RFID Poker Table

ECE 445 Final Report

Darren Liao (darrenl4)
Khuselbayar Bolor-Erdene (kb40)
Satyam Singh (satyams2)

TA: Eric Tang

Dec 8, 2025
Project No. 18

Abstract

This project tackles the challenge of making live Texas Hold’em home games easier to follow by automatically tracking the full game state on a physical table. It focuses on the engineering problem of reliably sensing cards, chips, and player actions in real time and fusing those signals into a consistent digital representation of the game that casual players can understand at a glance.

The resulting system is a smart poker table that embeds RFID readers under the surface to detect hole cards and community cards, uses an overhead camera and computer vision to estimate chip stacks and betting actions, and connects wirelessly to a central server and mobile clients so players can view the current state, bets, and odds without manual bookkeeping. Measured performance shows that the RFID subsystem, camera streaming pipeline, and chip-detection pipeline meet the required latency and accuracy targets. These results demonstrate that live poker gameplay can be digitized with high reliability and low latency, improving accessibility and reducing errors in casual home games.

Contents

1. Introduction

Poker is a widely played card game that blends probability, psychology, and strategy, and Texas Hold’em has become the dominant variant in both casinos and casual home settings. In informal games among friends, players often lack a professional dealer or dedicated tracking tools, so they rely on ad‑hoc agreement to manage cards, blinds, betting rounds, and side pots. This manual bookkeeping is error‑prone and cognitively demanding, especially for newer players who are still learning hand rankings and betting conventions. Misdeals, misread hands, and disagreements about the pot or action order slow games down and can create friction in what is meant to be a social activity [1], [2].

Online poker platforms address many of these issues by automatically enforcing rules, evaluating hands, and displaying a clear game state to every participant. Cards, chips, and turn order are tracked digitally, leaving players free to focus on decision‑making. However, these systems lose much of the social and tactile experience that makes live poker appealing, such as handling physical cards and chips or reading opponents’ body language. There is therefore an opportunity for systems that bring some of the structure and clarity of online poker into live home games without sacrificing the physical table experience.

The engineering problem addressed in this project is how to sense the physical elements of a Texas Hold’em game including cards, chips, and player actions, in real time and convert them into a consistent digital game state. This requires robust detection of multiple stacked RFID tags at each player position, low‑latency communication between sensors and a central controller, and reliable computer vision that can estimate chip stacks and betting actions under varying lighting conditions. The system must also present this information in a way that is accessible to casual players, who may not be familiar with technical interfaces but can benefit from clear visual cues about the current state of the hand.

To address these challenges, this project develops a smart poker table that combines embedded RFID readers, an overhead camera, a Wi‑Fi‑connected microcontroller, and a remote server with a mobile companion application. RFID antennas below the felt detect each player’s hole cards and the shared community cards; a camera provides a bird’s‑eye video feed for chip detection and action monitoring; and an ESP32 microcontroller aggregates sensor data and communicates with a game‑state server over standard networking protocols. The server fuses RFID and vision data, maintains the authoritative representation of the game, and exposes this state to player devices via a mobile interface that shows cards, pot information, and turn order tailored to each user’s permissions.

The objectives of the project are fourfold. First, the system must detect and track all hole cards and community cards in a standard Texas Hold’em hand with high reliability, even when two tagged cards are stacked at each player position. Second, it must update the companion application within approximately one second of any physical change on the table so the digital view feels responsive. Third, the computer‑vision pipeline must estimate chip stacks and pot size with sufficient accuracy to be useful in casual play. Fourth, the overall design should demonstrate that this architecture can serve as a foundation for future extensions, such as additional player seats, richer user interfaces, or more advanced strategy aids.

2. Design

2.1 Introduction

The RFID Poker Table is designed to automatically track a live Texas Hold'em heads-up game by sensing cards and chips on a physical surface and mirroring the game state in software. Figure 1 shows the high-level block diagram of the system, which is divided into four main subsystems: the power system, the sensor system, the communication system, and the remote system.

The power system converts 120 V AC from the wall into regulated 5 V and 3.3 V rails that supply the ESP32, RFID readers, and camera, ensuring stable operation up to at least 300 mA on each rail. The sensor system consists of three HF 13.56 MHz PN532 RFID readers mounted under the player and community card zones, along with an overhead camera that streams MJPEG video of the table for chip and action detection. The communication system is built around an ESP32 microcontroller and an I^2C multiplexer that poll the RFID readers, aggregate card IDs, and transmit updates over Wi-Fi to the remote server within a 1 s deadline. The remote system runs on a laptop and hosts the game-state manager, RFID card lookup, computer-vision pipeline, and REST API used by the mobile clients.

The final performance requirements for the project are: (1) detect and track all hole cards and community cards with at least 95% reliability during a standard hand; (2) propagate any card or chip change to the companion app within 1.0 s; (3) detect chip stacks and pot size with at least 90% accuracy under typical lighting; and (4) maintain at least 90% Wi-Fi uptime while polling up to three RFID readers and streaming video. During the semester, the block diagram evolved in two notable ways: the camera subsystem was migrated from an ESP32-CAM to a Raspberry Pi camera to avoid thermal throttling and low frame rates, and the number of RFID readers was reduced and routed through an I^2C multiplexer to simplify wiring while still supporting stacked-card detection at each position. These changes reflect the key performance factors for the system, RFID read reliability under stacking, end-to-end latency through the ESP32 and network, and robust video streaming for computer vision which are analyzed in more detail in the following subsections.

2.1.1 Block Diagram

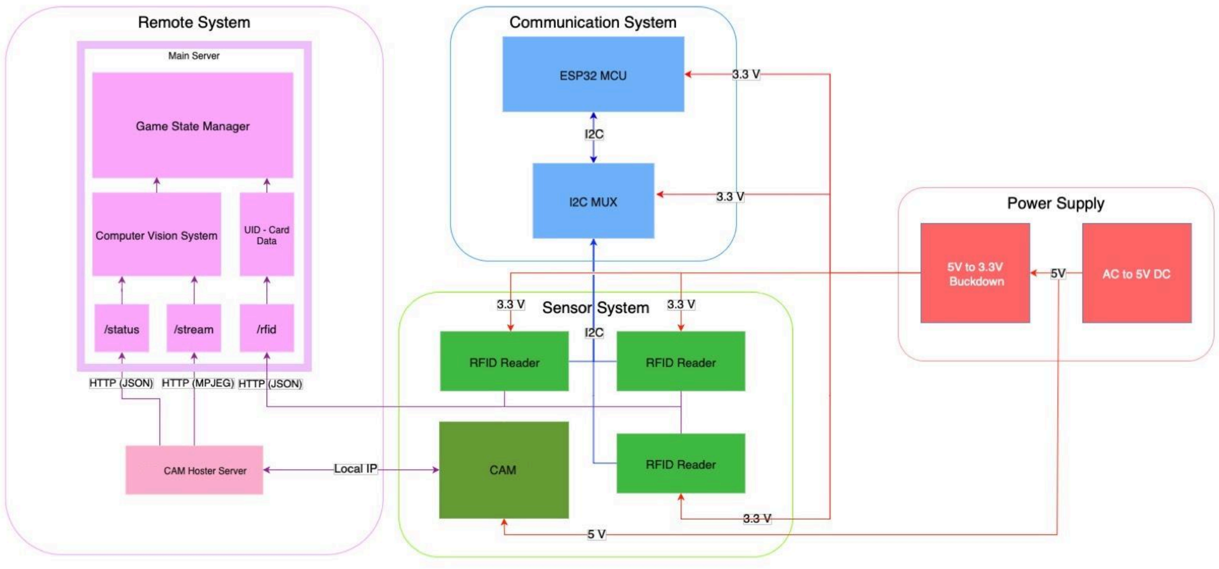


Figure 1. Block diagram of the RFID poker table

2.2 Sensor System

The sensor system is responsible for detecting all physical elements of the poker game. This includes the cards on the table and chips in betting zones, and then transmitting that raw data to the communication system for processing. It consists of two independent sensing modalities: high-frequency RFID readers that identify cards by their embedded tags, and an overhead camera that captures a bird's-eye video stream for computer vision based chip detection.

2.2.1 RFID Reader

The RFID subsystem uses three PN532 RFID modules operating at 13.56 MHz to detect playing cards embedded with passive HF RFID tags. Each module is positioned under a critical zone on the table: one under Player 1's hole cards, one under Player 2's hole cards, and one under the community card area. The PN532 was selected for its native support of ISO/IEC 14443 Type A tags, I²C communication interface, and ability to detect multiple tags in rapid succession through anti-collision protocols.

Each PN532 module connects to the ESP32 via I²C through a TCA9548A multiplexer, allowing all three readers to share the same SDA and SCL lines while being addressed independently on separate channels. The multiplexer resolves the address-collision problem that would otherwise occur with three identical PN532 modules on a single I²C bus. Firmware running on the ESP32 polls each channel in round-robin fashion with a 30 ms timeout per reader, enabling the system to scan all three zones approximately 11 times per second under ideal conditions.

The key challenge in RFID detection is stacked‑card reading. In Texas Hold'em, each player receives two hole cards dealt facedown in the same position, meaning two RFID tags must be reliably detected within a few centimeters of each other. Electromagnetic coupling between stacked tags can reduce read range and introduce errors, so the PN532's anti‑collision algorithm is critical [3]. Testing showed that with 30 ms per polling cycle and approximately 66 polling attempts within a 2‑second detection window, the probability of failing to detect both cards drops to 4%, meeting the 95% reliability requirement.

Standard poker cards were modified by affixing thin 13.56 MHz RFID stickers to the back of each card. Each sticker contains a passive HF RFID tag programmed with a unique identifier that maps to a specific card rank and suit in the server's lookup table. The tags are passive, requiring no battery, and draw power from the PN532's RF field when queried. This sticker-based approach allowed for rapid prototyping without requiring custom card manufacturing, though it does add slight thickness to each card.

2.1.2 Camera

For the Camera Sensor to be used in our system we needed a reliable, high availability, and quality streaming camera to reach our requirements of realtime capture, and broadcast to our server for computer vision processing. In our final iteration we settled on a Raspberry Pi Ribbon Camera since it was the most reliable compared to our other tested option such as the ESP-CAM. The ESP-CAM resulted in thermal throttling slowing down the CPU clock speed when ran more than 10 minutes which in turn resulted in our measured frame rate received on the server to dip down below 10 frames per second. Raspberry Pi Ribbon Camera, on the other hand, did not have this issue and delivered a steady frame stream broadcasted on its local IP address. The testing results are shown in section 3.

The communication method we used to broadcast was Motion JPEG (MJPEG)[4] which is a simple video encoding format where each frame of the video stream is sent as an individual JPEG image. Because every frame is independently compressed, MJPEG requires very little processing power to encode or decode compared to modern inter-frame codecs which is perfect when running on a computationally limited hardware. It’s commonly used in embedded systems, webcams, and low-latency video applications where consistent image quality and fast frame delivery matter more than bandwidth efficiency. We use MJPEG because it gives our system a reliable, low-latency video stream that’s easy to decode on devices like the ESP32 or a lightweight computer vision pipeline, ensuring smooth real-time chip detection without overloading the processor.

2.3 Remote System

2.3.1 Companion App

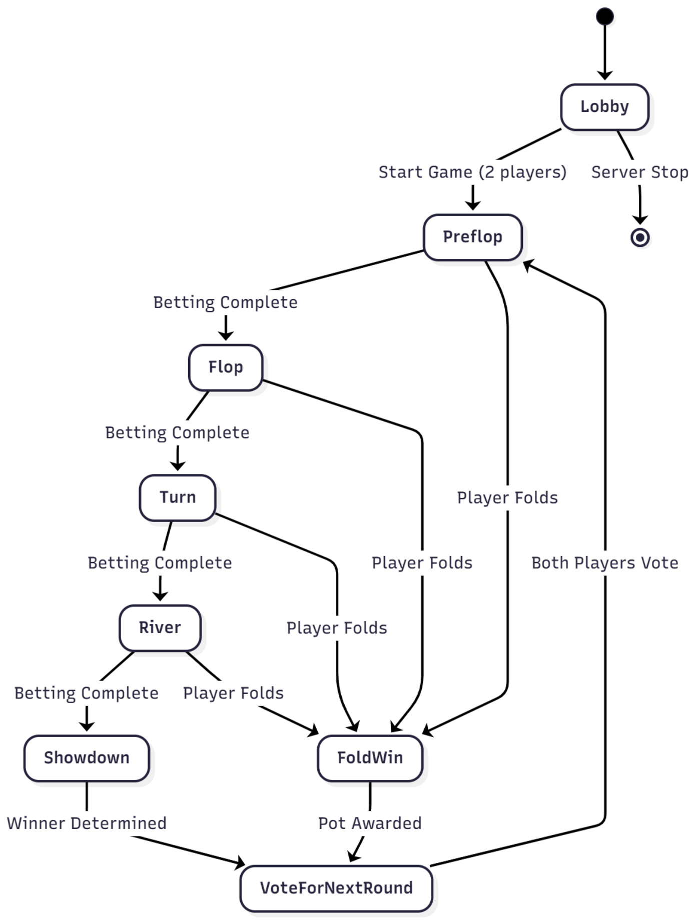


Figure 2. Game state logic represented as a finite state machine

The game server, hosted on a computer, is responsible for maintaining the overall game state, receiving updates from the microcontroller, and sending those updates to all connected players through their companion apps. It can be viewed as three main components: a front end that displays the game, a backend that manages game logic and state, and a networking backend that handles player connections and update requests. The purpose of this state machine is to replicate poker gameplay logic, control the flow of the game, and ensure all apps remain synchronized to the same game state within one second of any change, since the server serves as the single source of truth for all players.

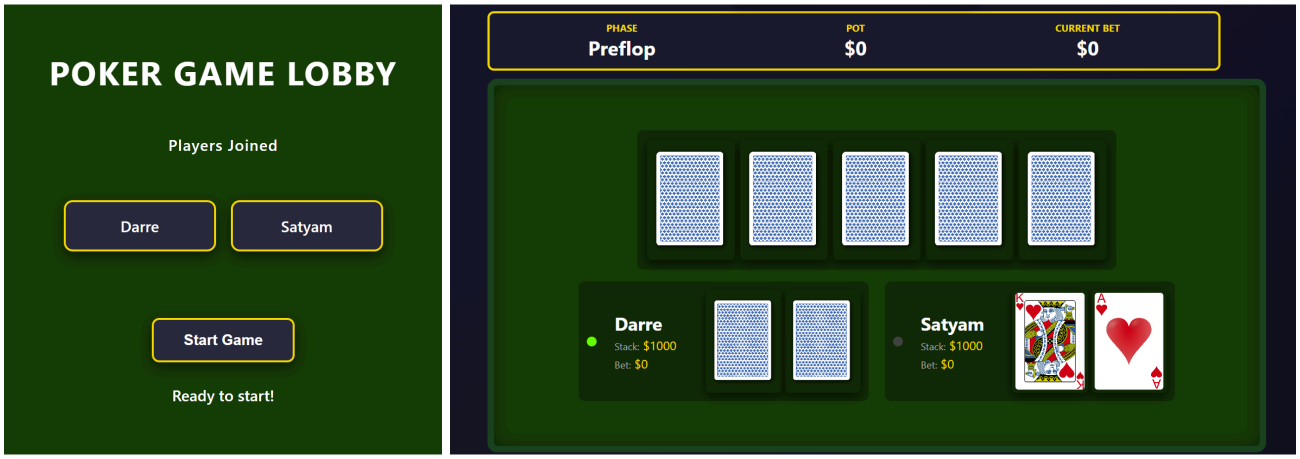


Figure 3. Server side view of the game including the lobby and broadcast view

Players connect to the server using the /join route, which registers them in the game lobby (shown on the left in the figure above). Up to two players can join at a time, and once both have connected, the host starts the game by sending a request to the /start endpoint. During gameplay, each player app continuously polls the server every 500 ms for updates to the game state. This ensures that when a card is scanned or any other change occurs, all players receive the update during the next polling cycle. While the server can view all card data, the total pot, current bet, and the current phase of the game, individual players see only limited information.

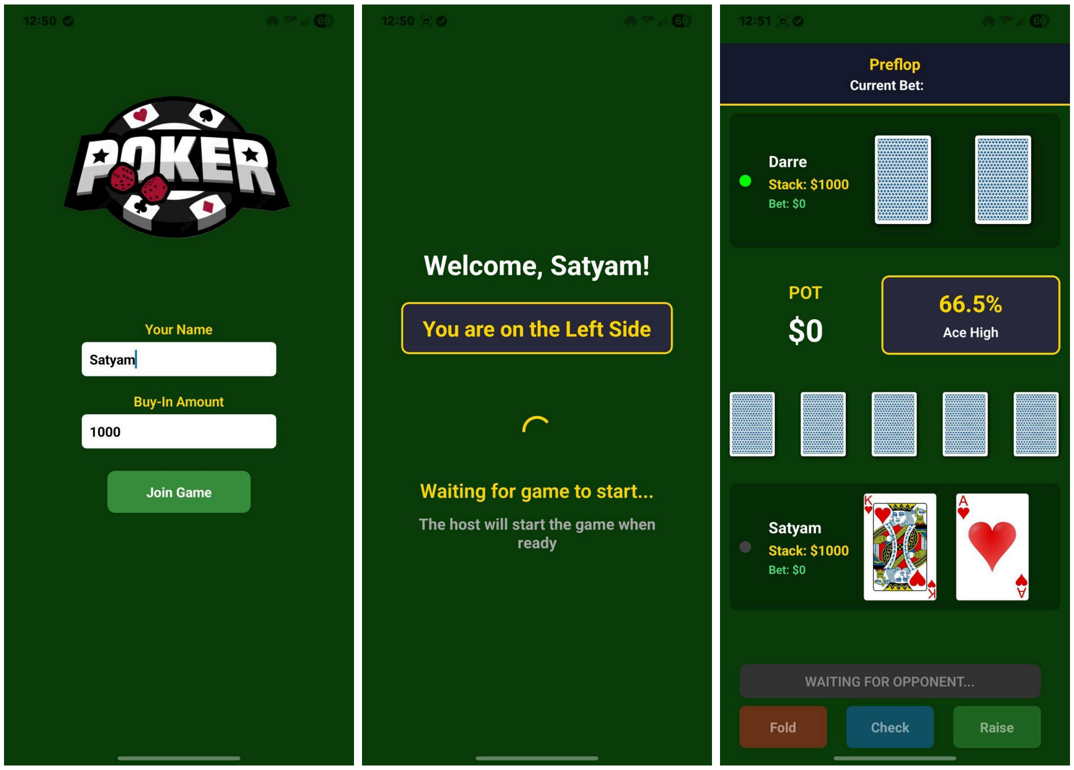


Figure 4. Player view of the game on the companion app (left: join screen, middle: waiting lobby, right: game)

When a player app polls the server, it compares the received state with the previous one to determine whether any UI elements need to be updated. If new cards have been scanned or if the gameplay phase has changed, the app updates its display accordingly. Each player can only see their own cards, their winning probability, and any community cards that are publicly revealed based on the current round. The winning probability is calculated using a 1,000-iteration Monte Carlo simulation of the remaining game state after each newly revealed card. A hand-ranking system is used both to evaluate the simulated hands and to determine the actual winner at the end of the game. If two hands rank the same, ties are broken by card values (e.g., a pair of tens beats a pair of twos). Refer to Table 1 below for a breakdown of the hand-ranking system.

Table 1. Hand ranking evaluations based on different poker hands

Hand	**Rank**
High Card	0
Pair	1
Two Pair	2
Three of a Kind	3
Straight	4
Flush	5
Full House	6
Four of a Kind	7
Straight Flush	8
Royal Flush	9

2.3.2 Chip Detection - Computer Vision

Our computer vision system combines classical image processing with a fine-tuned deep-learning model to accurately detect and evaluate poker chips in real time. We began by training a YOLOv8n [5] model, using a mix of publicly available labeled chip dataset [6] and our own custom-labeled images captured directly from the table. The data was organized into standard train, validation, and test splits, ensuring that the model learned robust chip shapes and colors during training, generalized well through validation, and was evaluated cleanly on unseen test images. Alongside YOLO detection, we incorporated contour extraction and color analysis to refine chip boundaries and classify chip colors with higher confidence. Each color class was then mapped to a specific chip denomination, and the program used two predefined betting zones on the screen to determine which player placed each chip. By combining model detection, contour processing, and zone-based assignment, the system reliably translates visual chip placements into structured betting information for our RFID poker table.

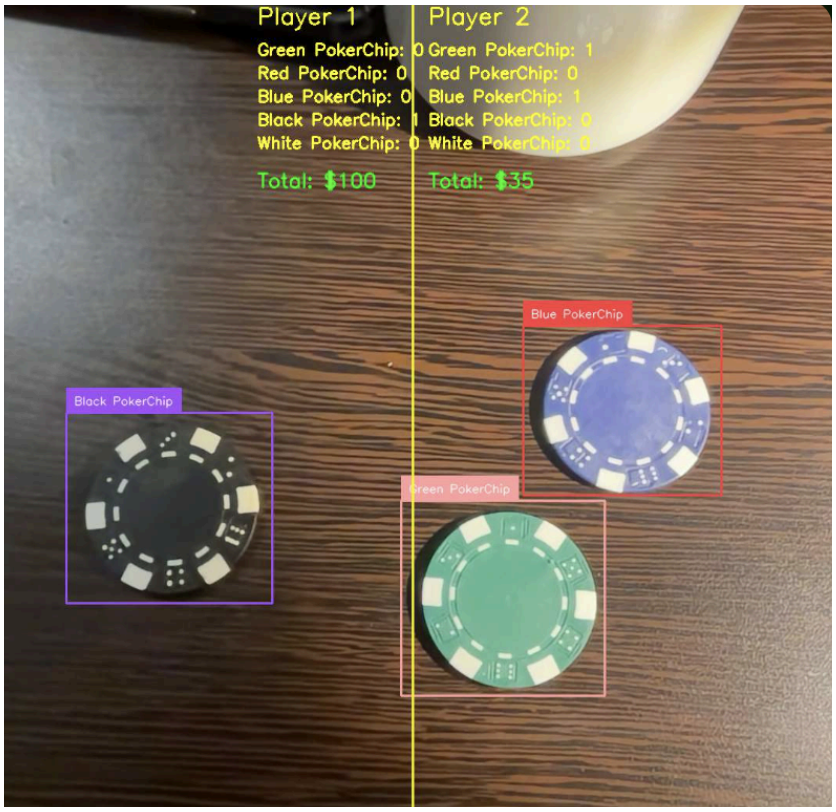


Figure 5. Chip detection logic shown visually with tagged chips and stream divider

This part of our remote system ran on a separate thread which first consumed the MJPEG stream broadcasted on the camera's IP address, processed each frame through object detection model, and calculated the bet sizes for each player. This design choice was done so that our remote system can serve the clients as well as perform chip detection in parallel.

2.4 Communication System

The communication system is the bridge between the physical sensors on the table and the remote game-state server. It is built around an ESP32-S3 microcontroller and a TCA9548A I²C multiplexer, and is responsible for polling the RFID readers, aggregating card detection events, and transmitting updates to the server over Wi-Fi. The system must operate reliably under continuous polling while maintaining low latency and stable network connectivity.

The ESP32-S3 was selected for its built-in Wi-Fi and Bluetooth as well as native I²C support. The dual-core architecture allows the firmware to dedicate one core to network communication while the other handles sensor polling, preventing blocking I/O from stalling the RFID read loop.

The TCA9548A I²C multiplexer solves a key limitation of the ESP32. The ESP32 does not have enough GPIO pins to scale the project for larger numbers of players. To make the design modular and capable of supporting more RFID readers, the TCA9548A was added to the system. This chip provides access to eight separate I²C channels and can be daisy-chained to support even more readers if needed. On a standard I²C bus, every device must have a unique address. The TCA9548A gives each reader its own isolated I²C channel while still communicating through one master connection to the ESP32. The firmware simply writes to the multiplexer's control register to select which channel is active, enabling one reader at a time. The multiplexer itself has a fixed base address of 0x70. To activate a particular channel, the program writes a value corresponding to that channel to the control register. For example, to enable channel 1, the firmware sends a command to address 0x70 with the control byte set to 0x01 (the bit that represents channel 1). This allows the ESP32 to communicate with the PN532 connected to that channel.

The firmware implements a round-robin polling loop that cycles through the three RFID readers on MUX channels 1,2, and 3. For each channel, the ESP32 sends an I²C command to the multiplexer to enable that channel, then issues a readPassiveTargetID() command to the PN532 on that channel with a 30 ms timeout. If a card is detected, the 7-byte UID is read and stored in a buffer. If no card is detected within the timeout, the firmware moves to the next channel. This strict timeout prevents any single reader from blocking the loop indefinitely. Once all three channels have been polled, the firmware packages the detected card UIDs into a JSON payload.

This payload is transmitted to the game state manager via an HTTP POST request to http://<server_ip>:5000/rfid_update. The ESP32 uses the Arduino HTTPClient library to establish a connection, send the JSON body, and wait for an HTTP 200 response. If the request fails due to network issues or server unavailability, the firmware logs the error and retries on the next polling cycle.

The communication system's reliability was validated through extended stress testing, where the firmware ran continuously for over 2 hours while polling three readers and maintaining an active Wi-Fi connection. During this test, the system achieved over 90% uptime (defined as successful HTTP posts on at least 90% of polling cycles), meeting the performance requirement.

2.5 Power System

The power system provides stable, regulated voltage rails to all subsystems on the table, ensuring reliable operation of the ESP32 microcontroller, RFID readers and I²C multiplexer. The Raspberry Pi camera is powered independently from its own 5V supply. The system accepts 120V AC wall power through a standard plug, converts it to 5V DC using an off-the-shelf power adapter, and generates a 3.3V rail on-board using a linear regulator.

2.5.1 Power Requirements

The power budget for the system was calculated based on the maximum current draw of each component:

- ESP32-S3 module: 500 mA peak during Wi-Fi transmission, ~80 mA idle

- Three PN532 RFID readers: 150 mA each when actively reading, ~50 mA idle (total: 450 mA peak, 150 mA idle)
- TCA9548A I²C multiplexer: ~10 mA

The total peak load on the 3.3V rail is approximately 960 mA (ESP32 + three readers + multiplexer). To provide margin for transient spikes and future expansion, the design targets a 3.3V regulator capable of delivering at least 1A continuous, and a 5V power supply rated for at least 2A.

2.5.2 Circuit Design

The primary power input is a 5V, 2A wall adapter (barrel jack connector) that plugs into the custom PCB. The 3.3V rail is generated using an AMS1117-3.3 linear regulator in a SOT-223 package. This regulator has a maximum output current of 1A, a typical dropout voltage of 1.2V. The AMS1117 was chosen for its low cost, ease of integration, and sufficient current capacity for the ESP32 and RFID readers.

Decoupling and filtering: Proper capacitor placement is essential for stable regulation. The design includes:

- Input capacitor: 10 µF tantalum capacitor at the 5V input to the regulator, placed within 1 cm of the input pin. This filters low-frequency supply ripple and provides charge storage during transient load changes.
- Output capacitor: 22 µF tantalum capacitor at the 3.3V output, placed within 1 cm of the output pin. This stabilizes the regulator's feedback loop and supplies instantaneous current during ESP32 Wi-Fi bursts.
- Local decoupling: 0.1 µF ceramic capacitors placed immediately adjacent to the ESP32's power pins and near each RFID reader power input. These filter high-frequency switching noise that can couple into sensitive analog circuits.

Tantalum capacitors were chosen for the bulk storage due to their high capacitance density and low ESR, which improves transient response. Ceramic capacitors were used for local decoupling due to their excellent high-frequency performance.

2.5.3 Thermal Considerations

Linear regulators dissipate power as heat according to:

$$P_{\text{dissipated}} = (V_{\text{in}} - V_{\text{out}}) \times I_{\text{out}}$$

For the AMS1117-3.3 under peak load:

$$P_{\text{dissipated}} = (5.0\text{ V} - 3.3\text{ V}) \times 0.960\text{ A} = 1.63\text{ W}$$

The AMS1117 in SOT-223 has a thermal resistance of approximately 50°C/W , so the expected temperature rise is:

$$\Delta T = 1.63\text{ W} \times 50°\text{C} / \text{W} = 81.5°\text{C}$$

At an ambient temperature of 25°C, the junction temperature would reach ~106°C, which is below the AMS1117's absolute maximum of 125°C but close enough to warrant thermal management.

2.5.4 Power Distribution

Power is distributed from the PCB to the three RFID readers via 8-pin header connectors that carry both 3.3V, ground, and I²C signal lines. Each reader is connected using ribbon cable or individual jumper wires. The multiplexer and ESP32 are powered directly from traces on the PCB.

3. Design Verification

3.1 Sensor System

3.1.1 RFID Reader

Following the verification steps described in Appendix A, the RFID reader successfully read the UID of all 100 tagged cards (100/100). The read speed was also significantly faster than the previous 1-second threshold and consistently remained below the median performance requirement. Across the 100 scanned cards, the median read time was 0.23 seconds, with a maximum of 0.65 seconds, which we believe may be due to human error during testing. These tests were done under the assumption that the tagged part of the card made full contact with the surface of the table and over 50% of the sticker fell inside the RFID zone marked on the board. These results could vary if both of these conditions are not met since the RFID has a limited range of detection.

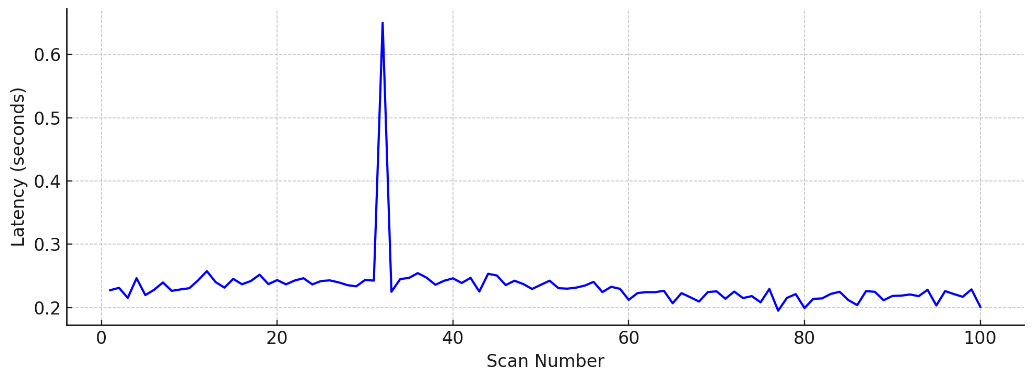


Figure 6. RFID scan latency over 100 scans

3.1.2 Camera

To verify the performance of our camera system, we evaluated the Raspberry Pi ribbon camera and its MJPEG streaming server running on the device's local IP. The camera served an MJPEG feed directly over HTTP, so we could easily measure frame rate, latency, and uplink behavior without additional encoding overhead. We tested FPS (Frames Per Second) by capturing timestamps of consecutive frames from the stream and confirming that the camera maintained a stable frame delivery rate. Latency was assessed by running a simple end-to-end delay test placing a motion event in front of the camera and comparing the real world timing to when the frame appeared in the client application, aided by a software timestamp, helping us verify that MJPEG kept delays extremely low. For uplink testing, we monitored network throughput while streaming, examining how much bandwidth the MJPEG images consumed at extended runs to ensure the system remained stable over Wi-Fi and did not saturate. Together, these tests confirmed that the Raspberry Pi camera's MJPEG broadcast provided a reliable, low-latency, and sufficiently performant video feed for real-time chip detection.

Table 2. Camera verification measurements at a resolution of 640×480

Metric	**Measurements**
Average FPS	29.6 fps (±1.5 fps)
End-to-end Latency	55 ms (±10 ms)
Uplink Throughput	1.36 MB/s ≈ 10.9 Mbps (avg frame size ~46KB)

3.2 Remote System

3.2.1 Companion App

The app was able to run on both an iPhone 16 (iOS) and a Samsung Galaxy S2 Plus (Android), making it compatible with both major operating systems. Another requirement was ensuring that game state updates were reflected within 1 second of a change occurring on the board or on the other player's companion app. As described in Appendix A, we tested this across 102 moves. The results showed an average update latency of 0.84 seconds, with a maximum of 0.97 seconds and a minimum of 0.73 seconds. These metrics meet the requirement for timely in-game updates.

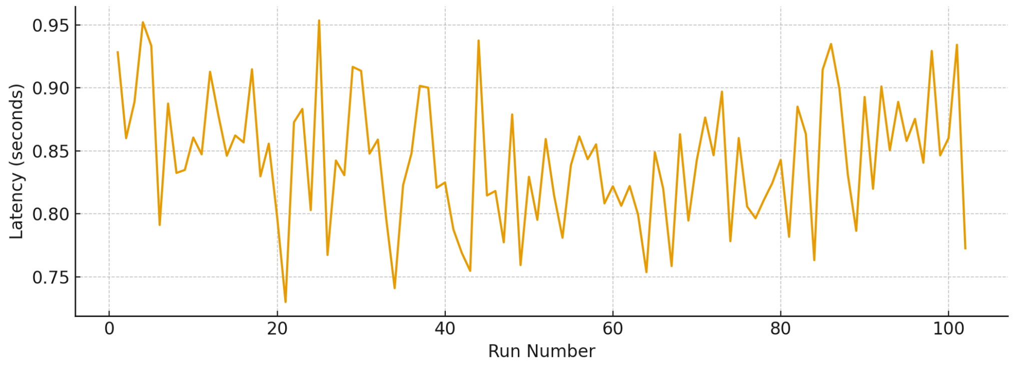


Figure 7. Game state updates over 102 moves

3.2.2 Monte Carlo Simulation

One key decision we made was determining the right number of simulations to find the sweet spot between the model's latency and accuracy. To do this, we ran the same Monte Carlo simulation multiple times using the same starting hand (Ace of Spades and 6 of Diamonds) and measured the model's accuracy in estimating the win probability. This estimated probability was then compared against the true win probability calculated beforehand. The figure below illustrates the trade-off between latency and accuracy in the Monte Carlo simulation. As shown, the model begins to level off in accuracy at around 3,000 simulations, after which the increase in latency no longer justifies the marginal gain in accuracy. The variations in latency can be attributed to network effects, while the variations in the Monte Carlo accuracy stem from the randomness inherent in sampling. Regardless, the accuracy curve clearly tapers off, while latency continues to increase. This suggests that 3,000 simulations is an appropriate configuration for this project, providing the best balance between accuracy and performance.

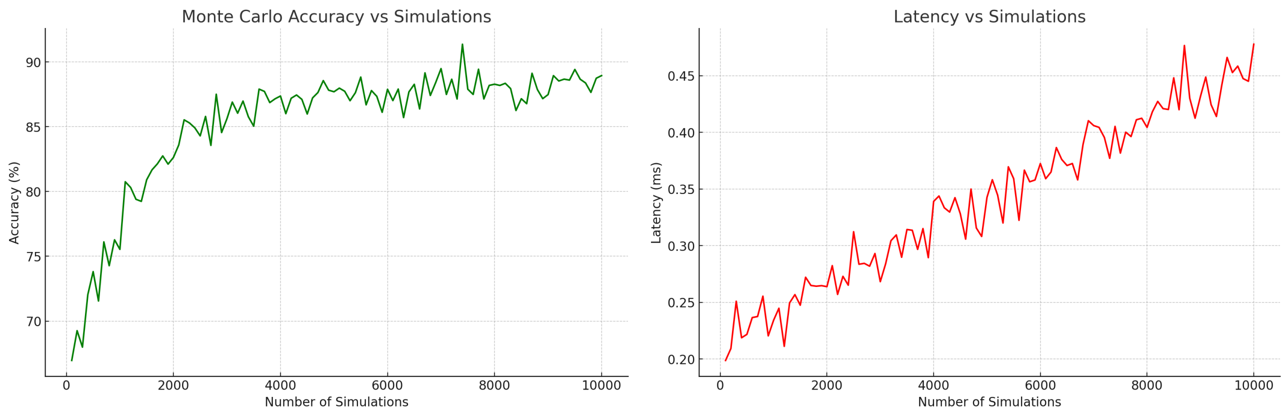


Figure 8. Left: Accuracy of Monte Carlo model over simulations. Right: Latency of Monte Carlo over simulations

3.2.2 Chip Detection - Computer Vision System

The model achieved strong overall detection performance on the poker chip dataset, reaching an mAP@0.5 of 0.919 on the testing data set, which reflects strong performance across all classes. The model performs exceptionally well on the Green, Blue, and White chip classes, with AP scores between 0.96–0.99 and consistently high precision and recall across all confidence levels. In contrast, the Red and Black classes show lower AP values (0.77–0.78), suggesting greater visual ambiguity or class imbalance. Despite these challenges, the model provides reliable, high-quality detection suitable for real-world deployment, especially given its lightweight architecture. The testing conditions we assumed were that all chips were laid out flat on a table without overlap, and the table was well lit around ~750 lux.

Table 3. Average precision over different color of poker chips

Class	AP (Average Precision)
Black PokerChip	0.775
Blue PokerChip	0.981
Green PokerChip	0.995
Red PokerChip	0.785

White PokerChip	0.967
All Classes (mAP@0.5)	**0.91**

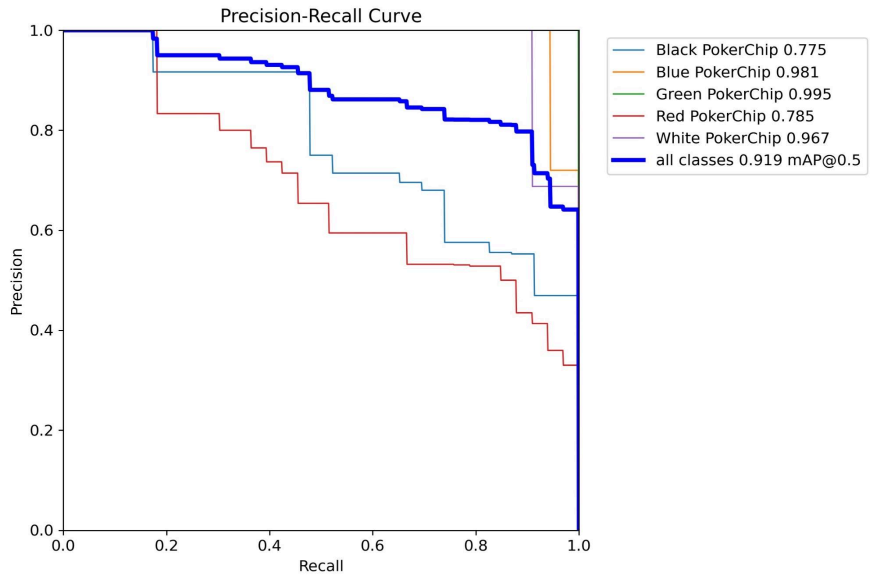


Figure 9. Line graph of the precision-recall curve of the different colored poker chips

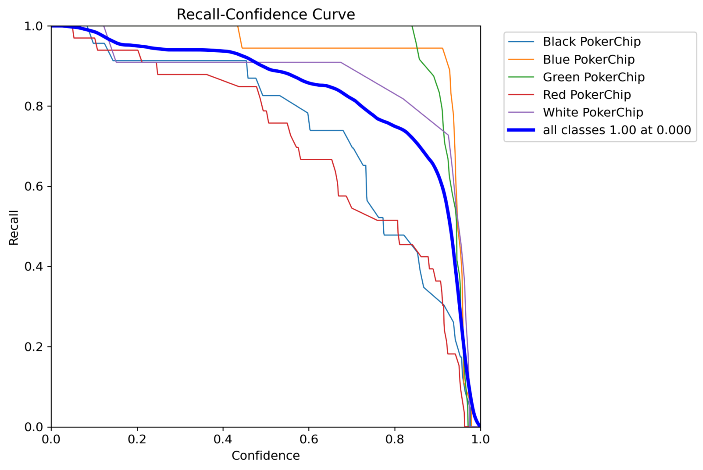
Another metric we used to evaluate the model is the precision and recall against confidence thresholds. These metrics are derived from the relationships among true positives (TP), false positives (FP), and false negatives (FN) to assess the correctness of detections.

Precision measures the proportion of predicted detections that are actually correct. A high precision value indicates that the model rarely produces false alarms. Formally:

$$Precision = TP / (TP + FP)$$

Recall measures the proportion of ground-truth objects that the model successfully detects.
A high recall value means the model is missing very few actual objects. Formally:

$$Recall = TP / (TP + FN)$$



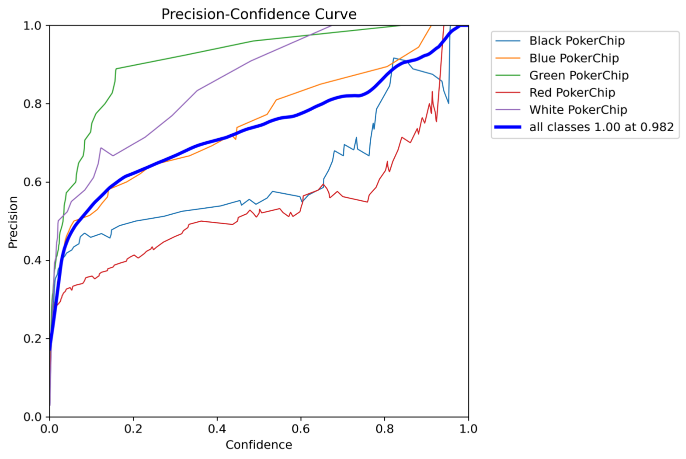


Figure 10. Left: Recall confidence curve. Right: Precision confidence curve.

Precision and recall are inversely correlated (trade off against each other), hence the F1 score combines them via the harmonic mean:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

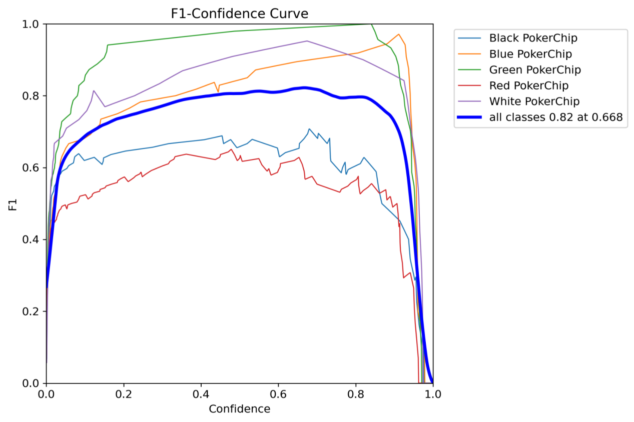


Figure 11. F1-Confidence curve of the different colored poker chips

The F1–Confidence curve shows an optimal operating threshold around 0.668, where the model best balances precision and recall. Overall, the results demonstrate that the model reliably detects most chip colors with high confidence, with clear opportunities to improve performance on more challenging classes.

4. Costs

The cost of this project can be divided into labor, testing/prototyping materials, and final construction. The total cost of labor can be estimated using the average annual earnings of a Computer Engineering graduate from UIUC, which is approximately $118,752 per year, or $57.09 per hour [7]. Assuming each team member works an average of 6 hours per week for 14 weeks, the total labor cost for the engineers amounts to $14,386. However, in order to build the actual poker table an estimated 4 hours of work from the machine shop was quoted. With the average of a machine shop worker in Champaign, IL being around $26.00/hr the total for this work would be roughly $104, bringing the total labor cost to $14,409.68 [8]. During the process of testing and developing the RFID board, several development and breakout boards were used to expedite prototyping and demonstrate proof of concept. The costs of these boards and their respective prices are itemized in the following section. Finally, all individual components used in the final version of the RFID board are listed below. Combining the total cost of development boards ($70.58), the final parts ($16.53), and labor ($14,386), the estimated total project cost is approximately $14,577.79.

4.1 Parts

Table 4. Parts list with corresponding quantities and price

Part	Manufacturer	Quantity	Price per ($)	Total ($)
PN532 RFID Reader	HiLetgo	3	11.49	34.47
TCA9548A I²C Multiplexer Module	Adafruit	1	13.12	13.12
0.1 µF 50V Ceramic Capacitor (X7R, 0805)	Yageo	3	0.08	0.24
22 µF 25V Ceramic Capacitor (X5R, 0805)	Samsung/Yageo	2	0.08	0.16
1 µF 50V Ceramic Capacitor (X7R, 0805)	Samsung	1	0.08	0.08
Tactile Switch SPST-NO, Boot/Reset	E-Switch	2	0.42	0.84
10KΩ 1% Resistor 0805	Vishay Dale	3	0.10	0.30
100KΩ 1% Resistor 0805	Vishay Dale	11	0.10	1.10
5.11KΩ 1% Resistor 0805	Vishay Dale	2	0.10	0.20
4.7KΩ 1% Resistor 0805	Vishay Dale	10	0.10	1.00

ESP32-S3-WROOM-1U Module	Espressif	1	6.56	6.56
USB-C 24-Pin Connector	GCT USB4105-GF-A	1	0.78	0.78
Linear Regulator 3.3V 1A (AMS1117-3.3)	UMW	1	0.68	0.68
10 µF 16V Tantalum Capacitor	KYOCERA AVX	1	0.35	0.35
22 µF 25V Tantalum Capacitor	KYOCERA AVX	1	1.01	1.01
Barrel Jack Power Connector	Nebj	1	0.71	0.71
8-Pin Male Header	SparkFun	7	0.75	5.25
Total				**64.12**

5. Conclusion

The final prototype operated as intended and met both the high-level functional requirements and the individual subsystem goals. By the end of development, the system was able to accurately read player and community cards, recognize chip values through real-time vision, and synchronize game information to a digital interface. This project also provided valuable hands-on experience in PCB design and revealed many practical lessons about I²C communication, including addressing conflicts, timing issues, and hardware debugging challenges that do not appear in simulations or datasheets.

5.1 Ethical considerations

The RFID Poker Table must be developed responsibly due to its connection to gambling. Although designed for recreational and educational use, it could be misused in unregulated environments. In accordance with the IEEE and ACM Codes of Ethics, engineers must protect public welfare and avoid contributing to harmful or illegal behavior [9], [10]. To address this, we clearly document that the system is not intended for commercial gambling and implement protections that prevent unfair advantages, such as restricting access to hidden card information. Honesty and fairness are also core ethical duties. We will accurately report limitations, such as occasional RFID or chip detection errors, and avoid overstating capabilities like tracking precision or possible AI features. Ensuring equal access to game information aligns with ethical principles of fairness and transparency [6]. Safety compliance remains equally important. The design will integrate proper AC-to-DC power conversion, overcurrent protection, and insulation to meet electrical safety standards [11], [12]. All RFID modules operate within FCC Part 15 regulations [13], and laboratory work will follow campus safety rules for soldering, wiring, and handling exposed electronics.

5.2 Future work

Future iterations of the system could expand the number of players supported while making the game more immersive. For example, each player’s phone could be replaced by a dedicated LED display mounted into the table, eliminating dependency on mobile devices and creating a more uniform interaction experience. The sense of realism could also be improved by supporting chip stacking recognition, which would require multiple cameras positioned at angles that allow the system to measure stack height or density. Additionally, enhancements to I²C reliability, such as automatic re-addressing, bus arbitration, or backup communication channels, would ensure scalability as more sensors and modules are added to the PCB architecture. Together, these improvements would push the system closer to a commercial, casino-style digital poker table with both high reliability and an intuitive player experience.

References

[1] K. Browne, "Home Poker and Informal Gambling: Social Play and Rule Negotiation," Journal of Leisure Studies, 2018.

[2] J. Palomäki, M. Laakasuo, and M. Salmela, "Expertise and Emotion in Poker Decision-Making," Journal of Gambling Studies, vol. 37, no. 2, pp. 457–474, 2021.

[3] Skadi, "Misconception 14: RFID Signals Can Pass Through Any Material," *RFID Label*, 11 months ago. [Online]. Available: https://www.rfidlabel.com/an-analyzing-guide-of-misconception-in-rfid-what-you-need-to-know/#:~:text=best%20read%20performance.-,Misconception%2014%20:%20RFID%20Signals%20Can%20Pass%20Through%20Any%20Material,different%20applications%20with%20careful%20planning. [Accessed: Oct. 12, 2025].

[4] Visioforge.com. (2025). HTTP MJPEG Video Streaming Implementation Guide | VisioForge Help. [online]. Available at: https://www.visioforge.com/help/docs/dotnet/general/network-streaming/http-mjpeg?srsltid=AfmBOopX_Q4D9sQC3azp8xQi59kenrmg-Ab9ljq7Qz3mFhoTjsshBf65 [Accessed 5 Nov. 2025].

[5] Ultralytics. “Explore Ultralytics Yolov8.” Ultralytics YOLO Docs, 28 Oct. 2025, docs.ultralytics.com/models/yolov8/#performance-metrics.

[6] topherspace (2024). Poker Chip Count Object Detection Model by topherspace. [online] Roboflow. Available at: https://universe.roboflow.com/topherspace/poker-chip-count [Accessed 5 Nov. 2025].

[7] University of Illinois Urbana-Champaign, “Computer Engineering,” *The Grainger College of Engineering*, 2025. [Online]. Available: https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/computer-engineering

[8] ZipRecruiter, “Machinist Salary in Champaign, IL: Hourly Rate,” *ZipRecruiter*, 2025. [Online]. Available: https://www.ziprecruiter.com/Salaries/Machinist-Salary-in-Champaign,IL. [Accessed: Oct. 11, 2025]. ziprecruiter.com

[9] IEEE, “IEEE Code of Ethics,” IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[10] ACM, “ACM Code of Ethics and Professional Conduct,” Association for Computing Machinery, 2018. [Online]. Available: https://www.acm.org/code-of-ethics

[11] UL Standards, “UL 60950-1: Safety of Information Technology Equipment,” UL Standards, 2014.

[12] IEC, “IEC 62368-1: Audio/Video, Information and Communication Technology Equipment – Safety Requirements,” International Electrotechnical Commission, 2018.

[13] Federal Communications Commission, “Title 47 CFR Part 15 – Radio Frequency Devices,” FCC, 2021.

Appendix A Requirement and Verification Table

Requirement	Verification	Verification status (Y or N)
Must continuously supply ≥300 mA at 3.3 V ± 0.05 V to ESP32, RFID Readers, and MUX	1. Power the board with a wall adapter or bench supply at nominal VIN. 2. Connect electronic load to 3.3 V rail at the load sense pads. 3. Use the oscilloscope: Probe across the 3.3V load sense pads (positive to VOUT, ground to GND)	**Y**
Must process RFID broadcast updates within ≤1 second	1. Put a scope probe on a GPIO that toggles high on PN532 detection and low when the network send completes. 2. Sniff network (PC) and record packet arrival times. 3. Measure GPIO pulse width (processing time) and GPIO rising edge ⟶ packet arrival (end-to-end). **Pass if:** All measured end-to-end intervals ≤ 1 s over 25+ trials.	**Y**
Must support simultaneous polling of up to 3 RFID readers	1. Connect 3 PN532 to distinct MUX channels, place three different tags 2. Run firmware that round-robins channels with minimal inter-channel gap and no per-channel blocking (use per-channel timeouts ≤ 10 ms). 3. Log detections (UID + channel) for 2 minutes. **Pass if:** All three channels report stable, continuous presence with correct channel attribution; no channel starves for >200 ms.	**Y**

RFID Reader must detect standard HF RFID tags (13.56 MHz) with ≥95% reliability	1. Connect 1 PN532 module to ESP32. 2. Place one 13.56 MHz tag directly above the antenna at the designed mounting distance 3. Disable Wi-Fi and other readers to isolate the test. 4. Present the tag for 1 s, remove it for 1 s. 5. Repeat for **100 cycles**. 6. Log every detection/removal event on ESP32 (UID detected, timestamp). **Pass if:** ● At least **96 detections** out of 100 cycles succeed. ● No false positives during “no tag” intervals.	**Y**
RFID Reader must detect a card placement/removal within ≤1 seconds	1. Use firmware timestamps (*t_detect_start* when PN532 read loop sees new UID; *t_publish* when message sent to network). 2. On PC, log message arrival time *t_rx*. 3. Perform 100 card placements and 100 removals spaced by ~2 s. 4. Compute *t_rx* − *t_placement* and *t_rx* − *t_removal* using synchronized clocks (ESP32 NTP + PC). **Pass if:** ● **100%** of events detected and broadcast ≤ 1.0 s after physical movement. ● Median latency ≤ 300 ms.	**Y**
CAM must stream realtime video with at least 24fps	1. Connect CAM to Wi-Fi 2. Start MJPEG stream 3. Mark time at each JPEG boundaries 4. Serial info log the time difference **Pass if:** Average FPS ≥ 24; frame drop < 10% per 30 s window.	**Y**
Must display updated game state within ≤1 seconds of change	1. Insert or remove a tagged card, and mark the physical action time with a pushbutton tied to ESP32 GPIO (firmware logs timestamp). 2. Insert or remove a chip from the pot, and mark the physical action time with a pushbutton tied to another ESP32 GPIO (firmware logs timestamp). 3. The ESP32 publishes an update message with an	**Y**

	embedded timestamp. 4. The client app logs the timestamp when the UI element changes. 5. Collect 100 + samples in various conditions (idle network, camera streaming, full table). **Pass if** ● 100 % of updates ≤ 1.0 s. ● No missed or duplicated state updates including cards and stack sizes	
Must run on both iOS and Android devices	1. Test on at least: ○ iPhone 12/13/14 or newer (iOS 16+) ○ Pixel 5/6 or equivalent Android 12+ 2. Perform a **full functional run**: ○ Login/authentication ○ Game lobby join ○ Card display updates ○ Betting actions (if applicable) ○ Video or camera view (if implemented) 3. Compare UI layout, scaling, and performance (FPS, input latency). **Pass if** ● All features behave identically and correctly on both platforms. ● No critical UI errors, crashes, or missing assets. ● Startup time ≤ 3 s on either platform.	**Y**
Must enforce access control so players only see their own hole cards	1. Deal two cards to each player; verify backend associates correct UID with each player. 2. On every client, check visible cards: ○ Player’s own two hole cards visible. ○ All others’ hole cards are hidden or back-faced. ○ Community cards visible to all. 3. Perform “role switch” tests: ○ Log in as different user ⟶ UI updates to show only that user’s cards.	**Y**

	Pass if ● Each user sees only their own hole cards and shared cards.	