ECE 445 Spring 2025
Senior Design Laboratory
Final Report

Automatic Card Deck Sorter

Team 37
Kyle Mahler - kmahler3
Rocky Daehler - walterd2
Alfred Hofmann - alfredh2
May 7, 2025

Abstract

The following report details our Automatic Card Sorter project. The goal of this project is to be able to automatically sort a deck of cards into two piles for games such as Euchre or Small Hand Poker which don't need every card. This report discusses the design of this project, as well as the successes and issues we encountered with our final product. Additionally, it goes into the requirements and their verification for all subsystems, the costs associated with making this project in a professional setting, and discussion of what could be changed in the future or when restarting the project.

Contents

1 Introduction....1
1.1 Problem.... 1
1.2 Solution....1
1.3 Functionality....1
1.4 Subsystem Overview....2
1.4.1 Block Diagram....2
1.4.2 Card Recognition Subsystem.... 2
1.4.3 User Interface Subsystem....3
1.4.4 Sorting Subsystem....3
1.4.5 Control Subsystem....3
1.4.6 Power Subsystem....3
2 Design....4
2.1 Design Procedure....4
2.1.1 Card Recognition System.... 5
2.1.2 User Interface.... 5
2.1.3 Sorting System....6
2.1.4 Control System.... 6
2.1.5 Power System....7
2.2 Design Details....7
3 Verification.... 9
4 Costs....10
5 Conclusion.... 12
5.1 Summary of Results and Future Work....12
5.2 Ethics and Safety....12
6 References.... 13
Appendix A HardwareFigure 5: The PCB layout of our final PCB design....14
Appendix B Requirement and Verification Tables....16

1 Introduction

1.1 Problem

For centuries, card games have been a staple of entertainment. With just the same standard 52 card deck, hundreds of different card games have been produced over this time. However, in some of these games, there may be distinct and precise rules about setting up and managing the deck. For example, Euchre only uses the 9s, 10s, Jacks, Queens, Kings, and Aces, meaning players must manually sift through the deck before playing. Organizing the deck before playing games of this nature can be extremely tedious and time consuming. Players want to spend their time playing the game, not on the preparation of the game.

Beyond game-specific needs, many households, casinos, and clubs face the issue of reorganizing mixed or shuffled decks. Again, rearranging the cards back into a sorted order can take a long time and is by no means an exciting task. In a competitive setting, it may be essential to maintain a sorted deck before play to ensure fairness. At places with the need for a large number of decks to be sorted, the need for this process to be automated scales up drastically.

1.2 Solution

To address the inefficiencies of manual card sorting, we propose an Automatic 52-Card Deck Sorter. This device will quickly and accurately organize a mixed deck into an order specified by the user. This solution eliminates the need for players to manually separate cards for games like Euchre, where only a subset of the deck is used. The sorter will incorporate a card recognition system to identify each card and a mechanical sorting mechanism to place them in the correct order efficiently. Additionally, a PCB based control system will manage the identification and sorting process, which will ensure accuracy and reliability. By automating this task, the device saves time, reduces human error, and enhances convenience for casual players and competitive tournament organizers alike. Whether preparing for a game, ensuring a properly ordered deck, or simply avoiding the hassle of manual sorting, this system provides a reliable and efficient way to manage playing cards, making it a valuable tool for both home and competitive settings.

1.3 Functionality

The high-level requirements of our project are as follows:

- The camera can recognize the cards by suit and rank. It can recognize and sort 1 card in **4 seconds**, which translates to **3 minutes** for a 52 card deck
- Cards are successfully sorted into two piles, a ‘Used’ pile and an ‘Unused’ pile. The system goes until all cards are sorted into one of those two piles, and will automatically stop once there are no more cards to be sorted.
- The system can detect and display a warning sign or an error message when something goes wrong. It will do this when the same card is detected twice in a row (indicating a jam), a card is unrecognized by the camera, or the deck is incomplete.

1.4 Subsystem Overview

1.4.1 Block Diagram

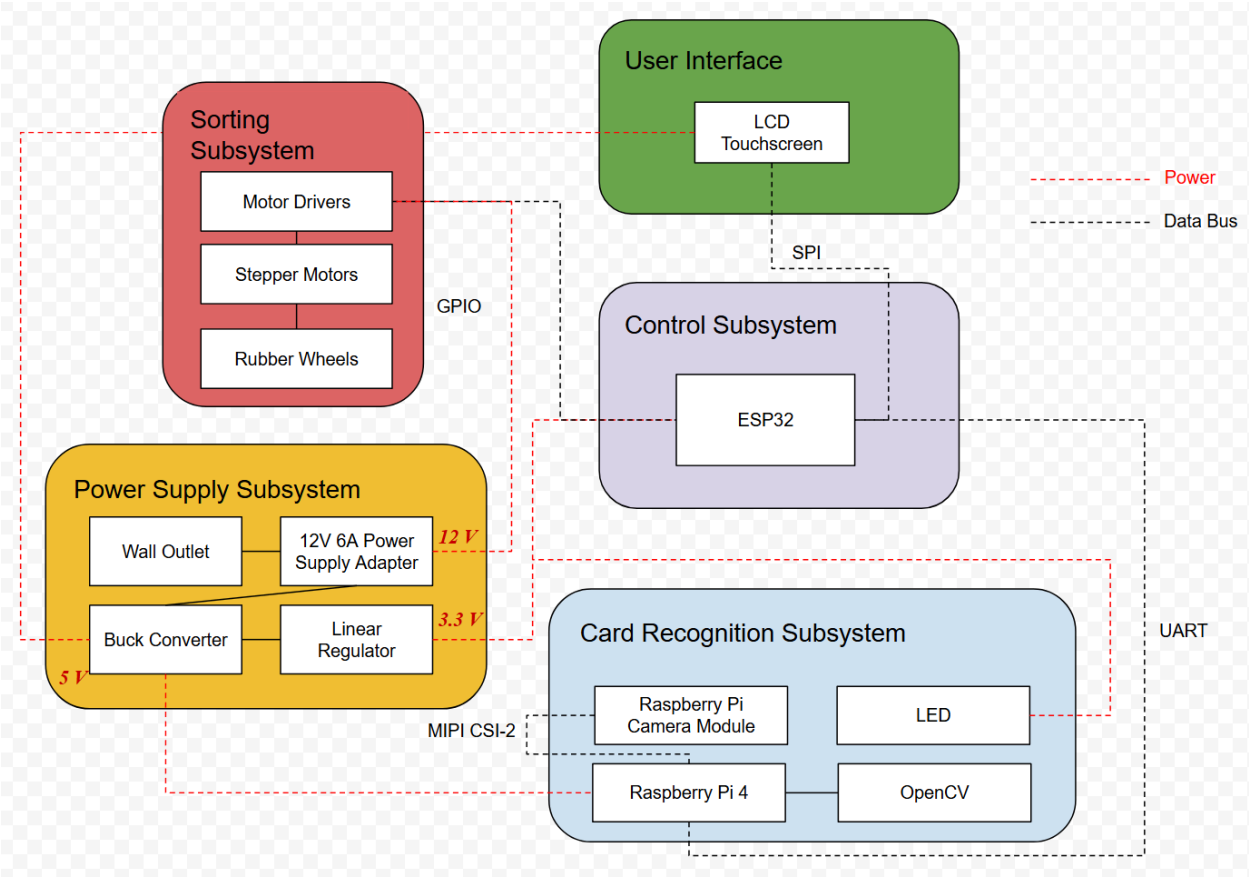

Figure 1: Our block diagram, showing our five subsystems and their connections.

Figure 1 shows the block diagram for our project. There are 5 different subsystems connected through two main communication protocols, Universal Asynchronous Receiver Transmitter (UART) and Serial Peripheral Interface (SPI).

1.4.2 Card Recognition Subsystem

The Card Recognition subsystem, upon receiving a signal from the control system that a new card has appeared, takes a picture of the upper right corner of the card. It will then send that image to the Raspberry Pi to be analyzed. This process gets repeated until the camera sends an image of there being no more cards.

1.4.3 User Interface Subsystem

The User Interface is a small screen with an assortment of buttons; Up, Down, Select, and Start. When the system is powered on, the screen shows a menu with a list of different games which have been preloaded for the system to sort into. Using the buttons, the user can navigate to their game of choice and hit the select button. Some examples of these games are Euchre, Short Deck Poker, or Crazy Eights. Then, when the Start button is pressed, the system begins sorting the cards. While sorting, the screen displays any pertinent information or errors - such as if a card jam is detected, if any cards are unrecognizable, etc.

1.4.4 Sorting Subsystem

The sorting system is responsible for moving the cards in a way to ensure they are appropriately sorted. This is done by placing the deck of cards on the top of two rubber tires, powered by two motors that slide each card to one pile or another through a slot roughly the size of one card. A metal cap is placed on top of the deck to add weight to the deck. This ensures that as the deck thins out weight and friction with the rubber tires is not an issue.

1.4.5 Control Subsystem

The control system is responsible for directing traffic within the device. It works closely with the user interface and the Raspberry Pi to receive information about the card type and game in order to make a decision about which stack the card should be sorted into. The control system also works closely with the sorting system, essentially telling it where each processed card should be moved to. Finally, the control system communicates with the UI early on in this whole process to store which cards we care about and which we do not.

1.4.6 Power Subsystem

The power system delivers power to all of the other subsystems. It uses a wall outlet power adapter to get 12 V for the motor drivers, a buck converter to bring that down to 5 V for the Raspberry Pi and LED strip, and a linear regulator to finally bring that down to 3.3 V for the microcontroller. This ensures that each component gets the voltage that it needs to operate without negatively affecting the performance of the other components/subsystems.

2 Design

2.1 Design Procedure

We were able to get help from the Machine Shop to come up with a design for our project. This final design involved three chambers in a line- cards are loaded into the top of the central chamber and then sorted into the chambers on either side.

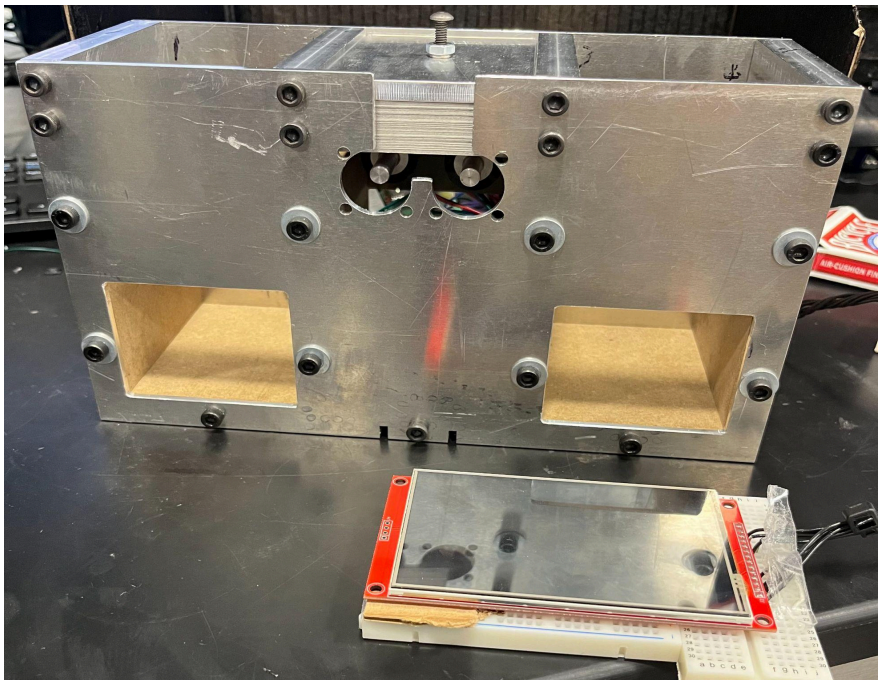

Figure 2: The Project on the day of our final demonstration.

Figure 2 shows the final product on the day of our demonstration. Figure 3 shows a multiview projection for the device as designed by Skee Aldrich, who built this project in the machine shop. Note that the final design has the touchscreen attached via a cable, so it can be accessed from any side of the device. Additionally, there is a weighted metal plate that gets placed on top of the cards which is not present in Figure 3.

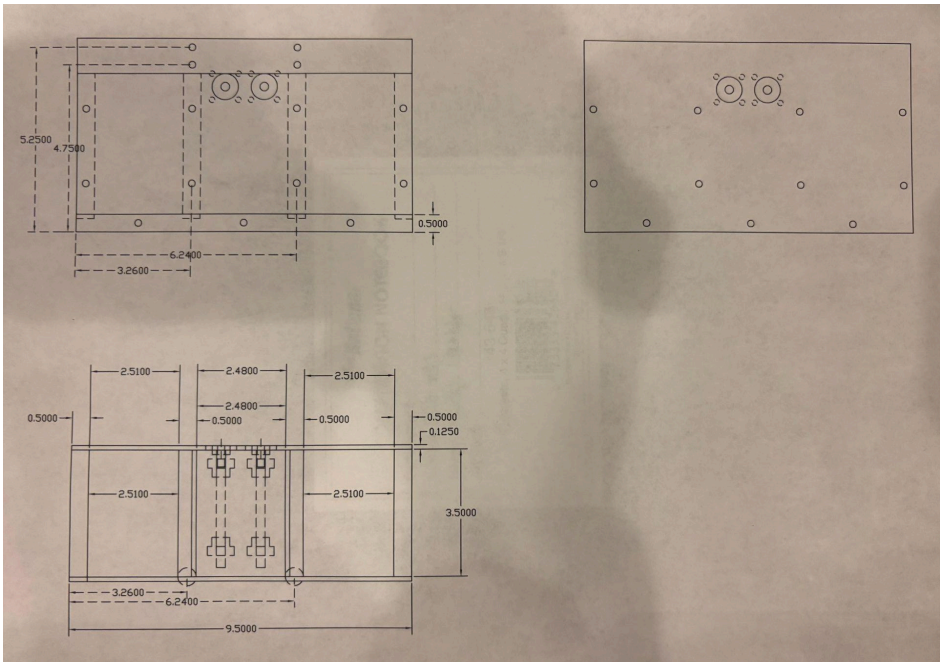

Figure 3: A multiview projection of Skee Aldrich's design for the project.

2.1.1 Card Recognition System

The Card Recognition system utilizes a Raspberry Pi module with a corresponding Raspberry Pi camera module. It also includes an LED strip to provide lighting in the central compartment of the device to illuminate the card suit and rank. The camera sits at the bottom of the middle compartment, pointing upward to target the upper right corner of the bottom most card to capture the suit and rank. This is done through a Python program on the Raspberry Pi based off of a similar project done by EdjeElectronics [1]. First, a rectangular box for the region of interest (the card's rank and suit) is established. This is then preprocessed by greying, blurring, and thresholding the image. The region of interest is split into an upper and lower half for the rank and suit respectively, and the largest contour is found in each half. This contour is then cropped and compared to pre-existing templates of all the suits and ranks, and finds the correct match. If not enough pixels are matched to any of the templates, then the card is deemed "Unknown". Once the card is identified, the program listens for a UART request from the microcontroller and sends a bitmask corresponding to the appropriate card. This process goes on indefinitely until power is lost.

2.1.2 User Interface

The user interface uses a touchscreen to communicate information to the user and to allow the user to control the system. Specifically, we use a Hosyond 4.0 inch SPI Module, which is a Thin Film Transistor (TFT) touchscreen with an ST7796 SPI chip. It has a 480 by 320 pixel resolution and a 4 wire SPI interface [2], which is connected to the ESP32 microcontroller on our device.

On the software side, the User Interface uses the TFT_eSPI library to facilitate the drawing of images and text to the screen. Additionally, this library has functions to help with receiving information about when and where the touch screen gets touched [3]. Initially, we had intended to use buttons and a non-touchscreen, but opted for the touchscreen due so that we could change both the amount and position of input options the user has at any given moment. The versatility of the adaptable interface allows for clearer communication of options and information to the user, and only requires one additional chip select output from the ESP32, making it more efficient in terms of Microcontroller pin usage than having multiple buttons.

2.1.3 Sorting System

The sorting subsystem consists of three main components, one being two bipolar stepper motors (1528-1062-ND from Digikey)[4], and the other hardware component being its drivers. For the drivers, we landed on using the DRV8825 from Pololu[5]. Finally, the ECE machine shop provided axles and rubber wheels to connect to the stepper motors, as well as the shell of the entire device in which the rest of our components would rest. As for software used for the sorting system, the two DRV8825's are controlled using the AccelStepper library[6]. For our design, we used an extremely simple function to sort the cards one way or another by simply rotating each of the motors by specifying a maximum speed, degrees of rotation, maximum acceleration, and a direction.

2.1.4 Control System

The Control System uses an ESP32S3-WROOM-1 microcontroller to facilitate the sorting of the cards. This subsystem is connected to all other subsystems in the project, taking as inputs touch signals from the touchscreen and card rank and suit messages from the Raspberry Pi. Based on these inputs, the control system sorts left or right based on the selected game, requests a new reading from the Raspberry Pi once a card has been sorted, and raises errors for the screen to display based on the recognized cards. To sort the cards, our project uses a bitwise AND between a bitmask representation of the cards used for the selected game and a one hot encoded representation of the card. Figure 4 shows an example of these encodings being compared.

Figure 4: A diagram showing how the control system compares a card signal to the games' bitmask

If the result of the comparison is 0, then the card does not fall in the range accepted by the selected game, and it's sorted into the unused pile. Otherwise, the card is sorted into the used pile.

The largest issue we encountered with the Control System is that we were unable to program the ESP32 on our Printed Circuit Board (PCB). Using arduino to program the ESP32 on the PCB as we had done for the ESP32 Devkit resulted in an exit status 2 error every time we tried, with the exception of one time when we were able to successfully program the board, though we could not reproduce this. Our best guess is that the timing of when the ESP32 needs to be entered into boot mode is very specific, based on the fact that we were able to successfully program once and the timing of pressing the buttons is the only thing that changed. Notably, using RTS and DTR also did not solve this issue. If we were to order another PCB for this project, including a USB connection for the ESP32 would likely make programming the microcontroller easier as this is the method we successfully used for the ESP32 Devkits, and the ESP32-S3 allows for booting via a USB-OTG connection [7]. Initially we had not included this because we were unsure whether our ESP32 model would support USB-OTG booting and the course wiki recommends using the USB-UART connection.

2.1.5 Power System

The power system deals with distributing power at the desired voltages to the various parts of our project. Specifically, the stepper motors require a 12 V signal, the Raspberry Pi requires a 5 V signal, and all other components require a 3.3 V signal. The system as a whole gets power from a wall outlet, which gets converted to 12 V by an external power cable. Then, the power system utilizes a buck converter breakout board to convert the 12 V signal to a 5 V signal, utilizing a shared ground. An LM1117 Linear Regulator is then used to convert the signal down to 3.3 V. Initially, we had planned on using battery power to make the device more portable, but eventually decided to use a wall outlet instead. While battery power would have made the project more portable, portability is not related to any of our high level requirements and is not necessary for most use cases of this project. There were concerns that a battery would deplete

quickly enough that replenishing the battery would become annoying, and the physical device needs to be somewhat large due to other subsystems, so portability ends up being something that wasn't feasible with the battery either.

2.2 Design Details

In designing our project, we created a PCB design that we were unfortunately unable to get most of our project incorporated onto. The final iteration of our PCB (layout and schematic included in Appendix A) is smaller than the size of a playing card (63.5 mm by 88.9 mm) so as to be able to fit in the bottom portion of the central chamber, under the stepper motors' axles. It includes the circuitry to allow for programming the ESP32 using a UART connection with DTR and RTS, a linear regulator, and pinouts for the variety of breakout boards and peripherals the project needs. The stepper motors required fine tuning for their maximum speed, acceleration, and the amount of steps each motor would rotate when sorting left or right. We eventually set these at 110 for a maximum speed and 1000 for the acceleration. The wheel on the side the card was going towards would move plus or minus 187 steps, and the wheel on the side the card was moving away from would move plus or minus 73 steps (with plus or minus indicating the direction of spin).

3 Verification

Testing and verification for each specific subsystem was conducted as each was completed. As can be seen in the requirements and verification table, timing requirements were very important to us, and each of these timing requirements were satisfied for every subsystem.

Other requirements for each of the subsystems are more specific for its specific task. For the camera, we were able to identify the correct card 99% of the time, 4% higher than our initial goal. The UI successfully displayed game options and error messages when appropriate, as well as sending and receiving signals efficiently and accurately. Likewise, the control system efficiently and accurately sent and/or received signals accurately and efficiently to each of the subsystems that required it. Finally, the power system worked flawlessly, delivering power to each of the subsystems within our needed range.

The sorting subsystem, however, plagued our design by failing to move just one card more than 85% of the time. Although jams and slipping mistakes were rarely made by the sorting system, the amount of two card pushes was enough to cause major problems when running the device as a whole. Below shows the number of errors among thiry different trial runs. High success and high failure numbers typically were recorded on the same day.

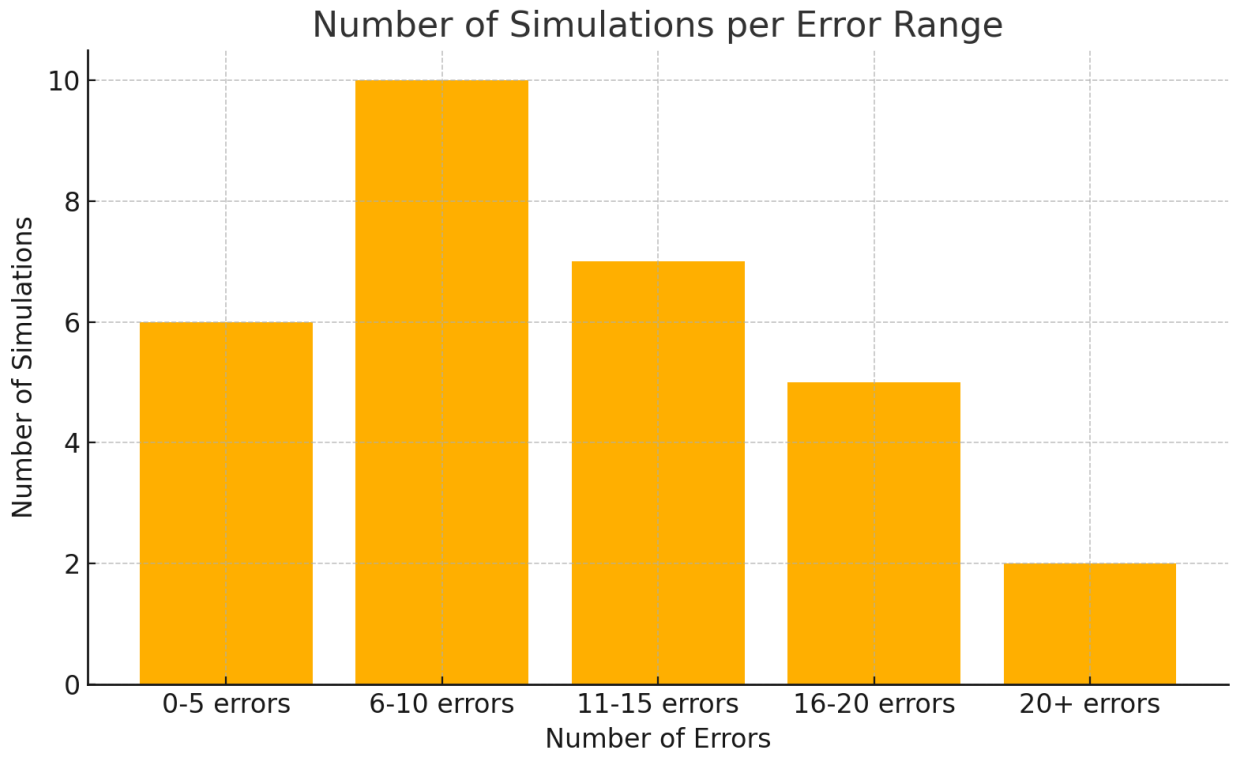

Figure 5: Error Frequency Among 30 Trials

4 Costs

Our cost analysis consists of three components. The first is the part list. This includes all items that we had to order to assemble the device, whether purchased from online vendors or ordered within the ECE building. Additionally, the machine shop costs are included in the cost analysis. We reached out directly to ask for a quote. Lastly, we include the total labor cost among all group members.

4.1.1 Part List

Description	Manufacturer	Part Number / Model	Qty.	Extended Price	Link
Nema 11 Stepper Motor 28mm	Iverntech	Nema 11	3	$51.27	Link
TFT Touch Screen LCD Display Module	Hosyond	ST7796S	1	$19.79	Link
Raspberry Pi Camera Module V2-8 Megapixel	Raspberry Pi	RPI-CAM-V2	1	$14.00	Link
DRV8825 Stepper Motor Driver Carrier	Pololu	DRV8825	2	$25.90	Link
LM1117IMPX-3.3/NOPB Linear Regulator	Texas Instruments	LM1117	1	$1.01	Link
LM1117T-3.3/NOPB Linear Regulator	Texas Instruments	LM1117	1	$1.59	Link
DFR0379 Buck Converter	DFRobot	DFR0379	1	$4.90	Link
Jack Plug Adapter Barrel Connector	California JOS	N/A	1	$3.97	Link
USB Type C Connector Board	Teansic	IC354	1	$7.99	Link
12V 6A Power Supply Adapter	COOLM	YU1206	1	$14.59	Link
Raspberry Pi 4 Model B 8GB	Raspberry Pi	Model B	1	$74.99	Link
ESP32	Espressif Systems	ESP-WROOM-32	1	$8.99	Link

Microcontroller					
Total				**$228.99**	

Table 1: The part numbers, quantities and costs of all electronic items used in the project

Table 1 shows a list of all the parts used for this project, including quantities and total costs. Based on the information in this table, the total price of all parts used comes to $228.99.

4.1.2 Machine Shops Costs

The majority of the physical design is constructed by the ECE machine shop. The construction of the device is estimated to take 2 weeks working 7.5 hour days. The labor wages for the machine shop are $56.12 per hour. We can find the total Machine Shop Labor cost as follows:

10 days x 7.5 hours per day x $56.12 per hour = $4,209

Total cost for the machine shop labor: **$4,209**.

4.1.3 Labor Costs

All of the members in our group are computer engineering majors. To compute the total labor cost for this project, we will find the expected per hour wage of a computer engineering graduate and multiply it by the expected number of hours we are working on the project.

To find the expected per hour wage, the average starting salary for a computer engineer is $109,176 [8]. The total number of hours worked per year is 2,080. The expected per hour wage is calculated as follows:

$109,179 / 2,080 hours = $59.49 / hour

Total labor cost:

$59.49 / hour x 180 total hours x 3 people = $32,124.60

Total Cost of Project: $228.99 + $4,209.00 + $32,124.60 = **$36,562.59**

5 Conclusion

5.1 Summary of Results and Future Work

We were able to develop a system that correctly identifies and sorted cards based on a game selected by the user. The recognition of cards, determining how they should be sorted, and physical movement of a card all worked together well for individual cards. Unfortunately, there was a persistent issue with the second lowest card being moved when the lowest card was moved. This was likely due to friction between the cards being an inevitable occurrence, and unfortunately we were unable to adjust any parameters to remove this issue altogether. When we set our final parameters for the stepper motors, the system worked very well, but due to small factors such as the dirtiness of the cards or the dirtiness of the wheels, as well as environmental factors such as the humidity, the amount of friction between cards was inconsistent and we weren't able to consistently push only one card.

For future work, there are a couple different things that could be helpful. Using some sort of pneumatic system to add a small gap between cards would help alleviate friction, though it would require modification to the chamber that currently holds the input deck and would likely make the device bulkier. Alternatively, somehow making the motors' setting automatically adjust could work to eliminate this issue as well.

5.2 Ethics and Safety

In designing the Automatic Card Deck Sorter, we recognize the responsibility to uphold ethical standards as outlined in the IEEE Code of Ethics. While our device is intended to improve efficiency and convenience in card games, it is important to acknowledge potential misuse and mitigate risk, particularly in gambling or competitive play settings.

A major ethical concern is the possibility of our device being misused to gain an unfair advantage in gambling or competitive card games. To align with the IEEE Code of Ethics [9], we must ensure that our design does not facilitate deception or unlawful conduct. Specifically, Section I.4 emphasizes the importance of maintaining ethical behavior in professional activities and rejecting any form of corruption, including actions that could enable cheating. To uphold these principles, our system will be designed with transparency in mind, ensuring that it functions solely as a fair and unbiased card-sorting tool.

Moreover, Section II.9 of the IEEE Code highlights the responsibility to avoid causing harm to others, whether through direct actions or by enabling unethical behavior. This project should not be used in a way that compromises the integrity of games, damages reputations, or results in financial harm.

Since our card sorter is designed for convenience, it is difficult to use for other purposes. There are no extreme size restrictions or intentional hidden mechanisms that would be ideal for the unethical rigging of a card game, so unlawful use of the device is naturally discouraged. By

addressing these ethical concerns, we ensure that our technology aligns with professional integrity and responsible engineering practices.

5.3 Uncertainties

Uncertainties remain for this project as we never fully determined the cause of the sorting subsystem not working well. Some days it would work extremely well with less than 5 errors, other days it would provide an error for nearly every other card. What we can be certain of is that in order for the Automatic card sorter to work perfectly, a different design for the sorting subsystem was needed. Perhaps this could have been sorting from the top, using an airblast like the professional shuffling systems do, or another alternative that we are not aware of yet.

6 References

[1] EdjeElectronics, “OpenCV-Playing-Card-Detector”, Github, 2025 [Online] Available: https://www.google.com/url?q=https://github.com/EdjeElectronics/OpenCV-Playing-Card-Detector&sa=D&source=docs&ust=1746670032700031&usg=AOvVaw11IHx-euoPhlCtMRahXSN8 [Accessed May 7, 2025]

[2] LCD Wiki “4.0inch SPI Module ST7796” Lcd Wiki, 2019 [Online] Available: http://www.lcdwiki.com/4.0inch_SPI_Module_ST7796 [Accessed May 7, 2025]

[3] Bodmer, “Github - Bodmer/TFT_eSPI.” Github, 2025 [Online] Available: https://github.com/Bodmer/TFT_eSPI [Accessed May 7, 2025]

[4] Adafruit, “1.8º 42MM Torque Hybrid Stepping Motor”, XY42STH34-035A datasheet

[5] Texas Instruments, “DRV8825 Stepper Motor Controller IC”, SLVSA73F, Apr. 2010 [Revised Jul. 2014]

[6] swissbyte, “AccelStepper”, Github, 2025 [Online] Available: https://github.com/swissbyte/AccelStepper [Accessed May 7, 2025]

[7] Espressif "ESP32-S3 Series Datasheet Version 2.0" July 2021 [Revised Apr. 2025]

[8] “Salary Averages”, ece.illinois.edu. https://ece.illinois.edu/admissions/why-ece/salary-averages (accessed May 7, 2025)

[9] IEEE. “”IEEE Code of Ethics”.” (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 05/07/2025).

Appendix A Hardware

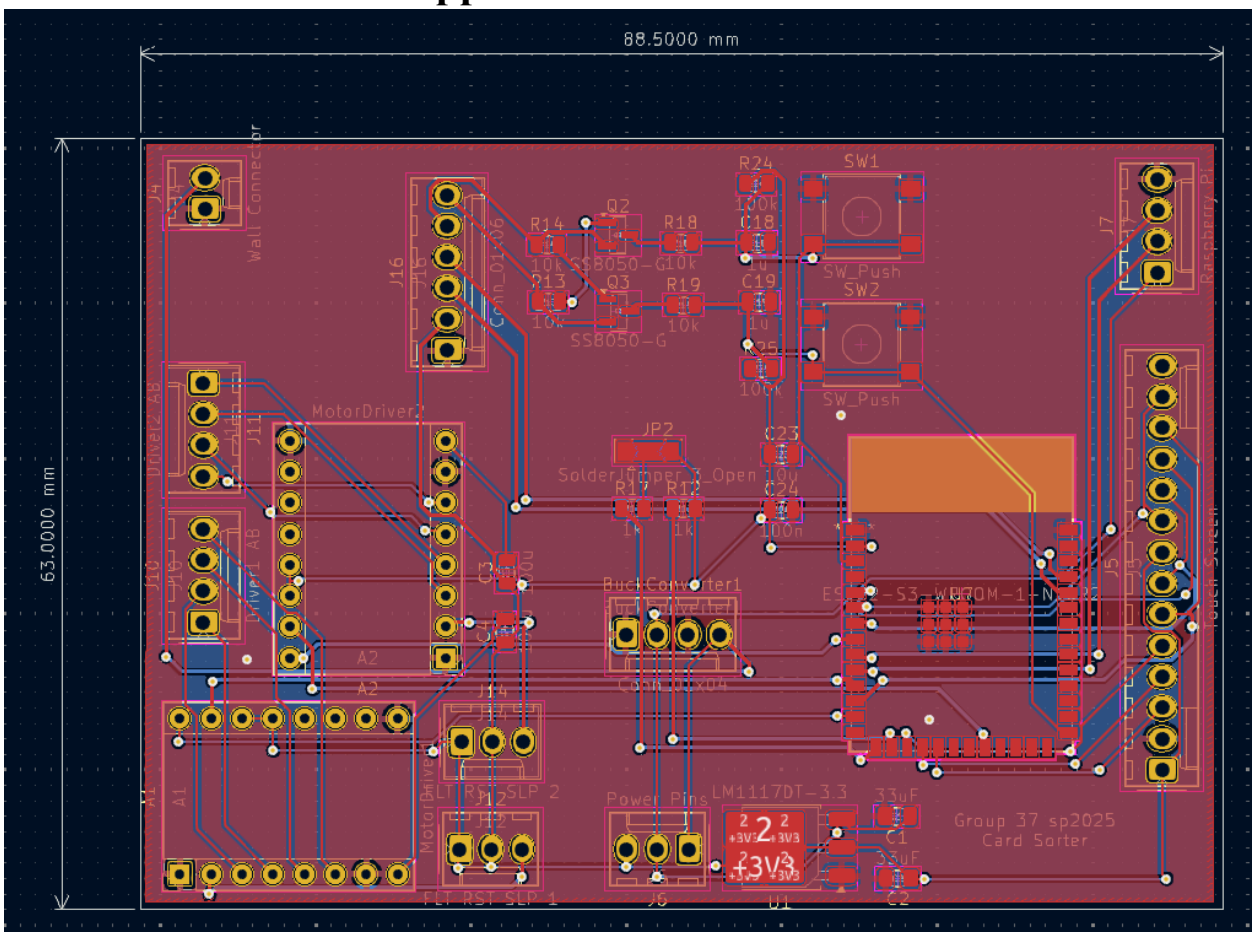

Figure 6: The PCB layout of our final PCB design

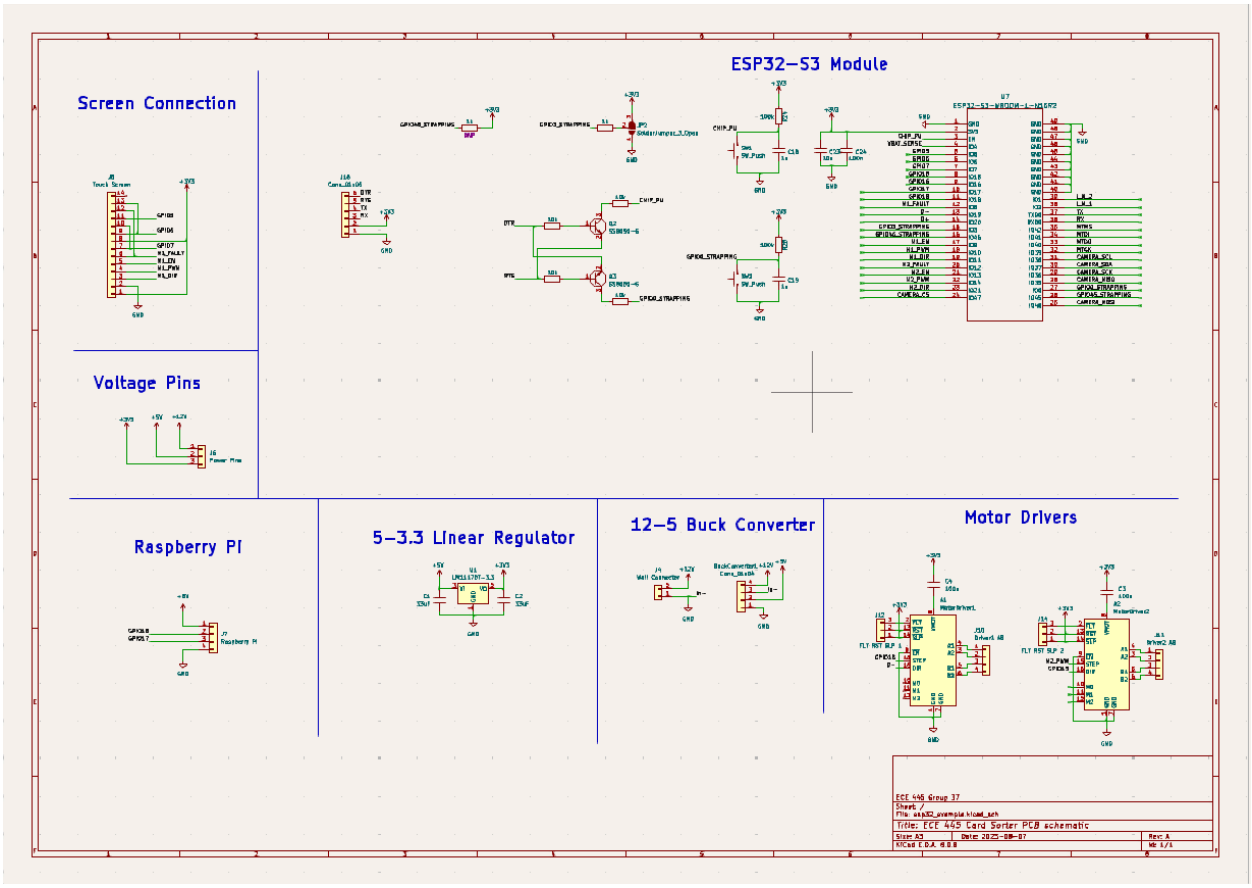

Figure 7: The circuit schematic of our final PCB design

Appendix B Requirement and Verification Tables

Camera System

Requirement	Verification
The Camera can take a clear picture of the card corner and send that card to the Raspberry Pi. The Raspberry Pi will identify the rank and suit of the current card, including labeling some cards as unrecognized, with 95% accuracy or higher.	1. Setup the camera pointing at a set of cards with the same spacing/dimensions as in the final design. 2. Have the camera identify all of the cards in the deck, including jokers. 3. Calculate what percent of the cards were identified correctly (jokers should be identified as unrecognized) and make sure that 95% or more were calculated correctly.
The Camera subsystem can turn an LED on within 1 second of sorting starting and turn the LED off within 1 second of the sorting ending.	1. Connect the camera subsystem up to the Control System. 2. Send a ‘sorting starting’ signal from the Control System and measure how long it takes the LED to turn on. Verify that it takes less than a second. 3. Send a ‘sorting ending’ signal from the Control System and measure how long it takes the LED to turn off. Verify that it takes less than a second.

Table 2: The Requirement and Verification table for the Camera System

User Interface

Requirement	Verification
The user can use the User Interface to select a game to sort the cards for. The User Interface will send a signal indicating which game was selected within 1 second of the button being pressed.	1. Connect the User Interface to the Control System. 2. Select a game on the User Interface. 3. Measure the time between when the game is selected on the User Interface and when the signal is received by the Control System. Verify that this takes less than 1 second. 4. Repeat this for all preloaded games.
When the user presses Start on the User Interface, a signal communicating what	1. Connect the User Interface to the Control System.

game/sorting has been selected will be sent to the Control System. This signal will be sent within 1 second of the button being pressed.	2. Press ‘Start’ on the User Interface. 3. Measure the time between when the button is pressed and the signal is received by the Control System. Verify that this takes less than 1 second.
The User Interface can display an error message when given a signal from the Control System. Specifically, the User Interface can display A Jam error message, An Unrecognized Card error message, or A Missing Card error message within 1 second of receiving a signal from the control system.	1. Connect the User Interface to the Control System. 2. Send an error signal to the User Interface from the Control System. 3. Measure the time between when the signal is sent and the error message appears on the User Interface. Verify that this takes less than 1 second. 4. Repeat this for all three error types.

Table 3: The Requirement and Verification table for the User Interface

Sorting System

Requirement	Verification
The Sorting System can move a card to either side based on an electronic input. It will move cards all the way into the selected tray, only resulting in a jam 15% of the time or less.	1. Connect the Sorting System to the Control System. 2. Send randomized sorting signals to the Sorting System. 3. Count how many times the system jams/fails to move a card all the way into one of the sorting trays. 4. Calculate the number of jams/number of cards and verify that this is less than 0.15
The Sorting System can move just the bottom card of a stack of cards without affecting the rest of the stack. It will move only 1 card 85% of the time or more.	1. Connect the Sorting System to the Control System. 2. Send randomized sorting signals to the Sorting System. 3. Count how many times the system moves multiple cards at a time. 4. Calculate the number of multiple card moves/number of move signals and verify that this is less than 0.15
The Sorting System will move a card fully into one of the trays within 2 seconds of receiving a signal from the Control System.	1. Connect the Sorting System to the Control System. 2. Send a signal to the Sorting System from the Control System and verify that the card is fully sorted within 2

	seconds of the signal being sent.

Table 4: The Requirement and Verification table for the Sorting System

Control System

Requirement	Verification
The Control System will give one of two signals to the Sorting System (indicating which direction to move the card) within 1 second of receiving the card's information from the Camera System.	1. Connect the Control System to both the Sorting System and the Camera System. 2. Send card information to the Control System from the Camera System. 3. Measure the time between the Camera System sending the information and the Sorting System Receiving the information, and verify that this is less than 1 second.
The Control System will recognize various errors and send a signal to the User Interface when they occur. The Control System accurately detects Jams (seeing the same card twice) 90% of the time or more.	1. Connect the Control System to the Camera System. 2. Create false jams by showing the camera the same card multiple times. 3. Calculate how many of these false jams the Control System accurately detects and verify that it is over 90%.
The Control System will recognize various errors and send a signal to the User Interface when they occur. The Control System accurately detects Missing Cards (detecting less than 52 cards in the deck) 90% of the time or more.	1. Connect the Control System to the Camera System. 2. Purposefully present the camera system with less than 52 cards and have the camera system go through all of these cards then stop. 3. Repeat this process 100 times. 4. Calculate how many times the Control System accurately detects the missing cards and verify that it is over 90%.

Table 5: The Requirement and Verification table for the Control System

Power System

Requirement	Verification
Provide consistent and accurate voltage to the other components, primarily a 12V ± 0.5V line to turn the motors, a 5V ± 0.5V, and a 3.3V ± 0.3V line for most other items.	1. Connect all subsystems to power, either to the 5V line or to the 3.3V line from the batteries or voltage regulator respectively. 2. Use a multimeter to verify that the

	12V line is within its tolerance range, the 5V line is within its tolerance range, and the 3.3V line is within its own tolerance range

Table 6: The Requirement and Verification table for the Power System