ECE/COE 1896

Senior Design

Cupboard Closer Final Report

Team 2

Prepared By: Brian Danielson
Sierra Gordon
Madeline Love
Matt Spering

Table of Contents

[When you want to update your table of contents (i.e. prior to turn-in), Right-click on table, select 'update field' and select 'update entire table']

Table of Figures

Table of Tables

1. Introduction

The Cupboard Closer is Group 2's solution to a common frustration: to the chagrin of mothers everywhere, cupboard doors are often left open after use. The prototype followed requirements that accounted for potentially impeding the user's access to the cabinet: the design could not be so large as to make the cabinet unusable – this affected the group's design of the native power solution and the housing the electronics would be enclosed within. The closure mechanism was required to be safe for the user, stopping the closing action if the camera inside detected motion entering the space. Finally, the cabinet door needed to be robustly connected to the motors that closed the door. The design was constrained by a number of standard factors within R&D such as time, manpower, and budget, though additional factors such as power consumption, connection latency, and size contributed to how the prototype was designed.

Within the system, specific hardware was designed for the purpose of limiting power consumption while maximizing closing mechanics: a 9 V AA battery bank was chosen for its small profile, necessitating a solution that would extend the life of the design. A novel design utilizing the motor operating under generation conditions allows for power to be recouped from each open/close cycle by reversing the motor polarity via a network of MOSFETs controlled by the native MCU. The motor itself is driven by a PWM signal generated by the MCU and is both controllable and sufficiently fast for user convenience. The onboard MCU receives signals from the sensors and drive system to determine cabinet state, prioritizing power saving through switch control to the drive and camera unit. The controller is also programmed with a hand-recognition algorithm and motion detection that communicates via Amazon Web Services servers for processing at fast speeds without requiring storage within the device itself. The prototype's hardware was tested using an array of multimeters and scopes to allow for the group to visualize the flow of electricity through the design; by quantifying the power consumption the efficiency and power recoupment of the system were calculated and plotted. The recognition algorithm compiled and analyzed hundreds of images of hands to achieve the most comprehensive, robust database possible. The prototype's initial results yielded findings in line with expected: hands were detected over 80% of the time. The communications were simulated upward of 100 times in less than 1.5 seconds, proving capable of recognition, communication, and signal generation to halt closure five times faster than the cabinet's closing time. The power consumption of the design based on testing yielded a battery life of 2.5 hours of continuous operation, while the regeneration system extended the battery life by over 10%.

Of the requirements discussed above, the prototype sufficiently fulfilled all of them: the housing of electronics and mechanical design does not impede the user's access to the cabinet, the cabinet is simple to open, and the recognition and communication time is quick enough to halt closure onto the user's hand. The prototype is not ready for market as of this time however, as the native hardware solution requires a redesign to account for component trouble encountered toward the end of the process. The mechanical design is functional but would need reproducibility over large economies-of-scale to sufficiently meet the goals of releasing the product to the public. Ideally, the accuracy of hand detection would increase provided more training to eliminate false positives and negatives and the companion application would be completed and released on app stores for most devices.

2. Background

Health and safety are the primary problems to be solved by the cupboard closer. Office and kitchen safety specifically are impacted by open cabinets that are not properly closed; in the office setting, the most common accidents are related to falls, striking an object, or an object striking the worker. Open cabinets left unattended pose a safety hazard that could lead to each of those incidents occurring, as a worker may fall over or be struck by an object which originated within a cabinet. According to the Albert Einstein College of Medicine, office workers are 2 to 2.5 times more likely to suffer a disabling injury due to falling in the workplace than their non-office worker counterparts. One of the primary prevention methods mentioned is to remember to close cabinets and drawers after using them. A straightforward way to prevent this occurrence is the elimination of the human element responsible for leaving a door ajar [22, 23]. Additionally, the kitchen setting creates danger due to sharp or hot objects and/or a lack of free hands from carrying something. Like the office setting, open cabinets within the kitchen are a hazard due to falling objects and are classified as a safety deficiency in a home setting according to the Department of Housing and Urban Development's NSPIRE Standard for Real Estate [24].

These safety risks carry economic consequences for the employer due to Worker's Compensation laws dictating the workplace have insurance to cover employees injured on site who require a doctor's visit or hospitalization [25]. In fact, according to the Bureau of Labor 305,390 people (about half the population of Wyoming) recorded a fall, slip, or trip that resulted in missing day(s) of work in 2019. These incidents were not all the result of an open door or cabinet; despite the per capita value being less than 0.1%, the cost to both employees and employers of missed workdays is statistically significant, given the increased rate of disabling injury that occurs in the office setting. Below is a table compiled from the Bureau of Labor between the years 2011-2019 detailing the number of such incidents occurring [26]:

Year	Annual
2011	302890
2012	289470
2013	296130
2014	316650
2015	309060
2016	292580
2017	290660
2018	306460
2019	305390

Table 1: Yearly Data Counting Instances of Workplace Falls Resulting in Missed Workdays; Bureau of Labor Statistics.

The Bureau of Labor Statistics also contains data from the same years on injuries from swinging or slipping objects resulting in missed workdays. This smaller list more accurately targets the number of people affected by open doors and cabinets than the broader compilation above:

Year	Annual
2011	12850
2012	12970
2013	12090
2014	12310
2015	12880
2016	12440
2017	14510
2018	12430
2019	13120

Table 2: Yearly Data Counting Instances of Workplace Injuries Due to Swinging/Slipping Objects Resulting in Missed Workdays; Bureau of Labor Statistics.

The cabinet closer design is a novelty amongst a crowd of door closure solutions. All the current iterations are purely mechanical designs, from small spring-wound hinges (Figure 1) for cabinets to hydraulic connecting rods like the stereotypical door closure mechanics seen in professional settings (Figure 2). A major component of the cupboard closer that differentiates it from these common iterations is the addition of electrical and software components. Instead of using hydraulics or windings, the cupboard closer utilizes a motor-based system that is controlled by a microcontroller fed by sensors and commanded via a digital interface or visual interaction from the camera.

Figure 1: Spring-Loaded Hinge; HSJJ via Amazon.

Figure 2: Hydraulic Commercial Door Closer; FORTSTRONG Hardware via Amazon.

The mechanical solutions do not provide the additional robustness contained within the software control. This may potentially result in additional injury from cabinets or doors slamming shut on a hand or obstruction.

3. System Requirements

This section details the functional and non-functional requirements of both the hardware and software modules.

3.1 Functional Requirements

3.1.1 Cabinet Closure Timing Requirements

The cabinet must close itself in a reasonable amount of time. As further detailed in (3.2.2), the hardware component must be reasonably fast; to this end, the PWM duty-cycle input to the motor drive must be sufficiently high to drive the motor switch. The motors themselves must be capable of a high enough RPM to pull the cord closed in a timely manner. The cabinet shall close within 30 seconds of inactivity. If a hand is present during this test period, the period will reset. Additionally, if motion is detected within this period, the period will be reset. This timing requirement was met by the prototype, the data for which is detailed in (7.2.2.1).

3.1.2 Motor-Cabinet Interface Requirements

The system must use a robust connection to connect the motor shaft and the door to enable the closure of the cabinet – the use of weak connections would result in breakage, hindering or disabling the prototype from completing its task. This can be completed using wound-wire connections or a more complex shaft connection as detailed in (5.1.3.3).

3.2 Software Requirements

3.2.1 Communications Requirements

3.2.1.1 Internet of Things (IoT) Connection

To maintain an IoT connection, the system shall have Wi-Fi capabilities.

3.2.1.2 Internet of Things (IoT) Messages

The system shall use an IoT protocol to communicate between the components of the system. The cupboard closer will capture an image and broadcast the files to the image classification algorithm. If it is determined that a hand is present, a message will be sent to the cupboard closer, confirming the door can close or stay open.

3.2.1.3 Internet of Things (IoT) Connection Lost

If the IoT connection is lost, the timeout period for the cupboard closer shall be long enough to assume that there is not a hand in the way. Upon reconnection, the system shall resume broadcasting images if the Internet connection is not lost.

3.2.2 Internet Connection Lost

Since the system is entirely dependent on the internet, if the internet connection is lost, the capabilities of the broadcast and image system shall stop.

3.2.2.1 Internet of Things (IoT) Provisioning

The provisioning process shall be confined to the companion app, with the functionality abstracted away from the user.

3.2.3 Internet of Things (IoT) Packet Lost

If a packet is lost in transmission, and it is essential for communication, it shall be resent immediately. Commands issued from the app to close the cupboard, if not received, shall display the failure to the user within the companion app. Commands issued from the cupboard closer to the cloud, including image processing requests, shall be cached upon failure, and retried immediately.

3.2.3.1 Communications Protocols

The system shall contain a protocol for acknowledgement of IoT messages' arrival to capture any packet failures.

3.3 Companion App Requirements

The cupboard closer shall be accompanied by a companion app, developed for Windows, to house the image classification algorithm.

3.3.1.1 Companion App Close

The companion app shall know if the cupboard is open or closed. If the cupboard is open, there shall be an option to close it remotely.

3.3.2 Companion App UI

The companion app will be designed to be reasonably useful; ease of use, habitual, and accessibility features will be considered.

3.3.2.1 Companion App Provisioning

The companion app will contain the functionality for creating and maintaining an IoT connection between the cloud, the app, and the cupboard closer. This provisioning process shall be abstracted away from the user.

3.3.3 Companion App Customization

The companion app shall contain the ability to customize the close timeout period within a range. This information shall be sent in an IoT message to the cupboard closer. Based on this customization, the timing set in requirement (3.1.1) shall be changed.

3.3.4 Companion App Wi-Fi Settings

The companion app shall contain the ability to customize the Wi-Fi network in which the cupboard closer lives and ensure that the device reconnects correctly.

3.3.5 Closing Decision Making Requirements

It is required that the system has safety features that will not allow it to close while someone's hand is in the way. The system will use the motion sensor in tandem with the camera’s hand detection described in (3.2.4) to ensure that all users hands are not going to be closed in on. While the motors should not cause any harm in closing, it is better to be safe than sorry.

3.3.5.1 Closing Timeout

The cupboard will start closing after 30 seconds of inactivity, meaning there is no motion in the cabinet or hands detected. Realistically, it may take longer if needed as in a real scenario, time is not of concern, but for testing and demonstration, it should be quicker, hence the 30 seconds.

3.3.6 Hand Detection

It is required that the system can identify a hand. It is desired that the system can detect a predetermined hand symbol which it then knows as the user wanting to close the door. To not close the door on any hands we must detect the hands. The hand detection must be accurate for the closing system to work properly. If the accuracy is not perfect, the finite state machine controlling the closing decision-making system can be modified to deal with that.

3.3.6.1 Hand Detection Outcome

The algorithm shall take an image as an input and determine if a hand is present. The algorithm shall output a label indicating “yes, there is a hand” or “no there is not,” and communicate that through requirement (3.2.1.2).

3.3.6.2 Hand Detection Timing

For the hand detection to work the algorithm for hand detection must be faster than the frame rate that is pulled from the camera. The frame rate can be lower to give the algorithm more time to process the image, but at minimum there should be two frames per second pulled from the camera. This results in the algorithm needing to take less than a half a second.

3.4 Hardware Requirements

3.4.1 Power Requirements

The system must be powered natively by an on-board battery system. The system requires native power due to physical constraints of the device: it must not interfere with the closing of the cabinet door and correctly proportioned such that it does not obstruct space within the cabinet itself.

3.4.2 Power Regeneration

The system shall be battery-operated, and the system should contain a power regeneration system to extend the life of the batteries.

3.4.3 Motor Drive Requirements

The drive-system for the motor must be controlled via PWM duty-cycle input from the on-board microcontroller. It must be sufficient to drive open an n-type MOSFET with an appropriate gate-voltage based on the chosen switch. The power bus feeding the motor drive must transmit enough power to the motor system such that the current provided does not impact functionality of the motor.

3.5 Ease of Use Requirements

The system shall be reasonably easy to open. It should not be a struggle to open the cabinet with the cupboard closer attached, and the closing mechanism shall not interfere with opening the cupboard.

3.6 Sensor Requirements

3.6.1 Motion Sensor Requirements

The prototype shall use a motion sensor able to detect motion of the door and obstructions of the door such that it can operate as a trigger for camera power and a safeguard in darker environments.

3.6.2 Camera Requirements

The camera should capture images with a clear resolution to accurately process them.

3.6.3 Wi-Fi Transceiver Requirements

The Wi-Fi transceiver shall be strong enough to create and maintain an IoT connection, as well as transmit and receive IoT messages. It also should be strong enough to transmit files for image processing.

3.7 Non-Functional Requirements

3.7.1 Communications Requirements

3.7.2 Image Classification Accuracy

The hand detection algorithm shall have at least 50% accuracy, though it should be higher, upwards of 75%.

3.7.3 Image Classification Latency

The hand detection algorithm shall produce a result for an image within 0.5 seconds.

3.7.4 Broadcast Accuracy

The system shall receive any IoT messages with 100% accuracy of their contents.

3.7.5 Broadcast Latency

The system shall transmit and receive IoT messages within 0.5 seconds.

3.7.6 Transmission Accuracy

Over 90% of all messages transmitted by any device will be received by their intended destination. 10% of messages are allowed to fail. If the failed messages contain files, requirement 3.2.1.2 will account for any necessary caching, or retries. If the failed messages are standard IoT messages, requirement 3.2.1.4 details the protocol for any misses.

3.7.7 Communication Range

The system shall be able to communicate within the range of a large room. IoT messages shall be accurately transmitted and received within 300 feet between the cupboard closer and the companion app.

3.7.8 Communications Security

The system shall communicate securely, so messages between devices in the local IoT will not be intercepted. The channel in which they communicate shall be abstracted away from plain view.

3.7.9 Physical Requirements

The device is required to be small enough to fit into a typical cabinet. It may not block items or interfere with normal use of the cabinet; this includes external power cabling as detailed in (3.3.2). A figure showing the prototype is contained within (5.2.3).

4. Design Constraints: Standards and Impacts

4.1 Design Constraints

4.1.1 Time

We must complete this project within the twelve weeks we have for the summer semester. This time constraint also includes the workload other team members have committed to. Each team member has contributed 5-10 hours a week individually to complete this project on an accelerated timeline. This design constraint is addressed in (8.1), where each team member has set goals for themselves, in addition to the checkoffs and presentations.

4.1.2 Budget

We are constricted to a budget of $200 from the ECE department to create our prototype. The team has considered this budget when selecting parts to build the prototype, and we were unable to afford the purchase of several components to test out their capabilities. Additionally, we are constrained in our use of cloud computing services, as there are limits to the free versions of the software we will be using.

4.1.3 Labor

The four of us on our team are the sole members developing the project. Team members must communicate clearly, ask for help when needed, and be responsible for their individual modules of the project. This communication will be crucial when the time comes to integrate our components to ensure a successful, testable prototype. Since we will be working within 12 weeks, team members will be responsible for addressing roadblocks and navigating individual obstacles, to ensure they are not holding up the progress of the prototype.

4.2 Project Constraints

4.2.1 Latency for Image Processing Algorithm

It is required that any message be broadcast and received within the timing described in requirement 3.5.1.4, and it is required that the image processing algorithm produces an output in the timing described in (3.5.1.2). For the worst-case scenario, from the time that the image is taken, to the time that an IoT message confirming or denying the presence of a hand, shall be at most 1.5 seconds. The project is constrained to produce an output for any given image within 1.5 seconds, which is a reasonable estimate for a cabinet timeout set in (3.1.1). This means that all parts must be seamlessly integrated, such that there are no bottlenecks at any point of this message transmission.

4.2.2 Size of the Prototype

The device cannot be so large that it obstructs the cabinet functionality. This size constraint dictates some of the trade-offs of the system that need to be accounted for such as motor size, battery size and quantity, camera size, PCB size, and cord obstruction during opening or closing. Figure __ shown in section (5.2.3) provides context to the prototype's size relative to a cabinet space.

4.2.3 Motor Power

Because of the size constraints above, the motor must be small enough to fit inside the cabinet yet powerful enough to pull a door shut. The output power equation, $P_{out} = \tau \, x \, \omega_{mech}$ dictates a trade-off decision between RPM and torque of the motor shaft.

4.2.4 System Power Constraints

The Cupboard Closer will be powered by six 1.2V AA 2.0Ah batteries in series, which limits the duration of operation. Summing up the current drawn from the motor, the ESP, and from the remaining peripherals results in approximately 600mA. Dividing the capacity of the batteries by this total draw gives us about 3.3 hours of continuous battery life. An entire cycle (assuming a seven second pause during *open*) runs about twelve seconds. This provides for ~1000 full cycles, although it bears repeating that this is a *very* rough estimate. In reality, a variety of unforeseen and unforeseeable factors will come into play.

4.3 Industry or Government Standards

4.3.1 Consumer Privacy – IEEE 802

In order to adhere to the standards in IEEE 802 [70], privacy is a concern that needed to be addressed within our prototype. While we were not sending sensitive information to the cloud, we are taking pictures, which if they contain sensitive information, will need to be protected. In our case, we are only interested in storing the images to ensure that the camera is working correctly, and to re-train our image classification algorithm as needed. This functionality would not be needed if the product is refined and ready to mass-produce.

To that extent, if the product was to be deployed as a consumer product in its current state, the images collected would only be stored in our AWS database for a maximum of 7 days, where it will then be deleted. Again, we are only interested in the collected images to ensure our prototype is working, otherwise there would be no need to store the images taken with the ESP32-Cam. This is accomplished by creating a rule within our database in AWS, as shown in the figure below.

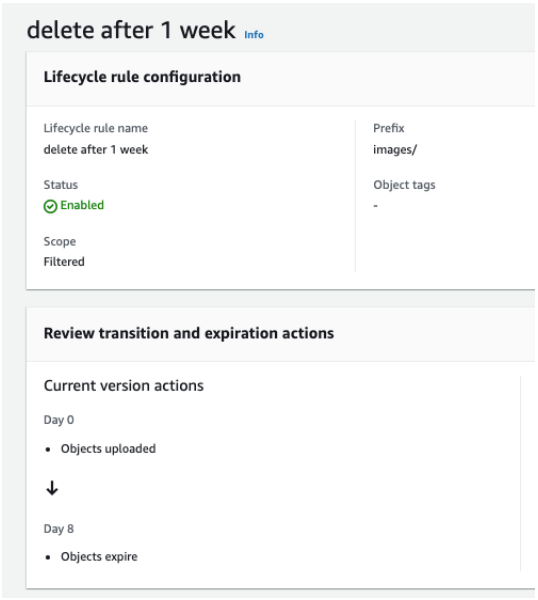

Figure 3: AWS Lifecycle Rule Configuration

Within the standards outlined by the IEEE 802 program, privacy and security are highly emphasized. In addition to ensuring that the images are deleted within a timely manner, the security that we employ in our design lies within the MQTT topics and protocols for our IoT communication. To implement this, we create topics in which the devices can communicate within a unique identifier. The MAC address of the ESP32-Cam will be used to create a secure topic between the camera and the companion app, in order to ensure individuality between a fleet of IoT devices. AWS MQTT protocol creates "topics", where devices can listen on those topics and receive all messages sent to that topic. In order to ensure that we only have one topic per device, we will create a topic to publish and receive on the ESP32-Cam, named after the unique MAC address of the camera. An example of this unique topic name is shown below:

```
incoming: 0C:B8:15:73:F2:2C/receive - {
  "type": "WIFI_CHANGE",
"id": "Maddie iPhone 12",
"pw": "12345678"
}
......WiFi connected
```

Figure 4: MQTT Message Displaying a Wi-Fi Change

We trust that the manufacturer of the ESP32-Cams will ensure that each device has a unique MAC address, and in our testing of two ESP32-Cams, we have observed that the MAC addresses are unique. We are not sure of the uniqueness of every camera, so in order to create a truly unique and private MQTT topic system, we may need to implore our user to generate a unique profile within the companion app. That, however, would be future work if this product was to enter the market.

4.3.2 IoT Device Encryption and Security - TLS

In addition to the privacy and security standards outlined in section 4.3.1, the protocols that are mandatory to create an AWS "thing" within a customized network of "things," are highly secure. As outlined in the standards in [71], TLS is a protocol to ensure that data is transmitted in a secure manner and is employed by AWS through its MQTT messaging protocol [33]. The standards for TLS are set by the Internet Engineering Task Force, as outlined in [72].

Within this protocol is an emphasis on decreasing eavesdropping and malicious tampering within these channels, which includes end-to-end AES-256 encryption, and certificates for AWS "things" in order to live within the IoT network. We have employed the use of AWS's MQTT protocol for our data transmission, and again, while we aren't interested in encrypting highly sensitive data as of right now, there may come appoint where we are concerned with potential malicious activity, and it is important to know that our transmitted data is protected.

The details of the MQTT protocol within AWS can be found within [73, 33], where the policies of creating secure device certificates are outlined. Specifically, the certificates that are needed to join the IoT protocols are public/private key pairs, which can be generated through the AWS IoT Hub [30].

4.4 Impacts in Non-Technical Contexts

4.4.1 Welfare and Safety

The cabinet's camera-based detection algorithm is in place to determine if an obstruction, specifically a hand, occurs. Despite the cabinet not closing rapidly, the prototype will be designed to avoid the complication of a person's hand being collapsed upon by a closing door. Also, the door closure corrects the safety risk of an open door on passersby who may injure themselves. The motors that were used are non-compliant with RoHS (Restriction of Hazardous Substances), eliminating the possibility for sale in the EU.

5. Summary of Design

5.1 Design Alternatives

5.1.1 Motor Drive Alternatives

An alternative to the motor drive design that was discussed was the use of an H-bridge to allow for reversal of motor polarity. After discussing with Dr. Brandon Grainger, the choice to use the simpler system was chosen to decrease complexity. However, a separate MOSFET array was used to allow for the polarity reversal during the regeneration state to prevent leakage from the power bus and maximize the energy stored by the capacitor.

5.1.2 Object Detection Algorithm Alternatives

There were two unique design choices that needed to be made to do object detection. The first was whether a model should be trained to specifically find a hand, or if we should detect motion. In the case of detect motion we would use what is called optical flow, the camera would act like a motion detector, it would look for how much the pixels change from frame to frame. Optical flow is a higher power and more complicated motion detector which we could just buy, and would specifically be able to see hands, which is what we wanted. The second design choice lies within the first design option, if we train to find a hand, the type of algorithm used to train it needs to be chosen. I ended up choosing to detect specifically a hand using a Neural Network algorithm because this algorithm had the better results. The original plan was to do hand detection with a Support Vector Machine algorithm, but when testing the trained models, it performed very poorly.

5.1.3 Communications Protocols Alternatives

The design needed to maintain an IoT connection to communicate its environment to other devices. At a minimum, the system needed to know whether the cupboard is open or closed and must communicate that status to the user. Our system does not need any large-scale industrial web services, nor does it need to house a large fleet, so our team looked for a low-cost web service. To complete the communications protocols outlined in the requirements, alternatives like Google cloud were considered. In our research, we noticed that the Google Cloud web service for IoT seemed to provide custom solutions to a large-scale IoT fleet [29]. Our process in Senior Design is focused on learning and applying concepts we have learned in previous classes, and if we had outsourced this communications design, we would have customizability, but we would likely not partake in the actual design itself. Additionally, creating and contracting a customized IoT web service would likely fall out of our budget.

5.1.4 Image Processing Alternatives

While Google cloud provided an alternative for our cloud communications needs, AWS was decided for the overall system design. Within AWS, a service called AWS Lambda was originally chosen to host the image processing, to determine if a hand was present [34]. This design, however, proved to not work within the project, due to a function size constraint. Any AWS Lambda functions were constrained at 250MB, including packages, and the packages for the image processing algorithm was at least 400MB. Therefore, while this was a part of the original design, it was removed in the end, and is therefore placed here as a design alternative.

5.1.5 Power Regeneration Alternatives

In our initial power regeneration design, we intended to recharge the six AA batteries used to power the cupboard. Over time, it became clear that this was not ideal – the purpose of the system was simply to extend the battery life of a single set of batteries. Charging them is actually a formidable task; typically, it is done by inserting them into a special device that plugs into a wall outlet. This device “trickles” current into the batteries over the course of many hours. Since we do not plug our cupboard into an outlet, it is difficult to imagine a method of retaining the charge generated by opening the door for long enough to facilitate this hours-long trickle charging. The

solution was to incorporate a supercapacitor – with each opening of the door it is charged, and with each closing, it is discharged into the power bus.

5.2 Design Overview

At a high level, the physical cupboard closer contains a motion sensor, a contact sensor, an ESP32-Cam, which contains a Wi-Fi transceiver and a camera, motors, an ATMega microcontroller, to control logic within the sensors, and a power regeneration circuit. These physical components drive the logic for the cupboard closer, ensuring the device can take pictures when the door is open, in order to determine if a hand is present in order to close.

We have a companion app, that houses an image classification algorithm, in order to determine if a hand is present before automatically closing the door. This is set on a timer within the ATMega, and is reset if motion is detected, or if a hand is present. In order to determine if a hand is present, the ESP32-Cam takes a picture when the contact sensor is disconnected and keeps taking pictures until the door is closed. After this picture is taken, it is packaged into a JSON message, to be broadcast over Amazon Web Services (AWS) in an MQTT protocol. This image is then received in our companion app, processed by an image classification algorithm, and the result of the algorithm is broadcasted back to the ESP32-Cam. There, it will tell the ATMega that a hand has been detected, and the timer resets. If the timer expires, the motors will automatically close the cabinet door, and the ESP32-Cam will stop taking pictures.

These modules are illustrated in the figure below, where the functionality of each component is highlighted in the diagram.

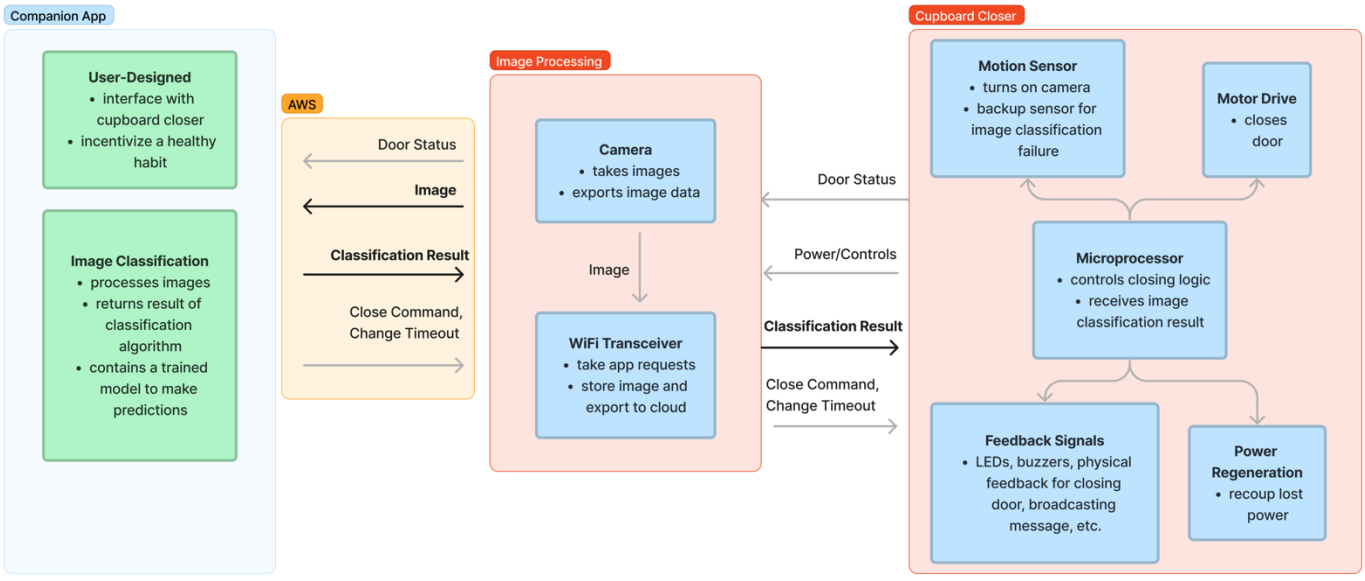

Figure 5: Design Overview

5.3 Electronics Design

5.3.1 Drive System and Regeneration Circuit Design

Much of the drive system circuitry remains from the original concept design. Additional switches (two PMOS and one NMOS) were added to allow for the regeneration state to take place and limit the leakage of energy before closing state began. In Figure 6 below, the circuit for the drive system design from our Altium schematic is shown. The diode and capacitor with the accessory label "_drive" are in place to protect the motor. The diode allows excess energy called back EMF which remains in the motor after operation ends to discharge into the capacitor. This energy is then stored until the next iteration of closing. The MOSFET labelled "sw_drive" is a high-energy n-type switch with voltage constraints at 60 V and current constraints at 1 A, both of which far exceed the possible power the motor draws. It is controlled by a PWM signal sent from a digital pin on the MCU and has current limiting and pulldown resistors for protection and control purposes, respectively. Its source is connected directly to ground to allow V_{GS} to remain in the triode (linear) state during operation, following the constraints $V_{DS} < V_{GS} - V_{Th}$ (where V_{DS} is 0 V and $V_{GS} - V_{Th}$ is 4 V) and $V_{GS} \geq V_{Th}$.

P-type MOSFETs were chosen for switching within the motor drive to eliminate the need for gate-drivers in the circuit, as the conditions for linear operation are inverted from the n-type switch described above. The switch labelled "sw_P1" disconnects the drive system from the bus during regeneration, which prevents current from flowing through a short-circuit to ground via "sw_N1". Switch "sw_P2" disconnects the high side of the motor while in generation mode from the reverse-biased diode, eliminating another potential short-circuit to ground. Finally, "sw_N1" connects the low side of the motor while in generation mode to ground such that it can act as a source for charging the supercapacitor (labelled "SC") while in regeneration mode. Each of these MOSFETs is controlled by a digital signal "relay_1" generated from the MCU which follows Figure 16's "r1" value.

A 5 V linear regulator fed by the high side of the motor during generation mode enables the switching between states 1 and 2 upon fall from 5 V to 0 V, which is further discussed in the following section on microcontroller logic. Additionally, two diodes are placed around the shunted supercapacitor to prevent backflow from the bus to the capacitor and from the capacitor to the drive system. A resistor connects the bus to the capacitor which allows for measurement and control of the current output supplied to the bus.

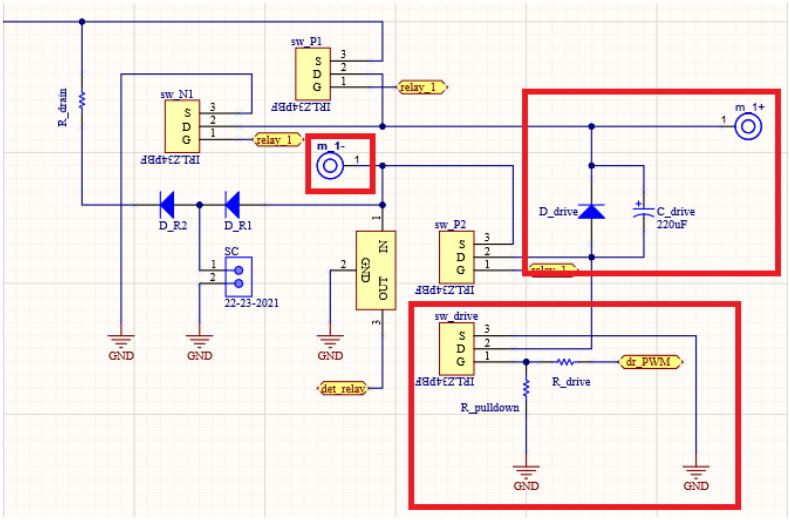

Figure 6: Motor Drive and Regeneration Circuit (Motor Drive Highlighted)

5.3.2 Power Bus Design

One of the major design changes made between the initial design and the prototype was the elimination of the 24 V bus that would feed the motor-drive system. Because the motor's operating voltage is 6-30 V inclusive, the battery bank of 6 AAs (7.2-8.4 V operating range) was sufficient to power the motor. This made the inclusion of a boost converter unnecessary at best and detrimental at worst, as the step-up of voltage also necessitates a step-down of current of the same proportion. Instead, a two-bus design was implemented: a battery-voltage bus and a 5 V bus fed by a linear regulator. The battery bus powers the motor drive system and the ESP32-CAM, the latter of which includes an onboard 5 V regulator, while the 5 V bus feeds the motion sensor, contact sensor, and MCU. For visual reference, Figure 7 has been provided below. Additionally, protection for components was added to prevent any backflow into the battery and smooth the voltage inputs to sensors and ICs.

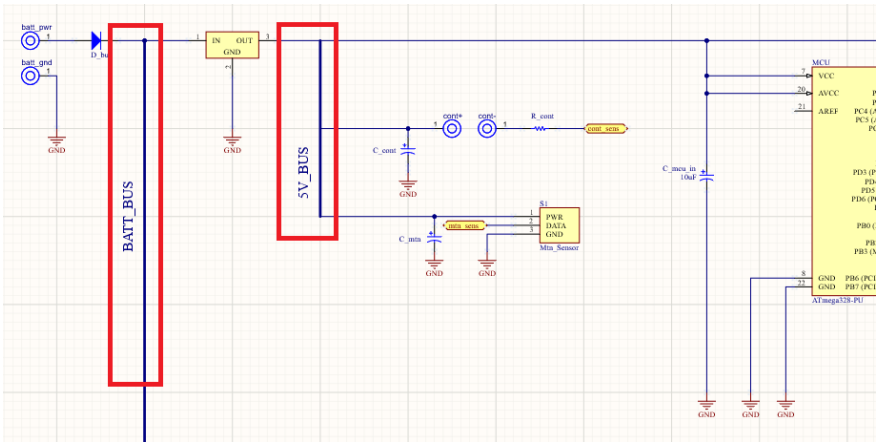

Figure 7: Power Bus Design

5.3.3 Breadboard, PCB, and Housing Design Implementation

Though the stated goal was to create a PCB for the electronics design to be housed on, the time constraint and a previous iteration using relays caused the need to pivot away from the PCB.

To elaborate, the relays chosen to conduct the switching did not appropriately account for the current generated during regeneration-mode, burning out the relays and rendering them ineffective. Attempts to drill through the PCB routing to bypass them in favor of jumper cables to the MOSFET iteration proved difficult to implement, thus an executive decision was made to pivot.

Due to the size requirement, a small breadboard was created to implement the new switching design during the final week that would fit inside the housing created for the prototype. This breadboard was created from two of the miniature breadboards used in junior design snapped together and housed all the necessary components for the prototype's operation. The housing, though originally created for the PCB to be drilled into it, was sufficient to contain this new implementation. A hole was drilled into the side of the housing to allow for cabling to reach the battery supply, mounted motor, and contact sensor. The two figures below show the breadboard's implementation and the final housing with electronics in it. The motion sensor and ESP32-CAM were mounted onto cutouts made on the lid of the housing.

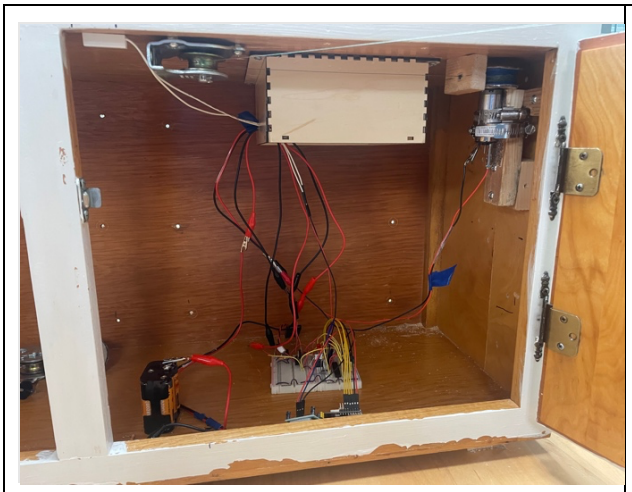

Figure 8: Unboxed Prototype

Figure 9: Housed Prototype

5.4 Object Recognition Algorithm

5.4.1 Trained Model to Detect Hands

As previously mentioned, the chosen algorithm to detect a hand is a Neural Network. To train the algorithm, the first step is to do the histogram of gradients of all images. The histogram of gradients is like edge detection, which comes in the form of a feature vector, this is what we train the algorithm on. Neural Networks are a black box, the input is the feature vector of all the images we are training on and then it outputs a classification of 0 or 1. Throughout the semester many models were trained and tested.

With time, the process of making models was streamlined, though time consuming. The first step was the histogram of gradients preprocessing step, in a function called "makeHOG." This took the most time, at least 2 hours, depending on the number of and size of the pictures. The feature vector of each image was added to an array, and the classification of the image, hand or not, was stored in a separate array. These arrays were then saved as CSVs so models can be trained

repeatedly without redoing this step, but every time new images were added to train with means this long step must be repeated.

Now that we have the CSVs with the data it is time to train the model which was done in a function I called “TrainAlg.” The accuracy of the model corresponded to the images it was fed and the size of the neural network. The more images the better the accuracy tends to be. The other thing that can be changed is the size of the neural network. The figure below is an example of what a neural network looks like. Each circle is a node, you can change how many nodes are in each layer and how many layers there are. The more layers and nodes per layer there are, the better accuracy the model had. There was a limit though, if the network is too big it will not converge if there is not enough training data. The more images collected, the bigger I could make the network before it would no longer converge. To summarize, the more images there were the better accuracy there was, but it also meant I could increase the size of the network, which also significantly increased accuracy. Once the model is trained it is saved as a skops file so it can be loaded again for testing and by Maddie’s app.

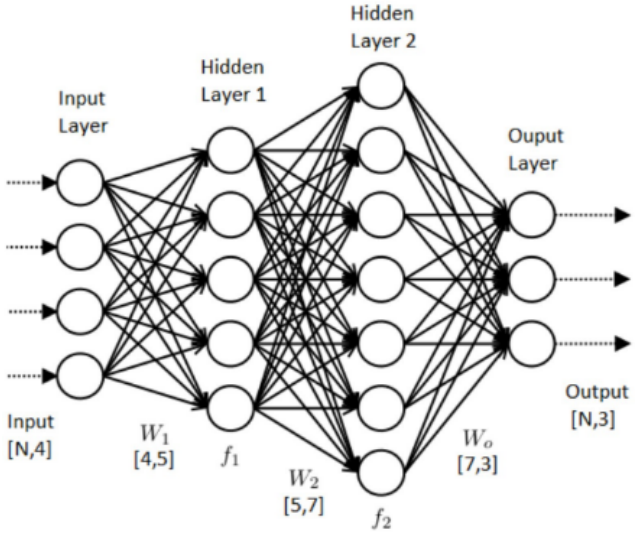

Figure 10: Reaching Local Minimums Instead of Global [45]

The last step is testing the model. There are different images from what the algorithm was trained on that the model is then applied to. The model will give an output of 0 or 1 to indicate hand or no hand, respectively. The model’s correctness can be verified by seeing whether the picture was pulled from the “hand or no” hand folder. The number of pictures correctly classified is kept track of so we can determine the accuracy of the model. If the accuracy is not particularly good, I would try retraining the model, making the size of the network and see if it would converge. If I cannot find parameters that make the model converge, I must collect more images and start the process again from the start.

5.5 Microcontroller Logic

The microcontroller also needed to be programmed to be able to make decisions on when to close the door, send the PWM signal to the motor and what to set the relays to. The figure below describes the finite state machine that makes these decisions.

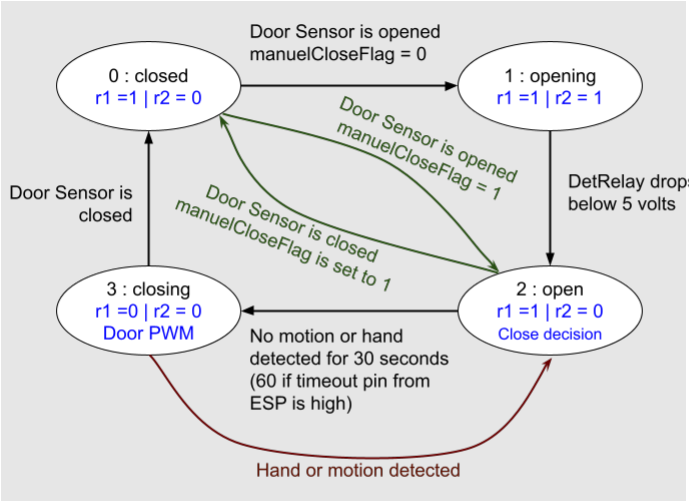

Figure 11: Microcontroller States Diagram

The closed state is an inactive state, not much is happening besides waiting for the door to open. Once the door opens, which we determine from the contact sensor, we enter the opening state. The opening state is where the regeneration is happening as one pulls the door open. We want to end that state as soon as the door is not being pulled so the capacitor does not discharge. This is done by looking at something that tells us how much voltage the capacitor is getting. Once it drops below 5 volts, the capacitor is no longer going to be charged, so we switch to the open state.

In the open state we are looking at the motion sensor and the output for hand detection from the ESP. There is a counter that will continue to increase if there is no motion or hand detected. If there is either a hand or motion, the timer is reset. The case of wanting to close the door through the app is similar, but the time out is short. The ESP will set a pin high when it gets the door close command from the app, after continuously reading it high for about a half a second it will also move to the closing state.

There is also the case of the door being closed manually in this state. If this is the case there will be slack in the wire, so if we try to go into opening state, the capacitor will just be discharged. To prevent this, there is a flag called “manualCloseFlag” if the contact sensor gets closed while in open state, this will then bypass the opening state when the door is opened once again, from the closed state. This whole interaction is in green in the figure above.

Lastly there is the closing state, this is where the motors are sending a PWM signal to close the door. Once the door is closed, indicated by the door contact sensor, we will move into the closed state, and we can stop sending the signal to the motor to close the door. The door sensor does go off when it is near the door, not fully shut so there is a small counter to delay when to stop moving the door and go into closed state. This way the door is firmly shut. An added safety feature is that if the motion detector goes off or a hand gets detected the state will be immediately switched to open. This will stop the door closing and once again wait for there to be no motion and hands detected.

5.6 Communications Protocols

5.6.1 Amazon Web Services (AWS) Solutions

An Internet of Things (IoT) is a system where multiple objects are connected for the purpose of transmitting and receiving data. Devices in this network are connected, and are often self-aware, collecting data through sensors in order to communicate through this network [27].

AWS is our preferred cloud service to implement this IoT network, as it is low-cost for our design proof of concept and can be scaled to a production-level product, if this product was to go to market. Within AWS, there are several cloud services that are suitable for our needs. First, AWS IoT uses an MQTT protocol to transport messages over a topic, where "things" in the network can subscribe and publish to these topics [30]. Additionally, AWS houses Lambda functions, where functions can be executed based on IoT messages that arrive, and other triggers within the suite of AWS products [35]. Finally, AWS S3 is a database storage solution where we can upload images and have them accessed by other AWS functions [34].

In order to complete our design, we were primarily interested in how the user could communicate to the cupboard closer and view the system state at any point. To satisfy the requirements 3.2.1.2, 3.2.2.1, 3.2.2.4, 3.2.2.5 the design of any IoT message had to be carefully selected. In order to differentiate between a hand detection, a timeout change, a close command, and a new Wi-Fi status, the messages are constructed with a "type" and a "payload." Because MQTT transmits data in a JSON format, these key-value pairs needed to be constructed in a clear way, in order to parse the messages on the ESP32-Cam and react appropriately. Additionally, we are interested in exporting the image off of the device, and we will follow the same protocol of MQTT messaging to communicate the image to the companion app.

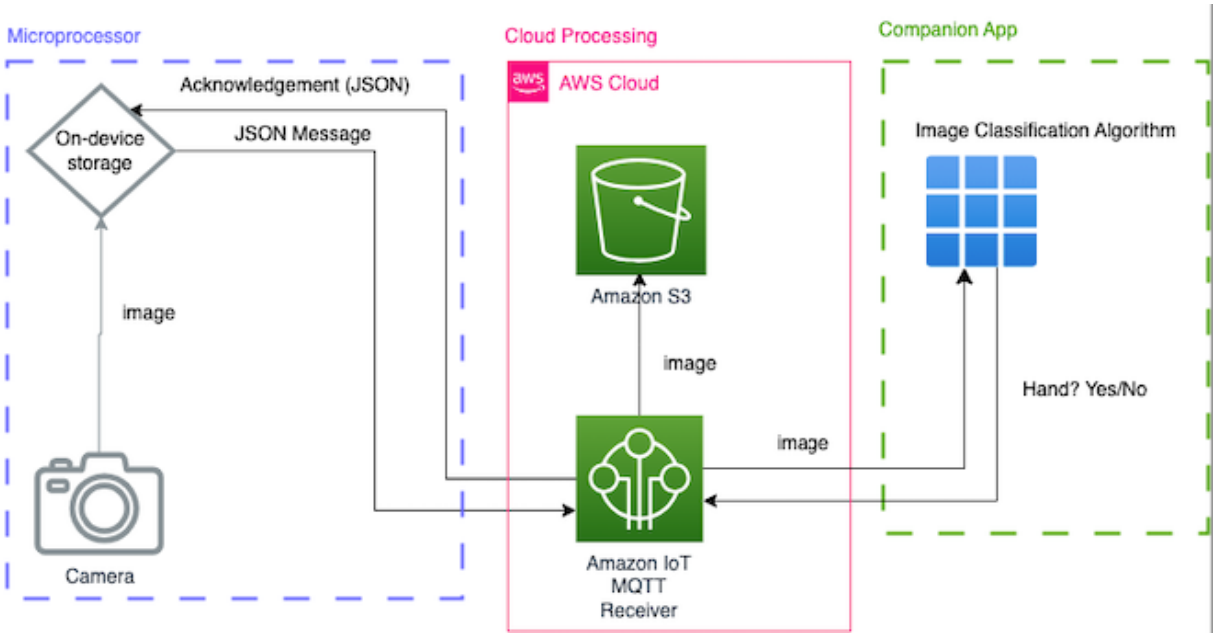

Figure 12: AWS Functionality Overview

5.6.2 AWS IoT Device Creation

As it is determined that AWS is the solution for our communications needs, we have decided to uniquely create topics such that the IoT protocols we implement in the device can be expanded to a fleet of smart-home devices, if necessary.

In order to create an AWS connection, a device needs to be registered within a private AWS account. This is accomplished within the IoT console, where certificates to ensure device privacy are created. These then need to be saved within the flashing code, or onto the device directly. In addition to the secure certificates that allow a device to connect to the IoT network within AWS, rules need to be created in order to allow the device to connect to the AWS account's IoT services. Once the device is granted permission to access IoT within AWS, the packages to connect the ESP32-Cam to AWS need to be installed within the Arduino IDE [74]. After ensuring that the device could connect to AWS and was granted access to the IoT functionality, it was important that the solution to the design problem connected to the cloud in a secure, private manner.

This is partly accomplished through the certificates generated by AWS, and the security within the MQTT protocol, but in order to add an additional layer of security, as outlined by the design constraint in section 4.3, a unique topic was created for each device using it's MAC address. AWS uses topics as a medium in which to publish messages on, such that any device within the AWS IoT network that is listening to that topic can receive the message, and any device that is allowed to publish to that topic may do so.

In our design, we used the ESP32-Cam's unique MAC address to create a publish and receive channel. The publish channel is used to transmit images off of the camera device, and the receive channel is used to communicate from the companion app to the cupboard closer device. It was important to create two channels, a publisher and a listener, based within the ESP32-Cam, because the camera takes actions based on the images it receives. If the camera is constantly publishing images to the same topic it listens to, then the device would become overwhelmed with a recursive publishing scheme, thus a two-channel system was necessary. The figures in the following section illustrate this MAC address schema for the receiving channels.

5.6.3 AWS IoT Messages

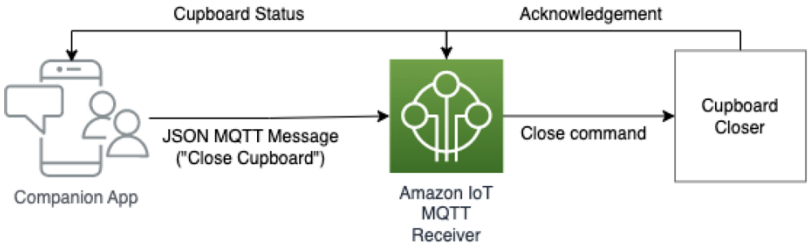

Figure 13: Cupboard close request generated from app by user.

As mentioned in the previous section, the body of the AWS messages needed to be created such that the ESP32-Cam could react to the messages appropriately.

1. *Timeout Message*

The figure below shows the structure of a timeout command.

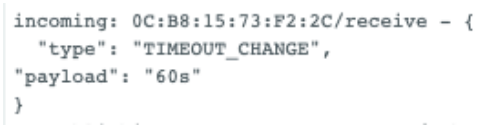

Figure 14: Timeout Message

Once this message is received, the pin responsible for setting this timeout is set to 1 if the timeout is 60s, and 0 if the timeout is 30s. This will then set the timeout accordingly on the ATMega microcontroller.

2. *Hand Detected Message*

The figure below shows the structure of a timeout command.

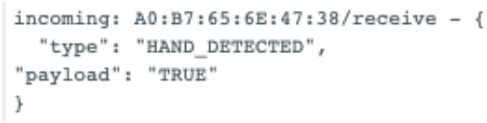

Figure 15: Hand Detected Message

Once this message is received, the pin responsible for communicating this result to the ATMega will be set low if the hand is detected, which will reset the timer on the ATMega.

3. *Close Command Message*

The figure below shows the structure of the close command.

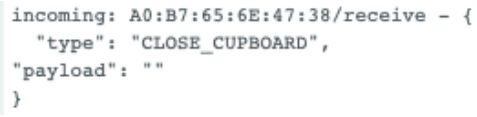

Figure 16: Close Command

Once this message is received, the pin responsible for communicating this command to the ATMega will be set high, and the cupboard will start to close. This pin will remain high until the door is closed.

4. *Image Message*

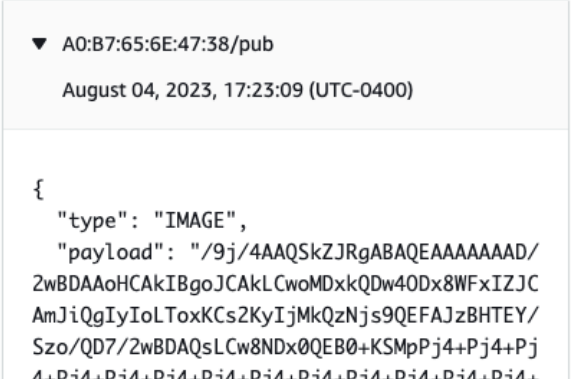

Figure 17: Image Message

The message that exports an image has base64-encoded data from the image capture within its "payload," where it is then received by the companion app, to then be processed and determine if a hand is present. A full image of that base64-encoded data will not be provided like the messages above, as it would exceed an image length appropriate for this report. However, the topic in which this image is published would be named after the ESP32-Cam's MAC address, plus a concatenated "/pub" at the end of the topic, as discussed in the previous section.

5.6.4 AWS IoT Lambda and S3 Storage

As outlined by the previous sections, the IoT functionality was crucial to ensuring that images could be correctly transmitted off of the device and received by the companion app to classify an image correctly. Once this functionality was complete, it was deemed necessary to hold onto the images for training the image classification algorithm, and to ensure that the camera was working correctly. This was accomplished using AWS Lambda, a function-focused service that executes whenever triggers are inputted into the AWS account, like an S3 upload, or an IoT message that contained an image. The AWS Lambda function was created to receive a trigger based on an image that had arrived, convert that to a JPEG, and then save that file to S3.

In order to implement this functionality, a rule within AWS IoT had to be created, where the message "type" needed to be "IMAGE," as illustrated in **Error! Reference source not found.**. The rule then triggers the Lambda function to only run when this type of message is encountered, and tacks on a timestamp in epoch seconds, to ensure that the images are saved uniquely. This rule is shown in the figure below:

```
SQL statement
SELECT *, timestamp() as received
FROM '0C:B8:15:73:F2:2C/pub' WHERE
type='IMAGE'
```

Figure 18: AWS Lambda Rule

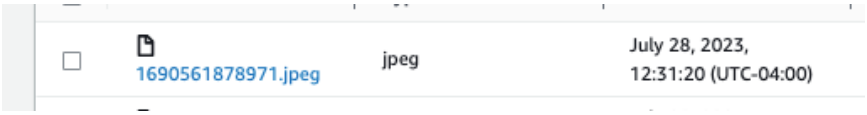

Figure 19: AWS S3 Image Storage

It is important to emphasize the design constraints outlined in section 4.3, where these images that are stored for testing and development purposes are likely not needed, and will be deleted after 7 days, in the product's current state. Additionally, this functionality is displayed in Figure 12: AWS Functionality Overview.

5.6.5 Cupboard Closer Software Development

It is important to emphasize that in order to create a working cupboard closer, the topics in which the cupboard closer communicates with the app would have to be created within the manufacturer, as the MAC address is used, and needs to remain secure. In addition, the only way

that the user can interact with the ESP itself is by changing the Wi-Fi, timeout, and closing the cupboard. The user is restricted for good reason, and the privacy measures are abstracted away from the user as a precaution.

5.7 ESP32-Cam Logic

In addition to creating the communications protocols, it became necessary that a finite state machine needed to be created within the ESP32-Cam, in order to interface with the ATMega, and receive AWS messages.

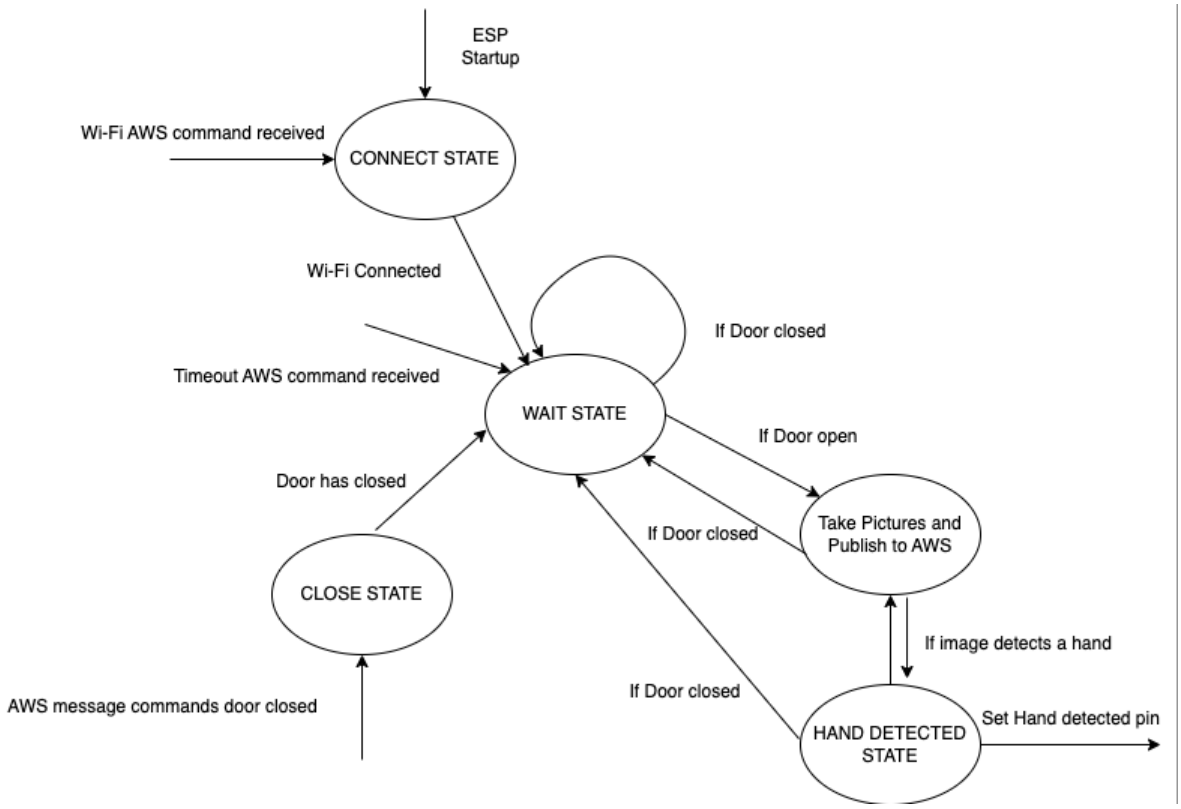

Figure 20: ESP32-Cam Logic

The figure above outlines the logic created for the ESP32-Cam, where the “wait state” occurs after the device is initialized and connected to the Internet, and the door is not open. Once the door is open, we transition into the “publish” state, where the camera constantly takes pictures and sends them off to the companion app to be analyzed. Then, if the result is a hand, the respective pin gets set low, and the ATMega then resets the timer. This state will always return to the “publish” state. If at any point, we receive a close command and the door is open, we will trigger the ATMega to close the door, and remain in that state until the contact sensor determines that the door is closed. Additionally, if the door is closed at any point, the “wait” state is entered again.

5.8 Companion App Design

5.8.1 Companion App Components

The companion app's primary focus for this design was to be the interface in which a user interacted with the cupboard closer device. Specifically, we wanted to ensure that the timeout command, the Wi-Fi change, the close cupboard command, and the image processing were all working correctly in order to demonstrate a working prototype.

A home page was created to house any future IoT devices that may be added to the network, as displayed in the figure below.

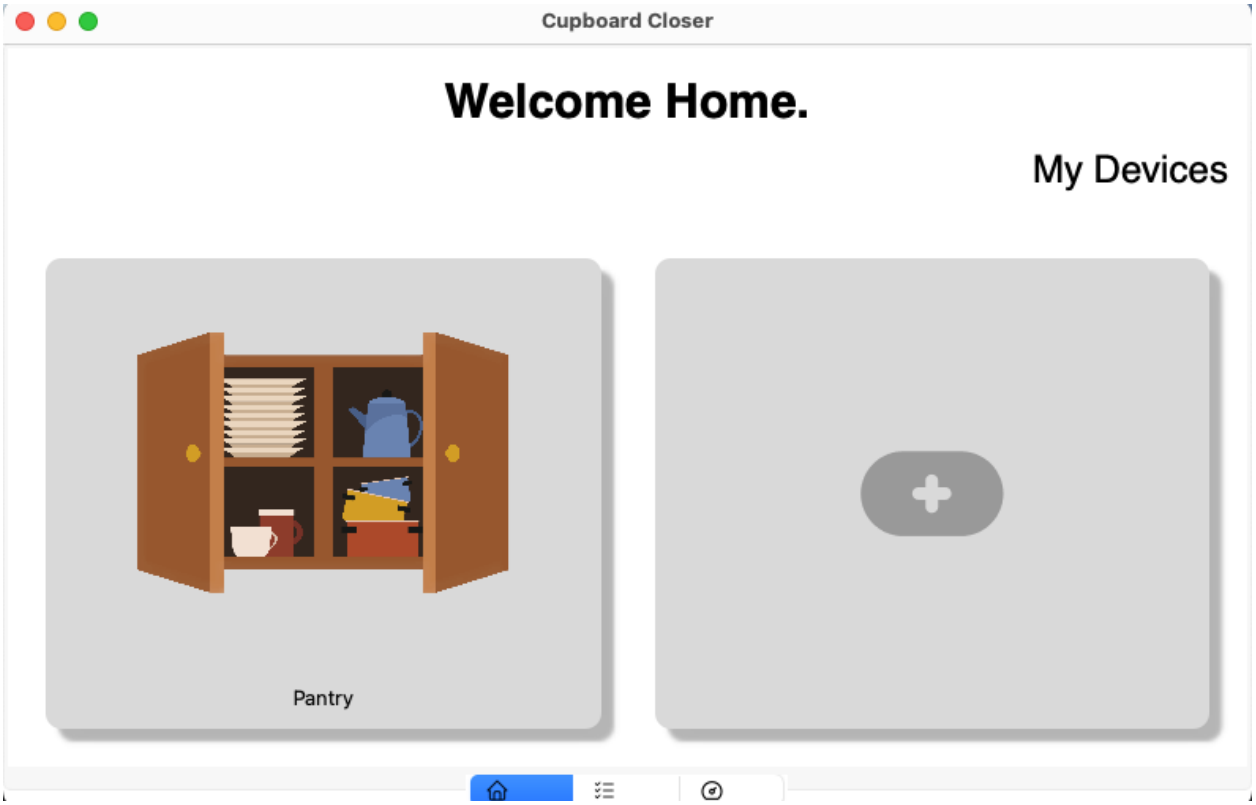

Figure 21: Companion App Home Page

The plus icon will create more devices to add to the local network, if necessary. Because we were only focused on one prototype and limited in our ability to create additional IoT "things," this functionality would need to be refined if created for production.

Figure 22: Cupboard Closer Interface

Once the pantry is clicked on the home screen, the screen above is opened. This displays the current status of the cupboard, where the motion and hand status of the cupboard are displayed by the icons. Additionally, if the cupboard is open, the “close” command button will be active.

Figure 23: Companion App Settings

In addition to the main cupboard closer interface, the settings screen shown above contains developer settings to ensure that the ESP32-Cam was receiving commands correctly. Here, a custom message to change the Wi-Fi can be sent, a timeout can be changed, and the name of the device can be changed throughout the companion app. The message functionality would need to be abstracted away from the user, but for testing and prototyping purposes, the functionality works correctly.

The final functionality of the companion app is a live video of the images as seen by the ESP32-Cam. This is demonstrated at the following link: https://www.youtube.com/shorts/9qUVGNG4FbY?feature=share

If this was to be created and put to market, the image stream would be removed, but it was useful to demonstrate the functionality of the ESP32-Cam and ensure that the device was working correctly as we completed integration.

5.8.2 Hooked Model – Future Work

In order to create a companion app that would be the most beneficial in supplementing our cupboard closer, it was important to think of the user first, before the required functionality of the device. In this case, the Hooked model was followed in order to generate healthy habits from our users and ensure that the user would be willing to change their behavior, and not rely on the cupboard closer 100% of the time. Within this model are four main components: Trigger, Action, Investment, and Variable Reward [75].

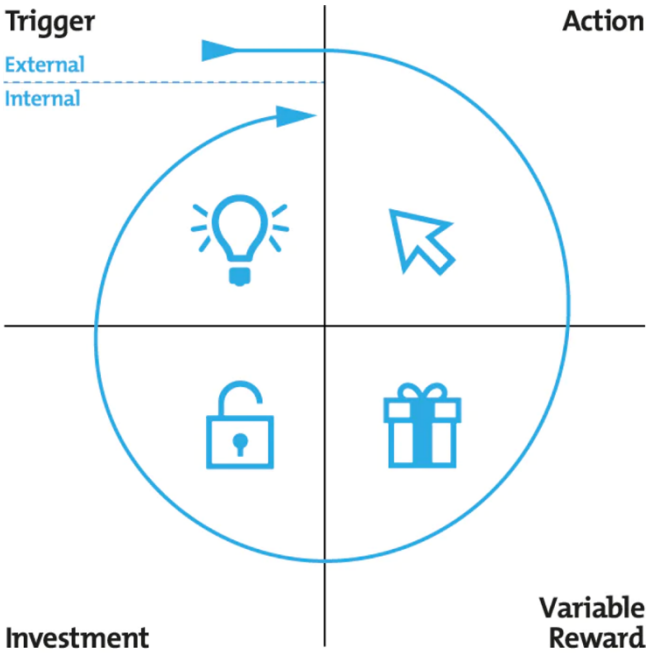

Figure 24: Hooked Model

The trigger hooks the user into the app- this can be an email, a push notification, a text, etc. This component is not implemented in our app, due to the nature of a python executable, and the safety concern of storing user information in an open-source prototype.

The action is the behavior done in anticipation of a reward. In this case, the user would be opening the app, noticing that the cupboard is open, and pressing the “close” button on the main UI. This would then automatically close the door.

Investment is where we ask our users to put in effort in order to improve their experience. In this case, if the user completes the cupboard closing action, the variable reward will change.

The variable reward is then a progress bar to share metrics throughout the week of operating a cupboard closer. This would display how often the cupboard door was closed manually, and how often it was closed automatically.

In order to ensure integration was to be completed within the 12-week time frame, features like the trigger and variable reward were put on hold, and would be implemented if this product was put on the market.

5.9 Power Regeneration and Opening Mechanics

The big idea is rather straightforward – when you manually rotate the shaft of a DC motor you generate a voltage across its leads. But what is true everywhere else is true here as well; the devil is in the details. Recall from our CDD that:

$$V_0 = K_e \times \omega \text{ [42]}$$

K_e is constant; a property of the motor itself. We determined early on that, for our motor choice, K_e is ~1.55 mV/rpm. When you slap a set of new batteries into the system, the power bus hovers around 8V. So, in order to discharge into that bus, we need to exceed that voltage (8V) on the capacitor – and to get that much potential on the capacitor, we need to exceed *that* voltage coming off of the motor.

Since K_e is fixed, this problem essentially boils down to rotating the shaft at a speed sufficient to generate the required voltage. If you take a glance at the testing section of this document, this might seem to be a rather trivial task – the final design could routinely produce 20-30V per open. But these values are *unloaded*. When the supercapacitor is inserted into the regeneration circuit, things become much more difficult. The motor shaft is small, and the torque required to rotate it is considerable. Additionally, the body of the motor itself is subject to various forces and stresses as the door opens – how do we minimize these? We also needed to ensure that the cable wrapped around the shaft traveled a reasonable amount of distance over the course of an open. In spite of all these challenges, we were able to make it happen. Here is how it was done:

5.9.1 Shaft Build-Up

Our DC motor has an 84:1 gear reduction ratio; for every external rotation, the internal shaft rotates 84 times. This ratio ensures that we can theoretically reach the needed voltages, but it comes at a steep price: it is prohibitively difficult to manually rotate the shaft. The cupboard door was nearly impossible to open given the shaft's native radius. During the few times that we did manage to get the door to open, the cable would inevitably snap. We needed to increase the circumference of the shaft by a large margin – this would increase the length of the moment arm, giving us more leverage, which in turn reduced the force required to spin the shaft. There were several iterations here, but the following is the final design:

Step 1: Take a metal bobbin (a small spool intended to be used with sewing machines – before buying these, we compared the diameter of their channel to the diameter of our motor's shaft – they were identical) and punch a hole through its central channel. This is demonstrated in the image below:

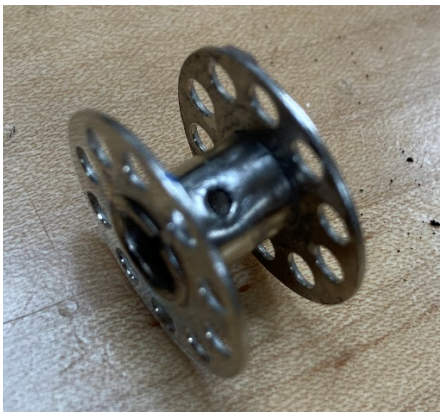

Figure 25: Bobbin with Hole Punched

Step 2: In order to increase the diameter of the shaft to an even greater extent, we superglued a set of steel washers to either end of the bobbin, and then proceeded to superglue a set of cardboard disks onto the *washers* – it might be objected that glue and cardboard are structurally weak, but the forces they undergo are minimal; their primary purpose is to keep the spooled cable in place.

Figure 26: Bobbin with Washers and Cardboard Disk

Step 3: At this stage we were ready to slide the shaft of the motor into the bobbin's channel. Once this was done, we needed to couple the two – we wanted to guarantee that a rotation of the bobbin assembly would produce a rotation of the motor shaft. This was done with a tension pin – a small, rolled sheet of steel that perfectly matched the diameter of the slot drilled through the shaft. Once the bobbin assembly was in place, we squeezed the pin with a pair of pliers and tapped it into one side of the punched hole, through the slot in the shaft, and then out the *other* side of the punched hole. The tension pin, now in place and un-squeezed, expanded to fit snuggly in the shaft – the bobbin assembly and motor shaft were now securely coupled. A great deal of blue painter's tape was then laboriously wrapped around the shaft, building up its diameter.

Figure 27: Tension Pin

Figure 28: The Completed Motor Shaft

5.9.2 Motor Mount and Peripheral Structural Elements

With the shaft built-up we were free to move on the remainder of the structural challenges. Securing the motor to the wall was a rather complex task – a proper mount had to ensure that the body of the motor did not rotate along any of its axis. This was done with several custom-cut wood elements, two worm-gear hose clamps, a bit of sandpaper, and a handful of nuts, bolts, and screws. Here a picture really is worth a thousand words:

Figure 29: Motor Mount and Guide

Note: also pictured (the block of wood in the upper left-hand corner) is the wire guide – this is designed to mitigate an issue we ran into while testing. In the event that someone closes the cupboard manually, the motors do not run, and the wire does not spool around the shaft. This results in a quantity of slack in the wire. When the door is then subsequently opened, the wire would occasionally get wound up below the bobbin assembly's lower disk. To prevent this from happening we introduced the wire guide, which guarantees that the angle between the wire and the spool is always square, even if there happens to be slack in the line.

One final consideration: it became apparent early on that a single open would not be sufficient to charge the capacitor to our target voltage. We modified our circuit design to accommodate charging it over the course of several opens (the capacitor can hold charge in the *closed* and *open* states, it accrues charge during the *opening* state, and it will only discharge in the *closing* state if its voltage exceeds the voltage on the power bus), but the *force* required to charge it – the force required to open the door – was still more than we were comfortable with. In order to do a given amount of work while minimizing the force required to do it, you must maximize the distance that that force is applied over. Accordingly, to keep the charging behavior of the capacitor constant, while minimizing the force needed to open the door, we needed to increase the travel distance of the wire. This was done with the addition of a pulley, mounted to the ceiling of the cabinet opposite the motor. The diagrams below indicate their relative positions and demonstrate how the pulley's addition increased our overall travel distance. 'M' is the motor, 'P' is the pulley, 'd_i' is the length of the wire when the cabinet is closed, and 'd_f' is the length of the wire when the cabinet is opened to ninety degrees.

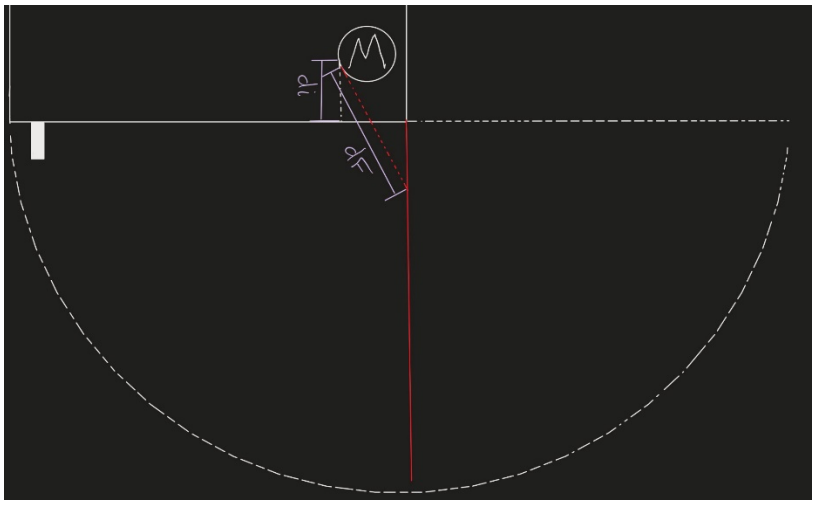

Figure 30: Cabinet without Pulley

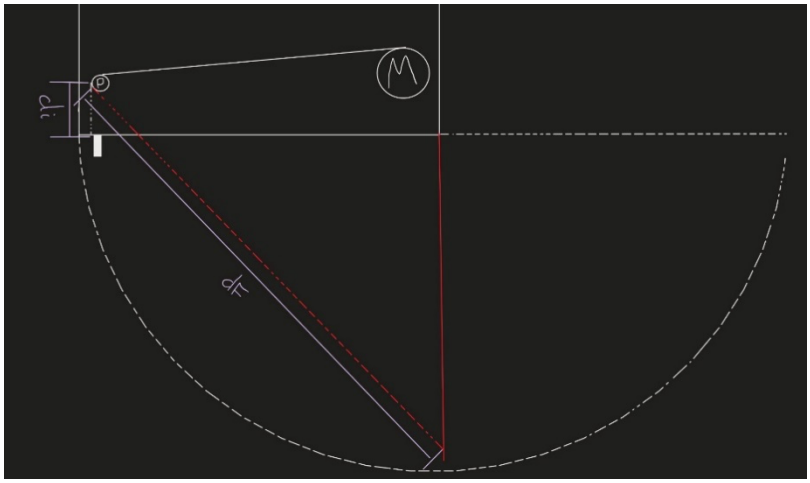

Figure 31: Cabinet with Pulley

6. Team and Timeline

6.1 Team

The team is made up of two electrical engineering students (Brian Danielson, Matt Spering), and two computer engineering students (Madeline Love, Sierra Gordon). The project combines elements of both electrical and computer engineering principles, and each team member has been assigned a domain of the project to lead. Brian Danielson led the power regeneration design, Matt Spering led the motor drive design, Sierra Gordon led the computer vision and image classification algorithms, and Madeline Love led the communications design.

6.1.1 Brian Danielson

Brian designed the power regeneration system. This consisted of two primary tasks – the translation of mechanical power into electrical power (generating it as well as storing it), and the delivery of that power back onto the main power bus. For the former, the design work was largely

electro-mechanical (mounting the DC motor and its peripherals, building up the shaft of the motor), and for latter it was mostly traditional circuit design. Brian also worked closely with Sierra to design the cupboard's finite state machine and realize its physical implementation (the choice of contact sensor, and the addition of a linear regulator to his regeneration circuit (this regulator signals to the MCU the transition from *opening* to *open*)).

6.1.2 Sierra Gordon

Sierra worked on the computer vision aspect of the cupboard closer. She collected images with the help of Maddie's ESP web camera server setup, to then create a model to detect a hand within the cabinet. She worked with the microprocessor to write a finite state machine that would determine when to close the door and what to set the relays to for the motor and power regeneration. Brian and Matt stated what the relays should be, and Sierra wrote it. Sierra worked with the ESP when initially setting it up and connecting it with the microcontroller. The ESP was primarily a part of Maddie's role, but Sierra assisted when needed.

6.1.3 Madeline Love

Maddie was responsible for the communications protocols between the microprocessor and AWS. She used an ESP32-Cam to create a web server to help Sierra train her algorithm and worked to ensure that the ESP could house an active AWS MQTT listener and take and publish pictures. Additionally, she created a companion app to visualize the state of the cupboard closer and researched healthy habit-forming habits in order to understand the user of this device, and tailor an interface for that user. She also created the companion app to house Sierra's algorithm, and successfully channeled the paths between an image being taken, a message to publish, and a receiver to classify the image on the companion app, in addition to the other IoT message requirements.

6.1.4 Matt Spering

Matt designed the motor drive system as his primary module. He also contributed to the switching array design for regeneration and aided as a secondary set of hands during mechanical implementations with Brian. He completed much of the circuitry implementation on breadboard and designed the PCB, though it was replaced in the final iteration of the prototype as detailed in the design section and further elaborated upon in the conclusion.

6.2 Timeline

WEEK	TASKS COMPLETED
1/2	Teams formed, design plan is discussed in general, team members are assigned modules and domains for the project. Initial research has begun.
3	Conceptual Design Document due. First order placed (6/2), contains Wi-Fi transceiver, camera, motors, motion sensor, buzzer, diodes, capacitors, batteries. Continued research.

4	Sierra – Spoke to Dr. Dallal for the second time, created an outline of what needed to be done and started trying to create models. Maddie – Completed end to end AWS communications, which included creating a message from a Python script, receiving it in the cloud, and triggering an AWS Lambda function. She also completed wireframes. Matt – Completed motor drive implementation on breadboard and began testing. Brian – Complete first iteration of regen circuit design, implemented on breadboard.
5	Continue to work on what we were doing the previous week along with preparing for the check off. Sierra – created test algorithms with a streamlined process. Setup how to preprocess images, train an algorithm, and test with separate images. Maddie – Refined the AWS communications algorithm, tried to work with the OV7670, and was unsuccessful. Matt – Doubled the motor drive implementation to power two motors simultaneously, completed testing of motor drives while unloaded. Brian – began work on motor shaft expansion.
6	Sierra – Tried to set up camera. Made new models to try and make better versions. Maddie – Continued working with the camera with Sierra, was unsuccessful. Created initial iOS functionality with SwiftUI. Matt – Helped trying to set up camera; Completed crude mounting of motors on cabinet and closed the cabinet for the first time. Brian – Continued work on motor shaft, began work on more robust motor mount.
7	All – midterm presentations. Sierra – got cabinet. Continued to try and get the camera to work, and eventually ordered a new one (ESP32-CAM). Tested connecting the microcontroller to door contact sensors and motion sensors. Maddie – Worked with Sierra to get images off of the ESP and connect to Wi-Fi. Matt – Helped trying to set up camera; Began Altium schematic design; Aided in first set of mounting iterations within the cabinet. Brian – Continued iterating expanded motor shaft and improved mount, added det_relay to regeneration circuit.
8	Sierra – Worked on verifying how to connect microcontroller with ESP32, motion sensor and contact sensor. Determined how to get images off the camera. Maddie – Worked on connecting to Wi-Fi with the ESP and pulling images from the camera. Matt – Finished PCB and submitted order together with Group 3; Continued aiding Brian in mechanical implementation. Brian – Finalized motor shaft expansion.
9	Sierra – Gather more images with camera, train new algorithms with the images and test accuracy. The model is also given to Maddie to integrate into her app. Continued microcontroller testing. Maddie – Decided to make the app a desktop app as opposed to a mobile app. Pivot from classifying the model in AWS to in the desktop app due to AWS limitations. Created AWS Lambda functions to receive the base64-encoded image, according to timestamp.

	Matt – Worked with Brian to make final iterations on mounting and closing mechanisms to allow for integration of regeneration and drive system; Initial integration of relay and drive system with MCU. Brian – Worked with Matt to begin integrating motor drive circuit with regen circuit. Continued mounting work. Worked with Sierra on FSM.
10	All – check off #2 Sierra – breadboard integration with microcontroller and motor tested along with relays. Also started ESP32 with AWS integration setup. Maddie – Improved the app. Finalized the logic for the ESP32's code. Matt – Put together the PCB with help from SERC. Brian – Finalized mounting system, troubleshooting all subsystems.
11	All – fully integrate all modules and test. Sierra – Tested integration between microcontroller and Maddie's code on the ESP32, were then able to test the timing and quality of hand detection. Maddie – Test ESP logic with microcontroller and gather message timing. Put final additions on the app. Matt – Attempted to integrate PCB with software and regeneration; Pivoted back to breadboard design to fully integrate and discovered relay and switching problems; Fixed switching problems using MOSFET array; Condensed breadboard to final implementation; Conducted testing on drive system and battery. Brian – Switched to MOSFETS on circuit, completed integration, completed testing.
12	All – Final presentations and report due

Table 3: Team Timeline

7. System Test and Verification

To thoroughly test the system, the team will be analyzing the accuracy of the image detection algorithm, communications latency, power efficiency, and motor efficiency.

7.1 Software Systems

7.1.1 Image Classification Testing

The quantitative testing is from the testing python script. As described previously it classifies a set of new images and see if it classified it correctly. The model used was tested on 766 images and it got 621 of them correct, which equates to 81% accuracy. The figure below is a screenshot of the terminal's output after classifying it. According to (3.5.1.1) we wanted accuracy to be above 75%; it is so the benchmark was surpassed.

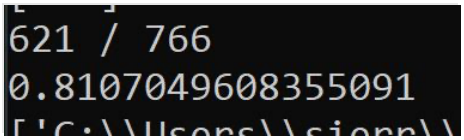

Figure 32: Terminal Displaying Model Accuracy

Once the system was set up and connected on the breadboard the model could be evaluated from a qualitative perspective. One observation is the model cannot detect hands that are too far away from the camera, the model can recognize the hand if it takes up most of the image. It is also important that the space is well lit when the flash is not on it will often have false positive and false negatives. The orientation of the hand is also important, if your hand is upside-down, relative to the camera, it will not recognize it. Both observations make sense for the model to struggle with since the training images all had the same orientation and relative distance of being close to the camera.

The other testing done for the classification was the speed. Of thirty-five trials the average speed of classifying the image was 0.124 seconds. In requirement 3.5.1.2 we wanted the classification to not take longer than 0.5 seconds, which is does, by far. This time includes the preprocessing step or resizing and doing the Histogram of Gradients on it, which based on experience tends to take a fair amount of time and is what takes up most of those 0.124 seconds.

Overall, the model's hand detection is more than adequate for our purposes and more than quick enough. It will not detect every frame of the hand, especially the blurry ones, but it does seem to detect every other frame of the hand at least, so the microcontroller will still know before the time out that there is a hand which is what really matters. More importantly, we rarely see it detect a hand when it is not there. In some cases, a face, wire, or a pair of pliers will set off the hand detection, but it does not go off randomly while looking at the ceiling or cabinet. That is important because if that were not the case, the door would never close. It is better to have a false negative than a false positive. Either way the model seems to perform about as well as we were hoping it would.

7.1.1.1 Microcontroller Testing

The microcontroller made the decisions on when to close the door, given the hand classification, motion sensor, and door input. There is not really a good qualitative testing metric for the microcontroller, but I did want to mention how I went about testing it. The testing was piece by piece, setting up and testing each component's connection. After setting up model inputs the finite state machine was tested to make sure it moved through the states correctly as it received inputs. Then tested connecting it to the motor, relays, and the ESP. We evaluated that everything was happening as expected in the case of the motors and relays. With the ESP we were verifying the signals being sent and received.

There really are not any quantitative tests to do on the microcontroller, it is more qualitative. Whenever adding a new component or connection I would check the microcontroller logic by changing the input and seeing what is does in response. A good example is when adding the logic to make the door halt closing and go back to open when there is motion or a hand detected, the red arrow in figure 8. When I uploaded the first set of code I waited until the FSM went to state three, closing, then connected the hand detection pin low, meaning a hand is detected. I expected it would go to state two, but it stayed at state three. Looking at my logic I realized it was an issue

with the order of my if statements. With the change, when the hand pin was tied low or the motion sensor went high the state switched, and the motor stopped. This process of making a change and watching the effect is how we tested the microcontrollers logic.

7.1.2 Communications Testing

The primary interest of the communications testing was to make sure that the timing for an image to be taken, processed within the companion app, and a result returned, was less than 30 seconds. As mentioned in the design, we were concerned with how long that large image would take, so to test the timing of AWS, we added timestamps within the ESP32-Cam code, and measured the epoch second difference between when it was sent, and when it was received by the companion app.

On average, over 50 messages, the average time to send and receive an empty AWS IoT message was 0.512 seconds. For an image, the results are displayed in the table below:

AWS Image Message	
Trials	**Time (seconds)**
65	0.733242688
76	0.626590061
65	0.794357318
100	1.304221478

Table 4: AWS Timing with an Image

Over 4 trials, with over 65 images sent each, the timing averaged out at about 0.7 seconds, but the fourth trial with 100 images likely increased in average timing because the ESP32-Cam became overheated. It is unlikely that the cupboard would take 100 images while open for 30 seconds, but in case it does, and the average goes up, we are still confident that the 30 second timeout is still a safe time, as 1.3 seconds does not come close to 30 seconds.

Additionally, we were interested in the speed of the camera itself, not just the AWS timing, so we ran 21 trials of leaving the cupboard open, placing a hand over the ESP32-Cam, and recording how much time it took to be displayed on the companion app. The results are listed in the table below.

Trial	Time (s)	Normalized Time (s)
1	6.24	6.24
2	6.42	6.42
3	17.46	8.73
4	8.73	5.7
5	11.19	4.35
6	16.32	3.56
7	5.7	4.04
8	4.35	3.83
9	3.56	4.53
10	4.04	4.91
11	3.83	3.91
12	4.53	3.89
13	4.91	4.48
14	3.91	4.25
15	3.89	4.02
16	4.48	3.59
17	4.25	4.75
18	4.02	4.48
19	3.59	*4.76*
20	4.75	
21	4.48	

6.221428571

Table 5: Time to Display Image on Companion App

The normalized time column gets rid of the outlier data, as we believe that the ESP32-Cam was operating under a stressful day of testing. Additionally, it seemed that the ESP32-Cam performed worse as the amount of activity in front of the camera increased. It behaved almost like the camera was focusing, ensuring that it captured a quality image, but once we slowed down our testing and ensured that the hand was the only thing in frame, the timing seemed to regulate itself. With the outliers removed, the average time it took to display a hand on the companion app was 4.76 seconds. Although this is a significant increase from the AWS timing, we still felt comfortable setting our default timeout to 30 seconds, as we believed that that gave us enough of a buffer to act in a safe manner.

7.1.3 Communications Testing - Note

It is important to emphasize the protocols in which AWS uses MQTT to communicate images. The protocol will keep retrying the message until it is confirmed to be delivered, so if a message is sent successfully, it is sent with 100% accuracy, if not, it is not delivered. This criterion

is not tested, as the cupboard closer is interested in the timing of a classification for safety reasons, not necessarily the quality of the image, nor if it arrived correctly.

7.1.4 Security Testing

As outlined in section 4.3, it was important that the system be designed with network security and privacy in mind, which is hard to test without breaking the law. In developing the prototype, it was necessary to include certifications onto the device in order to use AWS IoT functionality, and it was necessary to use MQTT, a secure network protocol, as that is the default standard within AWS IoT. The system was not able to work without these certifications, and it would be highly unlikely that the team could break this MQTT security, without hiring consultants or staying within the confines of the law. While we would have liked to explore this secure functionality, the team can confidently say that the individual AWS topics add another layer of security, and the certificates are necessary to ensure the device can connect safely and securely.

7.2 Hardware Testing

7.2.1 Door Closing Timing Testing

In addition to ensuring that the end-to-end image paths were not too long to infringe on the 30 second timeout, the door closing time was tested to ensure that the user experience would not be devalued. The results of this testing are in the table below:

Trial	Time (s)
1	18.03
2	18.25
3	17.4
4	18.99
5	17.6
6	17.17
7	18.18
8	18.37
9	17.77
10	18.27
Avg.	***18.003***

Table 6: Door Closing Timing

It was determined that for a 7-second set timeout, that the door would close around 18 seconds. For that to be extended to a 30-second timeout, the door closing itself would take roughly 11 seconds, which we determined to be an inconsequential value in our user experience. As further detailed in Table below, this value is primarily dependent on the length the door is opened.

7.2.2 Electronics Testing

7.2.2.1 Motor Drive Testing

The motor drive system was tested using an array of four multimeters for measurement and one oscilloscope for signal confirmation of the PWM output. Two of the multimeters measured

voltage: one with probes placed across the battery terminals to measure input power and the other with probes placed across the motor itself. The other two measured current and were placed in series out of the battery and into the motor. The "math" function was utilized to generate a full array of statistics during measurement on "fast" speed, allowing for the measurement of maximum values for both voltage and current. Additionally, the time of cabinet closure was measured using a stopwatch and the length of the open was measured using a ruler for the purpose of calculating RPM and energy consumption. Each cabinet closure length was conducted ten times, and the parameters found averaged as shown in the two tables below. For conciseness, the comprehensive trial data can be found in the appendix section at the end of the document.

The power peak was found using Ohm's law, with peak measurement to determine maximum potential consumption; a potential secondary method of using averages instead of peaks was considered, but the use of "Time On" allowed the maximum potential energy consumption to be calculated for only the "high" state of the PWM signal by dividing the closing time by two. "Time On" was equivalent to the closing time divided by two because the PWM signal operates under conditions of 50% duty cycle. Energy consumption was calculated by converting power consumption to mAh by the equation: $E_{mAh,peak} = 1000 \cdot \frac{P_{peak}}{V_{peak} \cdot t_{on}}$. RPM was found using the measured length of open using the following equation: $RPM = \frac{60 \cdot \ell}{t_{on} \cdot C}$, where "C" is the circumference of the motor shaft. The torque produced by the motor was found by the equation: $\tau = \frac{(V_{peak})^2}{R_{motor} \cdot \omega}$ once the rotational speed was converted to radians per second by multiplying the rotational speed in RPM by the factor $\frac{2\pi}{60}$. This was done easily compared to the process an AC motor would require, as the DC motor operates under purely resistive conditions due to the lack of frequency of the power signal generating inductance in the core.

Unlike power consumption, which was used to calculate the efficiency of the drive system, the energy consumption was found to judge the lifetime of the drive system itself off the battery bank. Recall the bank utilized a series of six AA batteries with maximum potential of 8 V total and capacity 2 Ah. Following the equation, $t_{hours} = \frac{E_{Battery}}{I_A}$, the lifetime of the battery supplying the drive system can be found. In this case, the battery would last approximately ten hours at peak current draw under constant closing conditions, scaling up to just above twelve hours at minimum current draw. Because of the length of operating time, draining the batteries to confirm this calculation was not possible under the course's circumstances. Future testing and iteration would allow for these numbers to be confirmed.

The percentage efficiency was calculated by dividing the power input to the system by the power input to the motor, while the percentage of rated current was calculated by dividing the average current input to motor by 0.2 A, the value listed on the datasheet. The percentages of rated speed and torque were similarly found, dividing the calculated averages by 114 RPM and 150 mN*m, respectively.

Averages for Varying Open Length (Motor Drive):									
Length (in):	Voltage Peak (V)	Current Peak (A)	Power Peak (W)	Time On (s)	Energy (mAh)	Speed (RPM)	Torque (mN*m)	% Efficiency	% Rated Current
12	6.9331	0.087	0.6011	3.086	74.554	46.566	124.064	46.649	43.350
15	6.8814	0.081	0.558	3.989	89.937	44.914	118.826	42.116	40.550
18	6.8586	0.091	0.622	4.904	123.508	43.824	135.515	38.676	45.350
21	6.8873	0.088	0.605	5.861	143.185	42.792	136.498	45.131	43.950
24	6.899	0.194	1.336	6.861	369.158	41.770	305.618	88.410	96.850

Table 7: Motor Drive Testing Averages

Averages for Varying Open Length (Battery):					
Length (in)	Voltage Peak (V)	Current Peak (A)	Power Peak (W)	Time (s)	Energy (mAh)
12	7.975	0.162	1.289	6.713	150.789
15	7.958	0.167	1.325	8.341	192.841
18	7.957	0.202	1.608	10.074	282.749
21	7.952	0.169	1.342	12.053	282.323
24	7.947	0.190	1.512	14.182	374.655

Table 8: Battery Testing Averages

Below in The torque-speed curve generated by the percentage of rated values does align with the ideal case in a DC motor: an ideal DC motor operates under linear torque-speed conditions, as the two parameters are indirectly related by equation $P = \tau \cdot \omega$ holding power constant. The stark drop between 37% and 38% of rated speed (occurring at lengths 21 and 24 inches, respectively) may be due to the same conditions discussed in the previous paragraph.

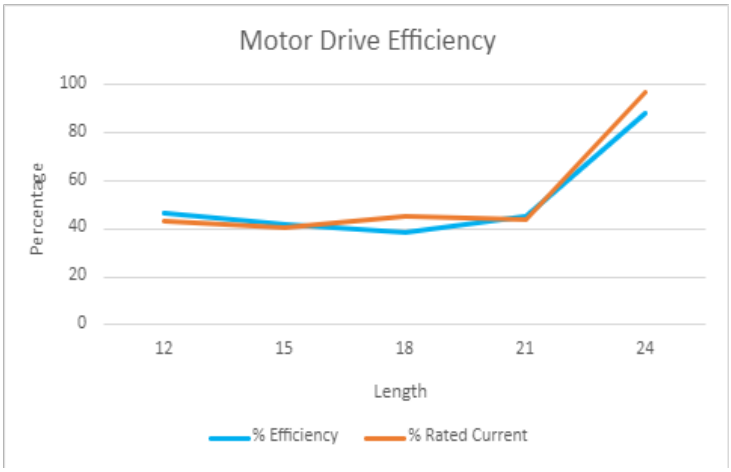

Figure 33 and Figure 34 are plots of the drive system efficiency and torque-speed curve; a notable discovery when tabulating the data for the drive system was the strong correlation between percent rated current and efficiency of the system. This indicates that the system itself loses a substantial amount of power during operation at lower power levels, which does align with motor operation itself but less so with the operation of the drive system.

One reason this finding may have occurred is the loss of power through heat within the switching MOSFETs that control the states, as they were not chosen and ordered for the role like many of the other components, rather obtained and utilized from the supply of ICs within Benedum. While they did serve their primary purpose, a definitive study and replacement of these components with a more robust relay system as originally intended may yield better efficiency at lower levels of power. Another potential reason for decreased efficiency would be the charging of the regeneration capacitor during the "closing" state; a potential consequence of the switching network not performing ideally and allowing leakage. That does not address why efficiency increases at the 24-inch length; this may be due to the power draw of the motors increasing to the point of preventing any leakage by utilizing a large enough percentage of the battery's potential output that most current flows to the path of least resistance. Another set of trials beginning in reverse would also eliminate the possibility that the heated MOSFETs or shunted capacitor saw increased resistance due to system stress over many closes. The addition of a more comprehensive relay network with appropriately chosen current ratings will be discussed within the section dedicated to future considerations, which would allow for testing to determine the cause of the phenomenon through type-II and type-III tests.

The torque-speed curve generated by the percentage of rated values does align with the ideal case in a DC motor: an ideal DC motor operates under linear torque-speed conditions, as the two parameters are indirectly related by equation $P = \tau \cdot \omega$ holding power constant. The stark drop between 37% and 38% of rated speed (occurring at lengths 21 and 24 inches, respectively) may be due to the same conditions discussed in the previous paragraph.

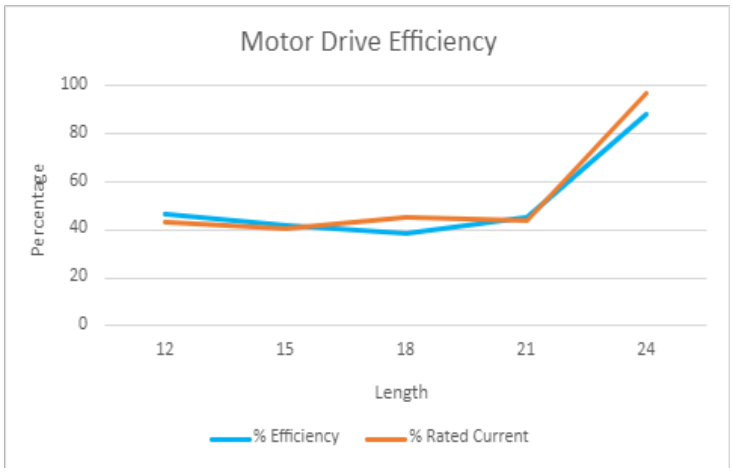

Figure 33: Drive Efficiency

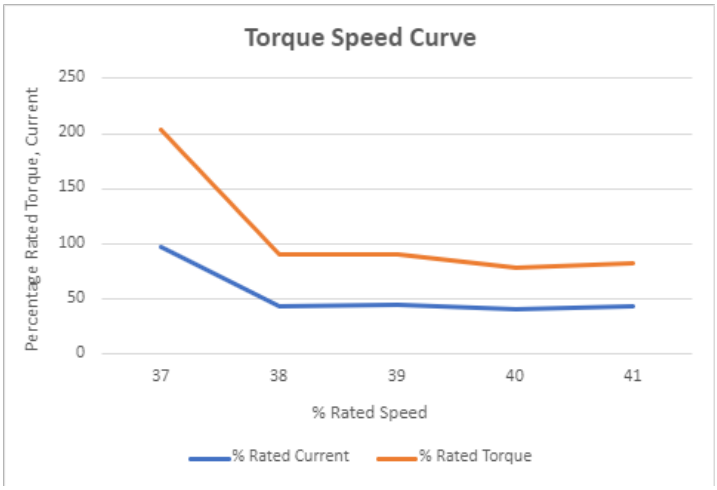

Figure 34: Torque-Speed Curve

7.2.3 Power Regeneration System Testing

Since the power regeneration system is in effect two cooperating sub-systems, I needed to obtain testing results for both. I will first describe the testing method and results of the electro-mechanical design, and then move on to the regeneration circuit.

7.2.3.1 Motor Mounting and Shaft Testing

Recall that output voltage of a generator is given by:

$$V_0 = K_e \times \omega$$

Preliminary testing (done prior to the first checkoff) indicated that the K_e constant of our chosen DC motor was about 1.55 mV/rpm. The test consisted of opening the door twenty times, while keeping track of a particular set of measurements. I obtained the number of rotations per open by measuring the travel of the cable and dividing it by the circumference of the built-up shaft (6.125 inches). I recorded the duration of each opening as well. Using this data, I was able to calculate the peak angular speed of each opening cycle, and then use that value to calculate the maximum theoretical output voltage. I then compared this value to the measured peak output voltage, read off a digital multimeter hooked up across the motor. The results were as follows:

TRIAL #	TRAVEL (in)	DURATION (s)	ω (rpm)	CALC.V. (V)	MEAS.V. (V)	% DIFF. (%)
1	21.00	1.51	124.80	19.34	25.59	32.29
2	20.50	1.26	146.10	22.65	31.02	36.98
3	11.75	0.78	135.30	20.97	29.20	39.24
4	16.40	0.93	158.40	24.55	25.59	4.23

5	20.05	1.71	105.30	16.32	21.50	31.73
6	19.15	2.03	84.70	13.13	17.05	29.87
7	15.20	1.53	89.20	13.83	19.56	41.47
8	11.15	1.15	87.10	13.50	17.79	31.77
9	18.75	2.66	63.30	9.81	15.44	57.37
10	18.05	2.15	75.40	11.69	17.84	52.65
11	12.40	1.35	82.50	12.79	20.25	58.36
12	14.20	1.51	84.50	13.10	18.07	37.97
13	18.90	2.45	69.30	10.74	17.21	60.22
14	11.10	1.16	86.00	13.33	15.09	13.20
15	17.80	1.96	81.60	12.65	16.34	29.19
16	15.80	1.73	82.00	12.71	17.93	41.07
17	14.1	0.95	133.30	20.67	26.61	28.79
18	14	1.53	82.20	12.74	15.73	23.46
19	15.1	2.18	62.20	9.64	14.65	51.96
20	19	2.34	73.00	11.31	15.40	36.10

Table 9: Regeneration Potential Testing

The average percentage difference is **37.38%**.

A few comments: The difference between the measure and calculated voltages is due to a variety of factors. The walls of the cupboard are not perfectly stiff, which results in a certain amount of give. The shaft itself has a bit of play, which steals a few degrees of rotation from every open. The cable, despite being fifty-pound test line, still stretches, and slides several millimeters around the built-up shaft. The body of the motor itself undergoes a small roll torque, or planar rotation. Despite the techniques employed to mitigate these influences, eliminating them totally would be a formidable task.

7.2.3.2 Current Supplied by Supercapacitor Testing

The second sub-system of my design is the regeneration circuit. Once the voltage across the motor leads is generated, it is used to charge the supercapacitor, which in turn "holds" this charge until the cupboard is in a closing state. Once the cupboard is in this state the capacitor discharges into the power bus. While this discharge occurs, less current is drawn from the batteries,

effectively extending their life. I tested this subsystem as follows: I took three digital multimeters and hooked them up (inline) on three separate nodes: (a) directly after the positive terminal of the battery, (b) directly before the positive terminal of the motor, and (c) on the discharge node going from the capacitor into the power bus. I then recorded the measured current readings over twenty closing cycles. All the multimeters were set to sample current in the fast-rate mode, which allowed me to record an accurate average for each trial. The results were as follows:

TRIAL #	*I.* SUPP. BY BATT. (mA)	*I.* SUPP. BY CAP. (mA)	*I.* DRAWN BY MOTOR (mA)
1	141.21	5 .71	91.62
2	150.39	8.36	92.34
3	154.88	5.35	101.78
4	149.04	7.57	83.6
5	147.28	4.63	94.18
6	155.96	4.14	79.99
7	160.72	7.49	106.37
8	158.85	8.51	101.07
9	144.45	6.11	92.54
10	149.2	8.86	82.75
11	150.62	12.22	93.7
12	151.78	8.51	85.63
13	148.36	6.14	82.49
14	161.45	10.85	80.17
15	157.12	5.19	105.15
16	143.74	4.90	77.91
17	149.02	13.28	90.93
18	154.93	10.62	75.68

19	161.26	8.50	118.06
20	152.81	7.02	96.81
AVERAGE	**152.15**	**11.99**	**91.64**

Table 10: Regeneration Current Testing

A few comments: The capacitor discharges straight into power bus, so the current supplied by it is likely being dispersed to the ESP camera and other peripherals in addition to the motor; however, during the closing state, the motor draws the majority of the current, so it was appropriate to measure the input there specifically (in any event, the discharged current simply has nowhere other than the power bus to discharge to – whether by the motor or some other module, it is being used, which reduces the draw on the batteries). With this in mind, we can reasonably conclude that, on average, the percentage-contribution of the capacitor is:

$$\frac{11.99}{91.64} \times 100\% = \mathbf{13.08}\%$$

This shows an increase in the overall lifetime of a set of batteries by more than 10%.

8. New Skills Acquired and Learning Strategies

8.1 Matt Spering

I am in an admittedly different position relative to most students taking this course. Much of my coursework has dealt with power systems and power electronics; however, I had not yet delved into motor drives before taking this course, as those classes are limited to the graduate field of study in ECE at the university. While my drive system itself was quite simple compared with most because of my motor selection, the overall implementation and learning process is scalable to larger motors or motors with different operating conditions such as alternating current. Control through signal generation was not something I had encountered directly prior to this summer, and experimenting with different iterations of the PWM signal controlling my drive system was enlightening from a power consumption perspective specifically. I am more interested in continuing my learning about control systems now than prior to taking the class, especially because of how shallow I needed to go for this prototype in relation to the depth of the field. The knowledge I gained here will be an important building block for my future work in the distribution sector, as relays and control systems are a necessity for management of power laterals going to residential, commercial, and industrial buyers. The primary source for much of this information came from Dr. Grainger and some of his graduate students, many of whom I met before this semester in some capacity. Outside of them, I used datasheets and guides online created by part manufacturers such as Texas Instruments for individual IC usage and device manufacturers such as Eaton for more high-level understanding of the system. Additionally, research conducted for the conceptual design such as sources [16][17][18][19] were used to familiarize or review concepts of motor drive design and control, including drive topology, PWM control, and capacitor choice for energy storage purposes.

I made a major stride on my mentality management this semester with regards to the process of R&D and learning. I have a history of being temperamental and becoming frustrated quickly when things are not going well; while those traits flared up occasionally, the combination of my group's support and strides I made myself prior to and during the summer allowed me to refocus and buckle down to keep grinding away when things were not going as smoothly as the ideal. The last week was especially challenging in this regard, with last-minute changes being made to ensure the prototype was operational by the final presentation. Brian specifically was crucial to helping me recenter and get back to doing what needed to be done. Other sources that helped me through this process include my personal and medical support system I developed during my semester off in the spring and fellow students in class who were gracious enough to provide help when I encountered a problem I had not seen before. This may be a different type of learning than a student may usually report, but it was important to my future as an engineer in industry.

The last learning step I took that pairs with the prior is increasing my troubleshooting knowledge. Electronics troubleshooting can be tedious, but through the manuals for the testing equipment, Brian's aid, and the input from other students in other groups, I was able to hammer down a process of testing my circuits through multimeters and scopes that expanded my knowledge of those devices and how I could apply that knowledge to future projects. Small gains such as discovering ways to measure transients, peaks and valleys of voltage and current, and finding new functions on every piece of equipment I used (power supply, function generator, multimeter, and scope) allowed for faster testing and more insight into what problems were occurring by comparing what I saw to datasheets. Using this information, I made significant strides in the final week to condense and ready the prototype for the demonstration. Overall, I increased my visualization of circuit design by providing numbers and therefore analytical context to what I was doing at the time.

8.2 Sierra Gordon

There were a couple of things I learned, the primary thing was about machine learning and computer vision. While I did not use it, I learned about optical flow for cameras to see motion. I learned about the histogram of gradients and what it is along with the process of how to create a model that detects objects. I had to learn how to save the CVSs, work with images, and save models. I also got more experience with hardware, I had to relearn how to wire a button, and program the Arduino Uno to read and write pins.

I would say the primary thing I had to learn was how to train a model. I had some background knowledge but doing it from scratch was quite different than in class. To figure out how to do everything I first talked to Dr. Dallal. He helped me get a direction and know what I needed to research to accomplish my goal. I also continued to ask him questions the first couple weeks to help me set up the pre and model creation. I used a variety of other sources to understand the concepts - histogram of gradients, neural networks, support vector machines, and optical flow - that I was implementing. In terms of coding, I found websites like geeksforgeeks, w3schools, scikit-image, among others, helpful in learning how to do everything I needed to do.

I initially felt very lost on where to start with how to make the model until I spoke to Dr. Dallal. He helped give me directions, so I could build an outline of what I needed to do and research. Once I had an idea of what I needed to do, I was able to start making it happen. The

knowledge, especially from guides on doing certain things in python, was used to solve the design problem. Things like saving a CSV file, setting up the neural network for training, and saving the model, were all things I did not know how to do. With both Dr. Dallal and the internet as a resource I was able to make my module successful and learn how to do what I needed to do.

8.3 Madeline Love

The biggest thing I took away from the development of this project was my ability to use AWS confidently. I was able to create a custom AWS suite, which suited our needs for communication, data storage, and triggered events, through AWS IoT, S3, and Lambda. I spent quite a bit of time ensuring that the path between these components was correct, because once the camera was operating correctly, I could focus on saving the camera data itself, instead of the paths I needed to refine.

Additionally, in creating rules and triggers for these AWS functions to deploy, I learned quite a bit about security, and why AWS has protocols in place to prevent malicious activity. It was helpful to see how these security rules could scale up to a large manufacturing process and ensure that each device is operating on its own channel, with its own rules. This setup, however, would have to be created for each individual device, and we were limited in our development of a single prototype, but that functionality was definitely visible in my exploration within AWS.

In observing Sierra's image classification algorithm, I was able to learn about how our hands were being classified correctly, and it was interesting to see the iterations she developed and how well they worked within our model.

Finally, I worked on my UX design skills by creating the companion app. I am very interested in front-end and user-driven development, so it was very rewarding to approach this problem with a user, use case, and constraints in mind, in order to create an app designed directly for our users. While I would have liked to implement more features, pivoting to a Python executable set me back, but if this were to go to market, I would love to refine that design.

8.4 Brian Danielson

Coming into the semester I was broadly familiar with motor/generator duality. I was totally ignorant of the specifics however and wasn't sure if a given motor could also be used as a generator in the same system. My primary outside source of information was "Running a Brushed DC Motor as a Generator". [42] This white paper went into great detail describing the mathematics of using a motor as a generator – it was here that I first encountered the important constant K_e, for instance. From here I immediately turned to our motor's data sheet to check the value of ours, but alas, it was not listed. It turned out I was going to have to do a bit of independent research. Once the motors came in, I took one and placed its shaft into the end of a cordless drill and cinched the jaws of the drill up around it. I attached one end of a pair of alligator clips to the leads of the motor (now generator), and the other to a multimeter. I began spinning the drill at a constant pace, counting the revolutions of the shaft (with the aid of a dot drawn on by a Sharpie), and monitored the produced voltage on the meter. I did this many times, at many different speeds, and with the data from these trials was able to calculate K_e for our particular motor.

Knowing this value was vital to the power regeneration subsystem – if it had happened to be too low, we would not have generated enough power to charge the capacitor. It of course also

allowed me to calculate expected (or theoretically ideal) values in my testing regime, giving me the necessary data to compare my measured values against. Although a very modest experiment, it nevertheless was my own little introduction into research, and it was very cool to be able to operationalize my results.

The circuit design-work was entirely informed by coursework – ECE 101 and ECE 102 in particular were key here. I picked up a few other skills over the course of the semester as well, though these were mostly related to wood working – mounting the motor required the use of a power-saw and a few other tools, for instance.

9. Conclusions and Future Work

9.1 Team Conclusions, Potential Changes, and Future Considerations

Brian would have spent more time choosing the motor – although the one we ultimately ended up with functioned as desired, to spin it backwards (i.e. open the door) requires a bit more force than one would like. Were another iteration of the cupboard produced, whether to bring to market or simply because of a longer semester, minimizing its footprint would be his primary goal. Many components of the design were brought in to mitigate the effects of choosing such large components. A smaller and less muscular motor would've necessitated a smaller mount and allowed for an easier opening cycle.

Sierra would do more research into the components at the start and have gotten the ESP32-cam and Arduino motion detector in the first place. If Sierra had the ESP32 camera setup earlier, she could have taken pictures for the algorithm sooner and taken more of them to make a better model. Even if Sierra could not take the images earlier, she would want to take even more images since that would significantly increase the accuracy. She would have documented the MCU and ESP pins better and their connections. If she drew out the connections, there could have been much less confusion.

Maddie would spend less time configuring the ESP32-Cam now that she understands the functionality of the camera. Quite a lot of development time was spent debugging the functionality of the camera, especially with a Mac computer, but now that the learning curve has been bridged, the companion app could have more design work done. Additionally, she would have worked with the ATMega design more closely, in order to determine if the ATMega was necessary in this project, or if we could just operate the device on the ESP32-Cam.

Matt would make researching and developing the power bus a greater priority to the project with potential backup options; the supply to the prototype just barely covered the needs of the system, and having other options to pivot to would have made for more future functionality. Matt would also change the way he went about developing the PCB: while he would also spend more time designing the PCB itself, he would have made strides to fully integrate the circuit on breadboard before sending in the order. Some of the dovetailing was done to save time due to the time constraint, but completing the breadboard first would have ironed out several problems that cropped up once the PCB arrived. Given the chance to fine-tune the prototype over another semester, Matt would redesign the PCB to account for the new components used after considering a different topology for my drive system. Ironically, the final implementation of the drive system used a method remarkably like an H-bridge to control the polarity of the motors in different states.

A full H-bridge controlled by the MCU would allow for more robust control of the drive system and provide the same or better energy-saving if the switches were chosen and ordered instead of scavenged from Benedum. Alternatively, a relay array more suited to the higher power levels could be used as a replacement for the switching network. Future work would consider and implement these changes and complete testing to best judge the power savings when compared to the original finished prototype. This would be paramount to bringing the device to market, as a more concrete understanding of power consumption within the drive itself would allow for better future iterations, saving the potential buyer money on replacement batteries. Overall, more time would allow for greater scope of testing the prototype, and its potential as a product going to market could be determined once iterations on both the hardware and software provide more functionality for the potential user.

9.2 Evaluation of Prototype:

The group feels the prototype ended development in an appropriate position given the constraints of the course. There were a lot of unexpected hiccups for the project along the way. Often there were things that the group thought be straightforward but were often not: setting up the camera, opening the door, and running the image classification on AWS were a few of the major hurdles. The group often had to pivot, finding a new way to accomplish the same thing with each problem. Given all these small to big problems they were still able to accomplish a working prototype; given the challenges overcome and the time constraint of the project, the team is proud of what they have accomplished.

Naturally, the marketability of the device would be dependent on the differences in functionality compared with the current mechanics-based designs available at this time; the group believes that given more time in development and greater economies-of-scale, the price of the product could be brought down into competitive range with the mechanical solutions while providing significantly greater user functionality in the vein of "smart home" technology. The addition of functionality like remote closing, hand gestures, greater power sustainability solutions, and modularization of the mechanics would prove marketable to an audience of consumers who are interested in automating their home for increased convenience. Considering mechanical solutions are priced starting at $10 USD, the pricing goal per-unit would be under $50 USD with attempts to push down to $30 USD. Given the economies of scale for electronics design and manufacturing, the hardware component would decrease in price relative to units produced; the software component would be solely reliant upon increased server space within AWS. The major hurdle to overcome would be the aforementioned mechanical design and a streamlined modular mounting system. However, given the increasing prevalence and decreasing cost of printing a 3D model a reasonable design solution could be achieved for most cabinets.

Appendix

Motor Drive Testing								
Trial	**Voltage Peak (V)**	**Current Peak (A)**	**Power Peak (W)**	**Time On (s)**	**Energy (mAh)**	**Length (in)**	**Speed (RPM)**	**Torque (mN*m)**
1	6.877	0.081	0.557	3.050	68.625	12	46.964	113.264
2	6.888	0.096	0.661	3.320	88.533	12	43.144	146.356

3	6.905	0.086	0.594	2.890	69.039	12	49.564	114.411
4	6.911	0.086	0.594	3.150	75.250	12	45.473	124.812
5	6.975	0.082	0.572	3.225	73.458	12	44.415	122.969
6	6.946	0.079	0.549	2.840	62.322	12	50.436	103.894
7	6.934	0.077	0.534	2.890	61.814	12	49.564	102.868
8	6.973	0.082	0.572	2.960	67.422	12	48.392	112.832
9	6.97	0.116	0.809	3.20	103.111	12	44.762	172.484
10	6.952	0.082	0.570	3.335	75.964	12	42.950	126.744
Average	**6.9331**	**0.0867**	**0.601**	**3.086**	**74.554**	**12**	**46.566**	**124.064**
11	6.937	0.082	0.569	4.005	91.225	15	44.706	121.502
12	6.923	0.086	0.595	4.095	97.825	15	43.724	130.031
13	6.931	0.076	0.527	3.885	82.017	15	46.087	109.144
14	6.97	0.082	0.572	4.105	93.503	15	43.617	125.129
15	6.879	0.076	0.523	3.770	79.589	15	47.493	105.118
16	6.827	0.079	0.539	3.950	86.681	15	45.329	113.619
17	6.87	0.077	0.529	4.030	86.197	15	44.429	113.698
18	6.838	0.082	0.561	3.980	90.656	15	44.987	119.021
19	6.854	0.086	0.589	4.085	97.586	15	43.831	128.420
20	6.785	0.085	0.577	3.985	94.090	15	44.931	122.573
Average	**6.8814**	**0.0811**	**0.558**	**3.989**	**89.937**	**15**	**44.914**	**118.826**
21	6.828	0.087	0.594	4.875	117.812	18	44.074	128.708
22	6.895	0.091	0.627	4.880	123.356	18	44.029	136.086
23	6.858	0.102	0.700	4.805	136.142	18	44.716	149.386
24	6.821	0.1	0.682	4.945	137.361	18	43.450	149.910
25	6.823	0.081	0.553	4.770	107.325	18	45.044	117.165
26	6.832	0.107	0.731	4.865	144.599	18	44.164	158.064
27	6.831	0.084	0.574	5.040	117.600	18	42.631	128.532
28	6.937	0.083	0.576	4.930	113.663	18	43.582	126.158
29	6.881	0.086	0.592	4.990	119.206	18	43.058	131.241
30	6.88	0.086	0.592	4.940	118.011	18	43.494	129.907
Average	**6.8586**	**0.0907**	**0.622**	**4.904**	**123.508**	**18**	**43.824**	**135.515**
31	6.905	0.103	0.711	5.890	168.519	21	42.558	159.583
32	6.909	0.084	0.580	5.675	132.417	21	44.171	125.467
33	6.893	0.081	0.558	5.650	127.125	21	44.366	120.175
34	6.886	0.088	0.606	6.010	146.911	21	41.709	138.738
35	6.877	0.079	0.543	5.785	126.949	21	43.331	119.729
36	6.877	0.084	0.578	5.96	139.067	21	42.059	131.158
37	6.893	0.095	0.655	5.710	150.681	21	43.900	142.442

38	6.86	0.08	0.549	5.970	132.667	21	41.988	137.200
39	6.874	0.084	0.577	5.935	138.483	21	42.236	130.551
40	6.899	0.101	0.697	6.025	169.035	21	41.605	159.932
Average	**6.8873**	**0.0879**	**0.605**	**5.861**	**143.185**	**21**	**42.792**	**136.498**
41	6.901	0.202	1.394	6.995	392.497	24	40.955	325.035
42	6.894	0.198	1.365	6.635	364.925	24	43.177	301.895
43	6.903	0.196	1.353	6.945	378.117	24	41.250	313.216
44	6.887	0.209	1.439	6.805	395.068	24	42.098	326.500
45	6.907	0.198	1.368	7.02	386.100	24	40.809	320.015
46	6.9	0.197	1.359	6.715	367.460	24	42.663	304.257
47	6.903	0.179	1.236	7.005	348.304	24	40.896	288.521
48	6.895	0.191	1.317	6.805	361.043	24	42.098	298.727
49	6.897	0.174	1.200	6.745	326.008	24	42.473	269.818
50	6.903	0.193	1.332	6.94	372.061	24	41.279	308.201
Average	**6.899**	**0.1937**	**1.336**	**6.861**	**369.158**	**24**	**41.770**	**305.618**

Table 11: Comprehensive Motor Drive Testing

Battery Testing (Drive and Regeneration)						
Trial	**Voltage Peak (V)**	**Current Peak (A)**	**Power Peak (W)**	**Time (s)**	**Energy (mAh)**	**Length (in)**
1	7.985	0.159	1.270	6.690	147.738	12
2	7.979	0.155	1.237	6.500	139.931	12
3	7.981	0.165	1.317	7.140	163.625	12
4	7.977	0.154	1.228	6.380	136.461	12
5	7.981	0.151	1.205	6.820	143.031	12
6	7.975	0.156	1.244	6.620	143.433	12
7	7.973	0.158	1.260	6.940	152.294	12
8	7.968	0.172	1.370	6.600	157.667	12
9	7.968	0.187	1.490	6.920	179.728	12
10	7.959	0.159	1.265	6.520	143.983	12
Average	**7.9746**	**0.1616**	**1.289**	**6.713**	**150.789**	
11	7.963	0.175	1.394	8.330	202.465	15
12	7.961	0.166	1.322	8.410	193.897	15
13	7.962	0.178	1.417	8.180	202.228	15
14	7.959	0.166	1.321	8.540	196.894	15
15	7.958	0.181	1.440	8.270	207.899	15
16	7.955	0.133	1.058	8.310	153.504	15
17	7.952	0.152	1.209	8.020	169.311	15
18	7.958	0.138	1.098	8.910	170.775	15

19	7.954	0.158	1.257	7.960	174.678	15
20	7.958	0.218	1.735	8.480	256.756	15
Average	**7.958**	**0.1665**	**1.325**	**8.341**	**192.841**	
21	7.959	0.235	1.870	10.360	338.139	18
22	7.959	0.183	1.456	10.500	266.875	18
23	7.959	0.209	1.663	9.650	280.118	18
24	7.958	0.215	1.711	10.310	307.868	18
25	7.958	0.191	1.520	10.180	270.053	18
26	7.958	0.191	1.520	10.030	266.074	18
27	7.957	0.157	1.249	10.060	219.364	18
28	7.955	0.212	1.687	9.700	285.611	18
29	7.955	0.206	1.639	9.780	279.817	18
30	7.955	0.222	1.766	10.170	313.575	18
Average	**7.9573**	**0.2021**	**1.608**	**10.074**	**282.749**	
31	7.955	0.158	1.257	12.310	270.136	21
32	7.955	0.181	1.440	11.610	291.863	21
33	7.953	0.155	1.233	11.960	257.472	21
34	7.953	0.181	1.439	12.120	304.683	21
35	7.954	0.179	1.424	12.190	303.057	21
36	7.954	0.156	1.241	11.930	258.483	21
37	7.951	0.217	1.725	12.100	364.681	21
38	7.952	0.163	1.296	11.960	270.761	21
39	7.949	0.151	1.200	12.000	251.667	21
40	7.947	0.146	1.160	12.350	250.431	21
Average	**7.9523**	**0.1687**	**1.342**	**12.053**	**282.323**	
41	7.946	0.204	1.621	14.310	405.450	24
42	7.947	0.203	1.613	14.190	400.079	24
43	7.946	0.18	1.430	13.780	344.500	24
44	7.947	0.205	1.629	14.660	417.403	24
45	7.948	0.194	1.542	14.020	377.761	24
46	7.948	0.174	1.383	14.600	352.833	24
47	7.948	0.185	1.470	14.260	366.403	24
48	7.947	0.181	1.438	13.880	348.928	24
49	7.946	0.197	1.565	13.640	373.206	24
50	7.946	0.179	1.422	14.480	359.989	24
Average	**7.9469**	**0.1902**	**1.511**	**14.182**	**374.655**	

Table 12: Comprehensive Battery Testing

References

[1] “How to use OV7670 camera module with Arduino ,” Uno, https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino (accessed Jun. 1, 2023).
[2] Vicapow, “Image kernels explained visually,” Explained Visually, https://setosa.io/ev/image-kernels/ (accessed Jun. 1, 2023).
[3] K. Mittal, “A gentle introduction into the histogram of oriented gradients,” Medium, https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa (accessed Jun. 1, 2023).
[4] M. Galkissa, “Training SVM classifier with Hog features,” Kaggle, https://www.kaggle.com/code/manikg/training-svm-classifier-with-hog-features (accessed Jun. 1, 2023).
[5] G. Boesch, “Object detection in 2023: The definitive guide,” viso.ai, https://viso.ai/deep-learning/object-detection/ (accessed Jun. 1, 2023).
[6] N. Khandelwal, “Image Processing in Python: Algorithms, Tools, and Methods You Should Know,” neptune.ai, https://neptune.ai/blog/image-processing-python (accessed Jun. 1, 2023).
[7] “Arduino - Motion Sensor: Arduino tutorial,” Arduino Getting Started, https://arduinogetstarted.com/tutorials/arduino-motion-sensor (accessed Jun. 1, 2023).
[8] “Arduino with pir motion sensor,” Random Nerd Tutorials, https://randomnerdtutorials.com/arduino-with-pir-motion-sensor/ (accessed Jun. 1, 2023).
[9] A. Rosebrock, “Histogram of oriented gradients and object detection,” PyImageSearch, https://pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/ (accessed Jun. 1, 2023).
[10] S. Mallick, “Histogram of oriented gradients explained using opencv,” LearnOpenCV, https://learnopencv.com/histogram-of-oriented-gradients/ (accessed Jun. 1, 2023).
[11] A. Waheed, “How to apply hog feature extraction in python,” Python Code, https://www.thepythoncode.com/article/hog-feature-extraction-in-python#:~:text=The%20Histogram%20of%20Oriented%20Gradients,image%20or%20region%20of%20interest (accessed Jun. 1, 2023).
[12] “OpenCV optical flow algorithm for Object Tracking: Mike Polinowski,” Mike Polinowski RSS, https://mpolinowski.github.io/docs/IoT-and-Machine-Learning/ML/2021-12-10--opencv-optical-flow-tracking/2021-12-10/ (accessed Jun. 1, 2023).
[13] C. Lin, “Introduction to motion estimation with optical flow,” Nanonets AI & Machine Learning Blog, https://nanonets.com/blog/optical-flow/ (accessed Jun. 1, 2023).
[14] W. by: G. D. Luca, “SVM vs Neural Network,” Baeldung on Computer Science, https://www.baeldung.com/cs/svm-vs-neural-network (accessed Jun. 1, 2023).
[15] “Histogram of oriented gradients¶,” Histogram of Oriented Gradients - skimage v0.20.0 docs, https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html (accessed Jun. 1, 2023).
[16] Petrova and Solovev, “DC Motor Controller,” IntegraSources, https://www.integrasources.com/blog/dc-motor-controller-design-principles/ (accessed May 24, 2023).

[17] “What Size Capacitor Should I Use?” Cadence, https://resources.pcb.cadence.com/blog/2022-what-size-capacitor-should-i-use (accessed May 29, 2023).
[18] “DC Motor Drive Circuit Design,” Cadence, https://resources.pcb.cadence.com/blog/dc-motor-drive-circuit-design (accessed May 22, 2023).
[19] G. Recktenwald, “Basic DC Motor Circuits,” Portland State University, https://cdn.sparkfun.com/assets/resources/4/4/DC_motor_circuits_notes_2up.pdf (accessed May 22, 2023).
[20] H. Sax, “How to Drive Motors with Smart Power ICs,” https://www.st.com/resource/en/application_note/an380-how-to-drive-dc-motors-with-smart-power-ics-stmicroelectronics.pdf (accessed May 29, 2023).
[21] “AWS Pricing,” Amazon, https://aws.amazon.com/free (accessed Jun 1, 2023).

[22] “What are the Top Injuries in a Typical Office and How Can You Avoid Them?” Albert Einstein College of Medicine, https://www.einsteinmed.edu/administration/environmental-health-safety/accident-injury-reduction-campagin/top-injuries.aspx#:~:text=Falling%20down%20is%20not%20only,fall%20than%20non%2Doffice%20workers. (Accessed Jun. 1, 2023).

[23] “Office Safety,” UC Santa Cruz Environmental Health & Safety, https://ehs.ucsc.edu/programs/safety-ih/office-safety.html, (Accessed Jun. 1, 2023).

[24] “NSPIRE Standard,” Department of Housing and Urban Development, https://www.hud.gov/sites/dfiles/PIH/documents/NSPIRE-Standards-v2.2-Cabinets.pdf (Accessed Jun. 1, 2023).

[25] “Pennsylvania’s Worker’s Compensation Act,” Pennsylvania Department of Labor and Industry, https://www.dli.pa.gov/Individuals/Workers-Compensation/publications/Pages/WC%20Act/WC-Act-Landing-Page.aspx (Accessed Jun. 1, 2023).

[26] Bureau of Labor Statistics Data Tools: Non-Fatal Workplace Injuries Database, https://www.bls.gov/data/home.htm, (Accessed Jun. 1, 2023).

[27] "What is Internet of Things (IoT)," Amazon Web Services. https://aws.amazon.com/what-is/iot/ (Accessed Jun 2, 2023).

[28] “MQTT Message Payload,” Amazon Web Services. https://docs.aws.amazon.com/iot/latest/developerguide/topicdata.html (Accessed Jun 2, 2023).

[29] "IoT Core," Google Cloud. https://cloud.google.com/iot-core (Accessed Jun 2, 2023).

[30] "AWS IoT Core," Amazon Web Services. https://aws.amazon.com/iot-core/ (Accessed Jun 2, 2023).

[31] "AWS IoT Device SDKs," Amazon Web Services. https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html (Accessed Jun 2, 2023).

[32] "AWS IoT Core Features," Amazon Web Services. https://aws.amazon.com/iot-core/features/ (Accessed Jun 2, 2023).

[33] "Device Communication Protocols," Amazon Web Services. https://docs.aws.amazon.com/iot/latest/developerguide/protocols.html (Accessed Jun 2, 2023).

[34] "Amazon S3," Amazon Web Services. https://aws.amazon.com/s3/ (Accessed Jun 2, 2023).

[35] "Using AWS Lambda with AWS IoT," Amazon Web Services. https://docs.aws.amazon.com/lambda/latest/dg/services-iot.html (Accessed Jun 2, 2023).

[36] "Rules for AWS IoT," Amazon Web Services. https://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html (Accessed Jun 2, 2023).

[37] "Create AWS IoT Resources," Amazon Web Services. https://docs.aws.amazon.com/iot/latest/developerguide/create-iot-resources.html (Accessed Jun 2, 2023).

[38] "ESP8266 Wi-Fi Module (WRL-17146) Datasheet," Sparkfun. https://cdn.sparkfun.com/assets/f/e/5/6/f/ESP8266ModuleV2.pdf (Accessed Jun 2, 2023).

[39] "ROB-15277 Datasheet," Sparkfun.

https://cdn.sparkfun.com/assets/6/9/d/c/c/rmspec.pdf (Accessed Jun 2, 2023).

[40] "OV7670 640x480 0.3Mega 300KP VGA CMOS Camera Module Datasheet," OpenHacks.

https://www.openhacks.com/uploadsproductos/ov7670_cmos_camera_module_revc_ds.pdf (Accessed Jun 2, 2023).

[41] "ATmega328P Datasheet," Microchip. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (Accessed Jun 2, 2023).

[42] "Running a Brushed DC Motor as a Generator," Portescap. https://www.portescap.com/en/newsroom/whitepapers/2021/12/running-a-brushed-dc-motor-as-a-generator (Accessed Jun 2, 2023).

[43] Grimshaw, Paul. "How Can Capacitors Charge a Battery?," Quora. https://www.quora.com/How-can-capacitors-charge-a-battery (Accessed Jun 2, 2023).

[44] "HHR150AA Datasheet" https://api.pim.na.industrial.panasonic.com/file_stream/main/fileversion/3515 (Accessed Jun 2, 2023).

[45] Andrea Manero-Bastin, "How to Configure the Number of Layers and Nodes in a Neural Network - DataScienceCentral.com," *Data Science Central*, Feb. 17, 2019. https://www.datasciencecentral.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural/

[46] "Arduino Reference," *Arduino.cc*, 2019. https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[47] "Secrets of Arduino PWM | Arduino Documentation," *docs.arduino.cc*. https://docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm

[48] K. Magdy, "ESP32 GPIO (Digital Inputs & Digital Outputs) - Arduino Tutorial," *DeepBlue*, Apr. 24, 2021. https://deepbluembedded.com/esp32-digital-inputs-outputs-arduino/ (accessed Jul. 28, 2023).

[50] "ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained | Random Nerd Tutorials," Mar. 10, 2020. https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/

[51] A. Pandit, “How to Use OV7670 Camera Module with Arduino Uno,” *Circuitdigest.com*, 2019. https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino
[52] Instructables, “OV7670 Arduino Camera Sensor Module Framecapture Tutorial,” *Instructables*, Oct. 24, 2016. https://www.instructables.com/OV7670-Arduino-Camera-Sensor-Module-Framecapture-T/
[53] “How to Wire and Program a Button | Arduino Documentation,” *docs.arduino.cc*. https://docs.arduino.cc/built-in-examples/digital/Button (accessed Jul. 28, 2023).
[54] Aishwarya, “How to send captured image from ESP32cam to AWS with code,” *Pigirl2020*, Apr. 16, 2020. https://pigirl2020.blogspot.com/2020/04/ESP32CAM-AWS.html (accessed Jul. 28, 2023).
[55] “ESP32-CAM HTTP Post Images/Photos to Server | Random Nerd Tutorials,” *Random Nerd Tutorials*, Mar. 27, 2023. https://randomnerdtutorials.com/esp32-cam-post-image-photo-server/ (accessed Jul. 28, 2023).
[56] “How to Exchange Data between Arduino and ESP32 using Serial Communication?,” *Programming Boss: Programming for Beginners*. https://www.programmingboss.com/2021/04/esp32-arduino-serial-communication-with-code.html#gsc.tab=0
[57] “Serial Communication Between Arduino and ESP32 CAM,” *Programming Boss: Programming for Beginners*, Jan. 04, 2023. https://www.programmingboss.com/2023/01/serial-communication-between-arduino-and-esp32-CAM-UART-data-communication.html#gsc.tab=0 (accessed Jul. 28, 2023).
[58] “ESP32-CAM WiFi + bluetooth Camera Module with Camera Module OV2640,” *botnroll.com*. https://www.botnroll.com/en/esp/3259-esp32-cam-wifi-bluetooth-camera-module-with-camera-module-ov2640.html (accessed May 07, 2023).
[59] “Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux) | Random Nerd Tutorials,” Jul. 05, 2019. https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
[60] “How YOU can Get Started with The ESP32-CAM for DIY Security Cameras,” *www.youtube.com*. https://www.youtube.com/watch?v=k_PJLkfqDuI (accessed Jul. 28, 2023).
[61] “ESP32-CAM - Getting Started & Solving Common Problems,” *DroneBot Workshop*, May 24, 2020. https://dronebotworkshop.com/esp32-cam-intro/
[62] “Supervised learning: predicting an output variable from high-dimensional observations,” *scikit-learn*. https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html
[63] “1.17. Neural network models (supervised) — scikit-learn 0.23.1 documentation,” *scikit-learn.org*. https://scikit-learn.org/stable/modules/neural_networks_supervised.html
[64] “python - How to remove a column in a numpy array?,” *Stack Overflow*. https://stackoverflow.com/questions/34007632/how-to-remove-a-column-in-a-numpy-array
[65] “sklearn.svm.SVC,” *scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.predict (accessed Jul. 28, 2023).
[66] “9. Model persistence,” *scikit-learn*. https://scikit-learn.org/stable/model_persistence.html (accessed Jul. 28, 2023).
[67] “API Reference — skops 0.8 documentation,” *skops.readthedocs.io*. https://skops.readthedocs.io/en/stable/modules/classes.html#skops.io.dump (accessed Jul. 28, 2023).

[68] Python Software Foundation, “csv — CSV File Reading and Writing — Python 3.8.1 documentation,” *Python.org*, 2020. https://docs.python.org/3/library/csv.html
[69] “How to Program / Upload Code to ESP32-CAM AI-Thinker (Arduino IDE) | Random Nerd Tutorials,” Feb. 04, 2020. https://randomnerdtutorials.com/program-upload-code-esp32-cam/
[70] IEEE SA, “IEEE Recommended Practice for Privacy Considerations for IEEE 802 ® Technologies,” *IEEE*, 2020.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9257130
[71] IEEE Cyber Security - SIG, “Transport Layer Security (TLS),” *IEEE*, 2023. https://site.ieee.org/ocs-cssig/?page_id=658
[72] T. Dierks and E.Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” *RFC Editor*, 2008. https://doi.org/10.17487/RFC5246
[73] "Key Management in AWS IoT," Amazon Web Services, 2023. https://docs.aws.amazon.com/iot/latest/developerguide/key-management.html
[74] Moheeb Zara, "Building an AWS IoT Core device using AWS Serverless and an ESP32," Amazon Compute, 2020. https://aws.amazon.com/blogs/compute/building-an-aws-iot-core-device-using-aws-serverless-and-an-esp32/
[75] Nir Eyal, “Hooked: How to Build Habit-Forming Products,” 2014.