ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

AUTONOMOUS CARD DEALER

Team No. 24

ROHIT CHALAMALA
rohitc2@illinois.edu

ADAM NABOULSI
adamjn3@illinois.edu

RALPH BALITA
rbalita2@illinois.edu

TA: NIKHIL ARORA

PROFESSOR: OLGA MIRONENKO

May 3rd, 2023

Abstract

This paper presents the design, implementation, and verification process of the Autonomous Card Dealer (ACD). The system relies on a combination of hardware and software components to automate the shuffling and dealing process, thus providing an efficient, customizable, and user-friendly experience. The mechanical hardware components include a card shuffler, rotating base, and card dispenser mechanism, controlled by the STM32 microcontroller and the Motor IC units. The sensing hardware component used for the ACD is an ultrasonic sensor, capable of taking distance readings to confirm players and take measurements. The user interface hardware components used for the ACD include one (1) 16x2 Liquid Crystal Display (LCD) and four (4) push buttons, capable of displaying messages to the User according to button inputs. The software components consist of various methods to automate the card shuffling and customizable dealing process, in which primitive functions are scheduled and executed. Each of the primitive functions are methods that simply send and receive signals from the microcontroller to each device. Collectively, the results of the project demonstrate the feasibility and potential of the Autonomous Card Dealer to the automation of various card games.

Keywords – Card Games, Autonomy, Microcontroller, Power, Motor Control, User Interface, Sensing

Table of Contents
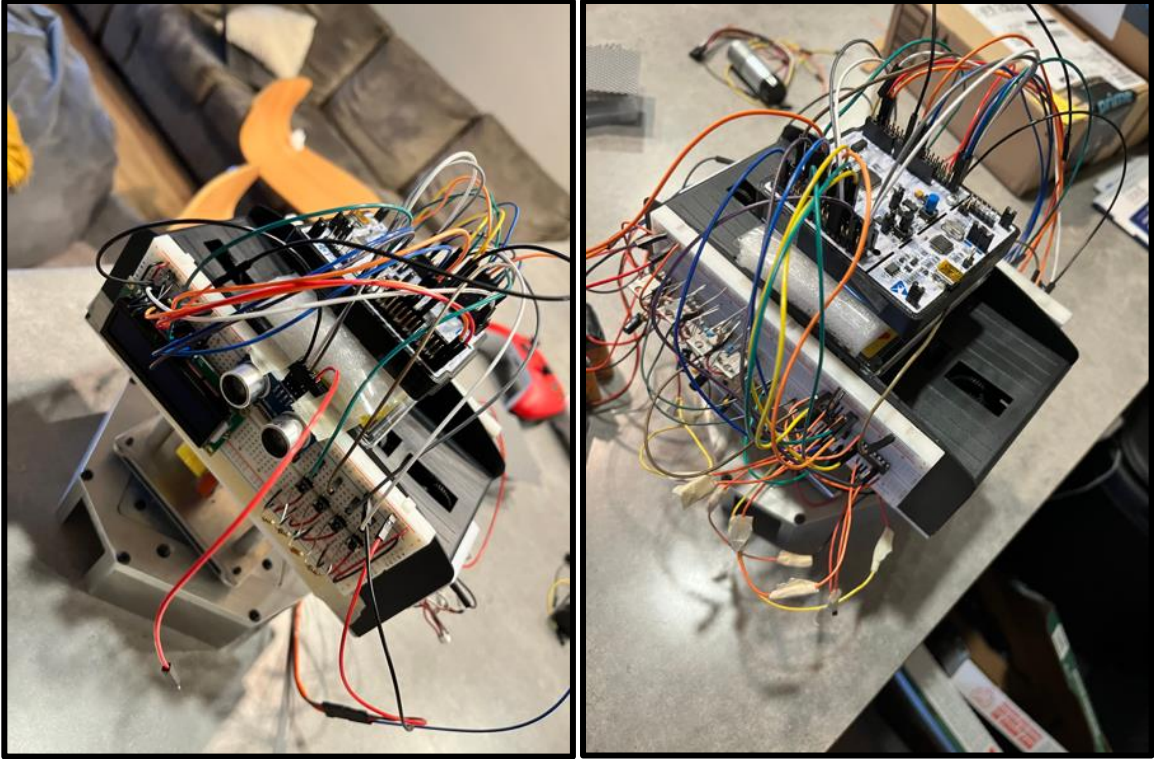
1 Introduction

1.1 Problem

Card games have been played by friends and competitors for centuries; to name a few: Poker, Literature, Uno, Rummy, etc. However, in many scenarios, players who deal and shuffle the cards are subject to cheating. Although cheating is rare, the repercussions of a cheating player and dealer can be expensive. Additionally, in casual gameplay, people may not have the energy to shuffle or deal cards. Thus, our goal was to create a fair card shuffling and distribution system in which the gameplay is smooth, effortless, and eliminates the possibility of cheating. To add complexity, our group implemented programmable game modes including but not limited to the ones named above. In these cases, we programmed the bot to deal cards to the players and set up the card game playing field as needed.

1.2 Solution

At a high level, we made the card game playing process effortless and fair by replacing the shuffler and dealer with a device. There are a few different components this project is made up of including the card shuffler, card distributor, and the user interface. The device ensures fairness because it gets rid of human error and prevents players from tampering with the deck. The card distributor would replace the dealer by being able to rotate and shoot out cards to certain locations on the table. The user interface contains buttons that allow the user to control turning the number of players, the game mode, and when to start shuffling or dealing throughout a game. Through designing and implementing this device, we intend to facilitate card games in an efficient, effortless, and equitable manner.

1.3 Physical Design

Regarding the physical design, we finalized our design with the mechanical team to ensure proper functionality. Below is a series of figures showing the mechanical design from a front, back, and top view. Our design involves a base holding the card shuffling and dealing system above it in place with a servo motor that can rotate the system 180 degrees. The card shuffler section consists of two DC motors to shuffle the cards individually into a card holder slot, and an ultrasonic sensor mounted on the outside to determine the distance to whatever obstacle it is facing. The cards are shuffled, piled, and held in the middle slot, ready to be dealt. The dealing mechanism of the system will deal cards as the base rotates the whole device around a table. The dealing mechanism consists of a DC motor beneath the card slot with a rubber wheel that pokes out from under the bottom of the card dealer to grip onto the cards and launch them out of the device. We have a liquid crystal display along with the buttons for parameter adjustment mounted on the outside of our system.

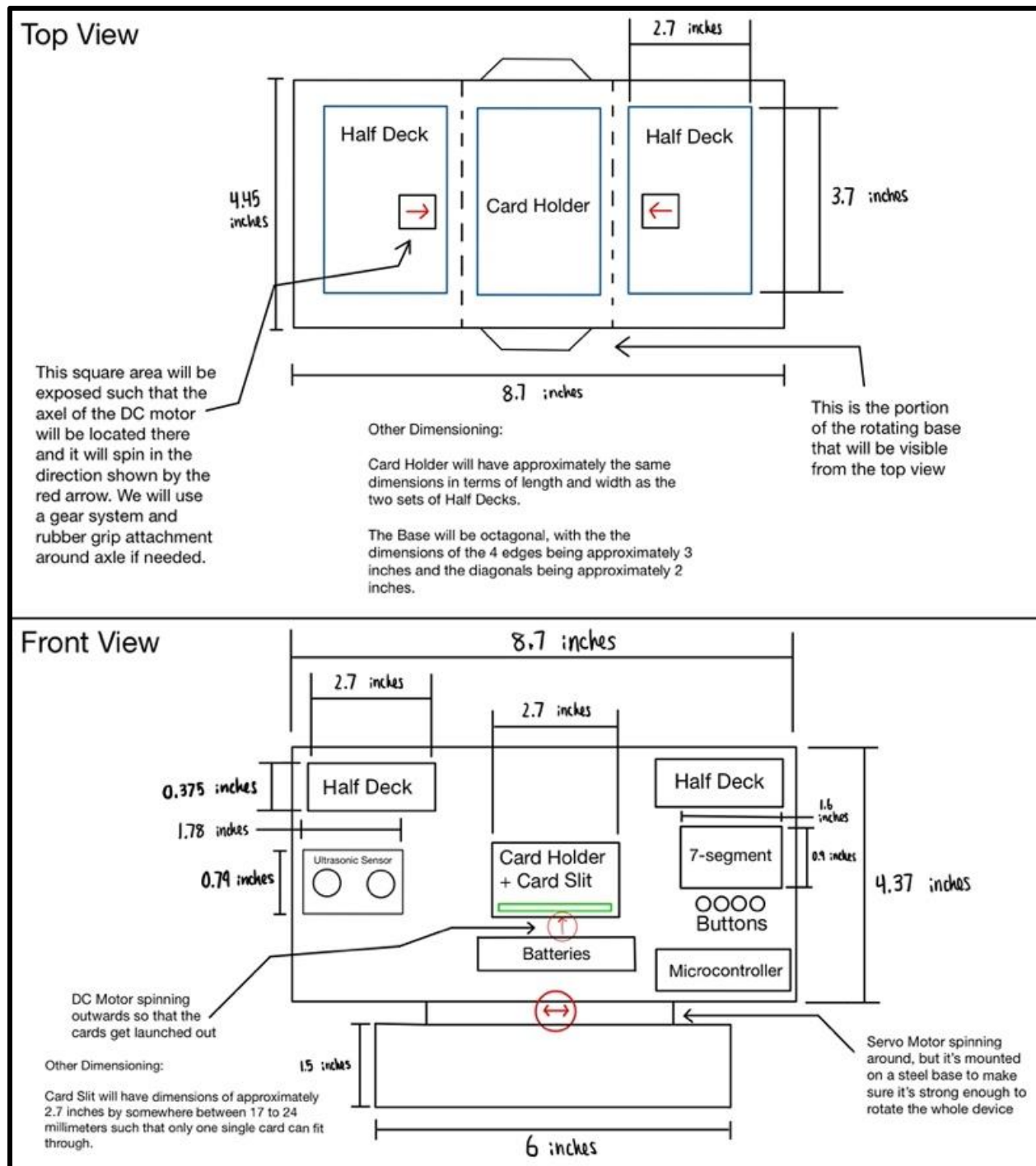*Figure 1: Card Shuffler/Dealer Device*

Top View

Half Deck

Card Holder

Half Deck

2.7 inches

4.45 inches

3.7 inches

8.7 inches

This square area will be exposed such that the axel of the DC motor will be located there and it will spin in the direction shown by the red arrow. We will use a gear system and rubber grip attachment around axle if needed.

This is the portion of the rotating base that will be visible from the top view

Other Dimensioning:

Card Holder will have approximately the same dimensions in terms of length and width as the two sets of Half Decks.

The Base will be octagonal, with the the dimensions of the 4 edges being approximately 3 inches and the diagonals being approximately 2 inches.

Front View

8.7 inches

2.7 inches

2.7 inches

Half Deck

Half Deck

0.375 inches

1.78 inches

0.79 inches

Ultrasonic Sensor

Card Holder + Card Slit

7-segment

1.6 inches

0.9 inches

4.37 inches

OOOO Buttons

Batteries

Microcontroller

DC Motor spinning outwards so that the cards get launched out

Servo Motor spinning around, but it's mounted on a steel base to make sure it's strong enough to rotate the whole device

Other Dimensioning:

Card Slit will have dimensions of approximately 2.7 inches by somewhere between 17 to 24 millimeters such that only one single card can fit through.

1.5 inches

6 inches

Figure 2: Theoretical Physical Design

1.4   High-level Requirements List

Throughout the course of our project, we were able to fulfill the following goals that we set out at the start of the semester:

1. **Shuffle a set of cards evenly.**
    The shuffling mechanism on this device shall generate a fairly shuffled deck of cards – a pseudo-random riffle shuffle. This entails that if a full deck of cards were to be split up into two sub-decks (assume the sub-decks are two different colors) with the same number of cards. When the shuffling process is carried out, the resulting deck would have 1-2 cards in a row (that are the same color) at maximum from either of the sub-decks when going through the deck from top to bottom.

2. **Distribute the cards to the players and set up the playing field.**
    Given an aligned and ready-to-be-distributed deck of cards, the dealing system shall shoot out the correct number of playing cards to each of the players (in the direction of the players) and the correct field setup for a given game mode– one card at a time.

3. **A functioning user interface with a display and buttons for parameter adjustment**
    Four buttons – Shuffle, Deal, Number of Players, Game Mode – will allow the user to adjust the parameters of the current game and get real time feedback through an LCD display.

1.5   Summary

The Autonomous Card Dealer (ACD) is an all-in-one device capable of shuffling cards, dealing cards to players, and setting up custom card game fields with a given user input. The system consists of four subsystems with a central microcontroller unit:

- **Power** (Voltage Regulators and 18V Power Source),
- **Motors** (Shufflers – DC; Dealer – DC; and Rotating Base – Servo),
- **Sensor** (Ultrasonic Sensor),
- **User Interface** (16x2 LCD Display and Buttons),

The purpose of Section 2, Design, is to discuss the design procedures and details. The following section, Section 3, will cover Requirements & Verification for each subsystem and each of their components. The purpose of Section 4, Costs, is to summarize the costs of materials and labor for this project. Finally, the Conclusion will cover our project's accomplishments, uncertainties, ethical considerations, and future work that could be done for the project.

YouTube Video Demonstration [17]

2 Design Details

2.1 Block Diagram

Our final design consisted of the following four subsystems with the STM32 NUCLEO-F411RE microcontroller development board [1] in the center: Power, Motor, Sensing, and User Interface. The power subsystem's purpose is to act as a stable and reliable power source for the various subsystems and their corresponding components. The motor subsystem's purpose is to control both the dealing and shuffling processes in the project. The purpose of the sensing subsystem is to detect when a player is present through distance readings. Once a player is sensed the dealer can distribute the right number of cards accordingly in the player's direction. Lastly, the purpose of the User Interface subsystem is to provide the user with the ability to select the current card game mode and number of players, and control when to shuffle or deal through push buttons and a display. These components will all work together to facilitate the shuffling and dealing process for a given card game.
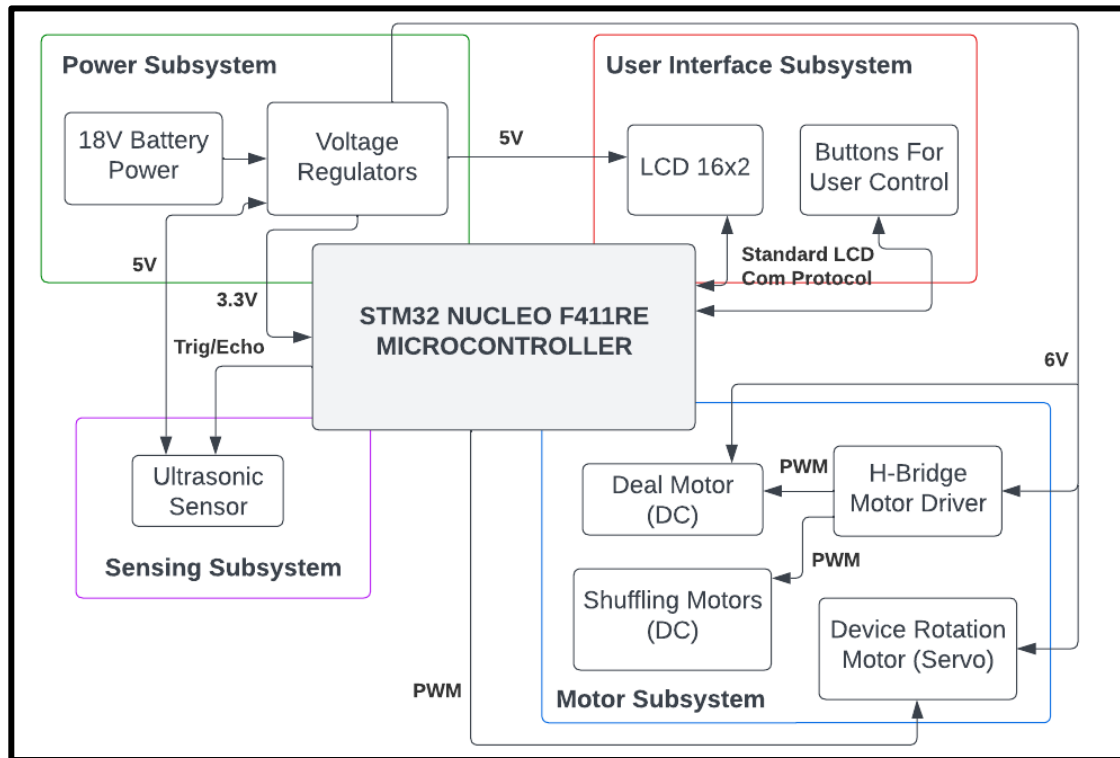

Figure 3: High Level Block Diagram
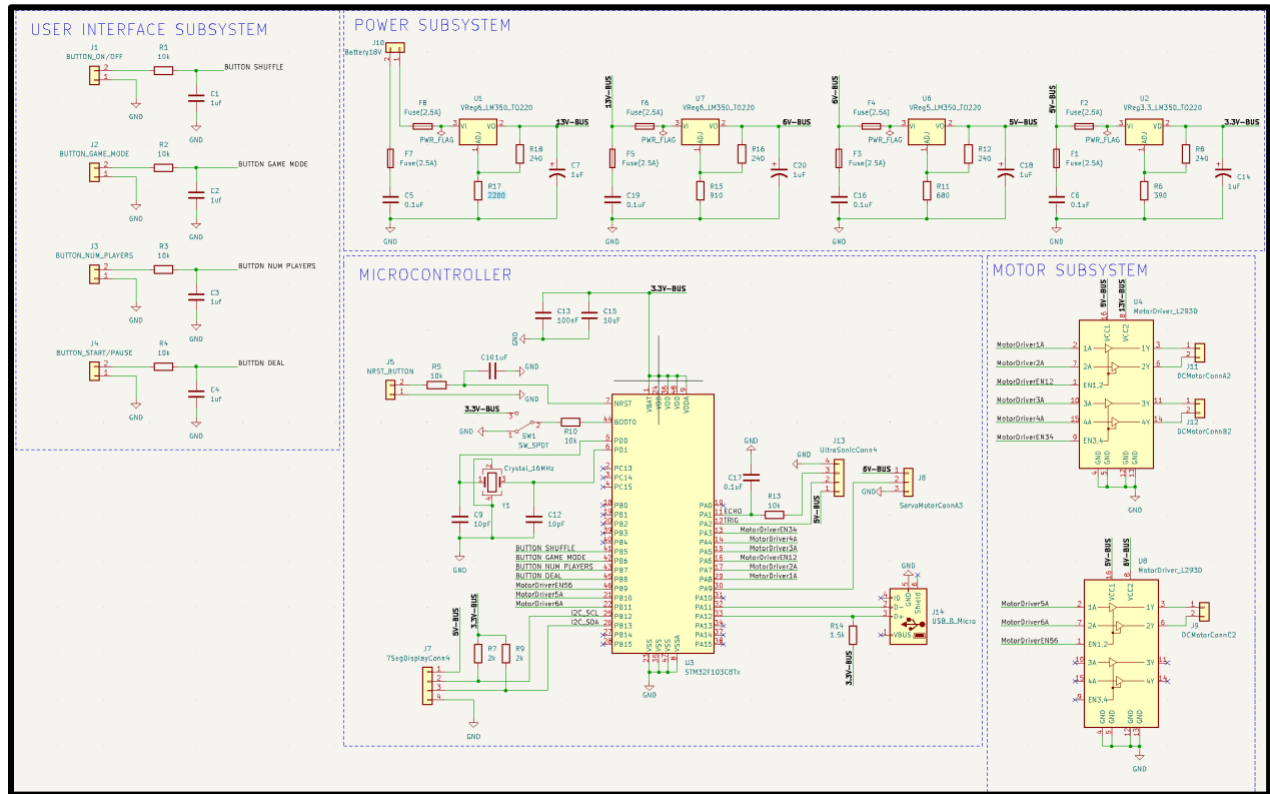
5

2.2   Schematic

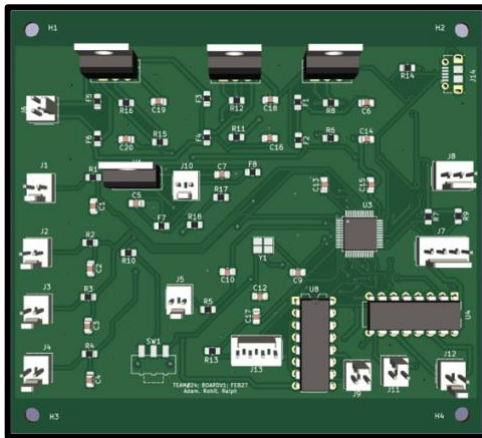Figure 4: Electronic Theoretical Circuit Schematic of the ACD [11]

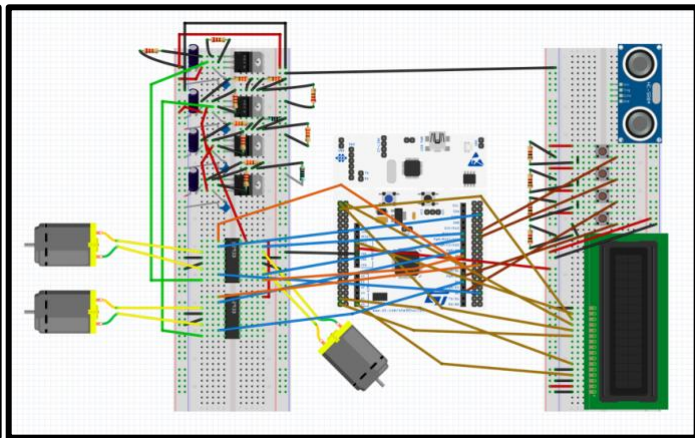Figure 5: Theoretical PCB 3D View

Figure 6: Breadboarded ACD Fritzing Layout

2.3   Power Subsystem

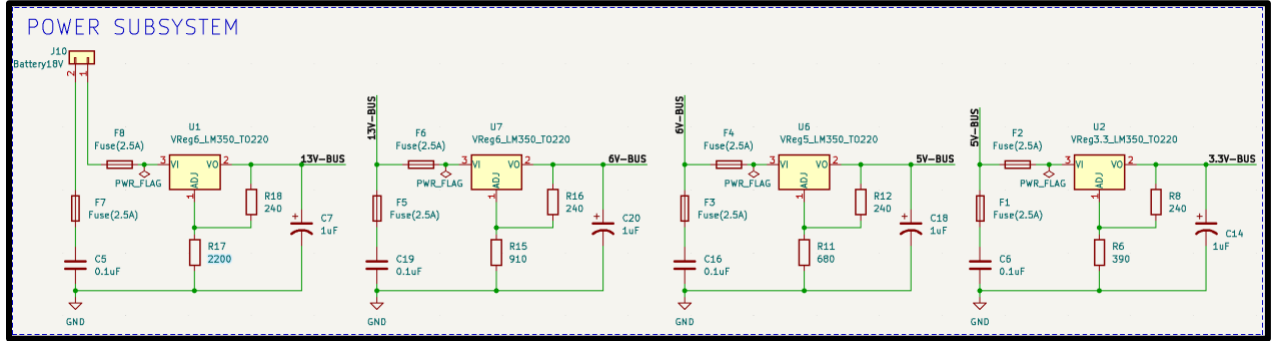Below is a schematic of the power subsystem for your reference.

Figure 6: Power Subsystem Schematic

We needed a variety of different voltage outputs from our power subsystem since the parts that we had chosen are rated for various operating voltage: 13V, 6V, 5V, and 3.3V. We regulated down from 18V (two 9V batteries in series). Due to the strength of the motors that we were using, the current consumption was very high at approximately 2.5A. So, we decided to go with the LM350 adjustable voltage regulator [3] which can withstand up to 3A of current. We also chose to put some fuses in our circuit to protect our board in the case that our current got too high. The voltage regulators each follow the same circuit structure where each output is fed consecutively into the next voltage regulator's input [Fig 6]. Details on the circuitry will be further explained below. To prevent the voltage regulator circuitry from heating up we purposefully stepped down in increments from 18V to 3.3V. Now to jump into the calculations of how we set up the voltage regulator circuits; the image below is from the LM350 voltage regulator datasheet.
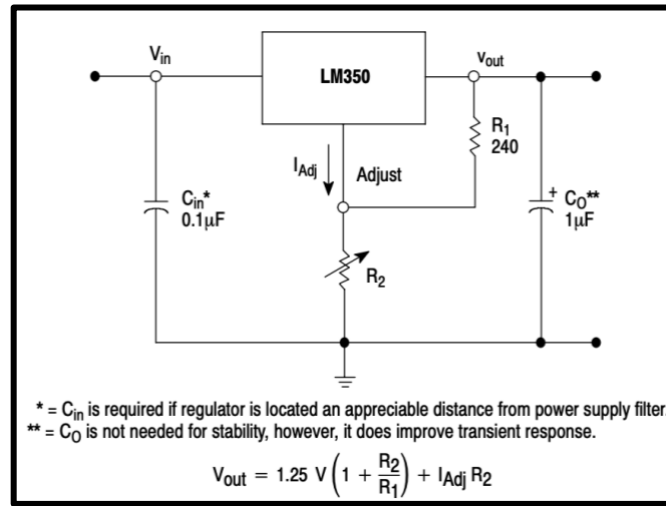

Figure 7: LM350 Voltage Regulator Circuit [12]

We can see that C_{in} = 0.1µF, C_o = 1µF, and R_1 = 240 Ω, are all fixed values. The only value we need to calculate and adjust is R_2. The input voltage does not affect the output voltage in this case because the chip is designed to maintain constant output regardless of what voltage is fed into Vin. The equation we see here is V_{out} = 1.25V (1 + R_2/R_1) + I_{Adj}*R_2, however the term I_{Adj}*R_2 is negligible due to I_{Adj} usually being below 100µA according to the datasheet. Therefore, the equation is V_{out} = 1.25 (1 + R_2/R_1) and we can apply this equation to find the necessary R_2 values for our four desired output voltages.

- 3.3V: 3.3 = 1.25 (1 + R_2/240); R_2 = 393.6 Ω but the closest standard resistor to this is 390 Ω.
- 5.0V: 5.0 = 1.25 (1 + R_2/240); R_2 = 720.0 Ω but the closest standard resistor to this is 680 Ω.

7

- 6.0V: 6.0 = 1.25 (1 + R_2/240); R_2 = 912.0 Ω but the closest standard resistor to this is 910 Ω.
- 13.0V: 13.0 = 1.25 (1 + R_2/240); R_2 = 2256.0 Ω but the closest standard resistor to this is 2200 Ω.

Given that the voltage regulator circuits can output the calculated voltages, we will be able to power everything that we need to in our project. In general, those were the main calculations and considerations made when designing the power subsystem for this project.

2.4   Motor Subsystem

For context, below is a schematic of the motor subsystem and the centralized microcontroller unit for pinout connection purposes inherent to the subsystem.
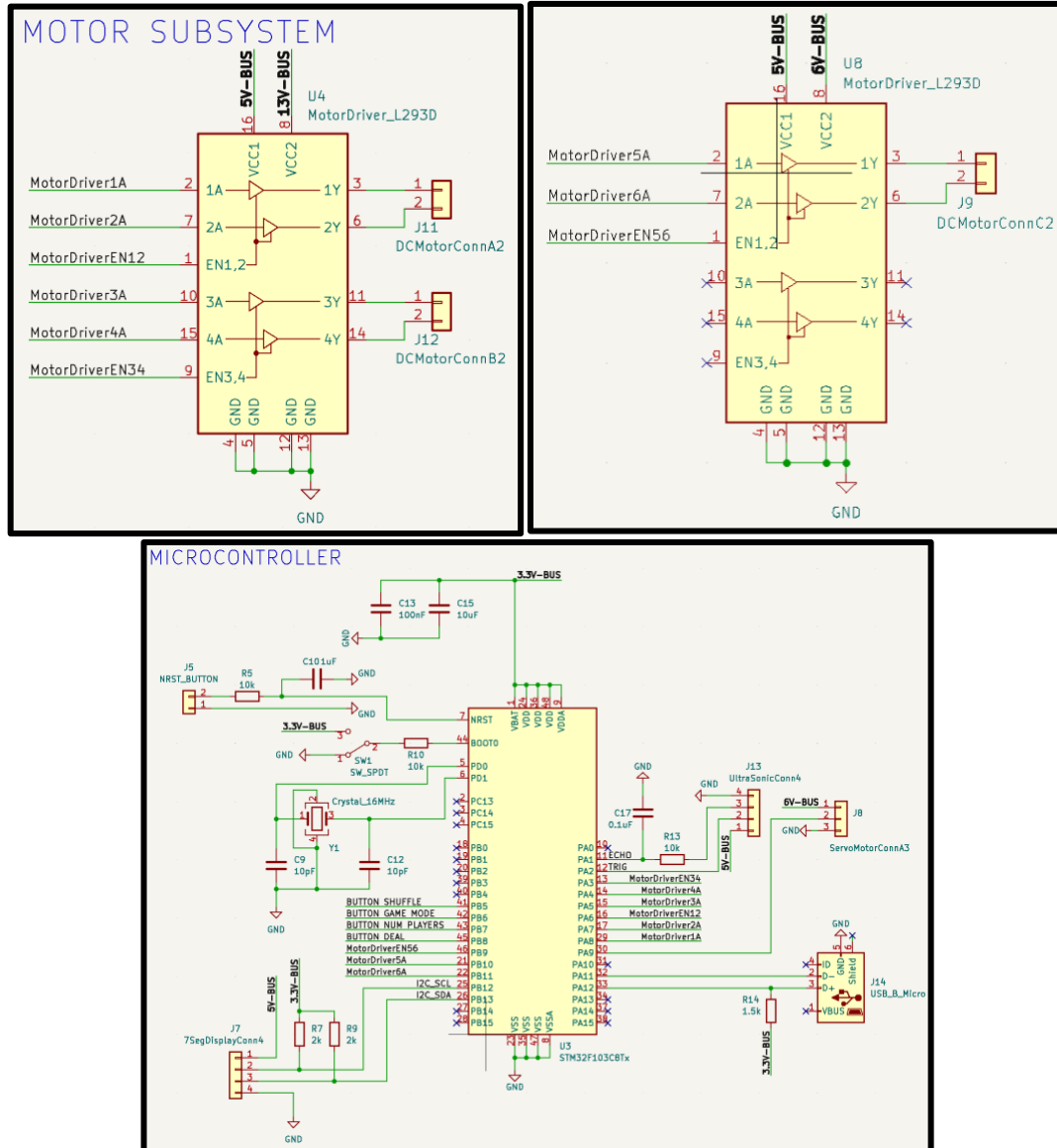
Figure 8: Motor Subsystem Components Schematic

The motor subsystem we designed consists of two H-bridge motor drivers. One of the motor drivers is used to control both of the DC shuffling motors, and the other is used to control the dealing DC motor. Lastly, we have a single servo motor for rotating the system that is controlled directly through the microcontroller pins. The DC shuffling motors runs on 13V (provided by the power subsystem and its

8

regulated output) at a constant speed when handling the shuffling. The dealing DC motor runs on 6V and performs one high speed rotation at a time to shoot out a single card through the rubber wheel attached to it. Lastly, the servo motor is directly connected to the microcontroller and powered by the 6V regulated source from the power subsystem. The L293D H-bridge motor drivers from STMicroelectronics [21] allow us to control the speed and direction of the shuffling DC motors and the dealing DC motor through PWM signals that they will receive from the microcontroller.
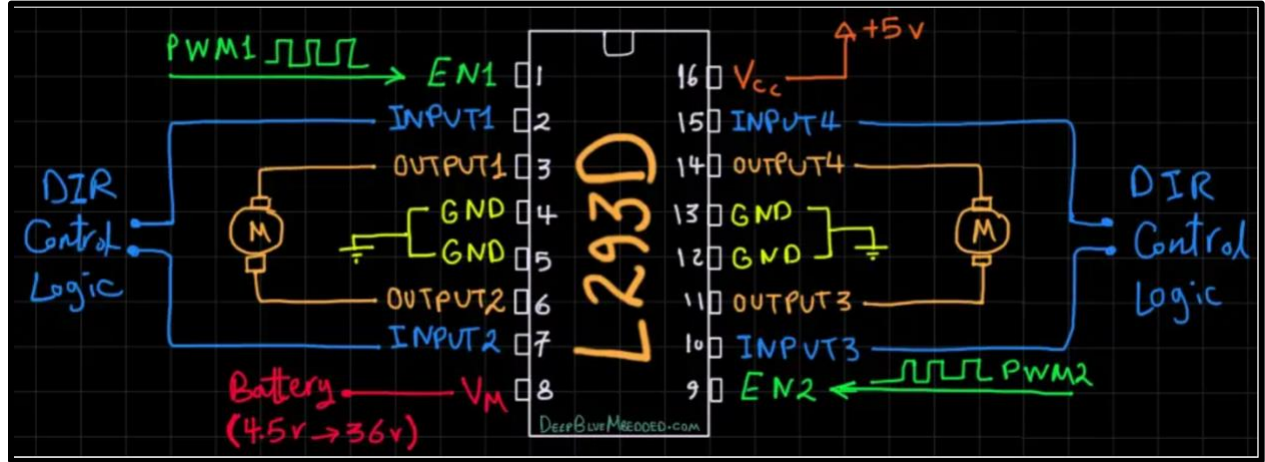

Figure 9: L293D H-Bridge Motor Driver Microcontroller Connection & Pinout Diagram [10]

Each motor driver carries four input pins, four output pins, and two enable pins. Thus, each motor driver is capable of controlling two DC motors at the same time. The enable pins on the drivers are connected to the microcontroller digital PWM I/O pins. The input pins are connected to regular digital I/O pins on the microcontroller. Lastly, the output pins are be connected to the two terminals of the DC motors. The motor drivers also have a V_{cc1} (used to drive the internal logic circuitry) and a V_{cc2} (used to power the internal H-bridge of the IC to drive the motors). For the motor driver controlling the two shuffling DC motors, V_{cc1} and V_{cc2} take in 5V and 13V respectively from the power subsystem's regulated outputs. For the other motor driver controlling the dealing DC motor, V_{cc1} and V_{cc2} take in 5V and 6V respectively. The servo motor simply has a power pin, ground pin, and control pin. The control pin is connected to a PWM digital I/O pin on the microcontroller such that we can control how much the motor rotates.

2.5   User Interface Subsystem

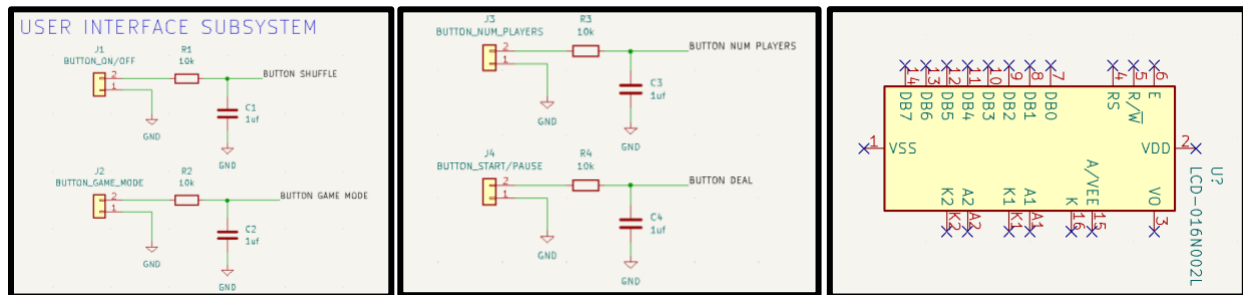For context, we have provided a schematic of the User Interface subsystem below.

Figure 10: User Interface Subsystem Schematic

The User Interface Subsystem was a lot more straightforward compared to the power subsystem. We used four buttons and an LCD 16x2 Display [7, 15] for user control, device status, and game information.

9

The four buttons are for initiating shuffling, initiating dealing, the game mode we are playing on (Literature, Poker, Rummy, Uno, Even Deal, Single Pile Deal), and the number of players (1-8). Most of the logic behind the user interface subsystem is done in software. When shuffling, we set the DC motor to rotate for a set number of iterations to ensure all cards are shuffled. When dealing, the first press indicates that we are dealing the correct number of cards to the player; any presses following this are used to set up the field or deal additional cards according to the game mode. For example, in the game mode Uno, whenever a player needs to draw a card, pressing the deal button again would deal them another card. For the game mode control button, whenever the button is clicked, the current game mode # increments until we reach 10; at this point, the game mode # resets back to 0. The reason for this is that we chose to allow 10 total game modes on our device, although, we can increase this number if wanted. As for the number of players, it will be a similar type of system but incrementing to a maximum value of 8.
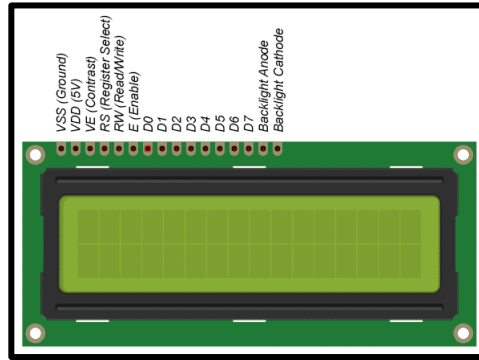

Figure 11: LCD 16x2 Pinout

The other portion of the user interface subsystem is the LCD display [Figure 11]. The pins we used were D4-D7, RS, E, and Backlight Anode which were all connected to GPIO outputs on the microcontroller. We also used VSS, VE, RW, and Backlight Cathode which were all grounded along with VDD connected to our 5V power supply. Pins D0-D3 were not needed to get the functionality we needed with the display.

2.6 Sensing Subsystem
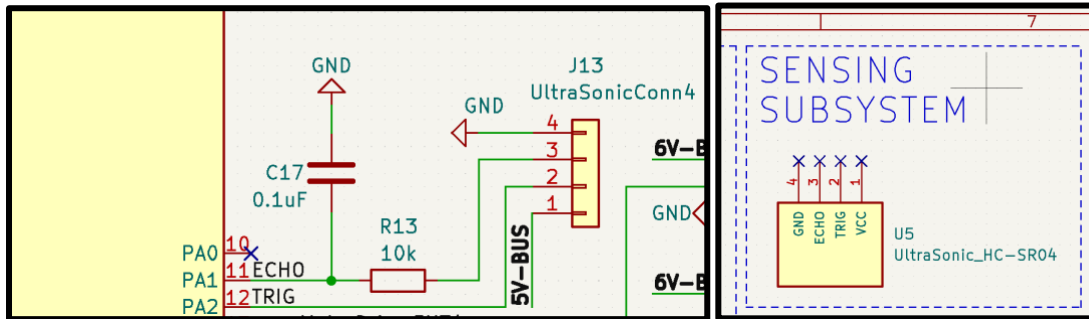
Figure 11: Sensing Subsystem Components Schematic

The ultrasonic sensor utilizes its own communication protocol involving trigger and echo pins. Ultrasonic Sensors [6] utilize sonar to take measurements of the distance between the object in front of the device and the device itself. It consists of a transmitter (TRIG) and a receiver (ECHO). The TRIG pin emits a 40kHz sound. The vibrations of sound resonate through the air, bounces off an object or surface, back towards the ultrasonic receiver. To calculate the distance to the object or surface in front of the ultrasonic sensor, we must measure the time between the initial transmission output signal and final reception input signal. We can utilize the following function [19]:

$$dist = v_{sound}\frac{(time\ elapsed)}{2} = (34.3\text{cm/ms}) * \frac{(time\ elapsed\ in\ ms)}{2}$$

$$= (34.3\text{cm/ms})\frac{(t_{rec} - t_{trans})}{2}$$

In which v_{sound} is the speed of sound (343m/s), t_{rec} is the final count on the timer when the echo pin reads a high signal, t_{trans} is the initial count on the timer when the trigger pin is pulsed high [16]. The ultrasonic sensor we purchased has a distance range of 2cm to 400cm (1in to 13ft). The device must be able to sense players at least 0.5 meters away, the radius of a professional standard poker table. The error on the device, with properly working code, is 3mm. The maximum reading angle (the wideness of measurement) is < 15 degrees.
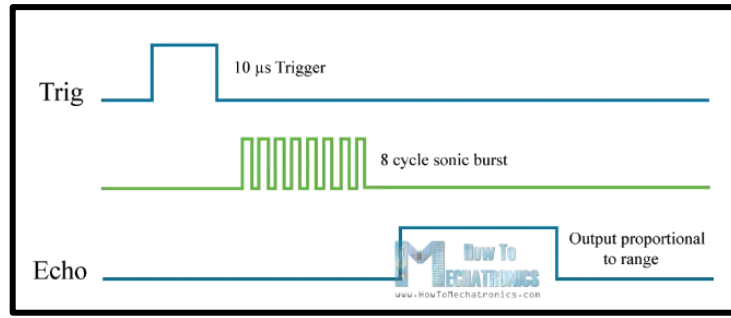

Figure 12: Trig/Echo Communication Protocol

2.7   Code Design

In regard to the code design for this project, there are five main functions to consider: DealCard, Shuffle, RotateCW, RotateCCW, and CalculateDistance. All of these functions are used in the main in order to assist in writing the code for each game mode.

DealCard is the function responsible for actually dealing the cards. It takes in a variable named iterations that represents the number of cards that need to be dealt out. The way that each singular card deal is done is through controlling the DC motors [10] such that we rotate the motor forwards, stop the motor momentarily, rotate the motor forwards again, and then rotate the motor backwards. This allows us to push out one card guaranteed followed by approximately half of the next card, but since we wrote backwards this card will go back into the deck. This method was decided upon after lots of testing to see how different strategies would distribute the cards out of the device.

Shuffle is the next function, and it is responsible for ensuring that the cards are getting shuffled pseudo-randomly in a riffle fashion. It also takes in a variable named iteration that represents the number of pairs of cards being shuffled into the device. The way that this code works is to turn on the first DC motor momentarily, then turn that one off and turn on the other DC motor, and lastly turn off both motors. In essence, this shuffles in one card from each side of the device and then stops the motors for each iteration. The first reason that we decided on this approach was because it would mean that the cards wouldn't collide during shuffling as the motor rotation is always offset. The other reason was because starting and stopping the motors allowed us to better preserve the 18V voltage source that we had, so that the batteries wouldn't die as quickly, because there wasn't a constant load on them for the entire shuffling process.

RotateCW is the next function, it is used to rotate the entire device clockwise a few degrees at a time. Like the other functions, it also takes in a variable named iterations that indicates how many rotations the device should perform. The reason that this function only rotates a few degrees per iteration is because when we are rotating clockwise, we want to give our ultrasonic sensor enough time to scan for players. Through our testing we determined that incremental movement clockwise was the best method for ensuring that our sensing subsystem worked successfully. The code for this function just involves setting the PWM signal for the servo motor [9] such that it rotates in one direction and then stops after it has rotated for a certain period of time.

RotateCCW is somewhat similar to the RotateCW function, but it is used to rotate the entire device counterclockwise approximately 60 degrees at a time. Like the remainder of the functions discussed so far it also takes in a variable named iterations that indicates the number of rotations the device should perform. The reason that this function doesn't need to be as precise as the RotateCW function is because on the backwards rotation we just want to get back to the original position, we don't need to continue scanning for players. The code for this function is similar to the RotateCW function other than the PWM signal has to be set in a way such that it rotates in the counterclockwise direction.

CalculateDistance is the last major function, and it is used to calculate the distance that an object is located away from the ultrasonic sensor in centimeters. The general idea for how this distance is calculated has already been described in Section 2.6: Sensing Subsystem.

2.8   Changes Made Since Original Design

In this section, we will discuss the various changes that were made on our original design and the reasons behind them.

2.8.1   Power Subsystem

The first change that was made was passing in 18V instead of 9V because we realized that our shuffling motor subsystem would require a much higher voltage than 9V.

As a supplementary change, we added another voltage regulator to regulate down to 13V as that was a suitable voltage input for our motor driver controlling the two DC shuffling motors. In addition, regulating from 18V down to 6V directly would garner a lot of heat.

Lastly, since our PCB did not end up working, we had to remove the fuses that we had initially planned for in our schematic, just because we did not have any 2.5A fuses that we could use for breadboarding purposes. We had initially used fuses rated for 2.5A because our servo motor for rotating the device could draw up to 2.5A of current at a maximum which was our highest current draw. However, we made sure that the voltage regulators we used could support 3A of current draw as a safety precaution.

2.8.2   Motor Subsystem

The motor responsible for the dealing mechanism was changed from a servo motor to a DC motor. The friction of a rubber wheel, combined with the torque, speed (rpm), and ease of direction control of a DC motor and motor driver proved to be a better fit for the system. The trajectory of the cards would be launched out of the ACD rather than pushed out of the ACD. Additionally, servo motors are specifically made for precise position control, however, in our case for dealing the cards, we simply needed enough torque, a high enough rpm, and direction control which can be done easily with a DC motor.

This change required another motor driver since each driver can only support two DC motors. However, we had 3 DC motors after this change that needed to be controlled. This extra motor driver gave us the ability to have bidirectional control for all our DC motors.

2.8.3 User Interface Subsystem

We initially had buttons called 'On/Off' and 'Start/Pause', but we replaced those with 'Deal' and 'Shuffle' as we realized it did not make sense in the scope of our project.

Since our PCB did not end up working and we switched to the Microcontroller Development Board, we decided to use an LCD display instead of a 7-Segment display as it would provide the user with more information rather than just two numbers representing the game mode and number of players. The LCD display can show whole lines of text indicating the settings that are being changed and the state of the machine (shuffling, dealing).

Lastly, we added a reset button to our project to allow the user to restart a game and change the parameters with ease. They will be able to set the device back to the original start state.

2.8.4 Sensing Subsystem

We originally planned on utilizing LIDAR technology to sense player locations but realized that there was a cheaper alternative to player recognition. We redesigned the sensing subsystem with an ultrasonic sensor to measure player distances as the device rotates: sound detected instead of light detected. We did not have to make any major changes from the Ultrasonic Sensor since the subsystem was straightforward.

3   Requirements & Verification

Detailed requirements and verification information can be found in the appendix and references will be provided for each subsystem below.

3.1   Power Subsystem
Appendix B, Table 1

Our requirements for the power subsystem were to ensure a stable regulated voltage supply for all the other subsystems. To power all the components sufficiently we needed the power subsystem to output 13V, 6V, 5V, and 3.3V consistently. These outputs were tested through breadboarding the schematic and probing the outputs of each voltage regulator. Through refining the resistors values in the voltage regulator schematic and continuously probing the outputs, we were able to achieve the voltage outputs that we needed. Results can be seen in the corresponding appendix section. We also made sure that the voltage regulators were capable of supporting 3A of current since our highest current draw component is the rotating servo motor which can draw up to 2.5A of current. This would ensure that none of the components burn if a lot of current is drawn at any moment during the device's operation.

3.2   Motor Subsystem
Appendix E, Table 1.

Our requirements for the motor subsystem were divided into 3 main parts: Shuffling DC Motors, Dealing DC Motor, and Rotation Servo Motor. To ensure proper and even riffle shuffling when a given deck split in two is fed into both sides of the card shuffler, we ran the following test: feed in 26 cards to the right and left shuffling motors with one of the half-decks facing up. A good shuffling output would entail that a maximum of 1-2 cards in a row appear face up or face down. After refining our motor driver code and continuing to test the shuffling mechanism, we were able to fulfill this requirement.

Regarding the dealing DC motor [10], the main requirement was to ensure that one card was dealt at a time through the slit in our physical enclosure and that the card would land roughly 5 cm out from the distribution point. Again, with refining the motor driver code and direction controls along with adjusting the size of the slit in our physical design, we were able to get an 80% rate one card dealing at a time.

Lastly, for the rotation servo motor [9] which was controlled directly through the microcontroller PWM signal (instead of through a motor driver), our main requirement was that the motor would be able to rotate the whole device in increments within a range of 180°. Through adjusting the PWM timer channel settings and testing the supplementary code, we were able to rotate in the motor in increments precise enough to deal to all players in a range of 1-8: ~11.25°, 12.9°, 15°, 18°, 22.5°, 30°, 45°, and 90°. Results can be seen in the corresponding appendix section.

3.3   Sensing Subsystem
Appendix D, Table 1.

Our requirements for the sensing subsystem were to correctly identify how far away each player is from the device. To get sensor readings, we needed the microcontroller to write a high signal to the output TRIG pin, followed by two reads on the input ECHO pin. The time difference between the two read signals is used to calculate the front distance [19]. The distance readings were tested through a set of measurements and

print statements. Results can be seen in the corresponding appendix section. We also ensured that the mounting of the ultrasonic sensor is secure, obstruction free, and tangle-free to improve the accuracy of the readings. The Requirements and Verification of the Sensing Subsystem can be referenced in the Appendix A, Table 2.

3.4   User Interface Subsystem
Appendix C, Table 1.

Buttons: Our requirements for the user interface subsystem were to ensure that the buttons function properly. Each button is responsible for changing the settings (number of players and desired game mode) or executing a sequence of methods (shuffle and deal).

Liquid Crystal Display [7]: Our requirements for the user interface subsystem were to ensure that the LCD prints strings of messages properly onto the display. It shall provide the user with immediate feedback, indicating which settings that are being changed or the sequence of methods being executed.

4   Cost Analysis

4.1   Parts and Materials

Refer to the [Appendix A: Bill of Materials](#) for all parts used on this project. In addition to the parts associated with the PCB, we also purchased batteries, extra wires, and a development board to run tests. The total cost of the materials adds up to $ 171.63.

4.2   Estimated Hours of Development and Cost of Labor

The members in our group are Electrical and Computer Engineers. According to the Grainger College of Engineering website [18], the average starting salary for ECE new grad is $105,352 in 2020-21. This is equivalent to $52.68 per hour.

$$\frac{3\ members}{1\ team} \cdot \frac{16\ weeks}{1\ member} \cdot \frac{10\ hours}{1\ week} \cdot \frac{\$52.68}{1\ hour} = \$15,770.88$$

4.3   External Materials and Resources

Below are a list of Materials and Resources that we utilized to complete the project. The estimated cost for external materials and resources is 30-unit hours * 52.68 = $1580.40

Machine Shop: Given that this project consists of many mechanical components, it is important to consult the ECE Machine Shop for guidance and assistance in creating a viable enclosure to hold the electronics and a durable system. We have consistently spoken to the Machine Shop after our weekly TA meeting on Wednesdays to ensure our progress is sustained.

Senior Design Lab Resources: Our team plans on utilizing the ECE's Senior Design Lab as a testing ground for the autonomous card dealer. Namely, we plan on using the soldering stations, computer stations, oscilloscopes, and power system supplies to test and showcase progress.

Development Resources: To create an autonomous card dealer, we will be programming on the STM32 ARM Microcontroller [1], in which we will learn to develop and test how the Microcontroller interfaces with the various subsystems involved in the project. We will be looking at resources to help us convert Sensor and Button Signals into useful data and control the autonomous card shuffler and dealer's respective motors in a scheduled manner.

4.4   Approximate Total Cost

When considering the cost of labor, materials, and external resources, the estimated total cost of the project is $171.63 + $15,770.88 + $1580.40 = $17,522.91.

5   Conclusion

5.1   Accomplishments

Our group was able to successfully complete the development of 1 working Autonomous Card Dealer, meeting most of the requirements outlined in the design document. During the demonstration, we were able to showcase the functionality of the system: display string messages on LCD, shuffle cards, get ultrasonic measurement readings, and deal cards to the players around a table one at a time as the device rotated.

In the process of development, our team was able to discover ways to make the user experience even easier. We thought about implementing a machine where the system would take information about the desired card game from user input. After pressing the start button, the system would automatically shuffle cards, deal players, deal field, and dispense the rest of the cards.

5.2   Uncertainties

Our group had a few uncertainties with implementing our design on the PCB. We had many small resistors and capacitors which made it difficult to solder all of the components with continuity across them. Additionally, the footprints were slightly too small which again made the soldering process difficult. To show a proof of concept and present a minimum viable product, our group was able to implement the project with the use of a STM based Development Board and a breadboard schematic that mimics that of our original PCB. Since our group ended up using NUCLEO-F411RE, we considered adding complexity to the project: replacing the four-digit seven segment display with a Liquid Crystal Display and adding extra functionality to the software components.

5.3   Ethical Considerations

Our group chose to follow the IEEE Code of Ethics [13] adopted by the IEEE Board of Directors through June 2020. The main purpose of this project is to prevent cheating and make card games fairer by eliminating the role of a dealer. It makes the game fair by allowing the machine to take care of anything outside of playing the game.

IEEE Code of Ethics: To uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities.

- To ensure the health, safety, and welfare of anyone who may use the card dealer we are developing, we will make sure to thoroughly test each of our components for proper functionality.
- As our project is quite mechanical, we will also frequently meet the Machine Shop Technicians to make sure none of the mechanical components or our design can prove hazardous to a user, or if any uncertainties surface during the development of our project.
- Our team will also meet with expertise (Teaching Assistants, Professors, etc.) and accept any constructive criticisms of our project.

IEEE Code of Ethics: To treat all persons fairly and with respect, to not engage in harassment or discrimination, and to avoid injuring others.
- To ensure proper teamwork etiquette, our team shall practice frequent and efficient communication via the following method: Discord servers with TA (Nikhil Arora) and text message group chats. Documentation and designs related to the project SHALL be shared virtually on a Google Drive to ensure ease of use and accessibility and physically on lab notebooks. This allows our team to keep track of the whereabouts of the project and have a running log of the progress the team has made.

IEEE Code of Ethics: To strive to ensure this code is upheld by colleagues and co-workers.

- To ensure this code is upheld by colleagues and co-workers, our group has decided to schedule a weekly meeting to discuss any violations of the code of ethics that may surface throughout the semester. If any matters are brought up, we will discuss them with our TA and/or the professor.

5.4   Improvements to the Current Design

1. Moving our design onto a working PCB
2. Using a Lidar sensor or computer vision techniques as opposed to an Ultrasonic for more accurate distance readings and thus more consistent dealing operation.
3. Using a better voltage supply such as a wall plug or a rechargeable battery pack that feeds into the power subsystem instead of lithium batteries.
4. Better motors for more consistent control of shuffling, dealing, and rotation.
5. Adding an arm lever or a spring to put weight on cards such that enough compression and friction is present when a given card is being dragged and shot out of the slit by the dealing dc motor.

5.5   Future Work

For future renditions of the autonomous card dealer project, we could consider adding the following features:

- **Gaming Console for Cards**: Store all common game modes on the chip as pre-programmed game modes the card gaming console and allow users to buy the premium pre-programmed game modes. Also capable of setting default game modes and
- **Adding Dealer Positions**: The system will rotate to the dealer before dispensing cards to the left (counterclockwise direction) of the dealer. The dealer label will automatically shift based on player.
- **Customize Game Modes directly onto the Device**: Have an option to customize and save a game mode onto the ACD solely using the buttons on the device (no programming required).
- **Computer Vision based Verification Process**: Utilize computer vision software to determine which cards have been dispensed and determine if two cards of the same rank and suit have been dealt: deck fault detection functionality.
- **Calculated Dealing Mechanism**: The program will use an array of distances which represent the readings of the ultrasonic sensors when confirming each player. It will change the torque applied by the dealing motor to 'shoot' the cards based off each of those distances. Different player = different distance = different torque applied.
- **Multi-Deck Support**: The system will be able to handle multiple decks at once.
- **Internet Connectivity**: Get automatic system updates.

6 References

1) "STM32StepByStep:Getting_started_with_STM32_:_STM32_step_by_step" (n.d.), [Online]. Available: https://wiki.st.com/stm32mcu/wiki/STM32StepByStep:Getting_started_with_STM32_:_STM32_step_by_step (accessed on 02/23/2023).
2) "STM32F103C8 Datasheet", ST Electronics (n.d.), [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (accessed on 03/24/2023).
3) "LM350 Adjustable Voltage Regulator", ElecCircuit (2022), [Online]. Available: https://www.eleccircuit.com/lm350-adjustable-voltage-regulator/ (accessed on 02/18/2023).
4) "LM317 3-Terminal Adjustable Regulator", Texas Instruments (2020), [Online]. Available: https://www.ti.com/lit/ds/symlink/lm317.pdf (accessed on accessed on 02/10/2023).
5) "HCSR04 Ultrasonic Sensor and STM32", ControllersTech (2020), [Online]. Available: https://controllerstech.com/hcsr04-ultrasonic-sensor-and-stm32/ (accessed on 03/15/2023).
6) "Ultrasonic Sensing Basics (Rev. D)", Texas Instruments (2021), [Online]. Available: https://www.ti.com/lit/an/slaa907d/slaa907d.pdf?ts=1682337584920&ref_url=https%253A%252F%252Fwww.google.com%252F (accessed on 03/15/2023).
7) "Interface LCD 16x2 with STM32 without I2C", ControllerTech (2020), [Online]. Available: https://controllerstech.com/interface-lcd-16x2-with-stm32-without-i2c/ (accessed on 04/18/2023).
8) "MM145453 Liquid Crystal Display Driver Datasheet", Texas Instruments (2013), [Online]. Available: https://www.ti.com/lit/ds/symlink/mm145453.pdf?ts=1682412730926&ref_url=https%253A%252F%252Fwww.google.com%252F (accessed on 04/19/2023).
9) "Interface Servo motor with STM32", ControllerTech (2020), [Online]. Available: https://controllerstech.com/servo-motor-with-stm32/ (accessed on 04/20/2023).
10) "STM32 DC Motor Speed Control using PWM & L293D Examples", Deep Blue Embedded, [Online]. Available: https://deepblueembedded.com/stm32-dc-motor-speed-control-pwm-l293d-examples/ (accessed on 04/19/2023).
11) "KiCad 6 STM32 PCB Design Full Tutorial - Phil's Lab #65", YouTube, July 5, 2022, [Online]. Available: https://www.youtube.com/watch?v=aVUqaB0IMh4 (accessed on [insert date accessed]).
12) "LM350 3.0 A, Adjustable Output, Positive Voltage Regulator", Mouser Electronics, [Online]. Available: https://www.mouser.com/datasheet/2/308/lm350-d-1192744.pdf (accessed on 02/23/2023).
13) "IEEE Code of Ethics", IEEE, [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (accessed on 02/11/2023).
14) "Code of Ethics: The Code Affirms an Obligation of Computing Professionals to Use Their Skills for the Benefit of Society", ACM, [Online]. Available: https://www.acm.org/code-of-ethics (accessed on 03/26/2023).
15) "Micropeta," LCD 1602 Parallel Display. [Online]. Available: https://www.micropeta.com/video60. (Accessed on 04/16/2023).
16) "Micropeta," HC-SR04 Ultrasonic Sensor. [Online]. Available: https://www.micropeta.com/video42. (Accessed on 04/17/2023).
17) "Autonomous Card Shuffler and dealer (ECE 445 Senior Design Project)," YouTube, 27-Apr-2023. [Online]. Available: https://www.youtube.com/watch?v=IucR9P-51Fk&feature=youtu.be. (Accessed on 04/28/2023).].
18) "Salary Averages | ECE ILLINOIS." ECE ILLINOIS. https://ece.illinois.edu/admissions/why-ece/salary-averages. (Accessed on 04/18/2023).

19) "Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino." Random Nerd Tutorials. https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/#:~:text=The%20ultrasound%20transmitter%20(trig%20pin,the%20reflected%20sound%20. (accessed 03/29/2023).

20) Texas Instruments. (2014). L293/L293D Quadruple Half-H Drivers [Data Sheet]. Available: https://www.ti.com/lit/ds/symlink/l293.pd (accessed 03/05/2023).

7 Appendix A: Bill of Materials

Below is a list of parts that are required for the making of one autonomous card dealer.

| Table 0 Parts Costs						
Part	**Part Number**	**Manufacturer**	**QTY**	**Retail Cost ($)**	**Bulk Purchase Cost ($)**	**Actual Cost ($)**
Microcontroller and Connectors to Subsystems						
Microcontroller	STM32F103C8T6	STMicroelectronics (Ordered by Rohit)	2	6.66	-	13.32
CAP CER 0.1UF 50V X7R 0402	GRM155R71H104ME14J	Murata Electronics	1	0.10	-	0.10
CAP CER 10UF 6.3V X5R 0402	GRM155R60J106ME05D	Murata Electronics	1	0.10	-	0.10
CAP CER 1UF 25V	GRM155R61E105KE11J	Murata Electronics	10	-	0.063	0.63
SWITCH ROCKER SPST OFF-ON	RF1-1A-DC-2-R-1	Switch Components	1	1.26	-	1.26
SWITCH SLIDE SPDT 200MA 30V	EG1218	E-Switch (Ordered 3/19)	1	0.68	-	0.68
CRYSTAL 16.0000MHZ 10PF SMD	ECS-160-10-36Q-ES-TR	ECS Inc.	1	0.95	-	0.95
CAP CER 10PF 25V C0G/NP0 0201	GRM0335C1E100JA01J	Murata Electronics	10	-	0.01	0.10
RES SMD 1.5K OHM 1% 1/10W 0603	RT0603FRE071K5L	YAGEO	10	-	0.053	0.53
CONN RCPT USB2.0 MICRO B SMD R/A	10118194-0001LF	Amphenol ICC (FCI)	1	0.42	-	0.42
RES SMD 2K OHM 1% 1/10W 0603	RT0603FRE072KL	YAGEO	10	-	0.053	0.53
CONNECTOR (2)	DF13-2P-1.25DSA(05)	Hirose Electric Co Ltd	1	0.30	-	0.30
CONN HEADER VERT 4POS 2.5MM	B4B-XH-A(LF)(SN)	JST Sales America Inc.	2	-	0.21	0.42
CONN HEADER VERT 3POS 2.5MM	B3B-XH-A(LF)(SN)	JST Sales America Inc.	1	0.18	-	0.18
Power Subsystem						
IC REG LINEAR POS ADJ 3A TO220-3	LM350T	Texas Instruments (Ordered by Rohit)	3	2.97	-	8.91
FUSE BRD MNT 2.5A 63VAC/VDC 1206	0685T2500-01	Bel Fuse Inc. (Ordered 3/19) (Used these fuses)	6	-	$0.28	$1.68

FUSE BOARD MOUNT 2.5A 32VDC 0603	ERB-RE2R50V	Panasonic Electronic Components	6	-	0.26	1.56
RES 240 OHM 5% 1/16W 0402	RC0402JR-07240RL	YAGEO	10	-	0.01	0.11
RES 910 OHM 5% 1/16W 0402	RC0402JR-07910RL	YAGEO	10	-	0.01	0.10
RES 680 OHM 5% 1/16W 0402	RC0402JR-07680RL	YAGEO	10	-	0.01	0.11
RES 390 OHM 1% 1/16W 0402	RC0402FR-13390RL	YAGEO	10	-	0.01	0.13
Motor Subsystem						
IC MTR DRV BIPOLAR 4.5-36V 16DIP	L293DNE	Texas Instruments (Ordered by Rohit)	1	$4.29		4.29
IC MTR DRV BIPOLAR 4.5-36V 16DIP	L293DNE	Texas Instruments (Ordered 3/30)	1	4.04	-	4.04
Shuffler		Rareidel	1	15.99		15.99
Servo Generic High Torque	ROB-11965	SparkFun	1	13.95		13.95
CONNECTOR (2)	DF13-2P-1.25DSA (05)	Hirose Electric Co Ltd	2	-	0.30	0.60
Sensing Subsystem						
Ultrasonic Sensor	375-HC-SR04	OSEPP Electronics	1	6.55		6.55
User Interface Subsystem						
RES SMD 10K OHM 1% 1/4W 1206	RT1206FRE0710KL	YAGEO	10	-	0.126	1.16
CONNECTOR (2)	DF13-2P-1.25DSA (05)	Hirose Electric Co Ltd	4	-	0.30	1.20
Extras						
CONN HEADER VERT 2POS 2.54MM	0022232021	Molex	10	-	0.168	1.68
CAP CER 1UF 25V X7R 1206	CL31B105KAHNNNE	Samsung Electro-Mechanics	10	-	0.094	0.94
CAP CER 10PF 500V C0G/NPO 1206	CC1206JRNPOBBN100	YAGEO	10	-	0.165	1.65
RES 2K OHM 5% 1/4W 1206	RC1206JR-072KL	YAGEO	10	-	0.035	0.35
RES 1.5K OHM 5% 1/4W 1206	RC1206JR-071K5L	YAGEO	10	-	0.035	0.35
CAP CER 0.1UF 250V X7R 1206	CL31B104KEHSFNE	Samsung Electro-Mechanics	10	-	0.155	1.55
CAP CER 10UF 25V X5R 1206	CL31A106KAHNNNE	Samsung Electro-Mechanics	10	-	0.128	1.28
RES 910 OHM 1% 1/4W 1206	RC1206FR-07910RL	YAGEO	10	-	0.038	0.38
SWITCH SLIDE SPDT 300MA 6V	JS102011SAQN	C&K	1	0.64		0.64
RES 390 OHM 5% 1/4W 1206	RC1206JR-07390RL	YAGEO	10		0.033	0.33

Item	Part Number	Manufacturer	Qty	Unit Price		Total
RES 680 OHM 1% 1/4W 1206	RC1206FR-07680RL	YAGEO	10		0.038	0.38
RES 240 OHM 5% 1/4W 1206	RC1206JR-07240RL	YAGEO	10		0.035	0.35
IC MCU 32BIT 64KB FLASH 48LQFP	STM32F103C8T6	STMicroelectronics (Ordered 4/14)	3	6.00		18.00
IC MTR DRV BIPOLAR 4.5-36V 16DIP	L293DNE	Texas Instruments (Ordered 4/18)	3	3.64		10.92
IC MTR DRV BIPOLAR 4.5-36V 16DIP	L293DNE	Texas Instruments (Ordered by Adam)	1	3.64		3.64
Amazon Basics 8-pack 9V Alkaline Performance All Purpose Batteries		Amazon Basics (Ordered by Rohit)	1	12.99		12.99
NUCLEO-F411RE STM32 Nucleo-64 Development Board with STM32F411RE MCU	Amazon	STMicrocontrollers (Ordered by Rohit)	1	26.22		26.22
Consumer Battery & Photo Battery 9V RECTANGLE	MN1604	Duracell (Ordered by Rohit)	4	2.52		10.08
Total			252			171.63

8 Appendix B Requirement/Verification Table: Power

Table 1 System Requirements and Verifications for Power Subsystem

Requirement	Verification	Verification status (Y or N)
Make sure that the 9V battery that we are using to power our entire project is providing 9 ± 0.1V. Otherwise, we might not have enough power for the system to function. *Make sure that the two 9V batteries that we are using to power our entire project are providing 9 ± 0.5V. Otherwise, we might not have enough power for the system to function.*	Change was made because we realized that 9V isn't enough to support the load of our entire project, so we increased the voltage to 18V which is two 9V batteries in series. We also made the tolerance (0.1V originally) a little higher (0.5V now) because the 9V batteries we were using usually started with a higher value than 9.0V. 1. Take the terminals of an oscilloscope and place them on the two terminals of the two 9V batteries, and then take a note of the readings. 2. Given that they are approximately 9 ± 0.5V, hook them up to the remainder of the power system and run the other power tests. 3. In the case that it is not outputting 9 ± 0.5V, the battery may be defective so replace it. Results: We measured the voltage output of all 6 9V batteries that we purchased using a multimeter and got the following readings: Battery 1 – 9.32V Battery 2 – 9.23V Battery 3 – 9.30V Battery 4 – 9.26V Battery 5 – 9.12V Battery 6 – 9.08V	Y
Must provide stable and regulated voltages of 6.0 ± 0.1V, 5.0 ± 0.1V, and 3.3 ± 0.1V to power the various subsystems of our project. *Must provide stable and regulated voltages of 13.0 ± 0.5V, 6.0 ± 0.5, 5.0 ± 0.5V, and 3.3 ± 0.5V to power the various subsystems of our project.*	The change was made because after testing the shuffling motors through the motor driver using 9V, it was not enough to power both shuffling motors despite running them one at a time while shuffling. So, we decided to add another regulator to regulate from 18V to 13V (for shuffling motors), 13V to 6V, 5V, and so on. We also made the tolerance (0.1V originally) a little higher (0.5V now) because the 9V batteries we were using usually started with a higher value than 9.0V. $$V_{OUT} = V_{REF}\left(1 + \frac{R2}{R1}\right) + I_{ADJ}\,R2.$$	Y
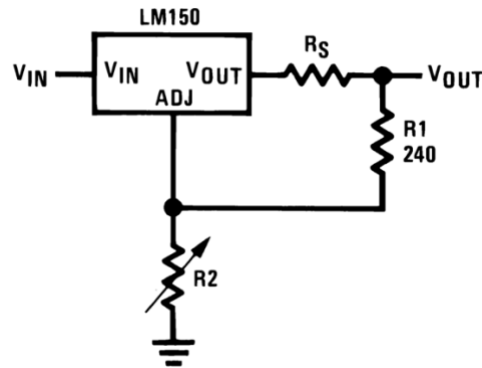
1. Set up the voltage regulator circuit for 13V output from the schematic on a breadboard.
2. Using the 18V battery as the input to the circuit, probe at the output to ensure that the output is $13.0 \pm 0.5V$.
3. If the output isn't $13.0 \pm 0.5V$, try various resistor values for R2 to determine which one would give us a more accurate output.
4. Set up the voltage regulator circuit for 6V output from the schematic on a breadboard.
5. Using the 12V voltage regulator output as the input to the circuit, probe at the output to ensure that the output is $6.0 \pm 0.1V$.
6. If the output isn't $6.0 \pm 0.5V$, try various resistor values for R2 to determine which one would give us a more accurate output.
7. Set up the voltage regulator circuit for 5V output from the schematic on a breadboard.
8. Using the 6V voltage regulator output as the input to the circuit, probe at the output to ensure that the output is $5.0 \pm 0.1V$.
9. If the output isn't $5.0 \pm 0.1V$, try various resistor values for R2 to determine which one would give us a more accurate output.
10. Set up the voltage regulator circuit for 3.3V output from the schematic on a breadboard.
11. Using the 5V voltage regulator output as the input to the circuit, probe at the output to ensure that the output is $3.3 \pm 0.1V$.
12. If the output isn't $3.3 \pm 0.1V$, try various resistor values for R2 to determine which one would give us a more accurate output.
13. Note down which resistor values work for R2 and if any resistors had to be modified.

Results:
13V Voltage Regulator – R2 = 220Ω, Output = 13.39V
6V Voltage Regulator – R2 = 880Ω, Output = 6.14V
5V Voltage Regulator – R2 = 760Ω, Output = 5.47V

	1. 3.3V Voltage Regulator – R2 = 430Ω, Output = 3.43V	
Our servo motor (the highest current consuming source in our device) has a maximum operating current of 2.5A so we need to make sure that our power subsystem is capable of being able to support this and not overheating.	1. Firstly, the voltage regulators that we chose in our circuit can support up to 3A. 2. We will run our servo motor using our 5V voltage regulator source. 3. While the servo motor is running, we will probe the output current of the servo using a multimeter and check if it's under 2.5A. 4. We will also make sure that none of the pieces in our circuit are overheating by checking the temperature of the pieces (by touch). 5. If there are any parts of the circuit that are overheating and/or the current across the servo motor is above 2.5A, then we will then add a resistor to decrease the current that is flowing through the circuit. 6. We will make note of the places where the temperature was high, and then the effect that the resistors had on this temperature. 7. We will continue this process until the circuit doesn't overheat and everything is working in a proper manner. Results: 2. Our servo motor was outputting ~2.2A at maximum while running and none of the components in the power subsystem and the motor subsystem were running hot, so we were good to go.	Y

9 Appendix C Requirement/Verification Table: User Interface

Table 2 System Requirements and Verifications for User Interface Subsystem

Requirement	Verification	Verification status (Y or N)
Must be able to communicate the data from the buttons (current game mode, number of players, shuffling, dealing) to the microcontroller and then communicate that data through I2C protocol to the 7-segment display. *Must be able to communicate the data from the buttons (current game mode, number of players, shuffling, dealing) to the microcontroller and then communicate with the LCD display.*	Change made because the LCD display allowed our Autonomous Card Dealer device to be much easier to interface with and get instant feedback on the state and settings set of the machine. The communication protocol is not I2C, but it is standard across most commercial LCD displays (RW, RS, EN, D3-0 pins). 1. The 4 buttons are hooked up to digital I/O pins on the microcontroller. The first step is to make sure that when a button is being pressed, the microcontroller is receiving a high value for that I/O pin. 2. If that isn't working, investigate the connections and the datasheet for the button and microcontroller. If it is working, then we can read that data into our software and store them as variables. 3. Every time one of the buttons is pressed, the value should be changing in the software, the value is dependent on the button. Ensure that is happening, if not something is wrong with the software. Debug the software until we get the expected results for variable values. 4. The button data should be stored in 4 different variables, we will communicate this data to the LCD display through the 4 data pins, the RW and RS signals. Note down if pressing down the buttons increments the count in software, and if not, how the problem was fixed.	Y
The User Interface must allow users to view their inputs on the LCD. The User will be prompted with messages that indicate what settings are being changed and the state that the machine is in: Welcome, Enter Num Players, Select Game Mode, Dealing Mode "hold on tight", Shuffling Mode "hold on tight",	The change that was made involves separating the previous requirement and verification into two separate parts. It allows for a more distinguishable requirement and makes it easier to verify. The verification procedure for this: 1. plug in power, turn on microcontroller, the expected message on the LCD is "Welcome :)" 2. press the num_players button, the expected message on the LCD will be "Please Select Num Players: %d", num_players. 3. press the game_modes button, the expected message on the LCD will be "Please Select Game Modes: %d", game_mode_select. 4. press the shuffling button, the expected message on the LCD will be "Shuffling Now, Hold on Tight". 5. press the shuffling button, the expected message on the LCD will be "Dealing Now, Hold on Tight".	Y

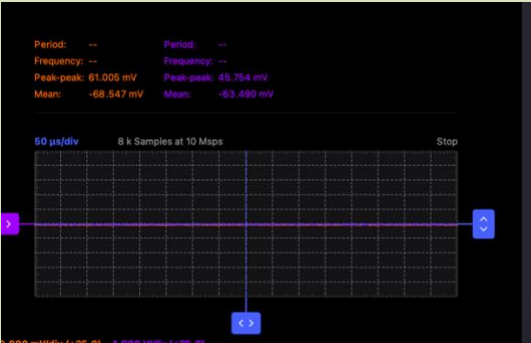|		
	The above are the sets of states and setting changes that the users can command and input to the system.	
Must be able to display four hex digits to the 7-segment display to represent the four buttons. It must have enough brightness on the 5V operating voltage that it can clearly be seen with your eyes.	This change increased the functionality of the device, making it easier for users to understand what each they are entering and how they are changing the state of the machine. The user can now communicate with the device and get instant feedback on what is going on.	Y
To increase the complexity of the project, our group decided to no longer use I2C communication protocol. Instead, we implemented code to communicate and write data to the on-device processor of an LCD.	Since we are no longer using a seven-segment display, the verification process will still hold similar expectations, just a different device and communication protocol. 1. We must communicate to the processor of the LCD via our 4 data pins. 2. We must implement a function to write a command to the lcd, lcd_write_command, using the rs, en, and data signals. 3. We must implement a function to write data to the lcd, lcd_write_data, using the rs, en, and data signals and its own separate data commands. 4. We must implement a function to write a string onto the first and second rows of the LCD display. 5. We then want to verify that when we send strings to the LCD display, they are visible and printing characters as expected. If this isn't occurring, note down what the display is outputting and potential solutions to fix this.	

10 Appendix D Requirement/Verification Table: Sensing

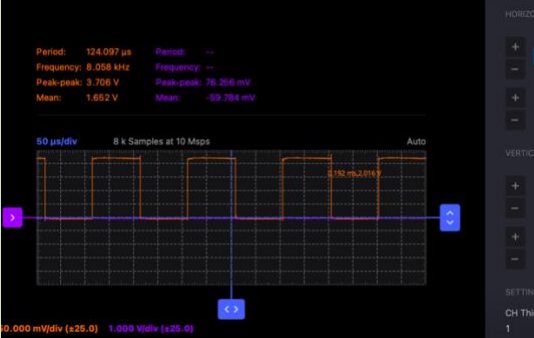**Table 3 System Requirements and Verifications for Sensing Subsystem**

Requirement	Verification	Verification status (Y or N)
Must be mounted on device such that the waves hit the people and bounce back	1. We will mount it at a correct angle so that it will hit the players no matter their height. 2. We will verify that this is working by testing with a group of people. We will have people sit around a table, and every time we move the ultrasonic sensor across a player, we should get some feedback. We will make sure we are getting some readings. We will write in our lab notebook about if when we move across a player, we get a reading or not. If we are not getting a reading, there might be something wrong with our mounting or our code.	Y
Must get accurate readings so that the device will know when to start/stop dealing cards as it rotates around the table from player to player.	1. We will verify firstly that we are getting some type of readings from the ultrasonic sensor when we move across players. 2. We can also print out the readings that we are getting from the players to see if they are accurate and reflect how far away the players are. We want to make sure that we are accurately capturing distances, so we know how far to launch the cards. 3. Whenever we get a reading, we want to stop the rotation and launch out a card, we can visually verify that this is occurring. We will write the readings that we get in our notebook. We will see if the readings are accurate regarding the distance of the player. If they aren't accurate, we might have to adjust the speed of the motors for which the card is being launched so that we are launching the cards the correct distance.	Y
Must be able to get readings from at least 2 meters away as that will ensure that all players at the table will be seen and dealt cards. *Must be able to get readings from at least 0.5 meters away as that will ensure that all players at the table will be seen and be dealt cards.*	*Changed from 2m to 0.5m due to the unreliability of the ultrasonic sensor that we have.* 1. We want this to be possible because some tables are a lot longer, so we need to account for this and make sure we can tell how far the players are, even if they are at the edge of the table. 2. This can be verified by having a player stand 0.5 meters away from the ultrasonic sensor, and then running the sensor and checking if we get any readings. If we get some type of reading, this means that it can read things that are 0.5 meters away. 3. The next thing that we will have to verify is that the readings are actually accurate and tell us that the player is 0.5 meters away. 4. This can also be verified visually once our device is working by checking that a player who is 0.5 meters away can get their card distributed to them.	Y

	In our notebook, we will write more about our findings from the ultrasonic readings. We will write about how well it reads from 0.5 meters away, and what the max range might be.	
Utilize this method to verify the accuracy of the ultrasonic sensor. Every measured distance should have an expected elapsed time in ms.	To test the accuracy of the ultrasonic sensor, we can use the following table of expected time_elapsed. Measured distance (cm) / Expected Time Elapsed (ms): 10 / 0.5831 20 / 1.1662 30 / 1.7493 40 / 2.3324 50 / 2.9155	Y

Measured distance (cm)	Expected Time Elapsed (ms)
10	0.5831
20	1.1662
30	1.7493
40	2.3324
50	2.9155

11 Appendix E Requirement/Verification Table: Motors

Table 1 System Requirements and Verifications for Microcontroller & Motor Driver Subsystem

Requirement	Verification	Verification status (Y or N)
Microcontroller: PWM pins must be able to output a stable 0-100% duty cycle PWM wave.	1. Set up our STM32 microcontroller on a breadboard and connect it to a power supply. 2. Connect an oscilloscope probe to the PWM output pins we will be using on the board. Connect the ground clip of the oscilloscope probe to the ground of the microcontroller. 3. Write a simple program to configure the PWM output pins as an output and set the output to a square wave with duty cycle's 0%, 25%, 50%, 75%, and 100%. 4. Monitor the oscilloscope outputs and ensure they correspond to the programmed duty cycle. Results: 0% Duty Cycle 75% Duty Cycle	Y

Period: 120.867 µs   Period: --
Frequency: 8.274 kHz   Frequency: --
Peak-peak: 3.782 V   Peak-peak: 76.256 mV
Mean: 758.714 mV   Mean: -59.357 mV

50 µs/div   8 k Samples at 10 Msps   Auto

750.000 mV/div (±25.0)   1.000 V/div (±25.0)

25% Duty Cycle

Period: 124.097 µs   Period: --
Frequency: 8.058 kHz   Frequency: --
Peak-peak: 3.706 V   Peak-peak: 76.256 mV
Mean: 1.652 V   Mean: -59.784 mV

50 µs/div   8 k Samples at 10 Msps   Auto

50.000 mV/div (±25.0)   1.000 V/div (±25.0)

50% Duty Cycle

Period: --   Period: --
Frequency: --   Frequency: --
Peak-peak: 36.603 mV   Peak-peak: 0.000 V
Mean: 3.239 V   Mean: -66.382 mV

Zoom: 30.156 ms/div   3.2 k Samples at 1 ksps   Stop

100% Duty Cycle | |
| *Motor Driver: Each input pin (1, 2EN, 3,4EN, 1A, 2A, 3A, 4A) must switch on and off when connected and given a logic high.* | 1. Connect the motor driver to a power supply that matches the specified voltage range for the device.
2. Connect the inputs to a logic high voltage source (3.3V).
3. Use a multimeter to verify that the voltage level on the input pins matches the voltage source.
4. Use an oscilloscope to verify that the input pins switch on and off as expected when the logic high voltage is applied.
5. Connect a DC motor to the output pins of the motor driver.
6. Repeat steps 2-4 with the motor driver connected to the motor, to verify that the motor driver outputs the switch on and off as expected when given a logic high.

Results:
For each pin tested, we used the ADALM 2000 and its oscilloscope option to check if a logic high was being fed in | Y |

Requirement	Verification	Verification status (Y or N)
	when connected. For each directional control pin, we saw the waveform shoot up to ~3.3V when connected. Regarding the Enable signals which take in a PWM wave, we simply set the duty cycle to 100%. Reference previous requirements for details on duty cycle measurements.	
Motor Driver: The Y1, Y2, Y3, and Y4 pins must be able to run a DC motor with the corresponding enable pin (1,2EN or 3,4EN).	1. Connect the motor driver to a power supply that matches the specified voltage range for the device. 2. Connect a DC motor to the output pins Y1, Y2, Y3, and Y4 of the motor driver. 3. Connect the corresponding enable pins 1,2EN or 3,4EN to a logic high voltage source (usually 5V or 3.3V). 4. Use a multimeter to verify that the voltage level on the enable pins matches the voltage source. 5. Apply power to the motor driver and verify that the motor runs when the corresponding enable pin is set to high. 6. Use an oscilloscope to verify that the motor output voltage is stable and matches the voltage supplied by the power supply. 7. Measure the current drawn by the motor using a multimeter. Ensure that the current does not exceed the rated current of the motor driver or the motor. Reverse the polarity of the power supply to the motor driver and repeat steps 5-7 to verify that the motor runs in the opposite direction.	Y

Table 2 System Requirements and Verifications for Rotating Servo Motor Subsystem

Requirement	Verification	Verification status (Y or N)
Rotation Servo Motor must not be defective	1. Connect a 6V power supply to the servo motor. 2. Connect the signal wire of the servo motor to a PWM output pin of a microcontroller. 3. Write a simple program for the microcontroller to send a PWM signal to the servo motor, commanding it to rotate to 90 degrees. 4. Observe the servo motor and ensure that it rotates to the commanded angle smoothly and accurately. 5. If the servo motor does not rotate or rotates erratically, double-check the connections, and ensure that the power supply is providing a stable 6V. If the servo motor still does not function properly, it may be defective or damaged. Consider replacing it if necessary.	Y
Rotation servo motor must output enough torque (>80 oz-in) to rotate the device	1. Mount the servo motor onto some surface (table with table). 2. Connect a 6V power supply to the servo motor. 3. Connect the signal wire of the servo motor to a PWM output pin of a microcontroller.	Y

	4. Write a simple program for the microcontroller to send a PWM signal to the servo motor, commanding it to rotate 180 degrees. 5. Slowly increase the load torque on the motor shaft using a torque wrench and gradually increase the torque in increments of 10 oz-in while monitoring the motor's response. 6. At each torque level, observe the motor's ability to reach the 180-degree rotation angle target. Ensure that the motor is not struggling or vibrating excessively, which can indicate insufficient torque output. If the motor can maintain the target angle at a torque level of at least 80 oz-in, then it is strong enough to rotate our whole device.	

Must have the capability to rotate at least 180 in increments of 11.25, 12.9, 15, 18, 22.5, 30, 45, and 90 degrees (1-8 players) clockwise in a controlled manner.	The servo motor given to us by the mechanical team was a 360-degree servo not made for precise position control, thus feeding in target angles through PWM and the typical servo motor frequency rating of 50 Hz (through timer channels on the microcontroller board modified to suit the typical 0.5ms-2.5ms pulse width range) was not an option with the given motor. The motor given to us was suited for 500μs-4000μs thus we had to make changes to the timer channel prescaler and counter period values and adjust them until we got somewhat precise control. 360-degree servo motors for reference are continuous and thus PWM inputs control the speed and direction of the motor rather than a target angle like 180-degree servo motors.
	
	1. Connect the servo motor to our 5V regulated power supply and to the microcontroller.
	2. Write and upload a simple program to the microcontroller to rotate the motor at a specified angle.
	3. Test the servo motor's ability to rotate in increments of 11.25, 12.9, 15, 18, 22.5, 30, 45, and 90 degrees. For each increment, do the following:
	    1. Set the motor to rotate clockwise and slowest speed according to timer channel settings and duty cycle on PWM microcontroller pin. Delay a set amount of time. After the delay, set PWM duty cycle signal to stationary.
	    2. Use the protractor to verify that the servo motor has moved to the correct angle.
	    3. If the incorrect angle is, adjust the delay and try again.
	4. Verify that the servo motor moves smoothly and without any jerky or sudden movements.
	5. Record the results and ensure that the servo motor can rotate properly by all the increments mentioned above.
		Y

| | Results:
11.25° – Measurement: 11°
12.9° – Measurement: 13°
15° – Measurement: 16°
18° – Measurement: 18°
22.5° – Measurement: 22°
30° – Measurement: 30°
45° – Measurement: 46°
90° – Measurement: 90° | |

Table 3   System Requirements and Verifications for Dealing DC Motor Subsystem

Requirement	Verification	Verification status (Y or N)
Dealing DC Motor must not be defective	1. Connect a 6V power supply to the motor. 2. Connect a switch to the power supply and the motor. 3. Close the switch and observe the motor. 4. If the motor starts spinning, it is not defective. 5. If the motor does not start spinning, double-check the connections and ensure that the power supply is providing a stable 6V. Try reversing the polarity of the power supply and repeating step 3 to see if the motor spins in the opposite direction. If the motor still does not spin or vibrates while spinning, the motor may be defective or damaged. Consider replacing it if necessary.	Y
Dealing DC motor needs at least 50 oz-in of torque to successfully launch a standard playing card (weighs roughly 100-150g) to 1 meter. *Dealing DC motor needs at least 50 oz-in of torque to successfully launch a standard playing card (weighs roughly 100-150g) to 5 centimeters.*	Change made because we realized that the DC motor that we are using doesn't have a whole lot of torque, even though we are running it with a 100 percent duty cycle, it still is unable to travel very far, we also realized that 5 centimeters is large enough and the players can reach for their cards. 1. Mount the motor with a load onto some surface (table with table again for example). 2. Attach a torque meter. 3. Connect the motor to a variable DC power supply and set the voltage to 6V. 4. Gradually increase the voltage until the motor reaches its maximum rated current and record the corresponding torque reading from a torque meter attached to the motor shaft. If the motor can output at least 55 oz-in of torque, it is considered suitable for dealing out a single standard playing card.	Y