Efficient Card Shuffler with Cut Card Insert

Team 53

Steve Mathew [stevem4]

Alex Lo Faso [alofaso2]

Matt Garrity [garrity6]

TA: Aniket Chatterjee

ECE 445 Spring 2026

Abstract

The Efficient Card Shuffler with Cut Card Insert addresses limitations with commercial card shufflers by adding several new features such as increased deck capacity from 2 decks to 4-6 decks, automatic card tray extension and retraction, and the ability to insert a cut card into the shuffled deck at any adjustable penetration level. This mechanism is comprised of 6 subsystems which incorporate different power rails built around an ESP32 MCU, with communication to various infrared sensors, synchronous FSR servo motors, stepper motor, linear actuator, and other mechanical components. Testing and verification found that while a custom pulley system for the cut-card penetration, and individual sensor and motor responses worked according to the project's needs, the shuffling mechanism's success was limited due to certain manufacturing constraints. Additionally, the PCB went through multiple design and assembly iterations and could be further improved upon to be fully incorporated into the design.

Contents

1. Introduction

Problem

Whether it's taking a break from work, taking a trip to the casino, or just kicking back and spending time with friends, people love to play card games. Many card games require the need to continuously shuffle between rounds of play. This process can become extremely tedious over time not only for card dealers but game players as well. Automatic card shufflers have existed for a while, but a lot of them have very noticeable limitations. Some of these limitations include limited deck capacities, the requirement for manual retrieval of shuffled cards, and an overall tendency to be error-prone and unreliable. In addition to this, these machines tend to be relatively difficult to operate. Games like blackjack require multiple decks and frequent shuffling, depending on how many people are playing and how many decks you are using. Blackjack also requires the insertion of a cut card before play can begin. This cut card ensures randomness in the shuffle and fairness to the players in the game. There currently are not many automatic card shufflers on the market with an increased deck capacity, and there are none with a cut card insert feature. This results in having to put a deck through a shuffler multiple times in pieces and then manually insert the cut card outside of the shuffler, adding a lot of time to the shuffling process. A lot of shufflers also require constant pressing of the shuffle button, causing the overuse of motors as well as wasted energy. Many card shufflers available on the market require manual removal of the card tray from the bottom of the machine, which is often quite cumbersome and adds a good deal of time to the process of shuffling. These are all fundamental problems that will be addressed in this project.

Solution

The solution is to build a card shuffler that comes equipped with the ability to tackle the issues all at once at the push of a couple of buttons. There will be an increase in the deck capacity to six total decks, up from the typical two. In addition, a system that the user to set the exact level of penetration in a deck shuffle to insert a cut card using a dial will be integrated, ensuring even playing conditions for the player and fairness to the dealer. It would also have a one-time press button that shuffles the deck, taking away the need to push the button down the whole time adding extra strain to both the button itself and the motors it is turning on. These features all together lead to a total of six subsystems, namely card detection, deck shuffling mechanism, cut card insertion, card tray extension, power supply, and the control module. There will be one button input and one dial input available to the user. The dial input will control the depth of the deck at which the cut card will be inserted. The order of operation for the machine will be to load the cards into the card inserts on top of the machine, use the dial to set the cut card penetration level, then press the shuffle button to shuffle all the cards together. At the end of the shuffle operation, the tray will extend automatically with the shuffled cards. There will be a delay once the cards are shuffled in order to allow time for the user to insert the cut card through the slot. This allows for a seamless transition from two separate decks to one final shuffled deck.

High – level requirements list

The project's success will be measured against these three high-level requirements:

- With the press of the start button, the device successfully shuffles 6 standard sized decks without manual intervention. Shuffling motors halt motion once all cards have been shuffled. Optical sensors accurately detect if cards are present.

- The shuffled product is extended out in the product tray after shuffling and/or cut card insertion completion. Once the cards are received from the collection tray, the tray will be retracted automatically. Optical sensors accurately detect if cards have been retrieved.

- Cut card insertion slot moves in accordance with the dial. The dial is measured as a percentage of deck penetration ranging from 10-90%, allowing for desired insertion at any reasonable level of deck penetration.

2. Design

Block Diagram

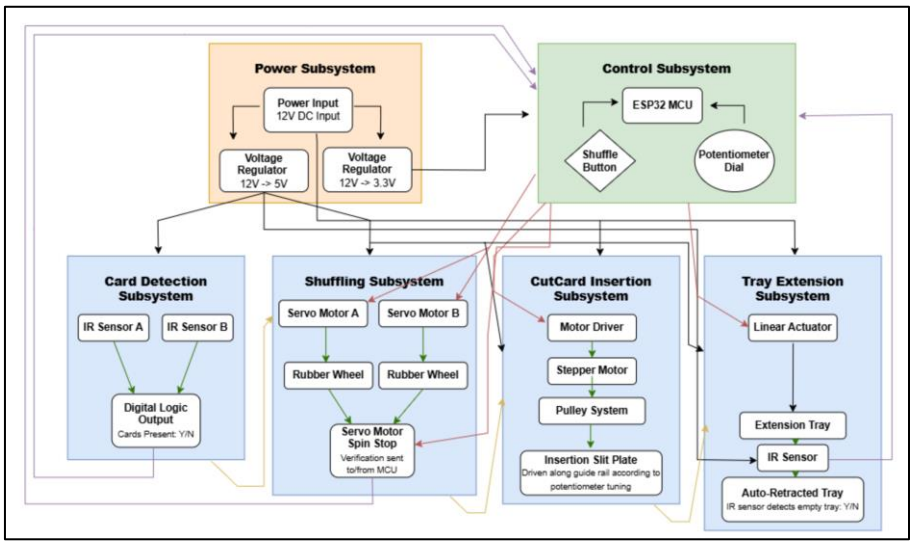


Figure 1: Block Diagram

- Black lines denote paths for power distribution to various subsystems.
- Red lines dictate digital signals from the MCU (control system) to various subsystems to trigger action items
- Purple lines feed data from subsystems to the MCU
- Yellow lines follow physical card progression from subsystem 1 – 4

Subsystems Overview

This device is composed of six subsystems that complete the full operation of the machine when integrated together. Figure 1 displays the electrical power + signal interconnectedness among the six subsystems, and a brief description of each is touched on below:

Card Detection Subsystem:

This first system is made up of two TCRT5000 reflective infrared sensor modules [2] and positioning them at the bottom of each of the pre-shuffle trays (the tray where the user will put the pre-shuffled decks of cards on). The sensor outputs are relayed to the ESP32 GPIO pins. When the sensors report an absence of cards, they read no reflection and communicate to the MCU that the shuffling

process either hasn't begun yet, or is complete, which feeds into the MCU's decision for the shuffling subsystem.

Deck Shuffling Subsystem:

This subsystem consists of two pairs of FS90R servo motors (continuous rotation) [6] fitted with wheels to allow for smoother transition with the cards. Each pair sits at the base of the 3D-printed pre-shuffle trays and is intended to spit out the cards 1-by-1, or at least approximately 1-by-1 for the most optimal shuffle pattern. The cards get collected at a collection tray below in the middle of the mechanism. The shuffling operation (when the servos rotate) only begins when the MCU reads that the IR sensors are at a LOW (cards are detected) and once the start operation button has been pressed. The servo motors follow the readings of the IR sensors to start and stop.

Cut Card Insertion Subsystem:

This system manages the vertical alignment of the cut card insertion plate. It is controlled by a 24 PPR (pulse-per-revolution) rotary encoder [7] that, when turned by the user a specific amount, will correlate to how much a stepper motor [8] will spin, and in turn pull the insertion plate up along the z-axis (height) of the shuffled card decks. This pulley system is a creative and untested design aspect when it comes to any card shufflers on the market. The insertion plate will travel along rails to ensure no jamming, and the stepper motor will be driven by a motor driver to ensure proper voltage supply.

Deck Tray Extension Subsystem:

The last of the physical subsystems, the deck tray extension and retraction, operates once all the shuffling operation has commenced and the shuffled card deck(s) are then able to be sent out of the body of the machine and extended for ease of access for the users. This is incorporated via an IR sensor module (the same as used in subsystem 1) mounted to the tray to track whether cards are or are not present on the tray. The tray extends via a linear actuator mounted to the bottom back of the tray which can push it forward and retract it once the user retrieves the cards (IR sensor reads high). The "polarity reversal" of the linear actuator is controlled by the MCU based on the IR sensor reading.

Power Subsystem:

The power supply is presented via a 12V battery input and needs to be stepped down to a 5V line and a 3.3V line. 12V is required for the motor driver (A4988) and the linear actuator driver (DRV8871), [10] while the 5V line is dealt to the servo motors, and the 3.3V line is sent to power the MCU and IR sensors. The power step-down on both the 5V and 3.3V line from the 12V was dealt with via adjustable MP1584EN buck converters [11] and required finetuning to match the output voltage that was sought.

Control Subsystem:

The control module was centered on the ESP32-S3-WROOM-1 MCU [4] when working on the PCB, and then to the ESP32-C6-WROOM-1 [5] Development module when transitioning back to the breadboard. The MCU was responsible for all the communication to the various peripherals, and almost all of the GPIO pins were made use of. Additionally, it housed the state machine architecture that we developed focusing on five states: IDLE, SHUFFLING, WAITING_CARDS, ACTUATING_OUT, and ACTUATING_IN.

Design Process

Design Alternatives

Our initial design involved the use of beam-brake sensors to detect jams in the shuffling mechanism. We later determined that these were unnecessary since the shuffle operation would be physically open and visible from the outside. In addition, these sensors overcrowded our design and would have clustered up the electronics in the shuffle mechanism. Removing these sensors freed up our physical design and made our electronics simpler while not sacrificing the operation of the machine.

The second alternative was a redesign of the cut card slot operation. Our original design consisted of lowering the slot to the bottom and then raising the slot to a preset level determined by the user through a dial input. We realized it would be more practical for the user to adjust the slot height in live time with a dial. This allows the user to correct his preference of penetration level in live time instead of having to redo the whole operation. Due to this change, we were no longer in need of the limit switches. Their only use was to detect the lowering of the cut card slot which we removed.

The final design alternative involved the physical build of our machine. As we started building, most of our machine was built out of cardboard. Cardboard was easy to work with but unfortunately it was not rigid enough. Pieces of the build would move freely and affect the movement of cards during the shuffle operation. In order to combat this, we changed the material of a substantial part of our design to PLA plastic. This rigidness was much more reliable.

Card Detection Subsystem

Each IR sensor module emits infrared light and measures the intensity of the reflected light from the bottom card. When the cards are present just above the sensor which is flushed with the tray, the reflected signal is very strong, but when the tray is empty, very little if any at all, light is reflected. The threshold voltage was able to be tuned with an on-board potentiometer which was employed to measure tests and check performance under different lighting conditions.

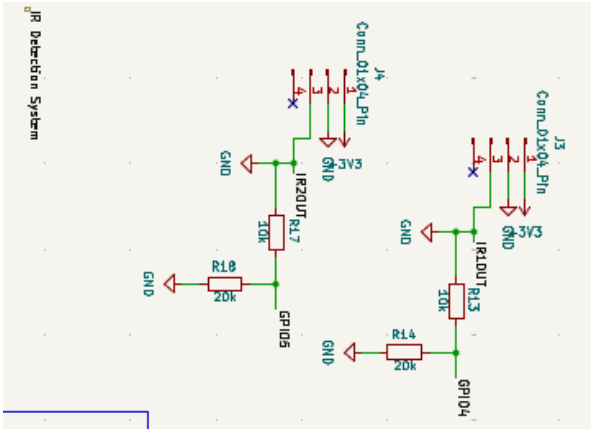


Figure 2: Schematic drawing of Subsystem 1 in KiCAD

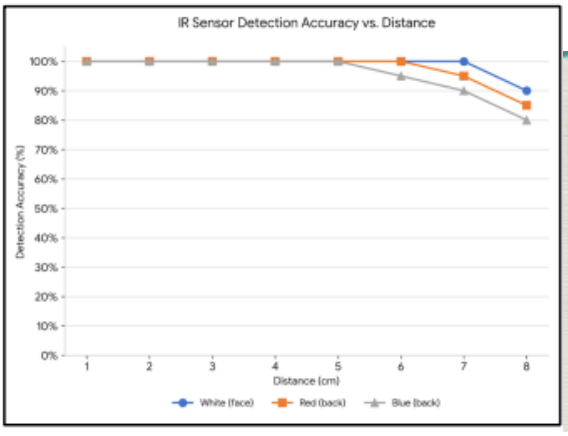
Electrically, the routing of this subsystem was simplistic due to the only components being the 2 IR sensors and routing them back to the MCU for data transport. It was important to add the pull-down resistors for each of the connections to the IR sensors so that the signal line is connected to ground and there is no floating voltage. To interface this with the PCB, the schematic included 1x4 connectors (one of which would be unconnected as there is no need for AO) which would be wired to the pins of the IR sensor. On the breadboard, a similar routing pattern was employed, just directly plugging in wires adjacent to the 3 pins on the IR module (PWR, GND, DO)

Physically, there were minimal design issues when building this subsystem. The main struggles involved determining how to attach the IR sensors to the trays effectively and managing card detection under different light settings. Plastic pads were added underneath the pre-shuffle trays to allow the IR sensors to be attached by screw into the bottom of the tray. To combat the light issue, IR sensor sensitivity was maximized. This ensured that even when there was shade present, the sensor would still detect the presence of a card.

Verifications

All of the requirements for this subsystem were met as seen below. The IR sensors on the trays both read values within the LOW (0 – 0.8V) and HIGH (2.0 - 3.3V) ranges. Tuning of the IR sensor module potentiometer was more than comfortable for the operating distance of the cards on the shuffle trays. Latency was also excellent for live reading and MCU performance.

Requirement	How it was Verified	Verification status
The subsystem must have a latency of less than 200ms to ensure the MCU can halt the motors immediately upon the last card being drawn	Live telemetry with serial monitor continuously logging HIGH and LOW readings when cards were placed in front of sensor	
Sensors should be able to identify card presence accurately. Comparator outputs a “high” level when cards are present and “low” level when cards are not present.	Using the multimeter, values with the high value read out to 3.2 volts and the low end measuring out to .112 volts	
Comparator [3] must produce output signals within an interpretable and safe range to the ESP 32	Output signal levels were within safe tolerance to ESP32 and proper subsystem behavior ensured this was met	
The comparator threshold must be adjustable to account for different card back colors (red / blue)	[See graph below]	



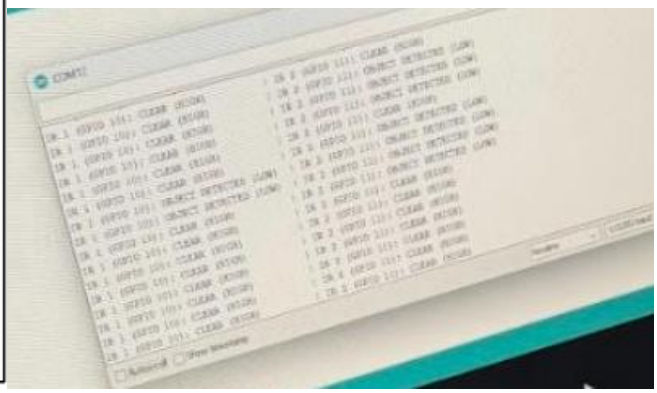


Figure 3: IR Sensor Detection Accuracy vs Distance for each card color backing

Figure 4: Live telemetry on serial monitor for IR sensor to MCU communication

Deck Shuffling Subsystem

```
else if (currentState == SHUFFLING) {
  if (SRV1state == LOW || SRV2state == LOW) {
    myServo1.writeMicroseconds(1700);
    myServo2.writeMicroseconds(1300);
  } else {
    Serial.println("Both IR clear. Starting 5s empty confirmation
    cardEmptyStartTime = millis();
    currentState = WAITING_CARDS_EMPTY;
  }
}
```

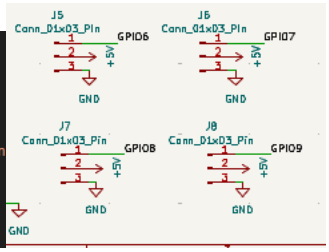


Figure 5: Code for servo rotation under SHUFFLING operation in state machine. 1300 and 1700 control the speed of servo motors in opposite directions

Figure 6: Schematic for connectors interfaced with servo motors (PWR, GPIO, GND)

In the software side, this subsystem was controlled by the second state in the state machine (SHUFFLING) and after the IR sensors detected no more cards present, it would wait for 5 seconds for the servo motors to clear out any last cards that might be stuck, then halt movement, and then move onto the next state.

3D printed trays were designed to create an area to mount the sets of servo motors. After they were mounted using tape set to a height where they barely protruded from the tray, they were wired up to the connectors that attached them to the Development kit. The motors then fed the cards through a 3D printed slot created to fit over the tray and make the cards fall through one at a time. Once they cleared through the slot, they would drop through the funnel down to the collection tray.

The biggest mechanical challenge was the design of the card drop slot. A great amount of friction between the bottom edge of the card slot and the tray surface caused cards to jam. After several design iterations, this was partially resolved by adding sloped lead-in edges to the ends of the slots that the cards would pass through, reducing the contact angle and letting gravity do more work. A secondary issue was the lack of organized card stacking in the funnel, which was being prototyped and attempted to fix but not fully resolvable within the project timeline. Due to the immense amount of prototyping that was required, we stuck with a mostly cardboard iteration so that we could modify when needed to try to fix the funnel.

Verifications

This subsystem was the most error prone as mentioned above. As such, we were only able to achieve partial functionality in our requirements and verification.

Requirement	How it was Verified	Verification status
Motor wheel must advance one card per motor actuation under nominal conditions to ensure acceptable shuffling (success rate > 90%)	Verified the success of this operation visually with different color card backings in each tray Also recorded burst and shuffle metrics after each trial [see graphs below]	
Shuffle operation will not begin until	Visually verified and implemented	

collection tray is fully retracted.	in software state machine	
Beam-break sensors must generate a pulse for at least 1ms duration so that it can be detected by MCU. Shuffling should pause in event of a jam	Nullified the use of beam-break sensors in favor of simplicity of design and interior mechanics	
Servo motor torque must be able to overcome card friction without deteriorating cards	Checked visually after each shuffle if cards were being deformed	

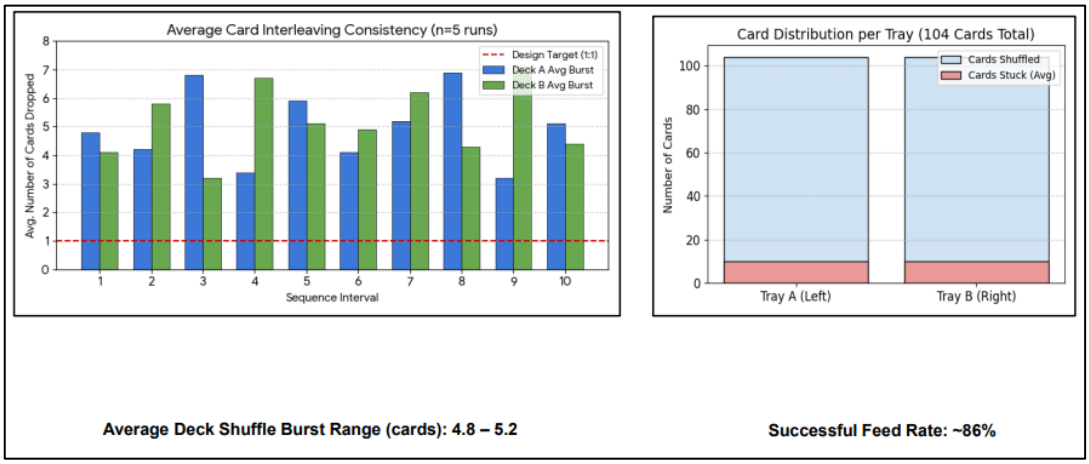


Figure 7: Test results of Requirement #1, showcasing the issues with the 1:1 shuffling and the "bursts" of cards that would come out when the motors would spew the cards from the tray

Figure 8: Showcasing test results of how the last 5-15 cards would typically get stuck due to no weight pushing them down on the wheels. This data was collected as an average over 10 independent runs.

Cut Card Insertion Subsystem

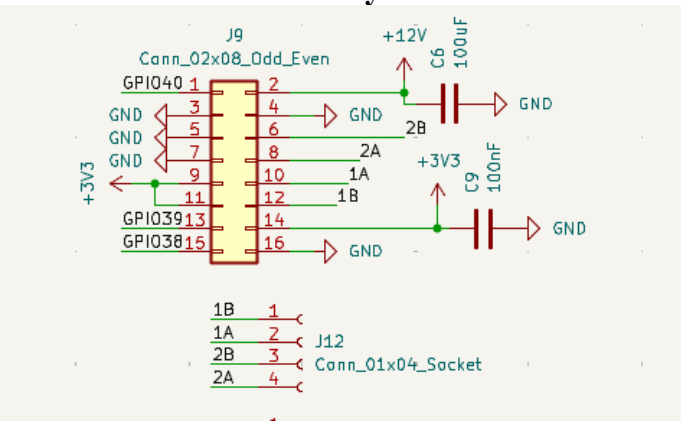


Figure 9: Stepper motor driver (A4988) [9] interfaced with design, 3 GPIO communication lines and connection with NEMA17 motor.

Figure 10 (below): Code for our mapping of rotary encoder tick to stepper motor movement. Instead of trying to set number of continuous ticks to some percent of motor shaft rotation, we said whenever tick is read, move some set amount (set variable in runFullRotation() helper function)

```
bool currentStepBtn = digitalRead(BTN_STEP_EN);
if (lastStepBtnState == HIGH && currentStepBtn == LOW) {
  controlMode = !controlMode;
  digitalWrite(EN_PIN, controlMode ? LOW : HIGH);
  Serial.print("Stepper: "); Serial.println(controlMode ? "ON" : "OFF");
  delay(50); // debounce
}
lastStepBtnState = currentStepBtn;

if (controlMode && encoderTriggered) {
  Serial.println("Encoder triggered -> running step");
  runFullRotation();
  encoderTriggered = false;
} else if (!controlMode) {
  encoderTriggered = false;
}
```

This subsystem consists of a plastic frame with a motor mounted at the top attached to the cut card slot through a pulley and fishing line. The slot moves vertically up and down guided by metal rails on its left and right sides. A dial input from the user is used to adjust the height of the slot along the rails.

Once we made the design changes mentioned in the design alternatives section, this subsystem was fairly easy to implement. When constructing the rails for the slots we ran into some minor jamming issues as the slot moved up and down the rails. This was combatted through sanding out the inner rectangular hole of the slot which was the part that was in contact with the rails. We encountered several issues when attaching the fishing line to the slot and the pulley. Multiple times we accidentally snapped the fishing line when trying to manually move the slot during testing. This was a minor inconvenience that we were able to overcome through being more careful and retying the line. One aspect that worked very well was the margin of error from the rotation of the dial to the vertical movement of the slot. Each dial turn corresponded relatively well to one movement of the slot height which allowed us to have a high level of customization.

Verifications

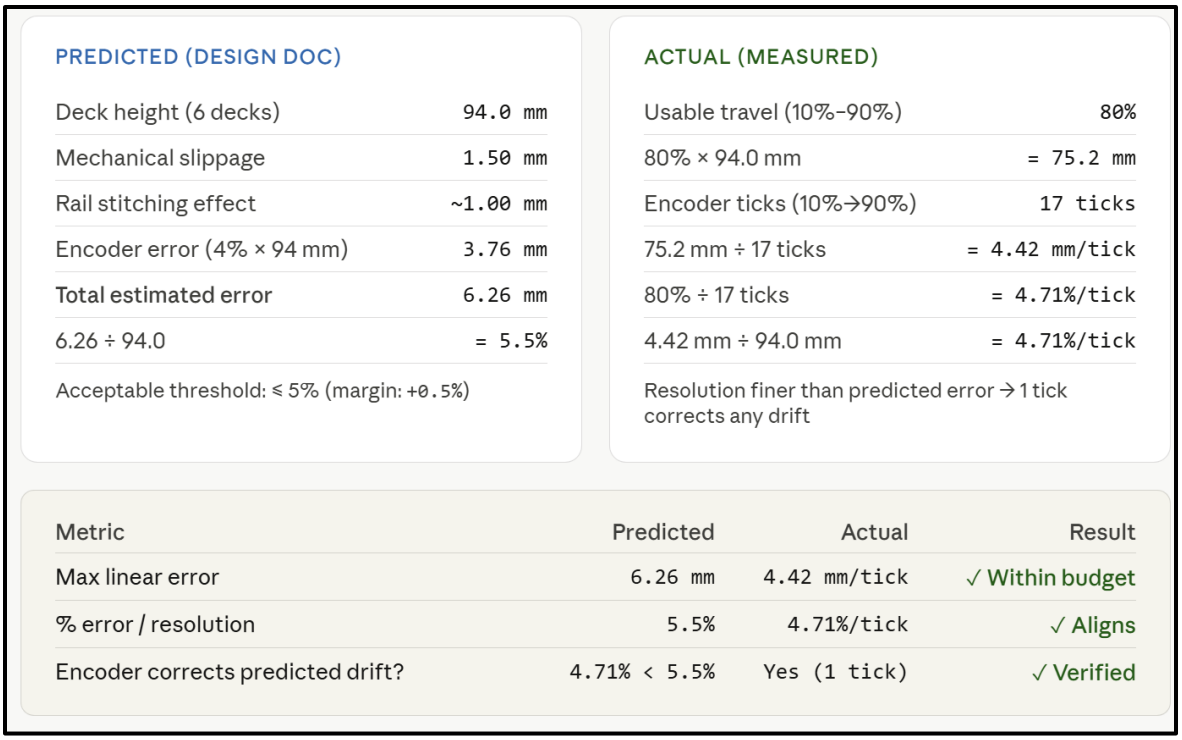


Figure 11: Depiction of predicted vs actual cut card adjustment precision

The image above depicts the predicted vs actual data that was gathered through the testing process of the cut card insertion slot.

$$H_{slot} = \left(\frac{P}{100} \times \text{Total Steps}\right) \times \left(\frac{\pi D}{S}\right)$$

Eq.1 Describes the vertical height of the slot for a given penetration setting. P is the desired penetration percent, D is the pulley diameter, and S is number of steps per stepper revolution. When applying this to worst-case error based on tolerance analysis, encoder contributes 0.04*94mm = 3.76mm of error. This in conjunction with estimated mechanical slip results in total linear error of 6.26mm (discussed in design document) and estimated 5.5% error from desired user cut-card input. We exceeded these expectations in our actual design.

Requirements	How it was Verified	Verification Status
Rotary Encoder input must be able to map deck depth from 10 – 90% at increments of 4% to ensure user input is properly mapped	Rotate encoder by set detents. Calculate number of ticks it takes to travel height of card deck and measure individual penetration height each tick has. Shown in figure above	
Limit switches must trigger within reasonable time when contact is made from lowering of the pulley system (3.3V to 0V within 1mm of physical contact)	This requirement was no longer applicable since we removed the limit switches from the design. See design alternatives section.	
Slot travel along the vertical axis must remain free from jams and the insertion window should remain relatively flat	Freely move slot up and down rails by hand and by rotary encoder control. Check for mismatches in height for both sides after several runs with ruler.	

Deck Tray Extension Subsystem

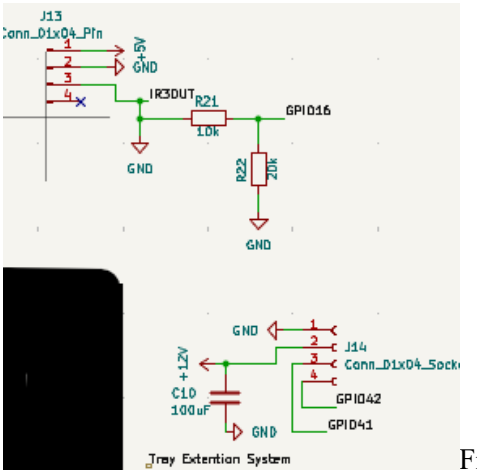


Figure 12: IR sensor and Linear Actuator driver schematics

The design of this subsystem involved an IR sensor, linear actuator, motor driver, two guide rails, and a plastic frame and tray. The plastic tray is mounted on the guide rails in order to extend and retract in accordance with the IR sensor detection. When the cards are present on the IR sensor and all prior subsystems are completed, the linear actuator will begin extending until it is fully it is in its maximum position. From there the cards will wait to be retrieved, once retrieved there will be a slight delay then the actuator will retract back to its starting position.

We had several issues with the physical assembly of this subsystem. We initially attempted to hot glue the rails to the plastic encasing, but they unfortunately were not secure enough. Our solution was to secure the rails by screws which worked out very well. Outside of this the remainder of the subsystem was assembled well, and we had full completion of the project.

Verifications

Requirements	Verification	Verification Status
Optical sensor must detect deck removal to allow for repeated shuffling processes (maintain logic HIGH vs logic LOW signals at appropriate settings)	Use multimeter to measure sensor output voltage when cards are present (should read > 2.4V Use multimeter to measure sensor output when cards are removed (should read < 0.8V) Ensure timing difference between removal and voltage drop is below 100ms	
Tray needs to extend once shuffling operation is concluded.	Visually ensure that the cards have cleared the feed trays before the bottom tray extends with the collected deck. Time the gap between the cards clearing the feed trays and the collection tray extending to find out	

	if there is the proper amount of delay. Confirm time taken is not tedious	
Tray needs to fully extend out of the body to allow for removal of the cards. (to full extent of linear actuator)	Measure the extension of the tray with a ruler and verify that it is extended out far enough for easy removal of the shuffled deck. This will be done visually and through use of human interaction	

Since the IR sensor used in this subsystem is the same as the one used in subsystem 1, the multimeter ratings are comparable for requirement satisfaction. Both the bottom two requirements can be verified through our final project video submission.

Power Supply Subsystem

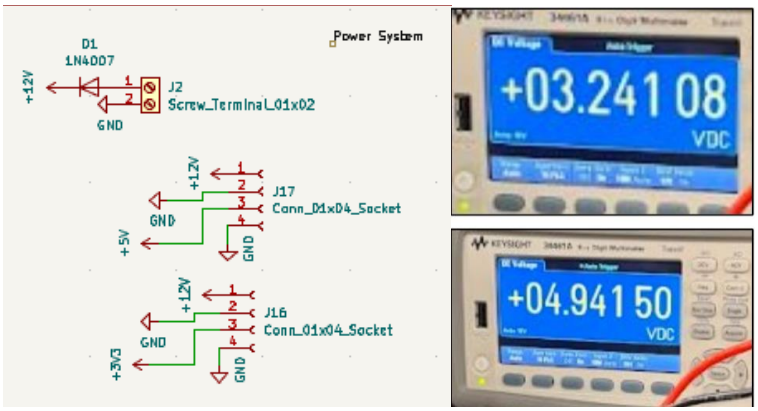


Figure 13: Schematic of buck converter module routing for power traces with battery plug-in

Figure 14: Output of buck converters after 12V input supplied

We built this subsystem using a 12v volt battery that we staged down through buck converters to make a dedicated 12-volt and 5-volt line with the 3.3 volts for the motors being supplied by the PCB. We chose to do it this way since it makes it the easiest for our project to remain modular, and a couple of our components required different amounts of power. We chose to use buck converters to smooth out the power that we are getting from the battery and try to prevent and overloading that can happen with a battery sometimes when it has a sudden inrush current that can damage different components. This subsystem had a lot of issues when it came to overloading our board with power. We burnt out our MCU chip a couple of times, and some of it can likely be connected back to the power supply system. In the end we chose to go back to using the bench power to control both of our dedicated power lines.

Initially, we were using LM317 regulators which required the use of resistors for a voltage divider that would step down the voltage. We followed this methodology: For the 5 V rail with R1 = 240 Ω and R2 =

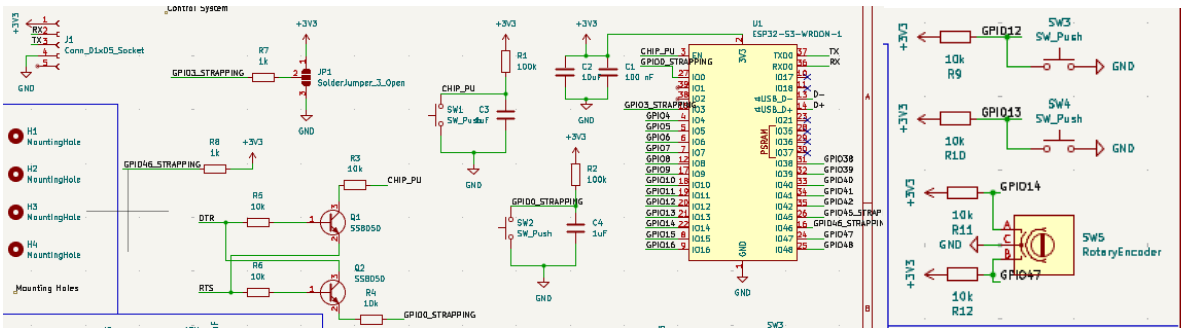
720 Ω: V_out = 1.25 × (1 + 720/240) = 5.0 V. For the 3.3 V rail with R1 = 240 Ω and R2 = 390 Ω: V_out = 1.25 × (1 + 390/240) = 3.28 V, within 5% of the 3.3 V target. (Eqs. 2 & 3)

Verifications

Requirements	How it was Verified	Verification Status
Voltage outputs are of correct magnitude and within a reasonable level of tolerance Ex. 3.3V rail must maintain voltage of within 5% of 3.3V under a max load of 500 mA to power the ESP32 and other sensor lines	Use multimeter to measure output voltage at various points along the line Ensure readings are within tolerance threshold [see images above]	
The 5V rails for the servo motors must maintain proper power distribution when all 4 motors are in rotation	Connect oscilloscope to 5V power line Activate all 4 servos simultaneously (shuffling cycle) Monitor voltage to see if there is no voltage dip Ensures that power dist. is able to deliver at peak distribution for motors	
Traces routed from this subsystem are required to adhere to the correct minimum thickness to allow for proper power distribution	Examine PCB design to ensure traces are of the correct thickness. Measure voltage inputs at sensors and components to ensure they meet requirements.	

Control Unit Subsystem

Figure 15 (below): Schematic of control unit with all programming circuits in place and buttons & rotary encoder routing positioned



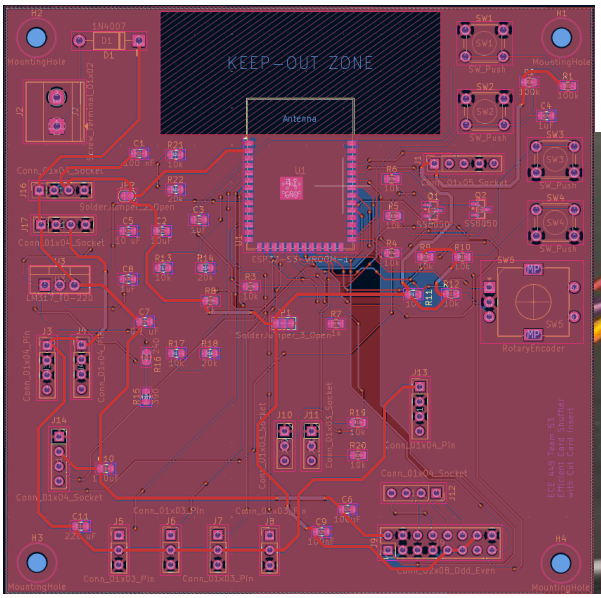
The control unit was powered by the MCU chip in the center of the project which in our case was the ESP32 Devkit. This devkit was connected using DuPont wires to each of our components through various GPIO connections. It was then programmed using the Arduino software through a USB-C cable that we plugged into the UART port on the chip [12]. Using this software, we were able to make the logic of the project sequential and not simultaneous. The programmed route depended on the low signals from the IR sensors in subsystem 1, then from the low signal from the IR sensor in subsystem 4. This route would not trigger however without a pushbutton input which we coded into being the initial condition to start the

project. For subsystem 3, we were able to program the rotary encoder in a way that when the dial recorded one tick clockwise, the pulley was raised 1/17 of the total distance. We also had all of our state machine report back to us in the serial monitor. This allowed us to debug easily since we would know which part of the operation the test would stall out on.

Verifications

Requirements	How it was Verified	Verification Status
Commands from input buttons and dials are correctly received and able to communicate across all subsystems.	Using a multimeter, measure the output of the buttons when they are pressed and not pressed and ensure reading corresponds to action.	
Must be able to continuously operate throughout multiple shuffling cycles to ensure proper operation (State Lockout).	When shuffle cycle is in operation, pressing the buttons should have no impact on the machine Ensure that no behavior is changed when a current cycle is in mode	
Read and store pulses from the rotary encoder accurately.	Open digital monitor and test rotary encoder movements Measure degree of turn and confirm it aligns with pulses	

PCB Implementation



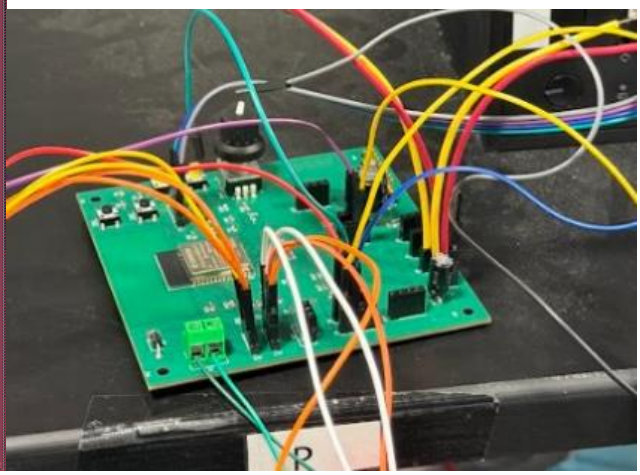


Figure 16: PCB view of our Round 4 Design

Figure 17: Assembled PCB with wiring for systems 1,2,5,6

Our custom PCB went through 3 major design revisions (Round 1, Round 2, Round 4). The final design was what we ended up fabricating as two separate boards.

Our first board was able to integrate subsystems 1, 2, 5, & 6 (all power and control + shuffling operation). We soldered all components onto the board and used it for testing up until this point. However, disaster struck when we used multimeter to verify a voltage reading and the pin connected to Vm and Gnd which caused a spark and short circuited, ruining the board and creating a rush of current spike to the MCU GPIO pins due to too much draw.

We then put together our second board but due to an issue with a solder bridge that led to an overheated UART bridge and MCU, this second PCB was also corrupted, and we were no longer able to program any firmware on it, despite putting another ESP32 MCU on it.

As a result, we were forced to go back to our breadboard implementation which was a huge blow but even in the short amount of time we had to revert back, we were able to reconfigure all our wiring to the breadboard and deliver an *electronically* successful project.

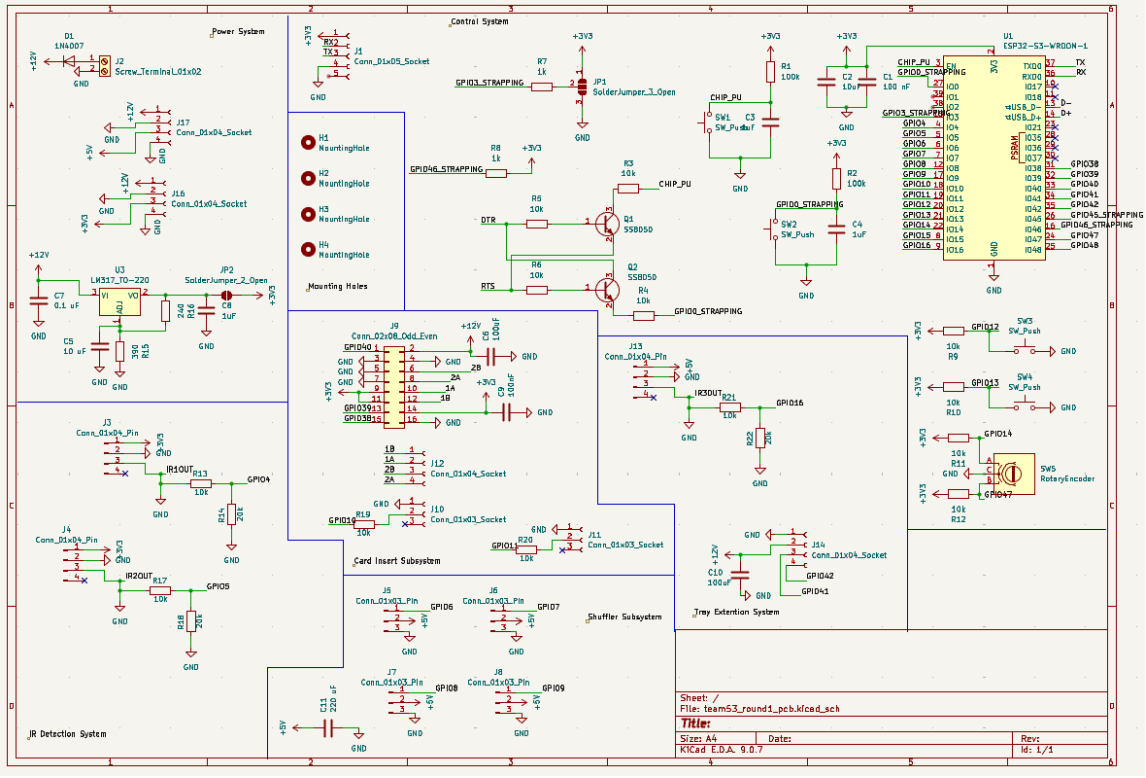


Figure 18: PCB Schematic

4. Costs

Below is the final list of the parts that went into the project. We ended up spending more than we had planned for. Most of these unexpected costs came from having to purchase parts we hadn't expected we'd need or having to replace parts that accidentally broke. Luckily, most of our additional costs came from our allocated ECE supply budget, so the individual out of pocket cost increase was minimal. Our final cost for all parts was $317.99. Our initial total cost from the design review was $202.76. This equates to roughly a $115 increase in unexpected parts cost. As mentioned above, we were able to absorb most of this into our $150 budget allocation.

4.1 Parts

Description	Manufacturer	Qty	Total Cost ($)
Continuous motion servo motor (4pck)	Feetech	2	53.98
IR Beam Break Sensor - 4pcs	MELIFE	1	6.99
MGN7 150mm Linear guide rail – 2pcs	Uxcell	2	37.38
Rotary Encoder 24PPR	Adafruit Industries	1	6.91
12V Stepper Motor SMALL bipolar	Sparkfun Electronics	1	15.27
Mini Linear Actuator 12V 4 Inch	SZAWINLI	1	32.99
2V 2400mAh AA NI-MH Battery Pack	CBB	2	0.34
ESP-WROOM-32 ESP32 Dev Board	HiLetgo	3	47.13
A4988 Stepper Motor Driver	AILUOMI	1	9.99
DRV8871 Motor Driver	Teyleten	1	12.99
CP2102 Serial Adapter UART Converter	HJHYUL	1	6.99
120pcs Multicolored wires	ELEGOO	1	6.98
EBOOT 3 pack MP1584EN DC-DC Buck Converter	EBOOT	2	15.84
Clear Fishing Line	WUINOID	1	5.99
120 pcs 40cm Wires	EDGELEC	1	11.99
Precision Knife	35435A11	1	5.39
USB-C>A 3ft Adapter	AK-B8173011	2	31.84
Electrical Tape Purple		1	3.46
Scotch Tape		1	5.54
Total			**317.99**

4.2 Labor

Our final labor costs exceeded our projections. We initially accounted for about 40hrs of total work per team member, this prediction underestimated the amount of work it took to complete the project. Our final labor costs use a figure of 60 hours, which we feel is more accurate. These results in $35/hr x 2.5 x 60hrs x 3 team members = $15,750. This brings our total cost to $15,750 + $318 = $16,068.

4.3 Schedule

Week	Task	Person
February 22nd – March 1st	Order parts from TA for subsystems 1&2	Everyone
	Work on PCB design and order by Thur	Steve
	Finish Design Document	Alex & Matt

March 1st – March 8th	Begin building subsystems 1&2	Everyone
	All ECE supply shop parts ordered by 3/3	Everyone
	All Amazon parts ordered by end of week	Alex
March 8th – March 15th	All parts ordered by 3/10 - final	Everyone
	Begin physical PCB wiring and interfacing with motors	Everyone
	Refine PCB design as needed	Steve
March 15th – March 22nd	Spring break	N/A
March 22nd – March 29th	Begin construction of remaining subsystems	Alex & Matt
	Finalize PCB design and reorder if needed	Steve
	Print any preliminary 3D parts	Everyone
March 29th – April 5th	Finalize construction of subsystem 3 and begin testing sensors	Alex & Steve
	Work on Assembly of Pulley system	Matt
April 5th – April 12th	Assemble all physical subsystems	Everyone
	Complete testing of sensor inputs and motor outputs	Everyone
April 12th – April 19th	Test machine and address inefficiencies	Everyone
	Reprint or replace any parts with 3D parts as needed	Alex
April 19th – April 26th	Mock demo	Everyone
April 26th – May 3rd	**Final Demo**	Everyone

Above is the schedule referenced from the design document. We were able to stay on track pretty well up until about mid-April. We ran into several setbacks during the assembling of the physical build. This unfortunately took time away from the test phase and PCB work. Nevertheless, we were able to put in a lot of work in the final couple weeks and put together our final design.

5. Conclusion

5.1 Accomplishments

We accomplished a good deal on our project this semester. We created the base of a card shuffler with the added features we set out to add. Both the cut card insertion and deck tray extension functionality work completely as designed and intended. In addition, we were able to get a partially functioning PCB before the time of our demo (before it fried) and were able to program it to perform most subsystems, indicating proper wiring design. Other than the physical design of system 2, we felt that our build was able to do what it was intended to, and incorporate a lot of new features that aren't present on the market.

5.2 Uncertainties

We are still uncertain about the reliability of the shuffle operation, but if we were given the chance to work on it again, we know what we would do to make it work. Across 10+ runs with 4 decks, an average of 10+ cards per run becamse stuck in the slots, yielding a somewhat subpar feed rate of 86% vs our goal of 90%. This failure was continuously an outcome of the last 10-20% of the cards getting

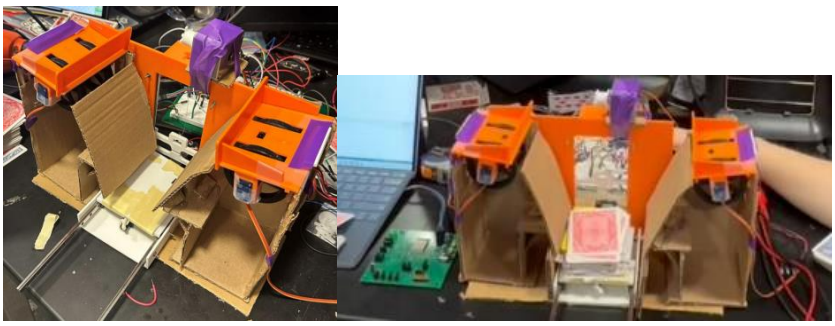
stuck when there was no pressure on it..A steeper funnel with more guidance as well as reduced friction on the slot would alleviate the issues we had been running into.

5.3 Ethical considerations

We had a few ethical concerns when it comes to our project mainly concerning codes 1.1 and 1.5 of the IEEE code of ethics. Considering the nature of our project being used in gambling situations, we put a large emphasis on making sure we "uphold the highest standards of integrity" [1]. This is important because any tampering with our device by the user could destroy the integrity of the game that they are playing. To prevent this, we changed our cut card to be more of a visual setting to prevent people from crunching numbers or somehow manipulating the bottom of the decks to make sure only the cards they had wanted to see in the deck would appear. We also wanted "to hold paramount to the safety, health, and welfare of the public" [1]. We did that by adding a 2 second delay once the cards were taken from the collection tray before the tray pulled back into the machine. This would effectively allow anyone who is grabbing their cards to get their fingers clear of the machine and away from any possible issues with pinch points.

5.4 Future work

As mentioned, before we know a lot of the changes that we would need to make to have our current machine become operable. However, if we were to continue working past that stage, we would want to change the material of the body from cardboard to either a PLA material or a thin metal to give it a nice complete look. We would also modify the funnel so that it would be a tighter channel so that the cards would organize more uniformly after being shuffled. Additionally, it would be great if we had more time with the PCB and more ESP's so that we could get the PCB back to functioning and implement the fully working PCB. We would also look into any other additional features for card games such as a setting for games that require a lot less cards such as euchre. If we wanted to take our product to market, we would also have to apply for a lot of licensing considering how closely intertwined our project is with gambling.



Figures 19 & 20: Front-view of final-prototype of Card Shuffler design

References

[1] IEEE Code of Ethics, IEEE, 2026. [Online]. Available:
https://www.ieee.org/about/corporate/governance/p7-8

[2] "TCRT5000, TCRT5000L Reflective Optical Sensor with Transistor Output," Vishay Semiconductors, datasheet, rev. 1.7, Jan. 2009. [Online]. Available:
https://www.vishay.com/docs/83760/tcrt5000.pdf

[3] "LM393, LM393A, LM2903, LM2903A, LM2903V, NCV2903 Dual Differential Comparators," ON Semiconductor, datasheet, rev. D, 2023. [Online]. Available:
https://www.onsemi.com/pdf/datasheet/lm393-d.pdf

[4] "ESP32-S3-WROOM-1 & ESP32-S3-WROOM-1U Datasheet," Espressif Systems, datasheet, v1.4, 2023. [Online]. Available:
https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

[5] "ESP32-C6-WROOM-1 & ESP32-C6-WROOM-1U Datasheet," Espressif Systems, datasheet, v1.1, 2023. [Online]. Available:
https://www.espressif.com/sites/default/files/documentation/esp32-c6-wroom-1_wroom-1u_datasheet_en.pdf

[6] "FS90R Micro Continuous Rotation Servo," Feetech, product datasheet, 2022. [Online]. Available:
https://cdn-shop.adafruit.com/product-files/2442/FS90R-Servo-Motor-Specification.pdf

[7] "Rotary Encoder — Mechanical, 24 Steps per Revolution (PEC11R)," Adafruit Industries, product page no. 377, 2023. [Online]. Available:
https://www.adafruit.com/product/377

[8] "ROB-10551 Stepper Motor with 28 cm Lead," SparkFun Electronics, product datasheet, 2022. [Online]. Available: https://www.sparkfun.com/products/10551

[9] "A4988 DMOS Microstepping Driver with Translator and Overcurrent Protection," Allegro MicroSystems, datasheet, rev. 9, 2014. [Online]. Available: https://www.allegromicro.com/~/media/files/datasheets/a4988-datasheet.pdf

[10] "DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense," Texas Instruments, datasheet, rev. C, Nov. 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8871.pdf

[11] "MP1584EN 3A, 1.5MHz, 28V Step-Down Converter," Monolithic Power Systems, datasheet, rev. 1.1, 2012. [Online]. Available: https://www.monolithicpower.com/pub/media/document/MP1584_r1.0.pdf

[12] "CP2102 Single-Chip USB to UART Bridge," Silicon Laboratories, datasheet, rev. 1.3, 2023. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf