI Dev Kit Software (A, P)
Detect Movement?
No
Yes
Object Detection
Bird or Squirrel?
Squirrel
Bird
Live Stream Take Photo
Custom PCB Software (TBA, R)
Ultrasonic Sensor
Delay (4 Hours)
No
Is Food Empty?
Yes
Close Hatch Activate Alarm
Mobile Application (R, D)
Notification
Storage
User Interface
Legend
Member Assigned to Block
John Hauff
Matthew Wilkinson
Paul Amoruso
Nikki Marrow
Statuses
To be acquired (TBA) - the block will be purchased or donated
Acquired (A) - block has been donated or purchased
Research (R) - block design approach is being investigated
Design (D) - block is currently being designed
Prototype (P) - block is currently being prototyped
Completed (C) - block design is a finished prototype

Figure 2 - Software block diagram

2.6.SD2 Software Block Diagram

In Senior Design 2, the group came to a better understanding of what the breakdown for the software components would be and the roles we played. Shown below in Figure 2.SD2 is the updated software block diagram. The new diagram better segments the various components of the device's design. For example, the frontend work on React Native and backend work on Express is clearly labeled and more specific to each task. John Hauff and Paul Amoruso split responsibility of the programming tasks for the Jetson Nano, while Nikki Marrow focused on the embedded software on the MCU on our PCB. John Hauff and Matthew Wilkinson also worked together on the development of the mobile application.

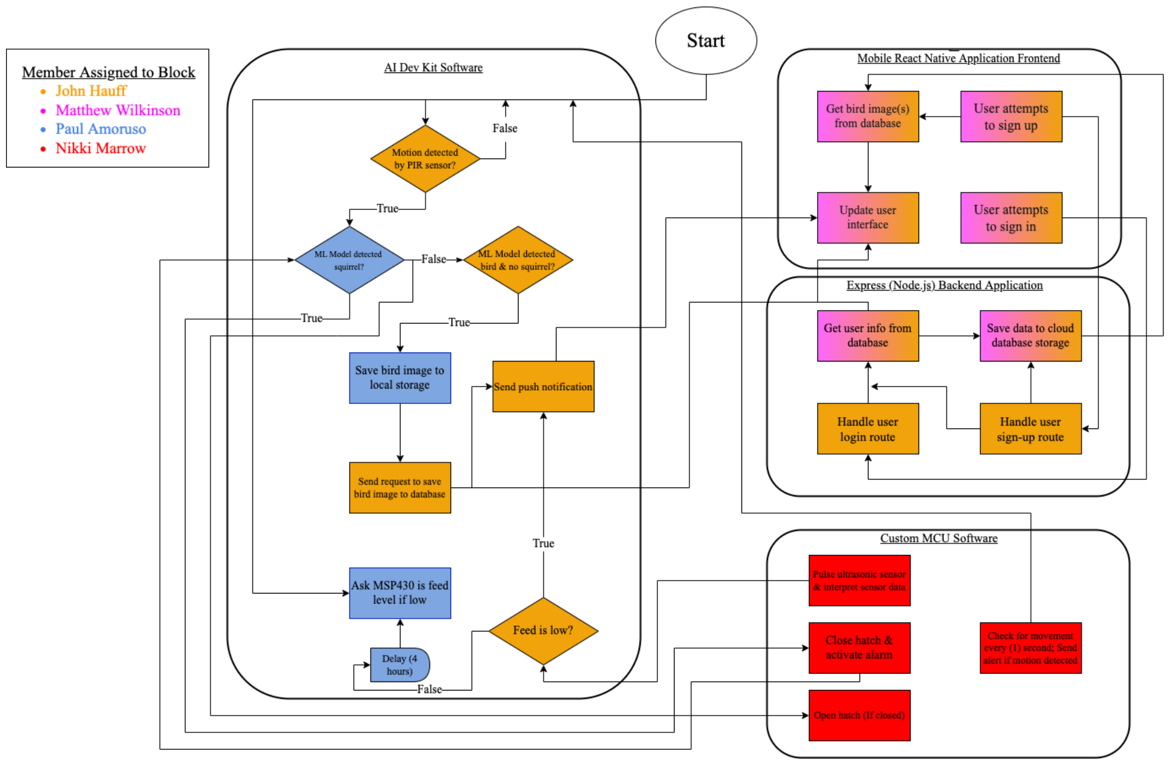
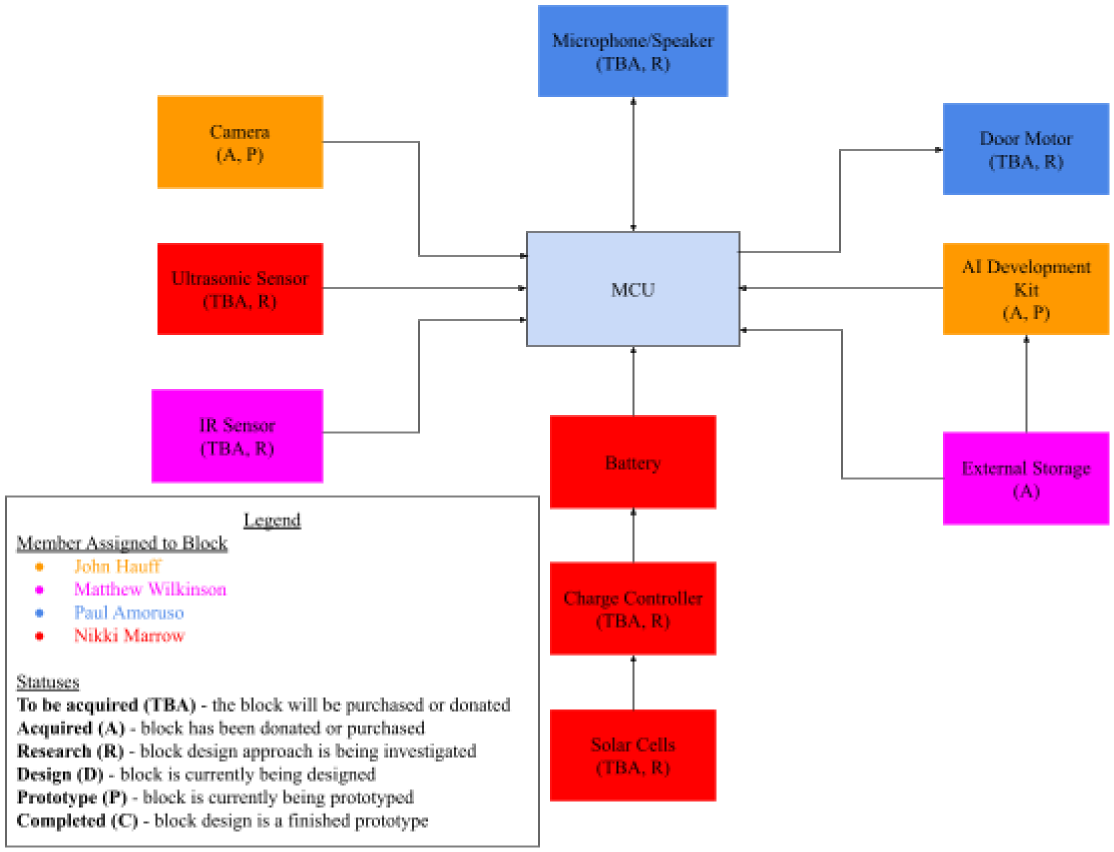

Member Assigned to Block
John Hauff
Matthew Wilkinson
Paul Amoruso
Nikki Marrow
Start
AI Dev Kit Software
False
Motion detected by PIR sensor?
True
ML Model detected squirrel?
False
ML Model detected bird & no squirrel?
True
True
Save bird image to local storage
Send push notification
Send request to save bird image to database
True
Ask MSP430 is feed level if low
Feed is low?
Delay (4 hours)
False
Mobile React Native Application Frontend
Get bird image(s) from database
User attempts to sign up
Update user interface
User attempts to sign in
Express (Node.js) Backend Application
Get user info from database
Save data to cloud database storage
Handle user login route
Handle user sign-up route
Custom MCU Software
Pulse ultrasonic sensor & interpret sensor data
Close hatch & activate alarm
Check for movement every (1) second; Send alert if motion detected
Open hatch (If closed)

Figure 2.SD2 - Updated software block diagram

2.7 Hardware Block Diagram

Figure 3 shows a block diagram that was also designed by our group to adequately provide a visual description of the hardware aspects that will go into the project, and who is leading the team on the individual hardware aspect. With this said, it is clearly evident that the hardware diagram is noticeably simpler than the previous software diagram. In this diagram, the center has the microcontroller with the necessary hardware parts that go along with the project either pointing to it or away from it, to say which way the necessary information/signal is going. This diagram also displays how individual hardware pieces interact with each other, such as how the external storage will directly interact with both the MCU and AI development kit. Similarly with the software, we assigned each other to certain blocks based on their confidence in the particular area. Therefore, Nikki Marrow received the most amount of hardware blocks for her experience in the hardware space, while John Hauff, Paul Amoruso, and Matthew Wilkinson each led two hardware blocks based on the experience and excitement over the particular device. In addition to the members assigned, we also add statuses on where each block is at the given moment. Thus, readers are able to look at our diagram to determine things such as whether or not the particular hardware device is in our possession and if we have completed a prototype of it.



Microphone/Speaker (TBA, R)
Camera (A, P)
Door Motor (TBA, R)
Ultrasonic Sensor (TBA, R)
MCU
AI Development Kit (A, P)
IR Sensor (TBA, R)
Battery
External Storage (A)
Charge Controller (TBA, R)
Solar Cells (TBA, R)
Legend
Member Assigned to Block
● John Hauff
● Matthew Wilkinson
● Paul Amoruso
● Nikki Marrow
Statuses
To be acquired (TBA) - the block will be purchased or donated
Acquired (A) - block has been donated or purchased
Research (R) - block design approach is being investigated
Design (D) - block is currently being designed
Prototype (P) - block is currently being prototyped
Completed (C) - block design is a finished prototype

Figure 3 - Hardware block diagram

2.7.SD2 Hardware Block Diagram

In Senior Design II the hardware block diagram was also updated to reflect our changes to the product and shifted to better reflect work input. Components were removed from the design all together, such as the solar cells and charge controller that would help to extend the battery life. The original design included some sort of IR sensor to monitor movement. In Senior Design II, we realized that using a passive infrared sensor was a great fit for our device, and so Figure 3.SD2 reflects this update. The updated diagram is shown below.

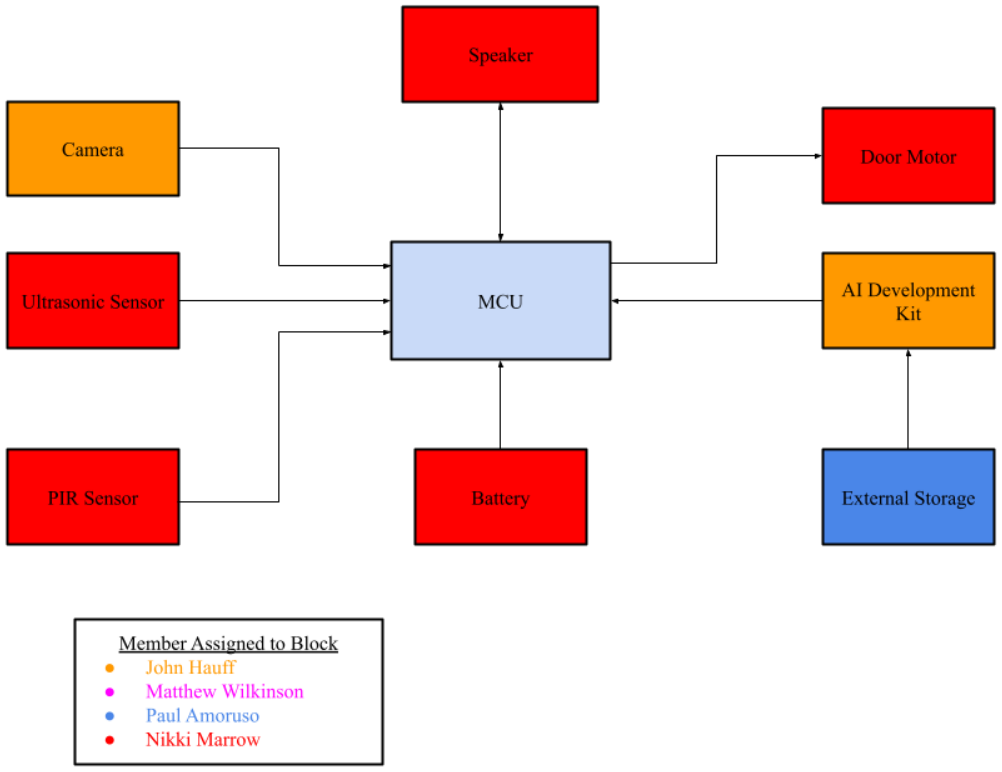


Figure 3.SD2 - Updated hardware block diagram

2.8 Mobile Application Design Diagram

The mobile application design diagram below is used to show a basic idea for what the mobile app will look like. The diagram shown in Figure 4 was purposely left simple, so that it can stand as an open canvas to which we could add additional aspects and features. With this being said, the core features this app will have is a login screen, notifications that will lead the user to a live stream, and a library of bird images showing pictures taken by the bird feeder. The idea for the notifications is that it will notify our phones when a bird has spent a substantial amount of time at the bird feeder so that a user is able to bird watch in real time from the comfort of their living space, in addition to being able to view a library of all the captured images. All in all, the design aims to be simplistic and easy to use, while providing a well rounded app that lets you interact and keep up with any notifications (such as low food) to allow you to feel connected and up to date on the status of the bird feeder.

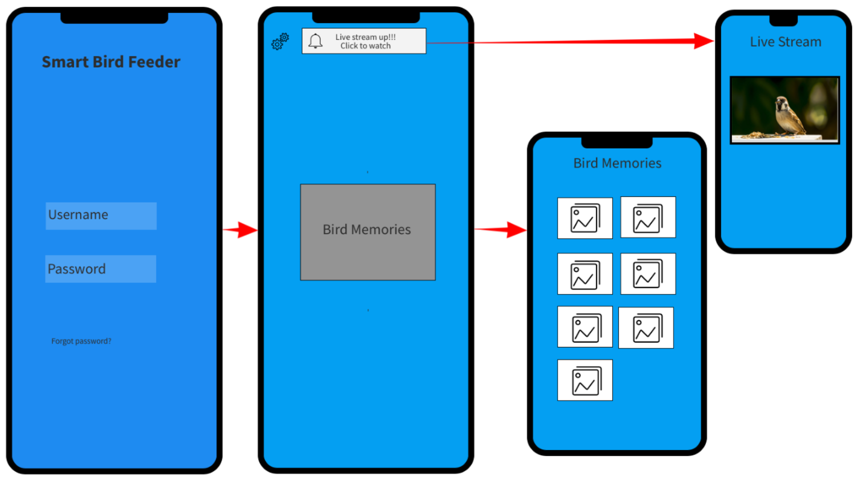


Figure 4 - Mobile application initial design diagram

Building on what was said and the visual description provided, the initial idea for the mobile application will have roughly four pages, shown in Figure 4. The first page will be the login/signup page, where the user can either create an account for the application, or login with an existing account. Then the user may navigate to the landing page after logging in or signing up. From the landing page, we aim to have an option available to navigate to a bird memories page which is essentially going to be a library for bird images taken by the bird feeder when the accuracy of the bird is very high, so the user can look back at all high accuracy labeled birds that have visited their Smart Bird Feeder. We also aim to have the user application contain the live stream for users to watch the birds eat the bird food at the Smart Bird Feeder when a user is not there to physically observe the bird feeder or if they do not wish to frighten the birds.

2.8.SD2 Mobile Application Design Diagram

The design for the mobile application transformed over time, and so it is not exactly the same as what is shown in Figure 4. Live streaming was a stretch goal that has not been met yet, and the application has a tabular screen layout, rather than a stacked screen layout.

2.9 User Guide Instructions

The Smart Bird Feeder will include setup instructions and a guide on how to use the device. In order for the installation to be as seamless as possible for not only during the demonstration of the prototype, but for the potential future end user. It is highly important that there is a proper list that contains simple instructions to follow. Therefore, in the design and construction of our product, we specifically aimed for ensuring various aspects of our project are as user friendly as possible. For instance, when designing the mobile application, we decided to keep the app as simple as possible and just provide the necessary options, such as connecting the device to wireless network, image library (bird memories), and live view. The first impression is often said to be the most important, thus, our group has taken that to heart as we decided the build for our product so that the first impression of the product during installation is as best as relatively possible.

Thus, in discussing all the steps that the end user needs to follow for a successful setup, we were able to narrow it down to nine relatively simple steps to follow. Therefore, we as a group recommend that for the best user experience, follow these guidelines.

1. Download the “Smart Bird Feeder” application from an app store.

2. After the application has finished downloading, access the application and click the “Sign up” button. Enter your email and personal information and click “Create Account”.

3. You will then be prompted with an email to confirm your account, click the application link and you will be directed to a page to enter your password, then fully sign into the application.

4. Plug in the Smart Bird Feeder using a USB cord and wait for the feeder to charge.

5. To connect the Smart Bird Feeder to your WiFi network, Select settings on the application, then click “Smart Bird Feeder” and input the WiFi networks password to connect.

6. Place the Smart Bird Feeder in its desired location (preferably a location with sunlight) and fill the feeding chamber with bird feed.

7. To view a livestream of a feeding event, click “View Feeding” when promoted with a notification.

8. To view pictures of previous feedings, click the “Memories” page.

9. Enjoy the Smart Bird Feeder!

2.9.SD2 User Guide Instructions

There were a few changes to the functionality of the device that are listed throughout the SD2 sections of the paper. Such as removing live streaming. So the user guide instructions are made more simple.

1. Download the “Smart Bird Feeder” application from an app store.

2. After the application has finished downloading, access the application and click the “Sign up” button. Enter your email and personal information and click “Create Account”.

3. Plug in the Smart Bird Feeder using a 12 V DC charging cord and wait for the feeder to charge.

4. Place the Smart Bird Feeder in its desired location and fill the feeding chamber with bird feed.

5. Flip the power switches on the battery and the PCB.

6. Wait for a notification of bird activity or low feed.

7. To view pictures of previous feedings, select the “Bird Memories” tab. While viewing the screen, pull down to refresh the gallery.

8. Enjoy the Smart Bird Feeder!

3. Technology Investigation and Part Selection

The initial step in designing the Smart Bird Feeder, once we clarified our intentions and requirements going into the project, was to perform a very thorough technical investigation on various aspects of selecting the necessary parts to use in the development of our product. Prior to investigating and comparing various parts of the project, we start off with exploring current existing products and projects similar to our design idea. This leads us to current similar projects that use automatic species detection, deterring squeals, solar power, and a combination of them. Once observing the various parts commonly used, we start looking at things such as machine learning technology options and comparisons to help determine the correct development board for us to utilize. Next, we dive deep into the various wireless communications as we explain our reasoning for going wireless. After this, we will move onto comparing the various microcontrollers based on their computation power, memory storage, and amount of input/output pins relative to our requirements. Then naturally, we need to look into the various motors that could potentially be applied to our product in order to successfully close the hatch on the bird food to keep squirrels away from it, and the right distance sensing technologies available such as the potential implementation of a ultrasonic sensor to determine if the food supply needs. Then, prior to addressing the technological comparisons of the software side of things, we investigate the implementation of solar energy for the charge controllers, and the comparisons of various serial communication technologies. Then to end off the technological investigation, we take a look at the software side of various comparisons between full stack development technologies such as Flutter and React Native.

3.1 Existing Products and Projects

There are currently other bird feeders on the market. Our objective is to use this section to distinguish what already exists regarding bird feeding in the current market, and to gain knowledge for our implementation. This knowledge will help shape our design so we can provide innovations that have yet to be incorporated into existing designs.

3.1.1 Automatic Species Detection

The ability to detect the species of birds coming into contact with a bird feeder is shown in a similar product, Bird Buddy [9]. In the Bird Buddy's design, the user receives a notification that a feeding event is taking place and has the option to take a manual picture which is then scanned for species detection. The species won't be detected unless there is manual input from the user, however. This design can be enhanced by introducing more automation.

In our design, species detection will be a constant process instead of a manual one for a multitude of reasons. For one, there will most certainly be a time where the user cannot access the live feed of the bird eating, and thus would miss the opportunity to snap a picture of the bird to determine its species. Our smart bird feeder will take a photo of the event regardless of participation from the user, so that the user will never miss an opportunity to see who visited their feeder. Also, this functionality will allow the user to receive a notification about what species is currently feeding before they have to check the live feed, so they can decide beforehand if they have an interest in the particular bird who's currently being filmed.

3.1.2 Deterring Squirrels

The ability for a bird feeder to deter squirrels has shown up in a few similar products, one being the Squirrel Buster Bird Feeder. The product uses animal weight to determine whether or not there is a squirrel on the feeder, and the weight pulls down the device, limiting access to the food until the squirrel leaves. This is effective, however there's no way to actually deter the squirrel besides just boredom. They might stick around to investigate and waste time the birds could be feeding. Also, this limits the types of birds that can access the feeder due to weight constraints. Finally, the owner won't get to see the number of squirrels or the effectiveness of the device in action, which would be an exciting addition.

With these thoughts in mind, we've innovated this idea in our invention to overcome these obstacles. The feeder will have a hatch door that closes when the machine learning processor detects a squirrel. This removes the weight component and is a more accurate method of determining if there's a feeding predator. The owner will also get an alert so that if they want, they can watch the process in action. Lastly, the alarm system will actively deter the squirrels from staying on the platform, which opens up more time for the birds to feed.

3.1.3 Solar Power

The Bird Buddy is the closest device in terms of capability to our own, given its species detection ability. This means it relies on a power source in order for it to function properly, via charging it by USB-C. This requires more input from the user of the product however, and since it's an outside device may go unnoticed for some time.

As a secondary goal, The Smart Bird Feeder will mitigate this concern by including a solar panel incorporated in its design. Given that the device is already outside, we've concluded that a solar panel will be an effective solution to this problem. The cost of adding a solar panel is worth the time it will save the user not having to charge the device.

3.1.4 Combined Innovation

The existing products give the team a good vision of the market our design will fit into. The process of innovating on what’s available in the market is key to creating a good product. By combining enhanced species detection capabilities, squirrel deterrent, and solar power into a single design, we’ll have a strong footing when entering the market.

3.2 Machine Learning Acceleration Technology Comparison

Our smart bird feeder project requires machine learning (ML) inferencing with computer vision to detect objects and identify them as either a certain species of bird or as a squirrel. The component of our bird feeder which would perform these tasks would need to meet various criteria such as (but not limited to) minimal power requirements, compatibility with a CSI camera, and high capability for tensor processing. Because of this feature requirement, our team sought a mobile development system equipped with technologies which could accelerate machine learning inferencing. With this in mind, we looked toward technology that uses edge computing for AI and ML applications. Using AI, ML, and computer vision in projects has become increasingly common in recent years. Due in part to this, there is a wide variety of options to select from to use in building an ML model to perform object detection.

3.2.1 Product Options

The following section lists the products that were considered for performing machine learning inferencing and shares each products' strengths and weaknesses along with our thoughts on how each product option would specifically benefit our smart bird feeder project. Some products listed below were eliminated from consideration early on, and reasons for their elimination are given. A more direct comparison of the features of each of these products will follow this section, where the choice for a machine learning technology product which performs machine learning inferencing with object detection for our device will have been filtered down from many to only the most promising options.

3.2.1.1 Coral Dev Board

The Coral Dev Board was the first board to be considered for performing fast machine learning inferencing. The reason for this development being our first choice is because upon initial technical investigation for edge computing, we discovered a very convenient tutorial from Google developers and engineers which could have saved us some time and effort by avoiding writing all of our software from scratch if we chose to use the Coral Dev Board. This tutorial even shared a potential design for a bird house which could be created using a 3D printer.

A large drawback for the Coral Dev Board is its upfront cost. The Coral Dev Board costs $129.99 USD at MSRP for the model that ships with 1 GB of RAM, and a model with 2GB costs $169.99 USD at MSRP. There is a model advertised with 2GB of RAM, but our team cannot find any sellers for this model as the following sections in this paper will show, the upfront cost of the Coral Dev Board is among the highest of any of the boards or hardware accelerator tool options that we considered for use with our smart bird feeder device. Another potential drawback that our team saw with using this product was its relatively young age in the market, along with its low flexibility with regard to ML models that it is able to run. The Coral's ML accelerator module is made to accelerate TensorFlow Lite models, so models not created using TensorFlow Lite (includes models that were created with Google's TensorFlow software library) must be converted to a TensorFlow Lite format before they can be run on the Coral Dev Board. This limits the possibility of using pre-trained ML models with the Google Coral Dev Board. The key reasons for why this board was kept for final comparisons and considerations are due to its clear and concise documentation and its powerful Edge TPU.

3.2.1.2 Raspberry Pi 4 + Coral USB Accelerator

The Raspberry Pi 4 is a series of single-board computers known for its low cost and its wide usage in the area of robotics and computer science. Our team first decided to consider the Raspberry Pi series of products to use for this project due to their popularity and vast support network in the engineering community, along with their relatively low cost. Our team was specifically interested in the Raspberry Pi 4 Model B. Some of the features of this product that stood out to us were its 1.5 GHz 64-bit quad core ARM Cortex-A72 System-on-Chip (SoC) processor, along with its inclusion of 2.4 GHz and 5.0 GHz IEEE 802.11ac on-board WiFi, 2 USB 3.0 ports, 2 USB 2.0 ports, a 40 pin GPIO header with 5V DC power input, 2 micro-HDMI ports for dual displays, and LPDDR4-3200 SDRAM (size varies by model), just to name a few. The included feature of Bluetooth 5.0 was especially enticing to our team at first, since for a short time we had considered using Bluetooth for communication between the smart feeder device and the user interactive application. However, since we have now decided to use WiFi for interacting with our user application instead, built-in Bluetooth is not currently necessary for our smart bird feeder device and is no longer considered a large selling point for the purposes of our project. The versions of the Pi 4 that contain 1GB and 2GB of RAM are discontinued and out of stock, respectively. Even so, at a cost of $55 USD at MSRP for the 4GB RAM version of the Pi 4, and $75 USD at MSRP for the 8GB version of the Pi 4, there is little to dislike with regards to the cost of this product alone.

The Raspberry Pi 4 alone is not very well-suited for training and running machine learning models, however. This is where the Coral USB Accelerator comes in. The Coral USB Accelerator would connect to the Raspberry Pi 4 and provide hardware acceleration using the USB Accelerator's Edge TPU. The Coral USB Accelerator device is similar to the Coral Dev Board in computational power, but lacks some of the features that the Coral Dev Board includes. This means that it lacks the ability to be connected to an external display and run an operating system on its own, and also lacks GPIO pins to connect to other devices such as the PCB that will operate the ultrasonic sensor and food mechanism for the bird feeder. If we chose to use this product, we would need to have another device with an OS installed to run the ML models on, which could increase costs. For these reasons, this product is no longer being considered for use in our smart bird feeder project.

3.2.1.4 Raspberry Pi 4 + Intel Neural Compute Stick 2

Our team again saw an opportunity to use the Raspberry Pi 4 in this project, only this time it would be combined with the Intel Neural Compute Stick 2. The Intel Neural Compute Stick 2 is similar to the Coral USB Accelerator in that it is a small form factor device built to be connected to another existing computer device to accelerate the device's machine learning capabilities.

Unfortunately, this combination of products suffers from the same drawbacks as the Coral USB Accelerator combined with the Raspberry Pi 4. For this reason, this product quickly fell out of consideration for use in this project.

3.2.1.5 Arduino Nano 33 BLE Sense

Based on the nRF52840 microcontroller (32-bit ARM Cortex-M4 CPU @ 64 MHz), the Arduino Nano 33 BLE Sense is a small form factor (45x18m) AI board developed with the main goal of enabling Edge Computing AI applications on an edge device in a network. It uses TinyML (an emerging area of deep learning) and TensorFlow Lite to enable the creation of these ML models. This board is meant to be an evolved version of the Arduino Nano with its improved processing speeds and larger program memory and RAM. Some of the other notable features included with the Arduino Nano 33 BLE Sense are as follow:

- 1 port for a microUSB connection for power and connection to a computer.
- Communications chipset supporting both Bluetooth Low Energy (BLE) and Bluetooth 5.0 with a processor that supports Bluetooth pairing via NFC
- 3.3V operating voltage
- 1 UART, 1 SPI, and 1 I2C serial communication pins
- Microphone
- Sensors for tracking gestures, proximity, light color and intensity, temperature, and humidity
- 9 axis inertial sensor

The first factor that drew our team to the Nano 33 BLE Sense was its low price. At \$31.10 USD at MSRP without pin headers, and \$33.40 USD at MSRP with pin headers, this device is very cost-efficient, which plays to the advantage of our goal to minimize the budget for our smart bird feeder device. Another feature of the Arduino Nano 33 BLE Sense that would benefit our project would be its inclusion of a built-in microphone, which we could potentially use if we wanted to recognize the species of bird(s) making noise in close proximity to our smart bird feeder on top of doing object recognition of birds and squirrels. Another enticing factor of this product is its low operating voltage of 3.3V. The low power requirements of this small form factor device means that it would not put much stress on the power supply for our smart bird feeder, thereby increasing the potential longevity of The Smart Bird Feeder's battery.

A drawback to the Arduino Nano 33 BLE Sense is its lack of any on-board peripheral ports. The only USB port on the Nano 33 BLE Sense is its aforementioned port for a microUSB connector. Another negative aspect of this product is its requirement to be connected to an existing computer device, and its additional requirement of being programmed using the Arduino IDE. The members of our team believe that using this IDE would limit us, and prototyping would also not be as simple as it would be with other ML celebration hardware with support for such things as full Linux OS. This device could potentially be used in combination with a device such as the Raspberry Pi 4, similar to the previously proposed combinations of the Raspberry Pi 4 and the Coral USB Accelerator or the Intel Neural Compute Stick 2, but more investigation would be required before attempting to do this, since we are not certain that the Raspberry Pi 4 would support an Arduino product such as the Nano 33 BLE Sense. Due to these reasons, our team has decided to no longer consider the Arduino Nano 33 BLE Sense for use in this project.

3.2.1.6 Jetson Nano Developer Kit

The Jetson Nano Developer Kit series is a family of products made for developers to train and run multiple ML models in parallel on a single small computer. There are two models of the NVIDIA Jetson Nano currently available for purchase, and the advantages and disadvantages of both of these options are subsequently discussed and considered in detail. As for similarities, all devices in the Jetson Nano family come equipped with ML accelerator hardware in the form of a 128-core NVIDIA Maxwell™ GPU, as well as a Quad-core ARM® A57 @ 1.43 GHz CPU, which are perfect for developing and deploying ML models on an edge device such as the Jetson Nano quickly and effectively. Both Jetson Nano Developer Kits also include the following features:

- 4Kp30 | 4x 1080p30 | 9x 720p30 (H.264/H.265) video encoder
- 4Kp60 | 2x 4Kp30 | 8x 1080p30 | 18x 720p30 (H.264/H.265) video decoder
- 64-bit LPDDR4 25.6 GB/s memory (size varies by model)

- Gigabit Ethernet, 802.11ac wireless
- microSD expandable storage (card not included with developer kit)
- Dimensions of 100 mm x 80 mm x 29 mm
- 40-pin header (GPIO, I2C, I2S, SPI, UART)
- 12-pin header for power, UART, and related signals
- 4-pin fan header

The true shining feature shared by the Jetson family of devices is the popular and acclaimed JetPack Software Development Kit (SDK). Any NVIDIA Jetson Nano Developer Kit is capable of training and deploying AI software (such as an object detection ML model for bird and squirrel recognition) using the NVIDIA JetPack SDK's AI platform. This SDK contains the L4T package with drivers for a full Linux OS, as well as libraries and APIs for deep learning, which would vastly simplify the object detection ML model development process for our team. This SDK paired with the numerous tutorials, sample projects, and extensive documentation available for the NVIDIA Jetson Nano products made the Nano a top contender for use in our project.

3.2.1.6.1 Jetson Nano 4GB Developer Kit

The Jetson Nano 4GB Developer Kit is the original model of the Jetson Nano. It consists of the developer kit module plus a carrier board for the module to sit on. This developer kit includes 4GB of 64-bit LPDDR4 25.6 GB/s RAM, two CSI camera connector slots, four USB 3.0 Type A ports, and one micro-USB 2.0 for connecting to other peripheral devices. The Jetson Nano 4GB Developer Kit can run in a 5W low-power mode or run at its default 10W power usage state. The low-power state option that the Jetson Nano 4GB Developer Kit is enticing to our team, since this would make the Jetson Nano Developer Kit even better suited to fit our specified power requirements. Although the Jetson Nano 4GB Developer Kit does not come with WiFi pre-installed, it does have an M.2 Key E slot for a developer to install their own WiFi module, or a WiFi adapter could also be connected to the device via one of the USB ports on the Jetson Nano 4GB Developer Kit carrier board.

3.2.1.6.2 Jetson Nano 2GB Developer Kit

Unlike the original previously mentioned 4GB Developer Kit, the Jetson Nano 2GB Developer Kit is labeled and marketed as reasonably priced for everyone and lacks some of the features that the 4GB developer kit has, but is still powerful enough to run virtually any ML model that the 4GB developer kit can run. The Jetson Nano 2GB Developer Kit includes a 128-core NVIDIA Maxwell GPU, a Quad-core ARM A57 @ 1.43 GHz CPU, a 2GB 64-bit LPDDR4 35.6 GB/s memory, a microSD storage, Gigabit Ethernet with 802.11ac wireless, a 1x MIPI CSI-2 connector, HDMI display port, and much more. Similar to the Jetson Nano 4GB Developer Kit, the 2GB version also has two module power modes which are the 10W default mode for higher performance, and the 5W mode for less energy usage. When comparing the developer kits, we see that there are many similar technical specifications, however there are some differences other than the memory size. Unlike the 4GB version, the 2GB version has one USB 3.0 Type A slot and two USB 2.0 Type A slots. Moreover, one of the other main differences between the Jetson Nano 4GB Developer Kit and the Jetson Nano 2GB Developer Kit, is that the 2GB comes with (depending on the region) the 802.11ac wireless adapter with an extension cable. This means that the 2GB version will be able to connect to the Wi-Fi seamlessly without the hassle of getting a Wi-Fi adapter. Moreover, the overall idea for the Jetson Nano 2GB Developer Kit, is to deliver an incredible AI performance at a relatively low price, so that there are more Jetson developers around in the world with the same software tools. In all, the Jetson Nano 2GB Developer Kit offers most of the same features as the Jetson Nano 4GB Developer does, with a few select features removed to save cost. Therefore, if our team chooses to use a product from the Jetson Nano Developer Kit family for developing our device, the most significant reason for choosing the Jetson Nano 2GB over the Jetson Nano 4GB would be its $59.99 USD at MSRP price tag, versus the $99.99 USD at MSRP price of the Jetson Nano 4GB.

3.2.2 Product Comparisons

At this point, all of the machine learning products that our team considered have been listed, and several options have been removed from consideration. What remains are the products which our team spent the most time and effort considering for use in our project. The following section compares these remaining products in the area of CPU and GPU clock frequency, power consumption, monetary cost in USD, choice of memory size and speed, and the amount of convenience offered by the miscellaneous features contained in each product

3.2.2.1 SoM CPU and Graphics Comparisons

In Table 4 shown below, the CPU and GPUs of the System-on-Module located on the Coral Dev Board by Google and the System-on-Module located on the NVIDIA Jetson Nano Developer Kit[1] are compared, along with the amount and speed of cache on each system. This comparison will assist our team in discovering which product is better suited for performing both novel and repeated computations necessary for nearly instantaneous object detection of birds and squirrels, communication with our PCB, and communication with our smart bird feeder application, without being significantly slowed down.

[1] The SoM on the NVIDIA Jetson Nano 2GB Developer Kit and the NVIDIA Jetson Nano 4GB Developer Kit contain identical CPU and GPU components.

Table 4 - Coral Dev Board v. Jetson Nano Dev Kit (CPU & GPU comparison)

	Coral Dev Board	**NVIDIA Jetson Nano Developer Kit**
CPU	Quad-core Arm Cortex-A53, plus Cortex-M4F	Quad-core ARM Cortex-A57 MPCore processor
CPU Clock Frequency	1.5 GHz [1]	1.43 GHz [4]
GPU	Edge TPU coprocessor; Vivante GC7000Lite	128-core NVIDIA Maxwell @ 921 MHz [4]
Cache	32 KB L1 Instruction Cache; 32 KB L1 Data Cache [3]	48 KB L1 instruction cache (I-cache) per core; 32 KB L1 data cache (D-cache) per core | L2 Unified Cache: 2 MB [4]

Table 4 shows that the Coral Dev Board has a higher clock frequency than the NVIDIA Jetson Nano, but the Jetson Nano evidently has more cache available, which can contribute to decreased time taken for repeated computations. Another noticeable advantage that the Coral Dev Board has over the Jetson Nano is its Edge TPU coprocessor, which is developed by Google specifically for high performance ML inferencing applications with TensorFlow Lite models. However, the Edge TPU architecture is not very flexible, and models would need to be either created using or converted into Google's TensorFlow Lite format to be functional. In terms of raw computational ability, the Coral Dev Board has the advantage here.

3.2.2.2 Power Considerations

Power consumption of the machine learning board used is one of the leading factors in our team's decision-making process for choosing a product, since our team has a goal of minimizing the overall power consumption of our smart bird feeder device to maximize battery life while minimizing the size of the portable power supply used.

The Coral Dev Board may be powered using a USB Type-C connection from a 5V, 2-3A power supply. The NVIDIA Jetson Nano Developer Kit's power requirements depend on whether the 2GB Developer Kit or the 4GB developer kit is being used. The 2GB Jetson Nano Developer Kit requires 5V⎓3A input via a USB Type-C connection, while the 4GB Jetson Nano Developer Kit may be powered by 5V⎓2A input from a micro-USB connection or may be safely run in the default 10W power mode by using a 5V⎓4A barrel jack power supply. The Jetson Nano 4GB Developer Kit may also be powered using its 40-pin J6 header, which contains two 5V pins with each accepting up to 3A of current. On the other hand, the Jetson Nano 2GB Developer Kit's two 5V power pins can power it at 2.5A per pin.

Considering the power that must be delivered to each of these boards for them to properly function, the Jetson Nano 4GB Developer Kit and the Coral Dev Board seem to require the least amount of power (5V⎓2A) when operating at their minimum capabilities. However, our team expects to run multiple applications on these boards at once, and machine learning models alone typically require a large number of resources to run, meaning that the system on the modules of the boards would be expected to be working at or near maximum capacity when running the object detection model for detecting birds and squirrels. Due to this, our team does not believe that 5V⎓2A would be a suitable state for the board to run in, and in the case of the Jetson Nano 4GB, it would need to be run in its 10W default state, which means that the 5V⎓4A power jack would need to be connected. The Coral Dev Board would also be expected to draw more than 2A of current while it operates with an ML model running, so it would actually be operating at or near 5V⎓3A, which matches the power requirements of the Jetson Nano 2GB Developer Kit. Because our team is seeking to minimize the size and cost of our power supply, while maximizing battery life of the power supply, we would like to avoid using a board which would require 5V⎓4A to be supplied by our power supply. Therefore, the Jetson Nano 4GB Developer Kit has a disadvantage in terms of power efficiency for our smart bird feeder device. Since the Jetson Nano 2GB Developer Kit and the Coral Dev Board were both created for developers who want to run ML applications on them, and their 5V⎓3A power requirements would be more feasible to meet for the purposes of our project.

3.2.2.3 Memory Comparison

The memory on the System-on-Module located on the Coral Dev Board by Google and the System-on-Module located on the NVIDIA Jetson Nano Developer Kits must also be compared to properly judge which system can handle more simultaneous tasks. Table 5 lists the RAM and Flash memory units available for the Coral Dev Board, the Jetson Nano 4GB Developer Kit, and the Jetson Nano 2GB Developer Kit.

Table 5 - Coral Dev Board v. Jetson Nano Dev Kit memory comparison

	Coral Dev Board	**NVIDIA Jetson Nano 4GB Developer Kit**	**NVIDIA Jetson Nano 2GB Developer Kit**
RAM	1, 2, or 4 GB LPDDR4	4GB 64-bit LPDDR4 25.6 GB/s	2GB 64-bit LPDDR4 25.6 GB/s
Flash Memory	8GB eMMC for 1GB & 2GB RAM, 16GB eMMC for 4GB RAM or expandable storage via microSD	16GB eMMC or expandable storage via microSD	Expandable storage via microSD

As Table 5 shows, all the boards under consideration use LPDDR4 RAM, and each product’s module offers expandable storage by connecting a microSD card. It is notable that the The Jetson Nano 4GB and the 4GB option available for the Google Coral Board are the evident victors in this category, but it is important to note that while the Jetson Nano 4GB Developer Kit does have more memory available than the Jetson Nano 2GB Developer Kit, there is a swap memory feature built into the Jetson Nano by default which essentially utilizes the memory from the microSD card connected to the Jetson Nano as a virtual memory substitute for physical RAM. This means that when the physical memory on the Jetson Nano 2GB is at capacity, the memory space can be virtually extended so that processes do not need to be killed. It is also notable that while a 2GB version of the Coral SoM is available, our team could not locate any retailer with a 2GB version of the full Coral Dev Board available.

3.2.2.4 Product Pricing

One of the most crucial aspects of picking a machine learning processor is price. Our team has set out to achieve the best price possible while still maintaining the requirements and functionality listed in section 2.4. In the sections labeled 3.2.2, the team is breaking down many aspects of the development kits, as to ensure our functionality goals will be met by our choice. As long as that metric is achieved and the price differential isn’t high enough to consider removing functionality from our design, then the lowest cost option will be selected.

The team has chosen a design philosophy that values functionality first, then price, then effort required. Hence, while the higher cost options have shown to have easier implementation, as long as the functionality requirements can be met with hard work our team will value price at a higher standard. We’ve also taken into consideration whether or not the extra effort will hurt the timeline of the project, which would ultimately limit functionality, which is our prime concern.

All of these factors taken into consideration, the price for the Coral Dev Board is $129.99 at MSRP. The price of the NVIDIA Jetson Nano 2GB development board is $59.00 at MSRP. Finally, the NVIDIA Jetson Nano 4GB development board is $99.00 at MSRP. Given these pricing dynamics, the Coral Dev Board is certainly the highest priced board. When taking price into consideration, the Jetson Nano 2GB Developer Kit is the best option.

3.2.2.5 Product Developer Software Comparison

In the search for the most ideal developer kit for the Smart Bird Feeder, it is highly important to not just look at the hardware, but also look at the software associated with each option.

In looking at the NVIDIA Jetson Nano Developer Kit, we see that NVIDIA provides a simple way of getting started by just inserting a microSD card with the system image and begin using the NVIDIA JetPack SDK once booted up. With the SDK being utilized across the entire NVIDIA Jetson family of products (in other words, all Jetson modules and developer kits). It is stated that the JetPack is compatible with NVIDIA's world-leading AI platform for training and deploying AI software in an attempt to reduce complexity and or effort on the developers' side. Furthermore, the SDK includes the most recent Jetson Linux Driver Package (L4T) with the Linux operating system along with CUDA-X accelerated libraries and APIs for deep learning, computer vision, accelerated computing, and multimedia. Moreover the JetPack is able to be installed and upgraded via the Debian package management tool on the Jetson and host Debian packages for JetPack components for installing on the host. When looking at the nano, some of the first things one may look at are the key features in JetPack. The top features include TensorRT, cuDNN, CUDA, Multimedia API, Computer Vision, developer tools (CUDA Toolkit), and supported SDKs and tools (such as the PowerEstimator & Deepstream SDK). With this said, it evident that JetPack provides features that seem highly applicable to the Smart Bird Feeder with TensorRT as a high performance deep learning inference runtime for image classification, segmentation, and object detection neural networks as one of the main software features of the bird feeder is to be able to use implement object detection for identifying bird species and squirrels. NVIDIA says that the TensorRT-based applications are able to perform up to 40 times faster than CPU-only platforms during inference.

When looking at the other developer kits, we see that Coral Development Board is marketed as a quick and easy board to use as well, with just requiring the user to flash Mendel Linux to it, and then accessing the board's shell terminal to update some of the software before being able to run an an image classification model on it. Coral states that they simplified the development with the Coral by making the Edge TPU compatible with the standard TensorFlow Lite API so that those that already have code utilizing TensorFlow Lite API, just need to change a couple lines of the code. They also go further to say that they helped to make development even easier by having made a wrapper library (the PyCoral API) to handle boilerplate code that is required with running an inference with TensorFlow Lite. Moreover, the PyCoral API which is built on top of the Tensor Flow Lite Python API and provides advanced features for the Edge TPU such as model pipelining across multiple Edge TPUs. The modules listed in the API overview include pycoral.utils.dataset, pycoral.utils.edgetpu, pycoral.adapters.common, pycoral.adapters.classify, pycoral.adapters.detect, pycoral.pipeline.pipelined_model_runner, pycoral.learn.backprop.softmax_regression, and pycoral.learn.imprinting.engine. With this said, if we were to utilize the Coral Development Board, we would most likely look into the pycoral.adapters.detect as it functions to work with a detection model, with it returning a list of Object objects with each containing the detected objects id, score, and bounding box. Due to our unique project requirements on object detection, it is important to note that if we intend to customize a base model, the process will require multiple build steps along with several hours of model training even with a powerful GPU. For example, retraining a base MobileNet model from ImageNet ILSVRC2012 cannot run on the Edge TPU and some of the required tools are not compatible with the Coral Development Board, so several steps must be taken on a separate powerful desktop computer.

Furthermore, when looking at the benchmarks provided by both the Jetson Nano and Coral's Edge TPU. When it comes to the Jetson Nano, it is able to run a relatively wide variety of advanced networks that include full native versions of popular ML frameworks such as TensorFlow, PyTorch, Caffe/Caffe2, Keras, MXNet, etc. With the supported networks are able to be utilized to build autonomous machines and complex AI systems by implementing image recognition, object detection & localization, pose estimation, segmentation, video enhancement, intelligent analytics, and other similar robust capabilities. On the other hand, when looking at the performance benchmarks provided by Coral, it is labeled that the individual Edge TPU is capable of performing around 4 trillion tera-operations per second (TOPS) while using about 0.5 watts for each TOPs to get 2 YOPS per watt.

Due to our project dependence on object detection, it is important that we take a look at several trained models provided by both Coral and NVIDIA to make an informed decision. Thus, when looking at the Coral Edge TPU, we can use real-time video at over 100 frames per second, with the ability to even run multiple detection models concurrently while maintaining a high frame rate. Coral provides several trained models that are compiled for Edge TPU, along with example code to run them and information on how to train our own model with TensorFlow. Some of the example code provided consists of python code for image recognition with video (to identify objects like bird and dog). For example, Coral provides documentation on a Dev Board that uses Edge TPU Python API to identify birds with machine learning inference, and the video feed processed via a MobileNet model which identifies different animals and detected bird species. When it comes to NVIDIA's Jetson Nano, they provide us with a coding environment by installing prerequisite libraries & downloading DNN models such as SSD-Mobilenet pre-trained on the popular 90-class MS-COCO dataset. Moreover, they provide several object detection examples with NVIDIA TensorRT.

Because NVIDIA's Jetson Nano Developer Kits and the Google Coral Dev Board use the MobileNet ML model in their tutorials for object detection, we have included Table 6, which displays performance benchmark results in frames-per-second (FPS) from tests run on each ML accelerator product. These results were provided by *NVIDIA Jetson Nano: Deep Learning Inference Benchmarks*.

Table 6 - FPS performance comparison [15]

Model	Application	Framework	NVIDIA Jetson Nano	Google Edge TPU Dev Board
MobileNet-v2	Classification	TensorFlow	64 FPS	130 FPS
SSD MobileNet-V2 (960 x 544)	Object Detection	TensorFlow	8 FPS	DNR
SSD Mobilenet-V2 (480 x 272)	Object Detection	TensorFlow	27 FPS	DNR
SSD Mobilenet-V2 (300 x 300)	Object Detection	TensorFlow	39 FPS	48 FPS

DNR stands for did not run.

Table 7 below displays the comparison between the time spent to perform a single inference on several popular models using Edge TPU. In the comparison, they are using trained models using ImageNet dataset with 1,000 classes. The values are found in the page called Edge TPU performance benchmarks under the *Docs & Tools* tab. MobileNet v1 and MobileNet v2 have nearly identical performance, so either of them would be suitable for use when limiting time per inference.

Table 7 - Time per inference in milliseconds [12]

ML Model	Dev Board with Edge TPU
MobileNet v1 (224x224)	2.4
MobileNet v2 (224x224)	2.6
MobileNet v1 SSD (224x224)	11
MobileNet v2 SSD (224x224)	14

3.2.2.6 Miscellaneous Features Comparison

There are some notable differences in the small features that the boards under consideration have available. In this section the differences will be highlighted in order to determine whether or not the changes should be taken into consideration for the product choice.

A crucial aspect of the product’s design is WiFi capability. Thus, this is the component in this section with the highest pull on the team's decision. The Coral Dev board comes equipped with a module with both WiFi and Bluetooth capability. The Jetson Nano 2GB Developer kit comes with a WiFi adapter included in the Kit which requires the use of a USB port. However, the Jetson Nano 4GB Developer Kit does not come with any WiFi support.

Given the importance of WiFi in the design, the Jetson Nano 4GB Developer Kit would have to be supplemented with a WiFi adapter. Given that the 4GB model is already priced higher than the 2GB model, adding an extra component would increase the price even further. This difference then plays a large role in the decision and skews the choice towards the Coral Dev Board and 2GB model. When considering WiFi capability, the 4GB lack of accessibility is a heavy detractor.

For USB support, the Jetson Nano 4GB Developer Kit has access to 4 USB 3.0 ports and 1 USB 2.0 Micro-B port, while the Jetson Nano 2GB Developer Kit has access to 1 USB 3.0, 2 USB 2.0, and 1 USB 2.0 Micro-B ports. The Coral Dev board contains a USB 3.0 Type-C port, USB 3.0 Type-A port, and USB 2.0 Micro-B serial port.

In our project, it’s important to distinguish between the two stages of the product’s lifespan and how this affects our USB requirements. During the prototyping, testing, and set up phase, our team will need access to three USB ports. One for the WiFi adapter, another for a mouse, and finally a USB for a keyboard. Since the mouse and keyboard require a very low level data transfer, they essentially gain nothing from USB 3.0 access. However, the WiFi adapter will gain an advantage from the higher data transfer speed, which means that during this phase of development, access to the USB 3.0 port is beneficial. After the testing period and during the time period of the product's lifespan, while the keyboard and mouse will no longer be required, the WiFi adapter will still be a required tool. Since each board listed here has the capability for a USB 3.0 port support, there will always be a spot for the higher speed port allocated to a WiFi adapter. Thus, we’ve concluded that the USB capabilities of all the devices listed suit the needs of our design, and can be ruled out of consideration after this research.

When taking into consideration camera connection ports, the Jetson Nano 4GB Developer Kit and Coral Dev Board both have 2 MIPI CSI-2 lanes, while the 2GB kit has 1 MIPI CSI-2 lane. Currently, the product is only designed with one camera in mind, but there could be benefits to another camera in the future. However, since our project requires only one camera this added functionality is considered unnecessary.

In terms of display capabilities, The Jetson Nano 4GB Developer Kit comes with an HDMI and Display port. The Jetson Nano 2GB Developer Kit and Coral Dev Board comes with only an HDMI port. However, the display mechanism will only be of use in the development phase, and the HDMI port is sufficient, so this aspect did not influence the group's design decision.

3.2.3 Machine Learning Product Choice

Our team has chosen to use the NVIDIA Jetson Nano 2GB. Its relatively low price of just $59 given its wide array of features, impressive processing power, straightforward user interface, machine learning capabilities, community support, and relatively low power usage all combine to make a fantastically enticing product for machine learning inferencing which is simple for developers at many levels to perform prototyping on and will suit the needs of our smart bird feeder device very nicely. The NVIDIA TensorRT SDK built on CUDA that can comes packaged within the JetPack SDK is especially enticing to our team, since it packages together many of the software components that we need for performing object detection, and some of our team members already have experience with the CUDA parallel programming model. Plus, if we can use the Nano 2GB Developer Kit, our device will be more cost-effective, but if we require more memory to run our ML model(s), we can always upgrade to the Jetson Nano 4GB Developer Kit with relative ease.

3.2.3.SD2 Machine Learning Product Choice

As stated at the end of the previous section, the group was considering the move from the Jetson Nano 2GB model to the 4GB model. During testing, the 2GB model was running with the full use of its RAM when utilizing the machine learning program. However, there are many other tasks this processor needs to perform. The team came to the decision during testing that in order to fully realize all of our requirements, we need more RAM. The additional functions that the operating system distribution on the Jetson Nano 4GB were also appealing, so ultimately, we switched to the 4GB model and kept the 2GB model for testing purposes.

3.3 Wireless Communication

The Smart Bird Feeder will need to have a means of communication. The Smart Bird Feeder is designed to be placed outside in order to attract birds to the feeder. Birds are known for being mostly skittish creatures, so if a user wants to maximize the potential of their bird feeder, they should place it a good distance away from themselves and not directly interfere during the feeding process. However, a main component of enjoying a bird feeder is watching the birds eat food from the feeder, yet the closer a user gets to the bird feeder, the less likely the birds will feel comfortable to stick around.

Our goal was to solve this problem by implementing a communication technology that could allow our users to stream the feeding process right to a capable device, so they can enjoy the feeding process without interrupting. There are also other secondary benefits to a communications network, as we could implement notifications for various bird feeder applications such as low bird feed, low battery, and memory storage of different feeding events.

The following subsections will discuss and provide considerations on the pros and cons of the communications technologies, for a detailed explanation on why certain technologies were picked over others. Thus said, the following subsections will get into the nitty gritty to help discuss the decisions made.

3.3.1 Wireless vs Wired Communication

We chose to use a wireless communication framework to achieve these benefits. To achieve the same goal with a wired connection, there would have to be a greater restriction placed on the distance away the feeder could be placed. Also, the user would have to have a compatible device that could accept the wired signal, or we'd have to provide one. Providing a device to watch the bird feeding take place would increase cost and given that most users would already have a wireless capable device it would be wasteful and redundant.

We've also found users are much more comfortable with downloading an application that achieves the same goal. Users are already accustomed to downloading compatible apps for their smart devices. We also considered whether or not users would have access to a wireless communication framework, and concluded that it's likely they would, given the need for these networks in everyday life. There's also a possibility that the network does not extend far enough to reach their desired bird feeder placement, but this issue would be even more apparent in a wired communication. One of the main goals is to keep this system easy to install in any backyard and stay free from any inconvenient situations where users would have to run a wire around their yard just to hook up the bird feeder. Moreover, a wired connection would also have to deal with various length wires to accommodate the needs of individual yards. Therefore, it is evident that in order to free everyone with the hassle that comes with wired transmission, our product will utilize the wireless network like so many smart devices these days.

3.3.2 Wireless Communication Options

Given that we've already decided to implement wireless communication for the benefits listed above, this section will break down the wireless communication options we considered. Choosing a good wireless communication framework will have effects on performance, access, battery life, and data transfer speed.

The implementation of wireless communication will allow The Smart Bird Feeder to transfer data with already commonly established devices such as smart phones with a clean appearance, and without depending on a long ethernet cable. The smart bird feeder is intended to be established outdoors, so this requires a robust communication protocol that can handle noise and physical barriers between it and the house for things such as trees and various plants and weather conditions. In looking at our system and the use of the developer kit, it will be suggested to utilize the developer kit to transmit the necessary data such as images, statuses, and live video streams. This means that the communication methods will be based on what is supported by the developer kit, and more specifically which ones of those that are supported are the best in terms such aspects of performance, reliability, and battery life. With most developer kits available for use in this product, the most commonly supported wireless communication technologies consist of Wi-Fi, Bluetooth, and potentially infrared. Due to the fact that our smart bird feeder will most likely need to transfer data to be stored on a cloud server, it is highly assumed that Wi-Fi will be the best bet, however the following subsections will look into what goes into each of the technologies and explain whether or not they fit for this system.

3.3.3 Wi-Fi

In order to have wireless communication for the obvious benefits listed prior, it provides Wi-Fi as a top contender to successfully transmit data from the bird feeder to cloud storage and mobile application. Therefore, it is important that we fully understand the Wi-Fi brand name that was trademarked by the non-profit Wi-Fi Alliance. This family of wireless network protocols are based on the IEEE 802.11 standard which defines the protocols that allow communication with Wi-Fi-enabled wireless devices such as mobile smart phones that will support our mobile application. Nearly every smart phone supports Wi-Fi to connect to the internet, as it has become a major wireless communication standard supported by many electronic devices and a common way of communication.

When considering the implementation of Wi-Fi as the means of communication, it is important to consider how large the range is in order to determine the usefulness of implementation. Wireless network’s range will often vary dramatically depending on the type of network, thus in order to determine the usefulness we shall start by looking at the common frequencies used. Moreover, Wi-Fi routers commonly operate at a 2.4 GigaHertz band that can reach up to 150 feet indoors and around 300 feet outdoors. This means that it must be recommended that users of our bird feeder must do some preliminary investigation to determine an area which will satisfy its needs to connect to the Wi-Fi. We will also need to recommend to users that unless they have strong signals outside their house, they are recommended to set the device relatively close as physical obstructions in homes, such as brick walls or metal frames which may reduce the range of a Wi-Fi network.

Due to the high likelihood of integrating Wi-Fi to The Smart Bird Feeder for communication to users, it is important to look at vulnerabilities brought about by this technology, so we are prepared for any surprises when using Wi-Fi. While Wi-Fi has been around for a while, it still has some flaws when it comes to network security. That being said, there may need to be an advisory that intruders could gain access to the network and potentially take advantage of the developer board within The Smart Bird Feeder. This network intrusion could happen due to a variety of different reasons such as a weak Wi-Fi password. While the Wi-Fi security vulnerabilities may be of concern, and is something that everyone should take seriously, it is believed that due to the lack of importance the bird feeder is in the eyes of hackers, it most likely is not a large concern. That said, a larger concern comes from the fact that Wi-Fi networks in this day and age are very much saturated with a load of many different Wi-Fi connecting devices. Due to a restricted bandwidth on a wireless network, the increased number of these devices connecting to the WiFi network are negatively impacting the network. One of the unique aspects of our product is uploading images to our digital library of birds to allow users to stay up to date with the bird feeder. Therefore, the requirements that The Smart Bird Feeder needs in order to function as planned is highly dependent on the network performance which may be affected as upload and download speeds are relatively sensitive to the number of simultaneously connected devices on the network. Moreover, even with the systems overhead, this form of wireless transmission is the most suitable for our smart bird feeder for its performance, reliability, and practicality.

3.3.4 Bluetooth

In looking at wireless communications that are commonly supported on object detection developer kits, bluetooth appears as a contender. Bluetooth is a short-range technology standard utilized for communicating data between devices over a relatively short distance via UHF radio waves within the ISM bands that run around 2.4 GHz. It is a low power radio that allows transfer of data over seventy nine channels in the unlicensed industrial, scientific and medical frequency band mentioned in the previous sentence. While this standard radio protocol is behind many day to day devices such as headphones, and wireless speakers, it is limited to a short range of about 10 meters (30 feet). Unlike Wi-Fi, Bluetooth varies in many ways, such as the fact that it was formerly standardized as IEEE 802.15.1 but now has been continued under the Bluetooth Special Interest Group (SIG). Furthermore, Bluetooth works on much shorter distances than Wi-Fi, and emphasizes direct communication between a packet-based protocol with a master and a slave architecture.

With what was mentioned, it should be noted that while Bluetooth is a commonly supported wireless communication protocol, it has some cons with the fact that it is short range and will limit the availability of places in which The Smart Bird Feeder may be placed within yards.

3.3.5 Infrared

Table 8 shows a comparison of several of the most common means of wireless communication in the modern era, one of which being infrared (IR) communication. Infrared (IR) is a commonly found wireless mobile technology that has a wavelength spectrum between the ranges of 780 nm and 1mm. This band of electromagnetic radiation spectrum has wavelengths above the red visible light. Due to the wavelength infrared is within, it is unable to be seen by the human eye and is said to not impact any mobile communication systems. However, the problem lies with implementing infrared, is the fact that it requires a direct line-of-sight connection between the transmitter (LED) and the receiver. This is because the light-based communication has a relatively short transmission range as it is unable to penetrate physical barriers such as walls. Moreover, this wireless communication is often used between relatively short ranges within the same room such as remote controls. Therefore, when looking at how our smart bird feeder is going to be used compared to the limitations of infrared, it is evident that relying on a technology that requires a direct line-of-sight between transceiver and receiver since it is loses connections quite easily due to physical barriers, does not match the needs of a reliable means of communications outdoors.

Table 8 - Wireless technology comparison

	Infrared	**Bluetooth**	**Wi-Fi (802.11)**
Range	1 m	10 m	91 m
Data rate	115 Kbps	3 Mbps	54 Mbps - 600 Mbps
Power consumption	20-50 mA	5 - 15 mA	50-110mA
Frequency	850 nm - 900 nm	2.4 GHz ISM Band	2.4 & 5 GHz

3.3.6 Wireless Communication Selection

In the technologic investigation we have done for the various wireless options available for our group to implement, we have unanimously selected Wi-Fi as the means of communication for The Smart Bird Feeder. When considering out of the three wireless technologies we landed between the utilization of bluetooth or Wi-Fi, since infrared depends highly on having line-of-sight and that there is a high chance that most phones will not support it. Thus, when comparing between the two left, we worried about bluetooths shorter range compared to Wi-Fi, and Wi-Fi's potential impact on upload and download speeds and security concerns. However, when discussing the pros and cons of both technologies, we concluded that we base our final decision based on what common products use, and what we believe is worth more to our consumers. That being said, many related projects often utilize Wi-Fi, thus proving its worthiness and reliability for smart devices, and the realization from the group members that range and ease of use is most important. Since ease of use is nearly shared equally between Wi-Fi and Bluetooth, the range is the next consideration. Therefore, due to the reliability and support for using Wi-Fi with Smart devices and the range provided by the Wi-Fi, we believe that our end users would agree with our decision that the use of Wi-Fi is the correct and appropriate choice for us.

3.4 Microcontroller

A microcontroller is a compact integrated circuit in an embedded system whose purpose is of a specific operation. For the integration and control of the several peripherals in the project, we needed a main microcontroller that would be able to make computations, store memory, and have enough input/output pins for the operation of the device. The main tasks of the microcontroller will be to receive and transfer data from the Jetson Nano development kit, control when the motor turns on, send and receive signals from the ultrasonic sensor, and send output signals to the speaker. The computer will be able to communicate with the Jetson Nano through either UART, SPI, or I2C. The microcontroller’s source of power will be a Lithium Ion rechargeable battery. An important aspect of the microcontroller is the ability to send and receive signals that are synchronous. This means that the signals are synchronized by a clock signal. This type of signal is necessary for sending PWM, or pulse-width-modulation signals. PWM signals work by varying the duty cycle, which is the amount of time that the signal is in the ‘high’ state. This type of signal becomes very important when controlling a servo motor, which will be discussed in section 3.5.1.3.

Before deciding on a microcontroller for this project, we must first understand the exact specifications that we will need. This includes aspects like the type of communication, number of pins, operating voltage, and size. Taking a look at the hardware block diagram in Figure 3, we can see all the peripherals that the microcontroller will need to support and control. The microcontroller will have to support a total of four peripherals. These peripherals are the communication between the MCU and Jetson Nano, ultrasonic sensor, external speaker, and door motor. This means that at least six I/O pins on the MCU will be utilized. This includes two for the transmitting and receiving of data to/from the Jetson Nano, at least one for the door motor control, two for the ECHO and TRIG pins on the ultrasonic sensor, and one pin for input to the external speaker. When looking for a microcontroller, we are not looking for the module with the most pins, but instead the module that will only meet our requirements, since more pins on a chip will mean a steeper price.

For the speed of the microcontroller, we need a module that will have enough speed to quickly communicate back and forth with the development kit and be able to quickly control the peripherals. However, too high of a CPU speed is not necessary, as this will just increase the cost of the device. For our requirements, ~16 MHz is an ideal speed.

Finding a microcontroller with a low-power mode is a major requirement for the project. Our marketing and engineering requirements define a device that will have low-maintenance by managing a long battery life. In order to reach these characteristics, we must find a microcontroller that can utilize a low-power mode option when it is not actively doing computation or sending/receiving signals from the peripherals. The three options in Table 9 all have this characteristic, as microprocessors without a low-power mode were not considered.

When considering memory and RAM, we again look at our requirements to understand what the best option for our project would be. Since the MCU will not be storing data, but only sending data to the development kit, where it will be stored on external memory, the memory specification of the MCU does not need to be relatively large. What is more important is the RAM. For all the microcontrollers listed below, the RAM is 2K x 8. This means that they have 2048 individually addressable units and can write or read 8 bits at a time. This also means that there are 11 address lines and 8 data lines.

3.4.1 Microcontroller Options

The following subsections under this section, addresses the various microcontrollers available for us to investigate and compare their individual features to determine which set of specifications are worth more to our Smart Bird Feeder. The options implemented are highly important for understanding the design of our programmable circuit board.

3.4.1.1 MSP430FR6989

The MSP430FR6989 is a microcontroller in a family of boards created by Texas Instruments. This board in particular is a 16 MHz module with a 2 KB RAM size. This microprocessor can reach 100uA/MHz in active mode, and reach a current consumption as low as 0.4 uA in standby mode. The MSP430FR6989 Launchpad Development Kit is very helpful in the development of electronic projects, as it features an LCD display. This could come in handy if we wanted to be able to see the data collected on the microcontroller before that said data is sent to the development kit. This microcontroller features all three types of communication protocols discussed: I2C, UART, and SPI. Since the group members already have this launchpad as it has been required in previous classes, this would help the budget of our project, since this would enable us to prototype with this microcontroller without the need to purchase a new unit solely for prototyping. Another thing to note is that a group member already possesses a book that details MSP430 microcontroller basics. This book of information would be a big help when developing the microcontroller functions. Overall, the MSP430FR6989 is a good choice for our microcontroller, as it has extensive capabilities, and has been rather popular for use in senior design projects.

3.4.1.2 MSP430G2553

The MSP430G2553 is an ultra-low-power microcontroller, like the MSP430FR6989. This device includes a 16-bit RISC CPU and 16-bit registers. In addition, this microcontroller features a digitally controlled oscillator which is able to wake up the system from its low-power mode in less than 1 us. Like the MSP430FR6989, our group members have experience with this microprocessor from previous classes, hence this would be a substantial benefit to choosing this module opposed to modules that we have not had experience with. However, this module has less community support than the Arduino microcontroller. This would be a drawback when researching and developing the programs for the project. In active mode, the microprocessor consumes 230 uA at 1 MHz, and 2.2 V. In standby mode, this drops to 0.5 uA. Compared to the MSP430FR6989, its active mode is slightly worse, with 130 uA more in current consumption than the FR6989. With five different power saving options, this is a good pick for the microcontroller in our design. Similar to the MSP430FR6989, several group members already have the TI Launchpad kit with this microcontroller, so there would be no need to spend money on a new unit for prototyping.

3.4.1.3 ATMEL ATmega328P

This Arduino microcontroller has a key benefit of being widely popular. Because of its relevance in electronics projects, there are a lot of resources and online help that would assist greatly with the development of our MCU program. Another benefit is its simplicity. This microprocessor’s main market is projects where a simple and low-powered microcontroller is needed, particularly Arduino modules like the Arduino Uno board. Table 9 shows a comparison of several microcontrollers. Investigating the power consumption of ATmega328P, we see that in active mode, this microprocessor consumes a current of 200 uA at 1MHz. This is comparable to the MSP430FR6989, which has an active mode current of 230 uA. The power-down mode of the Arduino microcontroller consumes a current of 0.1 uA, while the power-save mode consumes only 0.75 uA. The low power mode for this module is slightly higher than the two MSP microcontrollers, but not by much. A significant downside to the ATmega328P option is the programming language. The programming language used for this microcontroller is basically a framework based upon the C++ programming language. The framework is slightly different from C++, with additional special methods and functions. This may pose some difficulty when developing the microcontroller’s functions and could potentially be harder to program than the microcontrollers we are familiar with.

Table 9 - Microcontroller comparison

	MSP430FR6989	MSP430G2553	ATmega328P
Cost	$9.43	$3.38	$2.63
GPIO pins	83	24	23
Speed	16 MHz	16 MHz	20 MHz
Architecture	16-bit	16-bit	8-bit
Memory	128 KB	16 KB	32 KB
Operating Voltage	1.8V ~ 3.6V	1.8V ~ 3.6V	1.8V ~ 5.5V
RAM Size	2 KB RAM	0.5 KB	2 KB SRAM
SPI	4	2	Yes
UART	2	1	UART/USART
I2C	2	1	Yes
Package	80-LQFP	20-PDIP	28-DIP

3.4.2 Microcontroller Selection

For our project, we have chosen to use the MSP430G2553. There were several key reasons why this became our selection. The first is the low-power mode option. Out of the three microcontroller options, this module had the second lowest current consumption when in standby mode, with a current of only 0.5 uA. This standby mode will become very important when the device is in operation. The microcontroller will be used to turn on the motor and close the door, read the bird feed level with the ultrasonic sensor occasionally, and play sounds to the external speaker. These tasks will only be executed infrequently, so it is important that the microcontroller does not drain much battery when it is idle. Another reason that we chose this microcontroller was its simplicity and our previous experience with it. We had experience with both of the MSP430 microcontrollers, but given the extent of its potential application, the MSP430FR6989's many pins and functions will be too excessive for our project. The MSP430G2553 has the perfect amount of communication protocols that we need, and excess pins in addition to those necessary, allowing us the option to implement additional features if time permits or needs arise. Additionally, the small package size, 20-PDIP, will make this board easy to create a PCB with and replace if need be. For our PCB board, there are several sites in consideration for the manufacturer of our board. One of these sites is JLCPCB. This site is considered to have a good turn-around time, and has the MSP430G2553 microcontroller and ATmega328P in stock. However, they do not have the MSP430FR6989 in stock. Other PCB manufacturing sites are known to have very long lead times, and so if we want the option of re-ordering PCBs due to manufacturing issues or design errors, the option of being able to order from JLCPCB is a positive. Finally, opposed to ATmega328P, in which the group members have little or no experience, the MSP430G2553 is a microcontroller that all group members are familiar with. The group members are also much more acquainted with the C programming language than the C++ language that the Arduino microcontroller utilizes. In conclusion, we concluded that the MSP430G2553 is the best option as our microcontroller of choice and provides us with the most optimal parameters for our smart bird feeder design.

3.5 Motor Control

An important feature of this device is the automatic door functionality. In order to open and close the door, a motor must be used in conjunction with the door. Several motors were considered for this purpose. The aspects of the motors that were compared were the cost, torque, speed, size, acceleration, and voltage/current requirements. For the cost, the budget estimated a price of approximately $12 for the motor. Considering that we want the door to close relatively slowly, so as to not harm any small animals, it is not required for the acceleration to be highly rated. Likewise, speed is not an important aspect of the motor chosen. However, the size of the motor is highly important, as it must be able to securely and easily integrate into the bird house design. A larger sized motor would become an obstacle, as installing it with the relatively-small bird house would be challenging. Finally, voltage and current specifications of the motor are also highly important. Since we are powering the device with a rechargeable battery and solar cells, we want to find the lowest-rated motor in terms of voltage and current, as to have the least impact on battery drain. Other characteristics that are less important when deciding on a motor are the lifetime and efficiency.

3.5.1 Motor Options

In determining the type of motor we will utilize in our smart bird feeder, the following subsections under this section will take a look at DC motors, stepper motors, and servo motors to determine which will work best for our implementation. By investigating and comparing the following motor options, we hope to identify the most adequate motor for closing and opening the hatch that will physically block the bird food from the squirrels.

3.5.1.1 DC Motors

A direct current, or DC, motor is a class of rotary electrical motors that uses direct current in order to provide mechanical energy in the form of rotation. Many DC motors rely on magnetic fields to convert this electrical energy into mechanical energy. The two DC motors that were considered were the brushless and brushed types. In a brushed DC motor, brushes are used to connect the power supply to the rotor assembly. Inside, there is a permanent magnet which creates a permanent magnetic field. When current passes through the coils inside, the rotating coils inside will become an electromagnet and interact with the permanent magnetic fields already established. The repulsion between the two magnetic fields then causes the shaft to rotate. Unlike brushed motors, brushless motors do not use brushes. While the moving parts in a brushed motor include the rotor and the brushes, the only moving component in a brushless motor is the rotor. The main difference between the two is that in brushless motors, there is a control input from the microcontroller. The motor also sends out reference signals taken from a sensor inside the motor. This extra complexity makes brushless motors far more difficult to control. However, brushless DC motors are simpler in the mechanical sense than brushed DC motors. Between the two types of DC motors, brushless motors tend to be more efficient and require less maintenance, since there are no brushes that need to be eventually replaced after wear.

3.5.1.2 Stepper Motors

A stepper motor is a brushless DC motor that divides its full rotation into equal steps and as a result, is valued for its higher precision than other types of motors. This type of motor does not require any feedback to a microcontroller. However, this motor does require an input of pulses in order to control the rotational position. This is called pulse-width modulation. This would require input from the microcontroller. In a stepper motor, the number of north and south poles correspond to the number of ‘steps’. As the number of poles increases, the angle of each step decreases. The stepper motor is a good candidate for a motor, as the step precision makes the motor very accurate when controlling. Another great aspect of the stepper motor is that at low speeds, the torque is higher. For our device, we need a motor that is precise, yet on the slower side, so the stepper motor would be great in this aspect. However, the need for microcontroller input increases the complexity. Another drawback is the current draw. The stepper motor will consume maximum current whether there is a large load or not. Any energy not applied to a load will be lost as heat. Also, when large loads are applied to the motor, the motor may skip steps as a result.

3.5.1.3 Servo Motors

Servo motors are highly energy-efficient and can offer a lot of torque in a small size. The servo motor has three input signals, the ground, signal, and positive voltage line. In order to control the motor, an electrical signal is sent through the signal line. This signal will generally come from the microcontroller, and will consist of pulse width modulation. The duty cycle of the signal determines the position to which the servo moves to. The signal line of the motor accepts the PWM signal, while the positive line is connected to the voltage source, and it provides the servo motor power. The servo motor works mainly by use of a potentiometer internal to the motor. Also inside the motor is a DC motor and control circuit. While the motor is turning, the potentiometer’s resistance will change as a result. This allows precise control of the direction and movement of the servo motor. This way of operation is similar to the stepper motor discussed earlier. Another key feature of the servo motor is its proportional control. This means that the speed that the motor turns is proportional to the distance between its current and desired position. In other words, if the motor is already close to the desired position, it will turn slower than if it was farther away from this desired position. The main benefit of the servo motor that sets it apart from other motors is its closed-loop design. This means that the motor will send feedback to the microcontroller for compensation. A big disadvantage with the servo motor, however, is that these motors are usually limited to around 180 degrees of rotation. This may pose a challenge when designing a door that can be closed with only 180 degrees of rotation.

3.5.2 Categorical Motor Comparison

Table 10 shows a categorical comparison of each type of motor option discussed so far. This table offers a head-to-head comparison of each motor's lifetime expectancy, the amount of torque that each type of motor has, along with each motor's efficiency, cost, typical voltage and current, and control complexity (whether or not an external logic board is required to be connected to the motor for proper operation). The brushed DC motor has the shortest life expectancy, while the rest of the motors all have a roughly equivalent life expectancy, at over 10,000 hours.

Table 10 - Types of motors comparison

	Brushed DC Motor	Brushless DC Motor	Stepper Motor	Servo Motor
Lifetime	1,000 to 3,000 hours	over 10,000 hours	over 10,000 hours	over 10,000 hours
Torque	0.49 mNm at 5VDC	0.021 Nm at 12VDC	490 mNm holding at 5VDC	0.148 Nm at 6VDC
Efficiency	75%-80%	70%-90%	~65%	over 85%
Cost	$3.22	$14.95	$26	$7.95
Voltage/Current (typical)	1.5 - 24V 4A at 3V	4 - 24V 5 - 15mA	1.68A at 2.8V	200mA - 10A
Control Complexity	No MCU needed	MCU needed for feedback	MCU needed to send pulses	MCU needed to send PWM and for feedback

3.5.3 Motor Selection

For the motor in our bird feeder design, we are going to choose a servo motor. This type of motor was mainly chosen due to its feedback option. Since our motor will be opening and closing a door many times, it may not be wise to have a DC motor where we cannot determine the exact position at which the motor is positioned. A servo motor also ties other motors for the highest lifetime, so we do not need to be concerned about the servo motor failing from normal use. This could lead to the DC motor not closing completely or overshooting the position that it needs to stop at. With a servo motor, we can have feedback which lets the microcontroller know exactly where the motor, and subsequently the door, is positioned. This will make sure that every time the door is utilized, it is completely opened or closed. The servo motor that will be used is the FEETECH FS90-FB Micro Servo. This motor only costs $7.95 per unit, and includes positional feedback. This feedback potentiometer allows direct positional measurement of the output.

3.5.3.SD2 Motor Selection

In Senior Design II, our team still liked the choice of motor type, but wanted to ensure the chosen motor was of the best available. Listed below in Table 11 are the servo motors we considered. The team's specifications were focused around voltage, weight, and price.

Table 11 - Servo motor comparison

	SG90	FS90-FB	FS0403
Weight	9g	13g	5g
Torque	1.8kg/cm	1.5kg/cm	0.7 kg/cm
Maximum rotational angle	180°	180°	120°
Cost	$2.75	$7.95	$5.50
Voltage/Current (typical)	4.8 (.2A)	6V (.2A)	5V (0.2A)
Speed	0.1 sec/60°	0.1 sec/60°	0.08 sec/60°
Size	32 x 32 mm	24 x 22 mm	20 x 20 mm

We decided to use the SG90 micro servo motor due to its cheap price point and similar specifications to the others. While it was the largest of the motors available, it had the most torque while still maintaining the same base current requirements.

3.6 Distance Sensing Technology

One feature that our device will have is the capability of taking a measurement of the amount of bird feeder in the bird feeder container. This feature is important for achieving our goal of low maintenance. Since the mobile app will inform the user that the bird food is low, the user will not have to constantly go outside and visually check the bird feeder themselves. This lowers the time the user must put towards maintenance.

3.6.1 Distance Sensing Options

The purpose of the first two subsections below is to describe our technical investigation, so that it helps in the comparison of the distance sensing options. The two options investigated and compared in this section are the ultrasonic sensor and infrared sensor. The third subsection under this subsection, is to build on our technical investigation and compare between the two to derive to the option that will be used in this project.

3.6.1.1 Ultrasonic Sensor

Ultrasonic sensors are very popular in electronic applications and have been used in past UCF engineering courses. This sensor is easily integrated with the MSP430G2 to provide distance sensing ability. The wiring to the HC-SR04 module includes pins for Vcc, Trig, Echo, and GND. In Figure 5, the entire module is shown, including the pins and the transmitter and receiver. Other key aspects of the sensor are included in Table 12.

The important aspect of the ultrasonic sensor is the minimum and maximum range range. This means the minimum/maximum range that the ultrasonic sensor is able to detect distances at. Since the bird house will be relatively small, the sensor will have to be able to read distances of about 1 – 13 inches, depending on the depth of the bird feed container in the final design. The sensor must be able to detect the levels of bird feed by looking down vertically on the bird feed from the ceiling of the bird house. When the bird feed is fully filled, the distance from the sensor to this level will be ~1 inch. Since the minimum range of the sensor is 2 cm, this will not present an issue for our device. When the bird feed level is depleted, the distance between the bottom of the bird house and the sensor will be ~13 inches. Again, since the maximum range is 4 meters, this design will be feasible. Another good aspect of the sensor is the small size. The relatively small dimensions of the sensor will make it easy to integrate into the bird house structure with the rest of the electronics.

The ultrasonic sensor works by sending out a signal from its transmitter and then receiving the reflected signal back to its receiver. The time between transmission and reception is used to calculate the distance between the sensor and the object. In order to control the ultrasonic sensor, a signal will be sent from the microcontroller to the Trig pin of the sensor. This signal will consist of a 10 us pulse. This commands the sensor to start its distance detection by transmitting an 8 cycle burst of ultrasound at 40 kHz. Once the ultrasonic sensor receives the reflected signal, it then produces a pulse that is sent to the microcontroller through the Echo pin. This pulse goes high as soon as the eight pulses are sent from the transmitter of the sensor. Once the pulses are reflected back, this pulse goes low. Therefore, the pulse width is proportional to the time between transmission and reception of the ultrasonic pulses. In order to calculate the distance, we must multiply the speed of sound, 0.034 cm/us, and the time duration of the echo pulse, then divide this product by two. The pulse is proportional to the distance traveled from the transmitter to the object, and back to the receiver. Since we only want the distance of the ultrasonic sensor to the object, we divide the product by two.

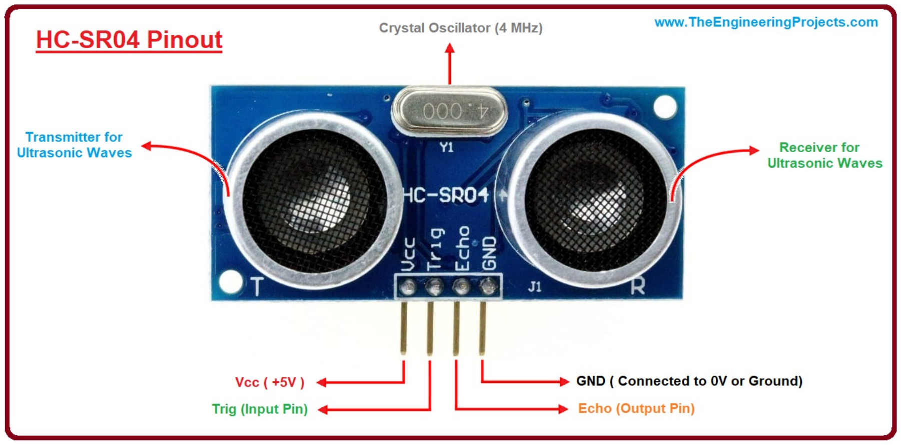

HC-SR04 Pinout
Crystal Oscillator (4 MHz)
www.TheEngineeringProjects.com
Transmitter for Ultrasonic Waves
Receiver for Ultrasonic Waves
Vcc (+5V)
Trig (Input Pin)
GND (Connected to 0V or Ground)
Echo (Output Pin)

Figure 5 - HC-SR04 Sensor [26]

Table 12 - Ultrasonic sensor specifications

Operating Voltage	5 V
Working Current	15 mA
Working Frequency	40 Hz
Maximum Range	4 m
Minimum Range	2 cm
Measuring Angle	15 degrees
Dimensions	45 x 20 x 15 mm

3.6.1.2 Infrared Sensor

The second type of distance sensing technology that was considered for our project was an infrared sensor. Infrared sensors work through the principle of triangulation. This means that the sensor measures the distance between itself and the object in question based on the angle of a reflected beam.

The process of distance detection works first with the IR sensor emitting a beam of light. The light beam is created using an LED. Next, this beam of light hits the object in front of the infrared sensor. This light beam is reflected off the object at an angle. The reflected light beam then travels back to the sensor, and lands on it at a different location. The sensor then uses this location where the beam was detected in order to determine the position of the object in relation to the sensor. Figure 6 displays the operation of the infrared sensor.

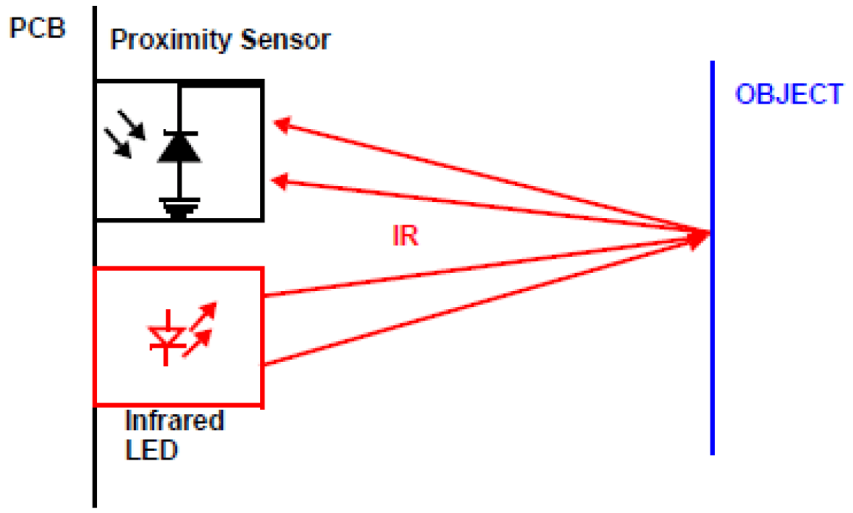


Figure 6 - Infrared LED to proximity sensor behavior diagram [25]

3.6.2 Distance Sensing Technology Selection

In order to determine which type of distance sensing technology is best for our application, we compare and contrast the two types discussed. The main advantage that IR sensors have over ultrasonic sensors is their ability to better detect edges of an area. A significant disadvantage of the IR sensor, however, is their inability to use them in sunlight. This is because of the interference that the sunlight causes. This can cause difficulty when trying to use the sensor outdoors or in the dark. Examining the main advantage of the IR sensor, we realize that this advantage would not apply to our project, since we do not need our sensor to determine the size of an area. Since our sensor will only need to know the distance, and will not need this aspect, the IR sensor is not necessary for this reason. Also, the huge disadvantage that IR sensors have when it comes to sensing in sunlight or night makes the ultrasonic sensor much more desirable. In conclusion, we will be utilizing the HC-SR04 ultrasonic sensor in order to measure the bird feed level in our project.

3.6.2.SD2 Distance Sensing Technology Selection

During Senior Design II, the team's perspective on the best sensor for our use did not change, but we did create Table 13 to better highlight the advantages or disadvantages of the decision.

Table 13 - Ultrasonic Sensor v. Infrared Sensor

	Ultrasonic Sensor	**Infrared Sensor**
Strengths	● Distance measurement ● Works well in dark environments ● Low power consumption	● Small devices ● Works well in dark environments ● Can measure complex objects
Drawbacks	● Distance constraints ● Unsuitable for fast objects	● Minimum detection range too large ● More expensive in cost than the ultrasonic sensor

The team also performed more research on the type of ultrasonic sensor we'd use for our design. The HC-SR04 was the first option as it was already available to us and fit our requirements. However research suggested that there was a Grove sensor that performed slightly better than our chosen sensor. The specifications are listed below in Table 14.

The specification that the team was most impressed with regarding the Groove were the current requirements. At 8 mA, this was half the value of the HC ultrasonic sensor. The other metrics were largely the same. However, the HC sensor was already owned by the group, and thus the group was familiar with how to code for it.

Table 14 - HC-SR04 sensor v. Grove IR sensor

	HC-SR04	Grove
Operating Voltage	5 V	5 V
Working Current	15 mA	8 mA
Working Frequency	40 kHz	40 kHz
Maximum Range	4 m	3.5 m
Minimum Range	2 cm	3 cm
Measuring Angle	15 degrees	15 degrees
Dimensions	45 x 20 x 15 mm	50 x 25 x 16 mm
Accuracy	3mm	2 mm

We began testing early with the HC sensor and didn't find the benefits to switching to the lower power sensor to be high enough given how small the change was when looking at our total energy requirements.

3.7 Solar Energy

Solar energy will allow us to reduce maintenance for our system, while also being mindful of the environment. Solar energy will provide a clean, green source of power to our system devices. Our solar energy system will consist of solar cells that convert the solar energy into electrical energy, a rechargeable battery which will store this electrical energy, and a charge controller placed between the solar cells and battery that will protect the battery from overcharging and overvoltage. The charge controller will also increase the life-span of the battery due to its capabilities. In this following section we outline the research and selection of the charge controller, rechargeable battery, and solar cells. We also do some calculations to determine the amount of charging time required to fully charge a battery.

3.7.SD2 Solar Energy (SD2 Stretch Goal)

One of the most challenging problems the group faced in Senior Design II was how to maximize the power potential of The Smart Bird Feeder. In SD1 the group was under the impression that the best way to do this would be solar energy. However in SD2, we realized the real issue regarding the project was the amount of energy consumed by the device. Our power requirements were high enough that adding a solar energy panel would not be enough to offset consumption by any meaningful amount. So we shifted our focus by including a more powerful battery and by attempting to limit the consumption of the Jetson Nano Developer Kit. We'll speak about this more in the coming sections more related to these changes. But for now solar energy has become a stretch goal as the team does not find it key to the success of the project.

3.7.1 Charge Controller (SD2 Stretch Goal)

The next consideration when creating our solar energy system is the type of charge controller. The purpose of a solar charge controller is to control and regulate the current flowing from the solar panel to the rechargeable battery. This prevents overcharging of the batteries. There are two main types of charge controllers. These are the PWM and MPPT type solar charge controllers. The main difference between these two is how the current is drawn out of the solar cells. With PWM controllers, the current is drawn out of the solar panel just above the battery's voltage. In contrast, with MPPT controllers, the current is drawn out of the solar cells at the solar panel's "maximum power voltage". This means that there is usually an increase in energy harvested using a MPPT controller over a PWM controller. Of course, with higher efficiency comes a higher cost. The PWM charge controllers are less expensive than MPPT controllers, and are ideal for smaller systems. The average price of a MPPT controller is around a few hundred dollars, while it is easy to find a PWM controller as low as $15. Since we are trying to achieve a low-cost system, we will be choosing the PWM type charge controller to regulate the charging of our rechargeable battery. For our initial testing, we will be choosing several charge controllers.

3.7.2 Solar Powered Battery (SD2 Stretch Goal)

Next, we must choose a rechargeable battery for our solar energy system. This battery will store the energy needed to power the components in our system. There are two main types of batteries that we can choose from. These are lithium and lead acid batteries. These are the two most popular types of rechargeable batteries, especially when it comes to solar energy systems. When it comes to capacity, depth of discharge, efficiency, and lifespan, lithium-ion batteries usually take the lead. Depth of discharge is the percentage of the battery that can be safely drained of energy without damaging the battery. Typically, lithium-ion batteries can drain 85% of the total capacity. However, lead acid batteries should not be discharged more than 50%. With efficiencies, lithium-ion batteries see around 95% efficiency, while lead acid batteries' efficiencies come closer to 80%. This means that lithium ion batteries can typically use ~15% more of their stored energy. In addition, higher efficiencies mean that lithium ion batteries will take less time to charge. This becomes very important when we want our system's power to rely mostly on solar energy. When looking at all the factors, it is clear to see that lithium ion batteries are the better choice. The difference in cost between lithium and lead acid batteries is relatively negligible with the size batteries we are looking for, and this will be worth the benefit of less maintenance of our system. For our initial design, we will be using two lithium ion rechargeable batteries found on Digikey.com. The batteries are rated at 3.7V and 2.5Ah each. The price of each battery is $14.95.

3.7.3 Solar Cells (SD2 Stretch Goal)

When we are choosing a solar cell, charge controller, and battery to power our system, there are several key aspects to consider. For the solar cell, we must choose which type of solar cells we want. The two most popular types of solar cells under consideration are monocrystalline and polycrystalline solar cells. In order to create monocrystalline solar cells, silicon is formed into bars and then cut into wafers. The reason that this type of solar cell is called "monocrystalline" is due to the fact that it utilizes single-crystal silicon. Monocrystalline panels tend to be more efficient because of this, and as a result are more expensive.

Polycrystalline solar panels generally are less efficient than monocrystalline panels. These solar cells are also made of silicon, however, the silicon wafers are formed in a different process. This process involves melting many fragments of silicon together, resulting in a multi-crystalline structure. This multi-crystalline solar cell allows less freedom of movement than the single-crystalline solar cells, resulting in lower efficiency. Although the price is higher for monocrystalline than polycrysalline, we are choosing to go with the monocrystalline solar cells due to their higher efficiency. Monocrystalline solar cells are readily available online.

Let us see how long it would take to charge a rechargeable battery given a certain solar panel and PWM charge controller. We will need a maximum current of 4A, and a voltage of 6V. For our system, we will have two lithium ion batteries each with a voltage of 3.7V and an amperage of 2.5AH. Together, they will provide our system with 7.4 volts, and a maximum current of 5A. In order to calculate the total charge time we must divide the battery's capacity (in watt hours) by the power of the solar panel. We then multiply this quotient by 2. This leaves us with the total charge time. To our battery's parameters into watts/hours, we must multiply the amp-hours by the voltage of the battery. This leaves us with 5Ah * 7.4V = 37 Wh. For our solar panel, we are choosing two 5V, 250mA solar panels. Together, they provide 10V and 500mA. This gives us a power of 5 watts. Calculating everything together, we get approximately 37Wh/5W = 7.4 hours for the total time it will take to charge our two batteries with our solar panels. Of course, this is only an estimation, and extensive testing will have to be done to create the most efficient and best suited solar charging system for our smart bird feeder.

3.8.SD2 Replacement Battery System

As the original solar design was pushed to stretch goals following a new understanding of power requirements, the team needed a method of powering the PCB. The power requirements of the system are listed in section 5.1.2.SD2, however for the purpose of this section the maximum current for the system is 1.8 Amps. The minimum current after implementing the power saving features on the Jetson Nano is 1.2 Amps. The team needed a battery that outputs 12 Volts and carries with it as many mAh as possible.

The battery bank that the team decided on is the TalentCell rechargeable 12 V 6000mAh battery bank as it had a barrel jack port capable of functioning well with our PCB. Compared to the previous 3.7 V 2500 mAh batteries that would need to be placed in series, this option was much stronger in terms of supporting our power requirements.

3.9 Piezo Speaker

In our design, we are to have a speaker whose purpose is to help deter squirrels from The Smart Bird Feeder. For the speaker, we have decided to use a piezoelectric speaker. The piezoelectric speaker works by using voltage applied to a piezoelectric material. Piezoelectric materials are capable of converting electrical energy into mechanical energy. This means that when a voltage is applied to a piezoelectric material, stress will then be induced in it. In the case of the piezoelectric speaker, when voltage is applied to the metal diaphragm inside, a stress is induced, consequently causing the diaphragm to undergo tension or compression, depending on the polarity of the voltage applied. This tension and compression causes the diaphragm to dish in either direction, and produces sound as a result. The visual representation of this is shown in Figure 7, where you can see the direction of protrusion corresponds to the polarity of voltage applied. In order to control the piezoelectric speaker, we will be using the MSP430G2553 microcontroller. The microcontroller will send out a PWM, or pulse width modulation, signal on one of its output pins to the piezo speaker. Depending on the PWM signal, the piezo speaker will emit a certain pitched sound. There are also many benefits to using a piezoelectric speaker, including the low cost and small size, making this a good choice for our design. In conclusion, this is a great option for our design, and will achieve our speaker requirement.

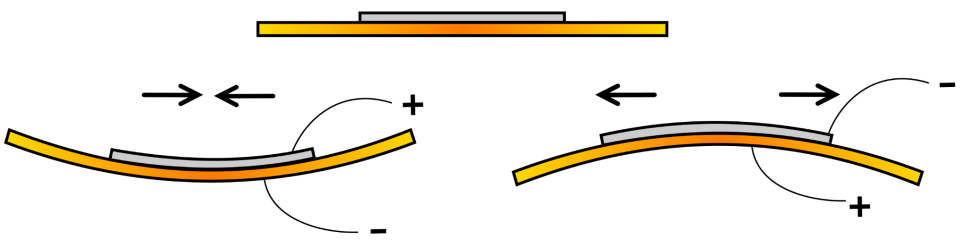


Figure 7 - Piezoelectric Effect [27]

3.9.SD2 Piezo Speaker

In Senior Design II, the group considered both 12 and 5 V alarms piezo alarms. 12 V alarms are considerably more common, however most of them carry a decibel range outside of our requirement range. Since the alarm will be in such close proximity to any backyard pest, the group wanted to remain cautious. The alarm the group selected is the Fielect Piezo Alarm. The specifications are listed below in Table 15.

Table 15 - DUDUA Piezo Alarm v. Fielect Piezo Alarm

	DUDUA Piezo Alarm	**Fielect Piezo Alarm**
Resonant Frequency	3.9 +/- 0.5KHz	4.0 +/- 0.5 KHz
Rated Voltage	12 V	5.0 V
Operating Voltage	3 ~ 24 V	1 ~ 30 V
Max Current Consumption	10 mA	5 mA
Min. Sound Output at Rated Voltage	95 dB	80 dB

However during late stages of the implementation stage, the Fielect Piezo alarm failed to provide sound close to its desired 80dB. The group tested other pre-owned alarms with the same circuitry and found that they functioned properly, so the issues with the Fielect alarm were manufacturing based in nature. After implementing the pre-owned alarms and the system functioning properly, the team decided to continue to utilize the pre-owned alarm, rather than purchasing another Fielect alarm.

3.10 Serial Communication Technologies

Communication between the Jetson Nano and our microcontroller is vital to the success of this device. The Jetson Nano must be able to send information to the microcontroller about when and how to operate its peripherals and collect data. In addition, the microcontroller must be able to send the data collected from the peripherals back to the Nano. In order to do this, a type of communication protocol must be decided on, which is common to both the Nano and the microcontroller. In the next few sections, we investigate these different protocols, and determine what is the best for our application.

3.10.1 I2C

I2C stands for inter-integrated-circuit. This is a serial communication protocol used for communication between modules and sensors. It is a bidirectional two-wire synchronous serial bus which only requires two wires to send/receive data, similar to the UART protocol. Since the I2C protocol uses an address system and a shared bus, it is able to allow many different devices to connect to the microcontroller using the same wires. The disadvantage of I2C is that it is slower than SPI, since there is less wiring.

The way in which I2C sends data from various devices on one line is by utilizing a SCL, or serial clock line, and SDA, serial data line acceptance. The SCL allows the master to send and receive data, while the SCL is the clock line for synchronizing the transmission of data. In order to send data, the master addresses the slave by sending the 7 or 10-bit address of the slave it wants to communicate with to all the slaves. The slaves then determine which one of them is the desired recipient, and then the correct slave returns an ‘acknowledge’ bit, which switches the SDA line low for one bit. The master is then free to send or receive data. Once all data has been transferred, the recipient will send another ACK (acknowledge) bit to verify that the transmission was successful.

A big drawback with I2C is the slower speed. Another disadvantage is the larger space taken up by the needed pull-up resistors. In conclusion, although this protocol is very helpful for systems with multiple slaves connecting to a master, this is not the setup in our project. Our device will only have the development kit communicating with the microcontroller, so the need for I2C does not exist.

3.10.2 UART

UART stands for Universal Asynchronous Reception and Transmission. This is a simple serial communication protocol that supports bi-directional communication, similar to the I2C. However, unlike SPI or I2C, UART is asynchronous, meaning that it is not synchronized with a clock signal. This protocol operates by having two data lines: a transmit line and receive line. By the way that UART is setup, communication can only occur between two devices.

Since there is no synchronization in UART, specific aspects of UART compensate for this. First, the two devices must be set to the same baud rate. This rate designates the speed of communication by defining the number of times a signal changes per second. Second, the UART adds start and stop bits to the messages in order to identify the start and end of messages sent and received. UART can operate in 3 different ways: simplex, half-duplex, and full-duplex. The half-duplex and full-duplex modes of operation both have data transferring in both directions, however full-duplex has the data transferred in each direction simultaneously, while half-duplex is not simultaneous. Since we need the Jetson Nano to send commands to the microcontroller, and the microcontroller to send information back to the Nano, we would be using either half-duplex or full-duplex for our UART communication.

The biggest advantage of UART is its simplicity. This communication protocol is very popular and widely used, so there is also plenty of documentation to support our development process. The lack of a clock signal also makes this protocol simpler to understand and develop. Finally, the inclusion of a parity bit helps with identifying errors when they occur.

A disadvantage of UART is that there is no possibility of having multiple master systems and slaves, as mentioned before. However, since our project will only call for communication between the development kit and the microcontroller, only one master-slave system will need a communication protocol. Thus, UART would work perfectly for this application. Some other disadvantages include the size of the data frame being limited to only 9 bits, and low transmission speeds. Low transmission speeds are not a big concern, considering that our engineering and marketing requirements don't call for quick communication and processes. Also, the data frame bit limit should not pose a problem.

3.10.3 SPI

SPI stands for Serial Peripheral Interface. This communication protocol is similar to I2C. SPI operates a full-duplex, meaning that data can be received and transferred simultaneously. Compared to I2C, SPI tends to run faster, due to its simpler protocol design.

SPI works by communicating via four different ports. When SPI is used in a multi-slave system, meaning multiple devices, each slave must have a different enable signal. This results in the SPI protocol being slightly more complicated than the I2C protocol. Since SPI is synchronous, it operates with a clock signal. This complicates the design of communication using SPI.

The four ports of SPI make this protocol more complicated than the UART protocol, however, still less complicated than the slave addressing system of I2C. A major advantage of this protocol is that it has the fastest communication among the three protocols. Unlike UART, this protocol does not have a start and stop bit, meaning that data is transferred continuously. A disadvantage of SPI is the lack of error check, whereas UART has the parity bit.

3.10.4 Serial Communication Protocol Conclusion

Comparing the three protocols, it is clear to see why UART is the best method for our application. The simplicity is ultimately what made the decision land on UART. This is also the reason for UART's widespread popularity for communication. Our microcontroller will be sending various information to the Jetson Nano. This includes the position of the motor, the bird feed level, and updates on the external speaker output. None of the information that will be sent to the development kit from the microcontroller will call for a clock signal, as the data being sent will all be simple numbers in binary form. Both I2C and SPI include clock signals in their protocols, and this is something that is simply unnecessary for our project. In conclusion, UART was found to be the most suitable choice for our bird feeder.

3.11 Full Stack Development Technologies

There are a number of different full stack development tools to choose from when choosing to implement a mobile application. Choosing which stack to build the application on top of is a process that's dependent on the needs of the developer. Stacks can be broken down in terms of an operating system, database, web server, and programming language. It's important for our project to pick a stack that is well documented, with components that have strong ties together. In the coming paragraphs, the choices will be compared for the success of the project. Then we'll break down the technologies chosen and their usefulness to the project.

3.11.1 Flutter

Flutter is a tool from Google that's based on the dart programming language. The members of our team do not know how to code in Dart, so our team took that into consideration when choosing a stack. However, coding in Dart is known to be easily understandable. Flutter uses built-in widgets and has quick performance. However, due to the young age of Flutter, there is less documentation online of using the tool. When considering the lack of familiarity and documentation on Flutter, our team is less likely to incorporate this over other stacks.

3.11.2 React Native

React Native is based on JavaScript, which is a highly known and well-supported programming language that our team already is proficient in. React Native is extremely common for building applications, and thus has the most documentation of the Stacks listed here. Our team also has experience coding in React Native from the Project of Object Oriented Software course at UCF. React Native is part of the MERN stack, which consists of MongoDB, Express, React, and Node. This stack allows the user to code in JavaScript on both the front end and back end of the application. Which means there is less programming language overhead when implementing the application. The previous experience with React Native and its MERN stack counterparts make React Native an excellent choice for our project.

3.11.3 SQL and NoSQL Databases

When choosing a database, it's important to consider the difference between SQL and NoSQL databases. SQL databases are relational, and as such they require that in order for data to be inserted, you must predefine the structure of the data before inserting anything inside. NoSQL databases however are much more flexible, the data can be stored in a number of ways, allowing for a more dynamic approach. For our project, we prefer a more dynamic approach to memory storage and prefer not to waste time defining the structure for our user's data. We are also more comfortable using a NOSQL database when implementing a React application, so it was an obvious choice for our project.

MongoDB is a database associated with the MERN stack and is a NoSQL database, and works well with React based applications. The cluster is easy to configure, and adding data due to the dynamic structure is efficient. MongoDB also has a tool called MongoDB Compass, which allows its developer to visualize the data that's been added, which will be very helpful in our efforts to test and debug test data.

3.11.4 Expo

Our team will be utilizing Expo to test our application. Expo is a framework that allows developers to test React applications with ease. Expo apps can be tested in conjunction with React native as soon as they're written just by saving progress. Expo will run a live server that can be used to immediately see changes in a browser to mimic mobile views, or there is an associated application that can scan a generated barcode and updates right on the phone. By using expo, testing changes can be done quickly and seamlessly.

3.11.5 Visual Studio Code

Our team's editor to build the bird feeding application is Visual Studio (VS) code, which is an open-source editor created by Microsoft. The editor supports extensions for the programming languages needed for the project, and includes useful features such as syntax highlighting, bracket-matching, auto-indentation, and more. This code editor should improve the team's efficiency for the development process.

3.11.6 Operating System

This section will break down the choices of operating systems, and discuss the differences between the two, and ultimately our decision to develop on which platform. The two platforms that form a monopoly on the application market are Android and iOS. Android is a mobile operating system, and is primarily formed on a modified version of the Linux Kernel. iOS is a proprietary mobile operating system that's developed by Apple with a much more strict base. This decision will be based on a multitude of factors that affect the development process and user experience after the launch of the product. For the developmental side, the team must consider whether or not the full stack development technologies are more catered to a certain platform. We also need to consider our coding experience. regarding the languages primarily used for the platforms and how this could affect development time. Finally, the extraneous startup cost and process of implementing the app into the store for download. For the user side, we need to consider the percentage of users who would have access to the application and the entry fee associated with using the product. Another key factor is the feature set available to both platforms. Finally, the team will discuss the possibility of incorporating the application onto multiple platforms if given the time to do so, but there will be a priority assigned to one of the mobile platforms.

The first consideration is whether or not the development platform is supported by the full stack technologies that we've decided to use. Our team is using React Native as a mobile development platform for reasons listed in section 3.5.2. React native supports both Android and iOS, but there are limitations to testing the application depending on what OS is being used to use react native. For example, when programming with a windows machine, there is no solid way to ensure that the application is functioning on iOS, iOS emulation has a low toolset and is finicky on windows machines. There is a tool called XCode which simulates an apple device, but this can not be installed unless you are on a macOS to begin with. When using android studio however, which is available on both macOS and windows, you can freely emulate an android device. Our project members have a mix of windows and macOS, but when developing on android, there are no concerns regarding emulation or testing. While react native is supported, there are limitations depending on hardware.

Each platform uses a different coding language to achieve its functionality. Android primarily runs on Java, while iOS uses a native programming language called Swift. Java is more open ended and is used for many other coding practices, while swift is streamlined for iOS development. Swift is considered to be easy to write and read, and also fast and powerful. Java is simple, distributed to many other applications, and is based on object oriented programming. Both however have a varied toolset and are capable for the task at hand. Our team is more concerned about the time spent learning a new coding platform given they are both powerful tools capable of creating our application. In that sense, the team is much more familiar with Java, having used it to program other applications in the past. Java is also integral in coding with React Native. Simply choosing the coding platform that we're more familiar with can impact the development time greatly, so we are leaning towards Java for our coding language.

The last developmental factor is the process of adding the applications to the respective app stores. Consumers are used to downloading applications from their respective smartphone stores, as it feels familiar and more secure. So in order to correctly implement our application, it's important to make sure the download process is reliable and feels legitimate. Most Android applications are downloaded from the Google Play Store. In order to submit an application to the google play store, you have to pay a one time fee of 25$. This fee is associated with a Developer Dashboard account, which once owned can upload applications to the Google Play store for free. iOS applications are exclusively downloaded from the App Store, an apple software integral to the apple mobile phones. Unfortunately, the App Store has a much more stringent upload process, which must be approved. The fee for an application is also annual, sitting at 99$. This means that choosing iOS would include a much higher cost, and more development time for approval. This factor is quite large when comparing the two platforms for our team, as price has been a key factor in many of our group's decisions.

When considering user experience, we need to take into account the amount of devices capable of using the platform we develop on. When taking a look at the total market share, Apple and Android make up 99% of the smartphone market. Android controls a larger share of the market worldwide, at about 87% of the total market, to Apple's 13%. However, our primary audience will be in the United States, in which the numbers are much more in Apples favor. In the United States, Apple controls about 53% of the market, while Android sits at 46%. So, when considering the user base for the North American audience it'd be beneficial to choose IOS as the development platform. While we prefer to focus on cases where the users already have a smartphone capable device, it's important to highlight the cost of entry for these two platforms. Apple products are more expensive than Android products by about 2.5 times as much. The average Android device is around $254, while Apple sits at $687. In the case where a user would need to buy a device to use our product, android would be the better option. However, this scenario does not affect the decision as much as the current user base does.

When considering all of the topics listed above, the team needs to make a decision for which platform to set priority development on given the stringent time frame of the project. It is important to mention that the team would prefer to incorporate both development platforms when developing, but due to time constraints this may not be possible. If given the opportunity, both platforms will have an application in order to cater to the largest possible user base. Overall, the benefits in developing for Android include the ease of testing, code knowledge, and associated cost of uploading to a store. While iOS marks a slightly higher user base, this difference is not enough to overshadow the benefits of Android listed above. Thus, our team has decided to use Android for its developmental platform.

3.12.SD2 Motion Sensor Technology

In Senior Design II, as current consumption issues were made more pressing the group decided in order to save power there needed to be a way to limit the consumption on the Jetson Nano. Through running intensive current tests, the group learned that by shutting off the machine learning processor, the current consumption of the system as a whole dropped ~60%. To do this, a motion sensor needed to be implemented in order to trigger the machine learning sensor to restart.

3.12.1.SD2 Motion Sensor Comparison

There were two motion sensor technologies the team compared and considered to implement into the design, as Table 16 shows. A photo resistive sensor and laser were considered, acting as a tripwire for any animal who broke the line of sight between the laser and sensor. The other idea considered was a PIR sensor that used thermal technology to monitor motion of objects that produce heat.

Table 16 - Laser sensor v. PIR sensor

	Laser Sensor (transmitter + photoresistor)	**PIR Sensor**
Operating Voltage	3.3-5V	4.5V - 20 V
Working Current	< 40 mA	65 mA
Maximum Range	~ 3 m	3 - 7 m
Measuring Angle	N/A	120 degrees
Dimensions	6.5 x 18mm	24mm*32mm*25 mm

The team originally considered the tripwire sensor given it has lower power requirements, however there were a few issues regarding the use of this sensor. Given that both components would need to be located on the perch of the feeder, it’d be easier for the animals present to shift the laser or receiver, which would cause a stream of never ending false positives and turn the machine learning processor on constantly.

The team chose to use a PIR sensor to monitor movement, as it uses thermal technologies to monitor for changes in heat signatures. This technology would limit the number of false positives by avoiding other inanimate objects from triggering the sensor, such as a falling leaf. This design allowed the sensor to also be mostly hidden in the enclosure. The specifications table for both sensors considered is shown below.

4. Design Constraints and Standards

This section discusses the constraints on developing this Smart Bird Feeder as well as the standards that apply to the technology that will be applicable to this project. The idea behind the constraints subsections is to help highlight the various concerns and constraints that play a major part of our schedule and part selection decisions. Then we proceed to the subsections on standards to lay the foundation for the sets of standards and regulations that this group must be well aware of as we develop and pick various aspects of The Smart Bird Feeder.

4.1 Constraints

The group had to balance the added features of development processing boards with the cost of those boards. For example, there already exists a board with strong bird-detecting functionality built in, however, it sits at nearly double the cost of the alternative option. Thus, price constraints limit the capabilities of equipment and add on extra working hours. Another constraint is the amount of memory offered by the developer kit that we use, since any company that creates the AI developer kit that we plan to use would only offer a finite number of kit models with a finite number of memory size options. Only once we have finalized the hardware and software designs for this project will we truly know what the memory requirements for our developer kit will be.

Time constraints create an interesting dynamic between monetary cost and the feasibility of our group completing certain aspects of this project, given our skillset. In situations where we find the time frame to be too short for our group to feasibly learn how to develop and execute original solutions for accomplishing our goals, the monetary cost of components may increase, since pre-built products and packages may have a higher overall monetary cost than most original do-it-yourself components. However, in situations where we feel that we are able to accomplish our goals in the given time frame, we may choose to develop some hardware/software components ourselves, which would save monetary cost by purchasing less pre-built products. We expect each group member will spend at least an estimated 13 to 15 hours per week working on this project, and our current monetary budget is $400 (see *Project Budgeting and Financing* for details). Similarly, shipping times and part availability also need to be considered, as some parts may not be available to order when our group requires them, or if they are available, the shipping time cost may be unacceptable. With regards to our machine learning object detection model, if we need to train our own model, or even retrain an existing model, the amount of time that we are able to put toward training/retraining a model will be a significant constraint on the number of bird species that can be detected, since it takes more time and resources to find data and train a model and the amount of data to find and use for training increases.

Since the smart bird feeder device belongs outdoors, power also stands as a concern. Our plans were to avoid the user recharging the device inside, if possible, and to be able to power the entire smart bird feeder device for up to 3 hours or more on a single charge. We are currently investigating self-charging options such as solar power. Also, the battery supply must be large enough to power the processor and peripherals, and this total power needed could only be determined after the system components have been chosen. With the size of our bird feeder, and the total power consumption, a constraint arose when trying to incorporate a battery that was powerful enough for our system, while also being small enough to physically attach to our device. To mitigate this concern, we looked into low-power modes for the processor so that power consumption can be reduced, but more investigation is required to know how much power and current the entire smart bird feeder device will draw, which will reveal to us how large our battery must be. Environmental constraints are also produced as a result of our smart bird feeder device being made for the outdoors. Our bird feeder and the electronics within it must be fitted to withstand the weather conditions common to the area in or around Orlando. This means that the device should be waterproof, substantially shock-proof, and overall able to avoid any damage or loss of materials due to external weather conditions which could cause corrosion, shorts circuits, etc.

Other variables that will surely be constraints for this project are the birds and squirrels that will interact with the bird feeder. We cannot directly control most, if any, of the actions of live birds or squirrels, so we will be under a constant constraint in terms of testing and demonstration which is controlled by the personal choices of the birds and squirrels. They must first choose to make their way over to the bird feeder before our device can be tested.

4.1.1 Economic Constraints

In designing The Smart Bird Feeder, one of the many constraints comes from the cost of construction. Due to the common situation most groups face in Senior Design is that there is no corporation funding the project, which leads to the project being self funded by the group members. When companies fund Senior Design projects, they usually provide a substantial amount of money to ensure that the end result. is highly reliable and works as intended. However, when projects are self funded, such as the Smart Bird Feeder, we must use our budget constraint as a motivation to be frugal yet inspire us to ensure the end product has the most bang for its buck. This means, we need to ensure that we spend adequate time in technical investigation and comparisons to ensure that we identify the most important components of our system so that we save money on non important parts so that we can invest in the important components. For instance, we may go with a 2GB Development Board that has one camera port vs a 4GB Development Board that has two camera ports, since we know that the 2GB can work adequately to identify species and we only require one camera, which will save us a substantial amount of money. Moreover, the group will also have the burden of buying various components to test, in the creation of our prototype, so that we can have an end product that does what it needs to do.

In continuation the economic constraint, another very important reason to keep the manufacturing cost low, is for the marketability of our product to show that it could be made into a real product to rival other bird feeders in the market space. The Smart Bird Feeder must be kept at an overall low cost, so that it is affordable and reasonably priced when compared to other smart bird feeders. The reason for this is so that our bird feeder can be proof that if we were to ever go to market with a finalized product, it could be affordable to potential customers. This means that even if we have extra money to afford the highest priced parts, we must consider that we want to have an affordable end product that has a very good balance between cost and quality/reliability.

4.1.2 Environmental Constraints

Another highly important constraint comes from the fact that it is important for our bird feeder to be as environmentally friendly as possible, yet still have all its features that make it so smart. Renewable energy has been becoming a highly sought after want in the market, for obvious reasons when it comes to aiming for a more sustainable society. One of the main reasons for our group to try to incorporate a solar panel to have renewable energy is so that it helps us ensure that The Smart Bird Feeder is portable to different parts of a yard without the concern of reaching a power outlet. The idea for utilizing solar energy came from a similar concern that was brought up when deciding if the internet connection should be wired or wireless. The debate for incorporating solar panels was to ensure that the system was hassle free, and does not have to rely on the customer being constrained by the access to power outlets. However, the constraints brought about from this decision make our product dependent on ensuring that it is placed in a spot that gets an adequate amount of sunlight. Moreover, not only will the bird feeder be dependent on being placed in a well lit place, but it will also require a battery pack to store that energy so that power can be consistent during various times of the day. In addition, the incorporation of a solar panel and battery will undoubtedly increase the cost of the overall product.

With this all said, the constraints put upon this project is not all on the desire of utilizing solar power, but as well as ensuring that the bird feeder does not harm the wildlife. Due to the fact that our end product is responsible for feeding birds, we must ensure that every aspect from the type of seeds/food dispersed to the squirrel deterring hatch and sounds are safe and do not harm any visiting animals. Two of the main concerns for The Smart Bird Feeder, is the randomized noises we use to deter squirrels does not have any negative effects such as hurting their ears, the second concern comes from the use of the physical barrier hatch that will cover the food when a squirrel is detected by the object detecting Development Board does not in any way physically harm the animals by closing on them. These two concerns are of high priority.

As usual, while we share many of the same concerns as our potential customers, it is obvious that these constraints are also fueled by the marketability of our product. For instance, implementing solar power is highly marketable as the craze for renewable energy is ever growing, however this constraint also puts other constraints on our products price and dependency on sunlight. Overall, while environmental impacts on our Smart Bird Feeder puts many constraints on the design and building, it also is highly important to us as a group, since we want to make sure that our product is safe to the environment it is in.

4.1.3 Social Constraints

The objective of The Smart Bird Feeder is to provide an interactive smart bird feeder that caters to anyone from professional bird watchers to someone that just wants to passively observe birds eat. This goal of having our bird feeder available to a large variety of customers, puts the constraint that ensures we make the product as affordable as possible, so that it is not limited to specific customers. Another possible social constraint comes from the potential push back of people that believe our product is against squirrels. Thus, placing the constraint that we need to specify that we want our users to get the same joy of feeding and watching birds as we get, without the hassle of dealing with squirrels stealing the bird food and scaring the birds.

With this said, an additional major social constraint that comes to mind other than the fact that we need to ensure that it is affordable, but it is also easy to install and use. It is important for our bird feeder to be easily installed in different places in a yard. This means that this constraint of making it easily installable will require us to determine a technique to connect the bird feeder to a support platform of either a pole or some type of mount on a wall or tree. In addition to making the bird feeder attach to a support structure easily, it must also be lightweight so that it helps in making sure that it is easy for most users to install. In terms of the ease of use in software, we are going to have to ensure that it will have a very friendly user interface and experience. Therefore, things such as connecting The Smart Bird Feeder to the wireless internet and navigating to the photo memories and accessing the live feed must be very easy for any user, as it must appeal to a wide variety of users. These challenges described above that were placed by the social constraints will definitely test our ability to design a unique product that can conform to the needs and wants presented to us as a group developing a sought after bird feeder.

4.1.4 Political Constraints

As we make our design, build, and connect the various components of the Smart Bird Feeder we are continuously searching for possible political constraints. Moreover, it seems that the only potential political constraint could come from possibly having a political figure propose a law to prohibit bird feeding in some way, or use of cameras in yards. Other than that, we as a group have decided that as long as everything our product does is legal, we will not have any relevant political constraints placed upon the use of the Smart Bird feeder.

4.1.5 Ethical Constraints

As we develop our product, we plan to aim for the highest standards of safety precision, so that we can reduce any possible safety concerns when handling and using the Smart Bird Feeder. This means in the production of the system, we cannot cut corners on any aspect of the overall product that could affect the end users or wildlife. Therefore, we plan to ensure that the randomized noise is well within a safe zone, and that the materials utilized are not potentially toxic or negatively affect the longevity of the bird feeder. Thus, the materials sourced were to be well documented and be part of the technical investigation of comparisons between various materials to determine a good medium between price, safety, and quality. Similarly built off the environmental constraints, the ethical constraints encompass both environmental concerns with such matters as ensuring the squirrel proof hatch is safely implemented where it does close on an animal or hurt them, and concerns on quality and safety of the overall bird feeder. The ethical constraints are highly important for a variety of reasons, that span from basic concerns of ensuring that the product is high quality for its price, to the important concerns that it is safe for both end users and animals. For instance, when it comes to safety, we need to ensure that the material it is built out of (such as plexiglass) is constructed in such a way that there are no sharp edges where someone could get cut from. In terms of longevity and quality of the product, we need to ensure that the material it is built from does not fade or disintegrate at a relatively quick manner from being exposed to elements of the outdoors. Overall, there will be no features or safety measures cut out from the product for solely financial reasons or lack of care.

Another topic of consideration when considering possible ethical constraints, has to do with ensuring that our bird feeder does not violate or infringe on any existing patents. This ethical constraint put on us by patent protections means that patent owners have the right to stop someone like our group from potentially taking advantage of their patented inventions, which means our product could not be commercially made, used, imported, distributed, or sold without the patent owner’s permission/consent. Due to our group's concern over accidentally infringing on existing patents in the creation/production of the Smart Bird Feeder, we plan on taking it seriously and follow a comprehensive investigation of existing patents. While it may be highly doubted that there are any possible issues with patent protection, we will attempt to ensure that we do not copy or use any protected designs or concepts, without the proper permission and credit given to the inventors.

4.1.6 Health and Safety Constraints

In building on the constraints mentioned already, our Smart Bird Feeder needs to be safe for the end user and any potential animal which might interact with it. The health and safety of our end user and animals of the Smart Bird Feeder and those around them are our utmost priority. When considering the constraints that deal with health and safety, we see that there are various aspects that must be considered, not only for the end user handling the product but also for the animals that interact with them. One constraint that must be addressed in our design is that since animals will interact with the bird feeder, there is possibility that animal could get caught in the bird feeder by for example trying to go into the food reserve, so it is highly important that in our design we take precaution that our design does not have any crevices or areas where an animal could lodge themselves into. Furthermore, with the concern for both customer and animal usage, we must consider that all electrical components are placed in a way that they are a safe distance from harm. For example, the camera that is used to capture video and images of the visiting birds and squirrels must be protected from any animals from chewing it, eating it, or interacting with it in a way that could hurt the animal. In continuation with the concern on housing electrical components so they are out of reach from visiting animals, another constraint on the design and build has to do with ensuring there are not exposed electrical wires that could cause a hazardous environment. As many are well aware, any electrical wire that is loose and not hidden securely from animals, are at risk of being chewed on and potentially electrocuting the animal that is in contact with it. That is why we plan on ensuring that our design takes into consideration that any wires or electrical wires are placed in such a way that they are unable to be messed with by curious creatures. Also, as mentioned many times prior, we will need to investigate a safe material to use in building the housing of the bird feeder that cannot poison the animals or hurt in any way.

In continuation with the constraints that revolve around our number one priority of providing the utmost health and safety measures, we must allocate sufficient time for technical investigation and comparisons on various potentially hazardous devices within the Smart Bird Feeder. For example, lithium-ion battery fire hazards are a concern as they have “high energy densities coupled with flammable organic electrolyte” [17]. Therefore, not only will we need to design a bird feeder that keeps the electrical components at a reasonable temperature, but also consider that the battery must not be exposed to elevated temperatures that can cause a thermal runaway. Another constraint that will be put on the development of the bird feeder, is the technical investigation and comparison between speakers along with the safe zone of audible noise (such as decibel level) that the particular speaker can output to not damage the end user or animals’ ears when interacting with the Smart Bird Feeder. The hatch that keeps the squirrels from the food is another important concern in the design, so it is important that the motor closes the hatch at a slow enough speed to allow any animals from getting stuck between the hatch and the food disposal. Overall, the importance that we place on such things as the type of non-toxic safe build materials that we utilize, along with correct design implementation to keep electrical components hidden and secure with safe use of battery and noise level, will help in the marketability of the Smart Bird Feeder.

4.1.7 Manufacturability Constraints

In the development of the Smart Bird Feeder, there was an array of various manufacturing constraints. One of which has to do with the building materials and components available. For instance, we need to not only look into the material that will make up the walls and house the electrical parts which are safe and can handle the specific use case, but also be able to be at a reasonable price readability. For example, after investigating the technical differences in such aspects as durability, weight and longevity, we might consider making the bird feeder out of plexiglass or plywood but depending on which one is more available in nearby home improvement stores and the price point at which they are at. In addressing some constraints on the housing of electrical components such as battery, printed circuit board, and development board, we might consider using a 3-D printer at one of the various labs provided by the University of Central Florida to print a more custom design. This importance on the manufacturability constraints ties in with nearly all other constraints in one way or another from the time it takes to build, to price point.

The manufacturability constraints also apply to the implementation of electrical components in our Smart Bird Feeder. Due to many components that make up the Smart bird feeder such as the ultrasonic sensor, printed circuit board, and servo motors, there are various points along the building process that might be altered depending on availability and expected time to become available of particular items. For example, when we decide which microcontroller or servo motor, we plan on buying to implement in the Smart Bird Feeder, our team will have to consider which product to buy based on whether or not the particular piece of technology is available to us and how long it will take for the part to be expected, on top of all the technical other pros and cons.

In continuation with the manufacturability constraints placed upon us in developing a prototype Smart Bird Feeder, we must also think ahead on how these constraints could play in with our end product if we were to make this into a company to mass produce. While many of the constraints stay the same from the designing and building of the prototype to mass production, there will be some difference in manufacturability in terms of the number of parts in stock. For instance, if there are thousands of parts (such as motors) available from a supplier, it should be okay, however if a supplier only has a hundred or less parts in stock, then mass production of our product could be affected if there are not enough parts available. And in some cases, suppliers might sell package deals for entities that need more than what a hobbyist might want, so this might make some parts used in our prototype differ from the suppliers we use if we were to mass produce. However, as of now, we will continue on our design mainly based on the availability of the parts that can satisfy our prototype.

4.1.8 Sustainability Constraints

In the design and construction of the Smart Bird Feeder, we aim to have our product last a considerable amount of time as all the parts picked and structured are for the intention of having an end product that can withstand various weather conditions. Our intention when designing this bird feeder is to keep it as self-sustainable as possible, by harnessing solar energy, notifying users of food levels, and being built out of some of the most reliable materials.

In considering all the possible constraints, it has come to our knowledge that the sustainability of the electrical components such as the printable circuit board and development board will be the greatest concern. The reason for being concerned over the sustainability of the electrical components is due to the fact that we need to ensure that they are housed in a way where they do not get wet from rain, but also have ventilation for very hot days. Ideally in order for our Smart Bird Feeder to successfully be used by a vast variety of users, we need to have a system that can withstand very hot temperatures found around North America that reach over one hundred degrees Fahrenheit. In addition, the bird feeder must aim to be able to withstand freezing temperatures that are also found across North America, which can reach below zero degree Fahrenheit. Therefore, in the decision process of figuring out which parts to use within the system, the temperature range of the operating conditions must be considered as one of the top constraints. Moreover, the housing of the electrical units along with the solar panel roof and walls of the bird feeder will be exposed to the elements and need to withstand such things as erosion and strong winds. This means that the housing material must be very durable and fastened tightly to withstand winds that could potentially be a hundred miles per hour. Which essentially implies that the Smart Bird Feeder needs to be designed for hurricane conditions, however, it would be most likely recommended to remove the bird feeder from its outdoor conditions during hurricane 3 winds. The purpose for making the bird feeder aim to withstand large winds, is because this product should be able to be kept outside all year long with the only maintenance of ensuring adequate sunlight, wireless connection, and maintaining constant supply of bird seeds. Moreover, the camera that is used to connect to the development board in order to capture videos and images must be positioned in the bird feeder where it is protected from the elements and animals by a piece of glass or plastic for instance, but also still have clear visibility for relatively high quality video and images for detecting birds and squirrels. This means there will be a lot of technical investigation on the best implementations to overcome the constraints listed, so the housing of the electrical components can protect against various weather conditions and animals, and still be able to perform for a long time compared to the average smart bird feeders currently available.

In continuation with sustainability constraints, we also need to consider the software side of things. For instance, in order to keep our product up to date and usable for a relatively long time, we need to keep pushing updates to the application that interfaces with the Smart Bird Feeder. In response to the constraints placed on keeping the software sustainable, we need to ensure that we develop well built software that is relatively easy to maintain and push updates if need be. For example, as we might need to continuously update our application as smartphone operating systems (such as iOS) update, so that our app can continuously be used on the same phone for multiple years.

Looking at all the constraints already given, it is also important for our group to consider the longevity of the product design. For instance, it is not only how long the actual product will function, but how long our product will remain in the market. When it comes to the love of bird feeding and watching, it has been around for a very long time and will not go anywhere. Thus, if we ever want to take our product national with a company, it is important for our bird feeder to get the most bang for its buck by utilizing modern technology to connect our customers with their friendly backyard birds in a whole new experience.

4.1.9 Time Constraints

In comparison with all the constraints put upon in the development of the Smart Bird Feeder, it could be easily said that the time constraints are possibly one of the most stressful for the group. The reason for the concerns for the time constraints, is due to the fact that our group needs to have the bird feeder fully functional and proven that it works well outside before the end of Senior Design 2. This means that if we have one semester, around 4 months in Senior Design 2, to have completed the software and hardware side of things and have enough time to test it out sufficiently. While in Senior Design 1, we were able to fully design and investigate nearly all aspects of the product so that once we go in Senior Design 2, we already know what needs to be done and how. However, the main concern is the potential hurdles we might have to get over for things such as the time we might have to wait for the printable circuit board to come in, and actual assembly of the bird feeder.

One of the main time constraints that our group might run into in Senior Design 2 is the time it takes to receive, test and install the PCB after ordering it. It is said that the PCB often takes the longest amount of time, as most groups in previous semesters often have to order it twice if the first one does not work as expected. This is why we plan on placing orders on all components such as the PCB in the first week, so that by the third week we can expect to start collecting and testing them to verify that they work as planned. Thus, we need to ensure that we plan in our schedules wiggle room for additional time if we need more time for resigning and reordering the PCB if need be.

Due to the lack of prior building skills from our group, it will be necessary for the group to address this concern in our schedule for allowing group members to take sufficient time in building the bird feeder. Even though our project does not rely too heavily on prior building skills, it is necessary that we do not underestimate the time it takes in taking designs on paper and actually shape it out on the material we use to build. Which brings us to the next concern of this time constraint, that we might need to account for accessing the necessary tools needed to shape the various parts of the bird feeder,such as the entrance where the food comes out of it and the support pieces to keep the wall. fastened together.

The next constraint that affects our schedules is the time it takes to commute in order for everyone to meet and build the Smart Bird Feeder. This has to do with the fact that three of the four group members live relatively far (around 30 to 45 minutes away) from the University of Central Florida, making group meetings in-person potentially time consuming. In addition to having a long commute to the University, we also do not live relatively very close to one another making meetings outside of the University also potentially time consuming if we have to meet at one of the group members' home.

Last but not least, we will have to wait for an unknown amount of time for the birds to find the bird feeder. In doing some reading on feeding birds via bird feeders, there have been articles mentioning that depending on the design of the feeder, location, and local birds, it can take anywhere from a couple of hours to nearly a month before birds start associating the bird feeder with food for them. This means that we will have to predict an uncertain amount of time to add in our schedule for testing the final product to see if it works correctly or if there is anything that needs to be improved before the presentation at the end of Senior Design II. Therefore, it is planned for our group to take consideration of this situation so that we as a group aim to complete the build of the Smart Bird Feeder ahead of schedule so that we can have enough wiggle room for testing it out in a real world usage for as long as possible before presentations.

4.1.10 Testing/Presentation Constraints

In the development of the Smart Bird Feeder project, our group has a slightly different scenario than others as we are constrained by various reasons when it comes to testing and presenting our product. One of the things that makes our group unique when it comes to testing our product, is that the bird feeder needs to be securely set up outside in an adequate location. Moreover, one of the main features of our project is object detection to determine bird species or squirrels. When it comes to object detection, it already takes a long undetermined amount of time putting it through machine learning to be able to detect bird species on the computer screen, but we have to also prove that it works well with wild birds outside that approach the bird feeder, which makes it a lot harder to test. This means that we have to set up the bird feeder in the right spot, and wait for enough time to successfully attract multiple bird species that we have trained for, so that we can determine if the level of accuracy is satisfiable in our product. Also, even though our project is to keep squirrels away from eating bird food, we need to allow for squirrels to show up to the bird feeder so that we can observe and determine if our attempts to keep squirrels away is successful enough to present, or of there is any tweaking that might be needed to do before the presentation. Unlike many other groups with various projects, our group must overcome the fact that we are reliant on the arrival of birds and squirrels to test it out.

The team is also constrained by the fact that not everyone in the group has a yard to test the Smart Bird feeder. Our initial idea was that it could be an interesting experiment in our presentation to have various group members test out the bird feeder in their yard to see if there is any difference based on location, however, only Paul Amoruso owns a private yard to test it out. Therefore, unless we have enough testing time to locate public spaces that birds frequent, our smart bird feeder device will end up only being tested at Paul Amoruso's house.

Lastly, when it comes to the presentation, we are faced with an interesting situation as the proof of functionality is highly placed on extra documentation to prove it did what it was supposed to do. This means that we might have to have someone such as Paul Amoruso to passively observe the Smart Bird Feeder during the initial installation to document his observations of animal interactions with it and potentially add extra multimedia for additional content to present multiple points of view of the bird feeder in action. In other words, Paul may want to capture images or videos with his own camera of various angles of the birds and squeals interacting with it, so the judges of our project can get a more comprehensive idea of the way it looks in yards. We of course will present its functionality by presenting images of the notifications and thorough tour of the mobile application that contains live feed mode and library of birds labeled by their species name. Thus, placing the constraint that we must keep the Smart Bird Feeder installed for enough time to capture enough interactions, or until we are satisfied with the results to present.

4.2 Standards

Standards are in every part of our life, from what we wear to the technology we use to communicate with one another. That is why, when building a product to be utilized for consumers, it is important to be aware of the sets of standards and government regulations put upon us when designing the Smart Bird Feeder. When looking at some examples of standards, we see how for example the radio frequency (RF) signals are subjected to a frequency spectrum that helps prevent interference, when it comes to wireless communications. Therefore, the following subsections will intend on specifying the standards involved in the various aspects that go into developing The Smart Bird Feeder.

4.2.1 IEEE 802.11 Standard

IEEE, or Institute of Electrical and Electronics Engineers Standards Association, is a subdivision of IEEE which develops an array of standards for many industries. These industries include those related to power and energy, artificial intelligence, consumer electronics, and health care. In addition, these standards brought forth by IEEE developers go through accredited consensus development processes that bring many volunteers to represent many viewpoints and interest to finalize it. The IEEE 802.11 standard has the purpose of providing wireless connections for fixed, portable, and moving stations within a local area. Moreover, this standard also offers regulatory bodies a way of "standardizing access to one or more frequency bands for the purpose of local area communication" [3]. In looking at the scope of the standard, we see that it defines a medium access control known as MAC along with several physical layer (PHY) specifications for wireless connectivity. In looking at the purpose in further detail, in the context of the $IEEE\,802.11^{TM}$-compliant devices the standard describes such things as the functions & services needed by devices to operate in networks (that are independent, personal, and infrastructure) as well as the aspects of device mobility within the networks.

In connecting the usability of the IEEE 802.11 standard with the implementation of the Smart Bird Feeder, it is important that the description provided with additional material in the standard is understood as we implement wireless communication. Moreover, section four of the standard attempts to explain how wireless local area networks known as WLANs are different from the traditional wired LANs. For instance, the standard states that the design for wired LANs differs from wireless as its design implicitly assumes that an address is equivalent to a physical location, while this is not always the case in wireless networks as the addressable unit is a station (STA). In continuation with understanding the impact of a wireless network, the standard discusses the media impact on design and performance as PHYs used in this standard are different from the wired media. Moreover, this standard looks at PHYs implementation differences in things such as how it uses a medium that has no absolute or readily observable boundaries outside of which STAs with PHY transceivers are known to be unable to receive network frames, its unprotected from other signals sharing the medium, less reliable medium than wired, dynamic topologies, lack full connectivity, time-varying & asymmetric propagation properties, and might experience interference. Therefore, due to the limitations on wireless PHY ranges, WLANs intended to cover a reasonable geographical distance may be built from basic coverage building blocks. After the IEEE document provides a general description on the architecture, the document goes into further description on MAC service definition, layer management, DS SAP specification, PHY service specification, frame formats, MAC sublayer functional description, MLME, and Security to name a few.

With the Smart Bird Feeder highly dependent on wireless communications for reasons mentioned earlier in the sections on technical investigation, it is imperative that we recognize the importance of the IEEE 802.11 standard. For example, similar to the section in technical investigations, the standard ensures to mention important information about Wi-Fi such as the frequency range allocated by the regulatory which is described in 15.4.4.2 the operating frequency range is discussed as the DSSS PHY is allowed to operate in the frequency range 2.4 GHz to 2.4835 GHz allocated as in China, United States and Europe, while the 2.471 GHz to 2.497 GHz frequency band is allocated by the regulatory authority in Japan. Overall, wireless communication plays an important part in our project as it will allow the bird feeder to play live video, and send images and notifications to the user of the Smart Bird Feeder.

4.2.2 Battery Standard

The battery we decide to use in the Smart Bird Feeder will be part of the basis of what makes our bird feeder stand out from most electronic bird feeders, due to the batteries ability to keep charge in order to power the electronic devices that rely on it to perform the actions that make up the complete system. With this said, the battery standards must be observed. One of which is the IEEE Std 1013^{TM}-2019 standard, labeled as the "IEEE Recommended Practice for sizing Lead-Acid Batteries for Stand-Alone Photovoltaic (PV) Systems" [2]. The document on the standard provides a systematic approach for deciding an appropriate energy capacity of lead-acid batteries to appease the energy requirements of electrical loads of stand-alone photovoltaic (PV) systems. Photovoltaic is in reference to the prosecution of electrical current at the junctions of two substances exposed to light, thus referring to the use of some type of solar panel to produce an output voltage and current. Moreover, the standard helps to explain the battery usage within a system where the load exceeds the output of the photovoltaic array. Therefore, this will help as we aim to utilize a battery that will support our Smart Bird Feeder during times when the solar panel is not receiving enough sunlight to produce enough power to keep the electronic devices running.

The scope of this IEEE standard will help with the recommended practice that illustrates an approach for sizing vented & valve-regulated lead-acid batteries in stand-alone PV systems, installations, maintenance, safety, testing procedures, and the considerations of other battery types. As stated in the previous paragraph of the function of the battery in a PV system, the standard also describes the factors for a satisfactory PV battery system. These factors include the autonomy, load determination, battery capacity & functional-hour rate determination, determining number of series-connected cells, cell capacity & battery size determination, battery sizing worksheets, battery characteristics, and examples demonstrating various aspects of battery sizing. For example, the discussion on the establishing the system design requirements based on the length of time that the stand-alone PV systems load should be supported only via the fully charged battery is in the Autonomy clause of the standard. In other words, the Autonomy clause is on the topic of the battery being utilized when the solar panel is not outputting power. Moreover, the standard helps as the length of time that the system is able to run solely from battery power is required to be known as the load that the battery will need to support. In further investigation, by answering these questions of the dc load current & its duration within an autonomy period with no power provided to the PV array. we can calculate the overall duty cycle imposed on the battery. The maximum load (for example daily) is used to compare different battery sizes in order to ensure that the battery acquired for the Smart Bird feeder can sustain. In addition, we must also consider the parasitic loss from such things as tare losses of charge controllers and inverters which need to be included as part of the running-currents, as well as determining/tabulating the maximum & minimum voltage each load device needs to operate properly with the consideration of voltage drops for each load device for ensuring an minimum operating voltage.

In order for the needs of the Smart Bird Feeder to stay operational in the scenario where it relies on a battery that is charged by the sun, it is important for documents specifically such as IEEE Std 1013TM-2019 to be fully comprehended to ensure that the battery used in our system is adequate to sustain our electronic devices at an operational voltage so that they may perform. This standards document is important in explicitly describing factors that go into choosing the correct battery, such as the consideration of the temperature adjustment section that informs that battery capacity is affected by the operating temperature, with cell capacity rating in the United States generally standardized at 25 degrees Celsius. Therefore, the recommended practice in the document helps us to determine an efficient safe-sizing lead-acid battery in a stand-alone PV system. In picking a battery for the Smart Bird Feeder, we will ensure that we determine the duration that the battery will be solely used, the amount of current supplied by the battery over a certain duration, number of series connected cells, as well as considering maximum and minimum voltage along with temperature demands. In doing so, this standard will be seen in other sections that discuss the technical decision making of what battery we end up picking for the Smart Bird Feeder.

4.2.3 C Language Standards

Like every electronic device the Smart Bird Feeder depends on code to perform the correct actions. Therefore, it is important that the international standard that defines the C programming language, known as ISO/IEC 9899 is understood. ISO/IEC 9899 is a joint effort between ISO and IEC along with contributing countries which allow the standard to be available. The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) come together to form a specialized system. In the development of the International Standards, national bodies that are members of ISO and or IEC participate through technical committees established by organizations to deal with particular fields of technical activity. Moreover, other organizations such as governmental and non-governmental organizations are in communication with the ISO and IEC to participate in the work.

Due to the addition of new devices and an extended character set, new features will potentially be added to this international Standard. With certain features considered obsolete and may be considered to be taken out of future regions in the International standard. The International Standard is divided into four major subdivisions that contain the preliminary elements, characteristics of environments to translate and execute C programs, language syntax, constraints, semantics, and library facilities. The overall scope of this international standard is to “specify the form and establish the interpretation of programs written in the C programming language” [4]. This means that the standard will specify such things as the representation of C programs, syntax & constraints per interpreting C language, semantic rules for interpreting per interpreting C programs, representation of input data for processing by C programs, representation of output data produced by C programs, and restrictions & limits imposed by a conforming implementation of C. Even though many may overlook the standards when developing software, it is important to be aware of standards such as this, due to its direct link with the software section for the Smart Bird Feeder. It is also important to understand what the International Standard does not specify. Moreover, the standard has no specification on the mechanism by which the C programs are transformed or invoked for use by a data-processing system, the mechanism through which the input data is transformed for use by the C program, the mechanism through which the output data is transformed after being produced via C program, and the size or complexity of a program along with its data that will exceed the capacity of a particular data-processing system or the capacity of a processor.

In simplifying a particular benefit or goal of this International Standard, is that it allows and promotes the portability of C programs among a variety of data-processing systems as its intention is for utilization by programmers and implementers. Thus, being said, the importance of the International Standard will make impacts on the software design of the Smart Bird Feeder. The impacts will be seen as the necessary guidelines will be met in order to allow for definitions along with the regained importance for realization of such things as operand types to be used in import arithmetic calculations to ensure for correct values, so that the Smart Bird Feeder operates in the intended manner.

4.2.4 IPC PCB Standards

Due to the fact that an integral part of The Smart Bird Feeder is designing our own printed circuit board, it is integral to look into the association which produces PCB related standards. Therefore, it is important to note that IPC (formally called the Institute for Printed Circuits) is known as the Association Connecting Electronics Industries, which aims to ensure a standardization of the assembly and production requirements of electronic equipment and assembly. The non-profit, member-driven organization has members represented in nearly all aspects of the electronics industry that include design, printed board manufacturing, electronics assembly, and test. This means that the standards IPC puts forth apply with all printed circuit board types from single-sided, double-sided, the multilayer. The idea is that by the organization regulating most of the standards commercial PCBs need to comply with, it will help ensure reliability and longevity of the products. In giving an example of how far reaching the standards go, some of the standards include design specifications, material specifications, general documents, performance documents and material standards to name a few. In looking at the design standards that apply for printed boards, IPS labels IPC-2220-FAM as a family of specification for board design with standards labeled IPC-221, IPC-2222, IPC-2223, and IPC-2225. Thus, there is the IPC-221B as the generic standard on printed board design which prescribes general requirements for the design of printed boards and other forms of component mounting/interconnecting structures for single-layer, double-layer, and multilayer boards. Next in the family of standards for printed boards, is IPC-2222B to prescribe the explicit requirements for designing rigid organic printed boards with supporting the generic IPC-2221 standard. In conjunction with IPC-2221, IPC-2223 establishes explicit requirements on the design of flexible printed boards along with the forms of component mounting and interconnecting structures, as IPC-2223 handles Sections Design. Standard for Flexible Printed Boards. In addition to the list of standards, IPC-2225 helps to establish the requirements with the considerations of thermal, electrical, electromechanical and mechanical for the Sectional Design Standard for Organic Multichip Modules (MCM-L) and MCM-L Assemblies. As can be seen, IPC provides tons of standards from generic requirements on printed circuit board design in the software to requirements on the traces, thickness, tolerance, and material properties.

With the investigation of IPC PCB standards, it is evident that the standards brought forth by IPC should be followed, as they help ensure that the results desired are achievable by the design. With the standards put in place, it helps to ensure that the end product will have enough reliability along with quality to adequately compete in the marketplace. This means that to ensure performance and longevity, it is integral that these standards are complied with throughout the manufacturing process. This also allows for seamless communication as IPC establishes the language for the global electronic industry. Moreover, these standards help to eliminate confusion and help ensure designers that production of electronic assemblies meets adequate quality tests so there is no need for delays and that the production goes smoothly.

5. Smart Bird Feeder System Design

This section consists of detailed descriptions of the Smart Bird Feeder Design in all its aspects. In this section, we start off with the design of all the hardware components connecting to each other such as the PCB and Development Board to the servo motor, then the section goes further and displays a 3D representation of the bird feeder, and finally it is all tied together with a very detailed system design of the software and its aspects. In the software subsection, there will be descriptions for the functionality in order to cover the necessary capabilities and components of the application. Therefore, the user interface, login/signup, landing page, and memories page will be described in the subsections of this system design.

5.1 Hardware

Figure 8 shows a rough draft of the outline of the PCB and the peripherals and power supply that connect to it. This figure displays the connections from the solar panel to battery to linear control on the PCB, the connection to the servo motor, and the NVIDIA Jetson Nano Developer Board.

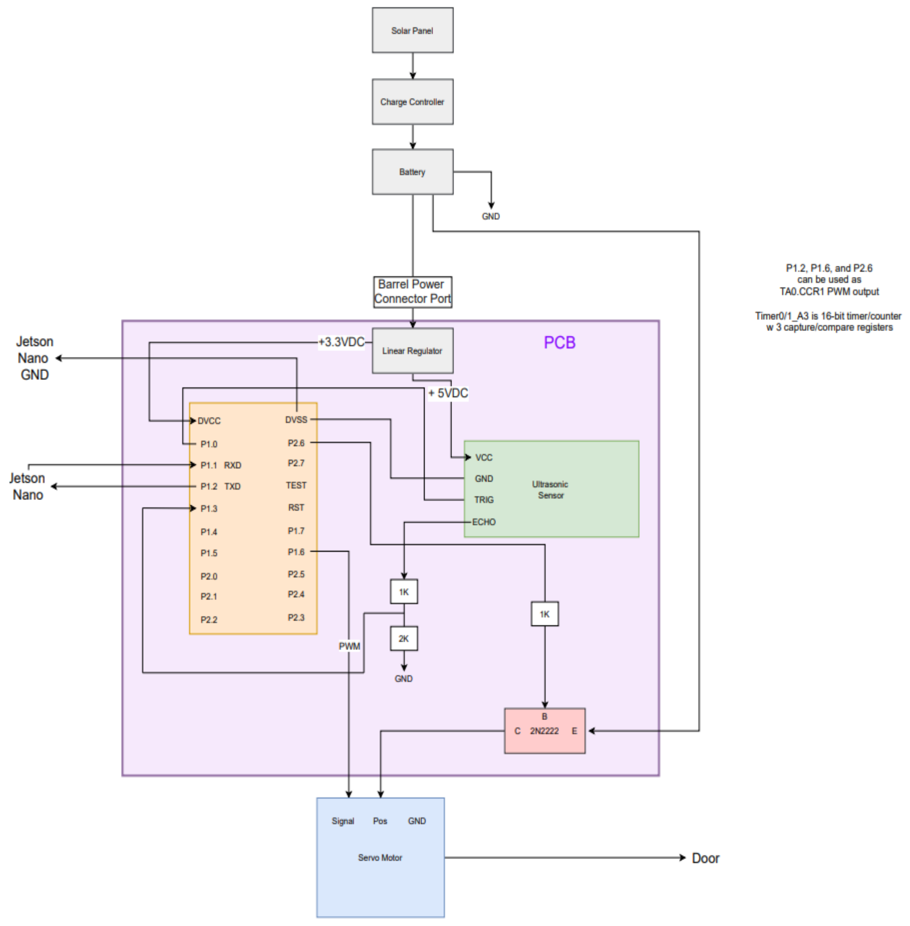


Figure 8 - Hardware block diagram schematic

5.1.SD2 Hardware

The team created an updated hardware block diagram to showcase a more accurate layout of components.

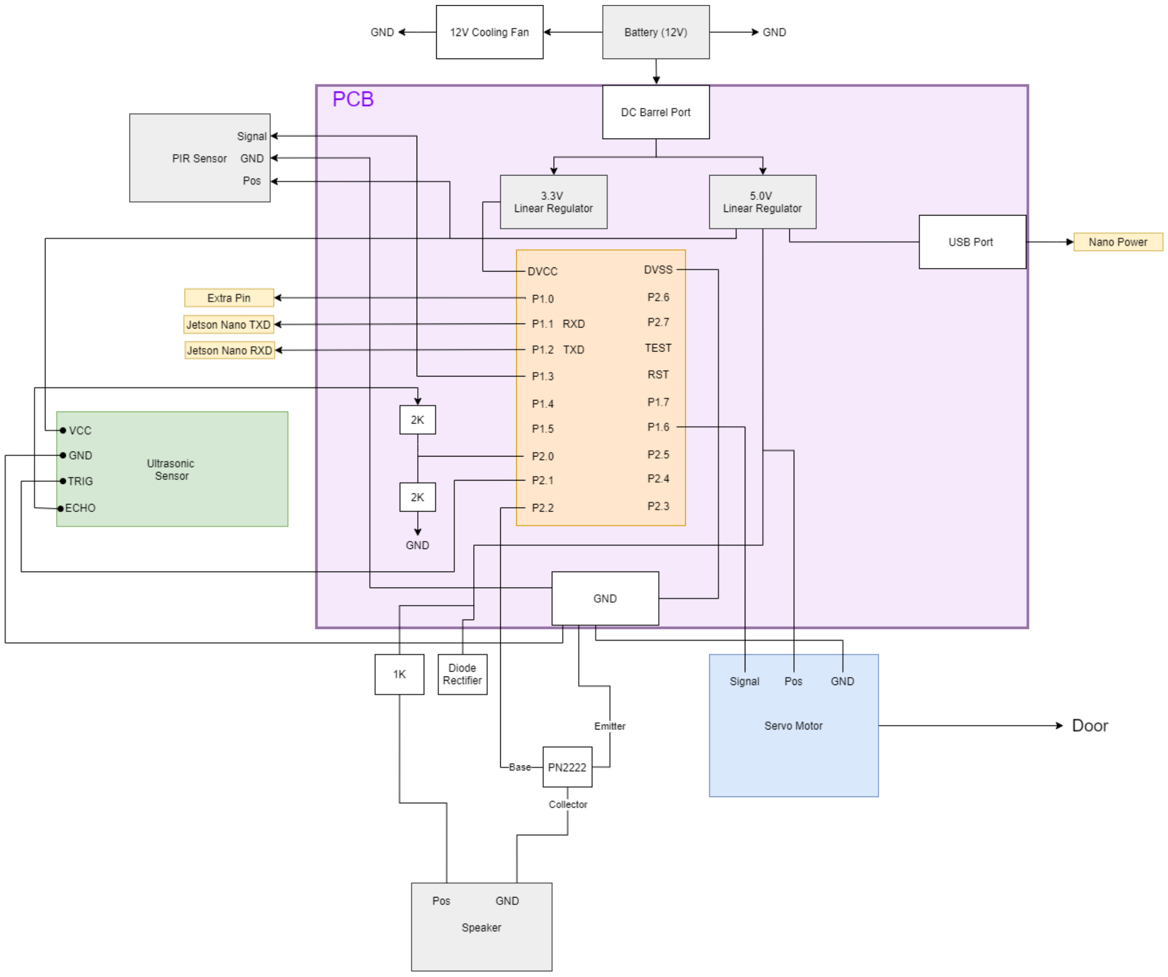


Figure 8.SD2 - Updated hardware block diagram schematic

The ultrasonic sensor was added as well as the Jetson Nano connection pins. The pin values have also been updated. The solar panel was moved to stretch goals and we added a new PIR sensor to save power. A cooling fan was added to offset the heat inside the enclosure due to the regulators and processor.

5.1.1 Birdhouse Design

Figure 9 shows a translucent view of the latest version of our 3D Computer-Aided Design (CAD) model for the smart bird feeder birdhouse. Our team's birdhouse design is inspired by a design that we found from a smart bird feeder project created by Google Engineers in the past. We used a CAD software called Onshape to create a 3D rendered design of the birdhouse.

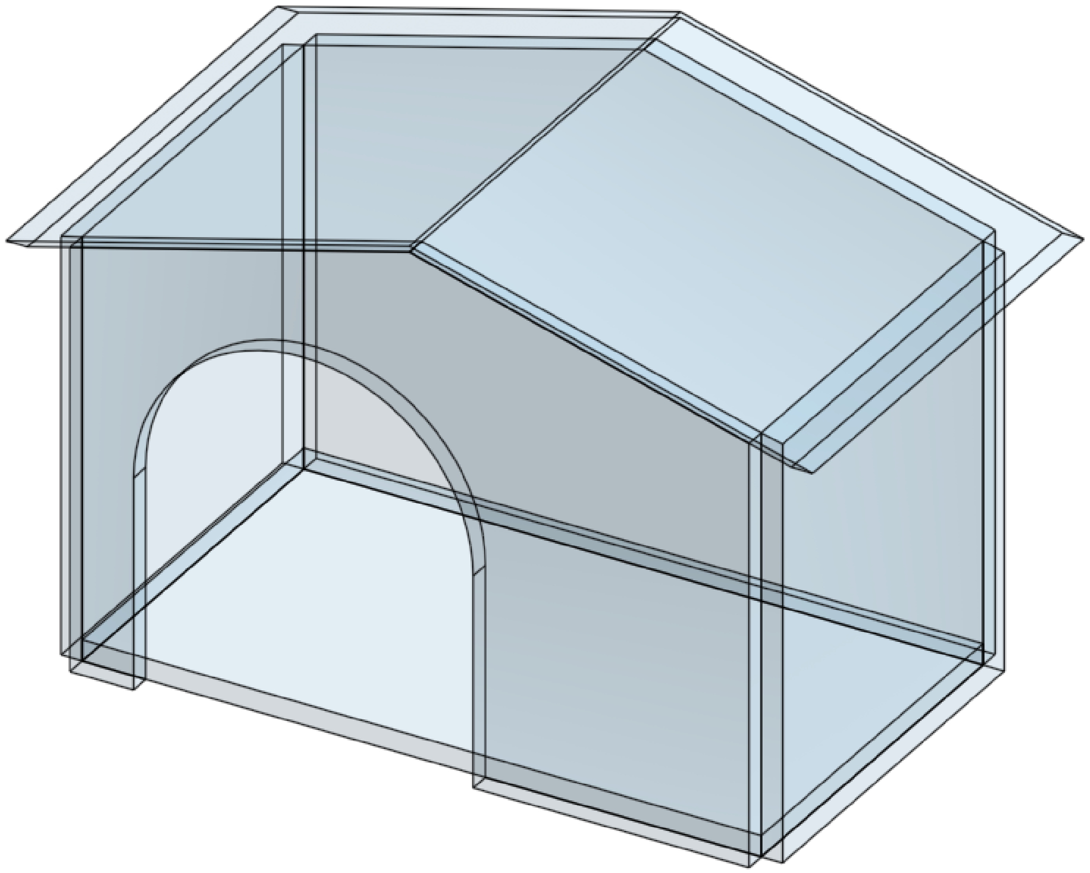


Figure 9 - 3D CAD render of birdhouse (Onshape)

Our team decided to make the birdhouse of the smart bird feeder a rectangular shape of dimensions 16" x 10" x 9.05", with a triangular-shaped sloped roof placed on top of the front, back, and side walls of the birdhouse. This was done to shield the innards of The Smart Bird Feeder from various weather conditions. A triangular shape for the roof was justified due to its sloped nature, which enables prevention of any buildup of additional mass (rain, snow, etc.) on top of the birdhouse, which if left unaddressed would weigh the birdhouse down and potentially cause structural damage to the birdhouse and electronics that are a part of our smart bird feeder device. As Figure 9 makes evident, there is plenty of space within the confines of this birdhouse for the remaining components of our smart bird feeder to fit into.

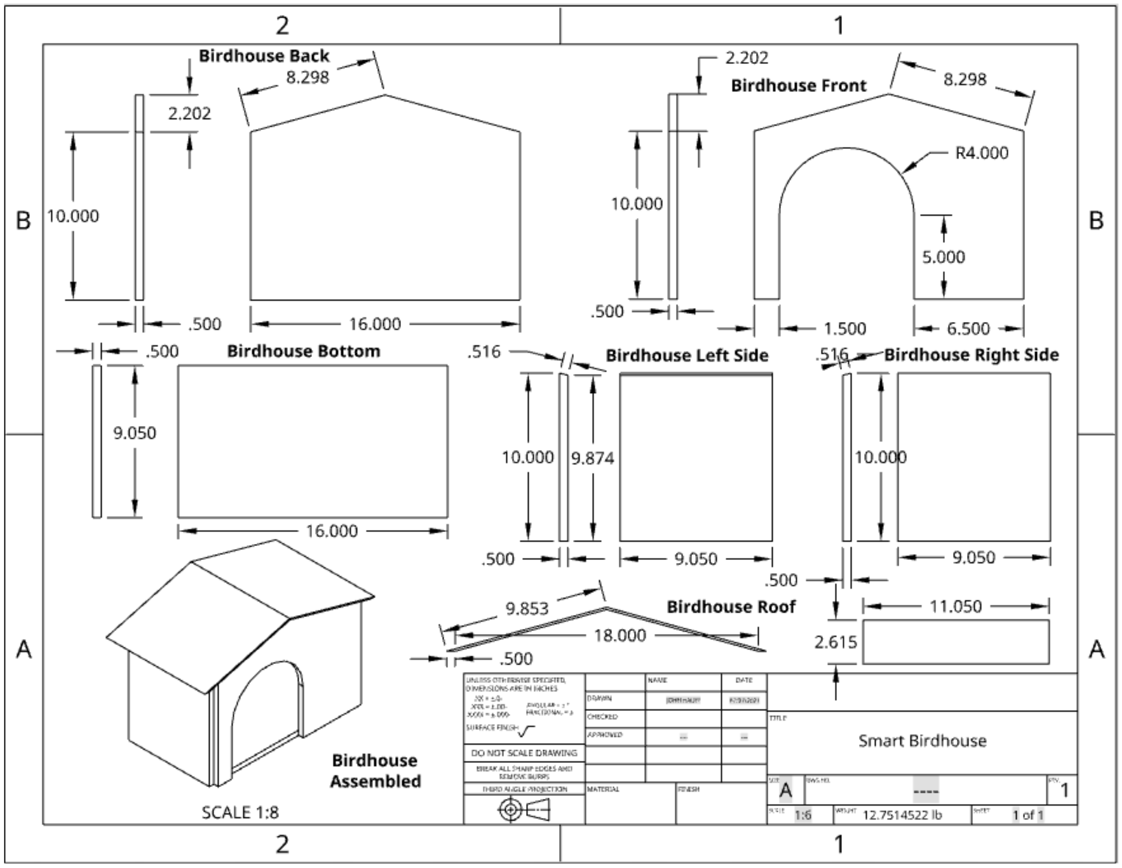


Figure 10 - Drawings of birdhouse structure[2]

Figure 10 displays drawings of the birdhouse design that our team created. From top left to bottom right, drawings paired with dimensions displaying the back, front, bottom, and left and right sides of the birdhouse structure are shown.

Our group plans to laser cut the necessary materials for the birdhouse, adhering as closely as possible to the dimensions shown in Figure 10. If time does not permit us to do this, we may instead choose to purchase a premade birdhouse and modify it to our needs. Therefore, the drawings shown in Figure 9 and Figure 10 are only initial drafts of what the final birdhouse dimensions and appearance will be.

5.1.1.SD2 Birdhouse Design

The birdhouse design changed significantly as the group began to experiment with the effectiveness of different designs. One of the first decisions made was to lessen the focus of mechanical design by beginning with a preconstructed bird feeder. This store-bought feeder is shown in Figure 11.

[2] Birdhouse dimensions were designed and measured in units of inches.

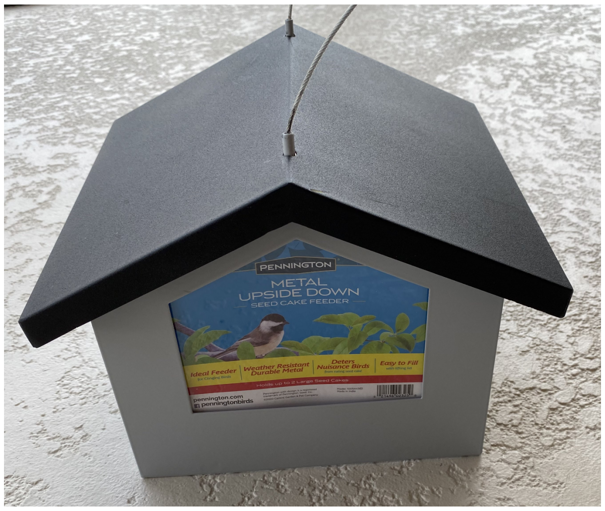


Figure 11 - Store bought feeder

There were multiple designs proposed to extend this pre-built bird feeder. The group considered many different hatch designs, and were constantly considering camera placement, trough design, and the storage of the electrical components. As shown in Figure 12, the design we created using CAD software and settled on included an external trough with an internal feed container, feeding chimney, and a perch for the birds to stand on. Although we acknowledge that more efficient or cost-saving designs may exist, because our discipline is ECE and not Mechanical Engineering, we did not prioritize perfection of the mechanical design of the bird feeder as much as we did for the electrical and software components of the device.

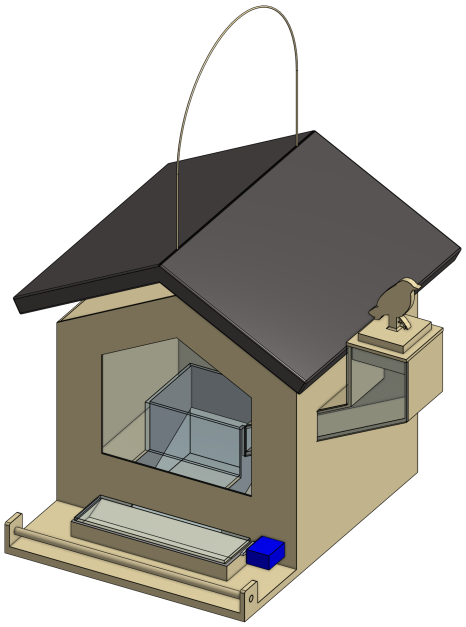


Figure 12 - Final bird feeder CAD mechanical design (Onshape)

5.1.2 Power Distribution

It is very important to determine the power distribution flow through the project. In order for our components to turn on, we must determine the correct power required for their operation. If we don’t supply the correct voltage/current, our system will not be able to operate. All of the components in the system will be consuming power. These components are the object-detection development kit, the microcontroller, the door motor, the ultrasonic sensor, and the speaker sensor. The development kit will be running in its 10W mode. This means that it will be requiring a voltage of 5V for operation, and could consume up to 2A for the board. Since the board will also have an on-board camera, this adds 1A of current consumption. The ultrasonic sensor, door motor, and speaker will also require 5V for operation. Finally, the microcontroller will require 3.3V, and will consume an active-mode current of 230uA. If all components are running, the total current that must be provided to the system will be approximately 4A. The full power requirements are listed in Table 17.

Table 17 - System hardware power requirements

	Voltage Requirement	Current Requirement
Development Kit	~5 V	2 - 3 A
MCU	1.8 - 3.6 V	Operating: 230 uA Standby: 0.5 uA
Servo Motor	4.8 - 6 V	500 mA
Speaker	5 V	4 - 6 mA
Ultrasonic Sensor	5 V	15 mA
Total	5 - 6 V	2.5 - 3.5 A

5.1.2.SD2 Power Distribution

Our team's understanding of system power requirements was one of the most significant changes between Senior Design I and Senior Design II. After testing the current requirements of all our components we came up with a much more accurate table, which allowed us to fine tune our power system, and look for solutions to the most power hungry device, the Jetson Nano Developer Kit.

Table 17.SD2 - Updated system hardware power requirements

	Voltage Requirement	Current Requirement
Developer Kit	~5 V	0.4 - 1.1 A
MCU	1.8 - 3.6 V	Operating: 230 uA Standby: 0.5 uA
Servo Motor	4.8 - 6 V	500 mA
Speaker	5 V	4 - 6 mA
Ultrasonic Sensor	5 V	15 mA
Fan	12 V	137 mA
PIR Sensor	5 V	65 mA
Total	3.6 - 12 V	1.2 - 1.8 A

After intensive testing the group concluded that the Jetson Nano consumes 1.1 A when it is running the machine learning module, however when the model is shut off, the current is reduced by ~60%. Since the nano accounts for a significant amount of the current consumption of the device, this became a key focal point for the group to focus on to reduce power consumption. However, given its integral to the design functionality of the feeder that the machine learning is always on, we needed a way to monitor when an animal arrived without using the machine learning module. We started prototyping a motion sensor that would allow the machine learning module to stay off unless prompted by the sensor. This would lower our current requirement for the majority of time and allow the team to meet its engineering requirement.

5.1.3 Voltage Regulators and Converters

In our system, the two voltages that will need to be supplied are 3.3V for the MCU and 5V for the rest of the components, including the development kit. Since our rechargeable battery will have a voltage higher than what is required in the system, we will have to utilize voltage converters. Since we need to convert the voltage from the battery to 5V and 3.3V, this will require two voltage converters. In the following sections, we will discuss the two types of voltage converters, our selection, and the design of the power regulation subsystem.

5.1.3.SD2 Voltage Regulators and Converters

In Senior Design 2, we extensively tested the switching regulators that were in the original design. We found that the regulators would output the correct voltages when no current was pulled. However, when a small current was pulled, these regulators were not able to output the correct voltages. Finding a solution to this issue proved difficult, due to the team's lack of experience with switching regulators. Utilizing the switching regulators was decided to be not necessary for the project. Instead, the team switched to using linear converters. We decided to use 3 converters in total. One converter is for 3.3V output. The other two converters are for outputting 5V. One of the 5V converters powers the Jetson Nano Developer Kit, while the other converter powers the rest of the peripherals which require 5V.

5.1.3.1 Linear vs Switching Regulators

There are two types of DC/DC converters: linear and switched. In a linear DC/DC converter, a linear component is used to regulate the output voltage. The control elements which step down the input voltage are normally arranged in a linear fashion, or in series. The main disadvantages of linear regulators are their poor efficiency compared to switching regulators, their higher heat generation, and their limitation to only step-down operations. In other words, the input voltage to a linear regulator must always be higher than the output voltage. Some advantages of the linear regulator include generally simple circuit configurations, need for fewer external parts, and lower noise.

In a switch regulator, a switching element is used to transform the input power into a pulsed output voltage. The basic operation works by turning on a MOSFET, or switch, on and off repeatedly. When the MOSFET is turned on, the regulator waits until the desired output voltage is achieved. Once this occurs, the MOSFET is turned off. This switching of the MOSFET repeats continuously, and is why this type of regulator is referred to as a 'switching' regulator. Switching regulators are generally more complex in design. These types of regulators also require the need for external parts, such as capacitors, inductors, and other elements. These are required in order to smooth the oscillating characteristic of the output power. However, switching regulators have some major advantages over the linear regulator. These advantages include their high efficiency, low heat generation, and options of both buck and boost conversions, in addition to negative voltage operation.

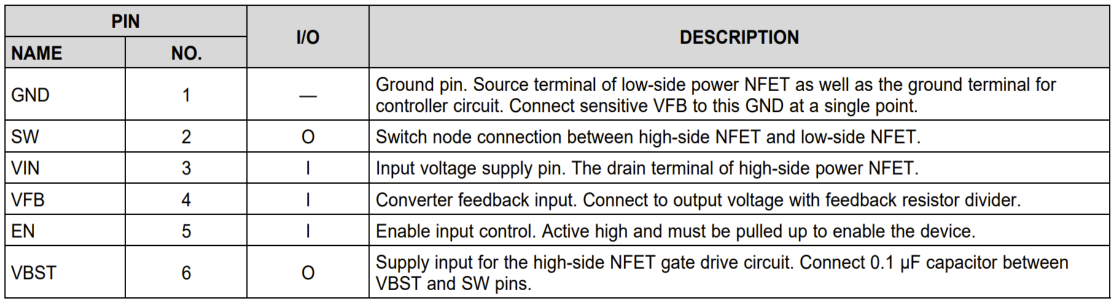
5.1.3.2 Voltage Converter Selection

For our design, we will be choosing to use switching regulators. The high efficiency of this type of converter is highly desirable. In addition, a team member has previous experience using switching regulators. This provides our team with more knowledge on switching regulators opposed to linear regulators, and will make the design process of the switching regulators less challenging.

5.1.3.3 Voltage Converter Design

For both the 3.3V and 5V step-down regulation circuits, we will be using the same module, the TPS565208DDCR. This buck converter is manufactured by Texas Instruments. Using the same module for both circuits is good for the budget of the project, and will also make the analysis and understanding of our circuitry less complicated. The TPS565208DDCR is a 4.5V to 17V Input, 5AA Synchronous Step-Down Voltage Regulator. Its package size is the SOT-23, a 184mm x 184mm module size. The most enticing features about this module is its optimized design for the purpose of minimum external components and low standby current. The output voltage range for this module is 0.76V to 7V, which perfectly encompasses our required system voltages of 5V and 3.3V. Its maximum output is 5A. Since our maximum operating current of the system is 3.5A, the 5A max will be sufficient. This module has 6 pins in total. These pins are the ground, switch node connection, voltage input, converter feedback, enable input, and supply input pins. The list of these pins and their full descriptions are listed in Figure 13. We created our power supply designs with help from the Webench Power Designer tool provided by Texas Instruments. In Figure 14, both voltage regulators are shown in the full power regulation schematic. The values for R1 and R2 for both converters were taken from the “Recommended Component Values” table in the module’s datasheet. This schematic was created in TI-TINA, also a Texas Instruments product. TINA-TI was used to perform simulation runs on the circuit and verify its functionality.

Taking a look at the schematic, we see that there is both input and output capacitance for each power supply. The input configuration includes the two 10uF capacitors acting as decoupling capacitors and a 100nF capacitor. The 100nF capacitor is optional and its purpose is to provide additional high frequency filtering. An important thing to consider when choosing input capacitors is to make sure that they are rated higher than the operating input voltage. The capacitor connected between VBST and SW is called the bootstrap capacitor, and is required for proper operation. It’s recommended to use a ceramic capacitor for this one. On the other side of the module, we have the output capacitance. These capacitors are vital to the converter design, as they smooth out the emitting voltage by reducing ripple voltage, and can either increase or decrease the stability of the feedback loop, depending on the chosen configuration.



PIN		I/O	DESCRIPTION
NAME	NO.		
GND	1	—	Ground pin. Source terminal of low-side power NFET as well as the ground terminal for controller circuit. Connect sensitive VFB to this GND at a single point.
SW	2	O	Switch node connection between high-side NFET and low-side NFET.
VIN	3	I	Input voltage supply pin. The drain terminal of high-side power NFET.
VFB	4	I	Converter feedback input. Connect to output voltage with feedback resistor divider.
EN	5	I	Enable input control. Active high and must be pulled up to enable the device.
VBST	6	O	Supply input for the high-side NFET gate drive circuit. Connect 0.1 μF capacitor between VBST and SW pins.

Figure 13 - TPS565208 pin functions

With the Webench tool, we are able to see at exactly what efficiency the power supply will be operating at, the cost of the materials, and much more. For the 3.3V converter, its design operates at an efficiency of 93.6%. The total cost of all the parts listed in the design come out to $1.26. For the 5V design, the efficiency comes out to 94.9% and a total bill of materials cost of $1.72. The efficiencies of both the power supplies are excellent, and should allow us to maximize our system's overall efficiency.

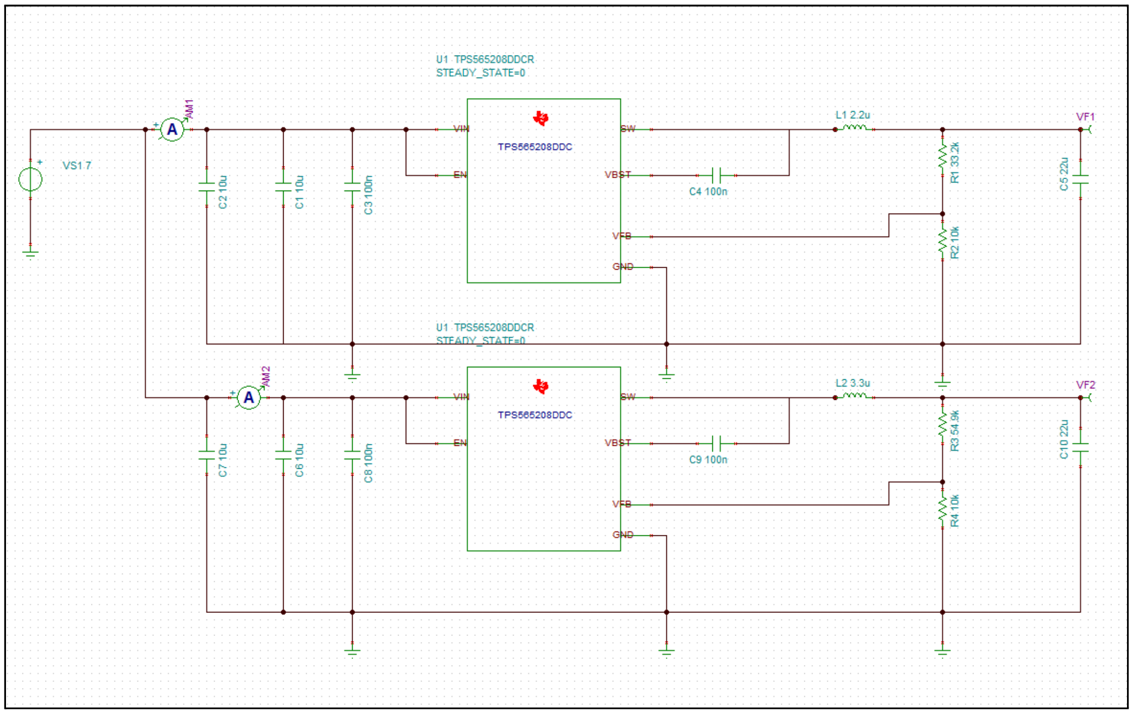


Figure 14 - 3.3V and 5V voltage regulator designs

5.1.3.3.SD2 Voltage Converter Design

In Senior Design 2, we changed our regulators from switching regulators to linear converters. The converter we decided to use was the LM1084, a linear converter from Texas Instruments. This converter has a maximum output current of 5A and an industrial temperature range of −40°C to 125°C. For the LM1084, in order to set the output voltage, 3 feedback resistors were used for each regulator. We are supplying 3.3 volts for the MCU and supplying 5 volts for the remaining components, which includes the development kit for object detection. We also have a battery that stores a higher voltage than is required (12V), and with the utilization of voltage converters, we are converting this voltage down to those required in our system. We added 10uF capacitors before each converter for input decoupling. We also added capacitance after the regulators for output decoupling. Since linear converters are less efficient than switching regulators, and thus must dissipate a lot more heat, we glued heatsinks on each of the converters with thermal glue.

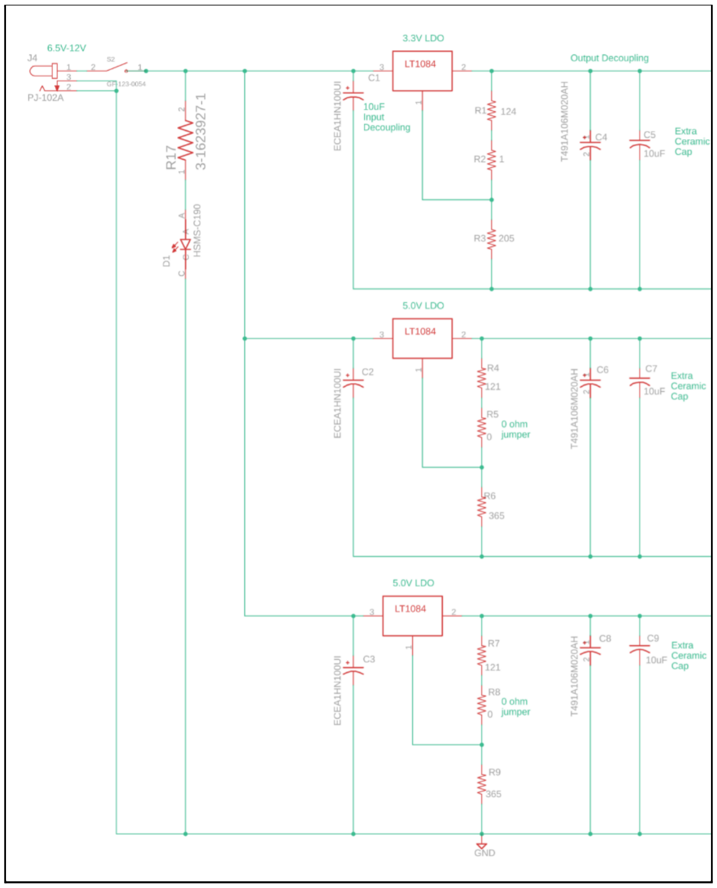


Figure 14.SD2 - Updated 3.3V and 5V voltage regulator designs

5.1.4 NVIDIA Jetson Nano Hardware Design

The NVIDIA Jetson Nano Developer Kit is designed and manufactured by the NVIDIA Corporation. Figure 15 shows a diagram of the UART connections on the Jetson Nano module, which sits atop the carrier board for the Jetson Nano. The UART connections that we will be focusing on using for our smart bird feeder device are the *UART_1_TXD* and *UART_1_RXD* connections. These connections are used for transmitting and receiving with UART serial communication, respectively, both route to the 40-pin header aboard the carrier board of the Jetson Nano Developer Kit, and the pins that they route to on the 40-pin header will be connected to our custom PCB via the pins containing the UCA0RXD and UCA0TXD pin functions.

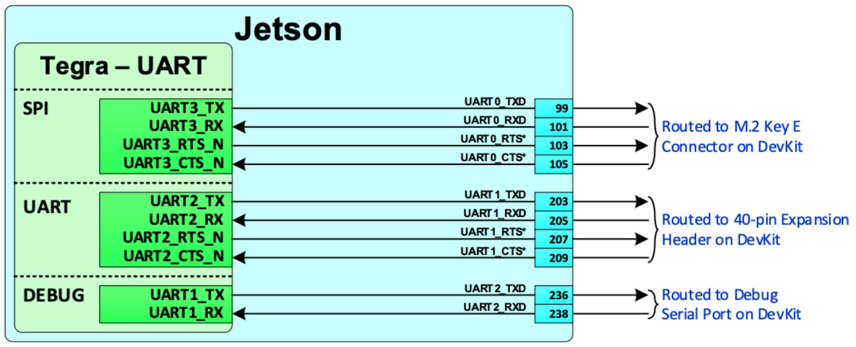


Figure 15 - Jetson Nano UART connections

The Jetson Nano Developer Kit will be powered via the 40-pin header. Pins 2 and 4 on the 40-pin header will be used to power the Jetson Nano Developer Kit, and 5VDC power will be provided to these pins from the custom PCB that our team has designed. Additional documentation related to the design guidelines for the Jetson Nano module, the Jetson Nano Developer Kit, and other Jetson products may be found at the Jetson Download Center [24].

5.2 Software

The team will use this section as a way to highlight the functionality and design of the software components related to The Smart Bird Feeder. It will be broken down by application design versus software designed for the device itself. Thus, we start this section with a sequence of subsections that dive deep into the plan for the user interface to have simplicity and functionality, then we discuss the login/signup, which is then followed up by a deep dive into the design and goals for the landing page of the mobile application. Moreover, once discussing our intentions on the most likely main page of the mobile application known as the landing page, we then continue the conversation of the application to discuss our intentions and design to the memories page that contains a virtual library of photos for all the images taken of visiting birds that were accurately identified by the object detection.

5.2.SD2 Software

The mentality towards app design stayed consistent in SD2. However throughout this section there are some small features mentioned that were not prioritized throughout development. The largest change was associated with the live streaming capabilities which could not be implemented in time, due in part to the Jetson Nano using RTP streaming, which our front end is not compatible with. Therefore, Live streaming and a few other additional design components were moved to stretch goals, but unfortunately left incomplete at the end of Senior Design II In order to prioritize other elements of the project that were deemed of higher priority.

5.2.1 Smart Bird Feeder Application Design

This section will be used to describe the functionality of the application's design. In order to cover the breadth of capabilities, it will be split into subsections based on key functions and components of the app.

5.2.1.1 User Interface

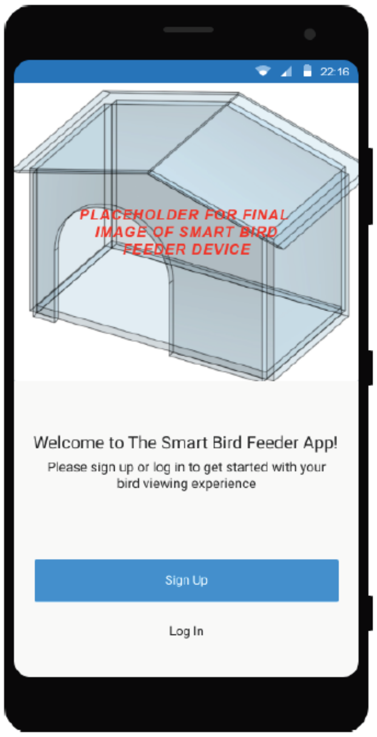
The application must be designed with simplicity and functionality in mind. The user interface needs to be approachable such that the users show no frustration towards the different mechanisms the application has to offer. In order to accomplish this the UI elements must remain consistent and common. For example, a bell symbol is synonymous with notifications, and keeping that symbol consistent in the Smart Bird Feeder Application will make our notifications function consistent with how users understand mobile applications. This design philosophy will extend to all UI elements.

The page layout must be purposeful, with a clear goal associated with each section of the UI. The login page, home page, memories section, and live streaming page all fill a purposeful purpose in the application, and the transition to each of the individual pages must be fluid. For instance, the streaming page should only be made clearly accessible via a feeding or deterring event, so as to not confuse the user when there is nothing occurring on the camera system.

The color scheme should be strategically used to convey levels of urgency towards different events, but also provide a comfort to the user's needs. A consistent set of colors is important to the overall feel of the application, with colors such as red being used for important things like notifications and live streaming.

5.2.1.2 Login/Signup

The application will start with a login/sign up page. Figure 16 shows a prototype of the welcome page that a given user will be met with upon opening the application. The user may either sign up for the app by creating a new account, or log in with an existing account if they have used the app before.



Welcome to The Smart Bird Feeder App!

Please sign up or log in to get started with your bird viewing experience

Figure 16 - Welcome: sign up/log in page (prototype)

If the user chooses to create a new account, they will select the button labeled "Sign Up". Figure 17 shows the sign up page of the application, which the user is redirected to by selecting the "Sign Up" button. Here, the user is first given the option of choosing a profile picture for their account. After making this optional choice, the user must enter their name, email address, and a username, then choose the "Sign Up" button on the sign up page. Any user's information will then be held in a temporary table within a database, waiting for confirmation from an email verification system.

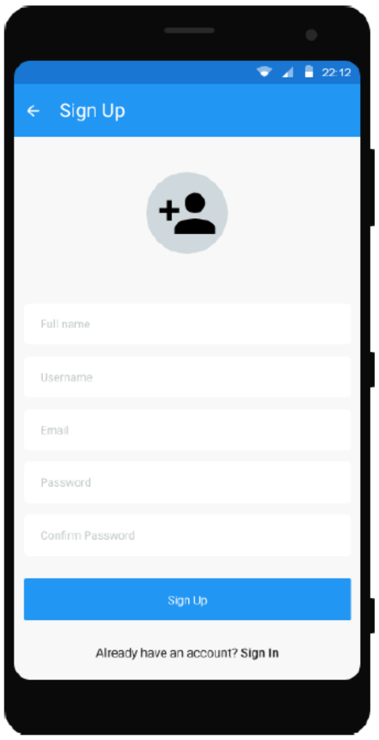


Figure 17 - Sign up screen (prototype)

The user will then be prompted with a message from the application requesting the user to confirm their email address in order to sign into their new account. After acknowledging this request, the user will be redirected back to the welcome screen of the application, and may choose to sign in to their new account from there (after confirming their own email address). When an existing user signs in, they may choose to have the application remember their login credentials on that device for the future use of the application.

Next, the user should be able to find their smart bird feeder device and connect to it as long as both their smartphone and their smart bird feeder are within range of the same WiFi network signal.

In order to increase security, it’s important to ensure the user completes a verification process. This process also saves space by avoiding fake accounts. A user will not be able to input a password to sign in to the application until after they have confirmed their email.

In order to store user information, we will be using a MongoDB database. The program MongoDB Compass allows us to visualize and change the database structure. Using MongoDB compass, a temporary database will store the user's email and username information for a set amount of time, unless the user confirms their account. When the confirmation process is complete, the user's information is shifted from the temporary table to a permanent data storage table.

Node.js and React make up the MERN stack that will house the applications functionality. The API being written in Node.js will collect a user's information and store it in the appropriate database. Node.js API will also send the confirmation email to the users email address. React will be used to create the login UI, and connect it to the API code.

5.2.1.2.SD2 Login/Signup

Verifying a user's email address was a feature that the team decided not to prioritize over completing application core functionality, so this feature has been moved to stretch goals. The updated login and signup screens are shown below in Figure 16.SD2 and Figure 17.SD2.

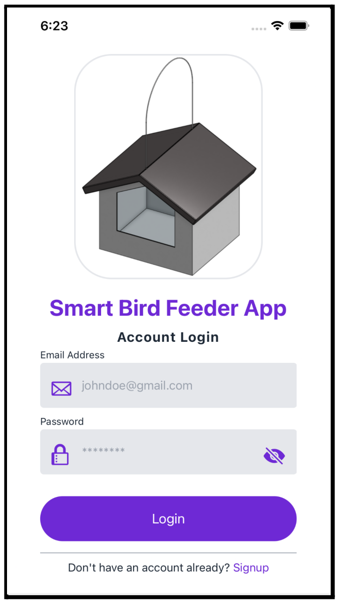


Figure 16.SD2 - Updated login screen

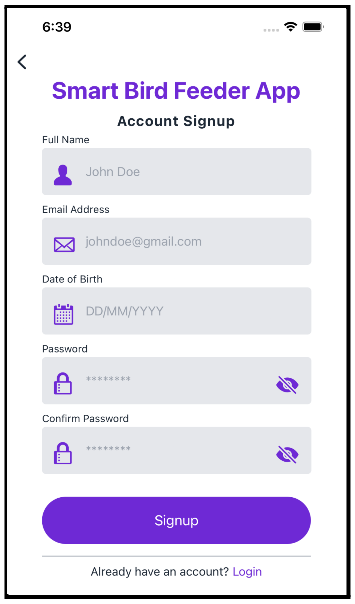


Figure 17.SD2 - Updated signup screen

5.2.1.3 Landing Page

Figure 18 shows a prototype of the application landing page, which will be home to access most of the functionality of the app. The settings button will lead you to the settings page, which will contain choices regarding notifications. The notification button will open up a popup menu where the user will have the option to view or clear recent notifications. During the event of a notification, a bar will appear across the top of the application for a limited amount of time, showing off the most recent notification. For example, if a feeding event occurs, the user will see a box appear on the top detailing what species of bird is feeding; clicking this box will transfer the user to the livestream page to enjoy the feeding taking place.

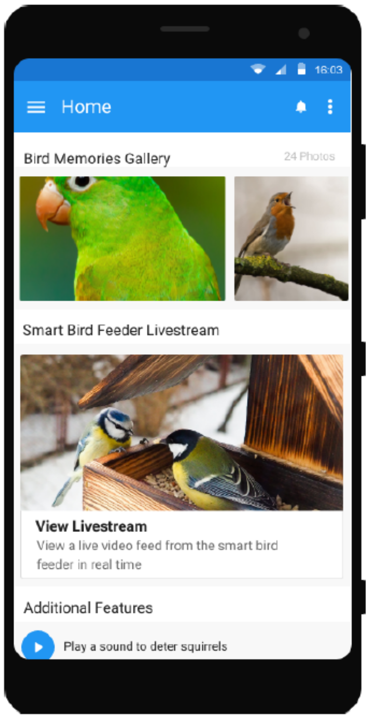

16:03
Home
Bird Memories Gallery
24 Photos
Smart Bird Feeder Livestream
View Livestream
View a live video feed from the smart bird feeder in real time
Additional Features
Play a sound to deter squirrels

Figure 18 - Landing page (prototype)

In the center of the landing page will be two options which will access the primary purposes of the application. The most important being the “Bird Memories Gallery” card area, which will preview saved images of the birds taken by the smart bird feeder. Upon selecting the title of this bird memories card, the user will be directed to a dedicated bird memories page containing the full gallery of saved bird images. The second card on the landing page is a card containing a preview and description of the live video feed from the smart bird feeder device. If a user selects this card, they will be taken to a page showing a live video feed from the camera built into the smart bird feeder device that they are connected to. From this page, the user may also manually capture still images from the feeder's camera remotely through the app. Of course, the smart bird feeder device will be programmed to automatically capture still images of the birds that it detects on its own, and predict the species of the bird as well. Finally, the "Additional Features" menu within the landing page contains a couple extra 'quick' features which may be selected by the user at any time. The first button with the "Additional Features" menu gives the user the option of manually playing a deterrent sound from the smart bird feeder device, which would essentially temporarily override the programming that the bird feeder has to automatically trigger the sound when a squirrel is detected. The second button offers the user the option to manually quickly capture a still image from the camera built into the smart bird feeder without the need to have the live bird viewer video feed pulled up on the user's app. Of course, there is no guarantee that a bird will be in sight of the feeder's camera, or properly detected by the smart bird feeder, when an image is manually captured, so results may vary when using this option for capturing images of birds.

The UI frontend design of this page will be done using React Native, with sub categories such as notifications being coded using Node.js. The species detection software will interact with the Node.js code for notifications, in order to detail that a feeding event is taking place and what species is currently feeding.

5.2.1.3.SD2 Landing Page

The ability to manually take pictures or activate the alarm had been pushed to stretch goals in order to complete the core functionalities of the application. Therefore, these functionalities were not implemented on the landing page of our application in Senior Design II. For an updated look at the landing page, see Figure 18.SD2 below.

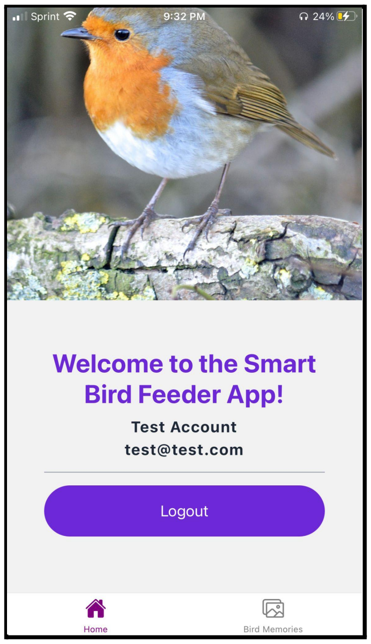


Figure 18.SD2 - Updated landing page

5.2.1.4 Memories Page

A prototype of the bird memories gallery page ("memories page" for short) is shown in Figure 19. The memories page will allow the user to access or delete any of the previously saved feeding memories. The memories will be listed in an endless scroll, with the picture and the name of the species indicating which memory is being viewed. The memories will be able to be sorted by species, or date acquired. There will also be a search bar on the top, allowing users to search for a specific bird species.

Upon selecting a memory, the image will expand to a full screen view, and the user will be able to easily view the name(s) of the bird species detected in the image, and given the option to save the photo, delete, or set the memory as a favorite. Within the bird memories page, the user may also filter all images by favorites or date added.

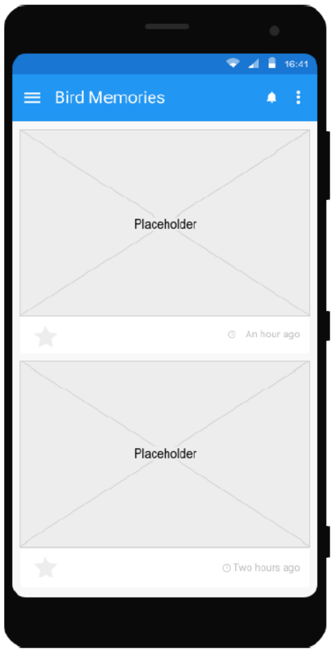


Figure 19 - Bird memories page (prototype)

Allowing the user to categorize the memories how they see fit, and the option to search for a specific memory are key functions that help to highlight the moments that the user finds most valuable for the device, heightening the connection between them and the smart bird feeder.

The images will be saved on the server application, Heroku. The server will link images to their respective dates and species names using the mongoDB database.

5.2.1.4.SD2 Memories Page

For an updated birds memories page, see Figure 19.SD2 below. The final design for the bird memories gallery included a carousel card view of all the saved bird memories from The Smart Bird Feeder showing the bird image, species name and time of capture, with a scrollable FlatList thumbnail list to select the memories from as well. Functionality for sorting bird memories by date/time in ascending/descending order, or by species name, were also implemented in this final design.

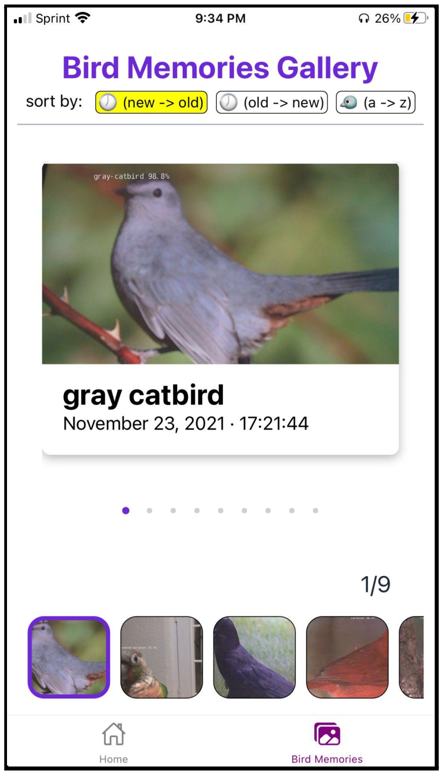


Figure 19.SD2 - Updated bird memories page

5.2.2 Computer Vision Annotation Tool Software (CVAT)

In order to train the ML model run on the Jetson Nano Developer Kit to detect various bird species and squirrels, we first need to annotate some images to train on. While speaking to a mentor of ours on object detection, CVAT was introduced to us as an easy to use free, online, interactive image annotation tool for computer vision. This tool which was open sourced by Intel helps in labeling objects in images with bounding boxes, as we intend to detect and label bird species and squirrels at the Smart Bird Feeder. Therefore, we will have objects like blue-jay, cardinals and squirrels that need to be labeled in the images so that we can train our model to identify them as well.

After creating an account on CVAT, we are then prompted to the tasks page where we can create and share labeling tasks amongst the group. Our group was able to make new tasks as we found new images and expanded on the bird species. However, one of the more stressful parts of this, came after naming the task and inputting the classes (labels such as blue-jay or cardinal) that will be used in the images, is the uploading part, as this means that we need to find a database or library of good quality images of the particular bird species. But once we found and uploaded a good amount of images to CVAT, we are on to the step where we can open up the task and start labeling each object in the image.

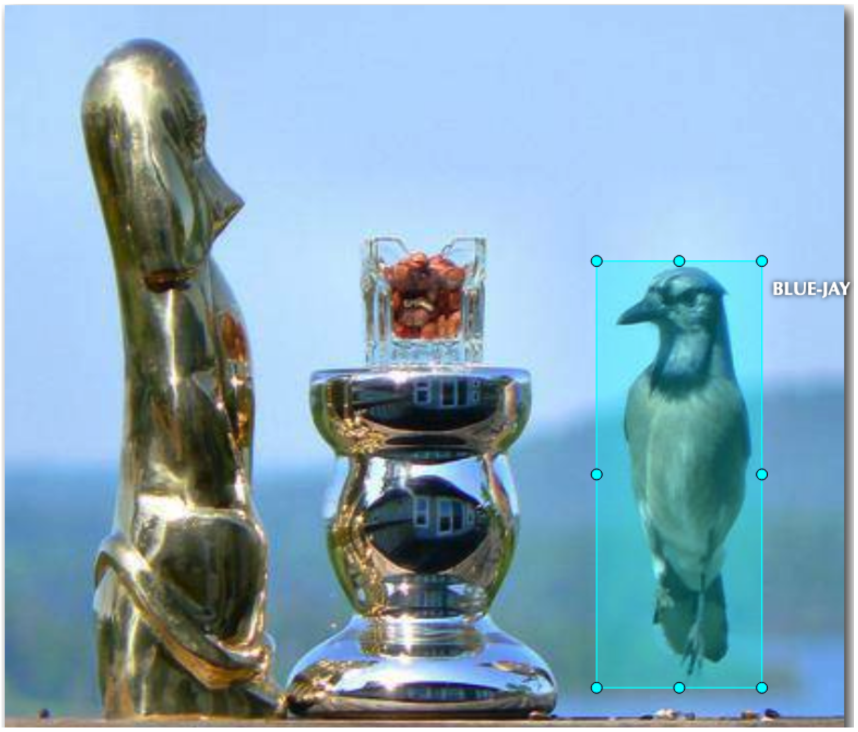


Figure 20 - CVAT labeling of blue-jay

Figure 20 (seen above) is one of many images of Blue-Jay's which needed to be labeled before training our object detection model on detecting when blue-jays visit the Smart Bird Feeder. Figure 20 gives a visual representation of what needs to methodically be done to hundreds of images of the various bird species and of squirrels. This step of labeling each of the images is easily one of the most if not the most time consuming parts, even when the jobs are shared where each participating member labeling can get around a couple hundred images to label. Moreover, the processes of labeling might be a little longer than some may expect, as it is important that when we label, we need to keep in mind that we should label the entirety of the object and that the box around the object does not have a lot of extra room. The idea is that we aim to box the object with a little bit of room around it because the edges are important for the object detection model, but we do not want too much unnecessary room around the labeled object.

Once all the labeling is done, we need to get it ready to upload to our development kit where the machine learning is done. Therefore, we go back to the tab of the tool where the task is, and click on the dump annotations. Under the dump annotations tab, we are given a list of various ways of downloading the collection of datasets for object detection. In our particular situation, we will click on the option of PASCAL VOC, which provides multiple folders with an annotations folder of all the xml file annotations corresponding to the images which are labeled. With these xml files help provide the machine learning code to know where in the corresponding image the label belongs with the corresponding label.

5.2.3 MSP430G2553 Software and Components

For programming our MCU, the MSP430G2553, we used Code Composer Studio (CCS). This application is an integrated development environment created by Texas Instruments. This application is specifically designed for Texas Instruments processors, such as our MSP430 module. Our team developed a program using the C programming language for the microprocessor which is able to perform the following tasks: send and receive data to/from the ML developer kit, control the movement of the motor, control the output of the speaker, and send and receive signals to/from the ultrasonic sensor. The following sections discuss the development process of the processor's software related to each of the components that the microprocessor interacts with.

5.2.3.1 MSP430G2553 Development

In order to develop the programming of the MSP430G2553, we used its corresponding development kit. The name of this launchpad is MSP-430G2ET. This kit provides an easy way to develop, troubleshoot, and flash the MSP microcontroller. It also provides onboard emulation and multiple LEDS for use. Figure 21 shows the Launchpad board and captures the 20 I/O pins to the microcontroller.

Pins P1.1 and P1.2 will be used for transmitting and receiving data to and from the development kit. These pins can make use of the USCI_A0 UART mode in order to send and receive data over UART. To interface the microcontroller with the ultrasonic sensor, pin P2.0 will be used for receiving the echo signal while pin P2.1 will be used to connect to the sensor's TRIGGER pin. Next we discuss the pins connected to the motor. Timer0_A3 and Timer1_A3 onboard the microcontroller can support PWM outputs, so we want to choose a pin that makes use of one of these timers as our PWM output pin to the servo motor. The pin P1.3 will work for this purpose. To apply voltage to the gate of the transistor in the motor configuration, pin P1.5 will be used. Finally, for the piezo speaker PWM, pin P1.4 will be utilized.

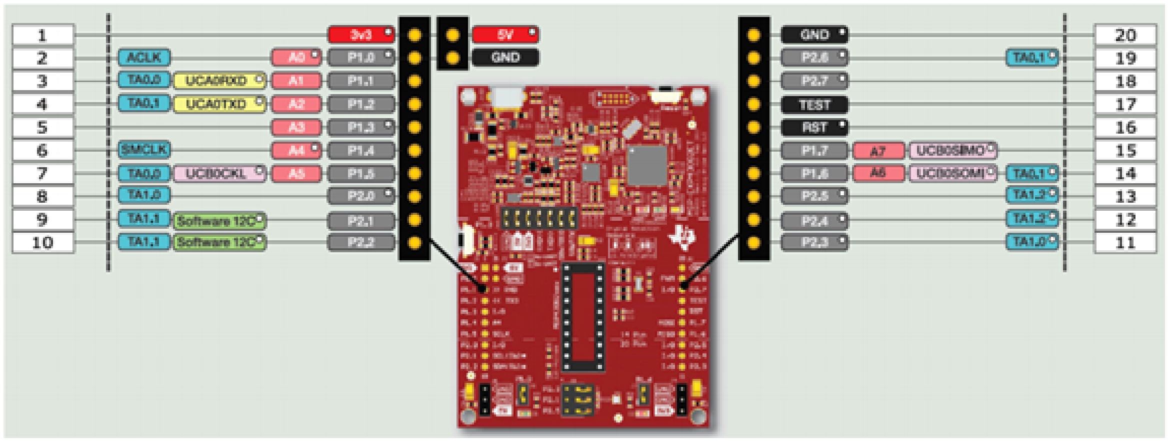


Figure 21 - MSP430G2ET [28]

5.2.3.1.SD2 MSP430G2553 Development

In Senior Design 2, we used different pins than expected for some of the outputs from the MSP430G2553. This came about because of more research that was conducted in Senior Design 2. Pins 1.1 and 1.2 are still used for UART reception and transmission. Pin 1.3 is used for the output of the PIR sensor. This pin is connected to the ADC module on board the microcontroller. Pin 2.0 is still used for receiving the echo signal from the ultrasonic sensor. Pin 2.1 is also still used for connection to the TRIGGER pin on the ultrasonic sensor. Pin 2.2 is used for sending a PWM signal to the transistor used for the speaker audio. Finally, pin 1.6 is used for connecting the MSP430 to the servo motor, and sending a PWM signal.

5.2.3.2 Motor

The devices connected to achieve motor functionality are the MSP430G2553 board, the Jetson Nano Developer Kit, and the motor. Upon receiving the signal to close the hatch due to a predator, the MSP430 provides power to the motor, which spins the hatch mechanism, thus closing the access to the feeding chamber. The command was programmed to power the motor with a specified PWM signal which establishes which position the motor should rotate until. Also, the Jetson Nano sends another signal to the MCU once it no longer detects squirrels. This signal tells the MCU to move the motor to the open position once again.

The signal starts on the Jetson Nano Developer Kit. This is an enable message telling the microprocessor to move the motor. The Jetson Nano Developer Kit is connected to the MSP430 using UART connection. It is programmed to send a signal through UART to the board if it senses a feeding predator while using its machine learning technology. Once the MSP430 receives the signal, it sends a specific PWM signal to the servo motor. This PWM signal enables the motor to move to the new position, either opened or closed. How the PWM signal works in relation to the servo motor is discussed in section 3.5.1.3.

There are several options for the configuration of the motor and the microcontroller. The simplest configuration would have the motor directly connected to the pins of the microcontroller. This means that the signals for the PWM and the positive voltage would be coming directly from the board. However, since our servo motor will require a voltage of ~5V, this option is not viable. The configuration we will be using involves connecting the PWM signal to the source of a transistor. The gate of this transistor will be connected to a different pin on the MCU. When the program function is called inside of the MCU, this ‘enable’ pin will drive the gate of the transistor, and allow the current to flow from source to drain, providing power to the servo motor from the output of the 5V voltage regulator. At the same time, the servo motor will send a PWM signal to the signal pin of the servo motor, and thus telling the servo motor which position to move to.

This event will also alert the user that a predator is attempting to access the feeding chamber. The Jetson Nano Developer Kit will send a signal over the WiFi network to the application server. It will also snap a picture of the intruder and send that to the application as well. The application server will have an API that activates, and uses its notification process to alert the user that a predator has attempted to feed, along with displaying the picture of the predator. React Native will be used to code the display for the notification system, and display the picture to the user.

After the ML program running on the Jetson Nano Developer Kit determines that the squirrel has left the vicinity, with help from the alarm system as a deterring mechanism, the Nano will send another signal through UART connection to the motor, which will then be powered again to spin the hatch back to the open position. It is not necessary to the project to send a notification or picture for this process.

The opening mechanism will have logic built in to avoid constant opening and closing in the case that the squirrel disappears from detection just for a moment. There will be a time frame of 15 seconds where the squirrel must be absent from view before the Machine Learning processor will open the hatch, and allow for the possibility of the defense mechanism activating again. There will be a longer delay regarding the notification and photo system. We do not want to frustrate the user in a situation where a squirrel is constantly reappearing, and they are thus receiving numerous notifications. The delay for the notification system will be 60 min, as it is unlikely the same squirrel will be returning to the feeding system after constant failure.

5.2.3.2.SD2 Motor

Unlike the design in Senior Design I, we were able to connect the motor directly to the MSP430G2553. In Senior Design 2, we did not use a transistor in the motor circuit, and instead directly connected the motor's PWM signal line to pin 1.6 on the MSP430. We also directly connected the motor's positive power line to a pin on the 5V header on the PCB. The ground of the motor is connected to a pin on the ground header on the PCB.

5.2.3.3 Alarm

The devices that will be connected to achieve alarm functionality will be the Jetson Nano Developer Kit, the MSP430G2553 board, and a small speaker.

The signal will start on the Jetson Nano development kit which will be connected to the MSP430G2553 board using UART connection. Once the machine learning processor senses that there is a predator, the signal will be sent to the MSP430 board to begin the alarm process.

The alarm will be connected to the MSP430 board output pin, which will be acting as a power source. When the board receives the command, the alarm will sound by providing the alarm with a power source. This state will be relayed to the user through the Jetson Nano kit, through a WiFi signal to the user application. This process will also snap a picture of the intruder, and provide the user with a notification to watch the deterrent in action.

The alarm will sound until the processor no longer detects the squirrel for a period of time, around 15 seconds. Once the squirrel has vacated the detection vicinity for this time frame, another signal will be relayed from the processor to the MSP430 board to shut off power to the alarm.

The alarm software will ensure that continuous notifications are not sent to the user in the instance the squirrel consistently harrasses the device in a small timeframe. The alarm system will activate immediately after it detects a squirrel, regardless of the amount of time the squirrel is not present. This system should ensure the squirrel does not linger due to the annoyance of the alarm and inability to access the food storage.

5.2.3.3.SD2 Alarm

In Senior Design 2, we made a slight modification in how the piezospeaker was connected to the MSP430. In order to utilize the 5V on board the PCB for maximum volume output, we had the 5V connected to the positive connection of the speaker through a PN junction diode and 1k resistor in parallel. The direction of the diode faces the 5V in order to limit current. The positive connection of the piezospeaker connects to the 5V after the diode and resistor, and the ground of the piezospeaker connects to the collector of the transistor. The PWM signal coming from pin 2.2 connects to the base of the transistor. Finally, the emitter of the transistor connects to the ground of the PCB.

5.2.3.4 Ultrasonic Sensor

The devices connected to achieve functionality for low bird feed alerts are the Jetson Nano Developer Kit, the MSP430G2553 board, and the ultrasonic sensor.

The ultrasonic sensor is connected to the MSP430 board pin to provide a connection. It is programmed to check for the depth of the food storage container at intervals of 30 minutes. The MSP430 board is programmed to check for a certain depth range to indicate a lack of food. As the amount of food volume decreases, the depth of the container increases. When the sensor reads a volume that is low enough to trigger the food alert, the MSP430 board sends a signal to the Jetson Nano through UART connection.

This signal is sent over the WiFi network to the user application. The application API activates and starts a notification process to alert the user that the bird feed level is low and needs to be replaced. The user then has the opportunity to replace the bird feed.

5.2.4 Machine Learning Object Detection Model

To detect bird species and squirrels, we will be utilizing the *DetectNet_v2* model as a base for our machine learning model algorithm. This model is provided by NVIDIA in the Transfer Learning Toolkit (TLT).

Thanks to helpful tutorials and examples provided by NVIDIA developers, we are able to use DetectNet to retrain this model with the SSD MobilNet V2 model working as a backbone feature extractor for the training of the model to perform object detection on any object classes of our choosing. In using this DetectNet_v2 model in combination with the tutorials provided by NVIDIA developers, we are fortunate enough to be able to keep our code simple and concise, and by writing only a handful of lines of python code, we are able to have a working ML model trained and running to detect bird species and squirrels. To do this, we utilize the *detectNet* function found within the *jetson.inference* library provided by NVIDIA, and we need only pass in a few select parameters to this function. These parameters include the selection for which architecture to use for the model, which in our case would be "ssd-mobilenet-v2", and a threshold value, which is problem-dependent and requires fine-tuning to get completely correct. Thus, we will only know the perfect threshold value to use for our ML model at the end of its development cycle, once we have tried enough threshold values to yield the results that we seek.

6. Overall Integration, PCB Design, and System Testing

The integration process will involve uniting the multiple blocks of our system, and creating a complete, whole device. Once we reach the overall integration stage of our project, we will have successfully programmed the development kit and MCU. At this point, we should have a fully fleshed-out program on the MCU that is able to control the motor, ultrasonic sensor, and speaker. We should also have a development kit that can accurately detect the difference between birds and squirrels, identify and classify different bird species, and upload images to our mobile application cloud. Finally, we should have a working mobile application. In these final stages of our project, we will be working on integrating the three main blocks of our project with one another. These blocks are the mobile application, the developer kit, and the MCU and its peripherals. Extensive troubleshooting and testing will be required during these last stages, especially when we are integrating the development kit and MCU with one another. We must make sure that communication between the two modules is accurate and efficient. Of course, before we are able to integrate the blocks with one another, we must design and order a custom printed circuit board. During overall integration, it may be required to re-order PCBs with design changes, as we will most likely come across problems that need to be solved with re-design. With overall integration will come system testing. The purpose of system testing is to verify that our system works perfectly. Through system testing, we will be able to test different cases to ensure that our device does not have any bugs or issues. In the next sections, we provide an overall schematic of the system, outline the PCB design process, overall integration, and system testing of our smart bird feeder.

6.1 Overall Schematic

In Figure 22, we provide the overall schematic of the entire system. This schematic was created using EAGLE, and will be used to create the board layout and gerber file for the PCB manufacturing. For this schematic design, we tried to achieve an organized and compact representation of the system. Since there are no symbols provided for the Jetson Nano and servo motor, we have labeled the pins on the pin headers that will be connected to the multiple connections on the Nano and servo motor. Since there were symbols for a piezo speaker and ultrasonic sensor, we added this to the schematic for better visual understanding. Looking at the power regulator sub-system, we see that instead of having the six capacitors in the previous mockup design provided by WEBENCH, we will be using only three capacitors, since the input voltage to the system will be directed to both regulator modules. In order to convert this schematic to a board layout, and later a gerber file, we will remove the labels and also the ultrasonic sensor and speaker symbols from the pin headers, since these components are not going to be a part of the PCB. The top left symbol in the schematic shows the DC female barrel port, which will allow connection between the battery and the PCB. It is then followed by the power regulators, and microcontroller.

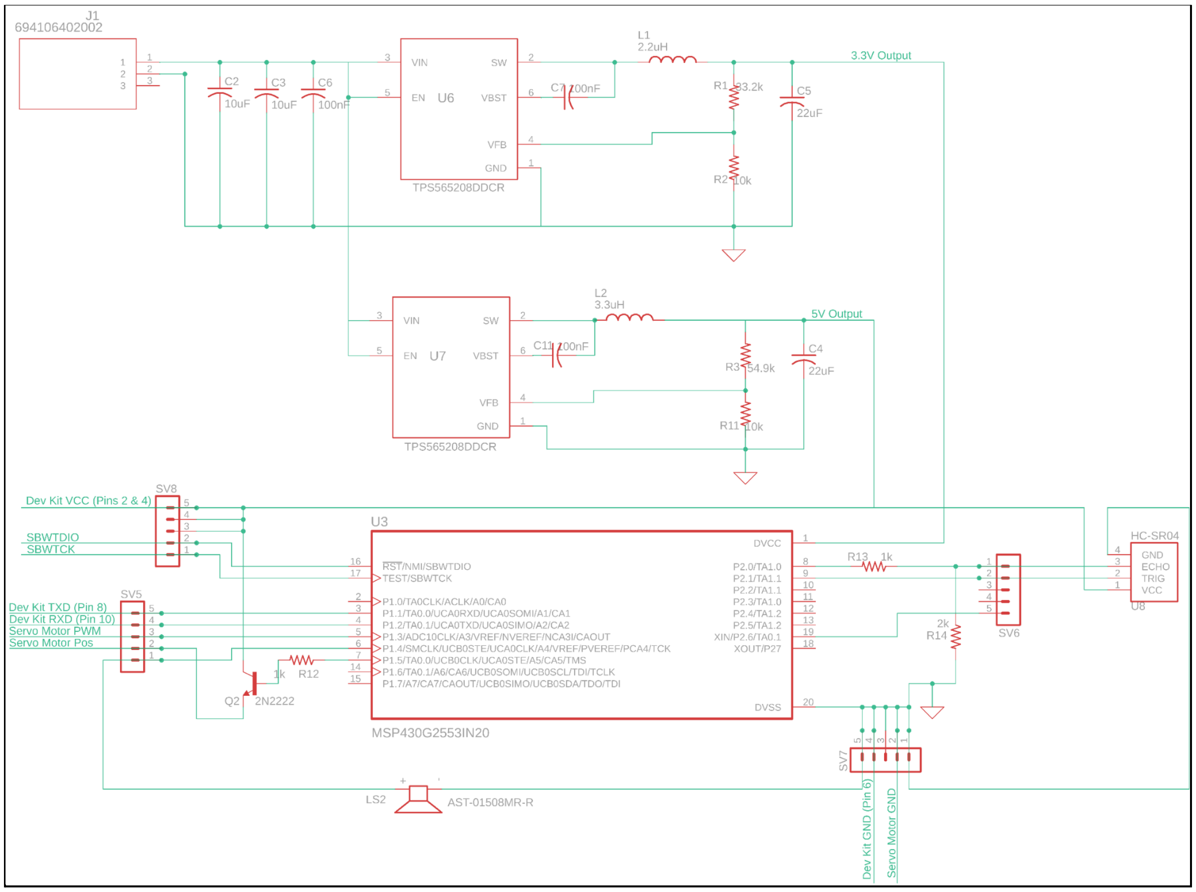


Figure 22 - Overall schematic

6.1.SD2 Overall Schematic

Here is the finalized overall schematic. We added more components to better outline the design, and highlighted and named each subsection of the design for clarity.

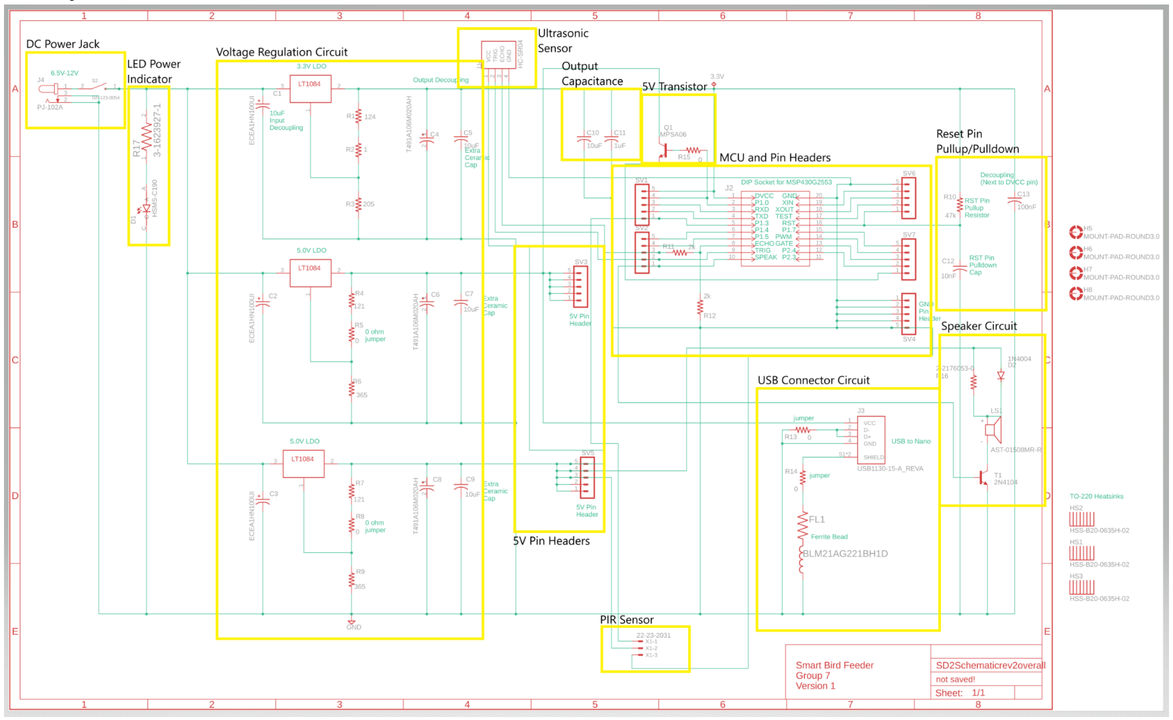


Figure 22.SD2 - Updated overall schematic

6.2 PCB Design

The PCB, or printed circuit board, is an important component for integrating our many device components into one system. PCBs are typically composed of flat laminated composite made from non-conductive substrate materials. The PCB will have layers of copper whose purpose is to connect the many components on the PCB. The PCB includes many parts, including the microcontroller and voltage converters. We needed to have circuits on our board which were able to convert the voltage input from the rechargeable battery to the required +3.3VDC for the operation of the MSP430G2553, and to +5VDC for the VCC connection on the ultrasonic sensor. We also needed to include an input port on the circuit board that allows us to connect the rechargeable battery to the circuit board. In addition, we needed to incorporate a way to connect the development kit to the circuit board.

Many steps had to be taken before we had a working PCB. The first step was determining the type of software that we needed to use in order to create our PCB design. The software that we used for our project is EAGLE. This is a software developed by Autodesk that provides schematic capture, PCB layout, auto-router and computer-aided manufacturing features. The reason that this software was chosen as opposed to other available softwares is due to the group members' experience using EAGLE in past UCF curriculum. Since we have already utilized EAGLE, we are most familiar with this software. This allowed us more time to work on other aspects of the project, instead of needing more time to learn how to use a software we are unfamiliar with.

In order to create a PCB design file with EAGLE, a process was followed. The steps in this process are shown below.

1. Create electrical schematic
    a. Retrieve footprints for necessary components
    b. Add components to schematic
    c. Connect components together
    d. Label and name all nets
    e. Give some parts specific values
    f. Execute Electrical Rule Check
2. Create the board layout
    a. Arrange components
    b. Run ratsnest command
    c. Create ground plane
3. Route the parts
4. Design Rule Check

After the PCB was designed in Eagle, we needed to export this file as a Gerber file. This is the most widely used file format for PCB manufacturing, and is what our chosen manufacturer, JLCPCB, required for production.

After we received our PCB from the manufacturer, testing was conducted before we were able to integrate it with the rest of the system.

6.2.SD2 PCB Design

Below is an example of our final PCB design.

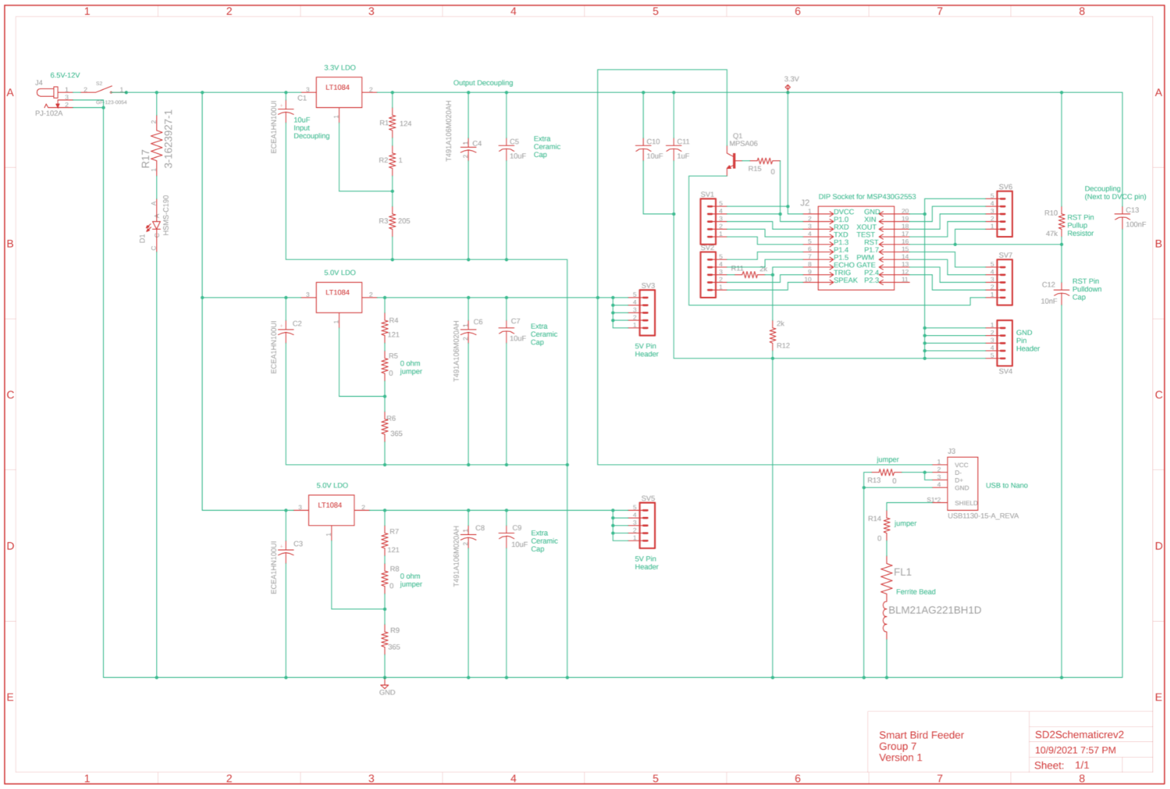


Figure 23 - PCB schematic

6.2.1 PCB Power Interfacing

In order to interface the battery with the PCB, we needed to decide on a charging interface between the two. The main way in which we connect the battery to the PCB is through use of a female and male barrel jack connector. In Figure 24, we show an image of a barrel jack female port and its labeled pins. In addition, we show a picture of the male barrel jack cable that was used for our power interfacing. The female port has three pins: the insertion detection (or shunt), the tip, and the sleeve. Typically, the Sleeve acts as the ground while the Tip is the positive voltage. The purpose of the shunt pin is to detect if a power supply has been plugged into the barrel jack. When nothing is plugged in, the shunt pin is connected to the sleeve, or ground. Once something is plugged into the port, the shunt pin disconnects, and acts as an open. This comes in handy when working with battery-powered devices, as this avoids overheating the battery when an external power source is plugged in. This feature was not beneficial to our application, however. On our PCB, the tip on the barrel port connects to the input to our voltage regulators. The positive and negative ends of the male plug connect to our rechargeable battery array.

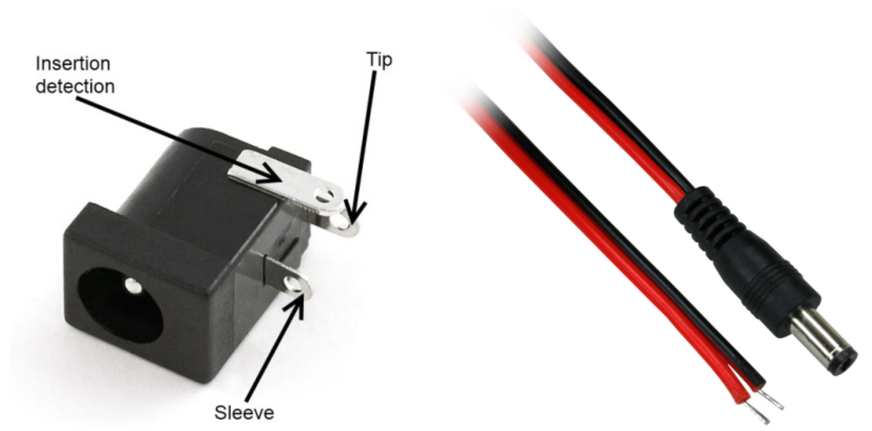


Figure 24 - DC power jack female (left) and male (right) connectors [29, 30]

6.3 System Testing and Prototyping

This section will outline how we tested the hardware and software components of our smart bird feeder device. It is important to ensure that each component in the system operates properly before deploying the final system in order to limit the potential for failure upon system deployment. In other words, since the overall Smart Bird Feeder's functionality and successfulness to perform all the necessary tasks that make it unique, we need to ensure that each aspect of it works as expected from it. This is why, one of the most crucial parts in building the foundation for this product, is the testing phase. Thus, in doing the testing we are able to test out the functionality of the prototype with various tests on the hardware and software. For instance, as a group we started off with testing the functionality of the microcontroller and machine learning acceleration hardware, prior to looking into such aspects similar to the application. In this section of system testing and prototyping, we have provided subsections that dive deep into the various components of the bird feeder, such as the hardware testing and prototyping, and the software testing. Within the hardware testing and prototyping, there will be a detailed evaluation on such things as the microcontroller testing, ultrasonic sensor testing, and machine learning acceleration testing and prototyping to name a few. Then we go on to provide a detailed description of the software testing and prototyping. In regard to the software side of things, one of the main focuses other than the software that goes into object detection, will be on the development and prototyping of the mobile application, as it will need to be highly functional as it will be the main means of connecting users to the bird feeder.

6.3.1 Hardware Testing and Prototyping

When looking at the various aspects we are testing and prototyping that make up the Smart Bird Feeder, we have decided to label them under one of the two groups, which are hardware and software. And while it may seem as if we have the two subsections separated, it should be noted that they are very much intertwined with one another in the overarching section. Before going into detail with the implementation and results of testing and prototyping of the various hardware parts, we start off with detailing the statuses of each in giving an informative introduction. We started our hardware testing, with the microcontroller and machine learning acceleration hardware as they were the first thing we had on hand, and partly due to the fact that they seemed to be the most difficult to test compared to some of the others. For instance, when it comes to the machine learning hardware, we took a look at the NVIDIA Jetson Nano 2GB Developer Kit. Some of the next components we moved on to after to test, were the ultrasonic sensor, motor, and ultrasonic sensor.

The first aspect of our smart bird feeder device system that was tested is the system hardware. A list detailing the system hardware that was tested or that we plan to test and prototype in this procedure, along with how each system hardware component was tested, is as follows:

- **Microcontroller**
    - Tested for proper UART functionality.
- **Machine learning acceleration hardware**
    - Tested functionality of Raspberry Pi Camera Module V2 when connected to the Jetson Nano Developer Kit.
    - Prototyped with object detection of multiple classes of objects (person, bird, laptop, etc.) using pre-trained ML model(s).
    - Prototyped object detection with recognition of bird species, recording accuracies observed from testing.
    - Prototyped sending information over WiFi to another application/device.
- **Ultrasonic sensor**
    - Tested by measuring the accuracy and uncertainty of depth readings.
    - Tested by placing an object at varying pre-measured distances from the ultrasonic sensor and comparing range measured by the ultrasonic sensor to the correct distance.
- **Motor**
    - Tested by measuring acceleration and velocity when in motion.
- **Speaker/alarm system**
    - Tested by measuring decibels and alarm consistency.
- **PIR Motion Sensor**
    - Tested by printing statements when motion is detected through UART, and verifying that print statements occurred only when there was motion

6.3.1.1 Microcontroller Testing

In one of the many group meetings our group had, an overwhelming concern came from the time constraints when it came to the time it takes to test and receive parts from the manufacturer. Therefore, this concern of verifying that parts such as the microcontroller work properly was a major priority to us, since if there was any issue, we would have to wait for another one to ship out in time for us to test again before adding it to the overall Smart Bird Feeder. With this being said, after we unanimously decided on the microcontroller that we believed would adequately suit the needs of our project, thus, we decided to waste as little time as possible before going into ordering the microcontroller. Once we acquired the microcontroller, we went straight into testing mode, led by Nikki Marrow, to not only verify that it was the right purchase, but also to determine that it worked correctly. Therefore, the remainder of this section will dive deep into the testing that was done on the microcontroller, before incorporating it into Smart Bird Feeder.

The microcontroller chosen for use in the smart bird feeder device is the MSP430G2553, an ultra-low-power microcontroller from Texas Instruments (TI), which our team members have prior experience using. Before integration into a printed circuit board, all initial testing of the microcontroller will be done using the corresponding launchpad created by TI, MSP430G2ET. System testing for this development board will be carried out using a multimeter and Digilent Discovery kit, provided by UCF. This kit includes many tools which will allow us to test the full functionality of the microcontroller, such as a USB oscilloscope and instrumentation system, oscilloscope probes, and breadboard breakout cables. With this kit, along with the multimeter that a team member currently owns, we will be able measure and analyze the multiple signals coming from the microcontroller. The oscilloscope will be particularly important when measuring the PWM signals on the microcontroller’s output pins. We are not able to measure these signals with a multimeter, since they will be varying with time.

In order to power the board, we used a USB to USB micro B converter cable. The USB was plugged into a PC, and the USB micro B end was plugged into the launchpad. The PC was loaded with Code Composer Studio, which our team used to program, debug, and flash our microcontroller. Onboard the MSP430G2ET launchpad is the eZ-FET Onboard Debug Probe. This module enables debugging and programming, as well as communication between the PC and launchpad. This gave us great debugging capabilities in the development process of our microcontroller. Housed in this module are power converters which convert the 5V from the PC down to 3.3V in order to provide input power to the microcontroller. In addition to the onboard debug probe, the launchpad provided a module which can provide real-time power consumption readings and state updates from the MCU. This was incredibly helpful to us when trying to achieve our low-power goals. We were able to observe exactly how much power our MCU is consuming under various tests.

Once we fully tested the microcontroller's outputs using the launchpad, we connected the multiple peripherals to the microcontroller's output pins on the launchpad board. This included connecting the motor, piezo speaker, and ultrasonic sensor. After connecting the peripherals, we ran the program using artificial development kit signals to command the microcontroller. These are serial inputs from the PC sent to the microcontroller over UART. These commands eventually came from the object-detection development kit instead. We then verified the program on the microcontroller through this method, and ran multiple test cases. This includes different orders of execution of each peripheral, and varying durations of time between each of these. For example, one of the tests looked like: PC commands MCU to open the hatch via the motor, 2 seconds later, the PC commands the MCU to take an ultrasonic sensor reading, and finally 1 second later, the PC commands the MCU to close the hatch again, and play a sound on the piezo speaker. Through this testing, we were able to identify any bugs in the microcontroller's program. An example of a bug we encountered is the microcontroller not being able to complete a task if the command was sent too soon after the previous command. Through repeated testing, and program editing, we were able to have a working microcontroller and peripherals system that works smoothly and efficiently.

Once the previous step in the testing process is complete, we connected the object-detection development kit to the launchpad. The peripherals were not connected until we could confirm working communication between the microcontroller launchpad and development board. We first verified this communication by observing the Terminal tool in the Code Composer Studio. We added a modification in our microcontroller program which outputs the serial data being received by the microcontroller from the dev board in the CCS terminal on our PC. The serial data being sent contained test data for the sole purpose of verifying communication. Once we verified this, we then sent the specific commands for operating the peripherals to the microcontroller. We were able to verify that the MCU is receiving these commands and subsequently providing the required signals to its output pins by utilizing our oscilloscope and multimeter to measure these. Finally, only after this had been verified, could we connect our motor, piezo speaker, and ultrasonic sensor to the MCU launchpad. The next steps in the testing process were to use our PCB instead of the MCU launchpad to verify the system process.

6.3.1.2 Printed Circuit Board Testing

Once we received our printed circuit board from the manufacturer, we began testing and integrating the board with the rest of the system. Before connecting the PCB to any power source or other components, we conducted a continuity test. This test involved using a multimeter to verify each connection on the board. For example, if we wanted to verify that a resistor is properly connected to a specific pin on the microcontroller, we placed one of the multimeter probes on the pin, and the other on the pad of the resistor connected to the pin. If the multimeter displayed a very small resistance (<1ohm), we could confirm that the resistor is properly connected to the pin. This was repeated for all components on the board. We also verified MCU pin connections to the pin headers. One of the most important continuity verifications was confirming the connection between all grounds on the PCB, and verifying that none of the nodes on the PCB were shorted to ground. In addition to the continuity test, we ran a component value test. This involved testing the values of each of the RC components on the board using the multimeter. The RC components encompass the resistors and capacitors. Unfortunately, we were not able to test the inductance values of the inductors, since multimeters do not have this capability.

Once we have verified that the PCB was manufactured correctly, we moved on to testing the PCB with a power source. Our power source is a 12V rechargeable battery pack. Once we connected the battery to our PCB, we were able to measure each of the pins on the pin headers. We verified that the voltage regulators outputted the correct stepped-down voltages.

After we verified the power, we needed to flash the microcontroller on the PCB. This was done by using the PC and the MSP430G2ET launchpad. We first used Code Composer Studio in order to flash the program to our PCB MCU. We then took the MCU out of the DIP socket on the launchpad and placed it in the DIP socket on the PCB. Once we had our program flashed to the microcontroller, we were able to test the PCB's full functionality with the power source and PC. We used a USB to serial converter in order to connect the PC to the PCB. Our PC sent the same serial signals to the PCB as the object-detection kit will. We then verified that the PCB was working properly after it received each command. Measurements were again taken using the multimeter and oscilloscope. Next, we connected the PCB to the peripherals. We sent the same commands from the PC to the PCB, and verified that the motor, piezo speaker, and ultrasonic sensor were working properly.

Finally, we connected the development kit to the PCB in place of the computer. This became our completed system. We then did all of the previous tests mentioned previously and more. This included the various test cases of varying orders of execution and timings. After we successfully completed these tests, we had a completely working device ready for demonstration.

6.3.1.3 Solar Energy System Testing

Testing our solar cells, charge controller, and battery system will be vital to ensuring a working power source for our system. We will begin testing the solar energy system by testing our solar cells, and making sure that they provide the correct output voltage and current. In order to accomplish this, we will need to find a spot outside where we can get ample sunlight throughout the day. We will first test the voltage output of the solar cells by connecting a multimeter to the positive and negative nodes on the cells. After verifying the voltage output, we will proceed to the current. In order to test the maximum current output, we will first need to determine a load resistor to use. This load resistor will have the resistance equal to the voltage output of the solar cells divided by maximum current output we expect to see. This should be provided by the specifications on the solar cells. The resistor used should also have a voltage rating higher than the maximum voltage output from the solar cells. Once we have our resistor, we will connect the resistor, multimeter set to current mode, and solar cells in series. Under direct sunlight, we should be able to observe the maximum current output from the solar cells on the multimeter.

Once we have verified good current and voltage outputs from the solar cells, we will connect the solar cells to the rechargeable batteries through the charge controller. We will first measure the current between the solar cells and the regulator, then between the regulator and batteries. We will verify that these are good operating currents. Once this is done, we will connect the battery to the microcontroller and verify that the microcontroller has normal operation.

6.3.1.3.SD2 Solar Energy System Testing

Since the solar energy system was changed to a stretch goal in Senior Design 2, and was not achieved, we did not complete solar energy system testing.

6.3.1.4 ML Accelerator Hardware Testing and Prototyping

The ML accelerator hardware chosen for use in the smart bird feeder device by our team is the NVIDIA Jetson Nano Developer Kit. Our team first carried out testing on this developer kit unit to ensure that it functioned as advertised and did not contain any defects or unforeseen quirks. After the Jetson Nano Developer Kit unit had been verified in system testing, we moved it to the prototyping phase, where we utilized various tutorials and repositories provided by NVIDIA to prototype our own Jetson Nano test unit to ensure that it is capable of meeting the requirements specified for our smart bird feeder device. As mentioned prior, our group lead by Paul Amoruso and John Hauff took advantage of the early acquisition of the ML development board to be one of the first parts of the bird feeder that we tested and prototype. The following two subsections on the testing and prototyping of the machine learning acceleration hardware will discuss in detail the purpose for testing, various tests performed on the hardware such as checking if the Raspberry Pi camera would work, and the steps we took in prototyping by cloning the jetson-inference repository for instance.

6.3.1.4.SD2 ML Accelerator Hardware Testing and Prototyping

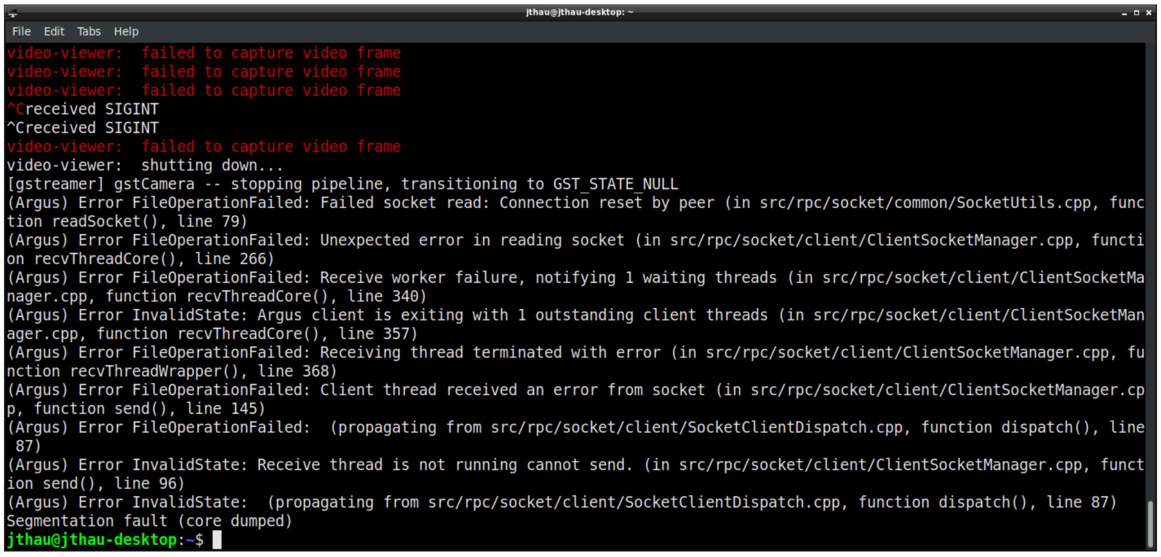
In Senior Design II, our team swapped the Jetson Nano 2GB Developer Kit for the Jetson Nano 4GB Developer Kit, yet the tests and prototyping done in the following sections which remain unchanged still yielded the same results.

6.3.1.4.1 ML Accelerator Hardware System Testing

This section will detail the procedures and results involved with testing the NVIDIA Jetson Nano Developer Kit. The purpose of these tests is to ensure that the Jetson Nano Developer Kit is properly functioning out of the box. These tests performed on the developer kit were picked by identifying what we were planning on doing with the kit, and then testing if the kit could do it. Moreover, it was highly important that we ensure the NVIDIA Jetson Nano Developer Kit worked in every aspect that we need it for as early in the processes of testing as possible, since any issues with the developer kit could cause for a delay in prototyping and the overall build of the Smart Bird Feeder.

6.3.1.4.1.1 Raspberry Pi Camera Module V2 Testing

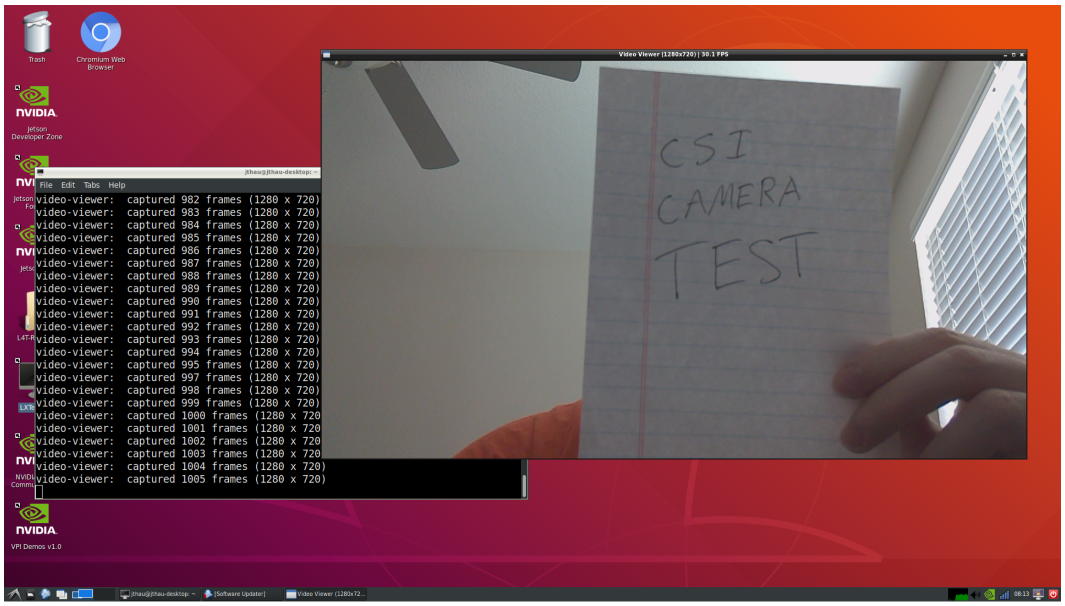
We tested the Raspberry Pi Camera Module V2 for proper functionality by connecting it to the MIPI CSI camera connector port on the Jetson Nano Developer Kit. With the developer kit turned on and providing power to the camera, we ran a program called *video-viewer*, which was included with the JetPack SDK, to test the camera. As Figure 25 shows, the program did not run properly at first, and errors were displayed as output to the console. We realized that connecting the camera before powering on the developer kit was required for the camera to function properly, and that the camera would often slip out of the CSI port on the developer kit. We solved this issue by connecting the camera to the CSI port before connecting the Jetson Nano to power, and by ensuring that the CSI port latch was fully secured before powering on the Jetson Nano. Once we figured this problem out, the camera functioned properly, and the program we ran opened a floating window that showed a live camera feed.



```
video-viewer:  failed to capture video frame
video-viewer:  failed to capture video frame
video-viewer:  failed to capture video frame
^Creceived SIGINT
^Creceived SIGINT
video-viewer:  failed to capture video frame
video-viewer:  shutting down...
[gstreamer] gstCamera -- stopping pipeline, transitioning to GST_STATE_NULL
(Argus) Error FileOperationFailed: Failed socket read: Connection reset by peer (in src/rpc/socket/common/SocketUtils.cpp, func
tion readSocket(), line 79)
(Argus) Error FileOperationFailed: Unexpected error in reading socket (in src/rpc/socket/client/ClientSocketManager.cpp, functi
on recvThreadCore(), line 266)
(Argus) Error FileOperationFailed: Receive worker failure, notifying 1 waiting threads (in src/rpc/socket/client/ClientSocketMa
nager.cpp, function recvThreadCore(), line 340)
(Argus) Error InvalidState: Argus client is exiting with 1 outstanding client threads (in src/rpc/socket/client/ClientSocketMan
ager.cpp, function recvThreadCore(), line 357)
(Argus) Error FileOperationFailed: Receiving thread terminated with error (in src/rpc/socket/client/ClientSocketManager.cpp, fu
nction recvThreadWrapper(), line 368)
(Argus) Error FileOperationFailed: Client thread received an error from socket (in src/rpc/socket/client/ClientSocketManager.cp
p, function send(), line 145)
(Argus) Error FileOperationFailed:  (propagating from src/rpc/socket/client/SocketClientDispatch.cpp, function dispatch(), line
 87)
(Argus) Error InvalidState: Receive thread is not running cannot send. (in src/rpc/socket/client/ClientSocketManager.cpp, funct
ion send(), line 96)
(Argus) Error InvalidState:  (propagating from src/rpc/socket/client/SocketClientDispatch.cpp, function dispatch(), line 87)
Segmentation fault (core dumped)
jthau@jthau-desktop:~$ 
```

Figure 25 - Error during Raspberry Pi Camera Module V2 testing

As Figure 26 shows, once the camera testing program ran properly without errors, the output printed to the console was the frames-per-second (FPS) and resolution (1280x720) that the Raspberry Pi Camera Module V2 was operating at with each passing second. The floating window on the right side of Figure 26 shows that the camera was able to capture a live video feed containing a view of a piece of paper with text written on it in the frame.



```
video-viewer:  captured 982 frames (1280 x 720)
video-viewer:  captured 983 frames (1280 x 720)
video-viewer:  captured 984 frames (1280 x 720)
video-viewer:  captured 985 frames (1280 x 720)
video-viewer:  captured 986 frames (1280 x 720)
video-viewer:  captured 987 frames (1280 x 720)
video-viewer:  captured 988 frames (1280 x 720)
video-viewer:  captured 989 frames (1280 x 720)
video-viewer:  captured 990 frames (1280 x 720)
video-viewer:  captured 991 frames (1280 x 720)
video-viewer:  captured 992 frames (1280 x 720)
video-viewer:  captured 993 frames (1280 x 720)
video-viewer:  captured 994 frames (1280 x 720)
video-viewer:  captured 995 frames (1280 x 720)
video-viewer:  captured 997 frames (1280 x 720)
video-viewer:  captured 998 frames (1280 x 720)
video-viewer:  captured 999 frames (1280 x 720)
video-viewer:  captured 1000 frames (1280 x 720
video-viewer:  captured 1001 frames (1280 x 720
video-viewer:  captured 1002 frames (1280 x 720
video-viewer:  captured 1003 frames (1280 x 720
video-viewer:  captured 1004 frames (1280 x 720)
video-viewer:  captured 1005 frames (1280 x 720)
```

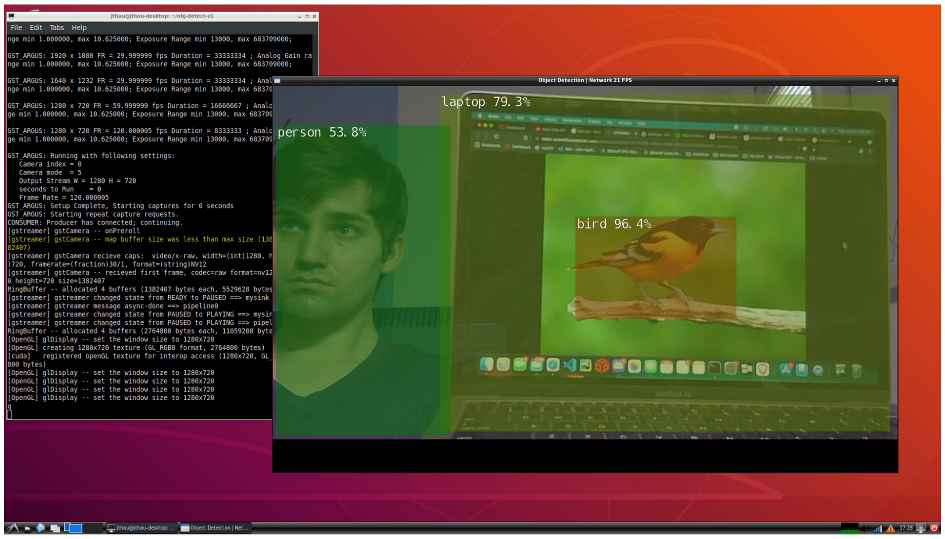
Figure 26 - Raspberry Pi Camera Module V2 test results

6.3.1.4.2 ML Accelerator Hardware System Prototyping

This section details the procedures that went into prototyping the NVIDIA Jetson Nano Developer Kit for use within our smart bird feeder device to perform object detection using machine learning to detect bird species and squirrels. Prototyping was first performed on the NVIDIA Jetson Nano Developer Kit by following along with the NVIDIA AI tutorials and courses [19], specifically the Jetson AI Fundamentals course. This course outlines how to initially setup the Jetson Nano with JetPack, which our team was able to successfully do with the help of this course. We started by cloning the *jetson-inference* repository [20] from GitHub, which contains all of the necessary tools to help deploy a deep learning model on the Jetson Nano Developer Kit.

6.3.1.4.2.1 Running A Pre-trained Object Detection Model

Our team wanted to prototype with the Jetson Nano by first ensuring that it could run a pre-trained ML model that could detect multiple object classes at once in each frame from a live camera stream. The ML model that was used for this step in our prototyping was the SSD-MobileNet-V2 model combined with the DetectNet model architecture from NVIDIA which was simply installed by following the tutorials mentioned previously. The SSD-MobileNet-V2 model contains 91 different COCO object classes, including a person, laptop, and bird class, which are the three classes being focused on for this stage of prototyping.



```
nge min 1.000000, max 10.625000; Exposure Range min 13000, max 683709000;

GST_ARGUS: 1920 x 1080 FR = 29.999999 fps Duration = 33333334 ; Analog Gain ra
nge min 1.000000, max 10.625000; Exposure Range min 13000, max 683709000;

GST_ARGUS: 1640 x 1232 FR = 29.999999 fps Duration = 33333334 ; Anal
nge min 1.000000, max 10.625000; Exposure Range min 13000, max 68370

GST_ARGUS: 1280 x 720 FR = 59.999999 fps Duration = 16666667 ; Analo
ge min 1.000000, max 10.625000; Exposure Range min 13000, max 683709

GST_ARGUS: 1280 x 720 FR = 120.000005 fps Duration = 8333333 ; Analo
ge min 1.000000, max 10.625000; Exposure Range min 13000, max 683709

GST_ARGUS: Running with following settings:
   Camera index = 0
   Camera mode  = 5
   Output Stream W = 1280 H = 720
   seconds to Run    = 0
   Frame Rate = 120.000005
GST_ARGUS: Setup Complete, Starting captures for 0 seconds
GST_ARGUS: Starting repeat capture requests.
CONSUMER: Producer has connected; continuing.
[gstreamer] gstCamera -- onPreroll
[gstreamer] gstCamera -- map buffer size was less than max size (138
82407)
[gstreamer] gstCamera recieve caps:  video/x-raw, width=(int)1280, h
)720, framerate=(fraction)30/1, format=(string)NV12
[gstreamer] gstCamera -- recieved first frame, codec=raw format=nv12
0 height=720 size=1382407
RingBuffer -- allocated 4 buffers (1382407 bytes each, 5529628 bytes
[gstreamer] gstreamer changed state from READY to PAUSED ==> mysink
[gstreamer] gstreamer message async-done ==> pipeline0
[gstreamer] gstreamer changed state from PAUSED to PLAYING ==> mysin
[gstreamer] gstreamer changed state from PAUSED to PLAYING ==> pipel
RingBuffer -- allocated 4 buffers (2764800 bytes each, 11059200 byte
[OpenGL] glDisplay -- set the window size to 1280x720
[OpenGL] creating 1280x720 texture (GL_RGB8 format, 2764800 bytes)
[cuda]   registered openGL texture for interop access (1280x720, GL_
800 bytes)
[OpenGL] glDisplay -- set the window size to 1280x720
[OpenGL] glDisplay -- set the window size to 1280x720
[OpenGL] glDisplay -- set the window size to 1280x720
[OpenGL] glDisplay -- set the window size to 1280x720
q
```

jthau@jthau-desktop: ~/obj-detect-v1
File Edit Tabs Help
Object Detection | Network 21 FPS
laptop 79.3%
person 53.8%
bird 96.4%
jthau@jthau-desktop: ...
Object Detection | Net...

Figure 27 - NVIDIA Jetson Nano 2GB object detection prototyping results

Figure 27 displays the results of prototyping with the NVIDIA Jetson Nano Developer Kit to perform object detection on various common objects. As Figure 27 shows, the prototyping performed on the NVIDIA Jetson Nano was a success. The window on the left side of Figure 27 shows the console output from the ML model printing in near real-time as it runs. This console output contains important information about what the ML model is doing, and with this tool, the team member prototyping the Jetson Nano Developer Kit could easily notice and record any problems or errors erupting from running the ML model, since the error would be printed out to the console. The floating window on the right side of Figure 27 contains a frame from the view of the Raspberry Pi Camera Module V2, and this frame shows multiple objects in the frame, as well as the frame rate that the program was capturing at the time. It can be seen that the ML model was able to detect one of our team members as a person with 53.8% certainty, which is not commonly considered a very high level of accuracy, but this confidence level is dependent on many factors, and fluctuates frame-by-frame, growing higher and lower with each new frame. The lower confidence level displayed for the person detected in Figure 27 could be due in part to the team member (person) being further from the camera than the bird and laptop, which were detected with accuracies of 96.4% and 79.3%, respectively. This could mean that the bird and laptop in the foreground of the frame caught by the Raspberry Pi Camera Module V2 may have partially obscured the person from the view of the camera and the ML by pushing the person into the background of the frame. This is an obstacle that we must not ignore when implementing this functionality in our final smart bird feeder device.

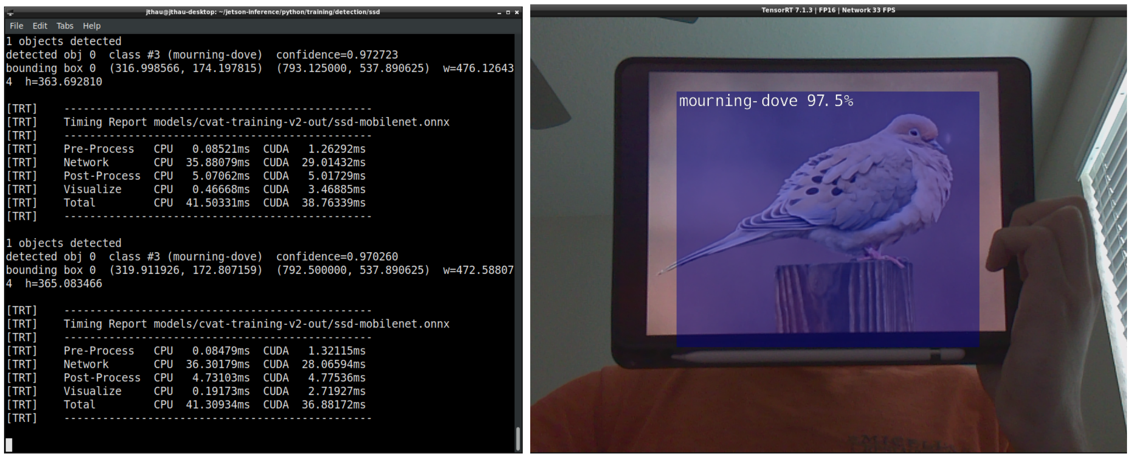
6.3.1.4.2.2 Retraining and Running A Pre-trained Object Detection Model to Detect Bird Species and Squirrels

The next step in prototyping with the NVIDIA Jetson Nano Developer Kit was for our team to take a pre-trained ML model, in this case SSD-MobileNet-V2, and retrain that model to perform object detection on the custom classes that we specify. In order to retrain the ML model to detect specific species of birds or squirrels, we first had to collect datasets of images for squirrels and each bird species, then label the datasets. The labeling could be done automatically using customized software, or manually by drawing bounding boxes around the objects in each image of the dataset, then we may still use a program like CVAT to create the metadata needed for the labeled dataset. At this stage, we chose the latter approach.

6.3.1.4.2.2.1 Detecting Bird Species

The first main task in the prototyping stage for live video feed object detection of birds species and squirrels was to detect bird species, the first subtask of which involved locating and downloading a suitable dataset with the bird species that we would like our device to detect. The dataset that we built comes from a mixture of the *Caltech-UCSD Birds-200-2011* dataset [21] and the *275 Bird Species* [22] found on Kaggle, an online source for downloadable datasets. For the purposes of saving time and effort at this stage in prototyping, rather than labeling and training with all fifteen species of birds that we plan to have our smart bird feeder capable of detecting, we only chose seven bird species to label and include as classes for the model to detect this time. The seven bird species that we trained with for this stage of prototyping were (represented as labels) *red-bellied-woodpecker, northern-mockingbird, mourning-dove, common-grackle, gray-catbird, cardinal, and blue-jay*.

Once our images were gathered together, we used the CVAT software to annotate each image in our dataset with a bounding box, then CVAT generated XML files using the PASCAL VOC dataset format for use with our annotated images to train the model. We used a Python script to run the training for the model, then once the model was trained, we exported the model as a onnx file. With this, the ML model was ready to detect objects from a live camera feed. Figure 28 shows a portion of the outcome of our prototyping. The image on the left in Figure 28 shows the console output from running the ML model on a live camera feed, displaying the CPU and CUDA usage while the model is running, and printing a message each time an object is detected in the camera's view, along with the confidence level of the model's object detection accuracy at that moment. The image on the right in Figure 28 shows the live camera feed with a bounding box tracking the detected bird species in the frame. Because the bounding box is colored blue, shows the label of *mourning-dove*, and lists a confidence level of 97.5% (very high confidence), it can be determined that the ML model was successfully able to detect the mourning dove from the camera's live video feed. Please note that the two images in Figure 28 were captured separately, so the confidence levels differ slightly.


jthau@jthau-desktop: ~/jetson-inference/python/training/detection/ssd
File Edit Tabs Help
TensorRT 7.1.3 | FP16 | Network 33 FPS
mourning-dove 97.5%

```
1 objects detected
detected obj 0  class #3 (mourning-dove)  confidence=0.972723
bounding box 0  (316.998566, 174.197815)  (793.125000, 537.890625)  w=476.12643
4  h=363.692810

[TRT]    ---------------------------------------------
[TRT]    Timing Report models/cvat-training-v2-out/ssd-mobilenet.onnx
[TRT]    ---------------------------------------------
[TRT]    Pre-Process   CPU   0.08521ms  CUDA   1.26292ms
[TRT]    Network       CPU  35.88079ms  CUDA  29.01432ms
[TRT]    Post-Process  CPU   5.07062ms  CUDA   5.01729ms
[TRT]    Visualize     CPU   0.46668ms  CUDA   3.46885ms
[TRT]    Total         CPU  41.50331ms  CUDA  38.76339ms
[TRT]    ---------------------------------------------

1 objects detected
detected obj 0  class #3 (mourning-dove)  confidence=0.970260
bounding box 0  (319.911926, 172.807159)  (792.500000, 537.890625)  w=472.58807
4  h=365.083466

[TRT]    ---------------------------------------------
[TRT]    Timing Report models/cvat-training-v2-out/ssd-mobilenet.onnx
[TRT]    ---------------------------------------------
[TRT]    Pre-Process   CPU   0.08479ms  CUDA   1.32115ms
[TRT]    Network       CPU  36.30179ms  CUDA  28.06594ms
[TRT]    Post-Process  CPU   4.73103ms  CUDA   4.77536ms
[TRT]    Visualize     CPU   0.19173ms  CUDA   2.71927ms
[TRT]    Total         CPU  41.30934ms  CUDA  36.88172ms
[TRT]    ---------------------------------------------
```

Figure 28 - Object detection of Mourning Dove

Furthermore, we introduced images of each of the seven bird species that the ML model was trained to detect in this stage one-by-one to the camera. Table 18 shows the confidence level returned by the model whenever it would correctly identify the bird species in the image currently being shown to the camera. Please note that these confidence levels are median confidence levels recorded for testing purposes, and in practice, the confidence level may vary to a higher or lower confidence with each passing frame. As Table 18 shows, the model was able to detect the common grackle with the greatest accuracy and confidence, while it had the most trouble with detecting Blue Jays, and could not detect the red-bellied woodpecker in the test image shown to the camera at all, no matter the position of the image.

Table 18 - Bird species detection prototyping results

Bird Species	Object Detection Confidence
blue-jay	51.0%
cardinal	72.1%
common-grackle	98.0%
gray-catbird	77.8%
mourning-dove	97.5%
northern-mockingbird	87.9%
red-bellied-woodpecker	-

6.3.1.4.2.2.1.SD2 Detecting Bird Species

In Senior Design I, the object detection was not equally accurate on all trained bird species, and was especially unreliable in low-light environments and when the bodies of birds were partially obscured. To solve this issue, in Senior Design II, our team re-trained the ML model several times before producing a suitable model to perform object detection of all 15 common Florida backyard bird species, and squirrels as well, with ≥90% confidence the majority of the time.

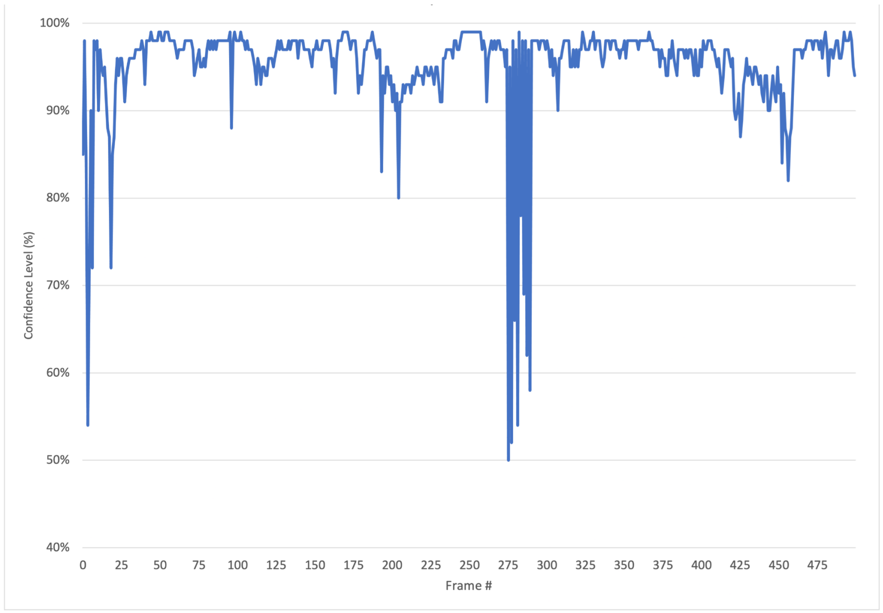


Figure 29.SD2 - Object detection confidence levels of bird species captured over 500 frames

Figure 29.SD2 and Figure 30.SD2 support this claim, as we measured and averaged the confidence level of the object detection on a moving bird image, and discovered that only ~6% of the time, on average, the object detection confidence would drop below 90%. We trained the best version of the object detection model by focusing more on the heads of birds and squirrels, rather than their entire bodies. This meant that when the bodies of birds and squirrels were obscured (possibly due to the limited field of view of the camera module), as long as the head of the animal was in view of the camera, it would be detected confidently. It was especially convenient to have the ML model only identify the heads of birds, because this meant that an image of a bird would only be captured when its head is in good view of the camera, thus increasing the chances of getting a nice bird memory image.

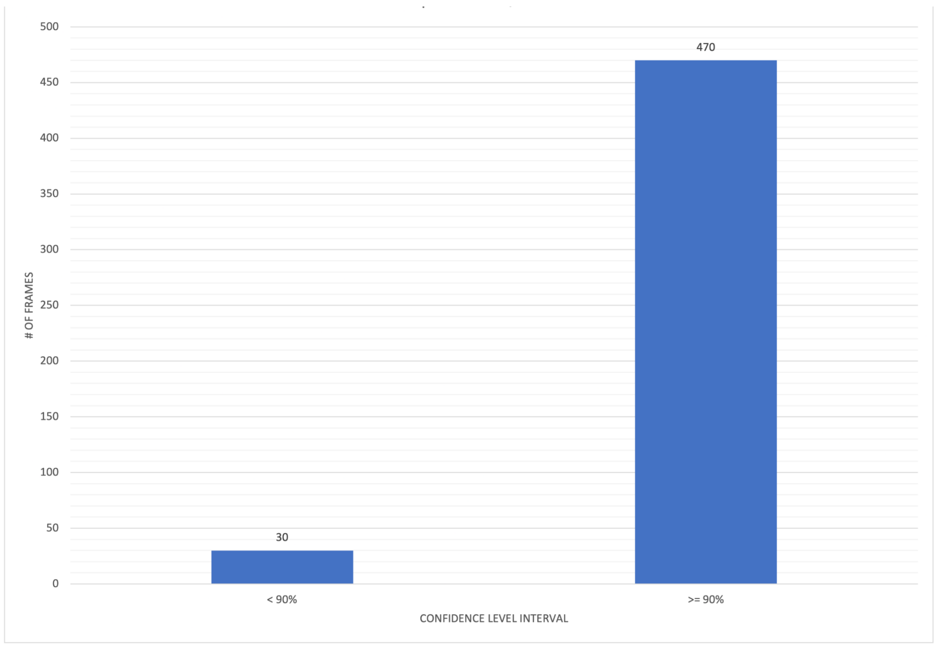


Figure 30.SD2 - Count of confidence levels captured with less than or equal to 90% confidence over 500 frames

6.3.1.4.2.2.2 Detecting Squirrels

Although prototyping has not yet been completed for the object detection of squirrels from a live video feed, the process to do so will be nearly identical to the process for detecting bird species. This phase of the object detection model development will be further explored in Senior Design II. The main difference between the tests run for detecting squirrels and detecting bird species is the number of classes of objects that will be tested for recognition. In this case, the ML model is still trained to detect bird species, but now it has an additional object class of *squirrel* which the model can label an object in the camera feed with. Therefore, the success of this stage of prototyping depends on the object detection algorithm's ability to differentiate between an object that is a bird (more specifically, the species of bird), and an object that is a squirrel. The Smart Bird Feeder's alarm system will be tested separately, but eventually, our team plans to test both the alarm system and machine learning model in combination.

6.3.1.4.2.2.2.SD2 Detecting Squirrels

In Senior Design II, our team re-trained the ML model to focus on detecting the heads of squirrels, which resulted in a detection interval of ≥90% confidence the majority of the time.

6.3.1.4 Ultrasonic Sensor Testing

The ultrasonic sensor hardware was vital to the success of the project. It's important that the readings were accurate and didn't produce false positives. The ultrasonic sensor test was used to ensure that the depth accuracy and uncertainty were in an acceptable state for the project's requirements. The testing process required some software elements to read the results, however this was unlike the software test that needed to be performed on the ultrasonic sensor to ensure that the assigned ranges provided the correct effects.

This test was strictly to test the hardware elements of the ultrasonic sensor. To perform this test, the team set up a blank surface to reflect the sensor light back. The surface distance was measured from starting point to surface location using a tape measure, by setting proper distance marks. The sensor was then set into place, and the team measured the readings accuracy. The uncertainty of the reading needed to be no more than half an inch from complete accuracy for the team to consider the ultrasonic sensor uncertainty acceptable. This first test acted as a basis for the sensor's functionality under perfect conditions. However, our practical application included a few more factors, including an uneven surface, and finally an enclosed space.

To perform the second test, the surface being tested had to be uneven, similar to the bird feed the sensor will be reading in its practical use. The material being used was bird feed in order to maximize accuracy of the environment. The sensor was hung from the ceiling, and the bird feed dropped into a bowl to create an uneven gravity effect. The team then took a string and stretched it from the sensor location to the highest point in the middle of the bird feed bowl. It was marked, and then measured using a tape measure. The sensor's reading should have been no more than ½ an inch from the theoretical result.

For the final test, an enclosed space was replicated. We then repeated the setup procedure of the previous test, however we covered the testing environment with a cardboard box to replicate an enclosed space. The experimental reading had to be no more than ½ an inch from the theoretical reading. This process ensured that the sensor was operating properly and ready to be incorporated into the device.

6.3.1.5 Alarm Testing

The alarm system acted as the main deterrent for squirrels to stay away from the bird feeding device. While the hatch mechanism ensured there was no expectation for food from the feeding device, the longer the squirrel persisted near the bird feeder, the less time would be spent utilizing the feeder for its primary purpose. So the alarm system was a vital component that needed to function for the sake of the bird feeder. The alarm needed to undergo testing to ensure it functions under the right circumstances and play at the proper decibel to avoid damaging ears or annoying nearby neighbors.

In order to pass the test, the alarm needed to play at 75-90 decibels to deter squirrels yet stay safe for animals and humans. The distance the alarm system is away from the creature hearing it will play a role in the decibels registered. Thus, we can not rely solely on the datasheet for the alarm to determine what decibel the alarm will play at near the intruder. Since these alarms also change decibels depending on the power source applied, the microcontroller will need to be used to apply the correct amount of power. In order to perform the test, a decibel measuring application will be used. The application will be tested by playing a sound from a computer system rated at a particular decibel. Once the application has been checked for accuracy, it can be used in the testing process.

The alarm system will be placed 8 inches from the decibel testing application, which will effectively mimic the distance from the alarm to the perpetrator in a practical application. The alarm will then be powered on, and the decibel monitoring application will measure the decibel of the alarm output. In order for the alarm to pass this test, the experimental result should match the 75-90 decibel range listed above. This test will ensure the desired range is met.

In order to ensure the alarm system does not exceed 90 decibels in any circumstances, a safety test will be performed. In the situation where an animal or person decides to get closer to the alarm system than is anticipated, this test will ensure that the audible range is not considered dangerous. To perform this test, the decibel testing application will be placed directly next to the alarm. In order to pass this safety test, the decibel data can not exceed 90 decibels.

6.3.1.6 Motor Testing

In testing the various parts that make up our project, the motor needed to withstand closing the hatch on the bird food to keep the pesky squirrels away was definitely on the list, as it plays a small but crucial part in the Smart Bird Feeder. The type of motor that we investigated to be on paper the most qualified for our project was a servo motor, known as the FEETECH FS90-FB Micro Servo. Therefore, at the time of acquiring the servo motor for the utilization of this bird feeder, we went straight ahead into testing. Thus, in this section we provide in-depth steps taken to test the functionality and overall quality of a FEETECH FS90-FB Micro Servo motor, by verifying that it can do such things as rotate a specific amount of degrees in a certain amount of time and force with and without the hatch attached to it.

When buying servo motors, it may not come across our minds as an initial thought that it is an important step to perform some functionality test, yet it is true that occasionally a servo motor might not rotate when trying to initially use them, which could lead to be possibly labeled as faulty without thorough testing. While testing processes of the motor can be easily implemented by just instructing the motor to rotate while being plugged into the MSP430G2553 board, and if that does not work, it can be simply verified of its condition of functionality by building a relatively quick circuit on a breadboard, where we can attempt to rotate it in a forward and backwards direction to see if it rotates accordingly. Knowing that the angle of rotation is controlled by the duration of the pulse, since servo motors work on the pulse width modulation principle. Furthermore, we also know that the motor is able to rotate from zero to one hundred and eighty degrees. Thus, we can build a relatively simple circuit to test the degree of rotation by attempting to control the motor via electrical pulses of a proper width. Therefore, by applying pulses of one, one and a half, or even two milliseconds we can determine if the motor works properly depending on the degree of rotation. Moreover, we can apply our knowledge of applying pulse to rotate the motor, to ensuring it can also rotate back and forth.

With this said, there will be two parts of the motor testing, the first will be without the hatch attached to the motor, and second with the hatch attached to the motor. The tests that will be undergone when the hatch is not attached to the motor will consist of testing the degree of rotation, the speed and time. In order to test for the specific degrees based on the width of the pulse applied, we will start off by applying common recommended pulse widths to the servo motor to identify how many degrees it rotates. Then we will mark the pulse width and the degree of rotation down in our notes for comparison later on when testing with the hatch. Next on the list of testing, we will observe the time it takes the motor without the hatch to rotate the various degrees such as 90 degrees or 180 degrees when the pulse is applied to it.

The next step will be to attach the physical hatch to the motor so that we can determine if the FEETECH FS90-FB Micro Servo motor is able to move the weight of the hatch without much or any noticeable complication, if any. The tests that will be used to determine how much the hatch affects the servo motor when connected will be tested based on performing the tests performed without the hatch and then comparing the results. For example, we will apply the same pulse widths to the servo motor as we did without the hatch to identify how many degrees it rotates. With these notes on specific degrees and the time it takes to rotate a particular amount of degrees will be directly applied to comparing with the tests earlier, to see how much of a difference if much at all is there when the weight of the hatch has to be moved. Moreover, this will correlate with testing the force at which the motor can move the hatch. In all, the testing described, will be another small yet important step in verifying that our only physical barrier between the bird food and the squirrels is able to actually work as it was intended to.

The tests listed above are crucial for the implementation of controlling the hatch to block off the squeals from munching on the bird food. For instance, by knowing that with the extra weight of the hatch of the motor might make the degree of rotation smaller that without the hatch, we are able to calibrate and determine the difference needed to be added to the applied pulse width. Moreover, by determining how long it takes the motor to rotate the hatch various degrees is crucial for timing when the motor needs to be signaled to close the hatch when a squirrel is detected, so that a squirrel does not get away with eating any of the food before the hatch can close all the way.

6.3.4 Software Testing

The other aspect of our smart bird feeder device system that was tested is the system software. System software includes any component of the smart bird feeder device that is required for proper usage of the device, but does not rely on any hardware that is unique or specifically calibrated to be used with the smart bird feeder device. A list of the system software that was tested in this procedure, along with how each system hardware component was tested, will be included below.

- **Planned Testing:**
  - **Sign up -** User information storage and email verification.
  - **Forgot password -** Forgot password functionality.
  - **Photo storage** - Properly take and store photos.
  - **Livestream** - Access livestream functionality.
  - **Notifications** - Low bird feed, livestream, predator detection, and power notification.
  - **UI and Buttons** - Making sure buttons are clickable and perform their expected functions across the UI application.

6.3.4.SD2 Software Testing

The testing protocols stayed largely the same besides removing tests that revolve around features that were moved to stretch goals and weren't able to be implemented.

6.3.4.1 Sign Up Testing

In order for a user to utilize the smart functionality of the bird feeder, they must sign up for the application. Many of the features rely on syncing the bird feeder to store information attributed to a specific user, and thus this functionality is essential. In order for a user to be registered for the application, their email and password must be stored inside of the MongoDB database, and receive an ID associated with their account that will be used to identify the user. There are two collections where data is stored during this process. A temporary collection will be used to save a user's information before they've completed their verification process. Once the verification process is complete, a user's information will be transferred to a permanent database. These tests will revolve around testing the sign up process and ensure the database is working properly. The criteria for passing will change depending on the test listed below.

The first test will focus around the temporary database collection. On the sign up page, a test email and password will be inputted. This information should then be stored inside of the temporary database. This test will pass if the information has been correctly stored in the temporary database yet not be submitted to the permanent database. To check the results, mongoDB compass will be used to check the database for the temporary and permanent collections.

The second test will focus around the verification process. A user should receive a verification email soon after signing up for the application in order to verify their account. Their login information is then sent to the permanent database from the temporary database. In order for this test to pass, the users should receive the email within 1 hour, and the users information should be immediately nested in the permanent database. Also, as a separate check, the data should be deleted from the temporary database. A timer will be used from the test sign up confirmation button to the arrival of the email link. The link will then be clicked, and to check the entry to the MongoDB database will be used.

The final test will ensure that under the circumstances where a repeat email is used to attempt to create multiple accounts, the system will not allow this. To perform this test, an account with an existing email will be used to attempt and sign up. To pass this test, the sign up page should submit an error message that the email has already been used to create an account.

6.3.4.1.SD2 Sign Up Testing

The testing for this section revolving around the email confirmation process was removed in Senior Design II due to this functionality becoming a stretch goal and not completed before the end of the project. So testing involving gathering the user's information and inputting it in the mongoDB database above was completed.

6.3.4.2 Photo Storage Testing

These tests will focus on the software components revolving around sending and storing photos. It's imperative to the design of our smart bird feeder that users are able to save the memories of the species of bird the device has recognized. The tests will involve the start sequence, sending the photo over a WiFi network, storing the photo in the database, new memories being created and accessed, and then the ability to download the photo. The criteria necessary to pass these tests will be varied depending on the test, but will build up to include every component of the photo sending process.

The first test will focus on the machine learning software knowing when to begin the photo storage process. When there is a livestream event, a photo will be taken at the point of highest species recognition that passes 80% accuracy. In order for this test to pass, the photo needs to be snapped when the detection level passes 80% accuracy, and then again at each 5 percent increase in species recognition accuracy. The machine learning device will then commit those photos to local storage. For this test, the machine learning processor needs to sense a bird species with 95% accuracy or higher. Then the local storage will be checked to ensure that there are 4 photos taken at each appropriate interval. Additional tests will then be performed at these intervals: 80%, 85%, and 90% corresponding to 1, 2, and 3 photos.

The second test is a repeat of the previous one, but the detection accuracy needs to not pass 80%. The test will pass if there are no photos stored in local memory after this event.

The second group of testing will focus on sending the photo to the application over the WiFi network. This process will begin at the conclusion of a livestream event and will begin with the photos stored in the local storage on the machine learning processor. The photo needs to be taken from local storage and properly sent to the application and stored in the database. In order for this test to pass, the photo with the highest percentage accuracy in local storage needs to arrive at the database associated with the user. To run the test, a livestream will be started and finished, ensuring the species accuracy was 90% or higher during the event. Then the tester should check the database using mongoDB to see if the photo arrived and is associated with their test ID. Then the tester will check the photo to ensure the highest quality photo was sent. As an additional test, they will then check local storage on the machine learning processor to ensure that local storage was deleted after the photo was sent.

The next test will focus on a new memory being created and shared with the user after a feeding event. This test should be performed directly after the previous test as the setup conditions are identical. Upon the database receiving the picture, the user should be able to access the memory associated with the feeding event, and see the picture. This test passes if the correct photo is incorporated with a new memory on the testers UI. The tester should click the new memory, and verify the results to match what's listed above. As an additional test, the user should be able to download the photo from the database, as well as favorite the memory so it shows up in the "favorites" section.

6.3.4.3 Livestream Testing

This testing section will focus on livestream functionality. The ability to stream the bird feeding directly to your device of choice is a key functionality. The test will pass if the livestream button opens up a functioning livestream of the feeder. To test, the user will press the livestream button and see if the image quality is good. This test is much more subjective than the other test, as the image quality that is acceptable is up for debate. The testing group will have to use their judgement to determine acceptable quality.

6.3.4.3.SD2 Livestream Testing

The team was not able to incorporate live streaming and therefore this test was not completed.

6.3.4.4 Notifications Testing

The heart of any smart product is its ability to notify the user of the various capabilities. Our notification system touches nearly every aspect of the project, and enables many of the features we find necessary to function properly. These tests will cover all the notifications that should arise under various circumstances, and test their functionality regarding interacting with the notification system. The requirements for passing the test will differ depending on the test performed.

The first test will revolve around the notification for low bird feed, ensuring that the ultrasonic sensor reading a range considered too low will send a response to the application. The test will pass if the notification is received on the application. The test will begin by removing food until the proper threshold is reached, and then the tester will wait for the notification.

The second test will focus around receiving a notification for a bird feeding event. The notification should also have the species of the bird currently feeding nested inside so the user can choose whether or not to view the feeding process. In order for this test to pass, the notification should appear directly after the machine learning processor senses the bird feeding and determines its species. The notification should also have the species listed. To test, the team will show an image of a particular species of bird, and wait for the notification process to begin.

The third test will center around the predator defense mechanism notification. When a squirrel attempts to eat the bird feed, the user will receive a warning notification to see the deterring process in action. This test will pass if the squirrel causes the predator notification to pop. The test will be run by simulating a squirrel eating, and watching the results to check for passing functionality.

The final test will focus on the new memory notification. This should appear following species detection and let the user know that a memory has been saved of the bird. This notification should also be clickable, which will lead the user to the memories page to see the new memory. To perform this test, the tester will simulate a bird feeding with a picture, and view the results to see if the test receives a pass.

6.3.4.4.SD2 Notifications Testing

The tests described in the section above were performed in Senior Design II many times, and luckily, all test results were positive, and notifications were received by the user in 10 seconds or less on every occasion. Evidence of the success of most of these tests can be found in our demonstration videos.

6.3.4.5 UI and Buttons Testing

These tests will be less structured than the testing listed above, and focus more around the performance of the application as a whole. It’s important that every button in the application works as intended. In order to perform this test, each button that should be clickable or have functionality will be listed out and tested individually for its performance. A checklist will thus be created, and the tester will cycle through the application to ensure that each button or UI element is behaving as expected.

Some examples would be the separate pages listed at the bottom of the main screen, clicking the memories page link should take the user to the memories page. The notification bell should bring up a panel for recent notifications, which should be clickable as well. Some buttons will also only be available during certain times based on specific events corresponding to the bird feeder. Testing these buttons will require the set up of the test listed above, so there will be repeated tests.

Running these functionality tests must be done at the end of the development process, to ensure that the added features to the application did not break any previously tested features. To pass this series of tests, all buttons and UI elements must work as intended. If the test fails, and the code is changed to accommodate, it’s important to categorize anything that could have possibly changed by the new code, and retest. This will be an ongoing process that will ensure the full and final functionality of the application.

6.3.4.5.SD2 UI and Buttons Testing

UI testing was successfully performed in Senior Design II, and all buttons, notifications, scrollable and draggable items properly functioned in the application. The tabular screen layout turned out to be a very desirable design choice, since during testing, the user could navigate between screens nearly instantaneously upon selection of another tab.

7. Administration

This section will document how our team will manage our time, budget and financing in order to accomplish our project in a timely fashion at a price point that is reasonable for a Smart Bird Feeder. This section will also discuss the tools that our team used to keep our development process organized and efficient.

7.1 Project Budgeting and Financing

Currently, this project is being financed entirely by our project members. The maximum budget for this project is currently $400 USD, split evenly among the four project members. We hope to minimize costs and avoid going over budget.

Table 19 - Project budgeting

Item	Price (USD)	Quantity
Camera	$14.99 to $49.99	1
microSD Card (64 GB)	$9.99 to $19.99	1
Developer Kit for AI Computer Vision	$59.99 to $99.99	1
Microcontroller	$1.00 to $3.00	1
Misc. Parts for PCB	$9.99 to $19.99	1
Bird Feeder + Electronics Housing	~$49.99	1
Bird Feed	~$19.99	1+
Microphone	$1.18	1
Speaker	$4.60	1+
Micro Servo Motor (6V)	$11.95	1
10 W Polycrystalline Solar Panel Charger	$29.99	1
IR Sensor	~$9.00	1
Solar Charge Regulator	$21.99	1
6V Rechargeable Battery	$22.22	1
Total	**$256.88 to $343.88**	**–**

Table 19 shows the current list of materials and costs that our project budget must revolve around. It is important to note that a number of these prices are merely estimations, and it is possible for the budget to vary in the future. One of the goals in our group is to keep costs at a low price, while also delivering a well rounded product.

7.1.SD2 Project Budgeting and Financing

In Senior Design II, the budget changed to reflect any design changes, including removing, adding, or replacing components. The team managed to stay close to the target price of $400 USD. The updated budget list is shown below in Table 19.2.SD2.

Table 19.SD2 - Updated project budgeting

	Item	Quantity	Price/Unit ($)	Cost ($)
1	NVIDIA Jetson Nano 4GB Dev Kit	1	99.00	99.00
2	Raspberry Pi Camera V2-8 MP	1	28.99	28.99
3	Intel Dual Band Wireless-AC 8265 Device	1	25.99	25.99
4	DC Power Pigtails Cable	1	8.99	8.99
5	4Pcs SG90 9g Micro Servos	1	10.99	10.99
6	HiLetgo Cp2102 USB 2.0 to ttl module	1	5.39	5.39
7	E-outstanding 5PCS PCB Mount	1	5.99	5.99
8	TalentCell Rechargeable 12 V 6000 mAh DC Battery Pack	1	40.86	40.86
9	Bird feeder	2	22.00	44.00
10	PIR sensor	1	9.99	9.99

Table 19.2.SD2 - Updated project budgeting *(continued)*

11	Nvidia fan	1	13.50	13.50
12	PVC flooring	1	20.19	20.19
13	Hatch	1	6.99	6.99
14	Plexiglass	1	30.06	30.06
15	DIP socket	1	3.99	3.99
16	TO-220 Heatsink	3	0.39	1.17
17	USB 1130-15	1	0.82	0.82
18	124 ohm SMD Res	5	0.26	1.3
19	1 ohm SMD Res	5	0.37	1.85
20	0 ohm SMD Res	15	0.41	6.21
21	365 ohm SMD Res	8	0.33	2.64
22	DC Port	1	0.64	0.64
23	PCB	5	6.40	31.98
24	USB to Barrel Jack cable	1	8.99	8.99
25	SWITCH SLIDE SPST 8.5A 125V	1	1.29	1.29
26	MSP430G2553 Microcontroller	1	3.45	3.45
27	PCB fan	1	~9.99	0.00 (recycled)
28	Ultrasonic Sensor	1	3.95	0.00 (recycled)
Grand Total				**$411.81**

7.2. Project Milestones

Senior Design I and II will test our ability to schedule our time and plan for deadlines. Each deadline will require us to coordinate with each team member and allocate our time in the most efficient way possible. In order to help plan through the two semesters, two tables were created, which list the important milestones ahead and their dates. Through Senior Design I, our main goals will relate to the technology investigation and development of our project. For each major milestone, we are to increase the length of our project documentation, and by the end of the course, we should have a complete 120 page document. This documentation should thoroughly document the complexities and complete break-down of our project design. Through Senior Design II, we will then be bringing this project design to life. This semester will include plenty of troubleshooting, redesign, and modification, as we attempt to produce the best version of our device. By the end of Senior Design II, we will have created a final project document and presentation, on top of our completed device. Following these schedules and working consistently towards our goals will allow us to meet these deadlines comfortably.

7.2.1 Semester I (Senior Design I)

Table 20 - Senior design I milestones

Week #	Dates	Milestone Description
1	5/17/2021 - 5/23/2021	Begin forming a project group and think about project ideas.
2	5/24/2021 - 5/30/2021	Study previous semester group projects and listen to idea recommendations to better form an idea for our project.
	5/24/2021 - 5/30/2021	Work on the Divide & Conquer (D&C), and get a consensus on who wants to be in charge of each block in the diagram.
4	6/7/2021 - 6/13/2021	Finalize the initial Divide & Conquer paper and submit it.
6	6/21/2021 - 6/27/2021	Finalize the updated Divide & Conquer version 2.0 document and submit it by Friday.

Table 20.1 - Senior design I milestones *(continued)*

Week #	Dates	Milestone Description
8	7/5/2021 - 7/11/2021	Finalize the 60 page Draft Senior Design I Documentation, then submit it.
10	7/19/2021 - 7/25/2021	Finalize the 100 page updated report and submit it by Friday.
12	8/2/2021 - 8/8/2021	Ensure that the documentation is finalized and perform any last minute corrections and finalizations.
		Submit final documentation report by Tuesday.

7.2.2 Semester II (Senior Design II)

Table 21 - Senior design II milestones

Week #	Dates	Milestone Description
1	8/23/2021 - 8/29/2021	Ensure that bird feeder structure and electronic components designs are finalized for assembly of version 1 of our device.
		Order parts for the first completed version of our smart bird feeder design.
6	9/27/2021 - 9/3/2021	Bird feeder structure should be complete and all electrical components are ready or close to ready to be installed.
7	10/4/2021 - 10/17/2021	Install the bird feeder in a testing location outside and perform testing to ensure it functions properly.
13	11/15/2021 - 11/21/2021	Prepare presentations so the device can be adequately demonstrated with the proper amount of information and functionality to prove that our project was successful.
15	11/29/2021 - 11/5/2021	Final project presentation and evaluation.

7.3 Project Member Contributions

In order to maximize the coherence of our team's project workflow, we chose to create specific (tentative) roles to assign to each member of this project. These roles serve to group together the various related tasks within this project. These roles were also created as a means of avoiding the potential of role ambiguity amongst our team members, which (if left unaddressed) can quickly lead to stunted project progress, poor efficiency, and an unbalanced distribution of workload for a given project team member. Of course, the roles listed here are only loosely binding, and should a team member seek to contribute to a project task which falls outside the scope of their given role, they are welcome to do so under the majority grant of the remaining team members. The roles that were used to complete this project, along with short descriptions of what the role consists of, and which team members held a given role, are as follows:

- ***Project Manager*** *- Organized team meetings, goals, and objectives. Also delegated various roles to team members and ensured that every necessary task was properly completed in a timely manner for the project to proceed properly. The project manager was chosen as the default representative to directly interact with the instructors of the Senior Design course, and consolidated questions and issues that arose within the team with regards to the project so that they may be efficiently and effectively answered and solved by an instructor, other members, or another mentor figure. The project manager was also responsible for taking notes at each project team meeting.*
    - Role assignee(s): John Hauff

- ***Software Engineer*** *- This role was assigned to project team members with experience with programming software and working with Command Line Interfaces (CLIs). Common tasks for this role were app development and programming embedded systems.*
    - Role assignee(s): Paul Amoruso, John Hauff, Matthew Wilkinson

- ***Electrical & Embedded Systems Engineer*** *- This role was assigned to project team members with a strong background in electrical engineering, as this role involved tasks that require knowledge of transistors, diodes, and power management, along with the ability to design embedded software.*
    - Role assignee(s): Nikki Marrow

- ***Hardware Engineer (Mechanical)*** *- This was a relatively small role designated for those who worked on the construction and design of any non-electrical mechanical components of the product, such as the birdhouse for the smart bird feeder device. Because our group consists of only three Computer Engineers and one Electrical Engineer, and the* *mechanical engineering discipline falls outside the scope of this course, the tasks related to this role were very limited for this project.*

  - Role assignee(s): Paul Amoruso, John Hauff, Nikki Marrow, Matthew Wilkinson

As mentioned previously, in order to complete this project in an orderly, efficient, and timely manner, our team chose to split the tasks of this project between our members in such a way that each team member had a roughly equal amount of work to contribute to the overall product as any other member.

- ***Project Documentation and Presentation*** *- Project documentation involves documenting all of the technologies considered and used in the design of the smart bird feeder product created for this project, updating papers, and the creation of the conference paper. Presentation work includes making slideshows, video editing, and the showcase presentation.*
  - Task assignee(s): Matthew Wilkinson spearheaded the creation and updates of any presentation/documentation work. Project documentation tasks were also assisted by every member of the team.

- ***Machine Learning (ML) Model Training and Deployment*** *- The members who hold the role(s) assigned to this task are responsible for completing various subtasks related to retraining and deploying a machine learning (ML) model. Such subtasks will consist of deciding which pretrained ML model to use to retrain for object detection of bird species and squirrels, as well as locating or creating a dataset suitable to feed to this ML model for retraining. The member(s) with role(s) assigned to this task will be expected to properly retrain, validate, and test the ML model that they choose to use for our smart bird feeder device. This task heavily involves development using the machine learning development kit selected for use in the smart bird feeder device, so experience with developing on a Linux platform is also necessary to complete this task. The members who hold the role(s) assigned to this task are also responsible for documenting and reporting their progress and design ideas to the rest of the project team for inclusion in the final EEL4914 and EEL4915 papers.*
  - Task assignee(s): Because this task heavily involves programming and using a Command Line Interface (CLI), this task was handled by any team member holding the Software Engineer role. However, John Hauff and Paul Amoruso will be the team members who will primarily contribute to this task, while Matthew Wilkinson serves more as an advisor with respect to this task.

- ***Birdhouse Design and Construction*** *- This task consists of designing and constructing the birdhouse that will house the electronic parts necessary for the smart bird feeder product. This involves locating suitable materials for the birdhouse (weather-proof, suitable durability, etc.), as well*

as carrying out the chosen method of construction of the birdhouse materials and structure itself (3D printing, laser cutting, etc.). The member(s) who hold the role(s) assigned to this task are also responsible for documenting and reporting their progress and design ideas to the rest of the project team for inclusion in the final EEL4914 and EEL4915 papers.
    - Task assignee(s): This task was handled by any team member assigned to the Hardware Engineer (Mechanical) role.

- ***Smart Bird Feeder Application Development*** *- This task consists of planning and executing the development of the user application that will pair with the smart bird feeder device. A team member assigned to this role must assist in choosing the stack that the application will be built upon (LAMP, MERN, MEAN, etc.), as well as have some experience with full stack (frontend and backend) development. The member(s) who hold the role(s) assigned to this task are also responsible for documenting and reporting their progress and design ideas to the rest of the project team for inclusion in the final EEL4914 and EEL4915 papers.*
    - Task assignee(s): This task was handled by any project team member assigned to the Software Engineer role.

- ***PCB Design*** *- Again, the member(s) who hold the role(s) assigned to this task are also responsible for documenting and reporting their progress and design ideas to the rest of the project team for inclusion in the final EEL4914 and EEL4915 papers.*
    - Task assignee(s): This task was handled by any project team member assigned to the Hardware Engineer (Electrical) role.

7.4 Project Management Tools

Throughout the duration of the smart bird feeder device's development, our team used multiple platforms and applications to effectively communicate with one another and organize ideas and tasks coherently. This section discusses the technologies used to keep our project well-organized and on track to meet deadlines and maximize our chances of success.

7.4.1 Discord

In deciding the best method of communication for our group, we considered a variety of messaging apps that are compatible with various devices and allow for organized discussions. In our search, we stumbled across Discord, as we were familiar with its service prior to Senior Design. Discord allows for users to create so-called “servers”, where we can create topic-based channels to talk, collaborate, and share about the specific topic without clogging up any group chat. While some may say this isolation between topics can lead to disconnect between members, this application was still on the top of the list for utilization of everyone in the group. Therefore, we decided to utilize Discord as our primary means of communicating with one another, for a variety of other reasons as well besides the versatility, and topic segregation. Discord allows for media upload upload, editing of text after being sent, pin messages (to emphasize importance and to keep important text from getting lost), bots that interact with text, and easy to use voice channels for meeting and discussions. Discord also allows you to give members of the server special permissions, so that they can set up and moderate channels on the server. However, a more useful aspect of Discord is its well marketed reliable software technology, which provides relatively low-latency voice and videos at any time of day. This ensures that our team can meet and clearly communicate by voice, text, or video whenever we would like to do so.

After choosing Discord to utilize for our primary means of communication for the various features this advertises and our familiarity with the application, our experience each differed slightly, but for the most part each group member had the same feeling towards the effectiveness of using it. In our discord server, John Hauff as the acting project manager, decided to create a multitude of channels for topics such as a general project discussion, hardware discussion, software discussion, bird feeder design, scheduled meetings, and of course an off-topic channel to mention a few. A common issue we each of noticed in the beginning of our discord utilization, was that the general project discussion channel was overflowing with messages, and started getting clogged up to a point that if someone in the group were to try to catch up all that was said by three other members would be a hard task to scroll through all the messages to read and keep track of them. Thus, John decided to amend the situation by utilizing Discord's ability to have various channels so that we can break each of the possible topics of discussion. After the creation of all the channels, productivity and efficiency of communicating through Discord soard. The only issues that arrived in the utilization of Discord as the primary means of communicating, is that notifications from the application were sometimes not very effective as Paul Amoruso reported to the group that once in a while not all the messages typed in Discord were being notified on his phone, causing Paul to have to check the mobile application occasionally. Moreover, Matthew Wilkinson also mentioned a couple times that he had trouble keeping up with all the messages in the various channels during times when a larger number of messages were being sent in multiple channels by the other group members.

Looking back at our decision to choose Discord as our primary means of communication, it is unanimous throughout the group that our decision was the best for us compared to other messaging applications. The reason for saying this, is due to the fact that after understanding the undesired side effects of using these applications, such as missing notifications or too many channels, our group members were able to work around them to get the most out of Discord key aspects that make it unique from so many other applications. For instance, while too many channels may be confusing, we narrowed it down to the most important channels topics and by using the right amount, we were able to keep the channels clogged free and still communicate without losing our messages. Then probably one of the most important features provided by Discord, and highly used by John and Paul to communicate, was the voice channel to discuss various topics about the project and the ability to share each other's screens when needed for such things as object detection training and utilization. Overall, we can honestly say that Discord as our primary means of communication was essential to our workflow.

7.4.2 Clickup

Clickup is a project management tool that can be used to coordinate collaboration amongst team members in a streamlined manner by organizing goals into specific tasks. Clickup provides the ability to propose tasks to a group for approval, assign specific tasks to members, subcategorize large tasks into smaller components, assign members to proposed tasks, and create deadlines to promote efficiency. Before using Clickup, the team was reliant on meeting notes via discord to document what needed to be done. This process can be time consuming to sort through, and ineffective in situations where less information was jotted down than needed to be. The free form nature of remembering what was discussed during meetings opens up many inefficiencies, and by supplementing Clickup into our workflow, many of these inefficiencies were further uncovered and dealt with.

One of the ways Clickup increased the productivity of our workflow was by breaking down large tasks into smaller more manageable subtasks. By discussing large tasks in the weekly meetings, and then attempting to assign a large subtask to a single person, the specifics of the task become lost to the other group members. Instead, by breaking down large tasks into documented subtasks, we can have a better understanding of what needs to be done to accomplish a particular goal. We can also better assign tasks to individuals who suit particular strengths, as some subtasks from a larger goal might be better suited to a particular person. The last way this assists the team is by creating a clearer timeline on how long a goal should take to complete. Clickup provided an excellent system to utilize this concept to its fullest.

Another way Clickup increased our team productivity was the ability to assign deadlines to tasks. Before Clickup was implemented, the team would use the weekly meetings to estimate how much could be done by the next team meeting. However, by not assigning specific timelines, less was ultimately achieved. The ease of assigning a proposed timeline to each idea and subidea created a flowing and efficient system, which something like word of mouth or a Gantt Chart could not achieve.

By shifting our task management system from discord notes to a more comprehensive program, we gained a significant amount of efficiency. These efficiency gains listed above not only affect our 120 page document, but shall continue to affect our project build going forward.

7.4.3 Text Messaging and Calls Via Cellular Device

Communication by text messaging or calling via a cellular device was used as a backup method of communication in the event that a member (or members) of the team could not properly communicate and/or interact with another team member in an effective and timely manner using other established tools such as Discord or Clickup, which were described previously. For instance, on rare occasions Discord may not adequately notify individuals of a group of every message sent in the mobile application, therefore, if a particular person was not receiving an important message via notification from the Discord application, then we would contact that particular person via their mobile number by text or call. While Discord and Clickup were highly effective at keeping the group well communicated, there were still situations where text messaging and calls via cellular device was necessary. For example, during one of the group meetings we planned in the middle of senior design one, a group member of ours nearly missed a meeting because they did not get a notification from Discord, where it was announced. Luckily in this situation, our group used their mobile number to send text messages and call via their cellular device to get in contact with the missing member in a relatively quick manner. Moreover, we also occasionally texted through our cellular devices passively when contacting individuals of the group privately in a quick manner. Most text conversations happened openly with all the group members in the group chat on Discord, but occasionally when having private conversations we were free to use the easiest method of contact for that occasion.

With all this said, text messages and calls via our cellular devices were used as an extra means of communication even though Discord was our primary means of sending messages and calling. Similar to what was previously mentioned, this means of communicating with group members was utilized only as a way of individually reaching out to a particular member without going through the Discord application. However, most private conversations happened through the Discord application. All in all, texting and calling via our cellular devices as a form of backup plan when we feel that it is easier or necessary to reach out to a particular member of the group for a variety of reasons.

7.4.4 GitHub

During the managing of collaboration between the various codes for the various parts of the overall development of the Smart Bird Feeder project, our group will and is currently using the well known web based version control collaboration platform known as GitHub. Having each code demanding aspect of the project having a repository on GitHub allows for members of the group to collaborate with one another by being able to download the newest version of the particular code and update it, then post it back for everyone to see and implement. For instance, if Matthew Wilkinson is working on the mobile application and needs help from another member who is able to help, then another group member like Paul Amoruso can download the mobile application code and add or tweak it. With the various aspects of our project that rely on code, such as the mobile application, and the code that runs the object detection and logic on the development board, it is important that we utilize such platforms as GitHub for reliability and potential efficiency. The reliability of GitHub comes from the fact that the code will be housed on the cloud and will have cloned versions saved on our local computers to decrease the risk of losing any code. And finally, this platform will potentially help with efficiency as we are not dependent on sending code to one another through unorganized messaging applications, but rather have a centralized platform where the newest working version of the code will be on for each group member to be able to clone on their local computer at any time or place with internet connection.

8. Conclusion

After handling numerous obstacles and challenges in our path, our team is confident that we have both fulfilled the requirements and expectations of the Senior Design I course, planned the designs for our smart bird feeder device, and properly investigated the technologies that will be an essential part of our smart bird feeder. Our team has taken measures throughout this project's lifetime to ensure that the project workload is split as evenly as possible among the ECE engineering disciplines, paying careful attention to not give a specific member or group of members an unfair amount of work. We have met every deadline set by the Senior Design I and Senior Design II courses and our own team so far, our report meets the final report requirements properly, and we are confident that our project contains substantial design, due to our inclusion of a custom PCB design and its integration with the NVIDIA Jetson nano Developer Kit in this project.

In designing and combining various sensors and electrical components to create our own original smart bird feeder, we have developed and produced an innovative smart bird feeder that takes advantage of computer vision and machine learning to distinguish between birds and squirrels visiting the feeder, and deters squirrels from loitering in the immediate vicinity of The Smart Bird Feeder. The feeder's squirrel deterrent system promotes the arrival of more feathery friends. We also took advantage of The Smart Bird Feeder's camera for other uses, such as capturing memories of birds' visits and storing the memories to be viewed in a user application. The application has given the team the opportunity to connect any and all features of the device directly to the user in a manner that maximizes the device potential. Our application fits well in a hobbyist's toolkit when accounting for a lack of time, as many of our features implemented ensure that the bird feeder provides complete functionality without having to constantly watch the device.

In regards to the creation and testing of the device, our team feels our research and development has put us in a prime position for success. The in-depth plans for testing each component of the device with strict parameters ensures that each component is likely to work as intended when combined to create the full product. Upon beginning development in Senior Design II, our team was confident that the testing structure we’ve established will result in a smooth testing and development.

With all of this being said, our team is also excited by our smart bird feeder device's high potential for scalability, especially with regard to its user application and machine learning capabilities. In the case of the user application, there is plenty of potential for additional features if more development time is given. With regard to the machine learning object detection model, this model can be scaled virtually indefinitely, namely because the object detection model has the potential to be trained to detect even more classes of bird species than our team will have it capable of detecting, and the object detection model may also be extended to recognize predators of bird feed outside of squirrels alone. The amount of species that can be detected is beginning with Florida backyard birds, but can easily be expanded to fit other markets if the test run goes well. Another interesting idea for extending this project would be to purchase various bird feeds that different species of birds prefer, then have the machine learning model detect the species of bird visiting the bird feeder, and have the feeder dispense the bird feed that matches that species of bird's preferences. As stated previously, though our team is confident that our device meets the requirements for the Senior Design course, the possibilities for extending this project are virtually endless.

The main goal that our team had in mind was to bring to fruition an easy-to-use bird feeder that is reliable, and reasonably priced to compete in the marketplace. In our preliminary design for the feeder, we aimed to house a series of electrical components with the ability to process the frames taken from the bird feeder, and identify the existence of birds or squirrels in these frames. With this capability, our device will be able to determine whether or not the servo motor must be utilized to close the hatch, in addition to capturing photos of the birds, storing them in memory, and sharing them over WiFi to our app as we decided that a wireless transmission of information from the feeder is the most logical in order to help prevent hassle of using a wire. In addition, there is a motion sensor responsible for ensuring the machine learning software is only active when there is motion in front of the feeder. Finally, in addition to closing the hatch on the feeder, we also added a speaker that plays a sound to discourage the squirrels from hanging around the feeder. All of these features help to promote our original design goals, and will allow any bird enthusiast the features they need to enjoy the hobby without constant supervision. In conclusion, by combining a variety of hardware and software features to create an enjoyable user experience, our original bird feeder design offers a great opportunity to make our smart bird feeder device stand out from the rest of the bird feeders on the market today.

Appendix

A1 References

[1] *About IPC*. IPC International, Inc. (2020, November 19). https://www.ipc.org/about-ipc.

[2] *1013-2019 - IEEE Recommended Practice for Sizing Lead-Acid Batteries for Stand-Alone Photovoltaic (PV) Systems*. IEEE Xplore. (n.d.). https://ieeexplore.ieee.org/document/8845030?denied=.

[3] "IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," in IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016) , vol., no., pp.1-4379, 26 Feb. 2021, doi: 10.1109/IEEESTD.2021.9363693.

[4] Index of /jtc1/sc22/wg14/www/docs. (n.d.). http://www.open-std.org/jtc1/sc22/wg14/www/docs/.

[5] NXP Semiconductors. (n.d.). i.MX 8M Dual / 8M QuadLite / 8M Quad Applications Processors Data Sheet for Consumer Products.

[6] *Jetson Nano 2GB Developer Kit*. NVIDIA Developer. (2020, October 29). https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit.

[7] Google LLC. (n.d.). Coral Dev Board datasheet. Coral. https://coral.ai/docs/dev-board/datasheet.

[8] NVIDIA Corporation, "DATA SHEET NVIDIA Jetson Nano System-on-Module" | DA-09366-001_v1.0 datasheet, Feb. 2020

[9] Bird Buddy. (n.d.). Retrieved July 04, 2021, from https://mybirdbuddy.com/

[10] Arduino nano 33 Ble sense. (n.d.). Retrieved July 05, 2021, from https://store.arduino.cc/usa/nano-33-ble-sense

[11] Klopfenstein, T., & Le, M. (n.d.). Smart Bird Feeder. Retrieved July 07, 2021, from https://coral.ai/projects/bird-feeder/#project-intro

[12] Edge TPU performance benchmarks. Coral. (n.d.). https://coral.ai/docs/edgetpu/benchmarks/.

[13] Jetpack 4.5 Archive. NVIDIA Developer. (2021, January 22). https://developer.nvidia.com/jetpack-sdk-45-archive.

[14] JetPack SDK. NVIDIA Developer. (2021, May 24). https://developer.nvidia.com/embedded/jetpack.

[15] Jetson Nano: Deep Learning Inference Benchmarks. NVIDIA Developer. (2021, January 5). https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks.

[16] *Lithium Battery Safety - EHS*. (n.d.). https://www.ehs.washington.edu/system/files/resources/lithium-battery-safety.pdf.

[17] PyCoral API overview. Coral. (n.d.). https://coral.ai/docs/reference/py/.

[18] Real-Time Object Detection in 10 Lines of Python on Jetson Nano. NVIDIA Developer Blog. (2020, January 31). https://developer.nvidia.com/blog/realtime-object-detection-in-10-lines-of-python-on-jetson-nano/.

[19] Jetson AI Courses and Certification: Jetson AI Fundamentals Course. NVIDIA Developer. (2021, July 6). https://developer.nvidia.com/embedded/learn/jetson-ai-certification-programs#course_outline.

[20] Dusty-Nv. (n.d.). dusty-nv/jetson-inference: Hello AI World guide to deploying deep-learning inference networks and deep vision primitives with TensorRT and NVIDIA Jetson. GitHub. https://github.com/dusty-nv/jetson-inference.

[21] Wah C., Branson S., Welinder P., Perona P., Belongie S. "The Caltech-UCSD Birds-200-2011 Dataset." Computation & Neural Systems Technical Report, CNS-TR-2011-001.

[22] Kaggle. (2021, July 4). 275 bird species also SEE 73 Sports Dataset. Kaggle. https://www.kaggle.com/gpiosenka/100-bird-species.

[23] Gillson, G. (2019, July 17). 26+ common backyard birds of Florida (Photos, ID). 26+ Common backyard birds of Florida (Photos, ID). https://www.whatbirdsareinmybackyard.com/2019/07/most-common-backyard-birds-of-florida.html.

[24] NVIDIA. (2021, July 21). *Jetson download Center*. NVIDIA Developer. https://developer.nvidia.com/embedded/downloads.

[25] SALLEH, A. S. (n.d.). IR proximity sensor working principle [Illustration]. https://www.researchgate.net/figure/IR-proximity-sensor-working-principle_fig2_301787143

[26] HC-SR04 Pin Details. (n.d.). [Illustration]. https://microcontrollerslab.com/hc-sr04-ultrasonic-sensor-interfacing-with-arduino-distance-measurement-example/

[27] PiezoBendingPrinciple.svg. (2011, February 17). [Illustration]. https://commons.wikimedia.org/wiki/File:PiezoBendingPrinciple.svg

[28] TI Launchpad Kit. (n.d.). [Illustration]. https://www.amazon.com/Texas-Instruments-Value-MSP430-LaunchPad/dp/B07J3R15JB

[29] Texas Instruments. (2011, April). SLAS735J MIXED SIGNAL MICROCONTROLLER. Dallas. https://www.ti.com/lit/ds/symlink/msp430g2553.pdf?ts=1636685136752&amp;ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430G2553

[30] Sparkfun. (n.d.). DC Barrel Power Jack/Connector [Photograph]. https://www.sparkfun.com/products/119

[31] Show Me Cables. (n.d.). 2.1mm DC Power Male Plug to Open End Cable [Photograph]. https://www.showmecables.com/2-1mm-dc-power-plug-to-open-end-cable-

22-awg-6-ft?gclid=CjwKCAjw0qOIBhBhEiwAyvVcfx6enYWApYD0tr35dKoV
mRBFyaB-OlcO0aM7DHVSwElQyTcFgf7FHxoCc1sQAvD_BwE