Automatic Trading Card Sorter

By
Andrejun Agsalud(agsalud2)
Steve Guzman (steveg3)
David Medina(davidrm3)

Final Report for ECE445, Senior Design, Spring 2024
TA: Nikhil Arora

May 2024

Project No. 32

Abstract

The Automatic Trading Card Sorter (ATCS) is a camera detection and analysis device built for all trading cards. In the current market, rare cards are not frequent and can take going through dozens of packs until something of value appears. This process is tedious and wastes precious time. The Automatic Trading Card Sorter aims to tackle this problem through simplicity and timely organization of trading cards. By using a drop down chute, the cards will neatly be piled into four categories while allowing for easy retrieval. This design is separated into a control, motor, camera, power, and user interface subsystems. This document will reflect on the design choices and process. Finishing off with a reevaluation of what was learned and changed that could be made.

Contents

Introduction

1.1 Purpose

With the collection of trading cards expanding, collectors and sellers face a significant logistical challenge; sorting through thousands of cards efficiently to identify and categorize them for storage or sale. According to business research insights, the trading card marker was valued at $956 million in 2021 with projection seeing it rise more and more per year (“Trading Card Market Size, Share, Growth | Forecast - 2031”). With individual cards being worth thousands of dollars nowadays, the stakes and accuracy of sorting through many packs of cards is critical. Reselling websites such as Ebay have witnessed a 142% boost in card sales in 2020 (Lindner). There are currently many ways to store and organize cards through booklets and trays; there are even playing card sorters you can buy. However, there is a clear lack of a tool that can help people go through trading cards specifically. The current manual sorting process is time-consuming and prone to error, leading to missed opportunities or damages to the cards. Not only this, but the use of fingers can possibly damage and smudge the cards, lowering the value of the trading card.

1.2 Solution

This team proposes a solution that aims to address the lack of a trading card sorting by building a device that will allow its user to efficiently sort through a pack of Pokemon cards without fail. This Automatic Trading Card Sorter will leverage existing technologies and combine them into a single package that will be easily fabricated. We aim to reduce the time and effort required to sort trading cards while also maintaining the card’s condition.
To implement the Automatic Trading Card Sorter (ATCS), we will aim for simplicity first. Based on the machine shop's input, it was found that it will be better to use a feeder that will dispense

one card to be read by a camera, be assigned a classifier value, and then moved to the appropriate sorted pile. The mechanism for dispensing one card will involve a clever design for the feeder and servo motor that will hold one end of the card, while the camera scans the other. For this solution, we aim to sort by 3 to 4 colors. Since most trading cards of color have the borders colored, only a small section of the card will be analyzed by the camera that will be using computer vision to tell the colors apart. Once a color has been identified, the onboard microcontroller will handle the logic of what to do with that color and decide what direction the platform motor should rotate. Cards are sorted by having the platform where they are being analyzed rotate and place the card into a bin of the appropriate color. There will be buttons available to pause the machine so the user can empty a bin and place it back and buttons for which colors are enabled to be analyzed. LEDs will accompany the color buttons so that the user is aware of which colors are currently being sorted.
Our solution will decrease the time and effort trading care enthusiasts go through when going through a new pack of cards, allowing them more time to organize their collections for storage or sale, and becoming a staple in the card collectors home.

1.3 Visual Aid

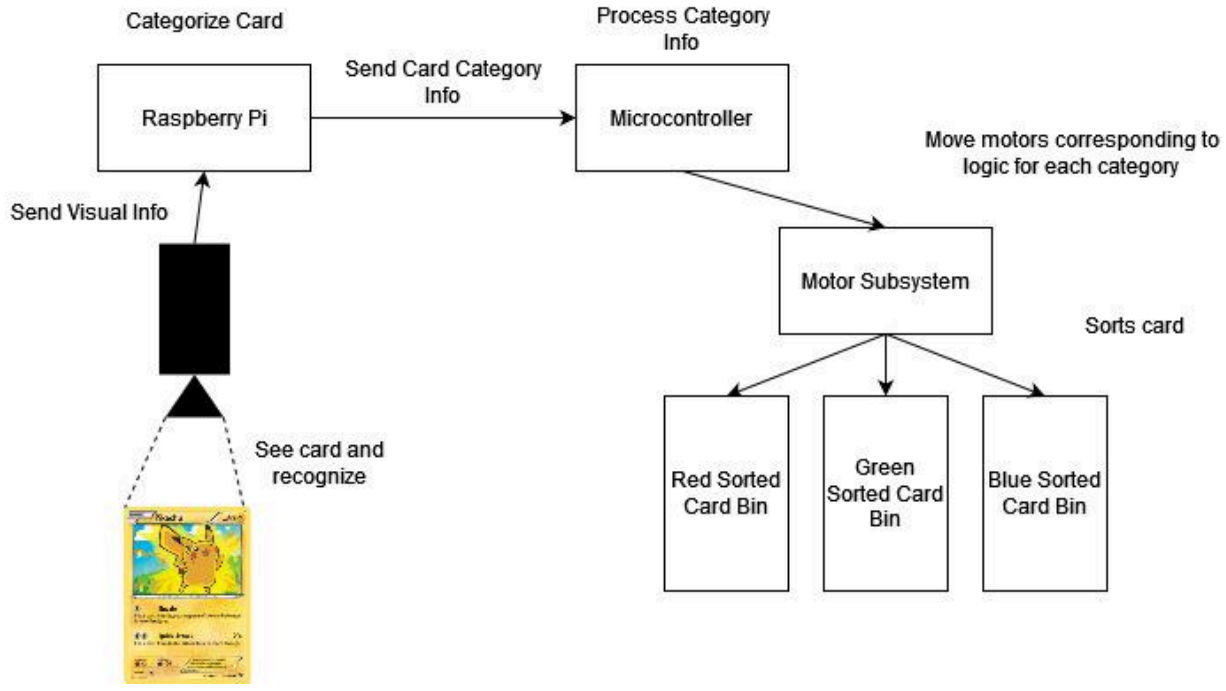

Figure 1: High Level Diagram

1.4 High Level Requirements

1. Two packs of Pokemon cards (22 cards) will be sorted by primary color with one extra bin for non-RGB cards. Maximum of 2 mis-colored cards with the goal of achieving at

least 80% accuracy in sorting. It is a success if 16 out of the 20 actual pokemon cards are sorted in the correct spaces.

2. A single pack (12 cards) should be able to be sorted in 30 ± 2 seconds.
3. Will identify RGB (primary colors) with 90% accuracy. Computer vision will recognize if color doesn't match.

1.5 Block Diagram

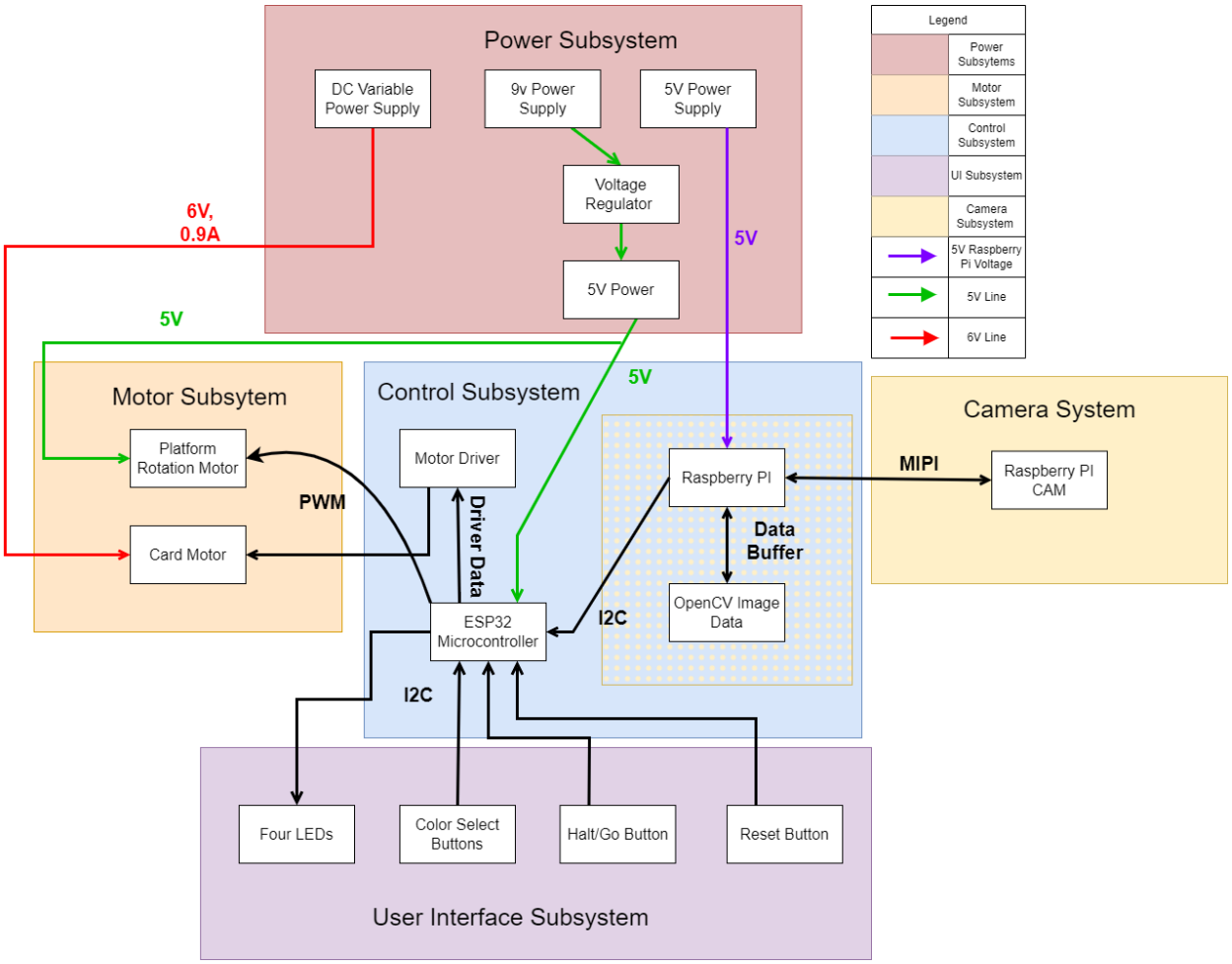

Figure 2: Final Block Diagram of Subsystems

1.6 Sub-system Overview

1.6.1 Control Subsystem

This subsystem is responsible for all the logical control. It uses a ESP32 as the microcontroller which in turn takes input from the User Interface and Camera subsystems to control the motor subsystem.

1.6.2 Power Subsystem

This subsystem provides power to the motor, camera, and control subsystems. It uses one 5V supply for the camera subsystem to avoid motor current spike interference. One motor uses a stepped down 5v that is also shared by the ESP32. While the other motor uses a DC voltage supply to power its high current.

1.6.3 User Interface Subsystem

This system allows the user to view relevant information on which sorting configuration is enabled and allows the user to press buttons to customize said sorting. Entire system is power and controlled by the control subsystem

1.6.4 Camera Subsystem

The system is responsible for monitoring the current card in the holding tray and determining what type of card it is so that the relevant bytes of data can be sent to the ESP32 for logical control. It does not provide any logical control of any other subsystem.

1.6.5 Motor Subsystem

This system is the main mechanical driver of the device. Based on logic from the ESP32, the platform motor will spin to the appropriate spot for said card and then spin the card motor to dispense into a sorted location

Design

The design of the device went through many iterations, in this section we will highlight the process that was done for each subsystem and what compromises had to be made. Overall, the ATCS was simple in concept and design, but both mechanical and electrical issues led to device having issues. The initial brainstorming of the device consisted of three main aspects. One was the holding of the cards along with their analysis. A design that could allow a camera to view those cards and also hold them posed a challenge that was overcome with help from the machine shop. The second aspect was the sorting of the cards themselves; we understood that having four slots was needed and would be able to accommodate easily into any design. Lastly, the dispensing of one signal card was crucial to ensure that no cards were accidentally thrown out. By thinking of the physical design, the needed electrical components were able to be gathered for the product to come together. One last note, this device was limited to Pokemon cards since they were the most easily accessible and had the color aspects we wanted since sorting by color would have conveyed the main focus of our device to a potential investor.

2.0 Overall Design Process

The design process of the Automatic Trading Card Sorter(ATCS) started off with a rough idea. As seen in Figure 3. This design overall encompassed the three points mentioned earlier for how we wanted the cards to be sorted. This initial design was meant to convey a concept for the machine shop to follow, so the measurements and spacing were not taken into consideration.

Talking with the machine shop, a build was given to us to bounce off of. As seen in Figure 4, the machine shop provided a nice base to test its servo and other components early in the process. With that working, the team settled on Figure 5. This device had a 9 inch radius(originally had inch) which made it larger and heavier. But the core aspects of holding/sensing, organization, and dispensing were included.

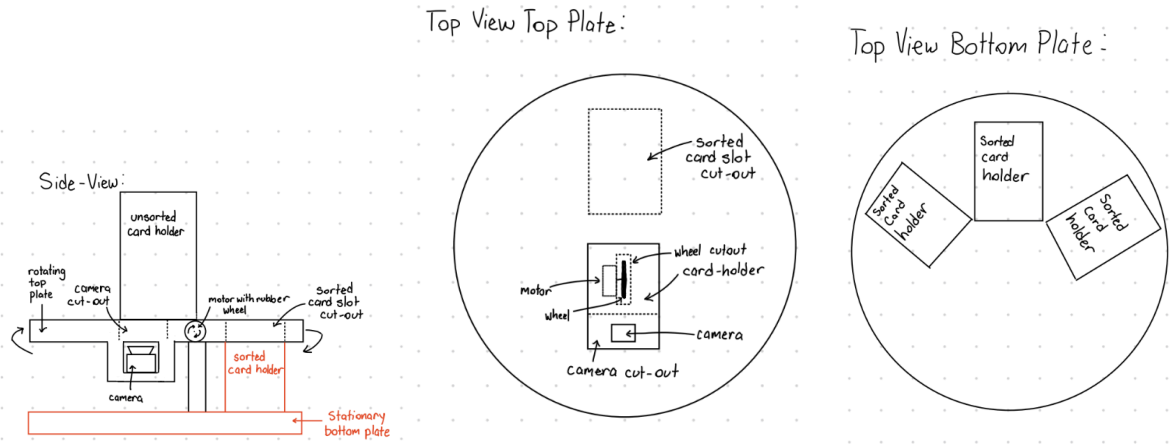

Figure 3: Initial Design Sketches

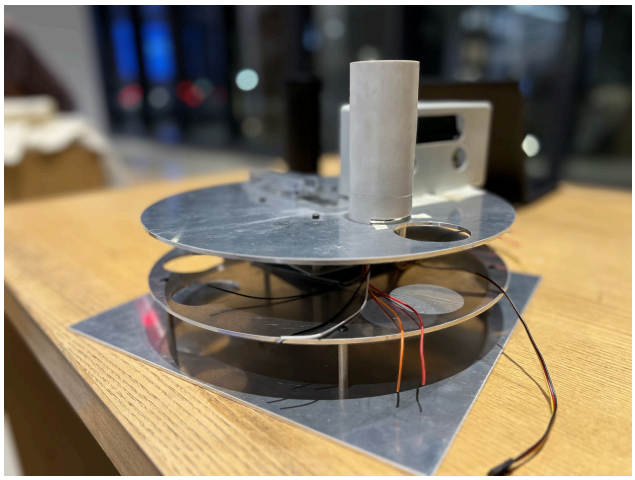

Figure 4: Initial Prototype

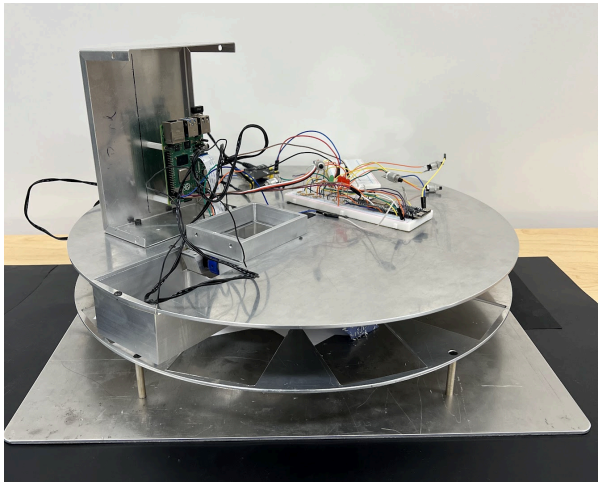

Figure 5: Final Device Design

2.1 Control Subsystem

The control subsystem is the logical center of the device. It takes i2c signals from the Raspberry Pi to determine which card to sort and uses PWM signals to control both motors. User input with the buttons and control of the leds is also controlled by i2c. The ESP32 itself can be powered in

two ways. One is through 5V and it will be protected by the Dev module itself. The other is 3V but this configuration does not have any protection of the ESP32. At the end, we settled on using 5V since it provided protection.

This Subsystem was also the main focus of the PCB. In Figure 6 you can see the layout of all the connections to components as well as the power subsystem that was part of initial design before changes needed to occur. The ESP32 would use many two-pin out ports to interact with leds and momentary switches(buttons). As well as take in power. In particular, you can see the wiring on the ESP32 and the names of the control wires for the leds, buttons, servo, and motor driver. The ESP32 has to have RX and TX pins configured for a programing daughter board that is supplied. On the top right you can see both versions of the programming pins needed to get the ESP32 programmed.

Overall, the ESP32 was chosen as the center for our control system due to its ease of programming compared to the STM microcontroller. While the GPIO pin was rated at 3.3V, in reality it could handle 5V which was all we needed for initial design. However issues did arise later on. This issues and changes will be explained in the power subsystem.

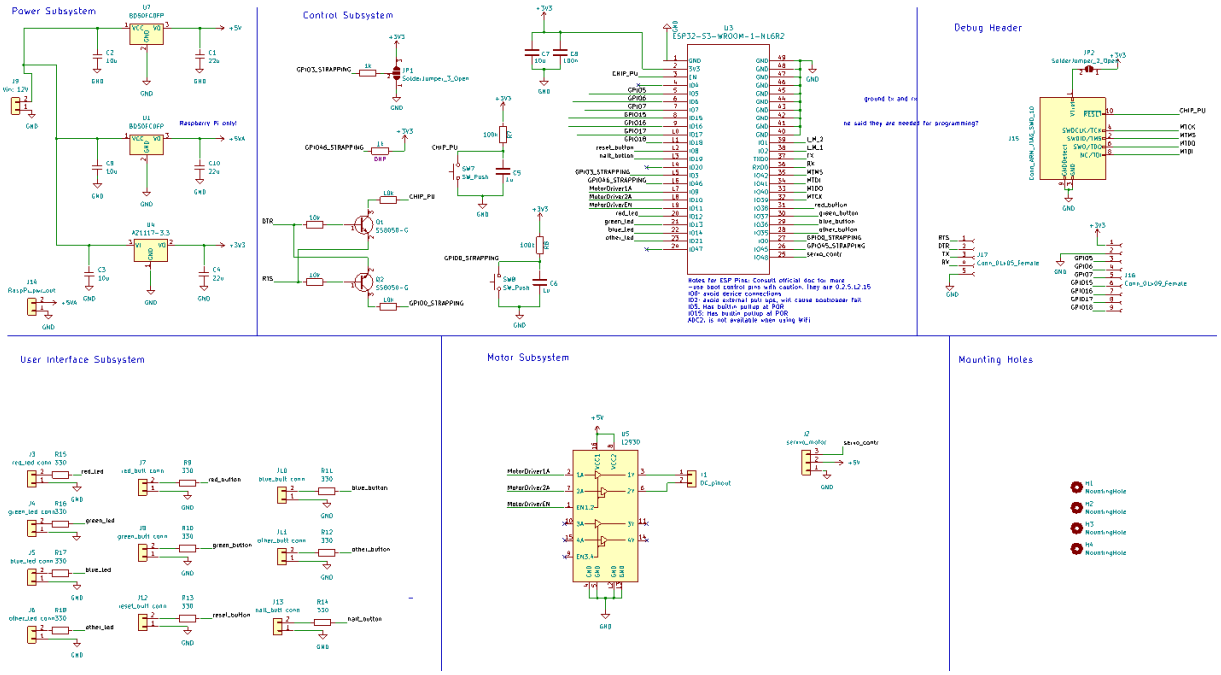

Figure 6: PCB Sketch of Control Unit

2.2 Power Supply Subsystem

The primary purpose of this subsystem was to provide power to the motor, raspberry pi, and ESP32(ESP32 could power UI). The system originally consisted of an independent 5v power supply for the Raspberry Pi to avoid current interference from the motors. The ESP32 would have been powered by a 9V lipo battery that supplied 3200mah. It would have been stepped

down to two 5V linear regulators. One for the ESP32 and the other for both motors. But this design changed to a modular system that would have included a single source of 12V that would have been stepped down to one 5V for Raspberry Pi only, another 5V for the servo motor, and a 3.3V for the ESP32. In Figure 7(top left of Figure 6, you can see what this system would have looked like. The power would come in from the bottom right and then be regulated.

There were many changes to this subsystem. Due to changes in parts at the necessity of the machine shop, many of our initial tolerances were failing. In particular, the DC motor provided for the card dispensing drew around 0.9A at 5V. The ESP32 could no longer independently supply the DC motor so a DC variable supply had to be used for testing.

The servo motor was kept the same from the previous team's project since it worked well and during initial testing, 5V worked perfectly fine with it. However as shown in Figure 5, the size of the device significantly increased. This meant that the weight on the servo increased as well making it work more to compensate. While yes, the 5V was enough to run it, in our final demo it had to be connected to the DC variable supply as well. Its current reached 500maH which is just around ~150maH too high for the ESP 32 to power.

However, the most critical power issue of all was the current. Initially, all our tolerances and calculations were based on the battery and independent Raspberry Pi supply. When moving to the single 12V power supply, the calculations had the linear regulators overheating. To compensate, we tried calculating with higher current and lower voltage but it still proved too much. In turn this meant that time was wasted in finding a single source and not the further testing of the motors under strain. By the time the final machine shop design was delivered, the circuit could no longer power both the servo and DC motor at the same time. Therefore for the sake of our demo, we highlighted the servo and what the sorting process would have looked like.

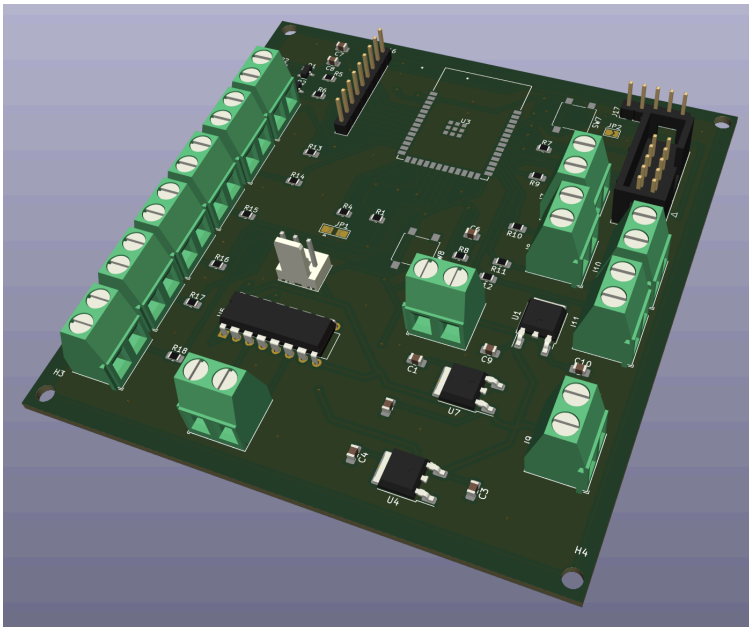

Figure 7: 3D model of PCB

2.3 User Interface Subsystem

The purpose of this subsystem is to give the user control over the machine for both safety and customization. It consists of buttons and LEDs connected to the ESP32. The buttons would be used for controlling the organization categories as well as starting/stopping, and resetting the machine. The ESP32 would be set up with a state machine that would change how it uses the information given to it by the Pi.

There would be a button and LED corresponding to red, green, blue, and yellow. The user would be able to push up to three of these in order to pick the categories for three of the card banks. The reset button would reset these choices to allow the user to pick once again. This would allow the user to have more choices over how the machine sorts. In Figure 8, you can see the machine shop configuration of the four leds and buttons.

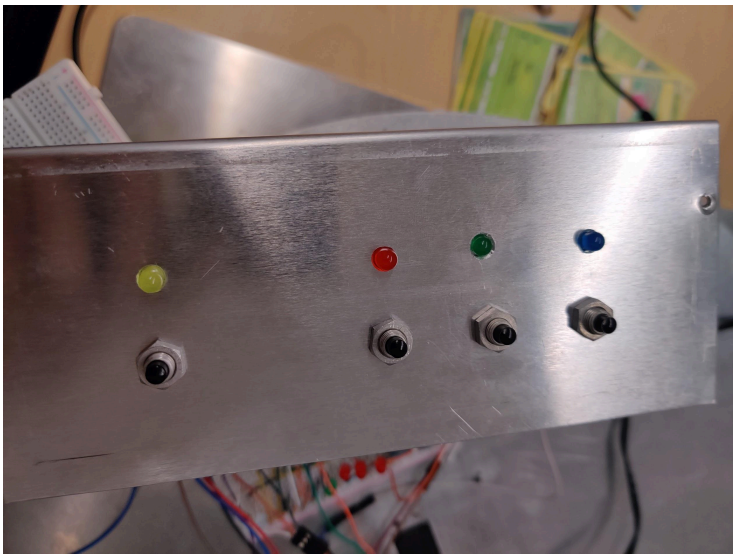

Figure 8: UI on finished build

2.4 Camera System

The purpose of this subsystem is to get data about the current card. It should be able to receive and process this data in real time to find out the card's most prominent primary color. So if the card is a grass type, it should return green. Finally this subsystem should be able to send that information to the control subsystem through I2c.

2.4.1 Integration with Raspberry Pi

It is made up of a raspberry pi 4B+ and pi camera module 2. The pi is running a 32-bit version of raspbian called Bullseye, due to its legacy support of the pi camera module 2 with other software. It will be running openCV for image processing and SMBus for I2C communication.

Initially the pi was going to focus on the feature detection aspect of OpenCV. The initial version of the software was able to identify pokemon cards in a database of images. It would do this by comparing the amount of features found in both cards and choosing the image that had the most shared features. This version of the software was extremely accurate since it would be able to analyze a “perfect” image of the card for information.

I2C communication was achieved through the use of two GPIO pins on both the Pi and the ESP32. The Pi was using the SMBus library and the ESP32 was using a library called wire.h on the arduino IDE. With everything set up, the pi would be able to send the ESP32 a byte of information at a time. This was used to send either an “r”, “g”, “b” or “o” character byte representing the identified color to the ESP32.

2.4.2 Design Changes

Due to changes in focal distance caused by the mounting of the camera, the software was revised to use the colorspace feature of OpenCV instead of feature detection. By using the hue value of the HSV colorspace, the software would be able to detect the average color of an area of the card. This is the value that would be sent to the ESP32.

2.5 Motor Subsystem

This system is the main mechanical driver of the device. It was composed of two motors. The servo motor would serve as a platform rotation motor that will direct the cards towards four slots. Depending on the card, the ESP32 would tell the servo to turn to the exact position for it to be dispensed. It ran on 5V-7V and took in 500mah. Our initial testing showed that the servo could work with 5V from am ESP32 but as mentioned in the power section, the original application was not possible due to weight. The servo itself was limited to 170 degrees but has a conversion kit that allows it to 360 degrees. This conversion kit would ideally be used in a future working version.
The DC motor is what caused more issues for the device however. Originally, we planned on using a stepper motor but due to the daughter board it required, we felt that it may overcomplicate the PCB. So we moved to a DC motor. Initially we used a 3.3V DC motor that took 300mah. It had to use a L293d h-bridge for switch control and higher voltage input. This IC was implemented in the PCB and was used in our final demonstration. This worked fine in tandem with the servo but unfortunately the machine shop recommended a higher speed 5v and 0.9A motor. Initial testing did show it working with the ESP32 and a 5V wall regulator. However, in tandem with the rest of the device neither the 5V supply nor the DC variable supply could power the servo and DC motor at the same time. Therefore we prioritized the servo motor since testing with design showed that cards would get stuck with the DC motor. The reason they got stuck was due to low torque of the DC motor and its inability to bear weight. In a redesign, this team recommends the use of a stepper motor to overcome the weight issue.

Design Verification

3.1. Motor Subsystem

The servo

Requirements	Verification
● The 5V servo should be able to reach all four card slots within it's operable range	● When running, the servo motor must be able to get the top plate in position to dispense a card in every position there is a card slot. ● The reset button must stop operation and return servo to a default starting position
● The DC motor should be able to dispense one card	● When running, the DC must be able to dispense one card into one of the four card slot positions.

In the development of the motor subsystem, the servo motor successfully met its design requirements by accurately reaching all four designated card slots, demonstrating its operational effectiveness in the sorting mechanism. This capability ensures that the servo can position the top plate to dispense cards into each slot accurately. However, challenges arose with the DC motor, which failed to dispense cards one at a time as specified. This issue has been identified as critical and will require further refinement and testing in the final stages of the project to ensure the system can reliably perform its sorting function.

3.2. Power Subsystem

Requirements	Verification
● Maintain the ESP32's 2.2V to 3.6V operational voltage with external power supply	● Use and monitor the ground and 3.3V pins on the ESP to check for steady voltage ● ESP should function independently without needing connection from USB
● Maintain the ESP32's less than 500mA current draw during normal operation(NO WiFi)	● Be aware of flash pins that spike during boot that may cause damage to components ● Use multimeter to check and verify nominal current flow ● Use DC supply to check what the optimal operating current for the DC and Servo are
● Circuit must protect the ESP32 from	● Do not share grounds with motor

current spikes from an inductor(motor) supplied with 6v, 0.9A.	power supply and ESP32 • The Servo's digital must never exceed 5V. To ensure this, 10k resistor is used on the digital pin and monitored with multimeter • The servos H-bridge connection is guarded by a diode to protect the circuit from spikes and monitored with multimeter

The power subsystem was tested to ensure the ESP32 operated within its required voltage range of 2.2V to 3.6V, using an external power supply. During tests, the ESP32 maintained a steady 3.3V, successfully passing this part of the verification process. Current measurements indicated that, with no Wi-Fi enabled, the ESP32's consumption was well within the expected range of under 500 milliamps, averaging around 0.083 milliamps. However, this figure increased significantly to between 300 and 310 milliamps when the servo was connected, pointing to potential current issues. A significant concern was the failure in protecting the ESP32 from voltage spikes caused by the motor, which resulted in the shorting of two units. This issue was attributed to grounding problems and an inadequately protected H-bridge, necessitating a redesign to ensure robust protection against such spikes.

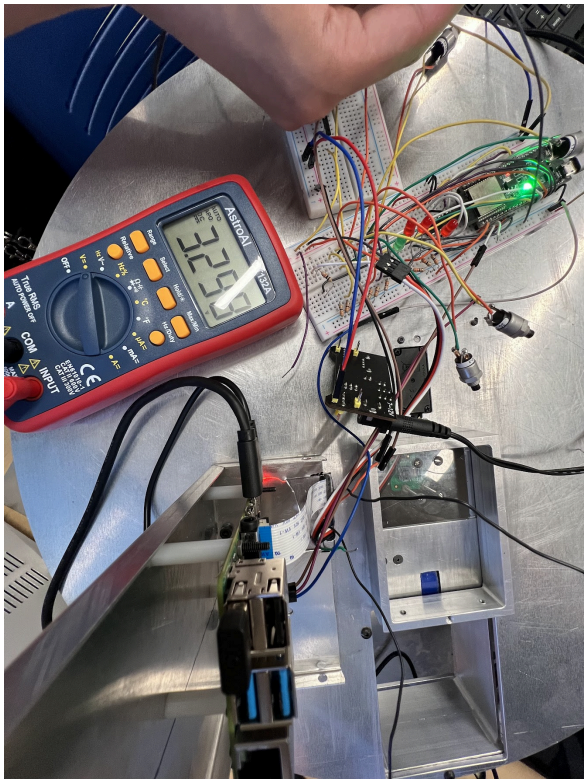

Figure 9 : 3.3V verification

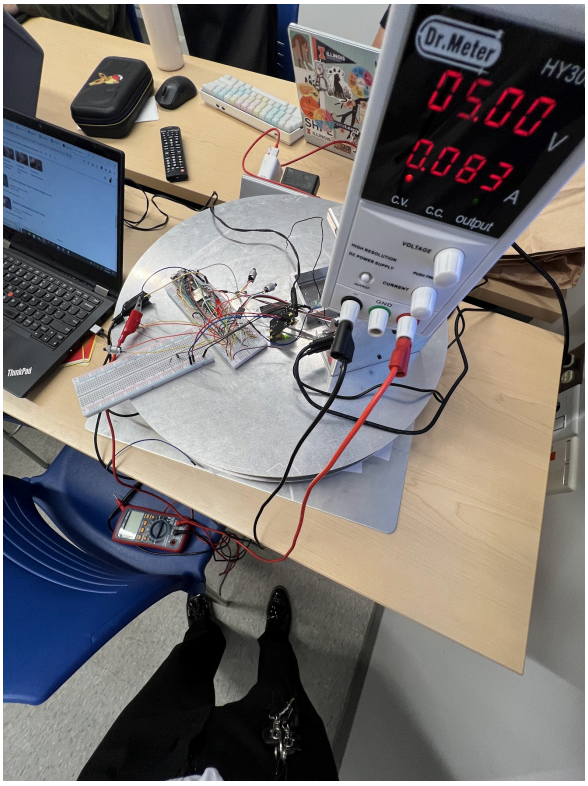

Figure 10: 300maH verification

3.3. User Interface Subsystem

Requirements	Verification
• Minimum response time of 0.5 seconds when switch is activated by user	• Onboard test circuit with led will verify if switches and the software can display the light up of the led within 0.5 seconds
• System must be able to halt within a 1 second	• When running, press switch and measure time to stop for all motors.

The user interface subsystem successfully met the minimum response time requirement of 0.5 seconds when activated via a switch or button, as demonstrated in testing configurations. This performance underscores the system's ability to quickly respond to user inputs, enhancing the safety and usability of the product. Although the full testing to verify the system's ability to halt within one second was not extensively conducted, preliminary results indicate that the quick response times observed are promising for achieving this requirement in further tests.

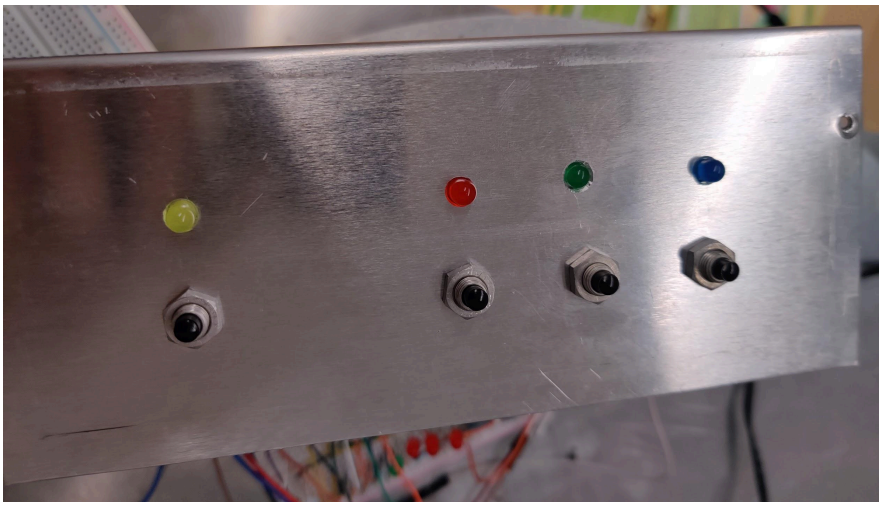

Figure 11 : User Interface setup

3.4. Camera System

Requirements	Verification
• The pi should receive a consistent camera feed from the Pi Camera 2	• Pi camera 2 should be properly connected to the pi. • The video feed should appear on the screen and react to new objects in range of the camera at a reasonable frame rate and resolution.
• Pi should be able to process video feed from the camera in real time	• Video feed should appear after running the software. • Data parsed from the feed should

	appear, be correct, and updated frequently.
• Raspberry pi should be able to identify colors with a 90% accuracy	• Place cards into the slot to be identified using the camera subsystem and check to see if the color is correctly identified 90% of the time.
• Camera subsystem should be able to send parsed data from the camera feed to the ESP32 using i2c	• The SDA, SCL, and ground pins of the raspberry pi should be connected to their corresponding counterparts on the esp32. • Color data in the form of a character byte should be found on the serial output of the esp32. This byte should match the output of the raspberry pi.

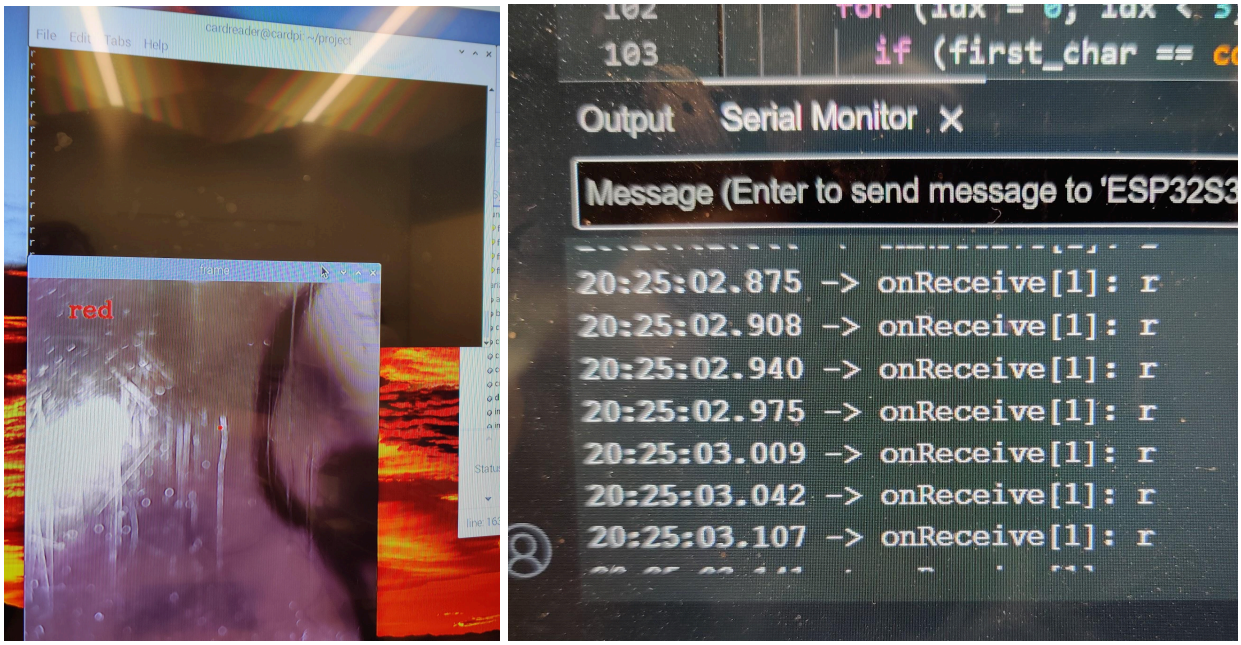

Figure 12 : Images of functioning real time Video and I2C communication

Above are images showing the verification of the software requirements. The bottom window of the left image shows the video feed with real time processing. There is text stating “red” showing that the average hue value around the red dot is within the value that would be considered red. The top window in the left image shows a line of “r”. This is the code printing to the terminal the character byte it’s going to send to the ESP32. The image on the right is showing the byte received from the pi being printed to the serial monitor.

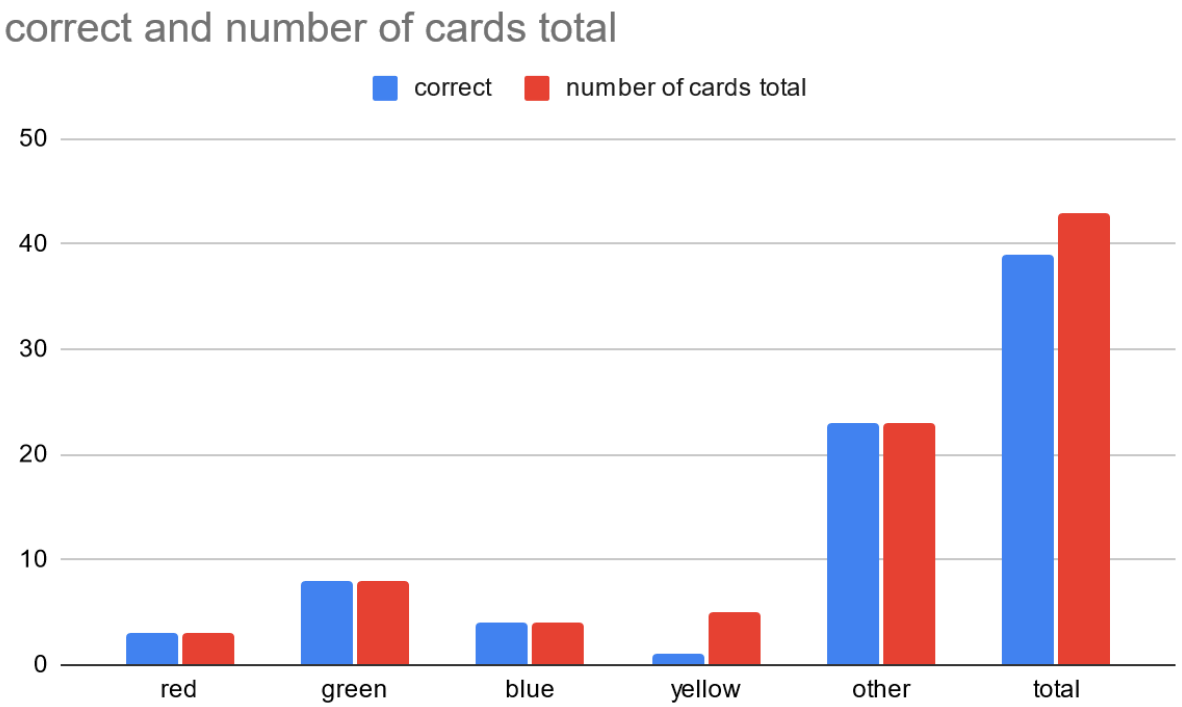

Figure 13 : Chart of Color Detection Accuracy

The figure above shows the verification of the accuracy requirement. Through testing around 43 cards, we found that accuracy was 100% when it came to red, green, blue, and other values. This, however, is not the case with yellow. We were using an LED to light up the viewing window of the camera. We did not realize that the LED had a blue tint that caused yellow cards to be read as green. So we ended with an overall accuracy of 90.697%.

3.5. Control Subsystem

Requirements	Verification
• The ESP32 should be able to receive data from the pi through i2c.	• The SDA, SCL, and ground pins of the raspberry pi should be connected to their corresponding counterparts on the esp32. • Data should be able to read through the serial out and match what the camera subsystem sees
• The ESP32 should be able to use input from the User Interface to change the categories cards are going to be sorted by.	• After giving user input, the cards should be sorted by the machine into groups based on the input. If the user pressed buttons for only red cards, the cards will be sorted into a red card pile and an “other” card pile.

For the control subsystem, the primary goal was ensuring that the ESP32 could receive data from the Raspberry Pi via I2C, a requirement that was successfully verified during the testing phase. The test results, illustrated in the camera system photos, showed that the ESP32 was capable of receiving a byte of data from the Raspberry Pi, confirming the functionality of this data transmission method. However, the capability of the ESP32 to alter sorting categories based on user input from the interface was not tested and remains an area for future development and verification to fully realize the system's dynamic sorting capabilities.

Costs

4.1 Parts and Cost

Description	Manufacturer	Quantity	Price	Link
Continuous Rotation DC Motor Servo Motor	Parallax Inc.	1x	$19.95	https://www.digikey.com/en/products/detail/parallax-inc/900-00008/1774454
Unipolar Stepper Motor Permanent Magnet Gear Motor Frame Size 4096 Step 5VDC	MikroElektronika	1x	$8.00	https://www.digikey.com/en/products/detail/mikroelektronika/MIKROE-1530/5724295?s=N4IgTCBcDa4BwCMCeArAtAFjgAgM4BcBTAB2wFsB7fCgJxAF0BfIA
Raspberry Pi 4 Model B 8GB		1x	$75.00	https://www.pishop.us/product/raspberry-pi-4-model-b-8gb/?src=raspberrypi
Raspberry Pi Camera Module 3		1x	$25.00	https://www.pishop.us/product/raspberry-pi-camera-module-3/
RF TXRX MODULE BT PCB TRACE SMD	Espressif Systems	1x	$3.35	https://www.digikey.com/en/products/detail/espressif-systems/esp32-s3-wroom-1-n4r8/16162637
IC REG LINEAR 3.3V 1A TO252-2	Diodes Incorporated	1x	$0.44	https://www.digikey.com/en/products

				/detail/diodes-incorporated/AZ1117CD-3-3TRG1/4470979
IC REG LINEAR 5V 1A TO252-2	Diodes Incorporated	1x	$0.47	https://www.digikey.com/en/products/detail/diodes-incorporated/AZ1117CD-5-0TRG1/4570580
LED GREEN DIFFUSED 3MM ROUND T/H	Würth Elektronik	4x	$0.17	https://www.digikey.com/en/products/detail/w%C3%BCrth-elektronik/151031VS06000/4489988
SWITCH PUSH SPST-NO 3A 120V	E-Switch	7x	$2.39	https://www.digikey.com/en/products/detail/e-switch/RP3502ARED/280448
TRANS 7NPN DARL 50V 0.5A 16DIP	Texas Instruments	1x	$0.47	https://www.digikey.com/en/products/detail/texas-instruments/ULN2003AN/277624
RES 510 OHM 5% 1/4W AXIAL	YAGEO	4x	$0.10	https://www.digikey.com/en/products/detail/yageo/CFR-25JB-52-510R/2306
CAP CER 10UF 16V X5R 0805	Samsung Electro-Mechanics	1x	$0.10	https://www.digikey.com/en/products/detail/samsung-electro-mechanics/CL21A106KOQNNNE/3886754
BATTERY PACK LI-ION 7.4V 18650	Dantona Industries	1x	$17.87	https://www.digikey.com/en/products/detail/dantona-industries/L74A26-2-1-2W/13692635
		Total Material Costs:	$168.46	

Adding together labor and material costs, the total cost of the project adds up to **$34,188.46.**

4.2 Labor

Assuming the average salary provided by the University of Illinois for computer engineering graduates of $109,176. We can assume an hourly wage of $50 an hour. With this, we can calculate the total labor for all partners to be:

($50/hour) x 2.5 x 84 hours x 3 members = $31,500

Based on estimates from the machine shop's quoted hours of 45 to complete the project with an hourly rate of $56 an hour, we can calculate the cost of the machine shop's labor:

($56/hour) x 45 hours = $2,520

Conclusion

5.1 Accomplishments

The detection of cards through openCV was surprisingly accurate despite suboptimal lighting and focal distance. OpenCV was able to correctly identify every non-foil card with 100% accuracy using feature detection and was able to identify card colors with 90% accuracy.

Communication between the pi and the ESP32 through I2C was quick and effective at transmitting data. The data from the pi got to the ESP32 almost instantly. Since the data was being updated in real time, it made it easy to apply moving the motors without many delays.

The logic for the ESP32 was correct despite the overall machine not working. Given a character byte of data, the ESP32 was able to spin a servo motor to the correct position and a DC motor by a specific amount. The problem being different power requirements of different motors.

5.2 Limitations

For this device, the main limitation was the power as mentioned before, not even the modular power supply could power both servo and DC motors. This didn't allow us to complete the project in a way we as a team wanted. We were hoping to at least get a theoretical system working since mechanically, there were faults with the card and motor mount that didn't allow for a steady flow of cards to be outputted. Those limitations would have to be worked through with the machine shop. Not only this but the camera rigging was to close the plexi glass view window in the card holder. This didn't allow for the Raspberry Pi cam to properly view the card. Overall, the power and mechanical limitations drastically limited the scope of what we could get done and with little time to adjust, as a team prioritized the aspects that worked.

5.3 Ethical Considerations

In the development of the ATCS we were aware of the concerns and safety a device such as ours could pose. We therefore chose to adhere to ethical codes established by the Institute of Electrical and Electronics Engineers (IEEE) and Association for Computing Machinery(ACM). Both of which stress the importance of integrity, fairness, and respect. By following these codes, we can ensure our device not only enhances user safety and welfare but also encourages honest and transparent work("IEEE Code of Ethics," 2023; "ACM Code of Ethics and Professional Conduct," 2024). By fostering an honest culture of honest criticism, thorough documentation, and acknowledging the limitations of the design, which supports continuous improvement and collaboration.

Safety concerns are paramount, with the ATCS design focusing on minimizing risks through comprehensive testing and secure enclosures for all mechanical and electrical components. We incorporate fail-safes like pause buttons But having the servo be limited to 170 ensured that no snaring of the cables or hands was possible. More could be done to ensure safety by enclosing the layers with a drape and implementing bins for the user to take out in the future.
Lastly, compliance with international safety standards such as ISO 12100 for machinery safety and ISO/IEC/IEEE 29119-1 for software testing guides our risk assessment and mitigation strategies, ensuring that the ATCS adheres to rigorous safety and operational benchmarks ("Safety of machinery," 2010; "ISO/IEC/IEEE 29119-1," 2013). These standards allowed us to structure our approach to identifying potential hazards and testing software conditions that could bear a wide range of potential scenarios.

5.4 Future Work

Moving forward the goal would be to focus on making the ATCS a product that is safer and more desirable to the target consumer. In order to do this, the ATCS would need to be faster, more accurate, safer, and have more robust features to make it a worthwhile investment.

This would mean a stronger card dispensing motor and a smaller card gap to make the speed and accuracy. For safety we would need to redesign the power subsystem to have more systems in place to prevent shorts. We would also need to create a PCB that can provide power to every subsystem to increase reliability and make things less exposed to the user. We would also need to create a more intuitive subsystem that implements safety features like start, stop, and reset buttons. Finally, we would also try to redesign the camera mount and viewing window to allow for feature detection. This would allow for cards to be further sorted by set, type, value, and price. Overall the system would have to function as intended and then some. Having these features ready for users would ensure satisfaction in our customers and complete our responsibility as engineers.

Work Cited

ISO/IEC/IEEE 29119-1, Software and systems engineering—Software testing—Part 1: Concepts and definitions, 1 September 2013, https://wildart.github.io/MISG5020/standards/ISO-IEC-IEEE-29119-1.pdf. Accessed 8 February 2024.

",." *IEEE Code of Ethics*, 19 May 2023, https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed 8 February 2024.

"ACM Code of Ethics and Professional Conduct." *Association for Computing Machinery*, https://www.acm.org/code-of-ethics. Accessed 8 February 2024.

Lindner, Jannik. "Trading Card Industry Statistics [Fresh Research] • Gitnux." *Gitnux*, https://gitnux.org/trading-card-industry-statistics/. Accessed 8 February 2024.

"Safety of machinery — General principles for design — Risk assessment and risk reduction (ISO 12100:2010)." *nobelcert.com*, 9 October 2010, https://nobelcert.com/DataFiles/FreeUpload/BS%20EN%20ISO%2012100-2010.pdf. Accessed 8 February 2024.

"Trading Card Market Size, Share, Growth | Forecast - 2031." *Business Research Insights*, https://www.businessresearchinsights.com/market-reports/trading-card-market-102964. Accessed 8 February 2024.