The Pecker Peeper

EE459 Project Report

Spring 2024

Team 19

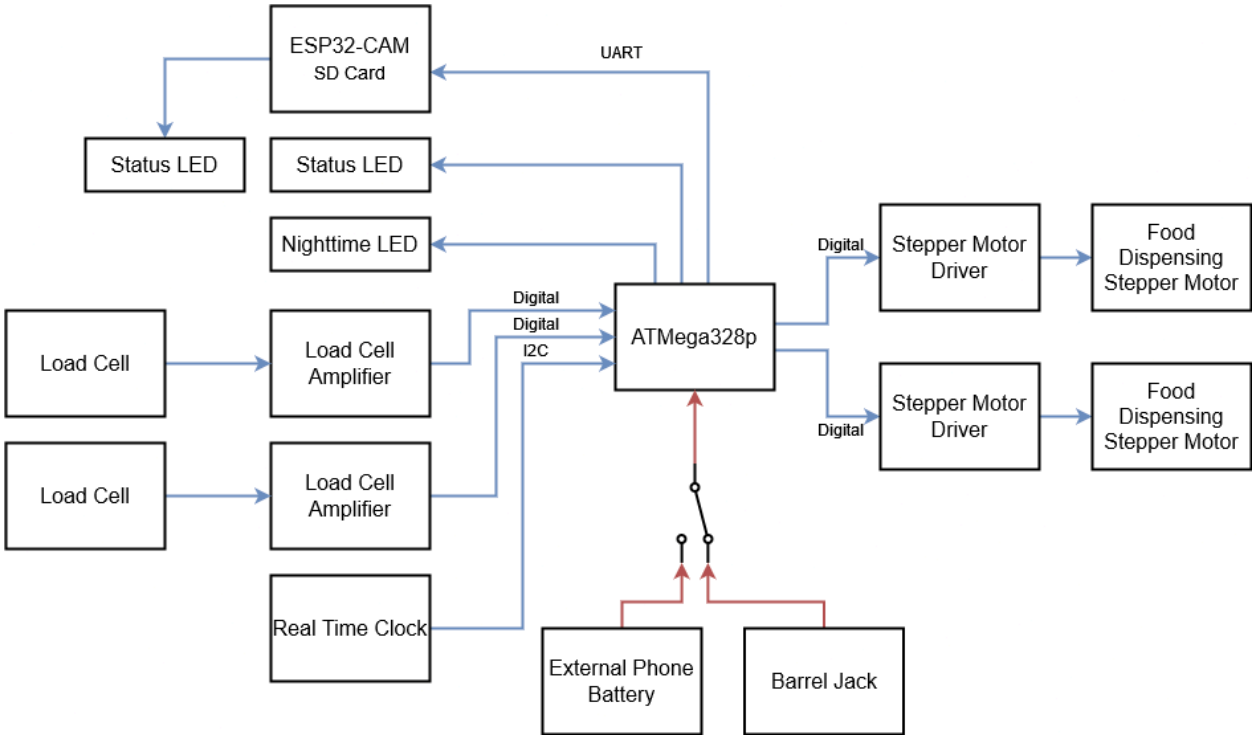
Brandon Gong, Isaiah Lee, Chur Tam

I. Introduction

One of the oldest hobbies, bird watching is a pastime that caters to every age group. Its accessibility and affordability ensure that it will continue to stand the test of time. However, as popularity for bird watching grows, the technology and resources need to as well.

Much of bird watching entails hours of waiting, sitting still with hours of patience. Our group proposes a solution to this problem, an automatic bird feeder and camera. It is a bird feeder with a camera attachment that will capture images of the birds that visit the feeder, saving these images to an SD card that can be processed by the user. The user thus can record hours of bird appearances without needing to invest a lot of time in waiting. This should be useful to anybody looking to identify local wildlife, whether it be a neighborhood hobbyist or a nonprofit trying to find endangered species.

II. Overview



System-level Block Diagram

Above is a simple system-level diagram of our project. An ATMega328p is the primary controller, reading the sensors and directing actuators and other functions. An ESP32-CAM provides picture-taking and uSD card functionality. Load cells are used to detect the presence of a bird, a real time clock (RTC) module is used to schedule various operations, and stepper motors are used to automatically dispense bird seed. The device can be powered directly with a 5V power supply, through USB-C (for example, an external phone charger battery), or through a barrel jack connector (for example, a wall wart).

III. User Interface

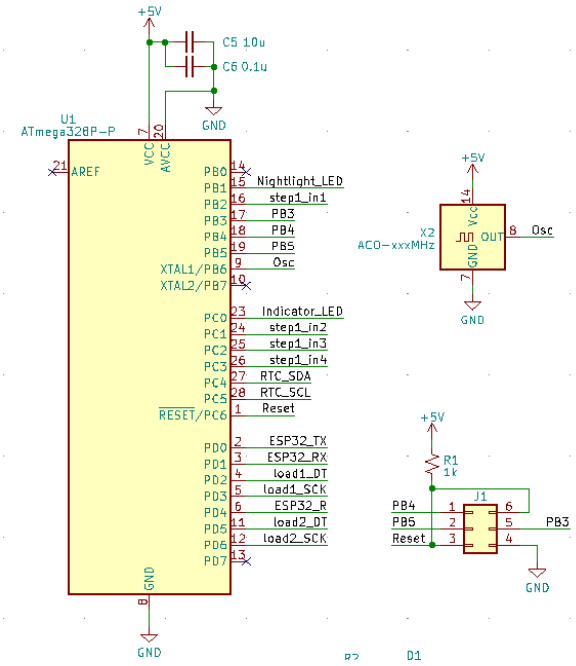
The primary function of the bird feeder, taking pictures, is activated by the load cell detectiong a relatively large weight on the perch. Aside from that, there is no other interface during regular operation.

For setting up the bird feeder, there are values that must be set: weight threshold for taking a picture, bird feed refill times, and sunset/sunrise times. These values are hard coded into the firmware so these values must be manually changed in the code and the firmware has to be reflashed.

IV. Electrical System

A. Primary Microcontroller: ATMega328p

The ATMega328p is the central controller of our system. It either directly reads or indirectly activates our various sensors and controls the stepper motors.



ATMega328p Pinout

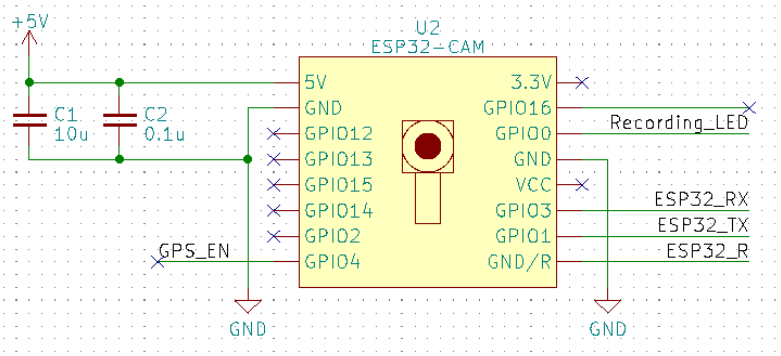
The code constantly polls the load cell and the RTC module. When a significant (non-calibrated) change is detected in the load cell reading, the microcontroller writes the RTC timestamp data to a UART message sent to the ESP32. Additionally, when the time matches preset feeding times, the microcontroller will actuate the stepper motors to dispense bird feed. When the time matches a preset night time, the microcontroller will turn a night light led to illuminate the feeding area.

B. Secondary Microcontroller/Camera/SD Card: ESP32-CAM

The ESP32-CAM is a development board that integrates an ESP32 with a OV2640 camera and a uSD card reader. This secondary microcontroller and camera combo can be commanded over UART by the ATMega328P to take a picture. The UART message contains 6 bytes of timestamp

data which is used to name and save the picture to the uSD card. At any point in time, the uSD card can be removed from the ESP32-CAM to retrieve the pictures.

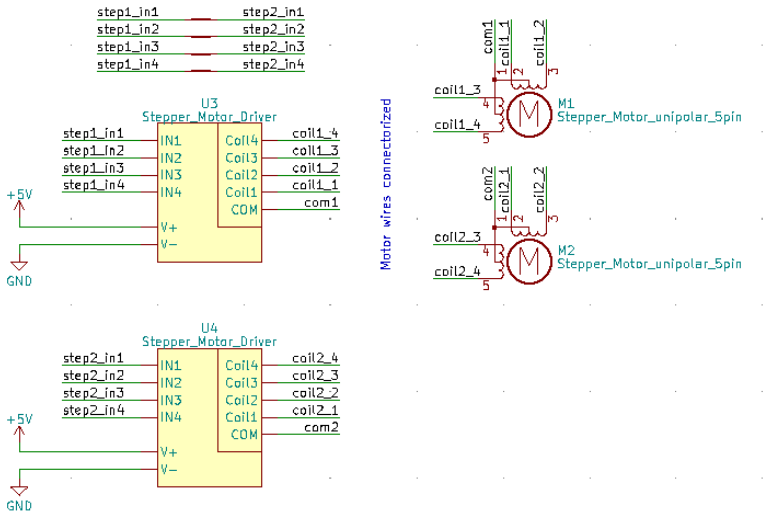
We chose to use the ESP32-CAM because of its diversity of functions. In additioned to the previously mentioned features, it also has wireless capabilities. Because it integrates so much in one small development board, it was an easy choice. One major disadvantage of the board is the lack of a USB connection. As a result, an external TTL serial to USB converter must be used to program the board. This comes in the form of a shield that covers up all the other pins of the ESP32-CAM and makes developing and debugging extremely inconvenient.



ESP32-CAM Pinout

The ESP32 is programmed using the Arduino IDE because most of the actively supported libraries are for the Arduino IDE. After initializing the camera, uSD card, and serial port, the ESP32-CAM waits for a UART message from the ATMega328p. Upon receiving a message, the timestamp data is extracted, a picture is taken, and the picture is saved with the timestamp data in the name of the file in the uSD card.

C. Stepper Motor

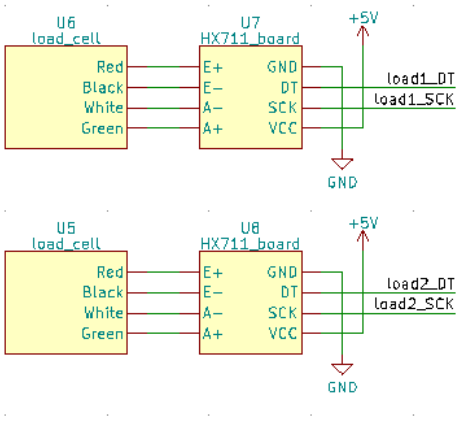


Stepper Motor Wiring

The motors are 28BYJ-48 unipolar stepper motors. They contain two coils with five taps and a breakout board featuring the ULN2003 stepper motor driver drives these connections. The driver itself takes in 4 inputs that can be controlled by the ATMega328p to spin the motor step-by-step in a given direction at a given speed.

Our custom library for these motors allows the user to encode a table of signals that a function can take a distinct number of steps through. This streamlines development and debugging as there are multiple ways of signaling the motors.

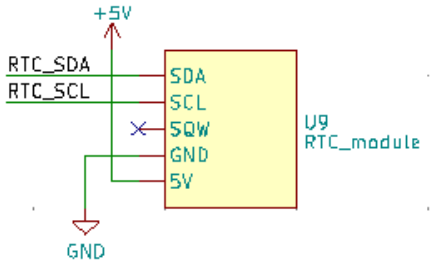
D. Load Cells



Load Cell Wiring

The load cells contain wheatstone bridge circuits used to detect force applied to the cell. We use it to detect when a bird lands on the perch of the feeder. The signals of the load cells are connected to breakout boards featuring the HX711 amplifier. The amplifier outputs the reading in a two-wire digital interface. The clock line is controlled by the ATMega328p and on every rising edge, a new data bit is written to the data line (which is subsequently read by the ATMega328p).

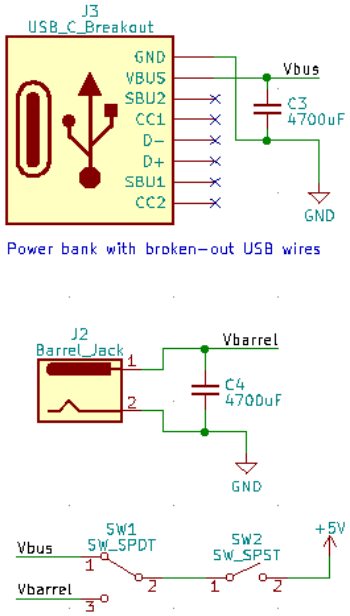
E. Real Time Clock



RTC Module Wiring

Our RTC was a Sparkfun module incorporating the RTC DS1307 chip with a CR1225 coin cell. The additional battery allows the RTC module to keep track of time even if the rest of the system is powered off. The module communicates the current date and time via I2C to the ATMega328p.

F. Power

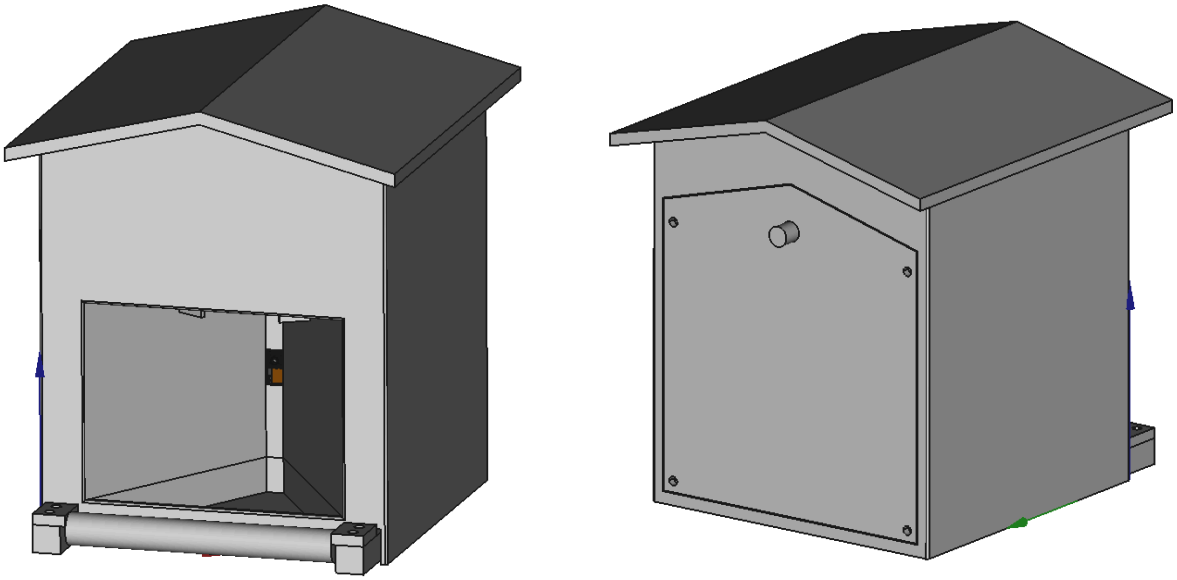


Power Wiring

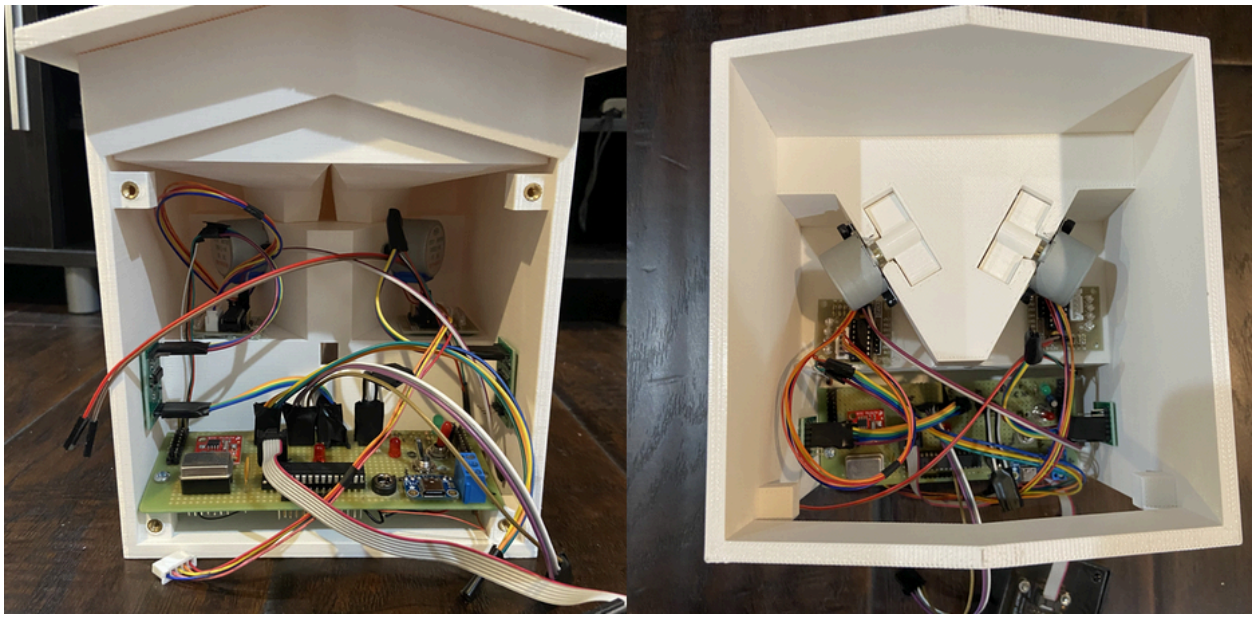
In addition to being directly powered by a 5V power supply, the bird feeder also supports two more practical power sources. The first is a power-only USB-C connector, which can be used either by a 5V USB-C power supply or an external phone battery. If the bird feeder is deployed away from a wall outlet, this offers a simple, easily rechargeable, and easily swappable mobile power source. The second is a traditional barrel jack, allowing the bird feeder to be powered by common 5V power supplies and wall warts. An SPDT switch allows the user to select which power source they want to use and an SPST switch allows the user to shut off power entirely.

Allowing the user to use a premade phone battery also ensures safety as these batteries are consumer products and already come with built-in battery protection.

V. Physical Construction



3D Assembly Render Front and Back

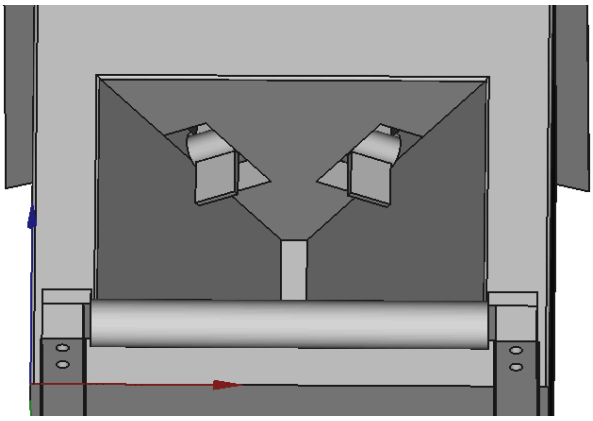


Electrical Wiring within the Bird Feeder

The physical model of the bird feeder was designed and assembled in FreeCAD and all parts were printed in white ABS using the Baum Family Makerspace's Stratasys F370 FDM 3D printer. Notable parts include the main body, the perch, the feed dispensing rotors, the feed funnel, and the roof.

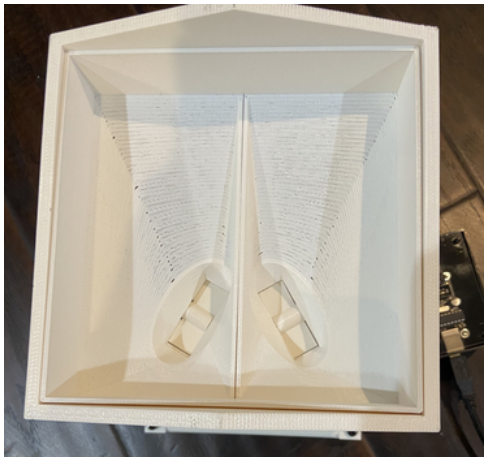
The main body has a bird feed bowl in the front with a hole for the camera. The bowl is roughly triangular to restrict the feeding area to a 60 degree area in front of the camera so nothing is missed. In the back, covered by the removable back panel in the image above, is empty room for the feed funnel and the electronics.

The perch is a rod placed in front of the feeding area. It is supported on either side by two load cells. It is intended for birds to land on the perch to be detected by the load cells.



3D Assembly Render Food Dispenser

On the ceiling above the feeding area are two slots with rotors controlled by stepper motors. The rotors can be rotated to allow a limited amount of stored bird feed to fall into the feeding tray.



Food Storage/Funnel (Top-Down View w/o Roof)

Extra bird feed can be stored in the empty area above the feeding tray. A removable 3D printed funnel allows the bird feed to cleanly fall into the food dispensing slots. The roof can be placed on top of the house to cover the extra feed and protect the feeder from weather.

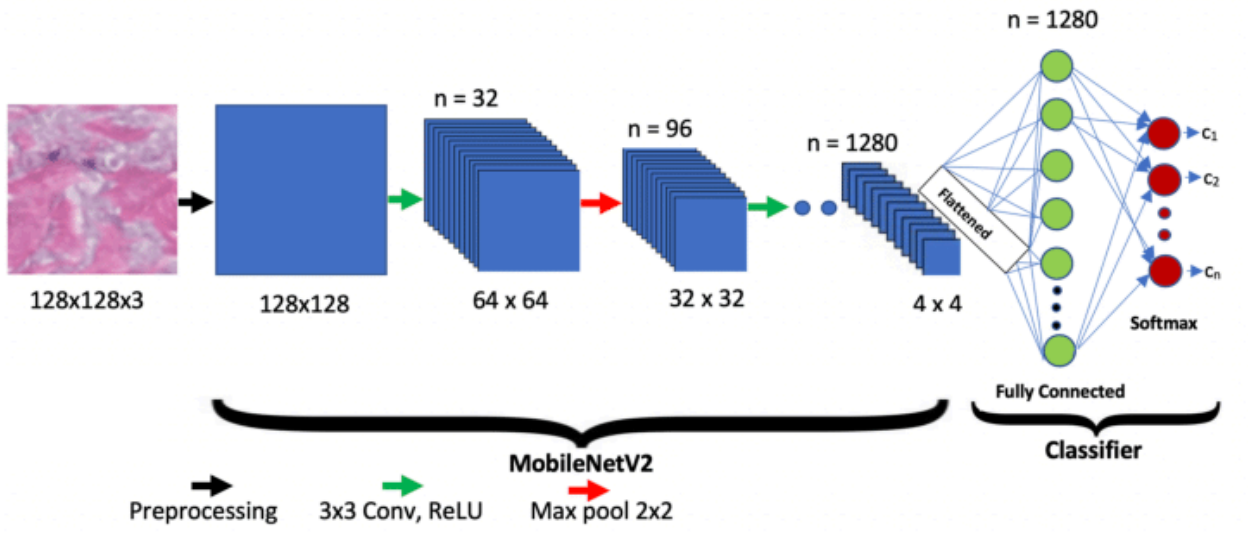
An additional note about the price of the 3D printed parts ($295.52). The price was unusually high because the Stratasys requires proprietary filament rolls. Additionally, the makerspace staff used relatively high infill because it is inconvenient to create a design with less infill using the Stratasys proprietary slicer.

VI. Machine Learning

While not implemented into the final design, the machine learning portion of the project was an avenue that was heavily looked into and we believed it important to include. The original intent with the machine learning portion of the project was to be able to identify the birds after they had their picture taken via the ESP32 and stored into the SD card. By using a convolutional neural network, the idea was to be able to trigger a “classification mode” by pressing a button that would retroactively go through the SD card and label all the images by their perceived species name. An important feature of this neural network is that it was planned to be run on the ESP32, rather than sent to an outward server, computer, or other processor. With the ESP32 SoC including 4MB flash memory and the development board including 4MB external PSRAM, we believed that it would be possible to fit a light model onto the board to run inference on our images.

The first step into being able to classify images was to find available datasets online. It was important that the model be at least somewhat accurate, and in order to ensure that we found multiple large bird image datasets online. A dataset of 11,000 images and 200 bird species was found on the public database Kaggle, which we used for the training of our models.

Next would be the actual model used to classify the birds, which had to be both lightweight and fairly accurate. To make the process more efficient, we used a model that was pre-trained and added a few layers to the neural network to make the classification more suited to our particular need for bird classification. After testing numerous different models, the MobileNet classification model was chosen to be the base of the program for being most efficient in terms of memory size to accuracy.

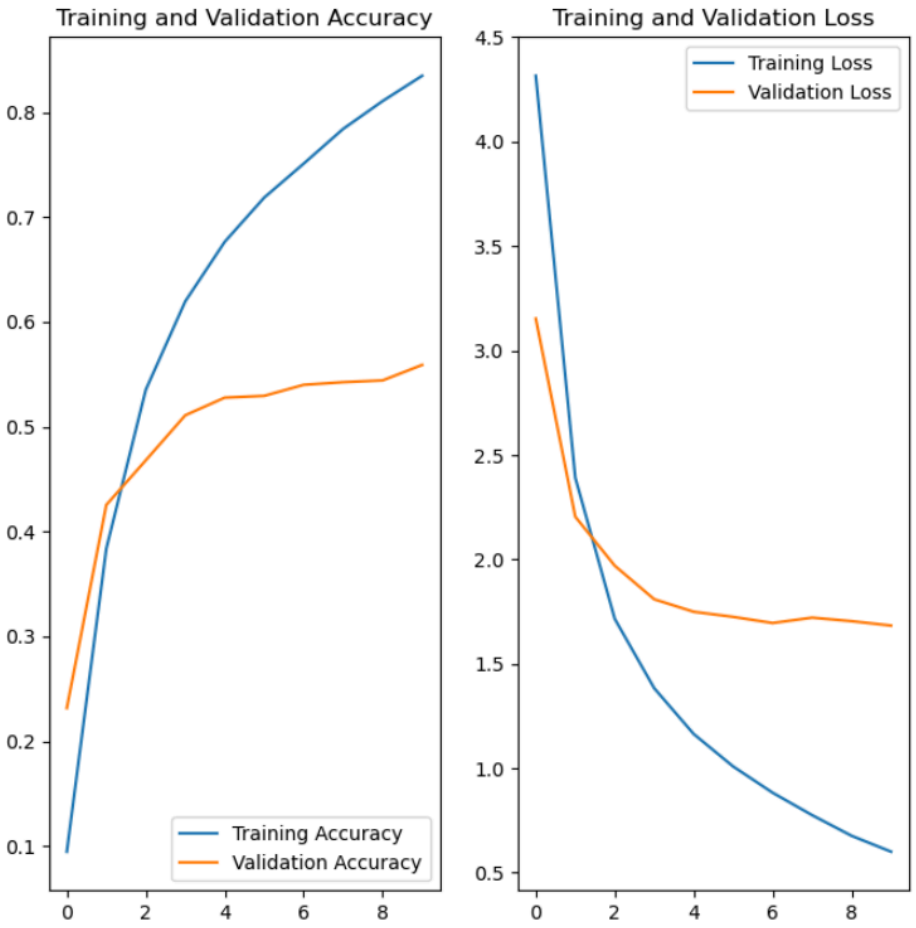


Visualization of MobileNetV2 Neural Network

After applying various data preprocessing techniques on the training dataset, and training for around 10 epochs, we were able to produce a model with around 56% accuracy in identifying the correct species.

```
Epoch 1/10
295/295 [==============================] - 97s 310ms/step - loss: 4.3154 - accuracy: 0.0946 - val_loss: 3.1528 - val_accurac
y: 0.2317
Epoch 2/10
295/295 [==============================] - 78s 265ms/step - loss: 2.3914 - accuracy: 0.3837 - val_loss: 2.2045 - val_accurac
y: 0.4254
Epoch 3/10
295/295 [==============================] - 81s 274ms/step - loss: 1.7156 - accuracy: 0.5353 - val_loss: 1.9698 - val_accurac
y: 0.4680
Epoch 4/10
295/295 [==============================] - 78s 264ms/step - loss: 1.3833 - accuracy: 0.6196 - val_loss: 1.8092 - val_accurac
y: 0.5110
Epoch 5/10
295/295 [==============================] - 77s 262ms/step - loss: 1.1641 - accuracy: 0.6764 - val_loss: 1.7493 - val_accurac
y: 0.5278
Epoch 6/10
295/295 [==============================] - 79s 268ms/step - loss: 1.0086 - accuracy: 0.7186 - val_loss: 1.7249 - val_accurac
y: 0.5295
Epoch 7/10
295/295 [==============================] - 79s 267ms/step - loss: 0.8830 - accuracy: 0.7510 - val_loss: 1.6955 - val_accurac
y: 0.5400
Epoch 8/10
295/295 [==============================] - 77s 263ms/step - loss: 0.7756 - accuracy: 0.7842 - val_loss: 1.7212 - val_accurac
y: 0.5425
Epoch 9/10
295/295 [==============================] - 78s 265ms/step - loss: 0.6768 - accuracy: 0.8107 - val_loss: 1.7043 - val_accurac
y: 0.5442
Epoch 10/10
295/295 [==============================] - 78s 263ms/step - loss: 0.6007 - accuracy: 0.8348 - val_loss: 1.6832 - val_accurac
y: 0.5590
```

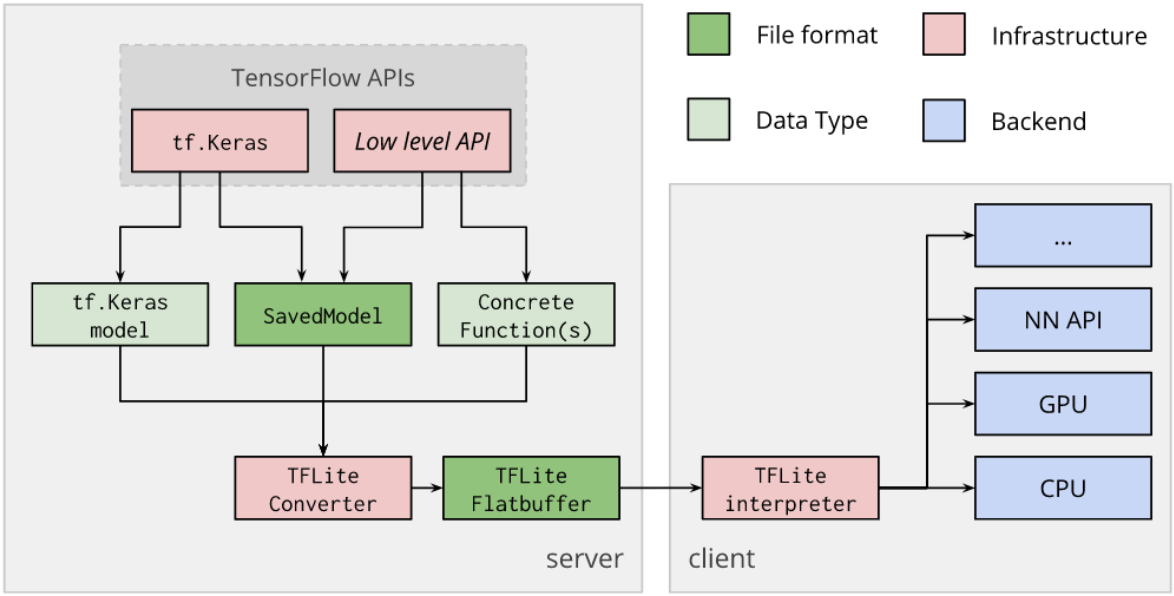
Model Training Epochs and Accuracy



Training Learning Curves for Validation and Training Dataset Accuracy

While not seemingly high, it only considered a perfect match a success, meaning that subspecies of a bird such as Grey Heron being classified as a Black Heron would be considered incorrect, even if they were technically the same species.

Once our model was trained and ready, we had to be able to run inference on the microcontroller. Our main plan revolved around using the Tensorflow Lite functionality, a utility that compresses Tensorflow models to hopefully be able to fit on the onboard memory of the ESP-32.



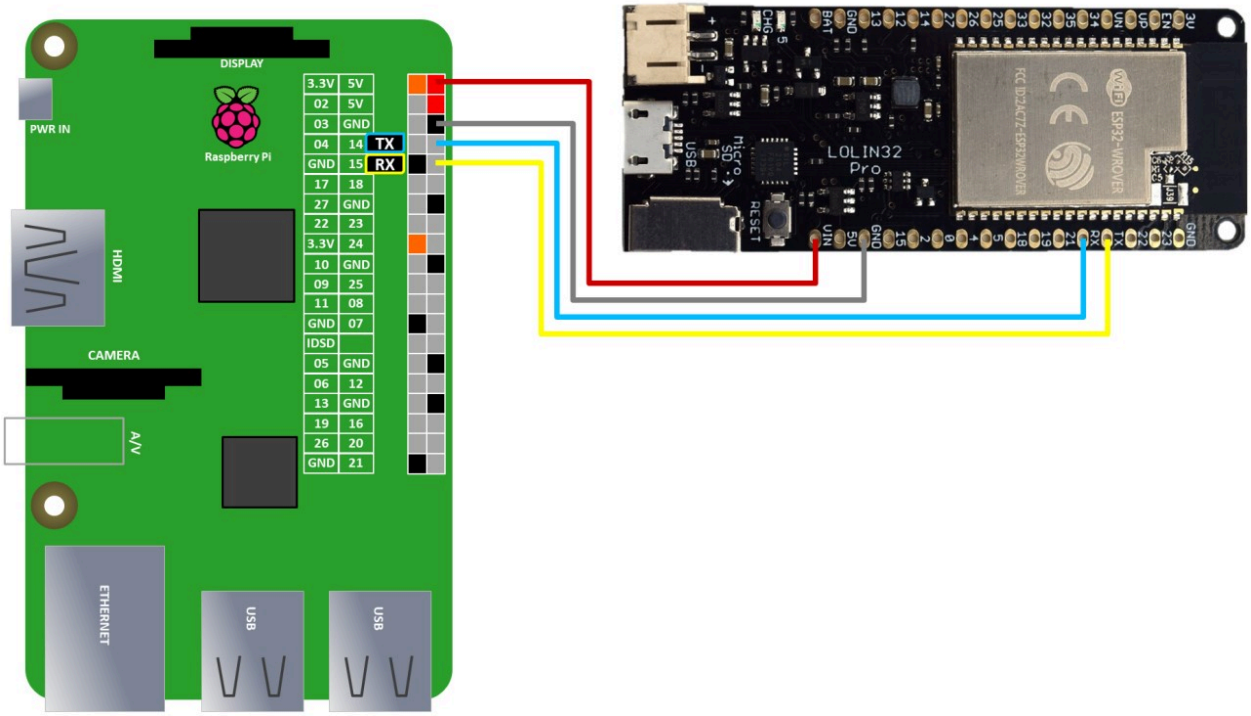
Tensorflow Lite Pipeline Overview

Since there was no operating system or filesystem on the ESP32, the base tensorflow model must be converted to a tensorflow lite model. This model was then converted to a raw bytestring that would later be decoded by the software that would later be flashed onto the ESP-32. The amount of storage saved by converting our model to lite only condensed it by a factor of around 5, which was much less than we had anticipated. The model we chose just barely fit onto the onboard flash memory of 4MB but due to internal reserved memory and memory overhead, the actual available memory was even smaller at only around 3.3 MB.

Once all this had been done, we encountered another roadblock. Even with the model uploaded and available, we would still have to take our pictures, convert them into a tensor that would be able to be read by the model, and only then could we obtain a result on the image classifier. Given that the smallest resolution of our image that we could set on the ESP-32 was a 320 x 240 color image, the resulting tensor was far too large to fit onto the onboard RAM. With rough calculations, each image as a tensor would be 900 kb of data. With the ESP-32 only having around 520 kb of onboard RAM, it was simply impossible to be able to fit a tensor onto the board to even begin to analyze. While there was an onboard external 4MB of PSRAM we later realized could be utilized, after calculating the actual cost of the convolutions that would be

necessary to run inference we found that there was no way the entire model's computation could fit on the device.

At this point the idea of using an onboard inference model was scrapped, and we decided to pursue the idea of running the bird model training on a separate module. The new idea was to use a Raspberry Pi as our inference engine, and simply send data to the Raspberry Pi to be classified and stored, instead of using an SD card to hold the data for the bird images.



Planned UART Wiring between ESP-32 and Raspberry Pi

With this in mind, we transferred our model to a fresh Raspberry Pi with Ubuntu 22.04, and were able to run our model in the linux environment successfully. Due to the way our birdhouse was designed, we would not be able to use the shield and its usb cable connection to communicate between the Raspberry Pi and the ESP-32. With this in mind we decided to transfer the image data over serial communication using the UART protocol and TX/RX pins on both the ESP-32 and Raspberry Pi. While we were able to get communication across between the two devices, we were ultimately unable to get the JPG images to transfer properly, as the resulting files simply would not properly be formatted once transferred as a bytestring to the Raspberry Pi.

While we could have continued to pursue the machine learning portion of the project, we decided to turn our efforts to making the birdhouse portable and battery operated. While using the Raspberry Pi most likely have worked, it would have completely eliminated the ability for the birdhouse to run for long periods of time on battery as the onboard Raspberry Pi would have taken too much power to support via the planned battery pack. This went against the main purpose of the project, which was to be able to leave the birdhouse for an extended period of time and collect data afterwards.

VII. Future Improvements

As we were unable to properly implement the machine learning portion of the project, a possible improvement could be to research and develop a custom and even more compact model that would be able to take advantage of monochrome images. Because of the fact that our model was built upon an existing model architecture, it was not able to process non-color images, and by converting the images to grayscale we could have reduced the number of channels down by a third, resulting in image tensors of around 300 kb which may have been able to fit on the microcontroller. We also could have looked into another way to possibly have another dedicated microcontroller other than a Raspberry Pi that could have been used to run inference and potentially have been low power enough to suit our needs.

For a final product, we would also want to implement power saving measures so that the camera can last longer on a battery. This can include putting the ATMega328p and the ESP32-CAM to sleep when not in use, removing the LEDs on the stepper motor drivers, using lower power microcontrollers to begin with, and more.

VIII. Conclusion

Our project, the Pecker Peeper, is designed to automate much of the down time of bird watching, which is especially useful for those with full-time responsibilities. Birds that feed activate a 5 kg load cell, which turns on a light and captures images of the bird on an ESP32-CAM. These images are then saved on an SD card, which can be retrieved by the user. The feeding process is also automated with stepper motors and a timing mechanism driven by the RTC.

Future endeavors with this project would include the implementation of the machine learning processing, which would identify the birds for the user, and the inclusion of a portable battery for outside use.

This senior design project allowed our team to build the project from the bottom up. We gained a better understanding of the design process and the common, unexpected issues that always come up when planning a product. For our future projects, work, and ventures, we will remember the lessons learned and tackle the development process with realistic expectations, giving way for timeline fluctuations and errors.

Signature Page

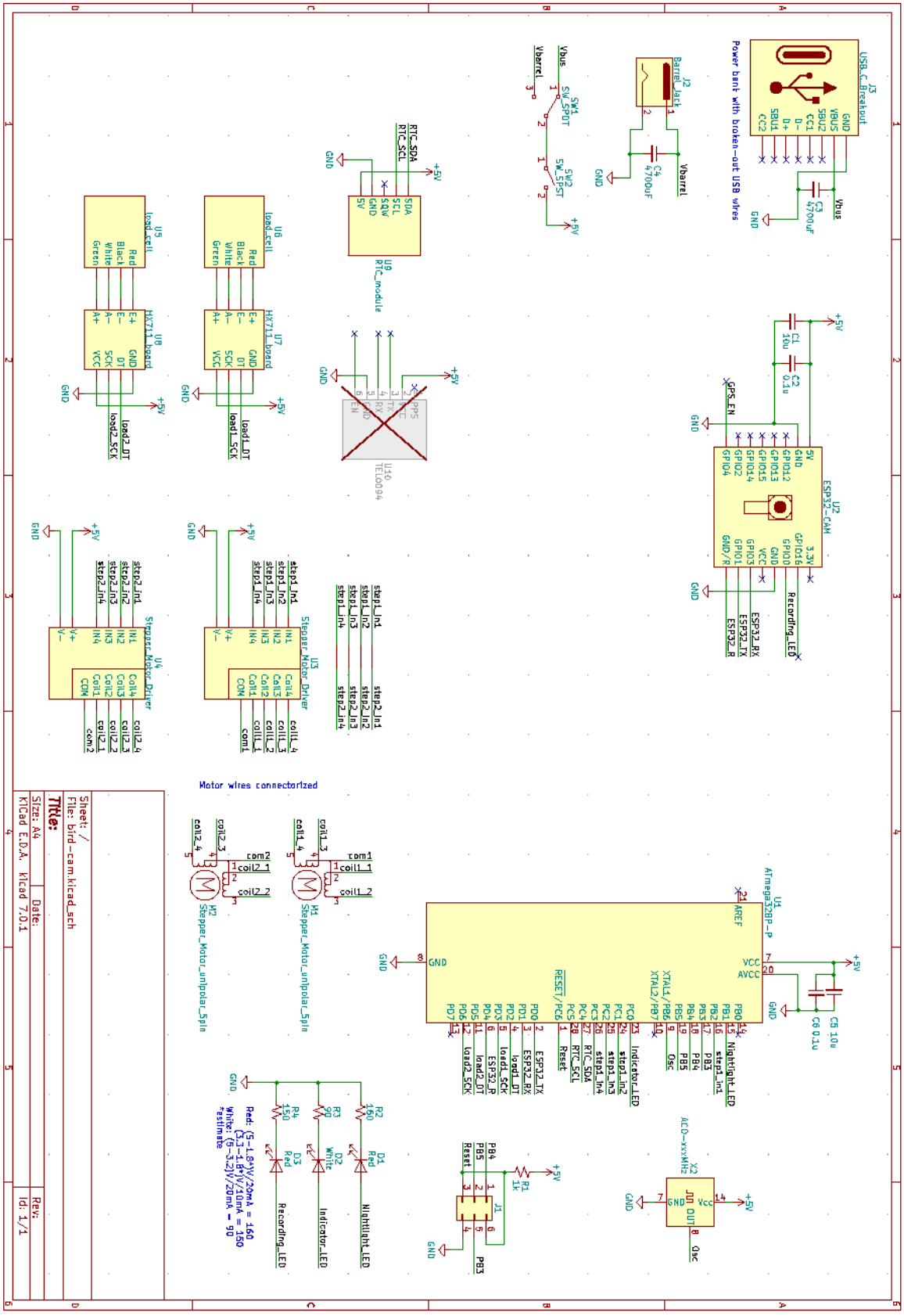
	Brandon	Chur	Isaiah
System Design	40	35	25
Hardware	55	25	20
Firmware	55	25	20
Machine Learning	10	10	80
Project Presentation	25	50	25
Project Report	35	30	35

Chur Tam

Brandon Gong

Isaiah Lee

Appendix A - Schematic



Appendix B - Parts and Cost Breakdown

	Name	Price	Purpose
Purchased Online	5kg Load Cell	$11.99	Weight sensing
	Stepper motor and driver	$14.99	Food dispensing
	ESP32-CAM	$14.99	Camera
	USB C Breakout	$2.95	Power
	10000 mAh power bank	$19.99	Power
	uSD card and adapter	$10.00	Video storage
	Heat set inserts	$19.90	Assembly
	M4 Machine screws	$9.99	Assembly
	M4x4 Button head screws	$9.19	Motor mounting
Purchased from BFMS	3D printed Parts	$295.52	Physical enclosure
Available in parts library	ATMega328P	x	Primary microcontroller
	3 LEDs	x	Nightlights and Indicators
	Sparkfun BOB-12708	x	Video timestamp
	Power jack	x	Power
	5V wall wart	x	Power
	JZC-11F Relay	x	ESP32-CAM power
	SPDT toggle switch	x	Power switch
	Pushbutton	x	Control
	Total:	$409.51	