PokerBuddy: Chipless Poker Companion

By

Austin Abraham

Lorenzo Dumitrescu

Vishal Ramvelu

Final Report for ECE 445, Senior Design, Fall 2025

TA: Eric Tang

05 May 2025

Project No. 85

Abstract

PokerBuddy is a wireless, chipless device designed to streamline in-person poker games by replacing physical chips and automatic tracking with electronic components. Each device manages game logic locally, including pot size tracking, player balances, and turn order. We accomplish this by using a microcontroller, tactile chip-value buttons, gesture sensors, and visual indicators. Devices communicate wirelessly to maintain a synchronized game state across all players. The system demonstrated accurate real-time updates with over 98% input recognition and consistent pot/balance tracking. It maintained reliable communication within an 8-foot radius and successfully integrated hardware and software for seamless gameplay. The final design achieved full functionality, demonstrating the feasibility of enhancing traditional poker with embedded systems while preserving the face-to-face experience.

Contents

1. Introduction

PokerBuddy is a chipless poker system designed to enhance in-person gameplay without the hassle of physical chips. Traditional poker can be slow, error-prone, and disorganized, especially when managing chips, tracking bets, and determining turns. PokerBuddy replaces all of that with a modular device that handles game logic, bet tracking, and player actions through tactile buttons, displays, LEDs, and wireless communication. Players still enjoy the face-to-face experience, but with smoother, faster, and more reliable gameplay.

1.1 Problem

Traditional poker games rely heavily on physical chips for betting, which can be cumbersome, error-prone, and subject to mismanagement or theft. Managing chip counts, handling physical money, and tracking bet amounts often slow down the game and can lead to disputes among players. In addition, determining whose turn it is during fast-paced games can be confusing and cause a lot of frustration between players. With the growing demand for digital integration in gaming, there is an opportunity to streamline the poker experience by eliminating physical chips and automating bet tracking and game flow. This is different from online poker because we want to maintain the in-person experience of playing against your friends face-to-face, but without the inefficiencies of standard chips and markers that represent blinds.

1.2 Solution

We propose a modular device that removes the need for physical chips while enhancing the poker-playing experience. Each player will use a dedicated device that features LED displays to show both their current balance and the money in the pot, along with a built-in turn indicator light that activates when it is their turn. We will use 10 buttons that correspond to different chip denominations and action buttons for quick and easy betting. We want to maintain the tactile feel and choose to use buttons for our design. These devices will wirelessly connect to manage buy-ins, track all player balances, and synchronize game status in real time, ensuring an efficient and error-free gaming experience. Although these devices will not track cards, they must handle the real-time logic of betting, maintaining balances, and managing turn order without relying on a computer.

The game logic is distributed and managed by the PCBs in each PokerBuddy. This means that each PokerBuddy keeps track of:

- Whose turn is it to bet (reflected by the turn signal LED diode)
- The current bet amounts and how they contribute to the pot (reflected by the LED display)
- The opponent's individual balances (reflected by their LED display)

These devices communicate wirelessly with each other and constantly relay game updates to change player balances, pot size, and turns accordingly. The system is designed to be portable and is powered by disposable batteries, ensuring flexibility and ease of use in various settings.

1.3 Visual Aid

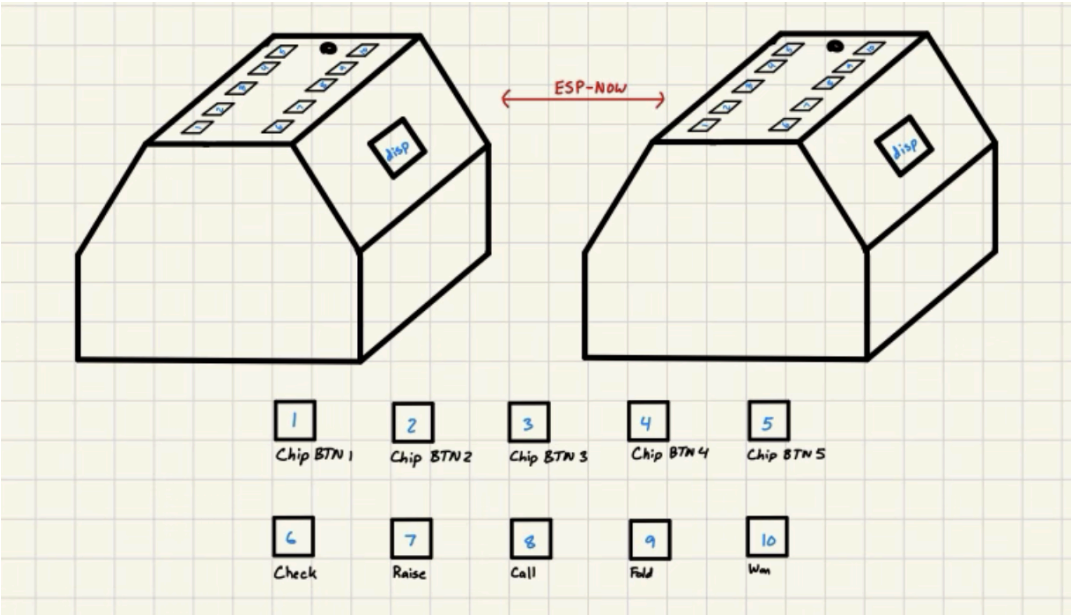

Figure 1: PokerBuddy Visual Aid

Figure 1 shows two individual PokerBuddy devices playing together. Each device will consist of ten buttons: CHIP_BTN_1, CHIP_BTN_2, CHIP_BTN_3, CHIP_BTN_4, CHIP_BTN_5, CHECK_BTN, RAISE_BTN, and CALL_BTN, FOLD_BTN, AND WON_BTN. Each chip button will be used in place of the chips that are used in poker. The check, raise, call, fold, and won buttons will be used in place of the actions to continue the progression of the game from person to person. The device will also have two displays and a diode used for a turn indicator. The displays will show the player whose device it is and the other players' important game statistics, like their balance and the pot commission.

1.4 High-Level Requirements

For us to consider our project a success, we must complete the following:

1. The game is mathematically correct. This means that 100% of the time, the machine will keep track of all the players' current balances, each individual's game pot and player contributions, along with the final calculations after each game has been completed.
2. The status of the game will update immediately and accurately after the user interacts with the device. This means that 98% of the time, the user's inputs will be recognized correctly, while 100% of the time, the machine will execute the proper command from the read input.
3. The machine will have reliable wireless communication as long as all the devices are within an eight foot radius of the table. This will ensure that all players know how close to the device to maintain proper gameplay and communication.

2 Design

2.1 Design Procedures

For our physical design, we chose to use 3D printing. This allowed us to form a well-designed enclosure which was able to host all the needed components (buttons, LED display, PCB, etc). Through careful iterations and analysis of what we wanted as the final product, we reached the final 3D printed version **[7]**.

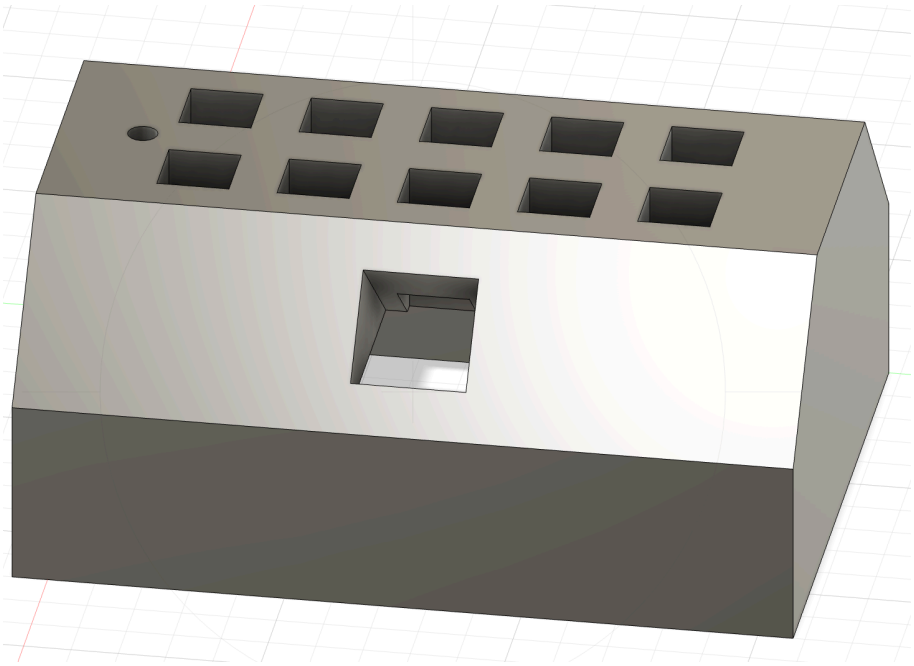

Figure 2: Final 3D Model of Enclosure

The design choices shown in Figure 2 reflect various areas that we wanted to adapt from the game of poker. Starting at the top, we designed our code to use 10 chip buttons (five for betting and five for action), so we added this into the enclosure. In addition, we added an LED indicator light to highlight whose turn it is quickly. By placing these on the top, we provided an intuitive and easy way of using these functions. We also placed the LED displays on the slanted surface of both sides to allow for everyone at the table to quickly understand the needed information at a glance.

The choices we made allow our product to maintain its aesthetic appeal while being straightforward to use. In addition, we added screw holes to the bottom of the enclosure to add in the bottom plate. What this allows us to do is place the PCB and all wiring inside and be able to hide it from plain sight, maximizing user experience.

This final version was brought about by several iterations as we noticed areas to improve. Figure 3 pictured below is an example of one of the earlier versions.

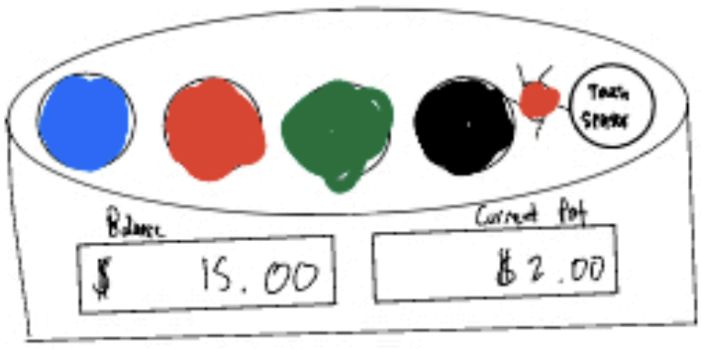

Figure 3: Initial Version of Enclosure

As seen above in Figure 3, there are various placements of the display, how many buttons to utilize, and overall sizing/placement to modify. Ultimately, in the final product we found that having 10 buttons allows complete functionality. In addition, the display can be placed on one side, both sides, two on one side, and some more permutations. By combining the balance and current pot size into one display, we decided to have one display placed on both sides. This lets both the entire table and you to quickly grasp the needed information.

We also changed the design to create a slanted placement for the LED display to once again create the most optimal angling for vision. Furthermore, as we finished more parts of the project, we recognized the need to place the PCB and wiring inconspicuously. To this effect, we decided the most optimal approach is to create screw holes in the bottom of the enclosure and have a removable bottom plate layer. By doing this, we allow for easy demonstration and minimal obstruction for the players, as it is not noticeable during gameplay. All these individual choices were built upon multiple rounds of testing and live testing as we got to see and evaluate each 3D printed version. In doing this, we slowly built up a better intuition on what creates an aesthetic, well-designed enclosure and evolved our product. Our final product contains several intricate design choices we made to create the best possible user experience.

The next area to analyze is design measurements. When creating our 3D model, we had to develop definite lengths for all required components. The calculations needed here were to identify the overall length, width, and height of the physical device I had to create, with proper spacing for buttons and LED displays. To start this process, we drew sample sketches to demonstrate the needed measurements.

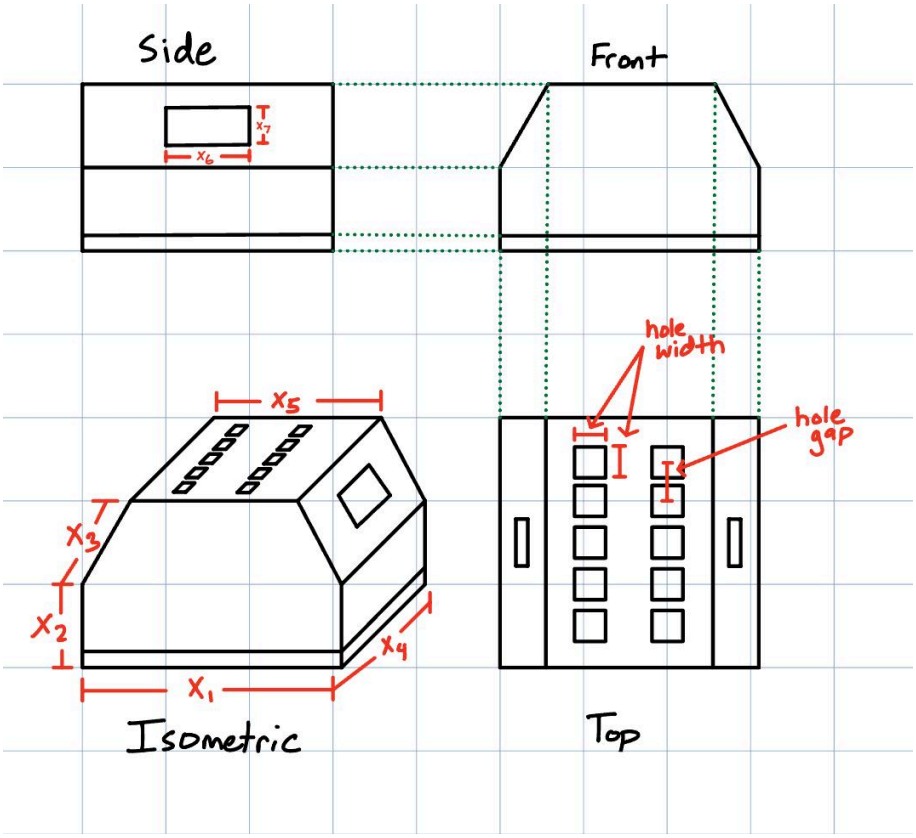

Figure 4: CAD Measurement Schematic

Each measurement in Figure 4 represents the exact CAD dimensions needed to create the final working design. The numbers and calculations corresponding to those are as follows. For the hole diameters, we are using Cherry-MX 1.00u Keyboard switches, which are each 0.5 inches in width. We have 10 buttons layered into two splits, meaning five per row. To have a presentable view, the gap between each button should be roughly 0.75 inch, so the calculation for x5 is 0.75 (left edge) + 0.5*5 (actual buttons) + 0.75*4 (between buttons) + 0.75 (right edge) = 7 inches. We follow a similar calculation for x4, which is 0.5 (top side) + 0.5 * 2 (two rows) + 0.5 (bottom side) = 2 inches.

The next calculation we can do is for x6 and x7. This is directly derived from the given parts we ordered, which are 1.5 in by 1.5 in. But, to best accommodate for wiring and spacing, from our testing we found that an extra 0.25 inch on both dimensions allows for best visualization resulting in x6 and x7 being 1.75 inches. Now that we know x6 and x7, we can find how long x3 should be to best accommodate the display. Ideally, the LED display is perfectly in the middle of the spacious slant to have a user-friendly interface. In this way, we tested slant lengths 6, 6.5, and 7 in and found 6.5 inches to be the best length, making x3 = 6.5 inches.

The next step is to find dimensions for x1 and x2 that ensure that the physical device is still lightweight and compact, but still dense enough to hold this layout. We also have to recognize that we have to place our PCB inside the enclosure and close it with the bottom plate, so we need a big enough gap. To this effect, based on our PCB and overall structure, x1 should be (1.25*x5), so x1 becomes 8.75 inches. Our

x2 should be there to give some height before slant, so we tried 1,1.5, and 2 inches to find the ideal height before settling on 1.5 inches. Now that we have all our measurements fully identified by finding relevant components and actual verification of design by 3D printing, we have found the measurements needed for the final enclosure.

Overall, through careful analysis of components used and multiple iterations of 3D printing our enclosure, we were able to identify the exact sizing needed with the design layout desired, which maximized user experience and allowed for an aesthetic physical design.

2.2 Physical Diagram

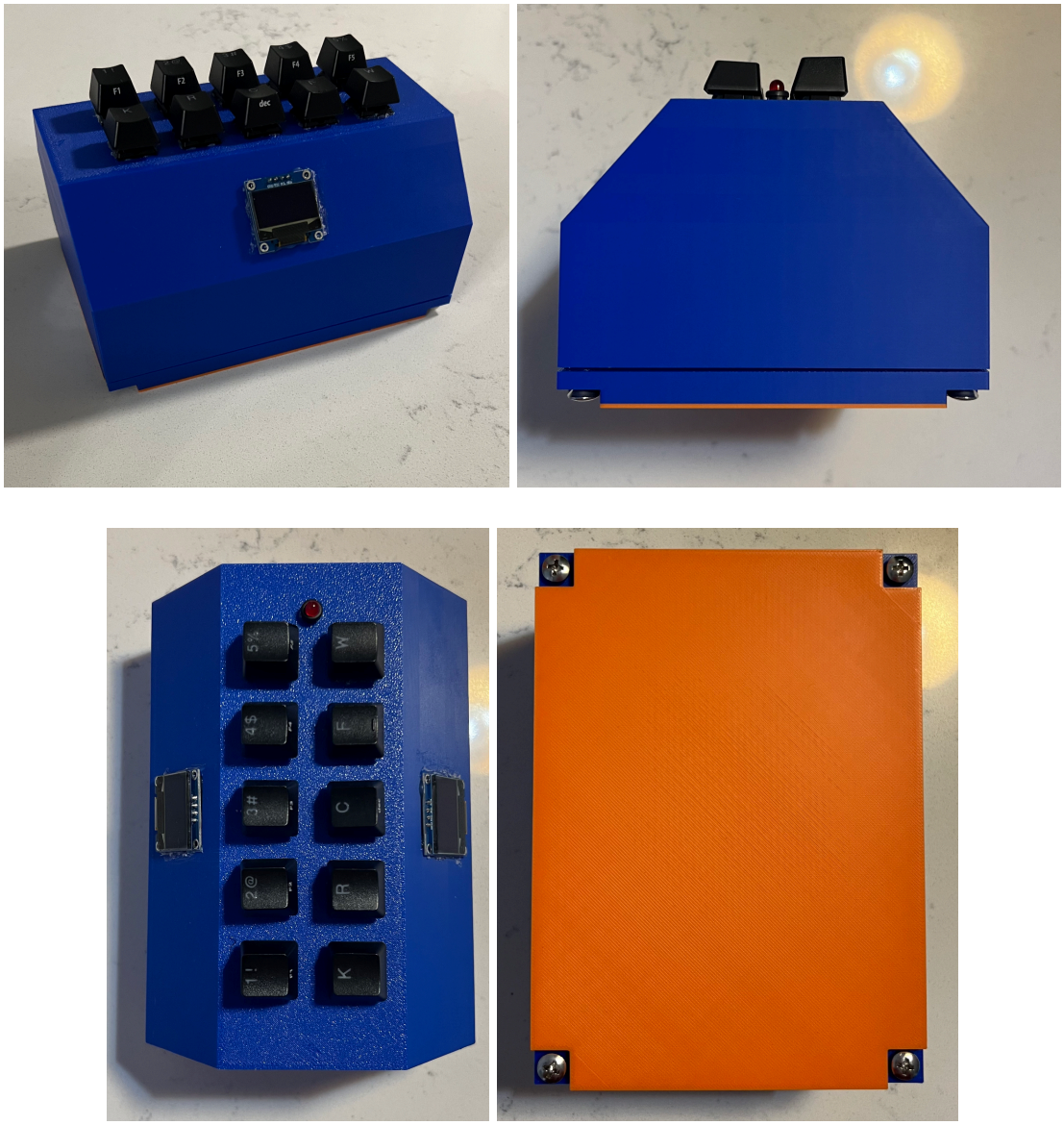

Figure 5: Final Physical Product

2.3 Block Diagram

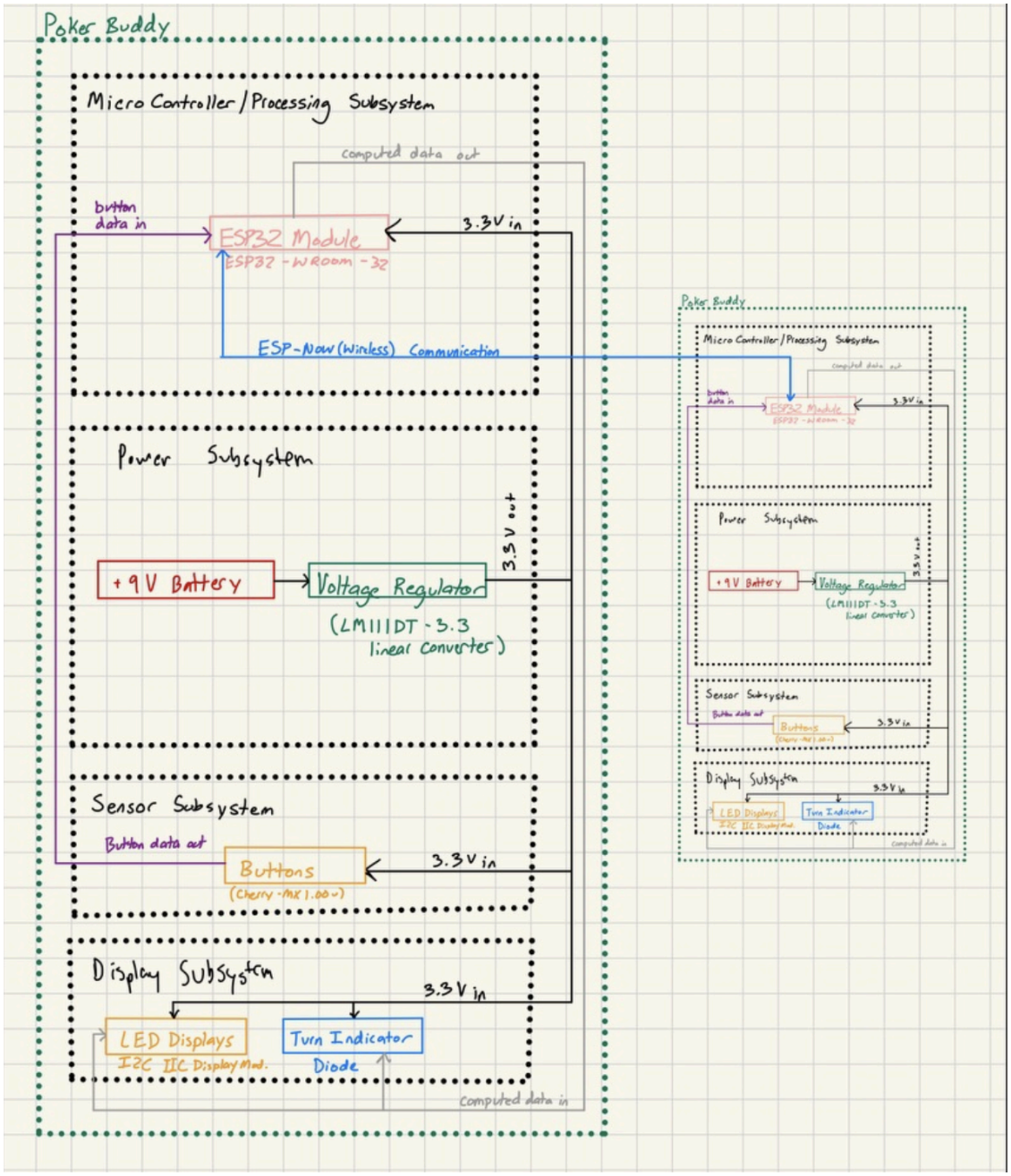

Figure 6: PokerBuddy Block Diagram

This block diagram seen in Figure 6 consists of four separate subsystems, including: Microcontroller and Processing Subsystem, Power Subsystem, Sensor Subsystem, and Display Subsystem. The Microcontroller and Processing Subsystem consists of the microcontroller (ESP32) and programming adapter. This subsystem will be used to control the other subsystems, read and write data, and do computations. The Power Subsystem is used to take in the power from the source and convert it to 3.3V to be used by the

microcontroller. The microcontroller has a power line through it, which can help power the other devices. The Sensor Subsystem consists of all the buttons. These devices will be used to interact with the device's user. The Display subsystem will consist of the 2 displays for user visuals and the turn indicator, which is a diode to display the user whose turn it is currently.

The subsystems work together to create our design, and each has an important task. The device starts with a 9V battery that is fed into the LDO to convert it to 3.3V. This power is then fed to the ESP32, which will receive user input from our 10 buttons. Once the microcontroller receives any user input, it uses our code to do all the game logic and arithmetic operations to figure out the next state. Once the state has been determined, the ESP32 will send data to the LED screen to be updated and will also update the turn indicator if necessary.

2.4 Subsystems and Block Description

Our device consists of four different subsystems. They are Microcontroller and Processing, Power, Sensor, and Display subsystems. The power subsystem consists of a 9V battery that is sent to an LDO that converts it to 3.3V to power the ESP32. The microcontroller and processing subsystem contains our ESP32, which reads inputs from our 10 buttons within our sensor subsystem. The microcontroller then computes all game logic and arithmetic to send it to the LED display and turn indicator LED in our display subsystem. These subsystems all work together to make our device function properly.

2.4.1 Microcontroller and Processing Subsystem

This is the core of the PokerBuddy design. It houses the ESP32-S3-WROOM-32 module **[1]**. This subsystem performs all the critical game logic, which includes tracking bets, pot sizes, player balances, and turn order. It does this while orchestrating communication between the various sensors and displays. It sends and receives data between the Sensor Subsystem for user inputs, then updates the Display Subsystem in real time, and then relies on the Power Subsystem for stable operation. The ESP32-WROOM-32 is connected to a USB adapter, which will be used to initially program this device so that it can read and write data, do computations, and deal with other devices **[4]**. It interfaces with the Sensor Subsystem via 3.3V logic-level GPIO and ADC, with sensor data required to be processed within 100 ms, and outputs to LED displays and a dedicated turn indicator diode, which must update within five seconds of a game event. The ESP32 provides built-in WiFi/Bluetooth connectivity to ensure a wireless communication range of at least 16 feet. This subsystem must operate at a stable 3.3V ± 0.1V and handle a minimum processing throughput that guarantees game logic execution in under 100 ms.

2.4.2 Power Subsystem

This subsystem supplies and regulates the voltage necessary for all the components to function reliably. It has a 9V battery feeding into a voltage regulator (LM1117DT-3.3 linear voltage converter) **[2]** to ensure a steady 3.3V supply for the microcontroller and other electronics. By providing clean, stable power, this subsystem is essential for maintaining the performance and longevity of the device. It directly interfaces with the Microcontroller and Processing Subsystem to prevent voltage fluctuations or power loss during play. It maintains the stability of the Microcontroller, Sensor, and Display Subsystems by guaranteeing a minimum of 500 mA of constant current with a low voltage ripple (<50 mV) **[2]**. Critical functions like wireless communication, sensor detection, or display updates would be jeopardized if the subsystem did

not fulfill these requirements, for example, by supplying insufficient current or large voltage dips, which would cause instability across the system.

2.4.3 Sensor Subsystem

Within the Sensor Subsystem, 10 buttons (Cherry-MX 1.00u Keyboard switches) **[3]** will control readings for fold, check, call, raise, and won gestures are recorded, and betting amounts. The Microcontroller and Processing Subsystem receives the data from these sensors, processes it, and uses it to update the game's state and activate any required wireless events or displays. It must send legitimate input signals to the Microcontroller Subsystem in less than 100 milliseconds and run at 3.3-volt logic levels. Hardware/software debouncing is utilized to minimize false triggers because accuracy is crucial—at least 98% of identified movements must be accurate. By resulting in missed or invalid betting inputs, removing any of these requirements (such as an adequate response time or high detection accuracy) would compromise the integrity of the game.

2.4.4 Display Subsystem

The Display Subsystem uses LED displays (0.96 Inch OLED I2C IIC Display Module 12864 128x64 Pixel SSD1306) **[8]** that show each player's balance and the quantity of pot in use, as well as a turn indicator LED (diode) to indicate whose turn it is. It also gives the players visual feedback in real time. When betting actions or the turn order change, it instantly updates the displays based on data and control signals received from the Microcontroller and Processing Subsystem **[10]**. Players are guaranteed to witness real-time changes in pot size and turn order as it updates within five seconds of valid inputs. Furthermore, in typical indoor lighting conditions, it maintains a legible brightness level while drawing power from the controlled 3.3V rail. The system would not be able to accurately tell players if certain requirements were not met, which would erode players' confidence in the betting calculations made by the device.

3. Design Verification

3.1 Microcontroller and Processing Subsystem

3.1.1 Requirement

The microcontroller and processing subsystem must be able to process game logic actions between devices correctly 100% of the time. This includes things like passing the turn, folding, winning the hand, calling, and checking.

3.1.2 Verification

To verify the accuracy of game logic, we took a two-step testing approach. First, we pressed the different action buttons (call, raise, check, fold, or won) 100 times in different simulations to test how often the microcontroller accurately recognizes and performs the action. In addition, in each of these simulations, we wrote out on a piece of paper the expected results to verify the output. Figure 7 is a table containing the results we found.

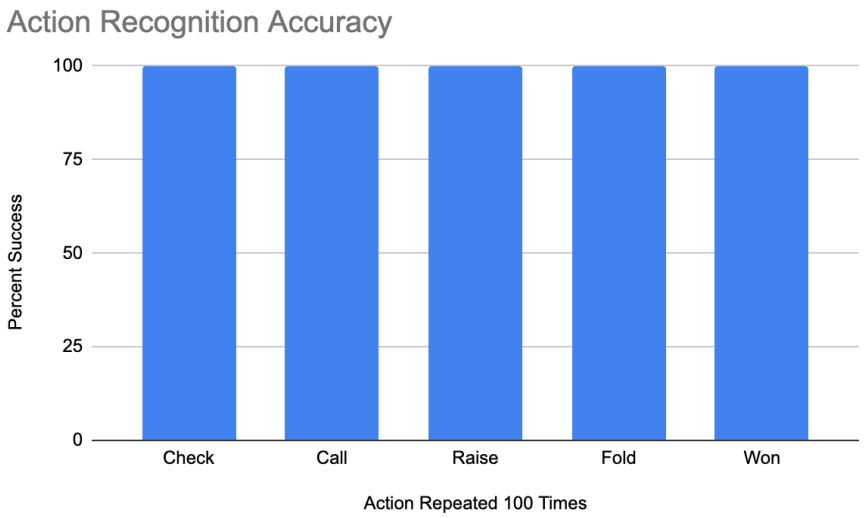

Figure 7: Microcontroller verification

As seen above in Figure 7, for all actions, the success rate for recognizing is 100%, verifying our requirement of recognizing game action 100% of the time. In addition, these actions displayed the expected output 100% of the time also as verified by our handwritten calculations. This proved all the requirements for the microcontroller subsystem.

3.2 Power Subsystem

3.2.1 Requirement

The Power Subsystem must reliably supply 3.3V to all of the other subsystems using a 9V disposable battery. It must accomplish this through the use of an LDO, specifically the LM1117DT-3.3. It must sustain consistent operation for a minimum of four hours under typical operating conditions (normal gameplay.) This requirement prevents other subsystems from getting damaged and allows reliable operation of the device.

3.2.2 Verification

The way we verified this was by seeing how long the machine is able to operate with a fully charged 9V battery and power under a normal load. We tested this by idling the game on the start screen for four hours and then checked to make sure the game was still running in the same state. The game consistently stayed running for longer than four hours verifying our requirement. In addition, we used a voltmeter to check the voltage across components and make sure 3.3V were being supplied to each of the subsystems before connecting everything.

3.3 Sensor Subsystem

3.3.1 Requirement

The sensor subsystem must recognize proper input even if buttons are rapidly pressed so that users can update their bet quickly for better gameplay and buttons pressed when it is not the user's turn will have no effect and not change any of the gameplay. We initially had an accelerometer as part of the sensor subsystem but when we redesigned the PCB we decided to not incorporate the accelerometer due to inconsistent readings.

3.3.2 Verification

To verify this, we started the game and pressed the chip buttons multiple times quickly. The amount deducted from your balance and into the pot should accurately be reflected, even though the buttons were pressed in quick succession of each other. Similarly, when out of turn and buttons are pressed quickly, no action should take place. Figure 8 is a table demonstrating the results we found.

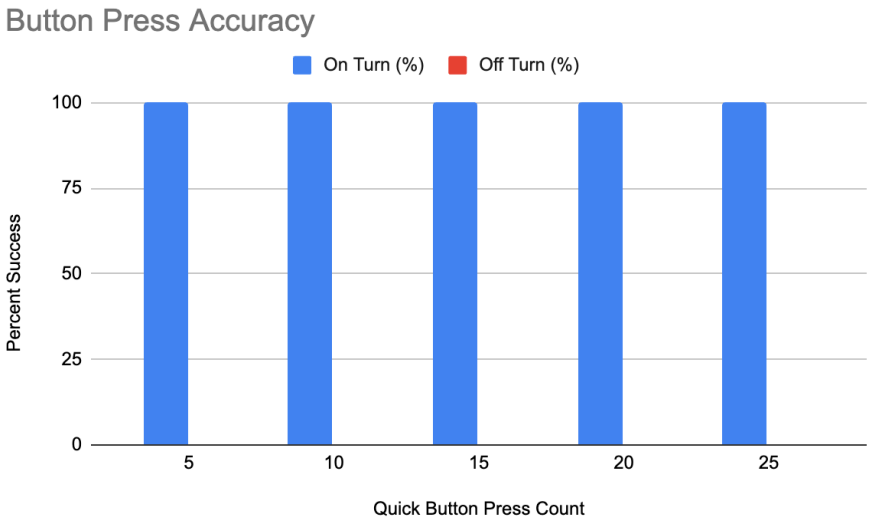

Figure 8: Sensor Verification

What Figure 8 shows is that no matter the number of times buttons are pressed in quick succession, the resulting success rate is still 100%. And when it is not your turn, the buttons are never registered proving both requirements set up for our sensor subsystem. Testing the accelerometer provided a success rate of less than 50% which would have compromised the integrity of the game. Sudden movements of the device gave false positives on the accelerometer which drastically decreased the reliability of the subsystem.

3.4 Display Subsystem

3.4.1 Requirement

The display subsystem shall power on within one second of system boot and refresh its content within one second of any user action. All text and graphics must remain legible at a one-foot viewing distance under normal indoor lighting, and both the player’s and opponent’s screens must always show identical game state information. The display must accurately present game details including small blind, big blind,

player balances, pot size, and chosen actions. The LED turn indicator must clearly illuminate the current player's turn.

3.4.2 Verification

To verify performance, power on the system and use a stopwatch to confirm the display activates within one second of boot. Trigger an action and employ a timer to make sure the screen refreshes within one second. Sit one foot from the device under typical indoor lighting to confirm readability of text and graphics. Observe both screens throughout gameplay to confirm they show the same content. Open the Arduino IDE serial monitor and compare printed values for blinds, balances, pot, and actions against what appears on the display. Finally, press any action button and verify that the LED turn indicator turns off for the current player and on for the next player. By performing these tests, we verified that our requirements for the display subsystem were true.

4. Costs and Schedule

4.1 Parts Costs

Part	Manufacturer	Quantity	Retail	Our Price
0.96 Inch OLED I2C IIC Display Module 12864 128x64 Pixel SSD1306	Hosyond	2	$5.99	$5.99
Cherry MX Hyperglide PCB Mount Switches	Cherry GmbH	10	$6.66	$6.66
Diode	Lite-On Inc.	1	$0.27	$0.00
USB Adapter	HiLetgo	1	$7.39	$7.39
9V Battery	Procell	1	$2.37	$2.37
Resistor (10k)	Stackpole Electronics Inc	19	$1.90	$0.00
ESP32-WROOM-32	Espressif	1	$5.49	$0.00
Linear Voltage Regualtor (LM1117DT)	Texas Instruments	1	$0.68	$0.68
		Total	$30.75	$23.09

Figure 9: PokerBuddy Parts List (per device)

The physical cost of designing the PokerBuddy is perhaps the cheapest part of the whole project. The display that we are using costs about $3, and we will be using two per device. The switches will come out to about $7 per device. We will be using a 9V battery, which we will approximate to be around $2. This brings the estimated cost for the physical components to around $23.70. This cost analysis is fully based on what we paid for our final product. Broken products and redesigns contributed substantially to our overall costs. Also, since we were able to utilize ECEB, we were able to get many parts for free, like the ESP32, diodes, and resistors. Doing this, we saved around $7 per device.

4.2 Labor Costs

Over a concentrated six-week timeline, our three-person team devoted roughly 15 hours per week each to this project, totaling 90 hours of effort per individual and 270 labor hours overall. At an average rate of

$50 per hour (reflecting our fully-burdened personnel cost), the projected labor investment amounts to approximately $13,500. Figure 9 establishes the core of our budgetary framework and will guide our resource allocation and milestone planning throughout the engagement.

4.3 Schedule

Week	Work by group member
1/20	We (Austin, Vishal, and Lorenzo) brainstormed initial ideas and wrote initial posts to the web board.
1/27	We (Austin, Vishal, and Lorenzo) discussed what we were going to do for our project and came up with PokerBuddy. All of us also did lab safety training.
2/3	We (Austin, Vishal, and Lorenzo) did our soldering assignment and kept discussing how we were going to implement the PokerBuddy
2/10	We (Austin, Vishal, and Lorenzo) worked together to write a proposal and team contract.
2/17	We (Austin, Vishal, and Lorenzo) worked on the PCB design and figured out what parts we were going to use.
2/24	We (Austin, Vishal, and Lorenzo) continued to work on the PCB design and figure out what parts we were going to use. We also submitted our first round PCB order.
3/3	We (Austin, Vishal, and Lorenzo) started putting together our Breadboard design for our Breadboard Demo.
3/10	Austin and Lorenzo started to work on implementing the code for the devices. Vishal started to prototype the CAD design through sketches and different schematics.
3/17	Spring Break
3/24	Lorenzo established communication between the ESP32s and wrote skeleton code for Austin to continue working on the following week. Vishal developed the initial CAD model through Autodesk Fusion and 3D printed the first version of the enclosure.
3/31	Austin started working on using the communication code to create a flushed out working skeleton of the code for our PCB. The team (Austin, Vishal, Lorenzo) realized that our first round PCBs did not have the proper ESP32 microcontroller in the KiCAD schematic and talked about revisions for a new layout.
4/7	Austin created the second generation PCB to be sent in for 4th round order. Lorenzo finalized the code that would later be used for the final Demo.

	Vishal updated CAD model to move placement of buttons + LED diode then reprinted to evaluate final integration.
4/14	Austin and Lorenzo soldered wires onto all of our Cherry-MX switches so that we would be able to use them with our fourth round PCBs. Vishal further developed designs incorporating new design choices and measurements, modified in CAD, and printed more detailed enclosures in ECE 2070. We (Austin, Vishal, and Lorenzo) did the team contract assessment.
4/21	We received the fourth round order. Austin and Lorenzo soldered 3 PCBs and tested all of them. The group (Austin, Vishal, and Lorenzo) met at the Union to figure out exact measurements for the CAD enclosure. Vishal finalized the CAD design (removable bottom plate, sizing, color) and 3D printed it in ECEB 2070. Austin and Lorenzo put together the PCB, buttons, LED, and displays within the enclosure. The team (Austin, Vishal, and Lorenzo) got together and practiced for the final demo.
4/28	Austin and Vishal created the slideshow for our final presentation. The team (Austin, Vishal, and Lorenzo) got together to practice and finalize the slideshow for our final presentation.
5/5	The team (Austin, Vishal, and Lorenzo) worked together to work on the final report.

5. Conclusion

5.1 Accomplishments

We successfully built a fully functioning chipless poker system that integrates wireless communication, game logic, and user input through our PCB. The core features, which included turn indication, pot and balance tracking, synchronized gameplay, and tactile inputs, worked with over a 99% input recognition and reliably within an eight-foot range. We also achieved a user-friendly design with a sleek enclosure and intuitive interface.

5.2 Uncertainties

Overall, our design was built to eliminate most uncertainty. Especially with a game like poker, the only place for uncertainty should be your opponent's cards. If our design created uncertainty in player balances or pot amounts, that would completely contradict the essence of the problem we are trying to solve. That being said, there are a few things that can create slight uncertainty.

We used the device for many hours at a time, and we never ran into a battery issue. However, due to the use of disposable batteries, we had no way of tracking how much battery was left in each battery. For our use, the device always just powered on, so clearly it had enough battery. This can be an issue if you are in a serious game and your battery spontaneously dies. Aside from this, the actual durability of the design is also a source of uncertainty. What happens if someone drops the device? This is something that we couldn't thoroughly test because we didn't want to jeopardize our project demo.

5.3 Ethical considerations

In developing and using our PokerBuddy, many ethical issues have to be considered. As said in the IEEE Code of Ethics, engineers must consider the societal impact of their work and avoid harm. This ties into the idea of addiction and responsible use **[6]**. With our project in particular, excessive gambling could become an easier problem to fall into. To prevent this, we added features to set betting limits, session timers, and provide consistent reminders/warnings through our LED display to take breaks during extended periods of playing. By considering this, we ensure that our project stays morally true while being a source of enjoyment. In addition, the IEEE Code of Ethics states that we have to protect user privacy and make sure of data security **[6]**. What this means is that because we are directly using people's money as information in the game, we have to give respective control to people so that only they can access their personal balance and stack size. This will be done by having limited access along with temporary memory usage. The stack balance displayed will only contain your amount and will only be stored for the duration of the game, and will be cleared after following ethical guidelines. Through representing information in this way, we consider ethical issues and provide a solution.

In developing the PokerBuddy, we took many steps to mitigate the risk, as well as considering the IEEE Code of Ethics. As stated in IEEE, we accepted any honest criticism from TAs + professors, and we tried our best to extend our technical competence through consistent effort **[6]**. We all learned various new areas to best finish our product, going from PCB design to soldering to CAD to coding a microcontroller. We did this while constantly modifying our final product based on various suggestions, highlighting our adherence to the IEEE Code of Ethics.

By addressing and adhering to these ethical guidelines, our PokerBuddy project can be used responsibly and safely by everyone.

5.4 Future work + Broader Impact

Currently, our code is designed for a heads-up poker game (two-player one-v-one), but it is also written in a way that is modular and simple to extend. We would like to build at least three devices in total so that we can test the extended code that has three person functionality. Furthermore, in the redesign of our PCB, we decided not to go forward with the use of an accelerometer. This is a design consideration that we made to ensure the reliability of our action buttons. Given more time, we could properly test the accelerometer and implement it if we determine it is a reliable replacement for the action buttons.

We also would like to explore rechargeable batteries that display the current battery percentage so the user knows when to plug their device in without risking a shutdown midgame. Lastly, we would like to

explore larger LED displays. Given our budget constraint, we used very small displays, but a large display would make everything much easier to read, and the game experience would be much more enjoyable.

In a broad sense, this project demonstrates how embedded systems can modernize traditional games while maintaining their social nature. The PokerBuddy served as a culmination of our hardware and software education as Computer Engineering students at UIUC. It required principles from digital logic design, PCB design, microcontroller programming, and embedded communication protocols.

Throughout the project, we also learned invaluable teamwork skills -- specifically on how to manage a full development cycle, from idea to implementation. We also learned how to rapidly iterate based on testing and feedback. Collaborating as a team lets us divide the tasks based on our respective strengths and weaknesses and hold each other accountable. Ultimately, PokerBuddy not only showcased our technical skills, but proved that we are able to take a real-world idea into a tangible, working system through collaboration and creativity.

References

[1]

Espressif Systems, "ESP32-WROOM-32 Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. [Accessed: May 6, 2025].

[2]

"LM1117DT-3.3 Datasheet," [Online]. Available: https://datasheet.ciiva.com/26904/lm1117dt-3-3-26904864.pdf. [Accessed: May 6, 2025].

[3]

STMicroelectronics, "LIS3DH," [Online]. Available: https://www.st.com/en/mems-and-sensors/lis3dh.html. [Accessed: May 6, 2025].

[4]

Random Nerd Tutorials, "Getting Started with ESP32," [Online]. Available: https://randomnerdtutorials.com/getting-started-with-esp32/. [Accessed: May 6, 2025].

[5]

Random Nerd Tutorials, "ESP-NOW Two Way Communication ESP32," [Online]. Available: https://randomnerdtutorials.com/esp-now-two-way-communication-esp32/. [Accessed: May 6, 2025].

[6]

IEEE, “IEEE Code of Ethics,” ieee.org, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html

[7]

“Getting started with AutoCAD | Autodesk,” Autodesk.com, 2024. https://www.autodesk.com/learn/ondemand/tutorial/getting-started-with-autocad

[8]

“DispCtrl LED Display Programmer’s Guide V5.0.1 LED Display Programmer’s Guide.” Accessed: Apr. 03, 2025. [Online]. Available: https://www.leddisplay.se/wp-content/uploads/2015/01/textskylt-manual-programmersguide-v501.pdf

[9]

“Help,” Autodesk.com, 2025.https://help.autodesk.com/view/ACD/2024/ENU/?guid=GUID-D696A745-3984-405C-90BE-D523E78E524F

[10]

L. C. Team, “Programming an LED Display Board | LED Craft,” LED Craft | LED Signs & LED Billboards, May 29, 2019. https://ledcraftinc.com/programming-an-led-display-board/

Appendix A Requirements and Verification Table

Subsystem	Requirements	Verification	Status
Microcontroller and Processing Subsystem	The microcontroller must boot up within 1 second of power-on.	Power on the system and measure the time to first successful execution of the startup code using an oscilloscope.	Y
	Must be able to process sensor inputs and output display updates within 1 second of user action.	Use a timer to log the time between sensor reading and display update.	Y
	Uses ESP-NOW to support reliable communication between 2 devices about poker game logic.	Have both ESP32 devices turn on a light on one board from a button press on another board. Repeat for different actions and ensure consistent performance.	Y
	Processes game logic actions between devices correctly 100% of the time. This includes things like passing the turn, folding, winning the hand, calling, and checking.	Use each action button to ensure that the action done was correct. Test for different actions and situations.	Y
	Processes mathematical calculations on each device before sending them to the other device.	Perform the math operations side by side with the device to ensure that the device is doing each calculation properly.	Y
	Every time a hand ends and a new hand is started, small blind and big blind alternate, and small blind ALWAYS starts the hand.	Start the game and play a hand. Once the hand ends, the small blind alternates and switches to the other person.	Y
Power Subsystem	The system must operate on a 3.3V regulated supply powered by a 9V battery.	Attach a voltmeter to check the voltage across the battery, and after the power has been lowered by the LDO.	Y

	The battery must provide at least 4 hours of continuous operation.	With a fully charged 9V battery and power under a normal load, test to see how long the machine is able to operate and if for 4+ hours.	Y
Sensor Subsystem	Buttons must output action within 1 second of pressing.	Press each action button and use a timer to check that the action was done within 1 second of a press.	Y
	Button presses output the proper action.	Press each action button and ensure that the proper action is done with each press on both devices consistently.	Y
	Accelerometer reliably reads call and check actions, where one single tap represents a call and a double tap represents a check.	Start a game and place the user in a position to call. Then, a single tap using the accelerometer to perform the action. Do the same action for checking and see if the desired action takes place.	N
	Buttons pressed when it is not the user's turn will have no effect and not change any of the gameplay.	Press each action button out of turn on the opposite device to ensure that nothing happens to either device.	Y
	Buttons rapidly pressed still read the proper inputs so that users can update their bets quickly for better gameplay.	Start the game and press any of the chip buttons 10 times quickly. The amount deducted from your balance and into the pot should accurately be reflected, even though the buttons were pressed in quick succession of each other. Can be modeled and tried for multiple actions to fully verify.	Y
Display Subsystem	Display must power on within 1 second after the system boots.	Power on the system and observe the display activation time with a stopwatch.	Y
	Display must refresh accurately within 1 second of an action being done.	Use a timer to check to see if the display updates properly within 1 second.	Y

	Displayed text/graphics must be readable at a 1 ft distance under normal indoor lighting conditions.	Sit in front of your device as if you are to use it, and ensure that you are able to read it.	Y
	Both displays on each device display the same thing, so both the player and the opponent is able to see it.	Look at each display throughout the process of the game to ensure they are displaying the proper and same thing.	Y
	The display properly displays information on the game, including small/big blind, balance, pot, and action.	Use the serial monitor in the Arduino IDE and print/poll the values of a player. Those values that are being outputted on the serial monitor should be identical to the values on the display.	Y
	The LED turn indicator properly shows whose turn it is.	Press any action button, the LED should turn OFF the player whose turn it is and turn ON the other player's LED.	Y