ECE 1896

Senior Design

Poker AI/ML Coach Conceptual Design

Team #12

Prepared By:
Seth Williams – Electrical Engineer
Benjamin Wu – Electrical Engineer
Nicholas LaVine – Computer Engineer
PJ Granieri- Computer Engineer

Table of Contents

Table of Figures

Table of Tables

1. Introduction

Poker is a game of strategy, probability, and psychology, but its complexity makes it extremely difficult for beginners to learn effectively in a quick manner. New players will struggle with decision making because of the many factors in the game, such as hand strength, river cards, betting amounts and the dynamic nature of the pot. This creates a need for a tool that can guide new players in real time to help them understand not only what decision to make but also why certain moves are stronger than others.

To address this challenge, our project proposes the design and development of a Poker AI/ML Coach. This system will combine computer vision, cloud-based machine learning, and embedded hardware to create an interactive assistant capable of analyzing live poker games and recommending actions. The coach will serve as both an educational tool for beginners and a strategy aid for casual players who want to compare poker decisions.

At a high level, the prototype will consist of four main components: image capture and preprocessing, cloud-based recognition and decision making, local decision display and onboard power system with a rechargeable battery. A camera module (OV5640) will capture images of the cards, rivers and chips on the table. On the device side, C++/Arduino code will handle tasks such as image detection, cropping, ordering, and packaging the data in JSON format. The formatted data will then be sent securely to Azure Cloud.

Once in the cloud, Azure services will process the data, and perform machine learning based recognition of the cards, river state, and chip amounts. This information will then be passed to a pre-trained AI/ML poker model, which will determine the optimal recommended actions (fold, call, raise). The decision will be sent back to the device via HTTP protocol and be displayed clearly on an LCD screen for the user to view in real time.

The entire system will be powered by an internal battery, ensuring that no external power source is required. This will enable the coach to be able to be used in different environments. This battery will take a USB-C Input, and all components in the prototype will run off battery power. The final prototype will demonstrate the full workflow of all the features listed above.

By doing all the features listed, the Poker AI/ML Coach will provide an innovative solution to the difficulty of learning and playing poker, making the game more approachable and enjoyable for new players while showcasing the integration of computer vision, machine learning, embedded systems with the hardware components of this prototype.

2. Background

Poker (specifically Texas Hold'em) is a widely played game of imperfect information that combines probability, psychology, and strategic decision-making. Millions of people play both live and online; the online poker market was estimated at 5.3 billion USD in 2024 [1] and continues to grow, indicating a large potential user base for tools that support learning and decision feedback. Tournament series like the World Series of Poker has hundreds of thousands of entries [2] and highlight the scale and popularity of the game. These trends point to significant demand for coaching, training, and assistive tools for both casual and serious players. A poker coaching device must be able to support several functions: card recognition, hand evaluation, and

a user interface that communicates advice without disrupting live play. As a training aid it can provide large teaching value by giving immediate, objective feedback that accelerates learning.

1. **MCU Module ESP32-S3_WROOM-1:** The ESP32-S3 is a dual-core microcontroller module designed for embedded AI and vision tasks. It runs up to 240 MHz and includes memory options up to 8 MB PSRAM and 16 MB flash and provides built in Wi-Fi 4 and Bluetooth 5 LE. These features make it an excellent choice for a portable poker coaching system, as it combines computing power with wireless connectivity in a compact package.

   A key advantage of the S3 series is its vector instructions for AI/ML acceleration, which helps speed up neural network interference on device. The module also has an integrated parallel camera interface (DVP) for popular CMOS sensors and efficient DMA handling, which ensures smooth image capture without overloading the CPU [3].

2. **Power – USB-C Charging and Portability:** USB Type-C with USB Power Delivery has become the dominant standard for charging personal/small devices and can supply different power levels, it can deliver up to 250 watts of power [4]. Using USB-C simplifies charging logistics and enables higher charging currents for fast recharge. For a battery-backed coaching unit, a USB-C capable charging and power-path design: (CC/PD Controller + charger IC + Point of Loads) [5] enables safe, standardized recharging and the option to operate while charging.

   Designing the power subsystem requires attention to thermal limits, inrush current, and USB-C Handshake and possible PD negotiation. For battery selection, energy budget estimates should account for camera capture, ML interface, and display refresh; typical small embedded systems can aim for several hours of operation on a few thousand mAh depending on duty cycle [5].

3. **Signal-Integrity / Machine Learning:** Two technical challenges dominate the vision pipeline: Capturing clean, low-noise images for reliable card recognition, and classifying playing cards in real time using ML.

   On the hardware side, camera interfaces must be routed and terminated correctly to avoid image corruption; power rails to the image sensor and MCU must be decoupled to prevent jitter [6].

   On the ML side, modern approaches to playing-card recognition typically use convolutional neural networks (CNNs) for classification or object-detection models for locating cards in frames. CNNs can achieve high accuracy for face/value/suit recognition when trained on datasets. For a portable device, common strategies are using a fast detector (like YOLO) to find card regions, then run a small classifier to determine rank and suit; use model quantization for on device interference; or offload heavier interference to a server if latency constraints and connectivity allow [7].

4. **Training a Model on Large Datasets:** To provide the most accurate feedback possible for the user, the learning model used within this poker coach design must be trained rigorously so it can always understand the best play. The challenge with that is poker is an extremely complicated game that introduces a lot of variability in every round. To combat this, the model must be trained on copious amounts of data from

professional players and extremely skilled robots. The more training data provided to the bot the better it will be at being able to handle any situation that may arise during the game so that it may provide the most accurate play at all times for the user.

5. **Custom Solution:** Existing solutions for card recognition and poker analysis demonstrate feasibility of reliable card detection using CNNs and object detectors; however, many of the are projects that rely on limited lighting or direct connection using a cable. Our design targets a small, battery powered, device that integrates robust hardware signal-integrity practices for consistent image capture under varied lighting, runs a lightweight, pre-trained ML program that can successfully make accurate predictions as well as be optimized for the chosen MCU to meet strict latency targets, and uses USB-C PD for convenient, fast charging. This unique combination; from optics and SI to embedded interference and human-centered feedback; enables a real-time, standalone coaching experience.

2.1 Definitions

2.1.1 Poker Related Definitions

- Hole Cards: the two private cards dealt to each player at the start of a hand.
- Community Cards: the five cards dealt face-up in the center of the table for all players to use.
- Pre-flop: the first betting round, before any community cards are revealed.
- Flop: the first three community cards dealt face-up.
- Turn: the fourth community card dealt face-up.
- River: the fifth and final community card dealt face-up.
- Showdown: the final stage of a hand when remaining players reveal their hole cards to determine a winner.
- Pot: the total amount of chips wagered during a hand.
- Fold: discard your hand and forfeit the pot.
- Call: to match the current bet.
- Raise: to increase the size of the current bet.
- Check: to pass the action without betting, available if no bet has been made.
- Betting Round: a phase in poker where players take turns deciding whether to fold, call, or raise.
- Big Blind & Small Blind: mandatory bets placed by players before cards are dealt, ensuring that money is in the pot.
- Hand Strength: the relative value of a player's cards compared to all possible hands.
- Imperfect Information: a game state where not all cards or intentions are known
- Equity: the chance of being able to win any specific hand at a given time.
- Expected Value: the expected value on a certain decision in regard to winning chances and return on investment.

2.1.2 Technical Definitions

- MCU (Microcontroller Unit): A small computer on a single integrated circuit used for device control.
- ML (Machine Learning): The ability to train a machine using data provided to it so the model can learn from the data and complete tasks using the knowledge gained from the training process.
- JSON (JavaScript Object Notation): a lightweight data format used to transmit structured information between the device and cloud.
- LCD (Liquid Crystal Display): the screen on which the device displays decisions.
- Latency: the time delay from image capture to displayed decision output.
- Inference: the process of using a trained ML model to make predictions on new data
- Training Dataset: a collection of labeled examples used to teach the Machine Learning model.
- Cloud Computing: using remote serves such as Microsoft Azure for computation and storage, instead of local processing on the MCU.
- HTTP/HTTPS POST: a protocol for securely transmitting data from the device to the cloud.
- PSRAM (Pseudo-Static RAM): a type of memory used in the ESP32-S3 to support image buffering
- USB-C PD (USB-C Power Delivery): a standard for negotiating power levels over USB-C, allowing safe, fast charging
- MLP (Multilayer Perceptron): a type of machine learning model that is very stable, quick to train, and easy to understand systematically.
- RL (Reinforcement Learning): a way to train a machine by having it learn itself by playing using a chosen reward system. Can be very computationally heavy, but very accurate.

3. System Requirements

This system is designed to provide real-time poker decision assistance through computer vision and AI-based decision-making. The user will use the system by placing the device in view of the playing area. The camera will capture cards, chips, and actions, sending the data to the MCU, and then transmit the data to a cloud-based image processing algorithms & AI/ML models for further analysis. The AI will then return the decision which will be displayed on an LCD.

The following requirements outline the necessary functions, constraints, and performance expectations for the system.

3.1 Functional Requirements

3.1.1 Camera Capture

- The OV5640 camera shall capture images of player hole cards, community (flop, turn, river) cards, and chip stacks.

- Captured images shall be transferred to the MCU via the parallel DVP interface.
- The camera shall support a minimum frame rate of 15 FPS to ensure timely recognition.

3.1.2 Image Preprocessing on MCU

- The MCU shall preprocess raw images by cropping, compressing, and formatting into JSON-compatible payloads.
- Preprocessing shall include resolution reduction to balance recognition accuracy and latency.
- The MCU shall perform preprocessing without exceeding 75% of the available SRAM.

3.1.3 Data Transmission

- The MCU shall transmit preprocessed data to the cloud via HTTP/HTTPS POST requests.
- All transmissions shall be encrypted (HTTPS)
- Failed transmissions shall be retried up to 3 times before returning an error to the user.

3.1.4 Cloud Processing

- The cloud server shall accept JSON inputs from the MCU and store them in temporary storage such as an Azure Blob.
- A serverless function shall trigger recognition algorithms (card, chip, pot) and forward results to the ML poker model.
- The ML poker model shall output one of the following recommendations: Fold, Call, Raise.

3.1.5 Machine Learning Model

- The model shall accept results from the image processing module and make predictions off the data.
- The model shall make predictions fast for real-time deployment
- The model outputs the predictions in a standardized format and sends them to the MCU to be displayed.

3.1.6 Decision Display

- The MCU shall receive the cloud decision output and display it on an LCD screen within 5 seconds of image capture.
- The display shall clearly indicate the recommended action and supporting game state information such as recognized cards and pot size.

3.1.7 Power Management

- The system shall operate on an internal rechargeable Li-ion battery.
- The battery system shall support charging via USB-C at 5V/2.5A minimum.
- The device shall be able to run solely on battery for at least 3 continuous hours.

3.1.8 User Interaction

- The system shall provide a physical power button to turn the device on/off.
- The system shall allow placement in view of the poker table without interfering with gameplay.
- The system shall not require the user to manually enter card values or chip counts.

3.2 Performance Requirements

3.2.1 Winning Rate

- The assistant shall achieve a success rate greater than 50% of hands participated in after 10 trials.
- A “hand participated in” is defined as the assistant choosing Call or Raise during pre-flop.

3.2.2 Profitability

- The assistant shall have a net profit after a minimum of 10 hands in at least 60% of test sessions.
- Profitability shall be evaluated using simulated play with controlled opponents.

3.2.3 Latency

- The system shall achieve end-to-end latency of less than 5 seconds from image capture to decision display.

3.2.4 Recognition Accuracy for Cards

- Player hole cards shall be recognized with a True Positive Rate (TPR) = 100%
- Community Cards (flop/turn/river) shall be recognized with a TPR >= 99%
- Misclassifications shall not exceed 1 error per 100 cards in test datasets.

3.2.5 Recognition Accuracy for Chips and Pot

- The chip count shall be recognized with a TPR >= 99%
- Recognized values shall be accurate to within +/- 1 chip for test scenarios

3.2.6 Prediction Accuracy

- The model shall be able to make accurate predictions based off the training data with a loss <= .05.
- The model shall not overfit the training data

3.2.7 Reliability

- The system shall remain operational for at least 3 continuous hours without reboot.
- The system shall successfully recover from at least 95% of transmission failures via retry logic.

3.2.8 Usability

- The LCD display shall present recommendations in a readable font size visible from 1-2 feet away.

3.2.9 Safety & Compliance

- The battery management system shall prevent overcharge, short circuit, and overheating.
- All components shall operate at low voltage (< 12 V) to ensure user safety.

4. Design Constraints: Standards and Impacts

4.1 Design Constraints

4.1.1 Time

The project must be completed within a single semester, which imposes significant limitations on system complexity, the number of prototypes that can be built, and the extent of iterative testing and training possible. Because of the short timeline, each team member is required to take on responsibilities that may involve learning new skills or technologies they have not previously worked with. This learning process takes time, which reduces the amount of time available for design, implementation, and refinement. As a result, mistakes or design flaws discovered late in the semester carry greater risk, since there may not be enough time to redesign hardware, retrain models, or thoroughly validate performance before the final deadline.

4.1.2 Manpower

The team consists of four members, each with limited weekly availability due to coursework and other commitments. Tasks must be divided efficiently, and design decisions must account for the team's combined skill set in embedded programming, machine learning, and hardware prototyping.

4.1.3 Financial

The project budget is constrained by the $200 cap from the school. Thus limits the choice of camera modules, microcontrollers, and computing hardware, which may affect performance and system robustness. Additionally, the budget restricts the ability to purchase evaluation or development boards. These boards typically range from $50–$100 each and are commonly used in industry to evaluate chips, test them under different scenarios, and gather real-world performance data before committing to custom hardware. Within the $200 cap for the entire prototype, acquiring such boards is not feasible. As a result, careful study of datasheets and reference designs becomes essential for understanding device behavior and reducing risk during PCB design.

4.1.4 Available Data for Training Model

The performance of the poker recognition AI is limited by the size and quality of the training dataset. Since poker card recognition datasets are not widely available, we must generate a significant portion of the training data ourselves, which restricts the amount of model training that can be accomplished within the semester. A lot of the poker data available online cannot be used due to it being incomplete or provided by novice players that will not benefit the prediction ability of the model.

4.1.5 Processing Power for Training Model

The ability to train an accurate model is limited by the computational power available for use on the laptop that will be used for training the future model. We must limit the amount of data used for the model and optimize it as much as possible to avoid having days of runtime which will slow down our design process substantially. This also limits our choices of models to specific algorithms that take less processing power to train and implement, which bottlenecks our ability to store resource-intensive models locally.

4.1.6 Prototyping Hardware

We are limited to hardware that is affordable and compatible with microcontroller-based development. Access to high-performance GPUs or cloud computing resources for training may be limited, potentially reducing model accuracy or requiring lighter-weight AI models.

4.1.7 Limited Reference Designs Online

Few open-source projects directly match the scope of an AI poker coach with real-time card detection and strategy display. This means the team must spend additional time developing and validating custom solutions rather than relying on existing reference implementations.

4.2 Impacts in Non-Technical Contexts

4.2.1 Environmental

The environmental impact of the prototype is relatively small, as it primarily consists of a single camera module, microcontroller, and display hardware. However, we can minimize e-waste by selecting Restriction of Hazardous Substances (RoHS) compliant components when possible and designing for low power consumption to reduce energy usage.

Furthermore, by using a USB-C adaptor for charging, the design supports sustainability efforts through standardization. As of 2025, USB-C has been widely adopted as a universal connector standard, meaning that users can rely on existing chargers and cables rather than purchasing proprietary ones. This reduces the number of redundant cables and power bricks that eventually contribute to e-waste. In addition, the European Union has pushed for USB-C adoption, because it simplifies charging infrastructure, which can significantly cut down on the production of unnecessary electronic accessories. The EU also mandated all personal electronics that can be sold in Europe require USB-C. By designing around USB-C, our project aligns with these broader sustainability initiatives, decreasing both the material footprint of our system and the long-term environmental impact associated with electronic waste.

4.2.2 Public Health

This project does not directly impact public health, but it may have an indirect benefit by reducing stress or cognitive overload for users learning poker strategy. By assisting users in decision-making, it may help avoid unhealthy gambling habits through structured, educational feedback rather than encouraging impulsive betting.

4.2.3 Global, Cultural, and Societal

Poker is played worldwide and is culturally significant in many regions. The system encourages fair play by objectively evaluating game states and strategies, which can help standardize training for both casual and competitive players. Since poker has gambling associations, we must also consider ethical use - the system is designed for education and training, not real-time gambling assistance in casinos.

4.2.4 Diversity, Equity, and Inclusion

This system can be used as an educational tool that makes poker strategy training accessible to beginners, regardless of prior experience. The AI component allows equal access to strategic insights, which can lower barriers for new players from diverse backgrounds.

4.2.5 Welfare and Safety

There are minimal safety concerns since the system operates at low voltage and does not involve hazardous materials or moving parts. To ensure user safety, all wiring and hardware will be enclosed in a box with only the camera, ON button, and LCD display being exposed. The design will also comply with basic electrical safety standards for lab and classroom use.

4.2.6 Economic

Economically, this system could reduce the cost of learning poker strategy compared to expensive coaching sessions or training software subscriptions. If developed further, it could represent a low-cost educational tool for poker clubs or online training platforms, potentially contributing to growth in the poker educational market.

5. Conceptual Design

5.1 Hardware Concepts

5.1.1 Power System Design

Modern electronic devices have mostly integrated the adoption of USB-C, which offers a universal connection as most personal electronic hubs are USB-C. This makes USB-C an ideal choice for a device such as the poker prototype. The system must be able to do three things seamlessly:

1. Power the device load directly from an external USB-C source when plugged in
2. Charge the internal battery to ensure operation when unplugged to USB-C source
3. Run solely from the battery when no external USB-C source is available

To generalize the three points listed above, if there is a USB-C source, and there is also a load present, the battery board needs to be able to both power the loads directly from the USB-C source as well as charge the battery at the same time so when unplugged from the source, the battery can power the loads.

To evaluate the best approach, both a 1S USB-C powered system and a 2S USB-C powered system are considered below.

5.1.1.1 1S USB-C Powered System

A 1S battery system is an attractive choice due to its simplicity and cost-effectiveness. Since it uses only a single cell it eliminates the need for balancing circuitry, which reduces both design complexity and component count. This allows the charging function to be implemented with a single-cell charger IC, which is typically smaller and less expensive than multi-cell chargers. The overall system can therefore be made more compact, which is an important consideration for small personal electronic devices such as the poker prototype, where board area and height are limited.

However, the main limitation of a 1S design is its voltage range of 3-4.2V, which is always below the 5V rail required for powering the LCD screen. This requires a boost converter to step up the voltage, which means there will be more current drawing vs a buck converter. As the battery voltage approaches 3V under load, the boost converter must pull proportionally higher current to maintain 5V output. This stresses the single cell as well as places thermal demands on PCB traces and power components, which may require wider traces, thicker copper, or even heatsinking to ensure safe operation.

While a 1S system works well for small to medium-power devices, it is poorly suited for high-power applications, since the current demands on the single cell and power path become

excessive. In such cases, a multi-cell configuration is more practical, as it allows operation at higher voltages and lower currents, reducing conduction losses and improving overall efficiency. For this reason, 1S systems are ideal for low-to-moderate power use cases like consumer gadgets, portable medical devices, or low-power wireless modules, but are not typically used in high-performance systems requiring sustained tens of watts or more.

5.1.1.2 2S USB-C Powered System

A 2S battery system operates in the 6-8.4V range, compared to a 1S 3-4.2V. Since the system voltage is always above the common rails needed in most applications, such as 5V or 3.3V, only buck converters are required to step down the voltage. This generally results in a more efficient system, as lower currents are needed to deliver the same power compared to a 1S configuration. The reduced current draw improves overall thermal performance, minimizes conduction losses in PCB traces and components, and helps extend battery life by reducing stress on the cells. Another advantage of the higher system voltage is that with higher voltage and reduced current draw, the system can support more demanding applications, such as industrial handhelds, portable computing devices, or high-brightness LED drivers, without the same thermal or current-handling concerns as a 1S design.

However, the tradeoff is increasing complexity. Because it uses two cells, cell balancing is necessary to ensure both cells charge and discharge evenly. Without balancing, one cell can become overcharged while the other remains undercharged, leading to reduced usable capacity, overheating risks, and accelerated long-term degradation, typically making the system being capable of less than 200 charges. Cell balancing can be implemented with a dedicated balancer IC, such as the BQ29209, or with a more advanced charger IC that integrates balancing, like the BQ25887. The higher operating voltage also requires that all power components (FETs, inductors, capacitors, and protection devices) be rated appropriately, which can increase cost and board area. See below for the concept design block diagram of a 2S battery system for the poker prototype.

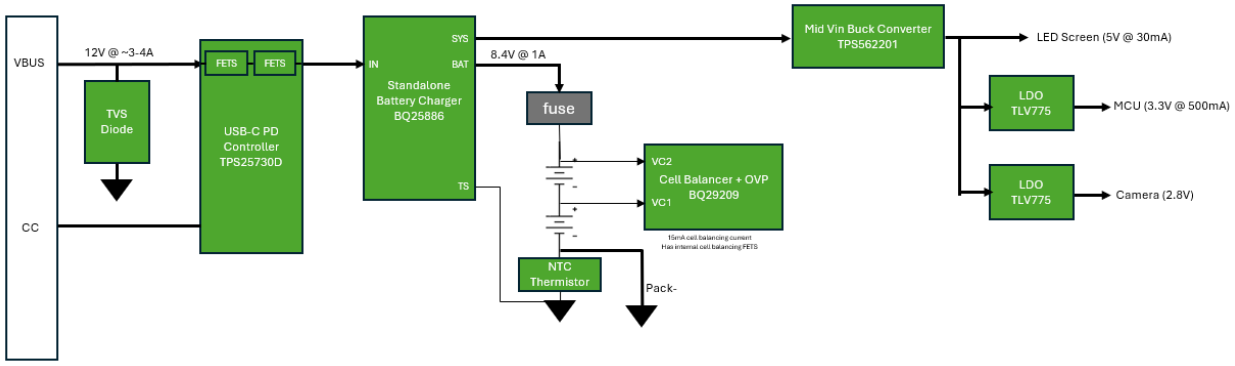

Figure 1: 2S Battery System

In general, while comparing both design choices, both systems look very similar, with the difference being a different battery charger that needs to be able to support charging a 2S vs a 1S, as well as the required cell balancer IC needed to balance multiple cells. Compared to a

simpler 1S solution, a 2S system is better suited for applications that demand more robust and efficient power delivery, making it a preferred choice when high power is necessary.

5.1.2 MCU Board Design

5.1.2.1 STM32N657 Chip

The STM32N657 microcontroller, which combines a dual-core Cortex-A35 application processor and a Cortex-M33 real-time processor in a single package. This architecture allows the system to dedicate one core to handling camera input, display updates, and other time-critical tasks, while the second core performs card recognition and strategy computation.

The STM32N657 also integrates a neural processing unit (NPU) capable of accelerating TensorFlow Lite models, enabling local inference with minimal latency. With clock speeds up to 1.2 GHz and support for up to 1 GB of external DDR memory, the system can process high-resolution frames efficiently. The chip's built-in MIPI-CSI interface supports direct connection to a CMOS camera module, while the integrated LCD controller drives an LCD display for real-time feedback.

All processing, from image capture to decision display, occurs within the same enclosure, making the system fully self-contained and independent of network connectivity. This results in low latency and a portable/robust form suitable for classroom or home use [8].

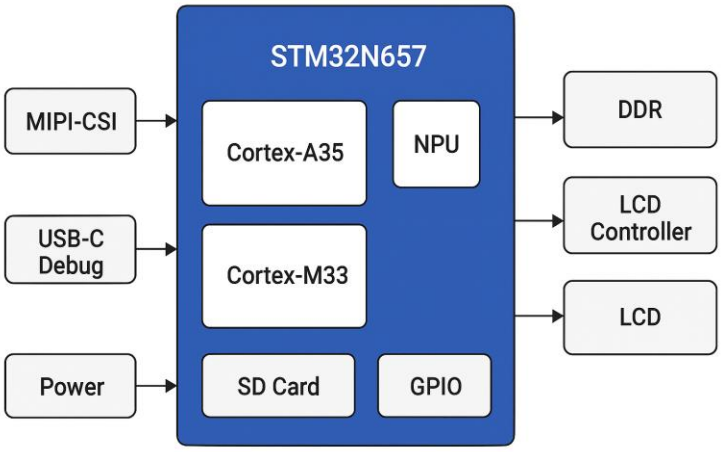

Figure 2: STM32 System Block Diagram

5.1.2.2 ESP32-S3-WROOM-1 Chip

Using an ESP32-S3-WROOM-1 module as the front-end capture and communication node. The ESP32-S3 is a dual core Xtensa LX7 processor running up to 240 MHz, well-suited for handling image acquisition and lightweight preprocessing tasks such as cropping, resizing, and image compression. Although it includes basic vector instructions and AI acceleration support, its limited 512 KB of SRAM and external flash storage make it better suited for transmitting preprocessed frames rather than running a full neural network locally.

The ESP32 captures frames from the camera and sends them via its integrated 2.4 GHz Wi-Fi interface to a local cloud server, where a large and more computationally intensive neural network model performs card recognition and strategy generation. The resulting recommendation is transmitted back to the ESP32, which then drives the LCD display.

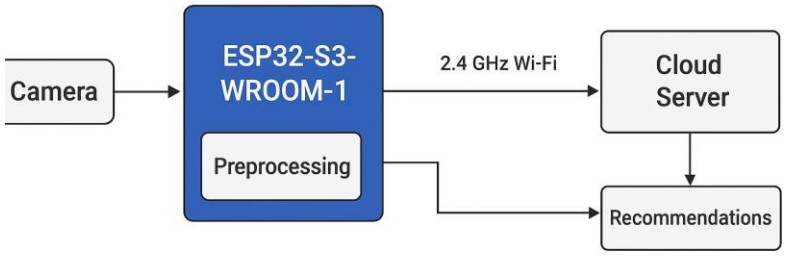

Figure 3: ESP32 System Block Diagram

This approach enables the use of highly accurate, resource-intensive models and allows for easy retraining and updates, since the primary compute node is external. However it introduces additional latency and depends on network availability, which can be a risk during demonstrations or in environments with unstable connectivity [9].

5.2 Software Concepts

This section describes our potential Software methodologies that we have explored for our prototype.

- Software:
  - Microsoft Azure Cloud Services
    - Allows for much greater storage capacity & processing power
  - API Protocols
    - Build API protocols for interacting between the device itself and the cloud
- Machine Learning/AI:
  - Image Processing Algorithm
    - Developing algorithm(s) to detect objects and identify specific features in a photo/video
  - Poker Agent Algorithm

- Developing a machine learning model to use given context to make a prediction

5.2.1 Cloud Services & Communication

The system requires robust communication between the embedded device and external computer resources. Cloud services such as Microsoft Azure provide the ability to offload compute-intensive tasks that cannot be executed on the ESP32-S3 due to its limited memory and processing capacity. By integrating Azure-based APIs, the device can transmit cropped image payloads to the cloud for advanced image recognition and poker decision-making.

In this Architecture, the ESP32-S3 handles lightweight preprocessing (cropping and compressing images, formatting into JSON) before transmitting them via HTTPS POST requests. Once in the cloud, the images are processed by containerized algorithms (Python/OpenCV/ML Models). This event-driven architecture can be implemented with Azure Blob Storage triggers to automatically invoke seamless serverless functions for processing [10]. The results are returned as structured JSON responses containing both the recognized state (cards, chips, pot size) and the recommended action (Fold, Call, Raise).

The advantages of this approach include:

- Scalability: Larger ML models and more accurate algorithms can be deployed without being limited by on-device resources.
- Reliability: Cloud-based APIs provide redundancy, fault tolerance, and security compliance.
- Flexibility: Models can be retrained, updated, or swapped without requiring hardware changes.
- Debugging/Analysis: Metadata can be optionally stored in temporary storage for debugging or replaying scenarios.

This design ensures that latency remains within the <5 second requirement. Prior research on ESP-32-based edge-cloud systems also demonstrates how cloud offloading improves model accuracy while maintaining real-time performance [11].

5.2.1.1 No Cloud Services (Local-Only Deployment)

An alternative approach to cloud integration is a local-only deployment, where both the image recognition pipeline and the poker decision model run solely on the ESP32-S3. This eliminates external dependencies and ensures the system operates without network connectivity.

Benefits Include:

- Minimal latency: No network transmission delays.
- Self-Contained Operation: Fully portable and independent of external infrastructure.
- Robustness: Immune to connectivity failures during demonstrations.

However, the drawbacks are significant:

- Model Complexity Constraints: Only lightweight algorithms can be supported due to limited MCU memory and computational power.
- Reduced Accuracy: Advanced CNNs or reinforcement learning agents cannot be fully implemented.
- Development Burden: Considerable effort required to compress and optimize algorithms for embedded execution.

While a local-only deployment is feasible for simple image recognition and basic strategy output, it is not scalable for production-grade usage. Such constraints align with findings in TinyML research, where on-device ML must trade accuracy for portability due to limited memory and computational resources. [12]. This option is best suited for fallback demonstrations scenarios or environments where cloud access is unavailable.

5.2.2 Image Processing Algorithms

Image Processing is a critical aspect of our prototype, as we are dependent on reading in visual data for our coach to make decisions. Ensuring that we choose the right algorithms is critical to the success of our prototype, so in this section we will explore a variety of solutions. The key requirement of our Image Processing Algorithm is to input visual data into our software for the coach to analyze and make decisions off of. There are two main aspects we need to consider: the limitations of the device's memory, and the limitations of the device's computing power. These two limitations serve as a major bottleneck for running heavier algorithms, so we explored some potential solutions for dealing with the limited memory and computing power using a cloud service provider. This would require two separate algorithms for image processing, one being a local algorithm on the device itself, and then a heavier, more computing-intensive algorithm that would be in the cloud. We also explored a sole algorithm on the device to see how feasible it would be for our prototype.

5.2.2.1 C++/Arduino Image Processing Algorithm

In scenarios where no cloud connectivity is available, the system must rely entirely on local image processing and decision-making performed on the ESP32-S3 microcontroller. This approach removes all external dependencies and ensures the device operates as a self-contained poker assistant.

Operational Flow:

1. Frame Capture: The OV5640 camera streams raw frames to the ESP32-S3 over the parallel DVP interface.
2. Preprocessing: The microcontroller applies lightweight image filtering, grayscale conversion, and binarization to reduce noise and simplify features
3. Region-of-Interest (ROI) Detection: Predefined pixel zones corresponding to player hole cards, community cards, and chip stacks are scanned.

4. Local Classification: Template-matching and simplified convolution-style kernels are applied to identify rank/suit features of the cards. Chip counts are estimated through contour detection and object segmentation.

Advantages:

- Operates independently of network connectivity.
- Provides deterministic, low-latency feedback (<2 seconds).
- Fully portable and robust against external service interruptions.

Limitations:

- The ESP32-S3's limited SRAM and computational power constrains the complexity of the Image Processing algorithms, reducing recognition accuracy compared to cloud-based pipelines.
- Card and chip recognition are more sensitive to lighting, occlusion, and camera angle due to the absence of deep-learning-based processing.
- Updating or retraining models requires firmware re-flashing, limited long-term scalability.

This design option emphasizes self-sufficiency and portability ensuring that the prototype remains functional in demonstration environments or offline scenarios, even though accuracy and adaptability are reduced compared to cloud-enabled operation.

5.2.3 Machine Learning/AI Algorithms

Being able to train a machine learning model to make systematically good predictions in such a complex game as poker can be extremely challenging especially with how much variability is present in the game. This makes it vital to use the most optimal algorithm for training our model.

5.2.3.1 Multilayer Perceptron

The Multilayer Perceptron is a neural network that utilizes multiple layers of neurons to learn from the data as well as make predictions on the same dataset. This type of model can train extremely fast compared to other models due to fast iterations of the data. The model can also be optimized efficiently to be even faster by adjusting the amount of neuron layers. The input layer of neurons for the MLP also matches how information from a game of poker would be recorded in a tabular format with sections for each aspect such as, position, stack, pot, cards, etc. This makes the model easy to train with poker data sets due to its comfortability with the type of data being fed into it [13].

The drawback to this model is that it will never surpass the data it sees. Which means that this type of model imitates the data it sees but can never provide predictions that may be stronger than that data alone. Unfortunately, this creates a ceiling for how talented the model can be and puts a lot of pressure on the data to be able to provide the model with the best information available. To use this algorithm effectively the first thing that must be done is to find and save a massive amount of data (thousands of poker hands) and save them in a consistent format. Once the data set has been acquired it is fed into the MLP's first neuron layer where the model begins

to analyze the data. The training of the model may take a while as it processes all of it, but once it is done the model will be saved and ready to deploy.

Although the model may take a long time to train once it is finished it will be able to give fast accurate predictions in real-time, which is exactly what the overall design requires. The finished model will be able to take in data from a poker game in real time and, using the similar characteristics between the real game and the data, output a strong action for what the user should do in their circumstance.

5.2.3.2 Reinforcement Learning Model

The other model that should be considered in designing the real-time poker coach is the Reinforcement Learning Model. This model trains through trial and error growing smarter and more accurate over time. The advantage it has over the MLP model is that it can always be improved. Where the MLP may have a ceiling, the RL can always learn more and be improved upon.

The overall idea of this model is that a reward condition is defined within the algorithm which is usually some type of feedback and based on that reward system the model continues to learn and attempt to get positive feedback. For example, in our instance the RL model could continuously play poker hands with the goal of winning fifty percent or more of the hands and whenever the model plays five rounds it is rewarded if it reaches the goal and not rewarded if it does not. Over thousands of iterations the model will slowly improve at poker to be able to get the reward as often as possible which will in turn create a model that has an extremely high win rate.

The main drawback to this model is the computational power it needs to be successful. For this model to be accurate it must run thousands to millions of poker rounds until it can consistently win games of poker and to do that, if our technology is not strong, could take days-weeks of time which is not available. So, although this machine learning technique has the potential to be extremely accurate the power needed to run it may not be available. To implement this model into the design first an environment must be created for the model to play the poker rounds. This can be created using pythons built in poker library which can simulate an infinite number of rounds. The next step would be to have the RL model play the bots within this library until it successfully wins a certain threshold of hands. Then the partially trained model could be placed into other poker simulators online to further train and become stronger. Once the model is up to standard it will be saved and deployed [14].

The positive is that once the model is trained and saved it can make predictions extremely fast with little latency. This makes the Multilayer Perceptron and the Reinforcement Learning model easy to integrate with our hardware due to the lack of latency.

5.3 Selected Design Concept

5.3.1 Selected Power Board Design Concepts

The selected concept moving forward is a 1-cell USB-C powered system, chosen for its simplicity and size compactness for the application's power needs. A single lithium-ion cell (minimum 3V, nominal 3.7V, 4.2V max) eliminates the need for a dedicated cell balancer IC and reduces protection circuitry requirements, since imbalance between multiple cells is not a concern. The reduction in system complexity directly translates to lower BOM cost, and smaller PCB layout size. Another big reason for the selected concept is that the system does not require high power delivery, a 1S approach is entirely sufficient. See below for the system functional block diagram of the system.

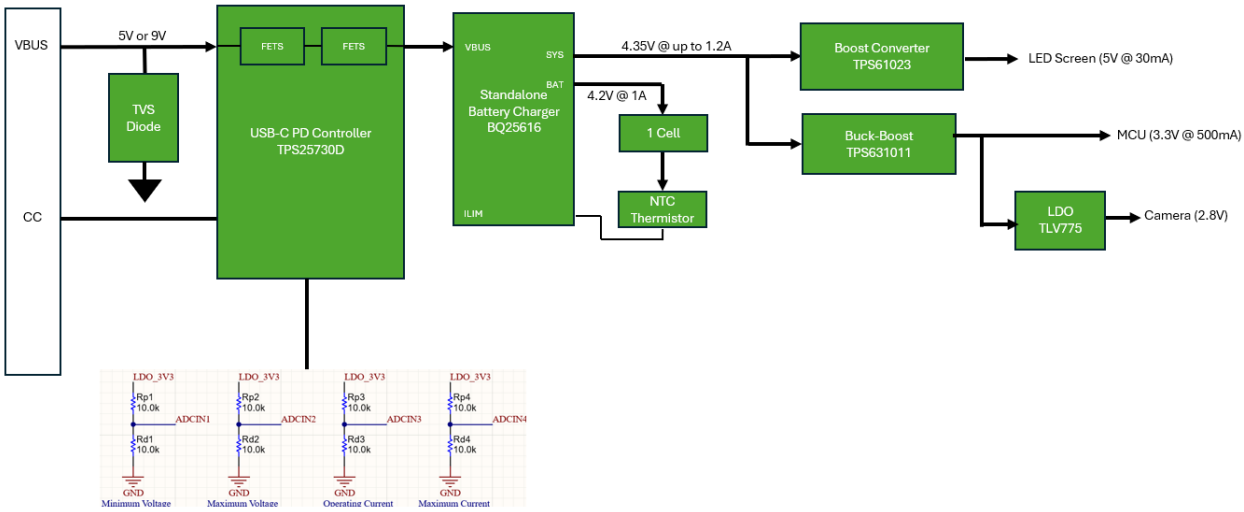

Figure 4: System Level Block Diagram

*Voltage Divider for USB-C Controllers are not exact, for image representation purposes

The combined loads of the ESP32, OV5240 Camera, and LCD screen only consume 2.1W when powered on, which is within the range that a single cell can support. The work for calculating power is listed below, and power requirements were pulled from the respective datasheets.

- Load Power Calculations

$$P_{LOAD} = P_{LCD} + P_{ESP32} + P_{OV5640}$$

$$P_{load} = (5V \times 0.03A) + (3.3V \times 0.5A) + (2.8V \times 0.104A) = 2.1W$$

- Load + Battery Charge Calculations

$$P_{TOTAL} = P_{BAT} + P_{LOAD}$$

$$P_{TOTAL} = (4.2V \times 1.5A) + 2.1W = 8.4W$$

Due to the overall system not requiring support for high power, the advantages of a 2S design, such as reducing current high-power systems, are not necessary in this application.

Because the battery charger being used cannot negotiate power with a USB-C source, the system requires a dedicated “negotiator” to handle this function, which is fulfilled by the TPS25730D sink-only USB-C CC/PD controller. Currently in the industry, there are few (if any) battery chargers that can directly negotiate power over USB-C, hence the need for an external power negotiator. The TPS25730D is an ideal fit because it requires no firmware development, with voltage and current contracts set by resistor dividers on its ADCIN pins [18]. This hardware-based configuration eliminates the need for an external MCU to manage the negotiation process. The TPS25730D always guarantees a minimum of 5V, since this is the default voltage of all USB-C sources. If the connected source supports higher PD voltages, the controller can be configured to request them automatically. Since the system operates at approximately 10W, it can be powered by 5V at 2A or, when available or by 9V at about 1.1A. In this design, the TPS25730D will be configured through its ADCIN pins to request 5V as the default operating voltage, with the option to negotiate up to 9V when supported. It can also support up to 5A, which is more than sufficient to cover the system load while simultaneously charging the battery if the device is being used at the same time.

Another method for negotiation is to operate as a simple USB-C sink using only 5.1k pull-down resistors on the CC lines to advertise 5V/9V sink capability. While much simpler to implement, this approach provides no protection, and usually an external power path with FETs to safely connect VBUS to the system. In contrast, the TPS25730D integrates its own power path and includes protection features such as overvoltage protection, undervoltage monitoring, overcurrent protection, reverse current blocking, short-circuit protection, and thermal shutdown. Refer below for a simplified schematic of the IC.

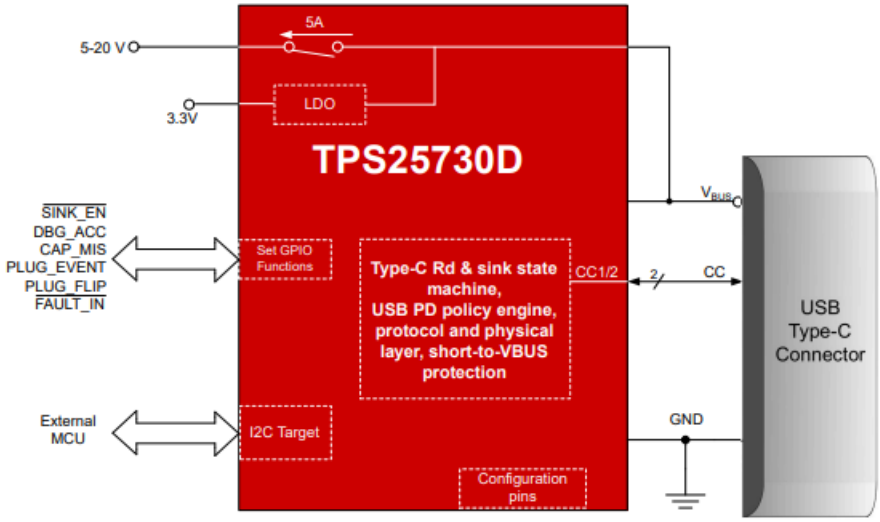

Figure 5: TPS25730D PD Controller Simplified Schematic

This integration reduces component count, lowers design complexity, and results in a robust and reliable system. Furthermore, because the TPS25730D is the first IC encountered by the input supply, its integrated protection features are important for protecting all downstream components. Alternative controllers such as the Infineon CYPD3177 [21] or the ADI

MAX77958 [22] provide similar functionality and could be substituted with minimal design changes, as they also support hardware-based configuration without firmware.

Once the PD controller has negotiated and passed the regulated voltage input downstream, charging of the single lithium-ion cell is managed by the BQ25616 standalone battery charger [23]. The reason for choosing a standalone battery charger is due to advantages in BOM cost as an external MCU is not needed to set charging parameters.

The BQ25616 operates using a constant-current/constant-voltage (CC/CV) profile. During the constant-current phase, the charger regulates charging current up to the programmed 1.5A planned to be used until the cell voltage rises to 4.2V. At that point, the charger transitions into constant-voltage mode, holding the battery at 4.2V while gradually tapering the current until termination. This CC/CV behavior ensures safe charging and extends the life of lithium-ion battery. Figure 9-5 in the BQ25616 datasheet shows this battery charging profile.

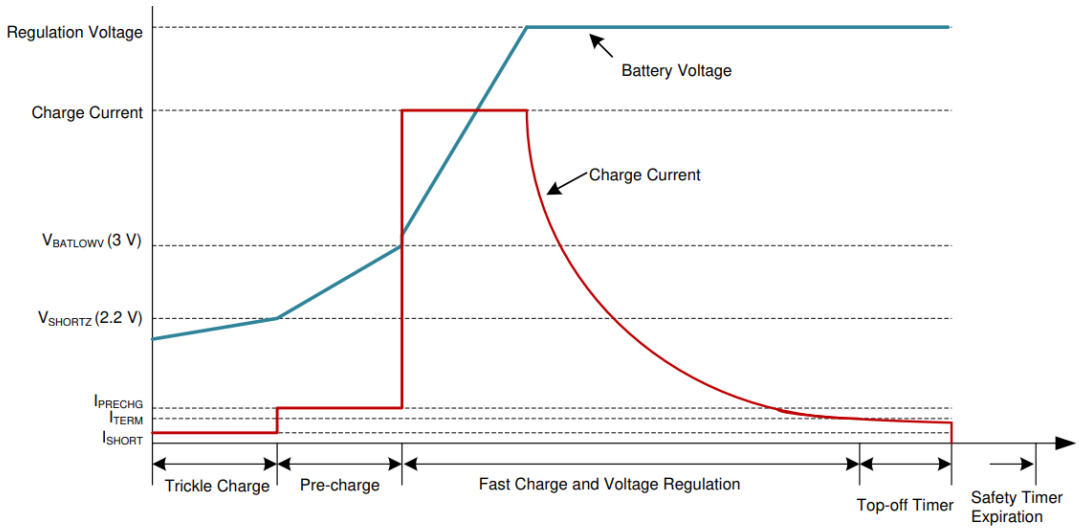

Figure 6: Battery Charging Profile

The BQ25616 also has a temperature sensing feature, where an NTC thermistor tied to BQ25616 provides real-time thermal feedback, stopping charging automatically if unsafe temperatures are detected. Another big benefit of the BQ25616 is that it has integrated BQ Power Path feature, meaning that the BQ can switch from the battery voltage or the source VBUS voltage, depending on if the source is plugged in or not. This feature description can be found in section 9.3.2 and section 9.3.3, which illustrates power up from either the battery or the input source. This integrated feature eliminates the need for an external multiplexer like the TPS2121, which helps reduce BOM costs and board space. Furthermore, with the power path feature, there is internal protection when switching between power source vs battery. This integrates the need for an external power FET. Refer below for the simplified application diagram for the BQ25616, which illustrates all the usable pins.

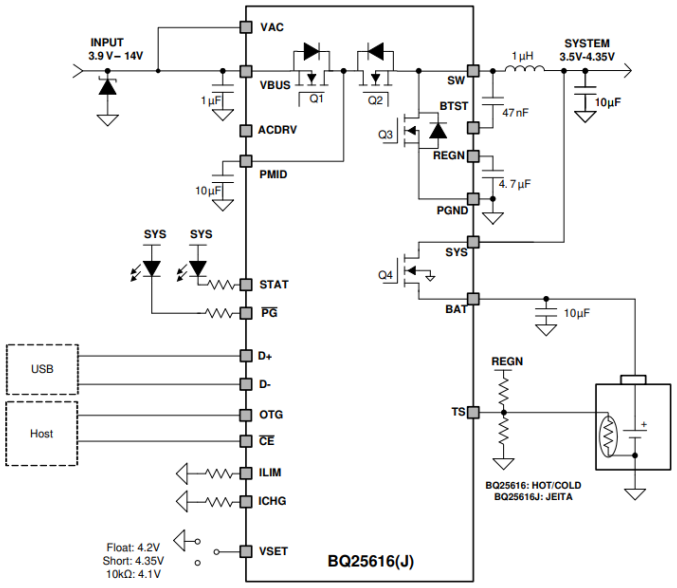

Figure 7: BQ25616 Typical Application Simplified Diagram

Finally, the regulated battery rail is converted into the required operating voltages for the system's peripherals using point-of-load converters. The LCD screen requires 5V at approximately 30 mA, which is supplied by a boost converter (TPS61023) stepping up directly from VSYS of the BQ25616 pin. According to datasheets, the ESP32 MCU requires a stable 3.3 V rail at roughly 500 mA, which is provided by a buck-boost regulator (TPS631011) that maintains regulation as the battery voltage swings anywhere between 3 and 4.2V. The reason for this choice of a buck-boost regulator vs an LDO is due to the higher efficiency. From this 3.3 V rail, a LDO regulator (TLV775) generates the 2.8V supply for the camera, which only draws 100–200 mA according to datasheets. These peripheral requirements were initially derived from datasheet specifications but will be validated with direct current measurements on the hardware prototype to refine the power budget. This converter arrangement maximizes system efficiency by using switch-mode regulators for significant voltage changes while reserving LDOs for low-current rails, ensuring each load is powered reliably without excessive power loss.

5.3.2 Selected MCU Board Design

The selected design concept for the MCU is the ESP32-S3-WROOM-1. This choice is driven primarily by development time, computational requirements, and budget constraints. While the STM32N657 standalone system offers a self-contained solution with low latency, it would require purchasing a higher-end MCU with external memory and a more complex board design, which would consume a large portion of the budget and leave limited funds for the camera, display, and other necessary components.

The ESP32, by contrast, is cost-effective, easy to source, and supports rapid prototyping. By offloading the heavy computation to a cloud server, we can use large, pre-trained models with higher accuracy and avoid the extensive model compression work required for embedded-only deployment. This significantly reduces complexity and risk while still meeting real-time requirements for card recognition and decision-making. Additionally, this setup allows future

scalability, as the server-side model can be retrained or upgraded without requiring hardware changes to the device.

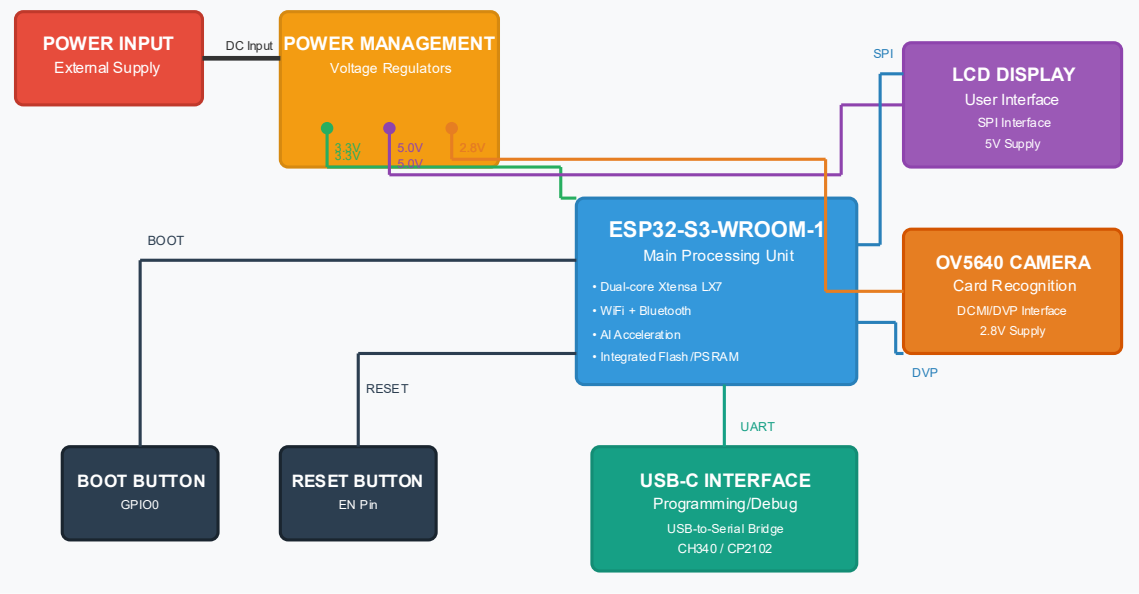

Figure 8: Selected Board Design

The board will include the ESP32-S3-WROOM-1 that will serve as the main processing unit. Along with a muti-rail power distribution system to accommodate the varying voltage requirements of the different components. There will a 3.3V terminal for the ESP32 module, 5.0V terminal for the LCD display, and 2.8V terminal for the OV5640 Camera module. This design ensures optimal power efficiency while maintaining signal integrity across different voltage domains.

The OV5640 camera module provides 5-megapixel image capture capability with autofocus functionality. The camera interfaces with the ESP32 through a Digital Video Port (DVP) connection, enabling high-seed image data transfer for real-time card recognition processing.

The LCD display serves as the primary user interface, connected via SPI protocol for efficient data transmission.

A USB-C connector provides programming and debugging capabilities through an integrated USB-to-serial bridge circuit. This interface supports firmware updates, system diagnostics, and development activities without requiring external programming hardware.

Two switches will provide essential system control functions. The boot button connects to GPIO0, enabling firmware update mode entry, while the reset buttons connect to the enable pin for system restart operations.

5.3.3 Cloud Services & Communication

A key aspect of our device will be the integration of Cloud services and storage. Leveraging a cloud provider such as Microsoft Azure allows us to overcome the device's memory and computation limitations by offloading heavier algorithms to the cloud. This ensures scalability,

higher throughput, and the ability to integrate AI/ML models without being constrained by embedded resources.

Additional advantages of cloud integration:

- Persistent Metadata storage: Game logs and player histories can be stored for long-term analysis. This enables statistical insights into player behavior and more informed decision-making.
- Wireless Communication: Eliminates the need for a physical wired connection to a laptop, improving usability.
- AI Model Hosting: Azure providers APIs and container services for running models reliably.
- Reliability & Security: Cloud providers offer built-in backup, failover, and security compliance measures.

Communication will be established via HTTP(S) REST APIs:

- Device will send cropped JPEGs or JSON payloads to API endpoint.
- Cloud: Processes with Python algorithms, returns JSON response
    - Example Scheme: {"action":"RAISE", "cards":["Ah, "10d", "7c"]}

If higher throughput is needed, extensions such as WebSockets or MQTT can be introduced.

5.3.4 Temporary Storage (CPU + Short-lived images)

To balance performance, cost, and system reliability, our architecture will use temporary object storage such as Azure Blob Storage as the intermediary between the device and cloud compute. Images and metadata are stored only briefly, enabling flexible event-driven processing while avoiding the overhead of persistent storage.

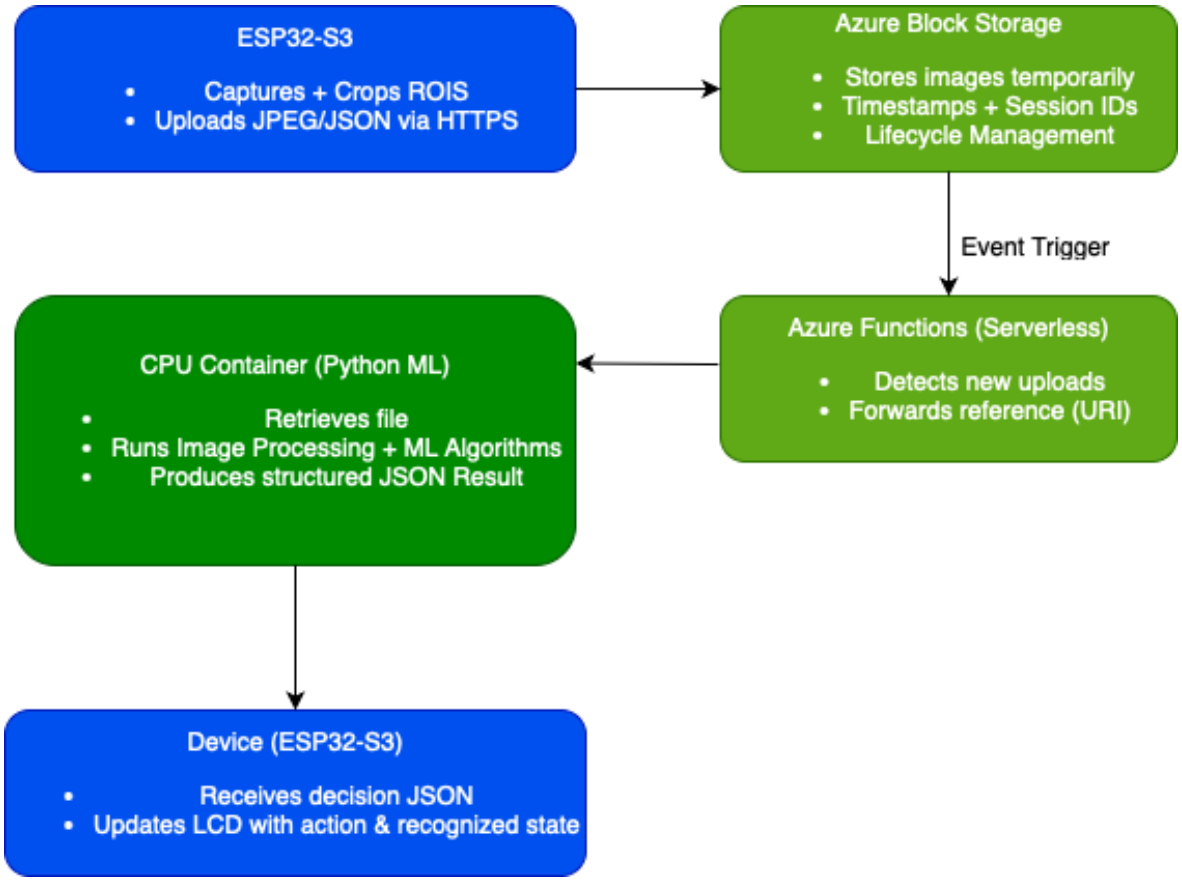

Figure 9: Cloud-Service Architecture Diagram

Operational Flow:

1. Device Upload:
    a. The device captures and crops regions of interest (ROIs)
    b. Cropped JPEGs or lightweight JSON payloads are uploaded to Blob Storage via HTTP(S)
    c. Each upload is timestamped and associated with a game/session ID.
2. Event Trigger:
    a. Blob storage is configured with event notifications.
    b. When a new file arrives, it triggers a serverless function such as Azure Functions
    c. This function forwards the reference (URI) to a CPU container for processing.
3. Model Execution:
    a. The container retrieves the image from Blob Storage.
    b. Python algorithms for Image Processing and Machine Learning/AI will run on the CPU.

c. The container outputs a structured JSON either of the image processing results or decision outputs.

4. Result Delivery:
    a. The decision JSON is returned to the device via REST API.
5. Lifecycle Management:
    a. Images are retained only for a short period of roughly 24-48 hours using lifecycle rules.
    b. Automatic deletion ensures cost efficiency, privacy, and compliance.

Advantages:

- Reliability: Event-driven triggers ensure no request is dropped; failed jobs can be reprocessed by retrying the storage reference.
- Debugging / Replay: Temporarily stored images allow failed actions to be re-run without needing new input from the device.
- Affordability: CPU-only containers and auto-deletion policies keep costs predictable, as well as $100 in credits for students.
- Security & Compliance: Temporary storage with enforced expiration minimizes the risk of sensitive game data persisting unnecessarily.

Drawbacks / Mitigations:

- Added Latency: The storage-trigger-container pipeline adds a small delay compared to direct streaming.
    - Mitigation: Keep blob sizes small (cropped ROIs, JPEG compression) and functions lightweight.
- Complexity: More moving parts (storage, event trigger, container).
    - Mitigation: Infrastructure-as-code ensures reproducible setup and automated monitoring
- Storage Costs: Even short-lived files incur minimal storage charges.
    - Mitigation: Lifecycle rules + compression keep usage negligible.

5.3.5 Image Processing Algorithms

The selected design concept for image processing adopts a hybrid approach that combines lightweight on-device preprocessing with more robust recognition and decision-making in the cloud. This balances the ESP32-S3's hardware constraints with the accuracy and scalability of cloud-based ML pipelines.

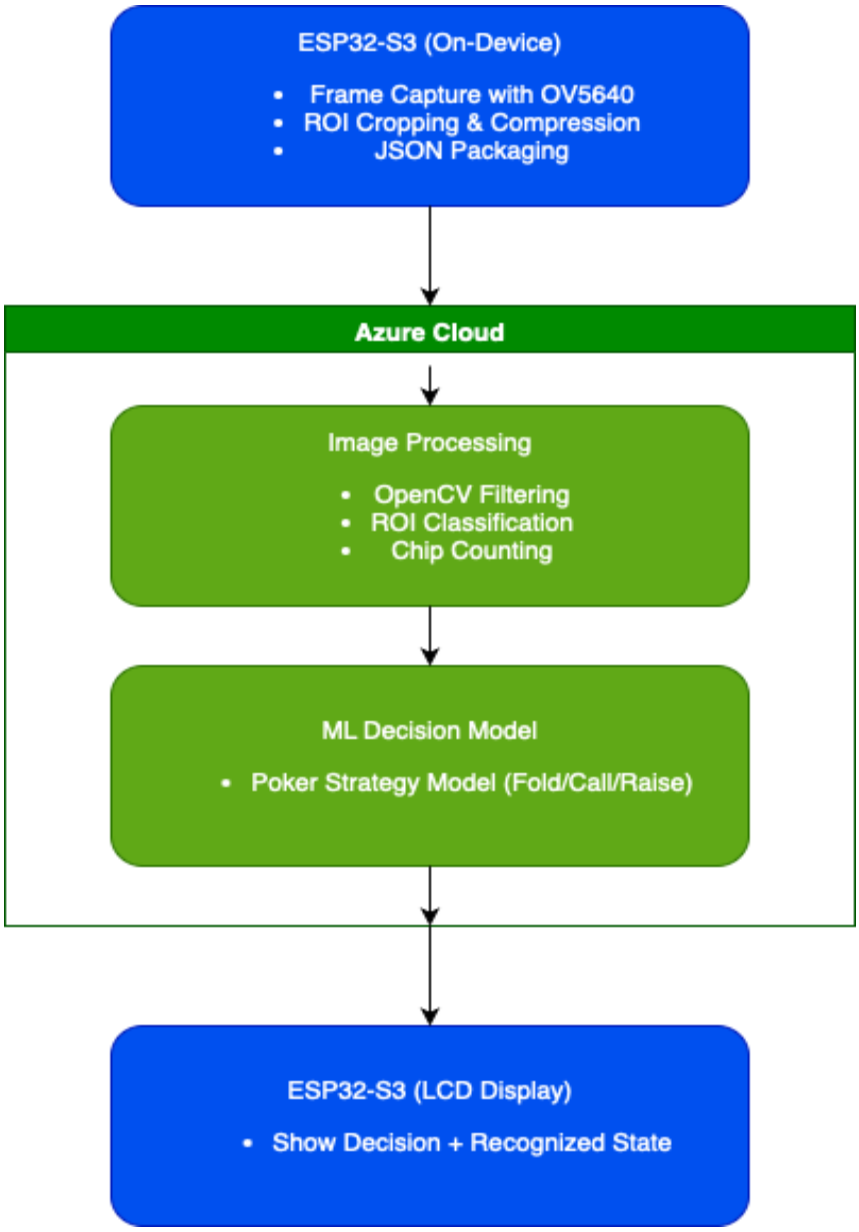

Figure 10: Image Processing Architecture Diagram

Hybrid Design Workflow:

- The ESP32-S3 captures raw frames from the OV5640 camera.
- A C++/Arduino preprocessing algorithm identifies regions of interest (player zones and events that occur within these zones, and blinds), crops the frame, and compresses the sub-images.
- Cropped data is formatted into JSON payloads and transmitted securely to Microsoft Azure via HTTPS POST requests.
- Cloud-based Python/OpenCV/ML algorithms perform high-accuracy recognition of card rank/suit, chip amounts, and betting actions

- Once identified, processed data would be passed onto the ML model for further decisions.

Rationale for Hybrid Selection:

- Bandwidth Optimization: On-device preprocessing reduces payload size, ensuring efficient transmission.
- Accuracy: Cloud-hosted CNNs and object detection pipelines deliver recognition rates exceeding 99%, outperforming local-only approaches.
- Scalability: Models can be retrained, updated, or swapped in the cloud without requiring firmware reflashing on the device.
- Latency Control: Preprocessing on-device ensures that the end-to-end system latency remains under 5 second requirement.

Advantages of Hybrid Approach:

- Balances efficiency and recognition accuracy.
- Leverages the ESP32-S3 for lightweight tasks while offloading complex algorithms to the cloud.
- Provides a flexible architecture that can evolve as models improve.

Limitations:

- Requires stable internet connectivity for full functionality.
- Slightly higher latency than local-only execution due to network transmission.

Overall, the hybrid design ensures that the Poker Assistant Coach prototype achieves robust recognition accuracy and real-time decision-making, while retaining enough flexibility to function in both connected and offline scenarios. This makes it the most practical and reliable image processing solution.

5.3.5.1 C++/Arduino Image Detection/Cropping Algorithm

To compliment the cloud services if used, we would develop low level algorithm for identifying and then cropping items of interest to be sent to the cloud for further analysis. The reason for this low-level algorithm is to provide a low memory & efficient solution to decrease the size of data being sent to the cloud. In short, by using this algorithm, we can improve the performance of our system as well as minimize latency between communicating with the device and cloud. The basis of the algorithm is as follows:

- Identify the initial assets of the player: Two cards delt, amount of chips left
- Have set points on the board/playing field that the algorithm will identify the following actions: Check, Raise, or Fold

    - In case in a raise, the algorithm will adjust the scope of its view accordingly (so if 3 players are in a game, and player 2 raises, it will account for the update and ensure to review player 1s action, and not ignore it)
- Once an action is identified, the action will be sent to the cloud for further analysis as well as adjusting the scope of the cameras view to focus on the next area of interest
- This component will also be responsible for reading the community cards

So, to describe how the algorithm works, it would use a pixel-cropping looping algorithm. This algorithm would essentially scan a specific section of pixels, in which we would have 5 main zones:

- AI Assisted Player Zone
- Player Action Zones (3 Opposing Players)
- Community Card Zone

Each specific zone would be scanned for actions and send the outcomes to be analyzed by the Python algorithms within the cloud. The reasoning behind this design choice is to help save bandwidth and the computation power needed both on the device. This algorithm would also be responsible for utilizing the MCU resources and communication with the cloud.

5.3.5.2 Python Card Recognition Algorithm

To supplement the initial image processing & cropping algorithms within the device, we will include a Python program to help analyze and identify the different pieces of data inputted through the device. We would be using a few libraries:

- OpenCV [15]:
    - Industry Standard Library, provides powerful computer vision library for real-time image and video processing, offering tools for filtering, feature detection, motion tracking, and object recognition
- Scikit-learn [16]:
    - A lightweight machine learning library that is focused on classical algorithms such as SVMs, decision trees, clustering, and allows for easy model training and evaluation
- Mediapipe [17]:
    - A Google framework specialized for fast, pre-built pipelines for features such as hands, poses, and face detection, optimized for real-time applications
- Supplemental: Pytorch [18] /TensorFlow [19]:
    - Deep learning frameworks that can be used to build and train neural networks, with Tensor flow emphasizing production/deployment and PyTorch prioritizing flexibility and research.

- May be overkill for our prototype, but if time to refactor allows, could be implemented

Ultimately, the use of these libraries would allow us to piece together the different pieces of the inputted data, with OpenCV's primary usage being to provide feature-detection, motion detection, and object recognition. Scikit-learns primary usage would be to supplement OpenCV with object recognition, helping extend the minimal functionality of OpenCV to cover the specific elements relating to Poker for our prototype. It would help detect the different cards, chips, amount of chips, as well as potentially hand gestures. With the hand gestures, we could use Mediapipe to ensure that we can detect specific gestures and not detect a stray hand as checking. Some potential edge cases that could be resolved from this are moving chips/cards into the action zone (False-Positive), moving items out of the action zone (False-Positive), and ensuring checks (True-Positive). Finally, if there is time to refactor and refine our program and prototype, we could potentially implement features with Pytorch and TensorFlow. The benefits that Pytorch and TensorFlow provides a deep learning framework in which we could refine the process of our image processing to handle more complex tasks and coverage of our image processing algorithms. I do not think Pytorch and TensorFlow are essential to the implementation.

5.3.6 Machine Learning Algorithm

The overall design for the machine learning algorithm we chose was the Multilayer Perceptron. This was due to a plethora of reasons. One reason was the speed and simplicity of the model. The MLP is fast to train with the available technology and easy to deploy onto our hardware. This is extremely important because we have limited space and power so for our design less is better without sacrificing results. This leads to another aspect of the MLP which fits perfectly, the ability to integrate cleanly with the poker features delivered to the board from the computer vision aspect of the project. The information from the game will be delivered in a format that consists of different aspects of each round such as chips, bet, and stack, which is exactly how a Multilayer Perceptron prefers to receive input. This will lead to an easy integration between the model and the computer vision in the project.

5.3.6.1 Implementing the Model

To design this model, we will use PyTorch which is one of the main python libraries for machine learning along with TensorFlow. We believe PyTorch is slightly more efficient and user friendly, so we decided to move forward with this library instead of tensor which has a higher learning curve. The first thing that must happen in the model is data acquisition. The model will receive the data and convert it into a numeric vector. It will pass the vector through its numerous neuron layers to eventually predict the action as well as the bet size if the chosen choice is to raise. As the vector travels through the layers of the model it will be analyzed based on the many characteristics available such as what part of the round it is, how much the bet is, and what cards the player has. Using this information, it will be able to learn from past data sets that had similar scenarios and further update its ability to make predictions. The outcome of the round will be used to determine whether the actions done in the data were optimal or suboptimal.

Along with the data being passed through the neural network we will utilize different types of hand analysis known to be applied successfully in previous poker applications. One method we

will use to calculate the winning chances of any certain hand within the model is the Monte Carlo method. This algorithm is implemented by running many simulations of different hands for the players opponent to try and predict the outcome of the round. The overall win percentage in the simulations is averaged and if the percentage meets a certain threshold set by our model, then the algorithm bases its decision off that as well as previous data from past games being fed into the model. The downside to this is that running the simulations makes the model slightly more computationally intensive, but because the MLP is already a fast-training model the tradeoff will be worth it.

Another aspect of the model will be a preflop hand evaluator that can judge the likelihood of winning based off the starting two cards of the user. There is a well-known python library that does this already called treys and using this library the model will be able to make extremely informed decisions in the preflop position of the game. The combination of learning and training from data collected from pro games, Monte Carlo simulations, and the hand evaluator will all come together to create an extremely accurate, fast, and reliable model that the user can utilize to get strong predictions in any step of the game without having to worry that they may be incorrect.

5.3.6.2 Testing and Deploying the Model

To make sure the model is trained accurately we will employ an eighty/twenty split on it which means we will train the model on around eighty percent of the total data and have it make predictions on the last twenty percent without being able to see the correct results until it makes a prediction. Not only will this let us determine how accurate the model is, but it will help train the model to fix its mistakes moving forward. To visualize the accuracy of the model we will generate a confusion matrix of predictions that will show the correct action and how the model decided to play. This will give us insightful information on where the model may or may not need more work in the future.

Once the model is fully tested and the group is satisfied, we will move on to deployment. This model will be saved as a .pt file that takes in the tabular data provided from the computer vision module. This .pt file will live in the cloud where the data is being stored by the computer vision. Once it takes in a set of data from a specific round it will be able to make a prediction in less than a millisecond and transmit it to the hardware application that displays the prediction to the user. The prediction will consist of the action the model suggests, how much the user should bet, the confidence level of the model, and potentially the equity of the hand.

5.3.7 Poker Physical Case

The final prototype will be housed in a dedicated enclosure designed to both protect the internal components and ensure the overall system is presentable for demonstration. The case will provide a professional appearance so that the project resembles a finished product rather than a collection of parts.

The enclosure will incorporate air gaps and venting to prevent heat buildup from the microcontroller, camera, and display. This passive cooling approach ensures that all components can operate within safe temperature ranges throughout extended use.

The enclosure will also provide secure mounting points and cable routing so that all subsystems-the camera, microcontroller, LCD screen, and battery board are contained in an organized and stable manner. This protects the electronics from accidental damage while maintaining accessibility for testing, maintenance, or future modifications.

Overall, the physical case will serve as both a functional housing and a visual presentation layer, balancing aesthetics, durability, and practicality for the final demonstration and presentation.

5.4 Sustainability Considerations

5.4.1 Minimizing Power Consumption

Minimizing power consumption during operation is essential. The ESP32 microcontroller is known for its low power requirements, especially in sleep modes. By optimizing our algorithms for energy efficiency, we can further reduce the active time of the chip, ensuring that image processing and communication are performed with minimal energy expenditure. Furthermore, with the design of the power system, selecting industrial ICs with high efficiency would be crucial for minimizing power consumption even further.

Beyond the microcontroller, the battery board itself was designed with efficiency as a top priority, ensuring that as little power as possible is lost in the system. This was done by selecting IC components that demonstrated the highest conversion efficiency under the expected load conditions.

Additionally, the power architecture was designed to favor switch-mode components (switch mode battery charger, buck-boost and boost converters) over linear solutions wherever significant voltage conversion was needed, since they maintain far higher efficiency. Linear regulators were reserved only for small voltage drops, minimizing their impact on overall power loss. By thinking through each rail and its requirements, the design was designed for optimal efficiency at the system level, resulting in less wasted power, longer runtime on a single charge, and improved sustainability of the device.

5.4.2 Incorporating Renewable Energy Sources

Incorporating renewable and alternative energy sources into our design can further enhance sustainability. For instance, integrating small solar panels could provide a sustainable power source, particularly in outdoor environments.

5.4.3 Reducing Electronic Waste

Reducing electronic waste is another critical consideration. By designing the system to be modular, we enable individual components to be upgraded or replaced without requiring disposal of the entire device. This approach extends product lifespan, improves repairability, and helps minimize the volume of discarded electronics.

Outside of design modularity, the adoption of USB-C as the universal charging standard, especially in the EU, ensures compatibility with a wide range of chargers and cables already owned by users. This avoids the need to ship proprietary adapters or connectors, which often end up as e-waste when devices are replaced. Since nearly all personal electronics and smaller devices now rely on USB-C, we can leverage existing cables and power supplies, significantly reducing unnecessary duplication and disposal of accessories.

In addition, the rechargeable lithium-ion battery system reduces waste by eliminating the need for disposable batteries. Rather than replacing the battery each time it is drained, users simply recharge the integrated cell via USB-C. Over the product's life, this dramatically cuts down on the number of single-use batteries that would otherwise be discarded. Furthermore, if the rechargeable battery eventually degrades, the modular battery board design allows for the cell to be replaced independently, without scrapping the entire device.

By combining modular hardware, standardized USB-C charging, and a rechargeable battery system, the design directly supports sustainability goals and helps limit the environmental footprint associated with consumer electronics.

5.4.4 Environmentally Responsible Materials

The selection of environmentally responsible materials and components is very important. Opting for parts made from recycled materials or those designed for recyclability aligns our project with sustainable practices. Furthermore, employing minimal and recyclable packaging for both the final product and any prototyping materials will help reduce the overall environmental footprint.

5.4.5 Ensuring Long-Term Usability

Ensuring long-term usability is vital for enhancing sustainability. By designing our software and hardware with upgradability in mind, we can allow for future enhancements without requiring complete replacements. Choosing components from manufacturers known for their longevity and support will also reduce the risk of short-term use.

5.4.6 Addressing Social and Economic Dimensions

Striving for affordability in our design will make the project accessible to a broader audience, while user-friendly features will cater to individuals with varying levels of expertise. Additionally designing for resilience; ensuring reliable performance across diverse operating conditions; will build user trust and extend the product's lifecycle.

6. System Test and Verification

6.1 Software Systems

6.1.1 Latency Testing

- Target: End-to-end latency < 5 seconds from image capture to decision display.
- Verification Method: timestamp logging at each stage (ESP32 capture, JSON transmit, cloud response, LCD update). Compare against requirement.

6.1.2 Accuracy Testing (Image Recognition)

- Target of 100% accuracy for player hole cards,>= 99% accuracy for community cards, chip counts within +/- 1.
- Verification Method: Run test dataset of labeled card/chip images through pipeline; calculate True Positive Rate (TPR), False Positive Rate (FPR).

6.1.3 Software State Management

- Target: Correct handling of poker round flow (pre-flop, flop, turn, river).
- Verification Method: Simulate sequences of images with known transitions and ensure software state updates align with expected game stage.

6.1.4 Error Handling & Robustness

- Target: System recovers from >= 95% of transmissions or recognition errors.
- Verification Method: Simulate dropped packets, corrupted JSON, or low-confidence recognition (<80% certainty). Check if retries or default safe outputs are triggered.

6.1.5 Cloud-Device Communication

- Target: Successful upload/retrieval rate >= 99% for JSON payloads.
- Verification Method: Send batch of N requests and verify N-1 successful round trips. Monitor with Azure logging + ESP32 serial logs.

6.1.6 Model Accuracy

- Target: Accurately predict the best possible action for a specific situation that arises in a poker round.
- Verification Method: Use an 80/20 split, training the model on eighty percent of the data and having it predict on the other twenty percent then comparing the predictions from the model to the actual data afterwards. This will provide us with information on how accurate the poker bot actually is.

6.1.7 Integration Testing

- Target: Confirm end-to-end flow (camera -> ESP32 preprocessing -> cloud recognition -> poker decision -> LCD).
- Verification Method: Use both simulated inputs (test images) and live demo scenarios. Ensure decisions and recognized state match expected outputs.

6.2 Hardware Systems

6.2.1 Power Board System Tests and Verification

Input Power Negotiation, Power Path and Battery Charging

Parameter	Target	Verification Method	Notes
USB-C Input Power	5V, 3A Max	Sweep load from 0-3A (if access to E-load), multimeter tests for voltage and current	Ensure USB-C Negotiates correct input power with USB-C Source
Battery Charge Voltage	4.2V ± 10%	Measure BAT charge voltage while input is present	
Battery Charge Current	1.5A ± 10%	Measure charge current when USB-C present using multimeter	
Power Path Source Priority	USB-C > BAT	If there is no USB-C source, power is pulled from battery	

Table 1: Power System

Point of Load Voltage Regulation

Parameter	Target Voltage	Target Load Capabilities	Ripple	Verification Method
5V Boost	4.9-5.1V	≥ 150mA	≤ 50mVpp	Load Sweep + Multimeter + Oscilloscope
3.3V Buck-Boost	3.2-3.4V	≥ 500mA	≤ 50mVpp	Load Sweep + Multimeter + Oscilloscope
2.8V LDO	2.7-2.9V	≥ 150mA	≤ 20mVpp	Load Sweep + Multimeter + Oscilloscope

Table 2: Point of Load Regulation

*Observe switching waveforms for all switching regulators to confirm IC operation

Point of Loads Efficiency

Parameter	Target	Verification Method	Notes
Boost 5V @ 100mA	>90%	Multimeter to measure input and output power	Need an E-Load or a way to test load
Buck-Boost 3.3V @ 300mA	>90%	Multimeter to measure input and output power	Need an E-Load or a way to test load
LDO 2.8V	>80%	Multimeter to measure input and output power	Need an E-Load or a way to test load

Table 3: Point of Load Efficiency

6.2.2 ESP32-S3 Testing

Parameter	Target	Verification Method	Notes
Power Consumption (Active & Idle)	Boot < 450 mA Normal < 200 mA Idle < 10 mA	Measure current with multimeter.	Need Multimeter, DC Power Supply.
Boot Reliability	= 100%	Repeated Power Cycles and Serial Monitor.	Need DC Power Supply.

Digital I/O signal Integrity	High = 2V – 3.3V Low = 0V – 0.8V	Using Oscilloscope and multimeter for signal checks.	DC Power Supply, Oscilloscope, Multimeter.

Table 4: Camera Testing

6.2.3 Camera Testing

Parameter	Target	Verification Method	Notes
Frame Rate	>15 FPS	Timestamped Frame Capture; measure the intervals.	
Power/Current Draw	<50 mA	Use multimeter to measure current.	Need Multimeter and Power Supply.
Image Integrity	Dead Pixels < 0.05%	Run through a program that will analyze and count dead pixels.	

Table 5: Latency Testing

6.2.4 LCD Testing

Parameter	Target	Verification Method	Notes
Frame Update Rate	<100 ms	Measure latency using serial monitor between frame send and visible update.	
LCD Current	< 2.5 mA	Use multimeter to measure current	DC Power Supply and Multimeter.
Pixel Integrity	Dead Pixels < 0.02%	Display test patterns, take pictures and then run through a program that will find dead pixels.	

Table 6: LCD Testing

6.2.5 Entire System Tests

Parameter	Target	Verification Method	

End-to-End Latency	<5 sec.	Use timestamping and serial output.	
Recognition Accuracy	>=99%	Test varied set-ups and compare outputs to correct answers.	
Endurance	>=3 Hours	Run a continuous session for 8 hours.	

Table 7: Entire System Testing

7. Team

7.1 Team Member Benjamin Wu

Benjamin is an Electrical Engineering major with a focus on circuit and system design. His coursework and internship experience have provided him with a strong foundation in PCB design, power electronics, and component selection. Benjamin will be responsible for the full design and implementation of the battery board, which includes component selection, schematic capture, and PCB layout. His role also includes integrating the power management circuitry with the rest of the system, verifying operation through testing, and iterating on the design if necessary.

7.1.1 Skills learned in ECE coursework

Through ECE1895 (Junior Design), Benjamin gained experience with PCB design and prototyping. He worked on creating board layouts, selecting appropriate footprints, and ensuring correct connectivity in the design. Additionally, he learned how to translate schematic-level designs into functional breadboard prototypes, which helped reinforce his understanding of circuit validation and debugging. This hands-on experience with both breadboarding and PCB design provided a strong foundation for system-level hardware development and prepared him to handle the iterative design cycle of schematic capture, layout, and testing. Complementing this, his coursework in ECE0101 and ECE0102 equipped him with the theoretical grounding in circuit fundamentals and microelectronic devices.

7.1.2 Skills learned outside ECE coursework

Benjamin has completed two internships with Texas Instruments, where he built skills in reading and interpreting datasheets, selecting and implementing ICs for system-level designs, and testing ICs with Evaluation Modules (EVMs). He gained experience setting up test conditions and comparing datasheet specifications to real device behavior. These internships also gave him exposure to a wide range of power management components. While comfortable with EVM testing, he has not yet designed a full system-level PCB and has not worked with multiplexers, BQ battery management ICs, or battery-powered designs.

To close this gap, Benjamin will study datasheets for the MUX, BQ IC, and supporting power components to understand their operation and pin functions. He will use application notes, design guides, and EVM documentation for reference designs and layout recommendations. Additional resources will include TI's E2E online forums, reference designs, and academic PCB design guides.

He will also seek expert input. From industry, he will use TI's technical customer support to confirm device selection and implementation. From faculty and experienced peers, he will ask for feedback on PCB design choices and manufacturability, as well as understand practical issues not covered in datasheets.

Through this combination of experience, research, and expert input, Benjamin will build the knowledge needed to complete the PCB design.

7.2 Team Member Seth Williams

Seth is an Electrical Engineering major with background in machine learning and hands-on experience in PCB design using Altium, along with soldering and assembly work. For this project, he is taking the lead on designing the main MCU module, making sure the board layout and hardware connections are set up to support the rest of the system.

7.2.1 Skills learned in ECE coursework

Through ECE coursework, several useful skills have been developed that directly apply to this project. In ECE 1895 (Junior Design), experience was gained in PCB design using Altium, which helped build confidence in creating and troubleshooting circuit layouts. ECE0101 provided a solid foundation in basic circuit knowledge, covering fundamental concepts that are essential for any hardware-focused project. In addition, ECE0102 introduced microelectronic circuits, giving practical insight into how smaller-scale electronic components operate and connect within a system.

7.2.2 Skills learned outside ECE coursework

Outside of ECE classes, there are several new skills that will need to be picked up over the course of this project. One of the main ones is learning how to design a custom PCB based on the ESP32 chip, and example videos have already been found that walk through the basics of this process. Another area of focus will be figuring out how to use the OV5640 camera module and an LCD, with tutorials providing guidance on how to operate and integrate them. In addition, surface mount soldering is a skill that will be important for assembly, so support will be sought from SERC, especially by consulting Bill McGahey and Jim Lyle for their expertise.

7.3 Team Member Nicholas Lavine

Nick is a Computer Engineering major with a strong background in machine learning and embedded systems. For this project he is taking the lead on designing the machine learning algorithm that will deliver the user with predictions on what action to take in the poker game. He is also responsible for the data collection/generation for the model to learn off of.

7.3.1 Skills learned in ECE coursework

Throughout the ECE curriculum, Nick has gained a lot of valuable skills. In ECE 301 and ECE 302 as well as ECE 1148 he became experienced with software development strategies and overall coding ability. In classes such as ECE1140 and ECE 1895 he grew proficient in working amongst teams to develop complex designs as well as the overall process of creating an in-depth engineering design. In other classes such as ECE 1195 and ECE 1175 he became experienced with embedded systems and integrating software within hardware to create different designs. All these skills learned have led to this project which involves complex code design and implementation for the machine learning module, being able to function within a team to create an overall design, and integration between the model and the hardware to complete the overarching system.

7.3.2 Skills learned outside ECE coursework

Outside of class Nick has participated in a research group led by Dr. Paul Ohodnicki for the University of Pittsburgh during his junior and senior year of college. During this time Nick was able to work with a group of engineers to develop a convoluted neural network machine learning algorithm as well as a graphical user interface that accompanied the model. Doing this has given Nick exuberant amounts of experience in developing, training, and deploying machine learning models which is extremely important for his aspect of the poker coach which consists almost entirely of developing the machine learning model as well as integrating it with other aspects of the design. The parallels between research work and the design project have been perfect for Nicks understanding and development of the final model for the poker coach.

7.4 Team Member PJ Granieri

PJ is a Computer Engineering major with a strong background in software engineering and applied AI systems. He has experience in image processing, cloud services, containerized deployments, and network applications, making him well-suited to lead the software integration aspects of the project. PJ is responsible for the Image Processing Module as well as the integration between hardware and software, specifically ensuring communication from the ESP32 device to the Microsoft Azure cloud services and back.

7.4.1 Skills learned in ECE coursework

Through his coursework, PJ has gained significant exposure to programming and system design across multiple languages and platforms. In ECE 301 and 302m he developed proficiency in C++ with a focus on object-oriented programming, data structures, and algorithm design. ECE 1140 and 1145 built his foundation in software development lifecycles, design patterns, refactoring, and teamwork under agile methodologies. ECE 1148 expanded his skills in algorithmic design and optimization, while ECE 1150 strengthened his understanding of networking principles crucial for device-cloud communication.

In more advanced courses, PJ has worked directly with embedded and AI-related systems. ECE 1175 and 1390 introduced embedded systems principles and image processing techniques, both directly relevant to the poker coach. ECE 1894 and 1895 provided hands-on experience with cloud-

native storage, and full-stack integration, preparing him to design scalable pipelines for this project.

7.4.2 Skills learned outside ECE coursework

Outside the classroom, PJ has broadened his expertise through self-directed projects and internships. He has experience deploying systems on Microsoft Azure, building containerized services with Docker and Kubernetes, and designing APIs with FastAPI. He has worked extensively with databases such as PostgreSQL and Neo4j, as well as emerging AI frameworks like Langchain, Langmem, and Agno for agentic AI development.

To strengthen the project's outcome, PJ anticipates consulting additional resources. For example, he plans to review image processing literature and lecture notes to refine the efficiency of low-level C++/Arduino algorithms and will seek guidance from cloud service experts to ensure reliable deployment of the Azure pipeline. He also intends to validate networking integration by referencing industry documentation and expert input.

Through this combination of academic training, practical software engineering experience, and a willingness to learn from experts, PJ is well-prepared to deliver the software backbone for the prototype and ensure robust communication between hardware and cloud systems.

8. Schedule and Budget Plan

8.1 Project Schedule

Week 1 (Monday 9/29)

- Ben: Test load draws of LCD Screen, ESP32 through modules to get real life data, set up test Lithium Ion Battery, read datasheets for battery charger and PD controller
- Seth: Prototype LCD screen and OV5640 camera on ESP32-S3 Development board to test if working with the ESP32 chip isfeasible.
- Nick: Finalize the collected data and prepare it for delivery to the machine learning model.
- PJ: Begin/Continue Prototyping C++ Image Processing Algorithms

Week 2 (Monday 10/6): CHECK OFF 1

- Ben: Configure and test TPS25730D EVM to negotiate correct power levels, have a rough draft of PCB layout completed.
- Seth: Finalize board components and create rough draft of PCB schematic.
- Nick: Begin developing the MLP model for predicting actions.
- PJ: Continue Developing C++ Image Processing Algorithms + Python Image Processing Algorithms + Cloud Hosted Services

Week 3 (Monday 10/13)

- Ben: Finalize power board PCB layout, generate Gerbers and submit PCB order, submit order for IC that are needed, and order PCB

- Seth: Finalize PCB layout, generate Gerbers and submit PCB order, submit order for any other components that are needed.
- Nick: Finish the model prototype and begin the testing portion of the model's production.
- PJ: Continue Developing C++ Image Processing Algorithms + Python Image Processing Algorithms + Cloud Hosted Services

Week 4 (Monday 10/20): MIDTERM PRESENTATION

- Ben: While PCB is still processing, set up 1S battery pack
- Seth: While PCB is still processing, simulate a simple program that makes the OV5640 and LCD work together on the ESP32-S3 Development board
- Nick: Test the model and optimize it for efficiency and accuracy.
- PJ: Software Development + Cloud Hosted Services + Networking Integration + Deploying C++ Algorithm Locally

Week 5 (Monday 10/27)

- Ben: PCB assembly, solder components and test
- Seth: PCB assembly, solder components and connectors.
- Nick: Continue testing and refactoring the model to produce the most accurate predictions possible.
- PJ: Software Development + Cloud Hosted Services + Networking Integration + Deploying C++ Algorithm Locally

Week 6 (Monday 11/03)

- Ben: Interface Power board with MCU Board
- Seth: Verify Power rails on completed PCB.
- Nick: Start to integrate with the Computer Vision module and hardware modules.
- PJ: Software Development + Cloud Hosted Services + Networking Integration + Deploying C++ Algorithm Locally

Week 7 (Monday 11/10): CHECK OFF 2

- Ben: Debug as needed, start creating 3D Body for Project, CAD design of 3D body done
- Seth: Confirm USB-C programming, boot/reset, run LCD and Camera tests.
- Nick: Test the capabilities of communication and refactor if necessary to make the model as efficient as possible.
- PJ: Refactoring Software + Cloud Services + Networking

Week 8 (Monday 11/17)

- Ben: 3D Body finished, assemble poker enclosure with hardware components
- Seth: Measure latency, FPS, and recognition accuracy.

- Nick: Add finishing touches to the model and produce visual results such as confusion matrixes for presentation purposes.
- PJ: Refactoring Software + Cloud Services

Week 9 (Monday 11/24)

- Ben: Prep for Final Presentation
- Seth: Debug as needed.
- Nick: Debug and prepare for the final presentation.
- PJ: Refactoring Software + Cloud Services

Week 10 (Monday 12/01): FINAL PRESENTATIONS

- Ben: Work on Poster + Final Report.
- Seth: Work on Poster + Final Report.
- Nick: Work on Poster + Final Report.
- PJ: Work on Poster + Final Report

Week 11 (12/08): FINAL REPORT DUE (12/12)

- Ben: Finalize Final Report
- Seth: Finalize Final Report.
- Nick: Finalize Report.
- PJ: Work on Final Report

8.2 Project Budget

Name	Vendor	Part Number	Quantity	Unit Price	Total Price
ESP32-S3-CAM Dev Board	**Amazon**	**NA**	**1**	**15.49**	**15.49**
OV5640 Cam Module	**Amazon**	**NA**	**1**	**14.24**	**14.24**
LCD Display	**JB Superstore**	**NA**	**1**	**19.99**	**19.99**
Jumper Wire (Male-Female)	**JB Superstore**	**NA**	**1**	**0.99**	**0.99**

Jumper Wire (Male-Male)	**JB Superstore**	**NA**	**3**	**0.99**	**2.97**
LED (Surface Mount)	**JB Superstore**	**NA**	**1**	**0.05**	**0.05**
SMD USB C Port	**JB Superstore**	**NA**	**2**	**0.78**	**1.56**
SMD Push Button	**JB Superstore**	**NA**	**2**	**0.20**	**0.40**
ESP32-S3-WROOM-1 Bare Chip	**JB Superstore**	**NA**	**1**	**5.50**	**5.50**
SMD USB-C Port	**JB Superstore**	**NA**	**1**	**0.15**	**0.15**
Zener Diode	**Digikey**	**BZT52C5V1-7-F**	**3**	**0.10**	**0.30**
Protection Diode	**Digikey**	**1N5819HW-7-F**	**1**	**0.27**	**0.27**
5.1K Resistor SMD	**Digikey**	**ERA-2AEB512X**	**2**	**0.10**	**0.20**
200 Resistor SMD	**Digikey**	**ERA-2AEB201X**	**1**	**0.10**	**0.10**
10K Resistor SMD	**Digikey**	**RC0603FR-0710KL**	**3**	**0.10**	**0.30**
10uF CAP SMD	**Digikey**	**CL05A106MP5NUNC**	**2**	**0.08**	**0.16**
1uF CAP SMD	**Digikey**	**JMK105BJ105KV-F**	**1**	**0.08**	**0.08**
0.1uF CAP SMD	**Digikey**	**CL05B104KP5NNNC**	**3**	**0.08**	**0.24**
USB-C CC/PD Controller	**Digikey**	**TPS25730DREFR**	**1**	**2.71**	**2.71**
Battery Charger IC	**Digikey**	**BQ25616JRTWR**	**1**	**2.51**	**2.51**

LDO	**Digikey**	**TLV75530PDBVR**	**1**	**0.28**	**0.28**
Boost Converter	**Digikey**	**TPS61023DRLR**	**1**	**1.08**	**1.08**
Buck-Boost Converter	**Digikey**	**TPS631011YBGR**	**1**	**1.25**	**1.25**
TVS Diode	**Digikey**	**TVS2200DRVR**	**1**	**0.54**	**0.54**
NTC Thermistor	**Digikey**	**NCP18XH103F03RB**	**1**	**0.17**	**0.17**
Lithium Ion Battery	**Amazon**	**115659 3.7V Lipo Battery 5000mAh 115659**	**1**	**13.99**	**13.99**
				Total =	**85.52**

8.3 Minimum Standard for Project Completion

At a minimum, the Poker Assistant Coach prototype must demonstrate the ability to capture, process, and display basic poker information in real time. If system complexity or time constraints prevent full functionality, the reduced minimal viable prototype will still prove the feasibility of the concept by meeting the following requirements:

- Camera Capture:
  - The OV5640 camera must successfully capture at least a single card or chip stack image and pass this image to the ESP32-S3
- Preprocessing & Transmission:
  - The ESP32-S3 must crop and compress the image into a simplified format. This image or metadata must be transmitted to either the cloud pipeline or a local-only algorithm.
- Recognition:
  - The software pipeline (cloud-hosted or on-device) must identify at least one card rank/suit or chip count correctly.
- Predictions
  - The model should be able to offer predictions that help the user keep their money net positive.
- LCD Display:

    - The LCD must output the recognized state and the recommended action clearly, even if only for a simplified test scenario.
- Power
    - The battery must power the system, even if it is unrechargeable

Success Criteria:

- System demonstrates the end-to-end workflow of Camera -> Processing -> LCD Decision Output.
- Recognition accuracy and decision-making do not need to meet final project targets but must prove functional viability.
- Even if limited to one card at a time, the system must show it can extract data and analyze it and produce an output to the user.
- Power board functions fully as expected, being able to charge the battery and power the loads for minimum 3 hours

This ensures that even under worst-case constraints, the project delivers a working prototype that validates the integration of hardware capture, image processing, and decision making & display, preserving the educational and demonstrative goals of the Poker Assistant Coach.

8.4 Final Demonstration

1. The OV5640 camera will be used to capture information on the poker table, including cards and game state. This verifies that the system can accurately sense and record the required input data.
2. The ESP32-S3-WROOM-1 microcontroller will process all captured inputs, run the poker decision-making logic, and generate the appropriate advice. This ensures that the computation and decision-making functions are operating as intended.
3. The processed results will be displayed on the LCD screen in real time, showing the advice or recommended action to the user. This validates that the information transfer from sensing, to processing, to output is functioning correctly.
4. The prototype will run fully on an onboard 1S lithium-ion battery, demonstrating the portability and practicality of the device without reliance on an external power supply.

The tests to be shown:

1. The camera feed will be tested by capturing cards on the table under normal lighting conditions to confirm image clarity and recognition accuracy.
2. The microcontroller's processing will be verified by providing test game scenarios and confirming that the generated advice matches expected outputs.
3. The LCD screen will be tested for readability and correct formatting of the advice displayed to the player.

4. The system's battery-powered operation will be demonstrated by running the device unplugged for the duration of the test, confirming energy efficiency and stability of operation.

Furthermore, the presentation will include data, images, and analysis that highlight the accuracy of card recognition, correctness of decision outputs, and stability of battery performance.

References

1. Poker Market Size
2. WSOP Popularity
3. ESP32 Datasheet
4 USB-C Facts
5. Power Design
6. Signal Integrity
7. ML Card Detection
8. STM32N657 Data Sheet
9. ESP32-S3-WROOM-1 Data Sheet
10. Azure Blob Storage
11. ESP32 Edge Computing Paper
12. TinyML
13. Multi-Layer Perceptron
14. Reinforcement Learning
15. OpenCV
16. Scikit-Learn
17. Mediapipe
18. PyTorch
19. TensorFlow
20. Texas Instruments TPS25730 Product Page
21. Infineon CYPD3177-24LQXQ Product Page
22. Analog Devices MAX77958 Product Page
23. Texas Instruments BQ25616 Product Page