ChipCaddy: A Home Poker Game Solution

ECE 445: Senior Design Laboratory

Team #16

Anish Rajesh, Justin Wang, Marvin Camras

Professor: Olga Mironenko

TA: Nikhil Arora

Abstract

ChipCaddy is a sensor-based device, specifically catered to the home poker game setting, that seeks to mitigate some of the intermediate parts of poker play while maximizing the number of hands played in a poker session. The reason for the "home poker game" distinction is due to the simplicity and feasibility of the design. The main features of the design include the pot-counter, chromatically organized bins, and reset as well as split pot capabilities. All three of the project's contributors are avid poker players, and the final product is directly inspired by the pitfalls of today's home poker gameplay and the growing poker market. The design of ChipCaddy was fun but challenging and completed modularly through the integration of five subsystem modules: the control subsystem, motor subsystem, sensing subsystem, power subsystem, and user-interface subsystem.

1. Introduction **1**
1.1. System Functionality 1
1.2. Subsystem Overview 2
2. Design **4**
2.1. Power Subsystem 4
2.2. Motor Subsystem 5
2.3. Control Subsystem 6
2.4. Sensing Subsystem 7
2.5. User Interface Subsystem 9
3. Design Verification **10**
3.1. Power Subsystem 10
3.1.1. Requirement 1a 10
3.1.2. Requirement 1b 11
3.1.3. Requirement 1c 11
3.2. Control Subsystem 11
3.2.1. Requirement 2a 11
3.2.2. Requirement 2b 12
3.2.3. Requirement 2c 12
3.3. Sensing Subsystem 12
3.3.1. Requirement 3a 12
3.3.2. Requirement 3b 12
3.4. Motor Subsystem 12
3.4.1. Requirement 4 12
3.5. User Interface Subsystem 13
3.5.1. Requirement 5a 13
3.5.2. Requirement 5b 13
3.5.3. Requirement 5c 13
3.5.4. Requirement 5d 13
3.5.5. Requirement 5e 13
4. Costs and Schedule **14**
4.1. Parts 14
4.2. Labor 14
4.3. Schedule 15
5. Conclusion **16**
5.1. Accomplishments and Uncertainties 16
5.2. Ethical Considerations 16
5.3. Future Work 17
References **18**
Appendix A: Requirement and Verification Table 19

1. Introduction

According to a market research study published by Zion Market Research, the demand analysis of Global Trading Card Game Market size & share revenue was valued at $6.39 Bn in 2022 and is estimated to grow about $11.57 Bn by 2030 [10]. Although gambling has its pitfalls, it has become one of the world's most predominant pastimes and as a result created a market of equal size. As the market for card games increases, so does the need for accurate, secure, and efficient home game systems. Current home games are set up with a simple set of chips, cards, and players, resulting in large amounts of time wasted counting, sorting, and dealing chips. Additionally, poker is a game highly dependent on probabilities associated with the number of cards in the deck as well as the current value of the pot at each betting street. The existing solutions to these problems are only present in casinos or public settings with near bottomless endowment and are not appropriate for the casual home setting.

To promote ease of play, and maximum efficiency in the number of hands played, we have come up with a solution featuring a combination of sensors, motors, and internal logic to sort poker chips and display the current value of the pot for all players. The usage of the device involves players placing the chips for their bet into the funnel apparatus of our device, and continuing gameplay as the device appends the count for each chip. The device will simultaneously organize the chips by color into their assigned bins as it appends the pot count on the LCD for all players to see. At the conclusion of the hand, players will be able to split the pot depending on the number of winners, reset the pot count, and empty the bins for the next hand. The beauty of this solution is that players do not have to perform any extra steps during the hand, compared to gameplay without our device. Thus, we are able to achieve increased efficiency in poker gameplay without creating extra nuisances for players. The specific engineering steps taken to fully implement this solution will be detailed in the latter portion of this document.

1.1. System Functionality

The device contains a color sensor module beneath the funnel of the device, which is responsible for differentiating our four different colored chips based on the frequency of light reflected upon them. After each successive color reading is a rotation of the servo to the respective bin, a retraction of the linear actuator to push out the chip, a contraction of the linear actuator, and finally an appending of the pot count for the chip sorted. After all betting streets are finished and the hand is over, the winner(s) of the pot may split the pot for the number of winners with multiple presses of the split button, reset the pot for the next hand, and collect their sorted chips from the bins. The LCD will then display a final tally of the total number of chips in the pot to safeguard against chip loss and theft before reading "Pot Count: $0.00" prior to the start of the next hand. To fully serve all these functions, four high level requirements were pre-defined, some of which were surpassed. These requirements are as detailed below:

1. The device should append the count within 5 seconds of the chip being read by the color sensor.

2. Upon ejecting all chips from our contraption, the winner of the pot will be able to reset the pot count to 0.
3. In the case of split or chop pots the user will be able to manually choose the number of ways the pot will be split, and the respective color denominations for the largest division will be shown on the LCD.
4. The device will keep a tally of the number of chips counted, and the number of chips counted for each color. It will then ensure the sum of the number of each colored chip matches the total number of chips and display the total number of chips inserted on the display.

These high-level requirements are essential in ensuring our device fulfills its purpose in optimizing speed of poker gameplay, while eliminating the minor nuisances of the home poker game setting.

1.2. Subsystem Overview

The device's ability to meet the high-level requirements noted above depends on the proper modular functionality of each of our five subsystems. These subsystem modules are the power subsystem, control subsystem, sensing subsystem, motor subsystem, and the user-interface subsystem. A description of each of the subsystems are shown below:

I. *Power:* To enhance the portability and usability of our design, we powered our device with a 6 V external battery pack. The 6 volts from the battery pack are directly used to power the linear actuator and rotational servo used for sorting chips. Additionally, a linear voltage regulator is used to down convert the 6 V supply to 3.3 V which is connected to our MCU, color sensor, and LCD.

II. *Control:* This subsystem involves the STM32 microcontroller. The MCU takes in readings from the sensing subsystem and the buttons from the user-interface subsystem. The MCU then moves the motor and actuator accordingly while displaying values on the LCD as needed.

III. *Sensing:* This subsystem involves the TCS3200 color sensor module. The sensor shines white light and receives reflected light as a current generated by its photodiode array. The generated current is then converted into a frequency used to differentiate the different colored chips. These frequency values are relayed to the MCU to perform internal logic for pot-counting and sorting.

IV. *Motor:* This subsystem involves the rotation servo motor and the linear actuator. The actuator and rotational servo motor receive pulse-width modulation signals from the MCU that trigger its movement. The logic coded into the MCU chooses the degree of rotation of the servo based on the color detected, as well as the subsequent retraction and contraction of the linear actuator.

V. *User-Interface:* This subsystem involves the LCD, and the two buttons: split and reset. The LCD will display pot count values based on color detection from the sensing subsystem and will also reflect changes triggered by the split and reset buttons. The logic for split and reset are performed by the control subsystem.

Not only do the subsystems need to function by themselves, but they need to be able function cohesively with the other subsystems. Many of the subsystems take in inputs from other subsystems for their function - our block diagram showing these connections is featured in Figure 1.

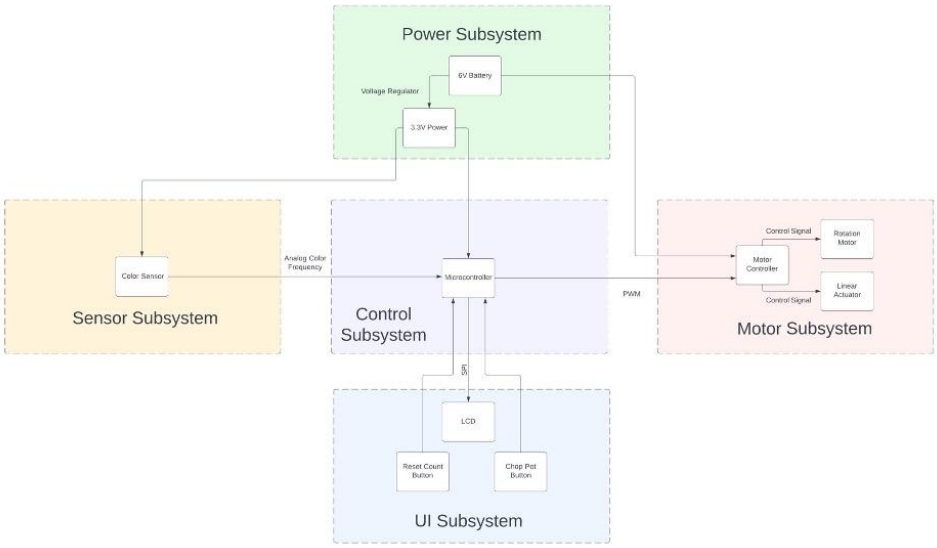

Fig. 1: High-Level Block Diagram

‘

2. Design

This section will convey the design process. For organizational purposes, this section will highlight the functionality and operation of each subsystem individually as well as how they communicate with the other subsystems to achieve collective functionality, as per the block diagram in Figure 1. Below in Figure 2 is an overall view of our PCB schematic.

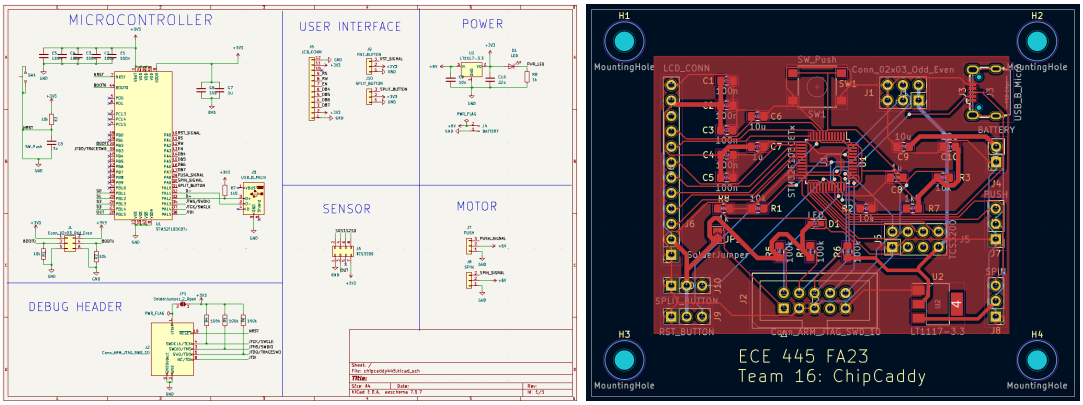

Fig. 2: Completed PCB Layout for ChipCaddy

2.1. Power Subsystem

Our device will be powered with a replaceable 6 V external battery pack. This design choice was made to avoid any need to plug into a socket, as our device is made to be a modular extension of a home poker game. The 6 V battery was specifically a NiMH battery pack, providing both modularity and the ability to be recharged. The need to find an outlet and avoid tripping over a large wire is something we wanted to avoid with our design. Prior to writing the design document, we planned to use a 9 V battery pack supply and two voltage regulators. Since the electrical components in our design required 6 V (motor subsystem) and 3.3 V (UI subsystem, control subsystem, and sensing subsystem) power, we quickly realized it made more sense to use a 6 V battery pack and a singular voltage regulator to step down from our initial 6 V input. The 3.3 V regulator was a straightforward design choice - the AZ1117 linear voltage regulator features a fixed 3.3 V output. We decided to use the LT1117 linear voltage regulator which serves as a direct alternative to the AZ1117. We utilized the design recommended in the device's datasheet [6] as pictured below. From a current perspective, the Tenergy 6 V Battery Pack that we use is rated at 2 A*h, meaning it can supply up to 2 A for an hour. The maximum current draw of all our peripherals added up is as follows: $I_{servo} + I_{actuator} + I_{sensor} + I_{LCD} + I_{MCU}$ = 180 mA + 460 mA + 10 mA + 3 mA + 30 mA = 683 mA. The current draw analysis tells us our voltage supply is more than adequate for running the peripherals required for full functionality of our device.

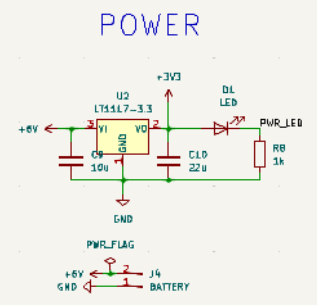

Figure 3: Power Subsystem Schematic

2.2. Motor Subsystem

The motor subsystem is in charge of the movement of our device. Our design features a HiTec 6 V rated rotational servo motor at the base for rotating and a 50 mm Actuonix linear actuator located at the side of the funnel base to dispense chips into their sorted bins. The motor at the base is capable of 210 degrees of rotation which is sufficient for rotating to all of the 4 bins. This motor was chosen because it could receive direct supply from our 6 V battery pack, has a minimal current draw, and can be interfaced easily with any MCU emitting PWM signals for control. The original design for our project included placement of the four bins along the four corners which would require a 360 degree rotation for our sorting motor. However, after discussions with the machine shop when moving along with building our device, there were concerns about wire entanglement. As the device rotated, we were worried that the wires coming from the PCB and peripherals would get caught on the multiple moving parts. Since our main goal was to count and sort the chips, we pivoted towards a design that contained all four bins within 210 degrees of each other, making the HiTec servo adequate. The Actuonix linear actuator was conveniently provided by the machine shop, and happened to take a 6 V supply with a long enough stroke length to push our chips to their respective bins. The servo motor and linear actuator are controlled through PWM signals that are outputted from the STM32 MCU.

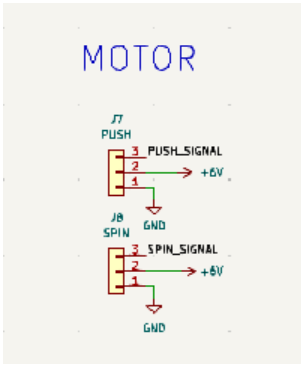

Figure 4: Motor Subsystem Schematic

We had planned to supply the power to the two motors through the 6 V battery, but for our final demonstration we used a 5 V output from the NUCLEO-F103RB development board. The lower voltage input directly impacted the amount of torque that our sorting motor was able to exert and because of that we had a little inconsistency with reaching the red bin.

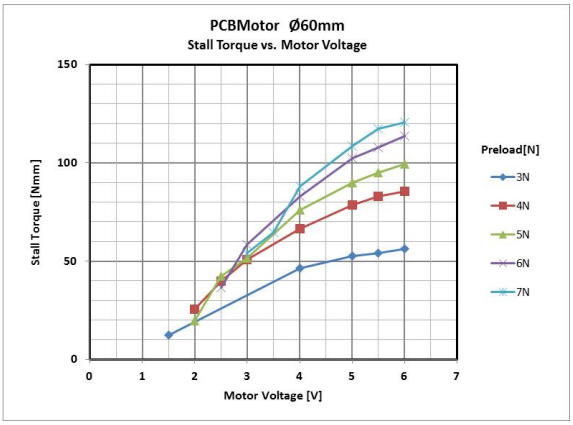

Fig. 5: Stall Torque v. Voltage Chart
Source: Adapted from [2]

We see from the voltage analysis shown in Fig. X that the stall torque of a servo motor is directly proportional to the input voltage that is provided. The stall torque of a servo motor increases as the voltage goes up due to the direct relationship between voltage and current in electric motors, as defined by Ohm's Law (V = IR, where V is voltage, I is current, and R is resistance). When the voltage supplied to the motor increases, it drives a higher current through the motor's windings, assuming the resistance remains relatively constant. Since the torque generated by an electric motor is proportional to the current flowing through it, an increase in voltage results in a stronger current and thus a higher magnetic field strength inside the motor. This enhanced magnetic field leads to greater torque output, particularly noticeable at the stall condition, which is when the motor is not rotating but is under maximum electrical load.

This issue was solved by adjusting our code, such that the motor would rotate slightly more as described in section 5.2 of this report. According to our analysis, if we had incorporated the PCB into our final device, this issue would not have risen.

2.3. Control Subsystem

The control subsystem is the brains of the operation. The control subsystem contains the 32-bit, 3.3V operable, STM32103C8T6 microcontroller. We originally planned to implement our design using the ESP32, however even though our project was logically complex it did not need the vast capabilities that the ESP32 had to offer, such as WiFi interface. Therefore, we thought the project would be simpler by utilizing the STM32 MCU instead. However, due to a severe lack of documentation it actually ended up being more challenging to use the STM32.

From a programming perspective, we sought to achieve full functionality on the Nucleo-64 development board, especially since we would be able to flash the program from the ST-link module on the board directly onto the MCU soldered to our PCB. We began by first attempting to write code on the STM32Cube IDE, but eventually pivoted to using the Arduino IDE paired with STMduino extension since it was more beginner-friendly. Using whatever minimal documentation on the internet we were able to successfully write code that interfaced with all our peripherals as intended. The most logically complex part of our code was configuring the split pot function. We utilized the greedy algorithm, which involves using the greatest valued chip for each division until this chip is depleted and then moving to the next greatest valued chip for each successive division. After doing this our code allowed for full individual subsystem functionality and overall functionality using the Nucleo-64 development board. On the last day before our demonstration we realized that our IC was soldered incorrectly on our PCB. After applying hot air to re-solder the MCU, the Arduino IDE was able to recognize the MCU on our PCB as a target and we were able to successfully flash our code onto our PCB. At this point, our design was quite robust and wired very carefully into the Nucleo-64 board. With fear of ruining the configuration by replacing the development board with our PCB, we thought it would be safest to demonstrate full functionality with the development board and show the PCB's functionality separately.

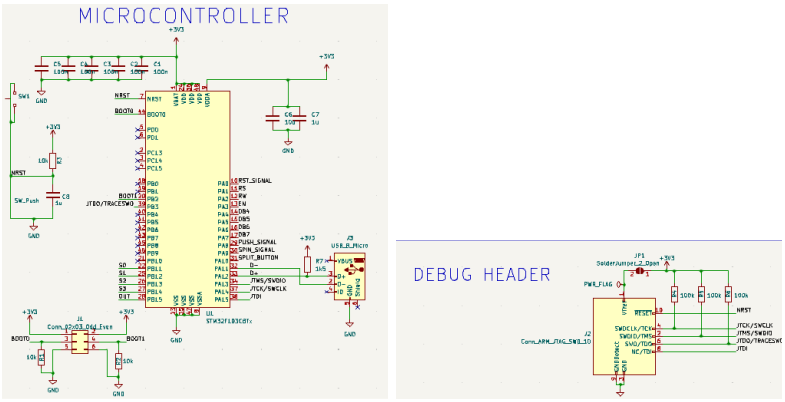

Figure 6: Control Subsystem Schematic

2.4. Sensing Subsystem

Our sensing subsystem serves as a vital component of our device, responsible for detecting the color chip at the bottom of the device and sending this signal to the microcontroller as a frequency. The microcontroller accepts this input and makes a conclusion on what color is detected. Based on whether red, green, blue, or white is sensed, the MCU will send PWM signals to the rotational motor to direct the aperture to the correct bin. The MCU will also append the total count onto the LCD based on what color is sensed.

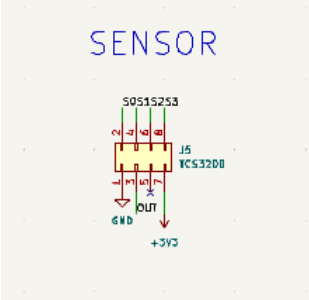

Figure 7: Sensing Subsystem Schematic

For ChipCaddy, we used a TCS3200 color sensor. The TCS3200 features a single-supply operation of 2.7 V to 5.5 V and utilizes programmable color light-to-frequency converters that combine configurable silicon photodiodes and a current-to-frequency converter on a single monolithic CMOS IC. The device shines light through four LEDs and the reflected light is detected and captured by the 8x8 array of photodiodes. 16 photodiodes have blue filters, 16 photodiodes have green filters, 16 photodiodes have red filters, and 16 photodiodes are clear with no filters. The photodiodes convert each color light to a current, which is converted to a frequency and sent as a square wave signal to the MCU.

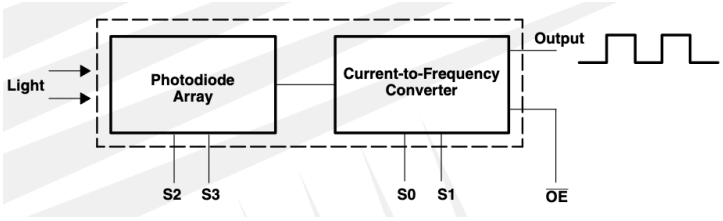

Fig. 8: Block Diagram of the TCS3200
Source: Adapted from [9]

The received frequency values were sometimes ambiguous, so it was imperative that our code could handle the wide range of values that were provided to the microcontroller.

```
String determineColor(int red, int green, int blue) {

  const int tolerance = 3;

  if ((isInRange(red, 15, tolerance) &&
       isInRange(green, 35, tolerance) &&
       isInRange(blue, 25, tolerance))) {
    return "Red";
  } else if (isInRange(red, 30, tolerance) &&
             isInRange(green, 24, tolerance) &&
             isInRange(blue, 20, tolerance)) {
    return "Green";
  } else if (isInRange(red, 33, tolerance) &&
             isInRange(green, 29, tolerance) &&
             isInRange(blue, 17, tolerance)) {
    return "Blue";
  } else if (isInRange(red, 10, tolerance) &&
             isInRange(green, 10, tolerance) &&
             isInRange(blue, 5, tolerance)) {
    return "White";
  } else {
    return "Unknown";
  }
}
```

Fig. 9: Code Snippet for Color Determination

Figure 9 shows our function determineColor() that was incorporated into our design. The function accepts the three output frequencies as arguments. If the frequencies fall within the range of values for a specific color, the MCU will conclude that the color chip is detected and send control signals to the motor and user interface subsystems.

2.5. User Interface Subsystem

The user-interface subsystem is the simplest out of the five subsystems. It consists of the NHD LCD, and two buttons designated for split and reset pot purposes. There was a large database of existing documentation for this component available.

Fig. 10: User Interface System with Buttons and LCD

For this subsystem, we started by showing text on the LCD using code from the Arduino IDE. The STM32 MCU is coded in the Arduino IDE, such that it communicates with the LCD via SPI protocol through the GPIO pins. Once we figured out how to map frequencies from the color sensor to integer values, we were able to get the pot-count to append on the LCD. From this point, we simply wrote conditional statements such that our logic for splitting the pot, and resetting the pot count occurred upon the press of the buttons. We wanted to be able to support situations in which more than two players won a hand. The code was configured such that with each press of the split pot button the pot would split three, then four ways and so on.

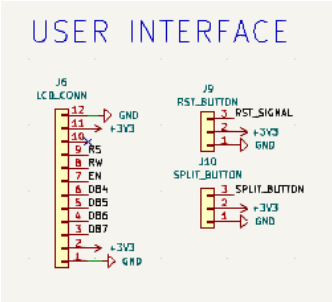

Figure 11: User Interface Subsystem Schematic

3. Design Verification

Our device features a variety of components each with different characteristics so it was important that we developed meticulous requirements and verification methods such that they could be verified by any individual regardless of their knowledge on the project. Below are the details of each of the requirements from Appendix A and the results of the respective verification procedures.

3.1. Power Subsystem

In this section, we will be discussing the results of verifying the requirements corresponding to the power subsystem. These requirements ensure proper voltage supply and current output of our regulators and battery supply. To be thorough, we added another requirement ensuring our regulator does not overheat.

3.1.1. Requirement 1a

The LT1117 voltage regulator provides 3.3 +/- 0.5% V output. Before even attempting to create our PCB schematic, we ran a number of simulations to verify our voltage regulation system. Below is a screenshot of a plot generated on LTSpice with a configuration identical to the description in the datasheet [6].

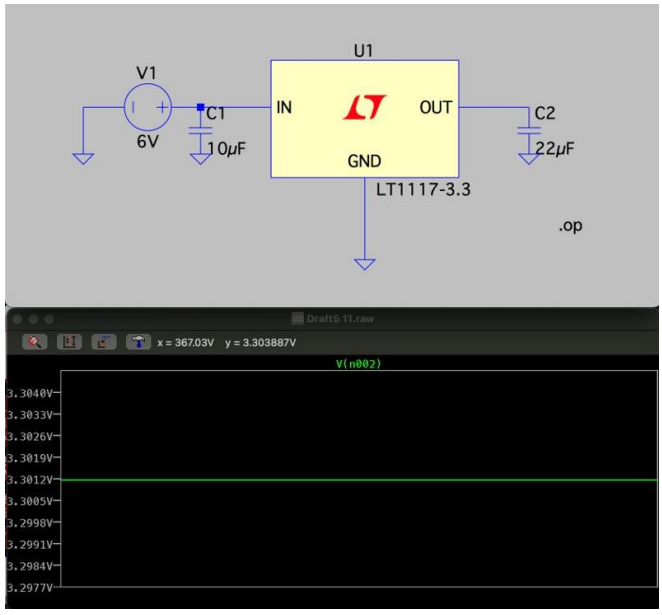

Figure 12: LT1117 LTSpice Simulation

When we received our physical PCB, the values we detected on all tracks designated to receive 3.3 V using the multimeter were within 0.5% of 3.3 V, even despite the allowable error in our requirement being so miniscule. When designing the PCB we made sure all power tracks

were thick and did not have any acute angles as per the PCB checklist guidelines provided by the ECE 445 course staff.

3.1.2. Requirement 1b

LT1117-3.3 voltage regulator supports at least 75 mA of current. This was easily confirmed as the peripherals connected to the voltage regulator on the PCB had a max current draw of about 43 mA, which we confirmed with a multimeter.

3.1.3. Requirement 1c

LT1117-3.3 voltage regulator does not exceed 150° C. Prior to testing the temperature with an infrared thermometer, we performed thermal analysis as a part of our tolerance verification in our design document. This calculation is provided below.

$$T_{junction} = T_{ambient} + P_{dissip} \times R_{\Theta} = 30℃ + ((V_{in} - V_{out}) \times I_{load}) \times 15℃ = 30℃ + ((6\ V - 3.3\ V) \times 43\ mA) \times 15℃ = 31.741℃$$

The infrared thermometer verified this analysis with a reading of about 29℃, most likely because the calculations were done with max current values, and the requirement was verified under no load conditions. Additionally, if the regulator was truly within 150℃ it would be extremely hot to the touch. The $T_{ambient}$ and R_{Θ} were taken from the LT1117 datasheet [6], while the I_{load} value was taken from the current draw analysis done in Section 2.1.

3.2. Control Subsystem

Our control system maintains the internal logic for our device, keeping track of the current pot count and color denominations, as well as sending control signals to the motors and LCD. This section will cover the requirements for this system to be complete and error-free.

3.2.1. Requirement 2a

Microcontroller is able to analyze data that is received from the TCS3200 color sensor. This requirement is quite simple, and frankly hinges on the proper function of the other subsystems. The focus of this requirement is that the components follow this particular order of movement: Analyze the Color → Rotate Base Motor → Actuator Retraction → Actuator Contraction. This requirement was fulfilled using our Nucleo-64 board, and was partially fulfilled with our PCB during our live demonstration. The requirement was only partially verified because we did not have an extra servo motor to demonstrate. However, since all other components functioned as intended with our PCB, and the same code was flashed from the development board, we can be confident that the motor would have worked had we had an extra one.

3.2.2. Requirement 2b

Microcontroller is able to perform the logic required to count the pot, and organize the colors as expected. Since our project conveniently includes an LCD, we are able to use it for verification purposes. Pre-counted stacks of varying colored chips were placed and it was verified the LCD read the correct value after each chip reading, as well as the correct total sum after all chips were counted.

3.2.3. Requirement 2c

Ensure that we are receiving a 3.3 +/- 0.1 V supply into the MCU. This requirement was verified as soon as we finished soldering our PCB. The multimeter reading on the V_{dd} pin of the MCU read a voltage value of exactly 3.3 V. We were able to test it while using the color sensor and LCD, which are the only components whose current draw would affect the stability of the 3.3 V, as they also get 3.3 V. The multimeter reading held steady at 3.3 V.

3.3. Sensing Subsystem

Our sensing subsystem consists of a TCS3200 color sensor. In this section, we will discuss our requirements that we have in place for our color sensor in order to ensure full functionality.

3.3.1. Requirement 3a

The microcontroller receives the correct RGB value corresponding to the chip that is inserted, based on information relayed from the TCS3200 sensor. We first tested the color sensor by analyzing the frequency values that it gave for each color chip and noting the differences with these values for each chip. These tolerance ranges were tabulated and incorporated into our program as shown in Figure 9. This tolerance range allowed us to have a 100% efficacy during our live demo.

3.3.2. Requirement 3b

The TCS3200 sensor receives between 3.3 +/- 0.5% V from the power subsystem. We verified this requirement by taking a multimeter and measuring 3.3 V at the V_{dd} pin of the sensor.

3.4. Motor Subsystem

This section will cover the verifications for our motor subsystem. Having a stable power supply for the motors would be the most important requirement for the subsystem to be fully functional as the PWM control signal is fully taken care of by our MCU.

3.4.1. Requirement 4

Both the motor and the linear actuator receive 6 +/- 0.5% Volts from the power subsystem. This was verified with a multimeter on our PCB. We ensured in our design that the tracks for these connections were thick enough to handle the higher current demand.

3.5. User Interface Subsystem

This section will cover the requirements for our user interface subsystem. This subsystem is responsible for the human-machine interactions that drive our device, so it is important we establish a good baseline for the system.

3.5.1. Requirement 5a

The remote microcontroller must be able to detect the press of the ‘reset’ button in at most a second of the press or less. Once we flashed our program onto our PCB, we were able to officially test this requirement with the PCB. The LCD reflected the pot-count reset instantaneously.

3.5.2. Requirement 5b

The microcontroller must be able to detect the status of the ‘chop’ button in at most a second of the press. This was verified with visual observation on the LCD. Multiple split situations were also verified.

3.5.3. Requirement 5c

The pot count on the LCD should read ‘0’ upon the press, hold, and subsequent release of the ‘reset’ button in a second or less. As mentioned in 3.5.2, this requirement was fulfilled following flashing the PCB with our code. The LCD reflected a count of “$0.00”, as soon as the reset button was released.

3.5.4. Requirement 5d

The LCD should display the number of ways the pot is being chopped, and the respective color denominations. We were able to get an instantaneous update on the LCD. With successive presses of the split button, the new monetary value of the divisions followed by the respective color denominations are shown on the LCD.

Figure 13: LCD with Appropriate Denominations Following Split

3.5.5. Requirement 5e

The LCD appends the pot count, according to the monetary value associated with the chip that is inserted. This requirement was visually verified during our live demonstration with our LCD. During the demonstration we took note of the value of each chip being sorted and we followed along as the value of the pot count displayed on the LCD would update within 5 seconds of the chip being detected.

4. Costs and Schedule

4.1. Parts

The construction of our device included a number of raw parts sourced from third-party manufacturers such as Digikey. Additionally, we sourced a raw PCB from PCBWay and hand soldered our SMD components to reduce overall cost. As a side note, some miscellaneous parts such as connectors were readily available in the Senior Design Laboratory itself, and are thus omitted from the overall costs of the device.

Description	Manufacturer	Quantity	Cost
LT1117CST-3.3	Linear Technology	1x	$6.30
HS-318 Servo	HiTec	1x	$11.99
STM32F103C8T6	STMicroelectronics	1x	$6.42
TCS3200	DFRobot	1x	$7.90
LCD	Newhaven Display	1x	$13.00
10u Cap	Kemet	1x	$0.24
22u Cap	Cal-Chip Electronics	1x	$0.40
6V Battery Pack	Tenergy Corporation	1x	$11.50
1u Cap	Kemet	2x	$0.20
NUCLEO-F103RB	STMicroelectronics	1x	$10.77
PCB	PCBWay	10x	$21.69
Total Cost			**$90.41**

4.2. Labor

To obtain the total cost of the project, labor fees for both group members and the machine shop will be taken into consideration.

Team Labor Cost = Pay Rate × Hours/Session × # of Sessions × # of Team Members

$$= \$40/hr \times 2.5\ hrs/session \times 60\ sessions \times 3 \ = \ \$18,000$$

Machine Shop Labor Cost = Pay Rate × Hours = $\$56/hr \times 40\ hrs \ = \ \$2240$

Total Cost of Project = $\$18,000 \ + \ \$2240 \ + \ \$90.41 \ = \ \$20330.41$

4.3. Schedule

Week	Item	Individual
8/21	Brainstorming	Team
8/28	Getting RFA Approval	Team
9/4	Getting RFA Approval	Team
9/11	Proposal, Team Contract	Team
9/18	Ordered Parts	Anish, Marvin
9/25	Completed CAD rendering of device	Justin
10/2	Completed PCB schematic, and layout	Team
10/9	Began initial testing on Nucleo board	Justin
10/16	Machine shop discussion about device logistics	Team
10/23	Completed programming internal logic, began integration onto dev board	Justin
10/30	Complete system integration on dev board, fully functioning subsystems	Justin
11/6	Got completed device from machine shop, completed wiring and packaging	Justin
11/13	Got PCB fully functioning	Team

5. Conclusion

5.1. Accomplishments and Uncertainties

We are proud to state that we have managed to achieve full functionality for our design, first with the Nucleo-64 development board, and next with our PCB, although it was not integrated into our design. Although some parts of our project did not completely function as intended it did not hinder us from meeting all our subsystem requirements and high-level requirements. The specifics on the device's minor shortcomings are detailed in Section 5.2 and 5.3. Otherwise, we were able to achieve the full functionality of our project as presented in the video submission and in our live demo. Despite the success, our final device had a few minor shortcomings. Other than our PCB not being integrated into our design our device had a little trouble with its rotation during our live demonstration. This can be attributed to our device being configured with the Nucleo-64 development board. As referred to in Appendix A, one of our requirements was that our linear actuator and rotational servo motor receive 6 ± 0.5% volts. Unfortunately, the Nucleo-64 development board only outputs 5 volts. This paired with the excessive weight of the rotational base prevented the motor from rotating to the furthest bin consistently. According to the datasheet [3], the motor will function as intended when supplied with power in the ranges of 4-6 V, however the motor only supports a torque of 38 oz-in at 6 V. Moreover, although the rotational servo has a maximum current draw of 190 mA and the Nucleo-64 development can supply up to 900 mA, the weight of our design can cause a spike in the rotational servo's current draw. From a simple stall torque analysis, we can be confident the lack of power supplied to the motor rated for 6 V is the cause behind this mishap. We were able to fix this issue by adjusting our code such that the motor rotates more to reach the furthest bin, but continuous usage of the motor this way could strain the motor long term. Since we have verified our PCB is powered with 6 V, we are sure that if we integrated it into our design the base would rotate as intended.

5.2. Ethical Considerations

Ethically, as a project that relates to money and the distribution of monetary equivalent chips, it is very important that we maintain an accurate count of chip value. Any error in the logic and sorting of the chips could result in an unfair financial loss to a player, which can compromise the entire game. The premise of our solution is to eliminate intentional and unintentional errors in home poker games, while increasing the efficiency of the game itself which adheres to Section I.1 of the IEEE code of ethics: "to hold paramount the safety, health, and welfare of the public"[4]. Since, our solution also attempts to eliminate manipulating pots, as described by our fourth high level requirement, it also supports Section I.4 of the IEEE code of ethics being "to avoid unlawful conduct in professional activities".

From a safety standpoint, any mechanism that uses motors and electrical components presents a safety hazard. The rotation servo is beneath our device, while the development board and LCD are in their own enclosures. In addition to this, our design will feature insulation

around any wires and loose electrical components to prevent any harmful contact. Not to mention, the device will reside in the middle of a poker table typically and thus should not catch any hair or jewelry.

It is also important to abide by the strict IEEE and ACM guidelines against plagiarism [4]. Although there are a number of chip sorting mechanisms available on the market today, none of them are directly targeted for home games. This is reflected in the cost of the device. As our product features proprietary hardware and software - targeting a brand new demographic - we can safely avoid any plagiarism.

5.3. Future Work

As previously mentioned, and as noted in our live demo and presentation, we were only able to get our PCB to work late at night prior to our demonstration. Other than, integrating our PCB into the design we have a couple of considerations for extending our design which we would have implemented, time-permitting. Firstly, we would like to improve the packaging, noise, and speed of the design. Although, majority of these concerns are mechanical issues, our project is a consumer item and making these changes would drastically improve user experience. Our project ended up being extremely large and quite loud, which can be somewhat of a hindrance to players during what should be a fun and social poker game. Additionally, we thought it would be nice if we accommodated another part of poker gameplay, being side-pots and all-ins. This situation involves situations when multiple players commit the entirety of their remaining chips to a hand and have varying balances in each of their stacks. In order to support this, our device would have to be designated for a fixed number of players with sensors corresponding to each player, and a larger LCD mimicking a scoreboard of sorts. This extension is not much more complex but requires more work and leaves more room for bugs that we would not have been able to solve in the original timetable. With these additions our proof of concept can take the leap to a legitimate consumer device.

References

[1] Datasheet - stm32f103x8 stm32f103xb - stmicroelectronics, https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (accessed Dec. 7, 2023).

[2] Henrik, "Stall torque vs motor voltage," PCBMotor, https://pcbmotor.com/speed-and-torque-measurement-o60mm/stall-torque-vs-motor-voltage/ (accessed Dec. 6, 2023).

[3] "HS-318 servo-stock rotation," ServoCity, https://www.servocity.com/hs-318-servo/ (accessed Dec. 6, 2023).

[4] "IEEE code of Ethics," IEEE, https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Sep. 14, 2023).

[5] NHD-0216HZ-fsw-FBW-33V3C - newhaven display, https://newhavendisplay.com/content/specs/NHD-0216HZ-FSW-FBW-33V3C.pdf (accessed Dec. 7, 2023).

[6] Positive regulators adjustable - analog devices, https://www.analog.com/media/en/technical-documentation/data-sheets/1117fd.pdf (accessed Dec. 7, 2023).

[7] R. Fee, "6 reasons why live poker is easier than online poker," Upswing Poker, https://upswingpoker.com/live-poker-vs-online-poker-easier/ (accessed Sep. 14, 2023).

[8] "Salary averages," *Electrical & Computer Engineering | UIUC*. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages. [Accessed: 23-Feb-2023].

[9] "TCS3200 color sensor - programmable color light-to-frequency converter," ams, https://ams.com/tcs3200 (accessed Dec. 6, 2023).

[10] Zion Market Research, "Trading card game market size, share and demand 2030," Zion Market Research, https://www.zionmarketresearch.com/report/trading-card-game-market (accessed Sep. 14, 2023).

Appendix A: Requirement and Verification Table

Requirement	Verification
1a. LT1117 voltage regulator provides 3.3 +/- 0.5% V Output.	1. Use a multimeter and measure the voltage at the output nodes of the voltage regulator to verify that it is supplying within 0.5 % of 3.3 V. 2. Use the multimeter to measure the voltage at inputs to verify that the voltage supply is being supplied correctly as well as to make sure that we have a stable connection between the Regulator and the devices. 3. Tabulate all of the measured values to ensure that the voltage output is accounted for as well as each input to the various devices.
1b. LT1117 voltage regulator supports at least 75 mA of current.	1. Use a multimeter at the output nodes to measure the output current at the voltage regulator.
1c. LT1117 voltage regulator stays under its maximum junction temperature of 150° C.	1. Take an infrared thermometer and measure the surface temperature of the voltage regulator and ensure that the temperature is under 150 ° C.
2a. Microcontroller is able to analyze data that is received from the TCS3200 Color Sensor. MCU then uses this data to rotate the base motor and push out the chips once we are directed to the correct bin.	1. Once the MCU knows the color on the chip that we are dispensing, we can then confirm the operation of the motor and linear actuator visually by verifying that first, the base motor rotates the actuator in the correct direction and after that, the actuator motor should dispense the correctly identified chip into the bin.

2b. Microcontroller is able to perform the logic required to count the pot, and organize the colors as expected.	1. We can verify accurate calculations and communication with the LCD by manually counting the value of chips in an inserted stack and seeing if that number is relayed to the display after sorting. We can then test the divisions the same way, do the calculations by hand and see if the LCD matches that value after hitting the split pot button. 2. Verifying that the colors are assigned to the correct bins can be verified visually.
2c. Ensure that we are receiving a 3.3 +/- 0.1 V supply into the MCU.	1. Use a voltmeter at the Vdd input pin to ensure that we are getting a 3.3 V supply into the MCU. Perform the measurement while sorting and dispensing a chip stack to ensure that we have a stable voltage supply throughout the full operation.
3a. The microcontroller receives the correct RGB value corresponding to the chip that is inserted, based on information relayed from the TCS3200 sensor.	1. Insert the maximum amount of chips the contraption supports , of varying colors. 2. Using serial debugging, record the values received by the microcontroller that corresponds to the TCS3200, ensuring that each value is closest to that of the respective color. E.g. if the RGB value for red is 320, the read value is 300, and all other colors are further away from 300 it is acceptable. 3. Record 2 trials detailed by steps 1 and 2 and record the color of the chips, what their expected RGB values were, and what they actually were.
3b. The TCS3200 sensor receives between 3.3 +/ 0.5% V from the power subsystem.	1. Insert a singular chip into the contraption. 2. Power only the sensor and use a voltmeter to measure the voltage supply to the sensor. 3. Repeat steps 1 and 2 for all four colors, and record data in a table.

4. Both the motor and the linear actuator receive 6 +/- 0.5% Volts from the power subsystem.	1. Insert three chips into the contraption. 2. Apply a voltmeter to both motor connections and record the values in a table for all the chip ejections.
5a. The remote microcontroller must be able to detect the press of the ‘reset’ button in at most a second of the press or less.	1. Using serial debugging, record the value received by the microcontroller when the ‘reset’ button has not been pressed. 2. Next, press and hold the ‘reset’ button. Using serial debugging, record the value received by the microcontroller, and ensure it is different from the value when unpressed. This should occur in at most a second, a stopwatch can be utilized to ensure this. 3. Finally, release the ‘reset’ button. Record the value received by the microcontroller using serial debugging. Ensure that this value is the same as the original unpressed value. 4. Organize all recorded values in a table. *Note: Refer to the sensing subsystem section for information regarding serial debugging.*

5b. The remote microcontroller must be able to detect the status of the ‘chop’ button in at most a second of the press.	1. Using serial debugging, record the value received by the microcontroller when the ‘chop’ button has not been pressed. 2. Next, press and hold the ‘chop’ button. Using serial debugging, record the value received by the microcontroller, and ensure it is the same as the value when unpressed (or previous value for +1 iterations). 3. Next, release the ‘chop’ button. Using serial debugging, record the value received by the microcontroller, and ensure it has changed from the unpressed default value.This should occur in at most a second, a stopwatch can be utilized to ensure this. 4. Continue performing steps 2 and 3 for three more iterations, ensuring that the value received by the microcontroller changes only upon the release of the ‘chop’ button using serial debugging. On the second of these two iterations, the value received by the microcontroller should match that of the default unpressed value of the ‘chop’ button.
5c. The pot count on the LCD should read ‘0’ upon the press, hold, and subsequent release of the ‘reset’ button in a second or less.	1. Put any number of chips through the contraption’s tunnel, such that the LCD does not display a ‘0’ pot count value. Next, press and hold the ‘reset’ button. The value on the LCD should read ‘0’, and should continue to read ‘0’ after the ‘reset’ button is released. The LCD should update to ‘0’ pot count in a second or less of pressing the reset button, a stopwatch can be used to check this.

5d. The LCD should display the number of ways the pot is being chopped, and the respective color denominations. If the pot cannot be evenly chopped it will display the color denominations for the greater split, or the greater two splits.	1. Put a number of chips through the contraption's tunnel that cannot be divided by two or three evenly using the available chip values. 2. Press and hold the 'chop' button. Ensure that the LCD shows a 2 way chop, the color denominations for the greater half of the split, in at most 2 seconds of the button press. 3. Release the 'chop' button, ensuring that the LCD shows the same information as the previous step. 4. Perform steps 2 and 3 two more times, ensuring that the LCD updates to reflect that of a 3 way chop on the first iteration. On the second iteration, the display should show the same information as it did prior to the first iteration of step 2- just the pot count and the color denominations for 1 winner. 5. Perform 1 final iteration of steps 2 and 3 such that the LCD shows the information for a 2-way chop once again. 6. Press and hold the 'reset' button, and ensure that the LCD only displays the pot count of '0' within at most 2 seconds of releasing the button.
5e. The LCD appends the pot count, according to the monetary value associated with the chip that is inserted.	1. Insert *x* amount of chips the contraption supports, of varying colors. 2. Ensure that this display shows the right monetary value corresponding to each chip that gets organized, and that the final sum is the correct value that was inserted at the start. The correct pot count should be displayed within 2 seconds of the linear actuator contracting after the chip is dispensed to its correct bin.