SlapBot: The Automated Slapjack Robot

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Samantha Verdi

Spring, 2025

Technical Team Members: Aimee Kang, Alex Beck, Michael Sekyi

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Todd A. DeLong, Department of Electrical and Computer Engineering

ECE Capstone Project Final Report: SlapBot
Team NoCaps
Aimee Kang, Alex Beck, Michael Sekyi, Samantha Verdi

Statement of Work

Aimee Kang worked on implementing OpenCV with the Raspberry Pi for the card detection process. Her work involved modifying, debugging, and testing the card identification code, configuring the Raspberry Pi to work with the encrypted UVA wifi network (eduroam), and assisting with miscellaneous tasks related to the physical design, such as building and gathering materials for the physical interface. She was the main point of contact for the functionality of the Raspberry Pi.

Alex Beck worked on integrating the Raspberry Pi to interact with the STM32 microcontroller. This work involved researching the best methods for communication between the two devices and testing both the Raspberry Pi and STM32 microcontroller to determine the best method. Additionally, he worked on the physical interface of the project by designing the enclosure for the components of the project.

Michael Sekyi worked on integrating the STM32 microcontroller to interact with the servo motor and robotic arm. This work involved creating the project environment, writing the algorithm, and configuring the system for real-time gameplay. This included determining the correct duty cycles for the slap and lift arm movements using PWM control signals, as well as integrating GPIO signaling with the Raspberry Pi to ensure seamless communication between the Raspberry Pi and STM32 microcontroller.

Samantha Verdi worked on the PCB and power design for the robotic arm, as well as physical modifications to the robotic arm. This work involved using Multisim and Ultiboard to create the PCB design, and reconstructing and augmenting the robotic arm with construction tools, respectively. Additionally, she worked with Michael to implement the robotic arm's functionality within the entire system of communication (STM and Raspberry Pi). She was the main point of contact for the functionality of the servo motor, and served as the lead carpenter for the group.

Table of Contents

Table of Figures

Abstract

Our capstone project, SlapBot, is an automated Slapjack playing robot. In Slapjack, players place cards face-up into a single pile and compete to be the first to slap a Jack card when it appears on the pile. The key to winning in Slapjack is quick reflexes, but for those who may be unable to react as quickly as others—whether it is due to old age or physical impairments—the competitive nature can be discouraging. The goal of this project is to extend the ability to play Slapjack to more people by creating an automated version to allow individuals to practice and improve their reflexes before playing with a group. SlapBot uses computer vision, a robotic arm, and STM microcontroller to control the motorized arm through a servo motor, and a Raspberry Pi with a camera module to identify the Jack card and prompt the arm through the STM32 microcontroller to slap the pile. Through OpenCV and the Raspberry Pi camera, the Raspberry Pi identifies the appearance of a Jack card and sets a GPIO pin high. Once the logic-driven STM32 microcontroller detects the pin as high, it triggers the algorithm to move the robotic arm for the slap.

Background

Slapjack is a simple card game for all ages. To begin the game, a standard deck gets distributed equally among all players. Then, holding their cards face-down, each player takes turns placing a card face-up in the center, creating a pile. Whenever a Jack is played, the first player to slap the pile collects all the cards in the pile and adds them to their own deck. A player is considered out if they have no more cards left in their hand. The player with all of the cards at the end wins the game, but players who have lost all of their cards have the opportunity to slap back into the game. This occurs when a player who is out slaps a Jack before any active players. The game primarily acts as a test of reflexes, as the player who consistently recognizes the Jack the soonest is most likely to win [1]. This project is designed for individuals who play Slapjack, but its focus narrows on individuals who wish to participate in a traditional game environment, but cannot due to a lack of reflexes, such as the elderly or physically impaired. The primary purpose of our capstone project is to increase inclusivity for those who want to participate in fast-paced entertainment, without facing the anxiety and discouragement of being unable to maintain the traditional pace of the game.

This project was selected due to a strong enthusiasm for the game, seizing an opportunity to combine our excitement with our technical skills and knowledge acquired through previous coursework and internship experience. Inspired by a past capstone group who created a robotic foosball machine [2], our project seeks to build on that creativity by designing a system that automates Slapjack, as both use computer vision for game automation. Beyond the setting of capstone projects, others have utilized computer vision to develop similar projects. A group of three from Stanford University were able to design a computer program to recognize card faces and classify them into their respective classes. They achieved an accuracy of 82-89% on their initial test set, but with the addition of a custom convolutional neural network, they achieved an accuracy of 100% on the test set [3]. Likewise, another project was also able to create a program

to recognize all the members of a standard deck and implemented an additional program to play Blackjack [4]. What differentiates our project from previous work is the incorporation of Slapjack with the inclusion of a robotic arm, a feature distinct from the projects mentioned above.

Prior coursework, as well as professional experience, has been instrumental in providing the foundation needed to address technical and non-technical challenges that arose during the development of this project. The ECE Fundamentals series (ECE 2630, 2660, and 3750) was important for circuit design and debugging strategies. Intro to Embedded Systems (ECE 3430) was crucial for interacting with the Raspberry Pi and STM32 microcontroller, specifically in using pulse-width modulation to control the servo motor. Software Development Essentials (CS 3140) helped in writing and testing clean, efficient code within a group.

Project Description

Performance Objectives & Specifications

The primary goal of this project is to develop an automated robot capable of playing Slapjack, combining technical precision with interactive gameplay functionality. The project can be divided into four main components: the robotic arm, card detection, embedded systems, and PCB design.

1. **Robotic Arm:** The robotic arm is the main mechanism used to execute the slapping motion in the game of Slapjack. The initial part choice for the robotic arm was the FLEXMAN Robotic Arm Kit from Amazon [5], however, due to shipping delays which caused a late arrival time, the Pololu Robotic Arm Kit was utilized instead [6]. Smaller in size and pre-built, the arm was easily incorporated into the project. The robotic arm is equipped with two Feetech RC FS5103B servo motors [7]: one is used to control the tilt of the arm, while the other controls the tilt. To control the slapping motion, only the servo that controls the height is used. Using pulse-width modulation, the duty cycle of the servo can be adjusted to move the arm up and down, a process managed by the STM32 microcontroller. As contact may be made between the player and robotic arm, a glove-like cushion made with styrofoam is added to make the game safe.

2. **Card Detection:** To detect the cards, the Raspberry Pi, Raspberry Pi Camera Module 3 and OpenCV were deployed to accomplish this task. Specifically, EdjeElectronics' OpenCV Playing Card Detector GitHub repository was utilized and adapted to correctly identify all of the cards in a standard card deck [8]. This implementation leverages OpenCV's prewritten functions for tasks such as image thresholding and contour detection. To detect the playing cards, the image of the card from the video feed is converted to greyscale, blurred, and thresholded. To identify the card, the program isolates the rank and suit of the card and compares them to pre-identified, trained images of cards of different ranks and suits. The program approximates the corner points of the

card contour and transforms the contour into a 200 x 300 image. Using the new image, the program takes a snippet of the corner of the card to isolate the rank suit. That corner snippet is thresholded and split into halves, where the top half is the rank and the bottom half is the suit. These halves are contoured and compared to the trained images. All card detection programs were written on Geany, an open source text editor installed on the Raspberry Pi [9].

3. **Embedded Systems:** This project consists of two embedded systems: the Raspberry Pi Model 3 B+ [10] and the STM32G071RBT6 [11]. As stated above, the Raspberry Pi is used to handle the card recognition of the project, while the STM32 microcontroller is used to handle the servo motor, which controls the robotic arm. To enable communication between the two systems, pin signalling is utilized to detect whether a pin has been set high or low, triggering a corresponding action. When the Raspberry Pi detects a Jack face card, a GPIO pin on the Raspberry Pi is set high. The STM32 constantly reads the output of the pin, and if the pin is set high, it triggers the slapping routine written in the STM32CubeIDE [12]. Initially, UART was the chosen communication protocol for the embedded systems, but as no crucial data was needed to send back and forth, GPIO signaling was a more straightforward and efficient alternative.

4. **PCB Design:** The PCB (printed circuit board) is responsible for powering the active servo motor for the robotic arm. Designed using Multisim [13] and Ultiboard [14], the PCB consists of a 2.1 mm barrel jack and pin headers for power and ground. The barrel jack takes an input from a 5V 5A power supply used to power the servo motor which operates at 5V and 1A, while the header pins are used to connect power and ground wires from the servo motor. Initially, the plan for powering the system involved using a voltage regulator to convert an external battery's output to specific voltages required by each component. However, this approach was modified as now the Raspberry Pi will use its own 5.1V 2.5A power supply and the STM32 will draw power from a computer using a USB cable, simplifying the power management system of the project.

How It Works: Block Diagrams, Schematics, Board Layout

In order to turn on the SlapBot, the power supplies for the Raspberry Pi and servo motor should be plugged into a wall outlet. To turn on the STM32 microcontroller, connect the STM32 to a computer using a micro USB to USB cable. To view the Raspberry Pi's interface on another display, use RealVNC Viewer [15] to establish a remote connection either via an Ethernet cable or the IP address of the Raspberry Pi. With a remote connection established, navigate the desktop to find the file labeled OpenCV Card Detection and run the CardDetector.py file. Executing this program will open a new window, displaying the video stream from the Raspberry Pi's camera module. A high-level overview of the entire SlapBot system can be seen in Figure 1.

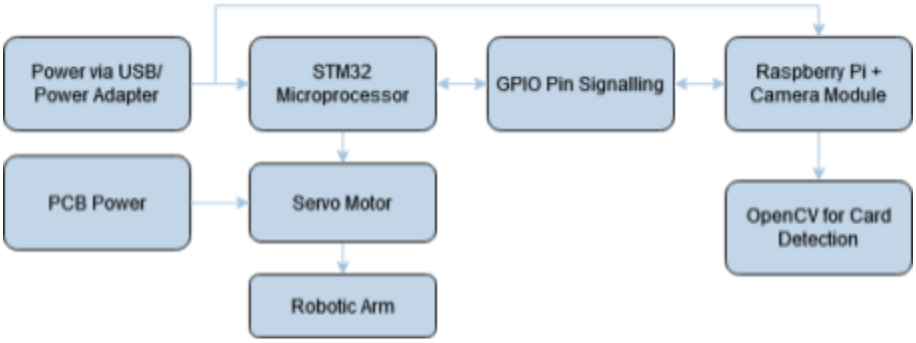

Figure 1. System Overview

In terms of gameplay, the dealer will distribute cards evenly among the players participating, where the dealer will also deal for the robotic arm. Players and the dealer will take turns placing cards in a pile with the face of the card showing towards the camera until a Jack card is placed. Once a Jack card is placed and detected by the Raspberry Pi, the Raspberry Pi will set a GPIO pin logic level high (3.3V). Using a female-to-female wire to connect the two GPIO pins of the Raspberry Pi and STM32 together, the STM32 will read the state of the pin. When the pin is set high, the STM will trigger the slapping algorithm. When the pin is set low, or idle, the system will stay idle.

During gameplay, if the player's hand slaps the table first before the robotic arm, that player wins the round and takes all the cards in the pile. If the robotic arm slaps the table before any player slaps, the dealer will take all the cards in the pile for the robotic arm. This process is repeated until one player takes all the cards in a standard deck, or the player loses all of the cards in their respective deck. A diagram of the control flow of the game can be seen in Figure 2.

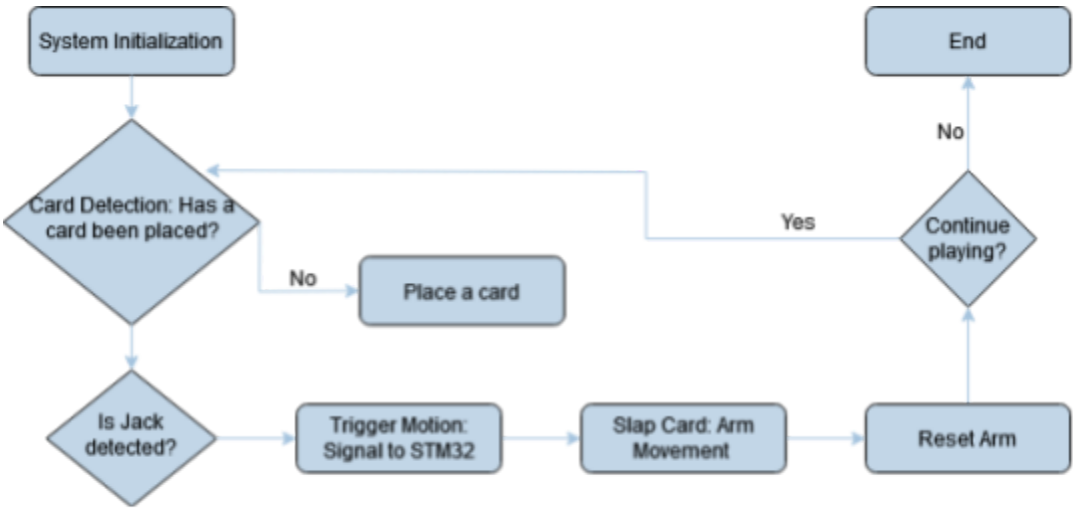

Figure 2. Game Control Flow

Technical Details

OpenCV (Open Source Computer Vision Library) is an open source framework used for computer vision applications [16]. As outlined above, the algorithm involves converting the card's image to grayscale, applying a gaussian blur, and performing thresholding to prepare the image for detection. The card identification method is achieved by isolating the rank and suit of the card, which are then matched against pre-identified training images. The program determines the card's contour corner points and transforms the contour into a flattened 200 x 300 image. Using the transformed image, a snippet of the card's corner is extracted to isolate the rank and suit and halved, where the top and bottom halves contain the rank and suit, respectively. Contours from these halves are compared to the training images for accurate identification.

The Raspberry Pi Model 3 B+ was the main system for image processing and card recognition for this project. It handled the computationally intensive logic required for real-time image processing, including capturing video feed from the camera and preprocessing the images (e.g., converting to grayscale, thresholding, and detecting contours). At the outset, the Raspberry Pi 2 Model B was the primary system for image processing [17]. However, due to the absence of onboard Wi-Fi, the decision was made to upgrade to a model of Raspberry Pi equipped with built-in wireless connectivity. Although the upgrade resulted in an increased thermal output, a heatsink case was incorporated to manage the higher temperatures effectively. Working in conjunction with the Raspberry Pi is the Camera Module 3 [18], providing a high-resolution video feed of the playing cards for processing. Initially, the HQ Camera was the primary imaging device of the project due to its superior image quality and adjustable lens compatibility [19]. However, the logistics of mounting the camera with the lens made this option less feasible as the lens is large in size and heavy. Regarding the Camera Module 3, the standard 200mm ribbon cable packaged with the camera module, used to connect to the Raspberry Pi, was found to be too short to provide an effective aerial view of the cards. To address this issue, a longer 400mm ribbon cable was integrated into the design, providing an effective aerial view of the cards and establishing an optimal height for accurate card recognition.

The STM32G071RBT6 was the main microcontroller used to handle the servo motor which controls the robotic arm. Utilizing the STM32CubeIDE as the main environment for C code development, the environment provided an integrated platform with debugging tools, peripheral configuration, and direct access to the STM32 HAL (hardware abstraction layer) and LL (lower layer) libraries, streamlining the development process. To control the servo motor, a 50 Hz PWM signal was required. The TIM2 timer on the STM32 was used to generate this signal using its internal 16 MHz clock as the source. Because TIM2 is a 16-bit timer, the direct division of 16 MHz (clock frequency) by 50 Hz (desired output) exceeds the maximum value the 16-bit timer can handle ($2^{16} - 1$). Therefore, a prescaler of 1599 is used to reduce the clock frequency to a value within the $2^{16} - 1$ timer limit. As a result, the counter period or Auto-Reload Register (ARR) value of 199 was configured to to satisfy the PWM equation and achieve a 50 Hz output. The calculation for this equation can be seen in Figure 3. The counter period and prescaler values are subtracted by one because the timer counts from 0 up to the specified value, as the hardware

registers in the STM32 employ a zero-based counting mechanism. Using the NI VirtualBench [20], duty cycles for the slap and release movements of the robotic arm were measured and determined to be 5% and 3% respectively. In the algorithm, the duty cycle percentages are multiplied by the counter period to determine the PWM value.

$$F_{PWM} = \frac{F_{CLK}}{(ARR + 1) * (Prescaler + 1)} = \frac{16\ MHz}{(199 + 1) * (1599 + 1)} = 50\ Hz$$

Figure 3. PWM Calculation

$$PWM_{SLAP} = ARR * Duty\ Cycle = 200 * 0.05 = 10$$

Figure 4. Pulse Width Value for Slap Motion

$$PWM_{LIFT} = ARR * Duty\ Cycle = 200 * 0.03 = 6$$

Figure 5. Pulse Width Value for Lift Motion

The program's functionality revolves around detecting a GPIO input signal and triggering a corresponding action from the servo motor. First the program initializes the GPIO and PWM peripherals. Outside of the main loop, the ServoSlap() function is used to adjust the PWM duty cycle of the servo motor, where the pulse width values for the slap and lift arm motions are passed as arguments in the function. Inside the while loop, the STM32 continuously reads the state of the GPIO input pin. If the pin is read high (3.3V), the program commands the servo to slap, calling the ServoSlap() function with 10 passed as the argument. After, the program pauses the arm at the slap state briefly using a delay, and finally calls the ServoSlap() function with 6 passed as the argument to lift and return the robotic arm to its natural state. Alternatively, if the GPIO pin is read low (0 V), the robotic arm remains idle.

As mentioned above, the PCB is responsible for powering the active servo motor for the robotic arm. The PCB, designed with Multisim and Ultiboard, is simple and consists of a 2.1 mm barrel jack and pin headers for power and ground. Two footprints for each component were placed on the board with the idea of powering both servo motors with a separate power supply, but the final product only required one. Initially, the plan for powering the system involved using a voltage regulator to convert an external battery's output to specific voltages required by each component, as the PCB was initially envisioned to power each specific component of the project. However, components like the Raspberry Pi and STM32 microcontroller will use their own power supplies to ensure they are receiving the correct operating voltage and amperage.

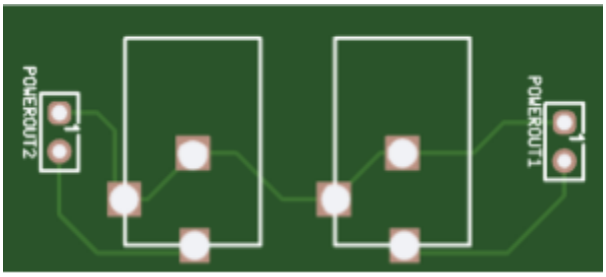

Figure 6. PCB Design Top

Figure 7. PCB Design Bottom

Test Plan

The test plan was structured to align with the project development process: the image recognition and algorithm was tested initially, with the servo motor and robotic arm tested after. Our plan was to test each subsystem of the project before system integration and final testing. The full test plan can be found numbered below:

1. Verify all cards can be correctly identified with the card recognition algorithm
2. Verify Jack card detection using GPIO signalling on the Raspberry Pi
3. Verify communication between Raspberry Pi and STM32 using GPIO pin signalling
4. Verify Jack card detection and Raspberry Pi and STM32 communication using the LED on the STM32
5. Tested the movement of the servo motor with the VirtualBench
6. Tested the movement of the servo using the STM32 algorithm
7. Tested full system integration
8. Conduct play testing for bugs, response time, etc.

In order to test the card recognition algorithm, we first uploaded pictures of the cards in the deck used to play the game to behave as training images. Once the images were uploaded, the card detector program was executed and each card was placed within the camera's field of view and correctly identified as the rank and suit of each card was displayed on the screen. Now that all cards are able to be correctly identified, testing Jack card detection was the next step. To test this functionality, Jack cards were placed in view of the camera and a GPIO pin on the Raspberry Pi would be set high if correctly identified. As mentioned earlier, UART was the chosen communication protocol for the embedded systems, but as no crucial data was needed to be sent back and forth, GPIO signaling offered a more straightforward and efficient approach. To test the functionality, a script was used to constantly set the GPIO pin on the Raspberry Pi high and low. On the STM32, the LED would only turn on when the pin on the Raspberry Pi was set high. This approach was also used to further test the card detection, where if a Jack card was detected, the GPIO pin on the Raspberry Pi was set high, triggering the LED on the STM32 to turn on.

With the embedded systems verified, testing the servo motor and robotic arm was the next priority. Using the VirtualBench, the duty cycles for the slap and lift arm movements were measured and confirmed. Using PWM and timer calculations mentioned above, the STM32 algorithm was developed to control the servo motor. With the movement of the robotic arm confirmed, full system integration testing began which included play testing for bugs and response time. The full system was tested multiple times during development, which helped ensure the success of the project. We successfully resolved several issues during testing, including ensuring the robotic arm's slapping response was limited to a single slap per trigger and implementing proper grounding to protect the servo motor from overloading, which occurred earlier in the testing process.

Physical Constraints

Design & Manufacturing Constraints

We had some time manufacturing constraints when it came to the part availability of our robot arm. Our original plan was to use the Flexman robotics kit, a DIY robot arm kit from Amazon coupled with servo motors. We were going to use these parts to create a custom arm for our project; however, the kit was set to arrive in early November, causing a noteworthy delay for our project. As a solution to this, our advisor provided us with a spare Pololu arm from his inventory, which was a perfect match for our project due to its compact size and straightforward interface. Additionally, we had some design difficulties with using interrupts in the STM code, since the results were inconsistent, so we used polling of the pin state in a while loop instead. Otherwise, our project went smoothly.

Tools

Throughout the development of this project, we used a variety of tools and software in order to produce deliverables and achieve the project's objectives. For the STM32 microcontroller, the STM32CubeIDE was the main environment for code development in the C programming language. As mentioned above, the environment provided an integrated platform with debugging tools, peripheral configuration, and direct access to the STM32 HAL (hardware abstraction layer) and LL (lower layer) libraries, streamlining the development process. As the IDE was used in a previous course, Introduction to Embedded Systems (ECE 3430), all group members were familiar with the tools, creating an efficient environment for code production. Alternatively, for the Raspberry Pi, Geany, an open-source text editor pre-installed on the Raspberry Pi, was used to adapt, compile, and run the card recognition algorithms. Additionally, RealVNC Viewer was essential in enabling remote access to the Raspberry Pi, allowing us to view and interact with the Raspberry Pi’s graphical user interface (GUI) during development and testing. National Instruments’ (NI) VirtualBench program was similarly essential during the testing and development process of the project. With the VirtualBench, we were able to confirm the duty cycle for the servo motor, as well as, behave as a power supply to test the functionality of the servo motor. In order to design the PCB, we utilized Multisim and Ultiboard to create our

simple design for the PCB. These tools facilitated the circuit simulation and design checks to ensure accuracy and reliability before physical production.

Cost Constraints

During development of this project, we did not face any cost constraints. To create the core of the project, we were able to use many materials that were free to us, including the Pololu Robotic Arm, provided to us by Professor DeLong, the power supply for the servo motor found in the NI Lab, and wood pieces for the physical design obtained from the FabLab at the architecture school. The majority of our budget went towards the embedded systems and the respective equipment for them, such as the heatsink for the Raspberry Pi and power cable for the STM32 microcontroller. Other purchases included spare parts and aesthetic equipment, such as the box that holds our parts in the final product.

Prototype Production

The main factor influencing prototype production is part availability. The availability of components played a significant role in determining which items were selected for building the project. Constraints such as shipping delays, limited stock, and compatibility with the system's design requirements required careful consideration during the selection process. Long lead times posed a significant challenge when ordering parts, as delayed deliveries disrupted the project's timeline. Similarly, items placed on backorder further impacted our scheduled progress, requiring adjustments to the development plan to accommodate these delays. For example, due to shipping and delivery delays, we had to change the initial robotic arm kit to an alternative robotic arm kit. This pivot caused us to change not only the robotic arm, but the motor and power delivery system we initially devised.

All parts used in the production of this project were readily available and the project became relatively easy to manufacture, but challenging to assemble, as we had to design a custom wooden enclosure to protect the embedded systems, wires, and the PCB. One improvement that could transition the current prototype into a production-ready version is the enclosure. This could require a deliberate 3D-printed design where all the major components of the project can all compactly fit together in a portable and aesthetic enclosure.

Societal Impact

Slapjack is a popular, fast-paced card game that brings players together in a quick reflex-based competition. However, for some individuals, such as the elderly, younger children, and those with cognitive/physical impairments, the need for fast reflexes to play the game can be discouraging, and can push new players away from the thrill of Slapjack. As most people are familiar with the game, it seemed important for our team to enable more people to play the game. The main goal of this project was to make Slapjack more accessible and less frustrating to play for those who are unable to slap quickly enough to stay in the game and have fun.

Public Health, Safety, & Welfare

One of the primary ethical considerations for this project is ensuring the safety of players interacting with the motorized arm. While there is always a moral obligation to create safe products to protect the dignity of the consumers, there is an even larger moral obligation to promulgate safety for consumers of this product, given the demographics of its target audience—primarily older individuals, young children, and those with cognitive or physical impairments. To address this concern, the player never comes in contact with the robotic arm, instead slapping the surface, rather than the pile of cards, eliminating the chance of coming in contact with the robotic arm. Furthermore, styrofoam is applied to the end of the robotic arm to further mitigate possible harm from contact with the robotic arm.

Global, Cultural, & Social Considerations

In designing our project, careful consideration was given to the various global, cultural, and social factors that influence its impact and accessibility. Our aim is to ensure that the project can be effectively applied across diverse contexts and aligns with the needs of a broad range of users. Our project hopes to bridge the gap between different social groups by making the game more accessible. By addressing barriers such as physical or cognitive impairments and designing with diverse demographics in mind, we hope to create a product that fosters greater participation and enjoyment for all.

Environmental & Economic Factors

In developing our project, we carefully considered the environmental and economic factors associated with its design and implementation. Our goal was to create a solution that is both cost-effective and environmentally conscious. Economically, the project is constrained by a $500 budget, which required us to select materials and components that balance affordability with functionality and reliability. Additionally, the use of readily available and reasonably priced parts ensures that the design is accessible for replication or further development by others. From an environmental perspective, the product's operational electricity consumption is minimal and unlikely to contribute significant energy use. The robot arm itself is primarily made of plastic and wood, materials that are recyclable. However, the arm paddings, which are essential to the safety of the game, are made of styrofoam, which cannot be easily recycled. To address this issue, during disposal, users could remove the styrofoam and bring it to a foam processing plant for, or keep the styrofoam. If this product were to be commercialized, the foam could be replaced with a biodegradable substitute. This would align with sustainable practices and reduce the product's environmental impact.

External Standards

In order to ensure that our capstone projects adheres to engineering best practices, specific standards will be followed to address safety, reliability, and environmental concerns. ISO 9241-210 ensured the ergonomic design of the user interface, making the system accessible for

individuals with cognitive or physical disabilities [21]. This standard emphasizes a human-centered approach, which involves understanding the needs and limitations of users and integrating them into the design process. For instance, foam padding on the arm was incorporated to minimize physical discomfort during possible interaction, providing a softer surface to reduce the possible harm of the slap. IPC-2221 guidelines were implemented in the PCB design to ensure safe, reliable, and effective circuit operation [22]. This included calculating trace widths and clearances to handle current and voltage safely, minimizing electromagnetic interference with ground planes and capacitors, and ensuring consistent power delivery during operation. Additionally, compliance with RoHS standards was achieved by using lead-free solder, non-toxic components, and recyclable materials like styrofoam and lightweight plastic [23]. These measures ensured that the project was environmentally friendly, safe, and reliable while meeting all functional objectives.

Intellectual Property Issues

This project has the potential to be patentable, as it integrates existing technologies in a unique and innovative way to create a novel design. By combining elements such as computer vision, motorized robotic arms, and custom algorithms for card detection, the project introduces a distinctive solution that offers both technical and functional advancements. The integration of all components makes for a unique user experience. However, there are ways that the project could become more patentable. For instance, implementing algorithms that predict or suggest strategies, or introducing adjustable difficulty settings. These enhancements would not only increase the functionality of the SlapBot, but also help differentiate it further from other gaming products on the market.

The first patent similar to the project we've devised is US Patent US20060001211A1 [24]. This patent relates to an "Automated Playing Card Identification System for Casino-Type Card Games", leveraging RFID technology to identify and monitor playing cards during casino games. The main difference between this patent and our capstone project is technology used to identify the cards. While we use computer vision to identify cards based on rank and suit, the patent explores RFID technology embedded in cards to automatically identify the rank and suit of a card. The patent makes an independent claim describing an automated system for identifying playing cards consisting of a deck of laminated playing cards, each containing a wirelessly pollable identifier positioned between the front and rear substrates, an antenna capable of transmitting a frequency to prompt the identifier to transmit its data, a reader to receive the transmitted identifier data, a game processor, and a graphics generator [24]. This claim is independent as it stands alone in defining the essential components of the system. Dependent claims of this patent include stating that the antenna is integrated into the playing surface of a gaming table, or that the graphics generator provides outputs for video fill and video key signals for television broadcast because they rely on the independent claim and narrow its scope by defining specifications.

The second patent similar to the project we've devised is US Patent US8657287B2 [25]. This patent describes a system that utilizes playing cards encoded with micro-dots containing rank and suit information. A specialized shoe reads these micro-dots as cards are drawn, allowing a game controller to monitor the game's progress and status. One of this patent's claim outlines the primary components of the system: playing cards embedded with micro-dots to encode the card's rank and suit, a shoe designed to read the micro-dots as the cards are dealt, and a game controller to process the data [25]. This claim is independent as it defines the core of the system. Dependent claims of this patent would include the details of the arrangement of the micro-dots on the cards and a description on how the game controller processes and displays information because it elaborates on aspects of the independent claim.

The third patent similar to the project we've devised is US Patent US11049359B2 [26]. This patent describes a "Chip Recognition System" to recognize chips on a gaming table using a combination of image analysis and artificial intelligence (AI) techniques. This patent claim describes the main components of the chip recognition system including a game recording apparatus that captures chip images using a camera, an image analysis apparatus that processes these images, multiple chip determination apparatuses, and a second apparatus that resolves discrepancies between the determinations and decides the correct count. This claim is independent as it outlines the core invention. On the other hand, dependent claims would include using trigonometry to determine the number of chips and specifying that the image analysis system can process partially obscured chips or chips hidden due to camera blind spots because they focus on narrowing the scope explained in the independent claims.

Timeline

Our proposed deadline for this project was November 27th, the day before our Thanksgiving break. By November 25th, we had successfully completed all core functionality, allowing us sufficient time to troubleshoot and address any bugs before transitioning to finalizing the interface, which was not completed until December 3rd. By completing the core functionality by our set deadline, we had time to troubleshoot and resolve any bugs before focusing on finalizing the interface. This structured approach allowed us to successfully deliver the completed project by our professional deadline of December 6th.

In our proposal, we planned on completing the research for the project by mid-September, having each component designed and constructed by the end of October, and have all components physically integrated into the project and tested by the end of November. Our team originally split responsibilities into two main subgroups—software and hardware. Aimee and Samantha were tasked with working on the software, implementing computer vision and image processing to recognize the Jack card. Michael and Alex were tasked with working on the hardware, designing the motorized arm and creating the custom PCB. The proposed Gantt chart of the project is shown below in Figure 8, illustrating the timeline for completing each component at various stages of the project. The chart is color-coded to clearly represent the tasks assigned to each group member.

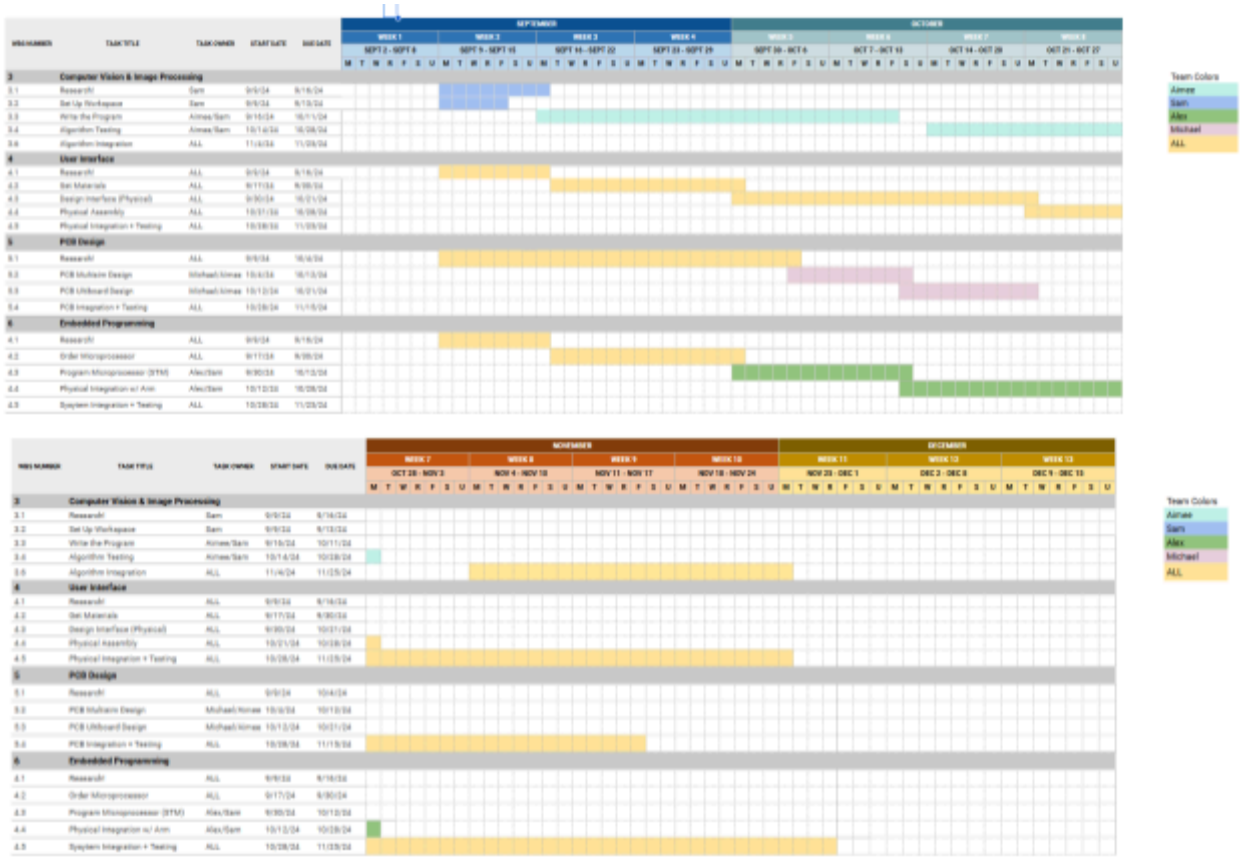

Figure 8. Proposed Gantt Chart

As progress was made on the project, our team roles changed based on our needs. Michael took on the role of general manager, as he was primarily responsible for the integration of all of the components, as well as overseeing each portion of the project, and acting as our purchasing manager. Samantha was responsible for the user interface and PCB design, and worked closely with Michael as she augmented the physical design. Alex was primarily responsible for the communication between the STM32 microcontroller and the Raspberry Pi, and worked with Michael and Samantha on the physical design. Aimee remained the main point of contact for the Raspberry Pi, implementing image processing for the card recognition program. The Gantt chart was modified to reflect these changes, and can be seen in Figure 9 below.

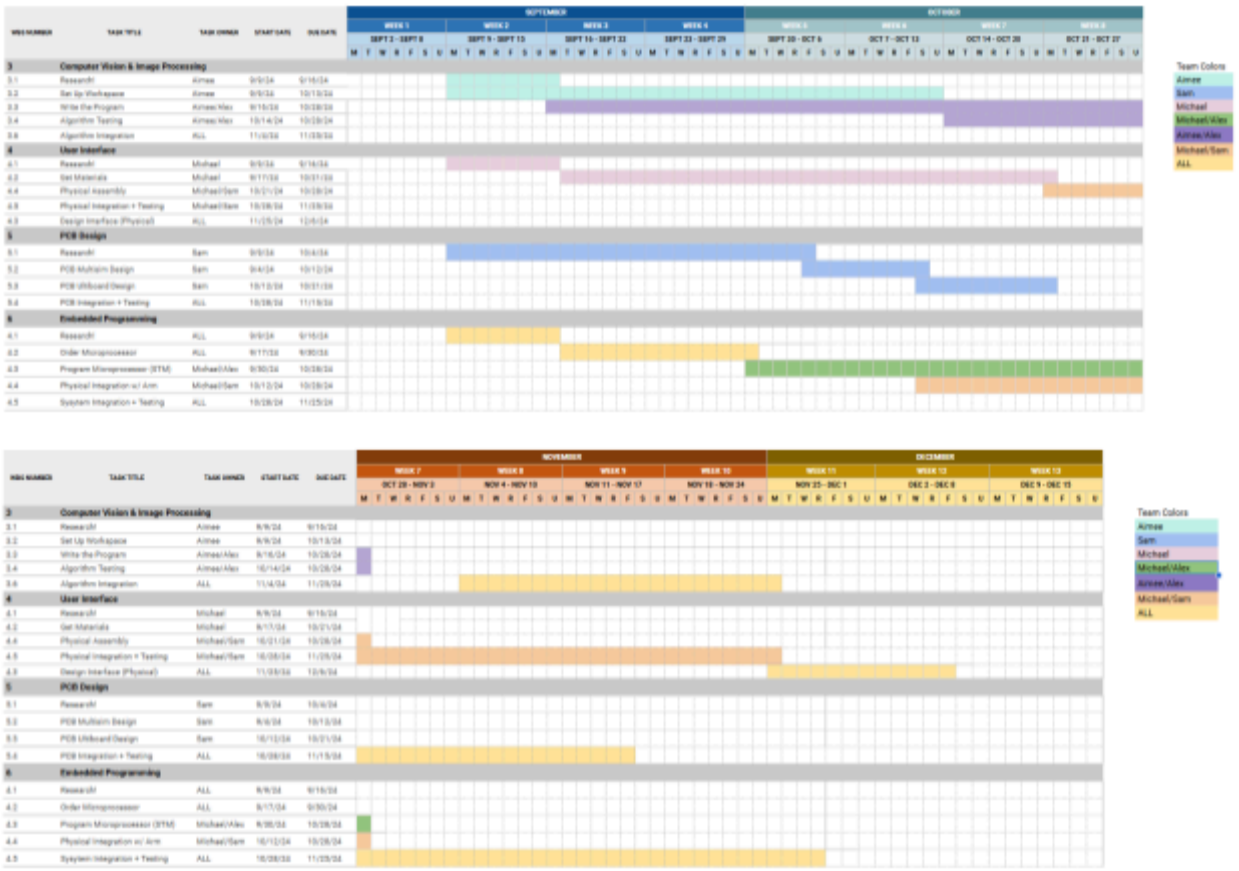

Figure 9. Final Gantt Chart

Schedule related setbacks we faced included technical troubles configuring the Raspberry Pi and camera and shipping delays with the FLEXMAN Robotic Arm Kit. While we were in the process of configuring the Raspberry Pi, we had to upgrade to a newer Raspberry Pi, the Model 3 B+, since the original, the Model 2 B, did not come equipped with network capabilities, unable to connect to UVA Wi-Fi. Additionally, there were troubles getting image recognition on the Raspberry Pi camera to work, resulting in experimentation with alternatives for the camera module, specifically the Camera Module 3 and HQ models. This resulted in a delay of setting up the workspace, pushing back the development of the Raspberry Pi program by a few weeks from our original timeline. Luckily, the configuration issues were resolved by mid-October and the algorithms for the Raspberry Pi were completed by the end of October. Afterwards, our group successfully tested and integrated these algorithms into the project, meeting the deadlines outlined in the original Gantt chart.

Regarding hardware, our group originally ordered the FLEXMAN Robotic Arm Kit on Amazon by mid-September to start designing the physical interface by the end of September. However, as a result of shipping delays, we were set to receive the arm mid-November, which would have been too late for the scope of our project. Fortunately, we received an alternative arm, the Robotic Arm Kit for Romi, in early October. Once we obtained the Romi, we were able

to move forward with development of the physical interface and integration of the robotic arm with the embedded systems. The STM GPIO communication with the robotic arm was completed by mid-October, and the combination of all systems was completely functional by mid-November, aligning within our original goal on the Gantt chart. We decided to focus on the aesthetic design of the project towards the end, finishing the entire project in early December.

Costs

In development of the project, the main cost-driven components include the embedded systems, the robotic arm, and peripherals like the camera. These components form the core of the system and significantly influence both the initial development expenses and the projected manufacturing costs for larger-scale production. In the Appendix below, detailed cost breakdowns of all components are provided, as well as the cost to produce 10,000 unit quantities. Automated equipment could reduce the cost it takes to assemble our product, but the addition of connecting wires introduces complexities that could make automation difficult.

Final Results

For our final results, we were successful in creating a fully functional prototype of the SlapBot. Once a Jack card is placed and detected by the Raspberry Pi, the Raspberry Pi will set a GPIO pin logic level high (3.3V). The STM32 will read the state of that pin and when the pin is set high, the STM will trigger the slapping algorithm that adjusts the PWM signal based on the pin signal. When the pin is set low, or idle, the system will stay idle. The camera is mounted on a wooden stand about a foot above the surface of the table, where it has enough distance to identify the cards without getting in the way of the player or the robot. The Raspberry Pi, STM32, and PCB are all enclosed in a compact wooden box, with an opening in the back for the wires connected to the components. The robotic arm is mounted on a wooden slab to keep it from tipping over, and the servo motor of the arm is connected to the STM through wires located inside the box. To ensure the safety of the user, the Romi arm has been padded with styrofoam, protecting the user from getting hurt and the SlapBot from potential damage. This makes it such that the SlapBot can realistically simulate Slapjack: as the player competes with the SlapBot, they can judge who wins by whose hand is closer to the card.

Once we had the system fully integrated, we were able to test it based on the grading criteria we had decided on in our proposal seen in Figure 10. Through a run of several trials, the camera was able to identify the Jack every single time it was on the screen, in the case that it was not overlapping with other cards. Per our rubric, we can confidently say the algorithm we use has at least a 75% success rate, since there was never a time when the card was misidentified, and it was a matter of placing the cards neatly for the camera to read. Every time the Jack was on the screen, the arm would slap, then remain down until the Jack was removed, or another card was placed on top. This fulfills our rubric criteria of having a “correctly working” algorithm for our robotic arm: it realistically models Slapjack by responding cleanly, as opposed to reacting erratically when a Jack is placed, such as slapping up and down (as it was in an earlier iteration

of the algorithm). To that note, for our final criteria, we were able to have the robotic arm fully built and functional, and there was never a time that the arm did not respond when a Jack was placed and identified. Overall, our final project met all of our criteria of achieving an A on the rubric.

Self-Assessment Rubric

Grade	Expectations
A	● Image recognition has at least a 75% success rate ● Arm is built, arm response is 100%
B	● Image recognition has at least 50% success rate ● Arm is built, arm response is 75%
C	● Image recognition has at least 10% success rate ● Arm is built, arm response is existent
D	● Image recognition is 0% ● Arm is built, arm response is nonexistent
F	● Image recognition is nonexistent ● Arm is not built

Figure 10. Self-Assessment Rubric

Engineering Insights

From our experiences working together as a team for this project, we each learned a great deal of new technical and soft skills. Based on what each team member accomplished on the technical side, we each gained knowledge in each of our specific areas of the project. Group members who worked with the Raspberry Pi and STM32 gained a deeper understanding of both devices by setting them up, programming in Python and C, and developing algorithms tailored to their functionalities. In addition, those who worked with the Raspberry Pi Camera Module gained experience in image processing and techniques in facilitating pattern detection from the Raspberry Pi Camera. Lastly, those who worked with the PCB gained technical expertise in development tools in designing custom circuits and the process of working with PCB production companies.

Throughout the course of this project, the group as a whole gained a comprehensive understanding of the engineering process from start to finish, acquiring skills in time and resource management, delegation, and teamwork. Through the research and development of the project, the team learned to react to unpredictable setbacks and adapt to major changes in the

project. The team also learned to adapt to each team member's strengths and weaknesses to give them the most potential in being efficient. Lastly, through the system integration of the project, the team learned to bring together what they each individually worked on and adapt to their individual styles of work.

If we had to give advice to future Capstone students, our recommendation would be to choose a topic you are passionate about. With every profession and especially for large projects such as the capstone, staying engaged with your goals and learning to enjoy the process of building and designing your project can make a significant difference. In addition, it's important to stay on schedule and enforce deadlines for getting major benchmarks for the project done. This also makes resolving logistical issues, such as shipping delays, and unexpected technical issues much easier. This is particularly important for capstone projects as these unpredictable issues are almost guaranteed to occur at a certain point due to the nature and pace of this course.

Future Work

This project has many possible avenues for future improvement. A potential improvement would be to expand SlapBot's capabilities to play Egyptian Rat Slap (ERS), a more complex variation of Slapjack that incorporates additional rules and a wider range of card combinations to slap [27]. Currently, the SlapBot algorithm recognizes the Jacks in the deck and the arm automatically slaps down after detection. It could be possible to enhance the algorithm to have the detection be more dynamic and include more conditions for when to slap. Implementing these new conditions would increase the complexity of the algorithm and make for a new game all together.

The SlapBot raises some broader questions about human-robot interaction. The interactive design could be a commercially available gaming companion or device found in casinos or toy stores. However, the design would have to be enhanced to consider pricing and scalability. The application also does not have to be limited to slapjack, but to other fast-paced card games or board games.

Future work should be mindful of motor capabilities and precision, as the fluctuating reliability of the motors was a massive hurdle during the semester. Consideration should also be given to how to standardize card recognition across different lighting and the time between card recognition and the slap should be readily adjustable. Durability and safety are also important for future considerations and increasing both factors is an important aspect to consider. All of these enhancements can make for a stronger design in the future.

References

[1] “Slapjack.” Accessed: Dec. 05, 2024. [Online]. Available: https://bicyclecards.com/how-to-play/slapjack/

[2] H. Burke, J. Long, A. Himley, C. Jenkins, and Z. Yahn, “Robotic Foosball Table,” University of Virginia. [Online]. Available: https://acrobat.adobe.com/id/urn:aaid:sc:US:b39909da-289f-42a7-bafc-1e4f0bd7c0b7

[3] M. Castillo, B. Goeing, and J. Westell, “Computer Vision for Card Games”, [Online]. Available: https://cs229.stanford.edu/proj2017/final-reports/5233806.pdf

[4] G. Hollinger and N. Ward, “Introducing Computers to Blackjack:”, [Online]. Available: https://www.cs.cmu.edu/afs/cs/academic/class/15494-s11/final-projects/2007/blackjack/IEEEHollinger.pdf

[5] “Amazon.com: FLEXMAN Robotic Arm Kit, 6DOF Robot Mechanical Arm Clamp Claw Kit, Programmable Aluminum Industrial Robot DOF Manipulator for Teaching DIY : Industrial & Scientific.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.amazon.com/Mechanical-Programmable-Aluminum-Industrial-Manipulator/dp/B0DC44PCF7

[6] “Pololu - User’s guide for the Robot Arm Kit for Romi.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.pololu.com/docs/0J76/all

[7] Shenzhen Feetech RC Model Co., Ltd, “FS5103B Servo Motor Product Specification.” Shenzhen Feetech RC Model Co., Ltd, Jul. 22, 2021. [Online]. Available: https://www.feetechrc.com/Data/feetechrc/upload/file/20211215/637751795943722136879177 8.pdf

[8] EdjeElectronics, *OpenCV-Playing-Card-Detector*. (2017). [Online]. Available: https://github.com/EdjeElectronics/OpenCV-Playing-Card-Detector/tree/master

[9] “Geany.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.geany.org/manual/current/index.html

[10] Raspberry Pi Ltd, “Raspberry Pi 3 Model B+ Product Brief,” Nov. 2023. [Online]. Available: https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf

[11] “STM32G071RB - Mainstream Arm Cortex-M0+ MCU with 128 Kbytes of Flash memory, 36 Kbytes RAM, 64 MHz CPU, 4x USART, timers, ADC, DAC, comm. I/F, 1.7-3.6V - STMicroelectronics.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32g071rb.html

[12] “STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.st.com/en/development-tools/stm32cubeide.html

[13] “What Is Multisim™ for Designers.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.ni.com/en/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim/multisim-designers.html
[14] “Ultiboard.” Accessed: Dec. 05, 2024. [Online]. Available: https://www.ni.com/en-us/shop/product/ultiboard.html
[15] “VNC Viewer by RealVNC®,” RealVNC®. Accessed: Dec. 05, 2024. [Online]. Available: https://www.realvnc.com/en/connect/download/viewer/
[16] OpenCV Developers, *OpenCV*. (2024). [Online]. Available: https://github.com/opencv/opencv
[17] Raspberry Pi Ltd, “Raspberry Pi 2, Model B Product Specifications.” [Online]. Available: https://cdn-shop.adafruit.com/pdfs/raspberrypi2modelb.pdf
[18] Raspberry Pi Ltd, “Raspberry Pi Camera Module 3 Product Brief,” Jun. 2024. [Online]. Available: https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf
[19] Raspberry Pi Ltd, “Raspberry Pi High Quality Camera Product Brief,” Jan. 2023. [Online]. Available: https://datasheets.raspberrypi.com/hq-camera/hq-camera-product-brief.pdf
[20] “NI VirtualBench,” https://www.ni.com. Accessed: Dec. 06, 2024. [Online]. Available: https://www.ni.com/docs
[21] “ISO 9241-210:2019(en), Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems.” Accessed: Dec. 06, 2024. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso:9241:-210:ed-2:v1:en
[22] IPC (Association Connecting Electronics Industries), “IPC-2221 Generic Standard on Printed Board Design.” [Online]. Available: https://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC_2221.pdf
[23] “RoHS Directive.” Accessed: Dec. 06, 2024. [Online]. Available: https://environment.ec.europa.eu/topics/waste-and-recycling/rohs-directive_en
[24] F. Lewis, K. Lewis, and L. Rogers, “Automated playing card identification system for casino-type card games,” US20060001211A1, Jan. 05, 2006 Accessed: Dec. 06, 2024. [Online]. Available: https://patents.google.com/patent/US20060001211A1/en
[25] V. Krishnamurty, D. Horvath, and S. Bodaly, “Intelligent table game system,” US8657287B2, Feb. 25, 2014 Accessed: Dec. 06, 2024. [Online]. Available: https://patents.google.com/patent/US8657287B2/en?oq=US8657287B2
[26] Y. Shigeta, “Chip recognition system,” US11049359B2, Jun. 29, 2021 Accessed: Dec. 06, 2024. [Online]. Available:

https://patents.google.com/patent/US11049359B2/en?q=(playing+card+recognition)&oq=playing+card+recognition
[27] “Egyptian Rat Screw.” Accessed: Dec. 06, 2024. [Online]. Available: https://bicyclecards.com/how-to-play/egyptian-rat-screw/

Appendix

Part	DigiKey Part Number	Price	Quantity
STM32 Microcontroller (NUCLEO-G071RB)	497-18337-ND	$11.04	1
Raspberry Pi Microcontroller	2648-SC0073-ND	$35	1
Raspberry Pi Camera Module 3	2648-SC1223-ND	$25	1
Robotic Arm Kit for Romi	2183-3550-ND	$89.95	1
PCB Board	N/A	$11.75	1
Raspberry Pi Power Supply	2648-SC0445-ND	$8	1

Total Cost: $180.74

Figure 11. Production Cost of One Unit

Part	DigiKey Part Number	Price	Quantity
STM32 Microcontroller (NUCLEO-G071RB)	497-18337-ND	$110, 400	10000
Raspberry Pi Microcontroller	2648-SC0073-ND	$350,000	10000
Raspberry Pi Camera Module 3	2648-SC1223-ND	$250,000	10000
Robotic Arm Kit for Romi	2183-3550-ND	$899,500	10000
PCB Board	N/A	$117,500	10000
Raspberry Pi Power Supply	2648-SC0445-ND	$80,000	10000

Total Cost: $1,807,400

Figure 12. Production Cost of 10K Units