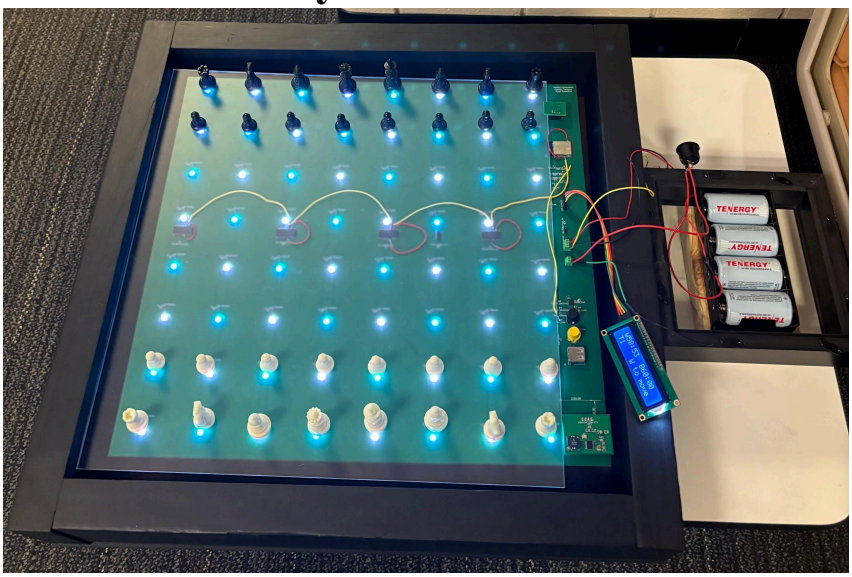
Pathfinder Chessboard

University of Central Florida



Group 35 Authors:

Andres Armendariz	Freddy Pacheco	Ryan Ramdihal
Electrical Engineering	*Computer Engineering*	*Computer Engineering*

Mentor: Dr.Sundaresh

Reviewers:

Dr. Qifeng Li
Dr. Kalpathy Sundaram
Dr. Azadeh Vosoughi
Dr.Chris Iannello

Table of Contents

Chapter 1: Executive Summary

The Pathfinder Chessboard is a smart embedded system designed to modernize physical chess without replacing it with a screen-based alternative. While online digital platforms have made chess more approachable through automated move validation, they sacrifice the social engagement of playing on a physical board. The Pathfinder Chessboard addresses this gap by integrating embedded systems into a traditional chessboard, enabling it to communicate and validate game rules, and provide interactive feedback while preserving the physical nature of a tabletop play style.

The primary objective of the Pathfinder Chessboard is to create a fully functional chessboard capable of recognizing piece positions, guiding legal moves through LED illumination, and enforcing gameplay rules in real time. The system will utilize an array of sensors embedded beneath each of the 64 squares to detect chess pieces, paired with individually addressable LEDs to indicate valid movement options. An ESP32 microcontroller serves as the central processing unit (CPU), continuously emulating the internal game state and synchronizing it with user actions on the board. The design further incorporates mode-selection allowing players to potentially switch from playing locally, to a speed chess gamemode, or to playing against an AI opponent.

The project establishes both baseline and advanced performance requirements. At minimum, the chessboard must reliably detect all piece movements, illuminate valid moves within an acceptable response time, and sustain continuous operation for at least 2 hours battery powered. A custom printed circuit board (PCB) will be developed to integrate the microcontroller, power regulation, LED drivers, and sensor interfaces into a compact and manufacturable design. Advanced goals include integration of a chess engine such as Stockfish for autonomous computer play, an implementation of a timed mode for speed chess, and audio or visual cues for move confirmation.

Functionally, the Pathfinder Chessboard enhances accessibility for beginners by reducing rule ambiguity and teaching through interaction rather than instruction. Simultaneously, it improves utility for experienced players through automated validation and a fast-paced game mode. This project distinguishes itself through its combination of real-time rule guidance and multi-mode operation.

After completing the Pathfinder Chessboard, it provided a modernized yet authentic chess experience that preserves physical gameplay while offering the adaptability of digital systems. By blending embedded design with human-centered interaction, the project demonstrates how traditional games can be reimagined for broader accessibility, educational value, and lasting engagement.

Chapter 2: Project Description

2.1 Motivation and Background

Chess is widely considered the most popular board game of all time. Originating around 600CE, it has long been regarded as a symbol of strategy and intelligence. Whether introduced through family and friends, pop culture or online media almost everyone in the world has at least heard about chess. Yet despite its popularity, learning how to play can be intimidating. Beginners often feel overwhelmed when learning rules, openings, and strategies and as a result often quit. Fortunately with the growth of digital platforms and artificial intelligence the opportunities to learn and experience chess have never been as accessible, interactive, or engaging than ever before.

However, many digital platforms sacrifice the immersive experience of playing on a physical board. Many players still value the traditional feel of moving pieces by hand but want the added benefits of digital assistance and connectivity.

The Pathfinder Chessboard bridges this gap by combining the physical experience of a traditional chess game with the functionality of an online platform. Our design consists of LED indicators embedded within the chessboard to guide players through valid moves and capture opportunities, making the game more accessible to beginners while still engaging for experienced players. This interactive feedback reduces the steep learning curve for beginners, encouraging skill development through visualization for potential move sets.

In addition to local two-player games, the integration of a computer AI and remote play offers versatility that extends the chessboard's appeal. The AI opponent provides a practice environment for solo players, while online connectivity via Wi-Fi enables real time matches with friends or competitors from anywhere. These features enhance the board's usability beyond recreational play, making it suitable for chess clubs, classrooms, and home use.

The Pathfinder Chessboard presents an intricate design requiring coordination of sensors for piece detection, LEDs for move guidance, wireless communication for online play, and custom PCB design for efficient power and signal management, showcases innovation by integrating traditional gameplay with modern interactive features.

2.2 Goals and Objectives

The primary goals of the Pathfinder Chessboard are to create a fully functional physical chessboard equipped with accurate piece detection and LED guidance. The system should allow smooth local two-player gameplay, providing real-time feedback to highlight legal moves and capture opportunities. Another key goal at this stage is to design and implement a custom PCB

that integrates the microcontroller, power regulation, sensor interfaces, and LED drivers into one cohesive system.

To meet these goals, our basic objectives are to achieve accurate detection rates for piece movement, ensure that LEDs illuminate the correct legal moves and capture squares within a timely manner, and demonstrate the stable gameplay for the duration of a full match without any subsystem failures. The chessboard must operate reliably in local two-player mode with all rules enforced, and the custom PCB should provide continuous power and coordination for all subsystems for at least 2 hours of battery-powered play.

Building upon the basic goals, the advanced goals focus on enhancing gameplay through additional features that improve user interaction and create variety. These goals include integrating a chess engine such as Stockfish to provide a computer opponent and introducing a speed chess mode complete with a timer and sound effects for move notifications.

The advanced objectives are to run the chess engine efficiently on the embedded platform, generating valid AI moves and integrating them to appear through the LED system. For speed chess mode, the objectives are to implement a timer within per move timeframe and to generate audio cues that alert players when a move is made or when time is running low.

The stretch goals for the Pathfinder Chessboard emphasize expanded connectivity and user experience enhancements. These include enabling remote multiplayer matches by synchronizing moves through a companion app, allowing a player to play against another remotely. Additional goals include incorporating customizable LED animations and a win-loss record display.

Corresponding stretch objectives include developing a companion app that can transmit moves between two Pathfinder Chessboards or between a Pathfinder Chessboard and a Wi-Fi enabled device, ensuring that gameplay remains synchronized across locations. The system should maintain minimal lag during move updates to provide a seamless remote play experience. Other objectives include creating dynamic LED animation patterns to highlight moves or outcomes and integrating a win–loss record feature that updates automatically at the conclusion of each game. These stretch objectives would elevate the Pathfinder Chessboard beyond local play, blending traditional physical chess with modern digital connectivity.

2.3 Features and Functionalities

The Pathfinder Chessboard is designed to combine the tactile experience of traditional chess with interactive benefits of modern technology. A central feature of the system is its LED-guided move indication, where embedded LEDs beneath each square highlight legal moves when a piece is lifted from the board. To improve clarity, valid moves will be displayed in one color, while capture opportunities will be distinguished by a different color. This guidance system enhances accessibility for beginners, reduces errors in gameplay, and creates a more engaging experience for all players.

The board supports three primary modes of play. First, a local two-player mode allows for traditional in-person matches. Enhanced with real-time LED guidance. Second, a computer opponent mode integrates a chess engine, such as Stockfish, enabling solo play. Third, a speed chess mode to test a player's quick thinking. A mode selection switch will allow users to toggle seamlessly between these modes.

Additional functionalities include automatic piece detection using magnets attached to the chess pieces and hall sensors embedded in each square to reliably register the presence and movement of chess pieces. This system ensures synchronization between physical actions and digital processing. A timer for each player may also be implemented for a player's move.

Potential extended functionalities may include game logging, which would record moves for post-game review, as well as a companion application for an online multiplayer feature.

2.4 Existing Products/Prior Work

2.4.1 SMART Chess Board

While researching for our design, we came across a previous Senior Design project from 2023, where the group also made a smart LED chess board titled “SMART Chess Board”. Their design made use of hall effect sensors to determine where and when a piece was lifted, and would then indicate on the chess board where valid moves were using LEDs. This group also made their own chess library to avoid having to alter an existing one, but found it difficult as time went on. This project gives us an understanding of one way to implement our systems, though we have chosen to alter ideas, such as adding in the ability to play against a computer player, and also deciding to use an existing open-source chess library that we can change as needed.

2.4.2 DIY Super Smart Chessboard

Another source we came across, this time while trying to develop our idea of being able to connect the chess board with Wi Fi, was an existing hobbyist project, named the “DIY Super Smart Chessboard”. This project makes use of a LED strip to light the board similar to our plan, though it is controlled by manually selecting the space the player wishes to move to through the use of buttons, which then light the LEDs, then after a manual confirmation, the player moves the piece. To make sure the move is legal in chess, libraries are implemented. While this program lacks a sensor to indicate when a piece is moved, we found value in reading how they implemented their ability to play against a computer player and human player using Wi-Fi.This project uses Wi-Fi capabilities granted by the raspberry pi board and various libraries it has to connect with other Super Smart Chessboards to allow online play. Though we do not plan to use the raspberry pi board, this project still provides an outline of what we are striving for, and also shows us one way to do it, which we can alter as needed.

2.5 Engineering Specifications

Table 2-1: System-Level Specifications

Parameters	**Target Value**	**Notes**
Chess Board Dimensions	Main chess board:<20x20x2 in Side box: <5x5x2 in (Length x Width x Height)	Standard chessboard size with space for electronic components.
Battery Life	≥ 2 hours	Duration of normal gameplay session on one charge
Manufacturability (Cost)	< $350	Budget target for prototype parts
Temperature	< 100 degrees Fahrenheit	Safe for indoor operation
Modes	3	The path finder chessboard should have a minimum of 3 different modes the user can select.

Table 2-2: Component-Level Specifications

Component	**Parameter**	**Subsystem and Description**	**Target Value**
LED	Response Time	Seamless movement	< 1 second
Voltage Regulators	3.3V & 5V stability	Oscilloscope measurement	±5% variation

Sensors	Detection Speed	Consistent detection	< 500 msec

2.6 Hardware Block Diagram

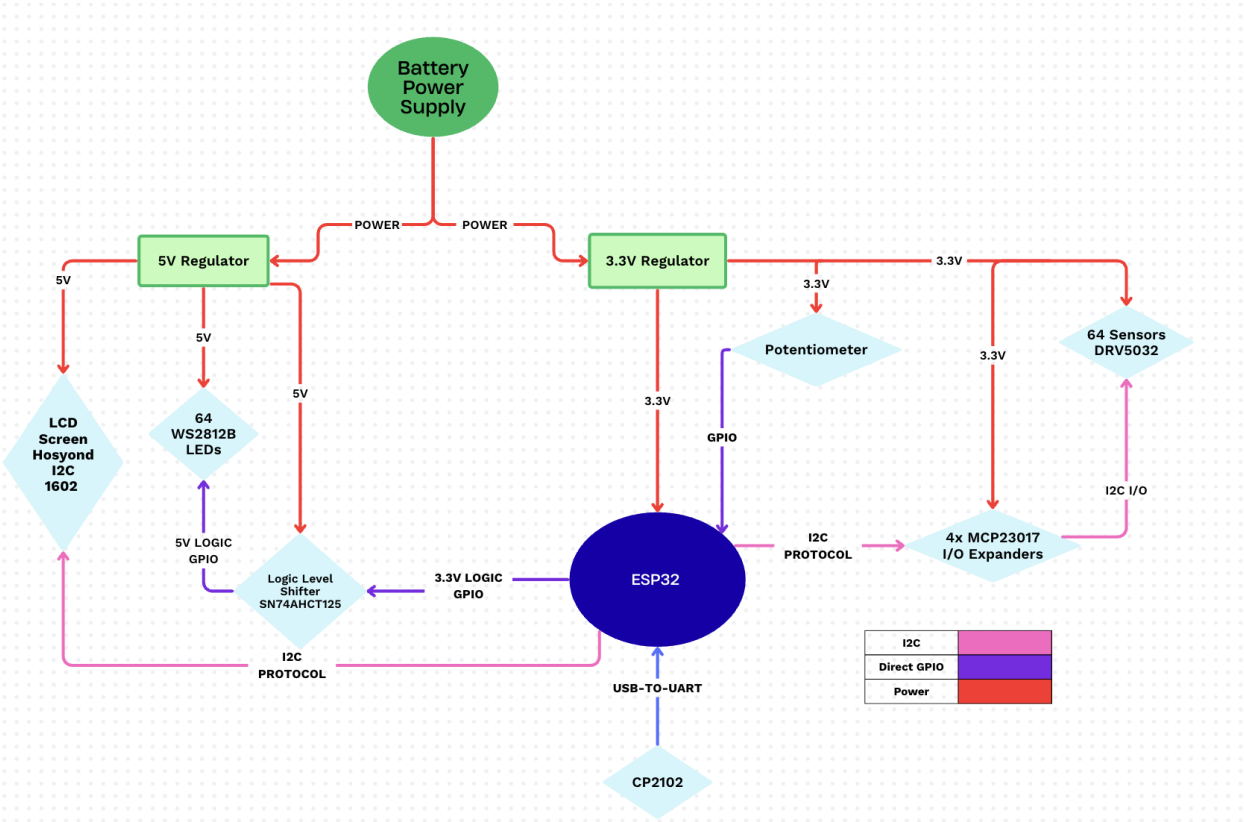
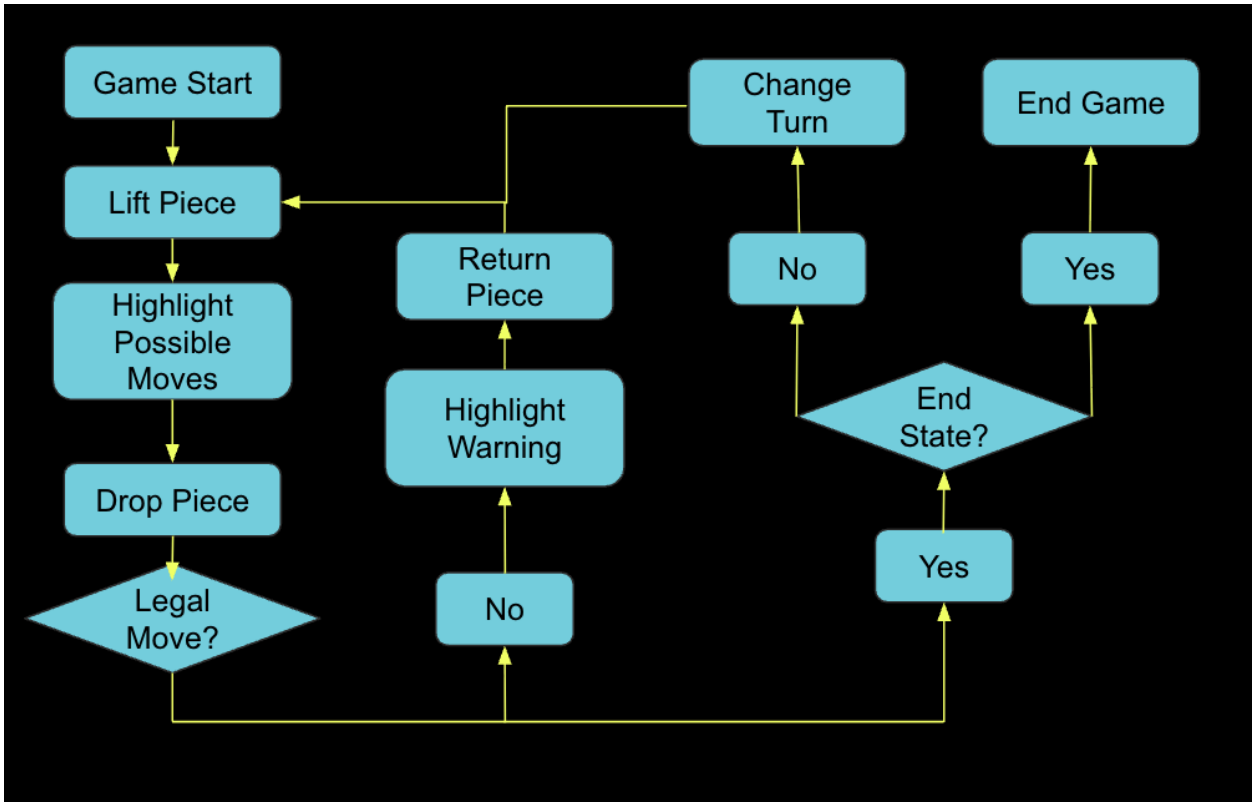


Table 2-3: Component Investigation Table

Block	**Status**
Battery Power Supply	Acquired
3.3 Voltage Regulator	Acquired
LCD Screen	Acquired
Microcontroller	Acquired
LEDs	Acquired
I/O Expanders	Acquired
Opponent Computer (Chess Engine)	Acquired
3 mode switch	(Software based using potentiometer)
64 Sensors	Acquired
Potentiometer	Acquired

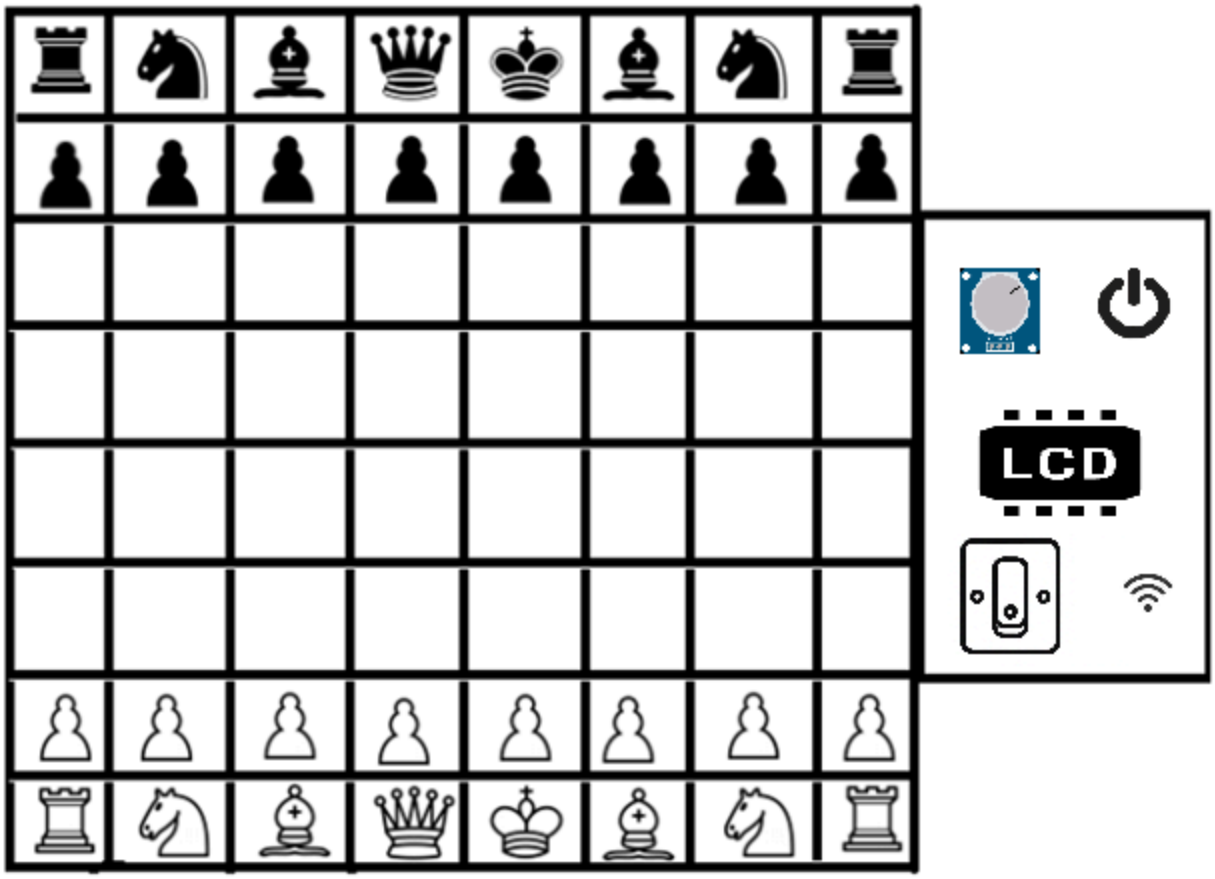
2.7 Software Diagram/Flowchart



2.8 Prototype Illustration/Blueprint

The image below is a prototype illustration of the PathFinder Chessboard.The illustration does not show the height of the box which would be around 1.5 inches. The dimensions of the chess board measure around 18x18x2 for the main chessboard and 4.5x4.5x2 for the box to the right of it. Underneath each of the 64 tiles of the Chessboard will be a magnetic sensor and a LED mounted on the PCB. The LEDs would be daisy chained with the microcontroller, with the regulators and other components embedded in the PCB as well. To the right of the chess board will be the on/off switch, 3-way mode switch, LCD screen, potentiometer used as a timer, and wi-fi antenna.

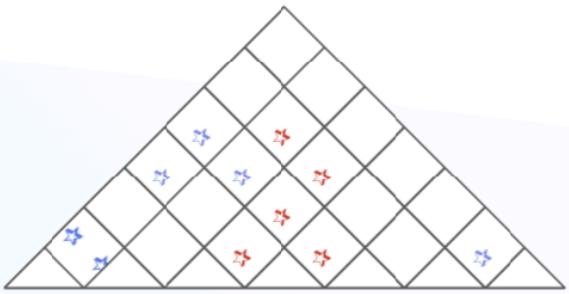
Figure 2-1: Component-Level Specifications



2.9 House of Quality

Key

Positive Correlation	☆	↑
Strong Positive Correlation	☆☆	↑ ↑
Negative Correlation	☆	↓
Strong Negative Correlation	☆ ☆	↓ ↓



Consumer Requirements	LED Response time	Sensor accuracy	Component Costs	Chess engine integration	Power Consumption	Material Quality	Weight
	Engineer Requirements						
Accessibility	↑ ↑	↑ ↑		↑		↑	↓
Power Efficiency					↑ ↑		
Speed	↑ ↑	↑ ↑		↑	↑		
Affordable Cost			↓ ↓				
Portability							↓ ↓
Durability					↑	↑ ↑	↓
Target for Requirements	≤ 250ms	≥ 90%	≤ $100	-	≤ 5 W	scratch resistant	≤ 10 lbs

Chapter 3: Research and Investigation

3.1 Hardware Technology Comparison and Selection

To develop an effective engineering project, it is crucial to research and investigate possible technologies that can be used. Different technologies have to be compared amongst one another to see which technologies are appropriate and specialized to the project's constraints and specifications. Limiting factors between technologies include how they interact, power requirements, cost and overall performance within the project's constraints and specifications. The choice in technology is an essential part of ensuring a successful and functional project. This section discusses and compares different technologies and explains the rationale behind the component selections made to meet the specific needs of the Pathfinder Chessboard project.

3.2 Hardware Part Comparison and Selection

3.2.1 Light Sources

A core specification of the Pathfinder Chessboard is the illumination system. The illumination system serves as the user interface for conveying optimal move guidance. The selected component had to satisfy a rigid set of criteria to achieve this goal. Such criteria included that the light source must be bright enough to remain clearly visible through the acrylic layer, but not so bright as to possibly harm the user. Additionally, it must also have the power efficiency necessary to keep the battery preserved for at least 2 hours. Finally, compatibility with the ESP32-S3 microcontroller is also a critical constraint that allows the illumination system to communicate with the rest of the components.

Three potential light sources were evaluated for this project: electroluminescent displays, light emitting diodes, and laser diodes. These components were selected due to their relevance in similar projects. The following sections describe each component, how they work, and the variating types of each component.

3.2.1.1 Electroluminescent Display

The Electroluminescent (EL) Display is a type of flat panel display created by putting a layer of electroluminescent material between two layers of conductors. When current is applied the electroluminescent material starts to glow. This type of display is valued for its uniform light output, thin profile and low power consumption. EL Displays require a high voltage AC power source in order to power them and usually have lower brightness and shorter lifespans compared to other light sources, which limits their use in applications that require higher visibility or color control. Although EL displays can be made thin and flexible they do require more circuitry to operate, which can increase the system size and complexity compared to other light source alternatives.

The most common types of EL Displays are Powder EL, Thin-Film EL, and Thick Film EL. The most common type among these is the Thin-Film Electroluminescent (TFEL) display.The TFEL display is nichely used in industrial, military, and automotive applications due to its extreme durability and ability to operate in extreme environments. Powder EL displays, on the other hand, are typically found in simpler devices such as backlighting or night lights due to their low cost and ease of production. Thick Film EL displays combine features from both the powder and TFEL display to create a brighter and more robust screen. While Thin-Film and Thick Film EL displays offer greater durability and performance they are generally more expensive to manufacture compared to the Powder El types.

3.2.1.2 Light Emitting Diodes (LEDs)

Much like the EL display Light Emitting Diodes (LEDs) produce light when a current is applied through the forward bias direction. The light is produced by the movement of electrons through the semiconductor junction, which releases energy that can be seen as visible light, invisible infrared, or ultraviolet radiation. The wavelength of the light depends on the semiconductor material and its bandgap energy. Different materials are able to produce different wavelengths with the two extremes being infrared on the lower wavelength and ultraviolet on the longer wavelength. Advantages with LEDs include their high efficiency, long lifespan, and small size which allows them to be used in a wide range of different applications. With power LEDs unlike EL displays do not require a high AC voltage in order to operate but instead operate on a low DC voltage. LEDs are also generally small which in turn makes them energy efficient and easier to control for projects where many different lights are needed.

The three most common types of LEDs are RGB, UV, and infrared. RGB LEDs can emit a single color of light and are the most common type found in indicator lights and simple displays. The RGB LEDs can also mix different shades of red, green, and blue in order to achieve different colors. By controlling the intensity of each color, these LEDs can produce millions of color combinations, which makes them useful for color coding lights. Another type of LED is the UV LED. These are commonly used in sterilization and disinfecting applications and can also be used in printing or other black light applications. These types of LEDs emit an ultraviolet light that can kill bacteria or viruses and can cause fluorescent or phosphorescent materials to glow.Similar to the UV LEDs infrared LEDs emit a light that is on the opposite side of the electromagnetic spectrum. Infrared LEDs however produce a light that is invisible to the human eye.These particular LEDs are commonly found in cameras and proximity sensors due to their ability to be invisible to the human eye.

3.2.1.3 Laser diode

Laser diodes, unlike LEDs, have a coherent, high intensity unidirectional beam of light that can travel much longer distances. Laser Diodes work similarly to LEDs as both use a PN junction made of semiconductor materials to produce light when an electrical current flows through them. However, laser diodes include mirrors inside an optical cavity which allows the

light to bounce back and forth in order to amplify itself. These components allow all the light waves to be in phase with each other and travel in the same direction which results in a highly concentrated beam that can travel long distances. Laser diodes require precise current control to operate properly as too much current can damage the device while too little will not produce a laser. Laser diodes also produce more heat compared to LEDs due to their higher power density which can oftentimes complicate systems. Despite all these requirements laser diodes are rated highly for their ability to produce extremely focused and powerful beams of light making them essential for implementations that require precision and long range transmission.

The most common types of laser diodes are Fabry-pérot, Distributed Feedback(DFB), and Vertical-Cavity Surface Emitting Lasers (VCSELs). Fabry-pérot are the simplest and most common type. Mirrors at both ends of the internal cavity are used to reflect the light back and forth. This type is commonly found in optical storage like CD, DVD and blue ray players as well as in laser pointers and short range communication systems. DFB laser diodes use diffraction grating instead of mirrors to produce a stable, single wavelength output which makes them ideal for fiber optic communication. VCSELs emit light perpendicular to the surface of the chip which makes them easier to manufacture and test. They are commonly found in computer mice, and facial recognition systems like face ID, and high speed communication.

Table 3-1: Illumination Component Comparison Table

Component	**Power Consumption**	**Color/Brightness Control**	**Cost**	**Lifespan/Durability**
Electroluminescent display	Low	Basic	Moderate	Medium
LED	Very Low	Excellent	Moderate	Long
Laser Diodes	Medium	Good	High	Medium

3.2.1.4 Illumination Component Selections

To determine the most suitable lighting technology for the Pathfinder Chessboard three components were evaluated: Electroluminescent displays, LEDs and Laser Diodes. Each component was assessed based on power consumption, color and brightness control, cost and lifespan and durability. Looking at the table we can see an array of the different options from which to pick the light source to be.

From this data we can conclude that the component that outperforms in all categories are the LEDs. They provide the optimal combination of very low power consumption, excellent color and brightness control, moderate cost that fits within the project budget and long lifespan that ensures reliability. The ability to use addressable RGB LEDs allows for color coding where different colors can represent legal moves, suggested optimal moves, and other game states. The

addressable RGB LEDs also have excellent compatibility with the ESP32-S3 microcontroller through GPIO control making them the clear choice for the Pathfinder Chessboard illumination system. For this project we specifically chose the smart WS2812B LEDs because of their ability to be daisy-chained which reduces the number of PCB traces and wires significantly. The WS2812B LEDs can also be addressed individually which allows for full control of which colors they can display and where. The individual addressability also means that they only need a single data pin from the microcontroller which simplifies the circuit design even further.

3.2.2 Sensors

In order for the illumination system to function there must be some sort of trigger. In our case this trigger will be set off by a sensor underneath the casing of the chess board. When picking a sensor we must consider several important factors to ensure reliable detection of chess piece movements. The sensor must be able to accurately detect what piece is placed on or moved from a square without being affected by the color or material of the chess pieces. It should also be compact enough to fit beneath each of the 64 squares on the chessboard without adding too much thickness to the overall design. The sensor should also be able to not pick up any false readings that could occur due to playing. Along with these constraints the sensors must also be cost effective and integrate easily with the ESP 32 microcontroller.

3.2.2.1 Hall Effect Sensors

Hall-effect sensors provide a reliable, contactless method for detecting the presence of a magnetic field, making them a strong candidate for determining whether a chess piece is placed on a specific square. They operate by generating a voltage across the sensing element when exposed to a magnetic field, enabling simple digital detection when a magnet is positioned above the sensor. This allows each square to register piece placement with high accuracy and minimal susceptibility to mechanical wear, since Hall-effect sensors rely solely on magnetic flux and have no moving parts.

However, adopting this sensing approach requires attaching a magnet to every chess piece, which increases both the material cost and the complexity of piece manufacturing. Because the sensors only detect magnetic fields, the orientation, strength, and placement of the magnets must be consistent across all pieces to ensure reliable detection. Misalignment or inconsistent magnet strength could result in false triggers or unreadable board states, especially during rapid piece movement.

While both linear and switch-type Hall-effect sensors are available, switch-type devices are better suited for this application due to their simple binary output and lack of need for analog signal processing. This makes it straightforward to integrate with the ESP32-S3 and the MCP23017 expanders, reducing firmware complexity and electrical noise concerns.

Overall, Hall-effect sensors offer clear advantages in terms of reliability, long-term

durability, and sensing robustness. However, their use also introduces additional hardware requirements and cost considerations that must be weighed carefully when determining whether they are the most practical sensing option for the full-scale chessboard implementation.

3.2.2.2 Capacitative Sensors

Capacitive sensors work similarly to Hall effect sensors except instead of a magnetic field the capacitive sensor works by detecting an electrostatic field. These sensors measure changes in capacitance that occur when a conductive or dielectric object comes near the sensor surface. The sensor then creates an electric field between two conductive plates and changes the capacitance by altering the distribution of the electric field lines. Capacitive sensors are used to detect the presence or absence of virtually any object regardless of material. However capacitive sensors could be prone to false triggers from hands, cables, or other conductive materials.

There are two main types of capacitive sensors: self-capacitance and mutual capacitance sensors. Self-capacitance sensors use a single electrode and measure the capacitance between the electrode and ground, making them simpler and less expensive but more susceptible to noise. Mutual capacitance sensors use two electrodes, a transmitter and receiver and measures the capacitance between them. The capacitance decreases as an object gets closer to touching it by disrupting the electric field between them.

3.2.2.3 Inductive Sensors

Inductive sensors are much like capacitive sensors in that they can detect material without contact, but unlike capacitive sensors they can only detect conductive metals. Inductive sensors work by generating an electromagnetic field and detecting changes in that field that are caused by the presence of metallic objects. The sensor contains a coil of wire that carries an alternating current, which creates a magnetic field around the coil. When a conductive object enters this magnetic field eddy currents form on the metal surface producing a magnetic field that opposes the coils magnetic field. The changes in inductance of the coil can then be detected by the sensor's circuitry. Inductive sensors are highly reliable and have fast response times, making them suitable for many applications. They are commonly used in position sensing and metal detection applications.

However, inductive sensors have some important limitations. They can only detect conductive metals such as iron, steel, aluminum, copper, and brass, which means they cannot detect non metallic materials such as wood, glass or plastic. Depending on the sensor size and the type of metal being detected, detection ranges from a few millimeters to several centimeters. Inductive sensors can also be affected by electromagnetic interference from nearby electrical equipment.

3.2.2.4 Piezo Electric Sensors

Piezo electric sensors work by applying mechanical pressure on the sensor which then releases an electric signal due to the piezoelectric effect. It also does not need an external voltage source or current source in order to operate. The piezoelectric effect occurs only in certain crystalline materials such as quarts, ceramics and some polymers. Mechanical stress causes the internal electric dipoles to realign and produce a voltage across the material. Piezo electric sensors are very durable and have long operational lifespans since they have no moving parts. Additionally, piezoelectric sensors have fast response times and can detect changes in pressure almost instantaneously.

Three common types of piezoelectric sensors include piezoelectric accelerometers, piezoelectric force sensors, and piezoelectric pressure sensors. Piezoelectric accelerometers measure acceleration through the stress of a force on a piezoelectric object. These are commonly used in vibration monitoring and motion detection applications. Piezoelectric force sensors measure the magnitude of an applied force by detecting the voltage generated when pressure is applied to the piezoelectric material. They are used in applications such as impact testing or tactile feedback systems. Piezo electric pressure sensors measure changes in pressure that are caused by the mechanical stress on the sensor. These sensors are used in dynamic pressure measurements such as in fluid or gas systems.

However, Piezoelectric sensors are best suited for detecting dynamic changes in pressure rather than static pressure. This means they generate a voltage signal when a force is applied or removed but do not maintain a continuous output while constant pressure remains on the sensor. This characteristic makes them ideal for detecting impact, vibrations or other pressure changes, but limits their use in applications that require continuous monitoring of static loads or constant pressure states.

3.2.2.5 Ultrasonic Sensors

Ultrasonic sensors are the most unique sensor out of all the other sensors being compared. Like most of the other sensors the ultrasonic sensor can output signals without needing to be touched. What separates the ultrasonic sensor from other sensors is the way it processes information. Ultrasonic sensors measure the distance to an object by using ultrasonic sound waves that reflect back to the sensor allowing it to measure distance. By measuring the time it takes for the sound wave to travel to the object and back, the sensor can calculate the distance using the speed of sound in air. Ultrasonic sensors can detect most materials except any type of object that is highly sound-absorbent like carpet or foam.

However, ultrasonic sensors have several limitations that affect their performance in certain applications. The accuracy of distance measurements can be affected by objects in the way and other environmental factors. Ultrasonic sensors also may struggle to detect objects with angled surfaces that scatter sounds away from the receiver. Ultrasonic sensors also have relatively slow response times compared to other types of sensors because they must wait for the sound wave to travel to the object and back. They also require external power to generate the

ultrasonic pulses and process the return signals. Additionally multiple ultrasonic sensors operating near each other can interfere with one another if they emit pulses at the same frequency, which can lead to false readings.

3.2.2.6 IR Reflective Sensors

Infrared (IR) reflective sensors operate by emitting IR light upward and reflected back into a photodiode. In theory, we can use IR reflective sensors to sense the bottom of our chess pieces, making them a potential alternative to Hall-effect sensors because these sensors do not require magnets inside the chess pieces and can provide fast and continuous detection. Since reflective IR modules are inexpensive and simple to interface with a microcontroller, they may initially seem beneficial toward our smart chessboard application

However, IR reflective sensors may introduce several reliability issues that can make them less suitable for a system that will require stable, repeatable detection across all 64 squares of a chessboard. One major limitation is how sensitive they are to ambient lighting conditions. Bright overhead lights, sunlight, or even the LEDs we will have for visual feedback of our board may alter reflection levels measured by the sensor. This can result in inconsistent readings that would require frequent recalibrations to adjust.

Mechanical alignment of the sensor will also impact the measurement of light sensed. The sensor relies on detecting light at a specific angle, as such, the chess piece must sit directly over the sensor's optical path. Small positional shifts can alter reflectivity enough to cause missed detection or false positives. Implementing these sensors would also require translucent sections in the board to allow infrared light to pass through, complicating the chess board's construction and reducing its visual quality.

Overall, while IR reflective sensors are technically capable of detecting the presence of a chess piece, their dependence on lighting and positioning make them significantly less reliable than Hall-effect sensors. Our chessboard must operate consistently in various lighting environments and with any standard chess piece design, making the infrared reflective sensors not an ideal choice.

Table 3-2: Sensor Comparison Table

Sensor Component	**Triggered by**	**Detects**	**More likely to cause False triggers**	**External Power required**	**Cost**
Hall effect	Magnetic fields	Magnetic materials	Low - only responds to magnetic fields	Yes	Moderate

Capacitative	Electrostatic fields/changes in capacitance	Any conductive or dielectric material	High - reacts to hands, moisture or cables.	Yes	Low to moderate
Inductive	Electromagnetic field changes	Conductive metals only	Moderat - can pick up interference	Yes	Moderate
Piezo Electric	Mechanical stress or vibration	Changes in pressure force or acceleration	Low - only triggers from impact or movement	No	Moderate to high
Ultrasonic	Sound wave reflection	Most solid and liquid materials	Moderate - echoes and angles can cause errors	Yes	Moderate
IR Reflective	IR light from an object	Presence of object above the sensor	High, ambient light interference	Yes	Low to moderate

3.2.2.7 Sensor Selections

To determine the most suitable sensor technology for the Pathfinder Chessboard, five sensor types were evaluated: Hall effect sensors, capacitive sensors, inductive sensors, piezoelectric sensors, and ultrasonic sensors. Each sensor was assessed based on what triggers it, what materials it can detect, susceptibility to false triggers, external power requirements and cost.

In order to determine what type of sensor will be used in this project we must take into account all the advantages and disadvantages of each sensor type. Hall effect sensors require magnets to be attached to each chess piece, which would increase the cost but would provide reliable detection with low false trigger rates. Capacitive sensors can detect any object without piece modification but have high susceptibility to false triggers from hands, moisture or cables during gameplay. Inductive sensors only detect conductive metals and would require metal pieces or modifications, making them less practical. Piezoelectric sensors do not require external power and have low false trigger rates, but they only detect dynamic pressure changes rather than static presence. This would require additional software to track the board state. Ultrasonic sensors can detect most materials but having the PCB underneath the board would make it tricky for the ultrasonic sensors to have precise detection.

From this data we can conclude that Hall effect sensors provide the best combination of reliable detection, low false trigger rates, moderate costs, despite requiring magnets to be attached to the chess pieces.

3.2.3 Displays

In order to interact with the user the Pathfinder Chessboard must also have a display in order to serve as the main visual output of the system. It will show information such as player turns, detected piece positions, and other useful information. In order to satisfy the specifications of this project we have to consider how these screens can communicate with the microcontroller. Size and power consumption also need to be taken into consideration when choosing a display. choose a display that can be programmed alongside our micro controller.

3.2.3.1 Organic Light Emitting Diodes (OLEDs)

Organic light emitting diodes (OLEDs) use a similar way to function as LEDs but are commonly used to produce high quality displays. OLEDs are able to display text, graphics and full color images.OLEDs can come in many different shapes in sizes but for our project we will be focusing on the smaller variants of OLED displays.These types of displays typically use high speed serial protocols like SPI or I2C. Furthermore it can be noted that OLED screens are also able to display in monochrome up to full color but this is more commonly seen in bigger OLED screens.

3.2.3.2 Liquid Crystal Display (LCD)

Liquid crystal displays (LCDs) are another common type of screen used in embedded systems and PCB projects. The LCD screen can interact with a microcontroller by using I2C, SPI, or parallel communication protocols. They consume relatively little power, making them an energy efficient option. One limitation however is that LCDs are typically limited only to display text and characters. Due to their simplicity and low cost LCD screens are a popular choice for many basic user interfaces.

3.2.3.3 Seven-Segment Display

The seven-segment display is the simplest and most affordable display option among those evaluated. It uses GPIO pins or I2C for communication and is designed primarily for showing numeric characters. The sizes of the 7 segment display are typically small but ease of integration makes it a practical choice compared to LCD or OLED displays.

Table 3-3: Display Comparison Table

Display	**Communication Protocol**	**Size (In)**	**Power Consumption**	**Display Capability**
OLED	SPI, I2C, Parallel	0.96-3	Low to medium	High

LCD	UART SPI, I2C	16x2 to 20x4 characters	Low	medium
7 Segment Display	GPIO, I2C	Small	Very low	Low

3.2.3.4 Display Component Selections

For the Pathfinder Chessboard design we decided to go with the LCD screen for a variety of reasons. We specifically choose an I2C based LCD screen to communicate with microcontrollers in order to efficiently allow for less pins to be used on the microcontroller. The LCD screen can come in different sizes and can fit on the side compartment of the Pathfinder Chessboard comfortably. The selected display also only needs to be able to display numbers or characters so there is no need for a display that has graphics or full color images. The LCD screen is more versatile than the 7 segment display but does not have the undesired added cost and power management complexity OLED display. This makes the LCD screen a perfect middle ground to take when picking a display.

3.2.4 Microcontrollers

The microcontroller is the brain of our system and controls and reads the data imputed by the sensor and outputs data to the LEDs.The microcontroller also has to be able to transmit or receive data from the potentiometer, buttons, and LCD screen. In order for the microcontroller to fully integrate with the system it will have to have I2C and analog communication protocols in order to communicate with the rest of the components. As a stretch goal the microcontroller should also have on board wi-fi if the user wishes to play with someone online.

3.2.4.1 ESP32

The ESP32, developed by Espressif Systems, is a dual-core 32-bit microcontroller that can have a clock speed up to 240 MHz. The ESP32-S3 in particular, offers up to 45 programmable GPIO pins, along with multiple communication interfaces including SPI, I2C, UART, and I2S. This extensive set of peripherals provides support for our sensors, displays, and LED chain making it well-suited for our Pathfinder Chessboard. Additionally, the ESP32 includes built-in Wi-Fi and Bluetooth capabilities to leverage into future expansions of the project, such as wireless connectivity.This makes the ESP32 highly adaptable for both the current chessboard design and future stretch goals, providing an avenue for expanding functionality without changing the underlying hardware.

The device is especially suitable because it can handle multiple concurrent tasks effectively. One core can continuously scan the 64-square sensor grid, while the second core manages real-time LED updates and communicates with the chess engine to validate moves. The dual-core architecture ensures that the system remains responsive even during complex operations, such as simultaneous piece movement detection and LED animation. Furthermore, the ESP32's GPIO count and flexible pin multiplexing allow for easy integration with I²C expanders like the MCP23017, supporting scalable sensor configurations without exhausting the available pins.

While the ESP32 consumes slightly more power than simpler microcontrollers such as the Arduino Uno or MSP430, its performance, peripheral support, and scalability make it the most suitable choice for the Pathfinder Chessboard. It offers a balance of processing power, real-time control, and connectivity, ensuring reliable operation for both sensor monitoring and LED display management.

3.2.4.2 Arduino Uno

The Arduino Uno, powered by the 8-bit ATmega328P microcontroller, operates at a clock speed of 16 MHz. It provides basic communication interfaces such as UART, SPI, and I2C, with a total of 14 digital and 6 analog GPIO pins.

While it is well-known for simplicity, affordability, and large development community, the Arduino Uno is not optimal for the Pathfinder Chessboard system. The board's limited memory and single-core 8-bit architecture restrict its ability to perform multiple tasks concurrently. Handling simultaneous sensor readings, move validation, and LED control would likely result in timing conflicts or slow response.

3.2.4.3 MSP430G2553

The MSP430G2553, developed by Texas Instruments, features a 16-bit architecture running up to 16 MHz and emphasizes ultra-low power consumption. The MCU includes common interfaces such as UART, SPI, and I2C, and operates within the 1.8 V - 3.6 V range.

While the MSP430 excels in low power consumption, its limited processing speed makes it unsuitable for the computational requirements of the Pathfinder Chessboard. The project's need to scan multiple sensors, manage LCD updates, and process game logic in real time, exceeds the MSP430's capabilities.

3.2.4.4 Raspberry Pi Pico

The Raspberry Pi Pico uses the RP2040 dual-core ARM Cortex-M0+ processor running at 133 MHz, making it a powerful yet low-cost option for embedded projects. It has 26 GPIO pins, whilst also supporting multiple communication interfaces, including I2C, SPI, and UART. This allows the Pico to interface with the 64 hall-effect sensors through I2C expanders like the MCP23017. Its dual core also allows one core to control LED updates, while the other core focuses on sensor polling, enabling a smooth real-time operation for both the sensor grid and WS2812 LED chains.

However, the Pico does have limitations for our smart chessboard. Unlike the ESP32, it does not have built-in Wi-Fi or Bluetooth, so adding wireless functionality would require further external components and limit our stretch goal. Driving a chain of WS2812 LEDs with precise timing may also be extraneous since the Pico does not have a dedicated peripheral for LED control and timing support like the ESP32.

The Pico is extremely affordable with many libraries available. Its lower power consumption and modern 32-bit architecture still makes it reliable for sensor and LED control. With a proper design, the Pico can manage all 64 chess tiles using I2C expanders while simultaneously driving the LED strips.

In summary, the Raspberry Pi Pico is a cost-effective microcontroller for the Pathfinder Chessboard. It has sufficient GPIO pins but lacks built-in wireless connectivity and the need for careful LED timing management make it less convenient than the ESP32. For a fully featured chessboard with wireless features and real-time LED updates, the Pico is viable but requires additional effort and careful software planning.

3.2.4.5 Teensy 4.1

The Teensy 4.1 is a high-performance microcontroller built around a 600 MHz ARM processor, making it one of the fastest grade microcontrollers available. It provides up to 55 digital I/O pins and supports communication protocols such as I2C, SPI, UART, and USB. This combination of high-speed processing and large GPIO availability make this microcontroller highly capable of handling complex embedded applications, including simultaneous sensor pilling and LED control.

For the Pathfinder Chessboard, this microcontroller offers exceptional performance for real-time operations. Its high clock speed allows rapid scanning of the 53 square sensor grid while still driving the WS2812 LED strips with precise timing. The large number of GPIO pins also makes it possible to interface directly with sensors or expanders without running into pin limitations, providing flexibility in hardware layout.

Despite its impressive capabilities, there are some trade-offs to consider. Unlike the ESP32, the Teensy 4.1 does not include built-in Wi-Fi or Bluetooth, which limits wireless expansion without adding external modules. It also has slightly higher power consumption due to its high-performance CPU, which may be a consideration for battery-operated designs. Additionally, its advanced features and higher speed make software development more complex compared to simpler microcontrollers like the Arduino Uno or ESP32.

Overall, the Teensy 4.1 is an excellent option for the Pathfinder Chessboard if maximum processing power and precise timing control are the primary priorities.However, the lack of built-in wireless functionality and slightly higher development complexity make it less immediately convenient than the ESP32.

Table 3-4: Microcontroller Comparison Table

Micro controller	Processing Power	Clock Speed	Development Environment	Communication Interface	Operating Voltage

ESP32-S3	Dual-Core RTOS support (HIGH)	MAX speed 240 MHz	Arduino IDE ESP-IDF	I2C I2S SPI UART ADC	2.3 V - 3.6 V (3.3V)
Arduino Uno (ATmega328P)	8-bit Single Core (LOW)	16 MHz	Arduino IDE	I2C SPI	5V
MSP430G2553	16-bit (MODERATE)	16 MHz	Code Composer Studio	I2C SPI UART	1.8 V - 3.6 V (3.3V)
Raspberry Pi Pico	Dual-Core ARM Cortex-M0+ (HIGH)	133 MHz	Arduino IDE MicroPython	I2C SPI UART ADC	1.8 V - 5.5 V (3.3V)
Teensy 4.1	32-bit, single-core	600 MHz	Arduino IDE, PlatformIO	I²C SPI UART ADC, DAC	3.3 V (logic level)

3.2.4.6 Microcontroller Selection

After evaluating the three microcontrollers, the ESP32 was selected as the primary microcontroller for the Pathfinder Chessboard. It has superior processing power, dual-core capability, and integrated wireless interfaces provide the performance and flexibility necessary for the system's functional requirements. The ESP32-S3's GPIO pins simply connect multiple hall-effect sensors embedded in the board squares and should be compatible with I/O extenders for additional pins when needed.

In addition, the ESP32's support for multithreading allows separate cores or tasks to manage different system components, for example, one core handling sensor scanning and rule enforcement while the other controls the LCD display and LED matrix. The microcontroller's compatibility with both the Arduino IDE and native ESP-IDF framework ensures simplified prototyping and scalable software development.

While the MSP430 offers exceptional power efficiency and the Arduino Uno provides simplicity, neither platform delivers the required real-time responsiveness or computational throughput. Therefore, the ESP32 represents the optional choice for the Pathfinder Chessboard, ensuring reliable operation and sufficient capacity for future enhancements such as wireless functionality or game data logging.

3.2.5 GPIO Expanders

Another critical detail we must take into account when choosing components is the number of pins available for use. Due to the amount of pins being used by the hall effect sensors the ESP32 by itself bottlenecks the entire subsystem and does not allow all 64 sensors to be used at once. To overcome this issue more GPIO (General Purpose Input/Output) pins must be added to the overall system via GPIO expanders. GPIO expanders are integrated circuits (IC) that allow a microcontroller to have additional I/O pins when enough built in pins are available. Different GPIO expanders will be discussed and compared in this chapter.

3.2.5.1 PCA9555

The PCA955 is a 16-bit GPIO expander chip commonly used in embedded systems to increase the number of available I/O pins on a microcontroller. This specific IC communicates with microcontrollers through the I2C protocol. Each of the 16 pins can be configured individually as either an input or output. The PCA9555 operates within a 2.3-5.5 voltage range and has an average price of $0.75.

3.2.5.2 MAX7300

The MAX7300 is an I2C compatible port expander that can provide up to 28 additional GPIO pins for embedded systems that require large scale digital I/O expansion. The MAX7300 operates in a 2.5-5.5 voltage range. The IC is commonly priced around $1.50-$2.00 and its features make it a robust option for applications demanding a high amount of I/Os.

3.2.5.3 MCP23017

The MCP23017 is another type of I2C compatible expander. This IC however supports a much wider voltage range from 1.8-5.5 V. The MCP23017 has three hardware address pins allowing up to 8 different devices to operate on the same I2C bus and provide 128 pins in total in a single system.

3.2.5.4 PCF8575

The PCF8575 operates in a 2.5-5.5V voltage range. The IC is commonly priced around $0.50-$0.80 and its features, including built-in pull-up resistors and interrupt capability, make it a cost-effective option for embedded systems. The PCF8575 also has 16 I/O pins for moderate digital I/O expansion.

3.2.5.5 MCP23S17

The MCP23S17 unlike other ICs mentioned uses SPI as its communication protocol. The MCP23S17 has 16 pins available for expanding the microcontroller. SPI communication offers

faster data transfer compared to I2C but lacks a standardized addressing scheme meaning that each SPI device requires its own chip select line which could make bus management more complex.

Table 3-5: Comparison of GPIO Expanders

GPIO Expander	Max Supply Voltage (V)	Min Supply Voltage (V)	Communication Protocol	Price	# of I/O Pins
PCA9555	5.5	2.3	I2C	$0.75	16
MAX7300	5.5	2.5	I2C	$4.00	28
MCP23017	5.5	1.8	I2C	$1.00	16
PCF8575	5.5	2.5	I2C	$0.67	16
MCP23S17	5.5	2.7	SPI	$1.78	16

3.2.5.6 Selection of Different I/O expanders

We chose the MCP23017 due to the nature of being able to use multiple devices due to its I2C compatibility. We are able to use 4 different MCP23017s in order to achieve 64 more GPIO pins for the microcontroller. We are also able to give each MCP23017 a unique address and control all hall effect sensors while only using the two I2C lines (SDA and SCL). Compared to the PCF8575 which also has 16 I/O pins and I2C communication the MCP23017 provides a wider voltage range of 1.8-5.5 V compared to the PCF8575 2.5-5.5 V. Another option was the MCP23S17 but due to the use of SPI we found it easier to address each individual I/O expander via I2C instead.

3.2.6 Miscellaneous Components

In addition to these specialized components we also needed to select more general ones. Selecting these components was less of a comparison between brands and models, but more thinking of how each would interact both with the system and the user.

3.2.6.1 User Input

For the user to interact with our system they need a way to input or alter signals easily. This is done through the use of external components, which, when the user alters them in some shape or form, input a signal to the board which in turn is processed by the system to change the desired setting.

3.2.6.1.1 Button

A button is the standard way of accepting input from the user. It allows the user to easily send a digital signal of HIGH at the press of a button, if it is configured with a pull down resistor like it is in our project. When deciding on where to use a button we looked for areas where a simple “yes” or “no” was needed and implemented the button there. The way the button works is by having two internal rails, each connected to separate connections, in this case one side to VDD, and the other to an input port on the board. When the button is pressed the two rails are connected, sending an electrical signal to the board. To ensure that there is no input when the button is not pressed, a pull-down resistor is added. This is a resistor connected to both the input port and ground.

3.2.6.1.2 Potentiometer

A potentiometer is a specialized component which allows for various voltage levels to be passed through depending on the level set. When looking for where to apply the potentiometer, we tried to find times where we needed variable input greater than 2 options. A potentiometer has three pins. Two pins are used as input leading to the last pin, which sends the signal to the dedicated source. In our project, one of the input pins is connected to VDD and the other to ground, with the final pin connected to an input pin on the board. The way a potentiometer works is by acting as a rheostat, when a user twists the pin on top it turns a wiper internally, which in turn changes the resistance, altering the amount of voltage let through, allowing for different signals to be sent.

3.2.6.1.3 Switch

A switch is a component which is used to change input for the user. It allows the user to alter a state from holding one condition to holding another. We use a simple two way switch, so when searching for where to implement a switch we searched for a place where the user would need to not only select between multiple options, but would need to hold those options.The way in which a switch functions is through the use of three pins. Two of those pins act as separate inputs from different sources, and depending on where the switch is, the third switch is connected to one of the input pins.

Table 3-6: User Input Comparison

Input Device	**Data Type Sent**	**Constant Stream?**	**Used In**
Button	Digital	No	Select between

			choices
Potentiometer	Analog	Yes	Select variable choices
Switch	Digital	Yes	Set conditions

3.2.6.1.4 User Input Selection

When deciding where to use each part we compared not just the parts themselves, but ways we could implement each using the other components. For example, instead of using a potentiometer we could instead use two buttons, and as we hold down one button we increase our variable and vice versa. This creates potential problems and unneeded complexity, however, as implementing two buttons requires two ports and more integration with the software. The problem also arises if the user holds down both buttons, so we simplify this with a potentiometer. This methodology can also work the other way, as we first intended on the system using a button to initial boot up, we later found that using a switch was more convenient on the user, to avoid accidental presses, and on the system, setting a switch to connect to the power was more efficient than a button.

Some examples of where we used these inputs include accepting confirmation on selected mode of play for the button, selecting the amount of time a game should run for the potentiometer, and powering the system on for the switch.

3.3 Power Supply Unit

3.3.1 Distribution of Component Power

The power design for any pcb is a crucial aspect to consider. One of the first steps in designing a power system is to calculate the maximum total power draw from all active components being used in the design of the chess board. When calculating the power we also need to keep in mind the constraints and specifications of the design. One of the constraints being that the battery life had to last for at least 2 hours. In order to reach a design with this constraint in mind the maximum power consumption wattage needs to be calculated. Due to overestimating power supply specifications as a safety margin for our component, active components that used a negligible amount of power were not taken into consideration. Using the datasheets we were able to find the maximum current drawn. The voltage supplied to each active component can then be multiplied by the maximum current drawn in order to get each component's maximum power consumption. Summing the maximum power consumed by each component can then give the necessary power constraints needed by the system.

Based on our calculations, the power output of the system is estimated to be around 7.45W. This would entail that in order for our design to meet the constraint of lasting at least 2 hours we have to choose a power supply with an energy specification of at least 14.9 Watt-hours (Wh). Below is a table outlining the different active components, their voltage and maximum current draws, their power usage, and the entire system's power and energy consumption.

Table 3-7: Power/Energy Usage Table

Component	**Voltage (V)**	**Current Draw**	**Power Usage (W)**
LEDs	5.0 V	1.28 A	6.40 W
ESP32	3.3 V	240 mA	0.792 W
LCD	5V	16.5mA	0.2442 W
I/O expanders	3.3V	4mA	0.0132 W
Other components	Negligible	Negligible	~0 W
System total Power (W)			7.45
System total Energy (Wh)			14.90

3.3.2 Batteries

Some of the primary specifications needed when choosing a battery are the nominal voltage(V), capacity(Ah), energy capacity (Wh). The nominal voltage of the battery must match near 3.3-5(V) devices in order to be regulated to 3.3 and 5 volts in order to keep efficiency during voltage regulation and therefore improving stability. Aiming for an overestimate of around 35% the batteries should have a capacity of around 4.02-6.10 (Ah) in order to power all the components in the system efficiently. Due to the system only using what it needs we can safely overestimate the capacity of the battery chosen. The energy of a battery can then determine how long the battery can deliver power.

Another constraint to consider for the Pathfinder Chessboard is portability. The unit that indicates how much mass or volume a battery will have is its energy density (Wh/kg). To take into account the portability the weight of each type of battery was estimated. To complete the portability constraint the power system had to be supplied by a lightweight non bulky battery power delivery system.

Safety is also an important specification to consider when researching batteries. Toxicity and combustion are two key aspects that must be considered when evaluating the safety of each battery type. Due to the chemistry of different batteries, some batteries are known to be more likely to explode compared to others. The same consideration applies to toxicity levels.

Although not a constraint, recharability is a specification that we can consider when picking a battery. Rechartibility offers for batteries to be reused without having to dispose of them when they deplete their voltage. Rechargeability does however add much more complexity to a system by adding a battery management system (BMS), which monitors and protects the battery during operation and charging.

3.3.2.1 Lithium Ion

Lithium Ion (Li-Ion) batteries are used as a simple rechargeable battery that can be made in almost any shape or size. These batteries are commonly found in very light and compact forms. Li-Ion has higher energy capacities compared to other battery chemistries. Each Li-Ion cell has a nominal voltage of around 3.7 volts and an energy density of 150-220 Wh/kg. Li-Ion batteries are often used for their small size and due to their small size and ability to recharge. However, due to the chemistry of Li-Ion batteries they do pose a serious risk of exploding if not managed properly with a BMS.

3.3.2.2 Lead Acid

Lead acid batteries are some of the oldest types of rechargeable batteries. Each lead acid battery cell has a nominal voltage of 2 volts and their energy density ranges from 30-50 Wh/kg. These batteries are known for their reliability, low-cost, and ability to deliver high power, which is why they are commonly used in automotive applications. However, they are heavy and bulky making them unsuitable for portable systems.

3.3.2.3 Nickel-Metal Hydride (NiMH)

Nickel-Metal hydride (NiMH) batteries are another type of rechargeable battery. Each NiMH cell has a nominal voltage of about 1.2 volts each and an energy density ranging from 60-120 Wh/kg. NiMH batteries are known for being relatively safe and environmentally friendly, but they do take up more space compared to lithium based batteries.

3.3.2.4 Alkaline

Alkaline batteries are non rechargeable with each cell having a nominal voltage of 1.5v except for their 9 V variation. They have energy densities of roughly 80-120 Wh/kg and are commonly found in household AA, AAA, C,D and 9 V batteries. They are inexpensive and widely available.

3.3.2.5 Nickel Cadmium (NiCd)

Nickel Cadmium (NiCd) batteries are a type of rechargeable battery known for their long cycle life. Each NiCd cell has a nominal voltage of about 1.2v and an energy density of 45-80 Wh/kg. NiCd batteries are capable of outputting high discharge currents making them more suitable for applications that require high bursts of power.

Table 3-8: Battery Comparison

Battery type	Nominal Voltage	Energy Density (Wh/kg	Lifespan (Cycles)	Size	Combustion likelihood	Toxicity	Recharability

	(V)	)					
Li-ion	3.7	250-670	300-1,000	Smallest	Moderate	Moderate	Recharable
Lead Acid	2	60-90	300-1,200	Largest	Low	High	Recharable
NiMH	1.2	60-120	500-2,000	Medium	Very Low	Low	Recharable
Alkaline	1.5/9	100-150	Single use	Medium	Very Low	Low	Not recharable
NiCd	1.2	45-80	1,000-2,000	Medium	Very Low	High	Recharable

3.3.2.6 Battery Selection

This system requires a battery solution capable of reliably supplying moderate to high current while remaining simple and safe to implement. Four NiMH batteries best meet these requirements compared to alkaline alternatives. Although NiMH cells may require a dedicated charger, they do not need a complex battery management system and are generally safe and stable. In addition, NiMH batteries perform better under heavier loads, maintaining their voltage more effectively than alkaline batteries, which tend to experience significant voltage drop at higher currents. While they are slightly heavier and comparable in size to alkaline cells, the side compartment of the Pathfinder Chessboard can accommodate them without issue.

For this design, four NiMH cells are used in series, each with a nominal voltage of 1.2 V, resulting in a total nominal voltage of approximately 4.8 V (ranging from about 4.0–5.6 V depending on charge level). This voltage range aligns well with the system requirements when paired with buck-boost regulators. Their ability to sustain higher current draw and maintain more stable voltage under load makes them a more reliable and longer-lasting option for this application.

3.3.3 Voltage Regulators

The PathFinder chess board has two main voltage rails due to different components specifications requiring different input voltages and currents. These are the 5 V and 3.3 V rails. The battery pack consists of 4 NiMH cells, which provides a voltage range of approximately 4.0-5.6 V depending on the state of charge, with a nominal voltage of about 4.8 V. Because this voltage does not directly match the required rails, voltage regulators must be used to safely and reliably power the system. This section will cover the different types of voltage regulators and

how they can be used.

The two main types of voltage regulators are linear and switching. Linear regulators operate by acting as a variable resistor in order to dissipate excess energy as heat and maintain a constant voltage. Switching regulators in contrast work by rapidly switching the input voltage on and off using inductors and capacitors in order to regulate the output voltage. A key difference between the two types of regulators is that linear regulators can only step down voltage, whereas switching regulators can step down or step up voltage. Linear regulators are more simple to implement, while switching regulators require more design complexity but provide significantly higher efficiency.

Switching regulators can be further categorized into buck (step-down), boost (step-up), and buck-boost (step-up/step-down) topologies, while linear regulators are commonly implemented as low dropout (LDO) regulators. Given the NiMH battery range, both the step-down (for 3.3 V) and step-up (for maintaining a stable 5 V rail) regulation must be considered.

3.3.3.1 LP3856 (Linear Voltage Regulator - LDO)

The LP3856 is a linear low-dropout (LDO) voltage regulator that regulates the input voltage to 3.3 V. The LP3856 has an input voltage range of 2.5-7 V, which meets the battery range of 4.0-5.6 V. The output voltage can be fixed to be 1.8, 2.5, 3.3, or 5 volts depending on the variant. The LP3856's maximum output current of up to 3 A which is sufficient for the 3.3 V rail.

However, because this is a linear regulator its efficiency depends on the ratio of output to input voltage. With a nominal input of 4.8 V and a 3.3V output the efficiency is approximately 69% and can be lower at higher input voltages. This results in power dissipation as heat, making it less ideal for higher current applications.

3.3.3.2 TPS61022 (Switching Voltage Regulator - Boost)

The TPS61022 is a boost converter, meaning it steps up the input voltage to a higher output voltage. It operates with an input voltage range of approximately 0.5-5.5 V and can generate output voltages up to 5.5V. This makes it suitable for maintaining a regulated 5 V rail when the battery voltage drops down 5 V. The regulator supports high output currents (around 2-3 A) which meets the 1.62 A requirement for the LEDs. The efficiency of this regulator is around 90% making it a strong candidate for the 5 V rail.

3.3.3.3 LM2576 (Switching Voltage Regulator - Buck)

The LM2576HVSX-3.3/NOPB is a buck (step-down) switching regulator meaning it reduces the input voltage to a lower regulated output voltage. It operates with an input voltage range of 4–63 V, which is well above the battery range of approximately 4.0–5.6 V from the 4 NiMH cells. This regulator provides a fixed output voltage of 3.3 V, making it suitable for directly powering the 3.3 V rail without additional configuration. The maximum output current of the LM2576 is 3 A, which is sufficient for stepping down to the 3.3 V rail.

3.3.3.4 TPS63000 (Switching Voltage Regulator - Buck-Boost)

The TPS63000 is a buck-boost converter meaning it can both increase or decrease the input voltage depending on the configuration of the regulator. It operates over an input voltage range of 1.8-5.5 V and can provide an output voltage range of 1.2-5.5 V. The maximum output current is approximately 1.2 , which makes it suitable for the 3.3 V rail but for higher current 5 V loads. In this design, it is primarily used as a buck converter to step down the battery voltage to 3.3 V. The efficiency of this regulator can reach up to approximately 96% making it highly efficient for this application.

3.3.3.5 LTC3113 (Switching Voltage Regulator - Buck-Boost)

The LTC3113 is a buck-boost converter capable of both stepping up and stepping down voltage. It operates over a wide input voltage range of 2.7 - 15 V. This regulator provides an output voltage range of 1.8–15 V. The regulator supports output currents up to 3 A, allowing it to support both the 3.3 V and 5 V rails. With this capability, this switching regulator can be used to efficiently regulate a wide range of input voltages to stable output levels. The efficiency of this regulator can reach up to 95%.

Table 3-9: Voltage Regulator Comparison

Regulator	**Type**	**Input Voltage (V)**	**Output Voltage(v)**	**Output Current(A)**	**Efficiency**
LP3856	Linear (LDO)	2.5-7	1.8,3.5,3.3,5	3	~73.3 %
TPS61022R	Switching (Boost)	0.5-5.6	2.2-5.5	~2-3	~90%
LM2576HVS	Switching (Buck)	4-60	3.3	3	~75-80%
TPS63000	Switching (Buck-Boost)	1.8-5.5	1.2-5.5	~1.8	~96%
LTC3113	Switching (Buck-Boost)	2.7-15	1.8-15	~3A	~95%

3.3.3.6 Voltage Regulator Selection

We decided to use the TPS63000 as the 3.3 V regulator and the LTC3113 as the 5 V

regulator. Both devices are switching buck-boost converters allowing them to maintain the full battery range of approximately 4.0-5.6 V. The TPS63000 provides high efficiency and sufficient current for the 3.3 V rail, while the LTC 3113 supports higher current demands for the 5 V rail. Using buck-boost topology for both regulators also provides flexibility, enabling the system to operate reliably with either NiMH or alkaline battery packs despite their different voltage characteristics. Compared to linear regulators, this approach improves efficiency, reduces heat dissipation, and extends battery life.

3.3.4 Logic Level Shifter

Alongside the main 5 volt supply the WS2812b LEDs require a logic signal between 4.5 and 5.5 volts. The logic output of the ESP32 is valued at 3.3 volts which does not meet our constraint of 4.5-5.5 volts. Due to this we need to appropriately accommodate the 3.3 voltage logic signal from the ESP32 to ensure functional use from the WS2812b LEDs. We can do so by using a logic level shifter to push the 3.3 voltage to the required 4.5-5.5 voltage.

3.3.4.1 SN74AHCT125

The SN74AHCT125 is a CMOS inverter that can be used to shift a 3.3V logic signal to 5V by using two stages to correct for the inversion. The device is powered by 5V which fits in well with the power specifications for this design. While this device provides a 5V output level it is however unidirectional.

3.3.4.2 BSS138

The BSS138 is an N-channel MOSFET that can be used in a bidirectional level shifter circuit to translate signals between 3.3V and 5V logic levels. The device requires external pull up resistors and other circuitry to create a functional level shifter. While this device offers bidirectional communication and is the cheapest level shifter compared, it is limited to only a single channel.

3.3.4.3 TXB0104

The TXB0104 is a bidirectional voltage level shifter that can automatically sense the direction of a signal's flow and can buff a 3.3 voltage signal to 5 volts. This component has 4 channels and operates at high speeds. While this device offers faster speeds it comes at a higher price per unit and consumes more power due to higher complexity compared to other logic level shifters.

Table 3-10: Logic Level Shifter Comparison

Regulator	Direction Type	Speed	Max Supply Voltage	Number of Channels	Price

SN74AHCT125	Unidirectional	Medium	5.5 V	4	~$0.5
BSS138	Bidirectional	Medium-Fast	50 V	1	~$0.2
Tx0104	Bidirectional	Fast	5.5 V	4	~$0.80

3.3.4.4 Logic Level Shifter Selection

We choose the SN74AHCT due to its unidirectional design being well suited for our output only signal requirements. The SN74AHCT also has the ability to reliably shift the ESP32's 3.3V logic to 5V. While the TXB0104 offers bidirectional capability and higher speed, the SN74AHCT125 uses a simpler implementation with lower costs. The BSS138 might also be the cheapest but compared to the SN74AHCT the BSS138 requires more components and thus requires more complex circuit design. Furthermore the SN74AHCT logic level shifter was most optimal for buffing our microcontrollers output signal.

3.4 Software Technology Comparison and Selection

To allow the hardware to communicate with our selected microcontroller, software must be used. Researching this software has a very high importance in engineering projects, as it significantly alters the efficiency and accuracy of it. We search for technologies which can seamlessly integrate our resources which are then described in this section.

3.4.1 Development Language

When first developing our project we had to decide our programming language we would use in conjunction with our ESP32. For this section we are not necessarily looking for one singular language, but rather the main language we use for our implementation. If existing libraries are in other languages and can be implemented we will take that option.

3.4.1.1 C

The C programming language is a language which is powerful and general purpose. C provides direct access to hardware and memory which can lead to high efficiency. This is important to our project as we are working with limited hardware and want to use it as efficiently as possible. C is also used in the official ESP framework, meaning there is plenty of documentation available on using the C language with it. On the other hand, the use of C means a much more careful and precise development is needed, as we have to manage memory manually. While this provides the possibility for more efficient code it also means small errors can greatly affect the whole system, possibly without us knowing.

3.4.1.2 C++

C++ is an extension of the C programming language, it incorporates object-oriented programming features such as classes, inheritance, and polymorphism. C++ is also part of the official ESP framework, meaning it will also have documentation. Due to the complexities added by the C++ language onto C, it will come at the cost of some efficiencies especially due to resource management.

3.4.1.3 Python

Python, unlike the last two languages, is more of a high level language, abstracting many details of the low level hardware, allowing us to focus more on problem solving. There is wide support for development of AI using python, meaning implementing the AI chess system will be easier using python as more resources are available to us. The main problem with the Python language is the requirement of an interpreter, while the ESP32 works directly with C or C++ it will not with Python. This means there will be a longer run time and delay using it compared to C or C++.

Table 3-11: Comparison of Programming Languages

Language	**Compatibility**	**Efficiency**	**Ease of Integration**
C	High	High	Medium
C++	High	Medium	High
Python	Low	Medium	Low

When deciding on which programming language to base our project off of we looked at three different factors. The first was compatibility, meaning how easily the language can be used with the ESP32, the microcontroller we decided to use. Then efficiency, meaning how efficiently the language interacts with the hardware when it is implemented. Ease of integration was the final point, which found how easy it would be to use the language when programming the system.

Looking at our comparisons we found the compatibility of C and C++ to be high with our ESP32 as they are both officially supported. For efficiency, we took into account the fact that we have to interpret it, which decreases the efficiency of Python and increases delays. Between C and C++, C is faster as there is less overhead from supporting add ons. For ease of integration we put Python at the lowest, as adding the interpreter will cause delays and complexity to arise. C++ was labeled as higher than C due to the complexities surrounding C and memory management. Overall we chose to go with C++ as our main language due to the overall positives involved with it.

3.5.2 Chess Game Logic and AI

A core foundation aspect of the Pathfinder Chessboard is the ability to light the board according to chess rules and the ability to play chess against a computer opponent. To achieve this, we need to implement the chess game logic into our system through the use of the ESP32 microcontroller we decided on using. We need this logic to be accurate to the current chess rules, as well as being efficient to make sure there is very little delay when lifting a piece or when the computer is thinking. Our first decision regarding implementing the game logic was whether to create it ourselves or using a pre-existing package.

3.5.2.1 Custom Made Logic

Building a custom chess logic package for our use would involve researching the rules of chess and implementing them using a programming language into a custom library. This brings several advantages compared to using a pre-existing library, for example we will have full control and customization. This allows us to only put in necessary parts that we need, which would also lead to a lightweight implementation, possibly decreasing the delay rate. Given we want to implement hardware with the program, we can directly implement any system we desire as we build the library, leading to a more sophisticated result.

While this may seem advantageous, it does come with its own drawbacks. The main factor in our project is time, as we are limited to a handful of months, and implementing a custom library, including the research and debugging which will go into it, would take a large amount of time. We also need to take into consideration our team experience, and while we are all experienced we may find some aspects too daunting or too out of scope, especially when trying to implement a computer player, given the amount of fine tuning these systems need.

3.5.2.2 Pre-existing Package

Our other option is using a pre-existing system which we will research online. This has the great advantage of freeing us the time which would have been spent on making a custom engine. Another reason to use a pre-existing library is it will be proven to work, as many of these libraries have been refined for years, which means we will not have to debug major sections or have to add obscure rules or edge cases. We will also not have to make an AI system for the computer player, which will also save us time and effort which can be spent elsewhere.

A pre-existing library does come with its own disadvantages, however, one being the limited customization and flexibility. Given that the system is already built, we will not be able to easily change the logic if we need to, and getting the system to communicate with the hardware will prove to be additional work. And while this is not a disadvantage, we will have to make sure we conform to any copyright rules or protection that the library we choose has.

Table 3-12: Comparison of Custom Made Chess Engine vs Pre-existing Chess Engine

Design Method	Development Time	Customization	Reliability
Custom Made	High	High	Medium
Pre-existing	Low	Medium	High

To determine the most suitable design method for the chess engine of the Pathfinder Chessboard we tested them against 3 aspects. We looked at the time it would take to develop the engine, both the game logic and implementing our hardware, how customizable the engine would be, and how reliable the engine could be.

From our comparisons we can see the pre-existing engine would be our best option. Since it is already made we only need to worry about connecting it to our hardware, and while this will certainly take time, making an engine from scratch would take more. For reliability we also saw a pre-existing engine having an advantage as these engines have been thoroughly tested, for some cases over the course of years, meaning they will be reliable compared to ours which could have overlooked bugs. While a custom made engine provides a high degree of customization that cannot be matched by a pre-existing engine, we have decided the other benefits outweigh this downside, and will be proceeding with a pre-existing engine.

3.5.3 Chess Engine Libraries

Now that we have decided to use a pre-existing game engine we have to choose one. We are looking for an engine that comes with a computer player included, can run efficiently on the ESP32, and is open source so we can modify it as needed.

3.5.3.1 Stockfish

Stockfish is an open source chess engine written in mostly C++. Where Stockfish shines is the strong AI opponent, with it being powerful enough that the engine has won multiple chess tournaments. We can easily change the strength of the computer, making it so both beginners and experienced players will make use of the program. Stockfish is also very reliable, having first been made 16 years ago there have been multiple years of fine tuning and documentation, so any problems that arise could be easily researched. Stockfish is distributed under the GNU General Public License, so we are free to use it for our project.

The main problem with the Stockfish engine is the resource consumption. Given the intensive computer opponent, the hardware requirements for the engine exceed the capabilities of the ESP32, so continuing with this engine will lead to either slow play, which is undesirable, or crashing, which cannot happen. A solution for this would be running the Stockfish engine on an external device, but this is also undesirable as it adds unnecessary complexity and communication.

3.5.3.2 Micro-Max

Micro-Max is an open-source chess engine written in C, and it has been famous for its compact form. Altogether, the engine only holds about 2 KB of code, but is able to play chess at a high level. This means it can easily fit on the ESP32 and run efficiently so there is little delay compared to heavier engines. It is open under the MIT license, meaning we are able to freely use and modify the code as we need to.

One drawback is that because the engine is so compact and heavily optimized, the code can be dense and hard to read, meaning making any modification or additions, such as

communication with the hardware, may be difficult or time consuming. The engine is also relatively new, only having existed for about a year, meaning there will be less documentation and information on the engine, and also increases the likelihood for small bugs or errors.

3.5.3.3 Sunfish

Sunfish is a python based chess engine which, like Micro-Max, is made to be lightweight and efficient, taking up less than 200 lines of code. This compactness is very useful considering we are using the ESP32. We are also able to easily modify the code since it is open source, meaning adding the communication to the hardware will not be too intensive.

However, the fact that this is a python based engine means we would need to port it to a language for the ESP32, this would either be done using a custom port or a separate host computer. This dissolves the advantage of Sunfish being compact, as needed to constantly communicate with another system or translate through a port will put strain on our project, leading to slower communication and delays. The playing strength of this system can also be called into question, as it is weaker than previous examples.

3.5.3.4 Python-Chess

Python-Chess is a chess library for python which is already built into python, meaning we would just need to import it instead of implementing it. Given its age and ease of access there is a strong ecosystem of documentation online, leading to it being easier to modify for our purposes.

This engine faces the same problems as Sunfish, however, since it is python based we would need a way to translate it, either with a custom port or separate host, which would increase the time and complexity. The other problem is the lack of a computer opponent, meaning we would either need to build our own or use a separate AI to play. This is not ideal, if we were to build our own it would be time consuming and complex, especially given our time range, and finding a separate AI to integrate into our project would also bring up more constraints, as we would need to make sure it is compatible with Python-Chess and not resource intensive.

Table 3-13: Comparison of Pre-existing Chess Engines

Engine	Resource Draw	Ease of Integration	Language	Player Strength
Stockfish	High	High	C++	High
Micro-Max	Low	Medium	C	High
Sunfish	Low	Low	Python	High
Python-Chess	Medium	Low	Python	N/A

To determine the most suitable chess engine, we compared them with 4 major points in mind. The first being resource draw, since we are using an ESP32 microcontroller we need to be careful and strict in how powerful the engine is, one that is too intensive will cause delays in processing which we do not want. Ease of integration refers to how simple it would be to both add the engine into our project and get it working, as well as how simple it would be to alter it to react with our hardware, we are looking for high ease of integration. For the programming language, we want to make sure it is one suitable for this project, and ideally one we have experience with. Finally, the strength of the AI is considered as we want this project to be used by beginners and pros alike.

For resource draw each engine was mostly similar in consumption, with Micro-Max and Sunfish both being made to be optimized, giving them the lowest draw. Stockfish, on the other hand, was shown to be very resource intensive, with the strong AI opponent being the reason, in fact it is so intensive that running it on an ESP32 does not seem feasible, as if we could get it to work it would be too slow for our purposes. The ease of integration for Stockfish, on the other hand, scored the highest. This is due to the fact that it is so well documented, there being years worth of trial and error and modification, and the cleanliness of the code, we would be able to see how the engine works and from that modify it to our needs. The two python based engines, on the other hand, have low ease of integration as described in their initial write ups, we would need either a separate system or custom port to use python. For player strength we looked at how powerful the AI systems could be for each engine, and found them all to be very similar, each engine can provide a variety of play strength, except for Python-Chess, which does not have an AI built into it.

Overall we decided on using the Micro-Max chess engine. This chess engine is the most well-rounded, able to provide a strong AI player, be modifiable, and not be too resource intensive.

3.6 Communication Protocol

The ESP32-S3 communicates with the MCP23017 I/O expanders using an I2C protocol. Each expander is assigned by a unique hardware address, allowing the ESP32 to access and control the unique pin connections of the chessboard grid. The setup enables a scalable operation while minimizing wire routing complexity and ensuring sensor polling is reliable across the PCB.

The WS2812B LEDs operate using a direct GPIO pin. The ESP32 generates pulses to set the color of each LED, accessing it by the linearization of the LED strip. In the software, the chessboard positions are mapped to LED indices in a double array, which simplifies visual control and animations.

Chapter 4: Standards and Design Constraints

4.1 Engineering Standards and Constraints

Engineering plays a vital role in societal development, driving technological innovation that improves daily life and enables new capabilities. From delivering electricity to entire communities, to creating new vehicles, to designing advanced electronic devices, engineering impacts almost every aspect of modern living. However, with these innovations comes a responsibility to ensure that engineering products operate safely, reliably, and as intended.

The more significant and useful an engineering development is, the greater the potential for harm if it malfunctions or is misused. For instance, an electrical distribution network powers homes and supports modern conveniences, but a failure could range from temporary inconvenience, such as the loss of lighting or climate control, to serious hazards like electrical fires or city-wide blackouts that disrupt essential public services. Similarly, software or hardware defects in consumer products may not result in physical harm but can cause dissatisfaction, breach of advertised specifications, and potential legal or regulatory consequences.

It is to mitigate these risks that formalized standards exist. Standards are documented technical guidelines developed by recognized organizations composed of engineering professionals. They define best practices for design, manufacturing, testing, and implementation, helping engineers ensure compatibility, assess risks, and comply with regulatory requirements. For the Pathfinder Chessboard, adherence to such standards is essential for both electrical and software components. Standards provide guidance in areas such as printed circuit board (PCB) design, soldering, LED integration, and software development, ensuring proper operation, usability, and safety.

In addition to standards, design constraints establish the limits within which a project must operate, including economic, technical, ethical, environmental, and social considerations. By defining both standards and constraints, engineers create a structured framework that guides the development process, reduces risks, and ensures that the final product meets both functional requirements and societal expectations. For the Pathfinder Chessboard, this structured approach guarantees that the product is safe, reliable, user-friendly, and feasible within the defined scope of the project.

4.1.1 Engineering Standards Overview Listing

Ensuring the correct design and implementation of the electrical circuitry is critical to the proper functionality of the product. By consulting various libraries and the resources of accredited organizations, numerous standards related to Printed Circuit Board design, electrical

lighting systems, and other implementations were reviewed. The following is a selection of standards deemed most relevant to the project's objectives and constraints, along with their corresponding descriptions.

Table 4-1: A brief outline of the Engineering Standards used for this project.

Standard	Type	Category	Application in Project
IEEE 29119-4	Software	Testing	Ensures robust testing and validation for firmware operation.
BS ISO/IEC 9899	Software	Programming	Defines structure, safety, and consistency for C-based embedded code.
IEEE 802.11	Electrical	Communication	Defines Wi-Fi communication standards for the ESP32.
IEEE 1789-2015	Electrical	LED Safety	Ensures safe PWM control for LED lighting without visual discomfort.
ANSI/UL 8750	Electrical	LED implementation	Sets safety and design standards for the chessboard's LED indicators.
IPC J-STD-001	Electrical	Soldering	Ensures high-quality solder joints and reliable electrical connections
IPC-2221A	Electrical	PCB Design	Governs layout, spacing, and trace routing for the chessboard PCB.

4.1.1.1 IEEE 29119-4 (Testing)

We recognize the importance of ensuring that the software powering our smart chessboard operates reliably and consistently under all expected user conditions. IEEE 29119-4 supports this goal by providing a structured set of software testing techniques that we can apply throughout the development to strengthen the quality and dependability of our product.

These techniques directly apply to functions within our system, for example, detecting piece movement, validating legal chess interactions, updating LED indicators, and responding correctly to unexpected user input. By using the techniques mentioned in this standard, we can design test cases that confirm the software behaves correctly with both standard gameplay scenarios and edge cases.

An important benefit of following 29119-4 is that it improves traceability. Each test we write will connect back to a specific requirement established for the project, such as response timing for a moved piece or LED outputs reflecting check or checkmate conditions. This ensures we maintain full coverage over the most critical features of the smart chessboard.

Additionally, the standard emphasizes risk-based testing, which is especially valuable to us. Components such as the Hall-effect sensors, LED lighting feedback, and move-interpretation logic are more likely to affect overall gameplay if they fail. Therefore, we will prioritize more rigorous and frequent testing on those high-impact areas.

4.1.1.2 BS ISO/IEC 9899 (C Programming Language)

BS ISO/IEC 9899 establishes the official specifications and requirements for the C programming language. This standard ensures consistency, safety, and interoperability across C-based systems, which is crucial when software directly interfaces with hardware components. Given that the smart chessboard incorporates an embedded processor responsible for continuous data exchange with sensors, communication modules, and LED indicators, strict adherence to a software standard is essential for maintaining reliable system behavior.

This standard covers critical topics such as proper syntax and structure, approved libraries, data representation, variable scope management, memory allocation, and secure handling of pointers. These guidelines work to minimize common programming issues such as undefined behavior, memory leaks, and timing inconsistencies that could otherwise result in system malfunctions. For example, mismanaging memory while tracking live chess piece locations or processing player movement could lead to incorrect state representation, affecting gameplay accuracy.

Additionally, BS ISO/IEC 9899 supports maintainability and scalability, qualities that are particularly valuable for a project expected to evolve with added features such as Bluetooth communication or autonomous move recognition. Consistency with this standard also facilitates debugging and future code integration by ensuring the software follows widely recognized norms within the engineering community.

Overall, BS ISO/IEC 9899 will serve as the primary guideline for the programming portion of the smart chessboard. Its role in the software design parallels that of the IPC-2221A standard for the electrical subsystem—each ensures that the design remains robust, safe, and

compliant with industry best practices. Following this standard strengthens the reliability of the chessboard's user-facing functionality while supporting long-term development and performance objectives.

4.1.1.3 ANSI/UL 8750 (LED Lighting)

UL 8750 outlines safety requirements for LED lighting systems, including electrical, thermal, and fire protection guidelines. Since the smart chessboard uses LEDs for board indication and user feedback, this standard ensures the lighting components operate safely under normal and fault conditions. Compliance helps prevent hazards such as overheating, electrical shock, and short circuits within the LED driving circuitry.

For this project, UL 8750 influences design choices such as LED current limits, power regulation, proper insulation, and wire routing on the PCB. By following this standard, the LED matrix can function reliably without creating risks for users interacting with the chessboard.

4.1.1.4 IEEE 1789-2015 (LED Lighting)

EEE 1789-2015 provides safety recommendations for LED lighting with a focus on preventing health risks such as flicker-induced headaches, discomfort, or photosensitive responses. Since our smart chessboard relies on LEDs to indicate player turns, piece movement, and special game states, ensuring safe lighting behavior is essential.

This standard supplies guidance on proper PWM frequency and modulation depth to avoid harmful flicker. By applying these recommendations during LED driver design, we ensure that users experience smooth, comfortable lighting during gameplay. Implementing IEEE 1789-2015 also supports long-term reliability of the LEDs while maintaining safety in typical home and competitive environments.

4.1.1.5 IPC J-STD-001 (Soldering)

The IPC J-STD-001, developed by IPC, is one of the most widely recognized and utilized standards in the field of electronics manufacturing.Originally released in the 1980s, it has undergone several revisions to reflect modern advancements in soldering technology, materials, and environmental considerations. This standard establishes the requirements for the quality and reliability of soldered electrical and electronic assemblies, making it directly relevant to the development of the smart chessboard project.

In the context of the smart chessboard, IPC J-STD-001 serves as a critical reference for ensuring that all soldered joints connecting the board's LEDs, sensors, and microcontroller meet industry-accepted standards of workmanship and reliability. Proper soldering directly affects the board's electrical performance, durability, and long-term functionality—key aspects for an interactive product expected to withstand frequent use.

The first section of IPC J-STD-001 defines the acceptance criteria for solder joints, specifying the required size, shape, and quality of each connection. Visual diagrams and detailed examples are included in the standard to help engineers identify acceptable and unacceptable joints. For the smart chessboard, this ensures that connections between the LED matrix, sensor grid, and control circuitry maintain consistent conductivity and mechanical stability. Poorly formed joints could lead to intermittent lighting or sensor failures, reducing the board's responsiveness and reliability.

IPC J-STD-001 categorizes assemblies into three classes based on performance and reliability requirements. The smart chessboard aligns most closely with **Class 2**, as it is a specialized consumer-grade electronic device that demands consistent performance and durability without requiring the extreme reliability of aerospace or military applications.

The standard's soldering methods section describes four primary soldering techniques—hand soldering, wave soldering, reflow soldering, and manual surface-mount soldering. Of these, hand soldering and manual surface-mount soldering are most relevant to the chessboard's assembly. These methods will likely be used to attach through-hole connectors, LED indicators, and small SMD components. IPC J-STD-001 provides specific temperature ranges, tip geometries, and dwell times to prevent thermal damage and ensure high-quality joints—guidelines that directly apply to the manual assembly phase of the project's PCB.

Figure 4-1: IPC J-STD-001 Soldering Standard

The environmental considerations section of the standard emphasizes lead-free soldering and the reduction of hazardous substances in compliance with global environmental regulations such as RoHS. This aligns with the team's goal of producing an environmentally conscious product that adheres to sustainable manufacturing practices.

Lastly, the rework and repair section outlines safe and effective methods for fixing soldering defects or replacing faulty components. These guidelines are critical for the smart chessboard, as they ensure that re-soldering sensors or LEDs during testing or maintenance does not damage the PCB or surrounding traces.

By adhering to IPC J-STD-001, the smart chessboard project ensures that all soldering work achieves high levels of reliability, mechanical integrity, and environmental compliance.

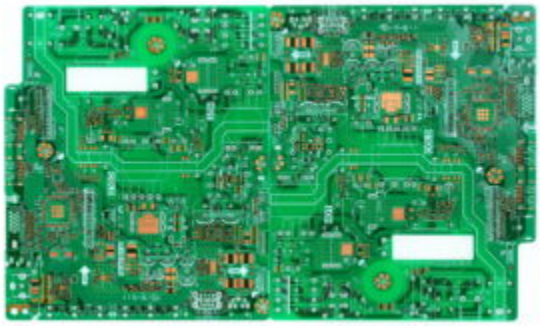
Following this standard minimizes the likelihood of cold joints, solder bridges, and thermal stress damage—factors that could otherwise compromise the device's accuracy and performance. Overall, IPC J-STD-001 supports the project's objective of delivering a durable, professional-grade electronic system that meets established engineering quality benchmarks.

4.1.1.6 IPC-2221A (PCB Design)

The IPC-2221A standard, titled *Generic Standard on Printed Board Design*, was developed by IPC, an association dedicated to creating and overseeing standards governing printed circuit board (PCB) design. As the name suggests, this standard provides general guidelines for designing printed circuit boards (PCBs) and is highly relevant to the smart chessboard project. Since the smart chessboard relies on a custom PCB to integrate sensors, LEDs, and a microcontroller, adherence to IPC-2221A ensures reliability, manufacturability, and electrical performance.

The general requirements outlined in IPC-2221A help guide design decisions for the chessboard PCB, including tradeoffs between board size, layer count, and routing complexity. For example, proper line spacing and trace width recommendations ensure that signals from touch or pressure sensors do not interfere with each other, while maintaining the compact size needed to fit beneath the chessboard squares. Environmental considerations, such as heat generated by LEDs or the microcontroller, are also addressed in the standard, guiding choices for materials and layout to prevent overheating during extended gameplay.

Figure 4-2: IPC-2221A PCB Design



Material selection guidelines are particularly applicable to the chessboard design. The standard emphasizes temperature tolerance, electrical properties, and structural strength, which are crucial for ensuring that the PCB withstands repeated stress from piece placement and potential accidental impacts. Selecting appropriate dielectric materials and adhesives, as suggested by IPC-2221A, helps maintain board integrity and signal consistency over time.

Mechanical and assembly considerations, such as board shape, rigidity, and component placement, directly impact the chessboard's manufacturability and durability. IPC-2221A recommendations for uniform board sizes, bow and twist limits, and vibration resistance are essential when designing a board that may be moved or handled frequently. The standard also guides component placement and datum reference use, which helps accurately position LEDs, sensors, and connectors to align perfectly with the chessboard squares.

Electrical properties recommendations, such as proper conductor spacing, power distribution, and crosstalk reduction ensure that the chessboard's sensors and LEDs operate reliably without interference. Transmission line techniques and electrical clearance guidelines help maintain signal integrity, especially in areas where multiple sensor traces run close together.

Thermal management principles from IPC-2221A are applied to prevent hotspots from LEDs or the microcontroller. Proper heatsinking and component spacing improve heat dissipation and contribute to long-term reliability. Similarly, quality assurance recommendations, including testing points and inspection coupons, allow validation of both electrical and mechanical functionality before final assembly.

Finally, hole and interconnection guidelines help define via types, drill sizes, and pad placements for mounting the microcontroller, connectors, and other components, ensuring structural strength and precise alignment. Documentation practices suggested by the standard facilitate future revisions or repairs of the chessboard PCB.

By following IPC-2221A, the smart chessboard's PCB can achieve a balance of reliability, manufacturability, and performance, ensuring that sensors, LEDs, and the control circuitry function correctly while withstanding mechanical stress and thermal effects. This standard serves as a foundational reference for designing a robust, high-quality PCB tailored to the chessboard's unique interactive requirements.

4.1.1.7 IEEE 802.11 (Wi-Fi)

As a stretch goal for our smart chessboard, we plan to explore wireless connectivity using the IEEE 802.11 standard. IEEE 802.11 defines specifications for wireless local area networks (WLANs), including Medium Access Control (MAC) and Physical Layer (PHY) protocols. Implementing Wi-Fi would allow the chessboard to communicate with mobile devices, online apps, or other boards for move logging and gameplay tracking.

While Wi-Fi functionality is not part of the core requirements, referencing IEEE 802.11 provides a framework for future development. Key considerations include reliable data transmission, low-latency updates, interference management, and proper antenna placement. By planning around IEEE 802.11 standards, we ensure that any future wireless implementation can meet common connectivity and interoperability expectations.

4.1.2 Design Constraints

In addition to adhering to established engineering standards, every project is influenced by inherent design constraints—factors that limit the team's options and define the boundaries of the final product's functionality and performance. These constraints may be quantitative, such as numerical limits on cost or dimensions, or qualitative, such as requirements for accessibility or user experience.

For the smart chessboard project, the most significant design constraints fall into two main categories: Economic and Social. The economic constraint defines the financial limitations of the project, including component costs, fabrication, and development resources, ensuring the project remains feasible within the available budget. The social constraint emphasizes the need for the product to be interactive, accessible, and assistive, guiding design decisions to maximize usability and positive social impact.

4.1.2.1 Social

A central objective of our project is to deliver an accessible and assistive chess experience by incorporating multiple game modes and an LED-based lighting system that guides piece movement. These features are intended to make gameplay more interactive and convenient, particularly for users who may face challenges with traditional chessboards, thereby creating a positive social impact.

The product should emphasize user-friendliness and interactivity. The motivation behind selecting this project was to design a chessboard that is simpler, more versatile, and more accessible than existing options. By offering several game modes, players can quickly select their preferred game, enhancing the board's adaptability and overall user experience.

Accessibility is a key foundation of this interactive experience, which introduces constraints on how the product must operate. One major social goal is to implement an assistive system that goes beyond conventional guidance, actively supporting players during gameplay. This ensures the board meets the intended objectives of interactivity, accessibility, and user assistance.

This assistive functionality is realized through a lighting system embedded in the board. The LEDs highlight specific squares and possible movement paths for selected pieces, helping players visualize strategies and tactics. By providing this visual guidance, the smart chessboard enhances both interactivity and accessibility, fulfilling the project's core goals.

4.1.2.2 Economic

The total estimated cost of the smart chessboard project is about **$325**, which is fully self-funded by the team members. This budget includes the costs of all components, tools, fabrication, and other necessary materials.

The main expenses are associated with custom PCB fabrication ($150), 3D printing filament ($50), and development boards ($20). Key electronic components, including the ESP32 microcontroller, I/O expanders, LEDs, and Hall effect sensors, account for a smaller portion of the total cost. The remaining budget covers additional materials such as the LCD screen, magnets, chess pieces, and frosted acrylic top, as well as minor costs for tools and miscellaneous items.

To minimize further expenditures, the team will utilize University of Central Florida laboratory facilities, including available tools and equipment, reducing the need for additional purchases. Free software resources and online design tools will also be leveraged for development, simulation, and calculations wherever possible.

By adhering to this budget, the project remains financially sustainable while covering all necessary components for the construction of the smart chessboard.

Chapter 5: Application of ChatGPT and Other Similar Platforms

The integration of artificial intelligence (AI) platforms such as ChatGPT, Copilot, and Claude, into engineering projects has increasingly become a valuable resource for research, development, and documentation. These platforms utilize advanced natural language processing to interpret technical queries, generate code, explain complex concepts, and assist with technical writing. In the context of the Smart Chessboard project, which involves embedded systems, custom PCB design, sensor integration, and software-hardware interfacing, AI platforms provide both guidance and support across multiple stages of development. The chapter explores the practical applications, advantages, and limitations of these tools within the project, while highlighting specific examples where they contributed to the design, implementation , and research documentation process. By evaluating both the benefits and potential risks of using AI in engineering workflows, this chapter provides insight into how such platforms can enhance learning and productivity in an engineering workspace.

5.1 Limitations, Pros, and Cons

5.1.1 Pros

The use of AI platforms in the Smart Chessboard project offers substantial advantages across multiple areas, including technical research, coding, design conceptualization, and documentation. One of the most notable benefits was the ability to rapidly access and synthesize complex technical information. For example, ChatGPT provided comprehensive comparisons of microcontroller options, such as the ESP32 or Arduino Nano, outlining differences in processing power, I/O availability, communication protocols, and energy efficiency. This rapid synthesis enabled informed component selection early in the design phase, significantly reducing the time otherwise spent on manual research. AI guidance also facilitated the sensor selection and integration process, summarizing the operational characteristics of hall-effect sensors, reed switches, and infrared detectors, while explaining trade-offs in accuracy, response time, and compatibility with the selected microcontroller. Beyond research and component selection, AI platforms assisted in software development by suggesting efficient algorithms for move detection, board state tracking, and communication between the microcontroller and PC-based GUI. Code templates, debugging tips, and guidance on error handling accelerated development and supported modular, maintainable software design. Finally, AI platforms enhanced technical writing and documentation by helping organize experimental results, draft clear explanations of system behavior, and format tables summarizing sensor accuracy, latency, and interface responsiveness.

5.1.2 Cons

Despite the advantages, AI platforms presented several drawbacks. Outputs were sometimes too generic or not perfectly tailored to the project’s specific hardware or software

configuration. For instance, code snippets occasionally reference incompatible libraries, generic pin assignments, or outdated methods, requiring manual adaptation. PCB layout suggestions were theoretical and needed verification through hands-on prototyping. Another drawback is the potential for overreliance on AI; although platforms offer ready-made solutions, depending too heavily on these suggestions can reduce independent problem-solving skills, critical thinking, and experiential learning, which are essential in engineering design projects. Additionally, AI cannot perform physical testing or validate experimental results, meaning the engineer must manually test sensor placement, move-detection accuracy, and hardware reliability to ensure correct operation.

5.1.3 Limitations

AI platforms also have intrinsic limitations that must be recognized. Accuracy is a primary concern, as AI-generated outputs may contain mistakes, outdated information, or incomplete recommendations. For example, suggested move-detection algorithms or sensor polling strategies might work in theory but be impractical for the specific combination of microcontroller and sensors used in the Smart Chessboard. Context-awareness is another limitation; AI cannot fully account for physical constraints such as PCB size, trace routing restrictions, power distribution, or interference between components. As a result, advice regarding component placement or signal routing often requires significant adaptation during prototyping to achieve reliable performance. Furthermore, AI cannot consider user-specific interactions, such as the speed of piece movement, weight, or potential vibrations on the board, all of which can affect sensor accuracy and responsiveness. AI also cannot conduct real-world testing or observe system behavior, so all theoretical guidance must be verified through hands-on experimentation, iterative debugging, and calibration. Finally, AI cannot anticipate complex interactions between software and hardware, such as timing delays, microcontroller processing bottlenecks, or electrical noise, which are only detectable during actual implementation. These limitations emphasize that while AI platforms are valuable tools for guidance, they cannot replace critical thinking, practical validation, or iterative design in complex engineering projects like the Smart Chessboard.

Table 5-1: AI generated table (Pros/Cons/Limitations). Prompt in Appendix E

Aspect	Pros	Cons	Limitations
Research and Design	Rapid access to component specifications, standards, and integration examples.	Occasional mismatches in real datasheet specs.	Must verify all part specifications manually before selection.

Coding Assistance	Provides debugging help and code structure examples.	May produce non-functional or incomplete snippets.	Needs validation on local compiler or IDE.
Documentation	Speeds up report writing and formatting.	May overgeneralize or use filler content.	Requires careful review to ensure project-specific relevance.
Learning Support	Helps explain unfamiliar technical terms and concepts	Risk of over-reliance can limit independent problem-solving.	Should supplement, not replace, fundamental learning.

5.2 Examples of Platform Use

AI platforms played a significant role throughout the beginning stages of the Smart Chessboard project, providing guidance, support, and efficiency improvements. One of the earliest and most impactful applications was during component selection and research. Assisting in evaluating microcontrollers, such as the ESP32 and Arduino Nano, by summarizing differences in their technical specifications. This rapid synthesis allowed the team to make informed decisions early in the design phase and reduced the need for extensive manual research.

Another practical example of AI support was in sensor selection. The team required sensors that could accurately detect chess pieces while remaining compatible with the ESP32 microcontroller. ChatGPT provided a comparative analysis of hall-effect sensors and capacitive sensors, highlighting trade-offs in detection accuracy, response time, integration complexity, and power consumption. Based on this guidance, the team selected a hall-effect sensor array optimized for the EP32's capabilities, ensuring reliable move detection with minimal software overhead. Although the AI recommendations were theoretical, they effectively narrowed the options and guided prototyping, saving time and reducing trial-and-error experimentation.

Table 5-2: AI generated table (Microcontroller Comparison). Prompt in Appendix E

Feature	ESP32	Arduino Nano	Raspberry Pi
Processor	Dual-core 32-bit Xtensa LX6 @ 240 MHz	8-bit ATmega328P @ 16 MHz	Dual-core ARM Cortex-M0+ @ 133 MHz
RAM/ Flash Memory	520 KB SRAM / up to 16 MB Flash	2 KB SRAM / 32 KB Flash	264 KB SRAM / 2 MB Flash
Connectivity	Built-in Wi-Fi and Bluetooth	Requires external modules for	No native Wi-Fi; requires Pico W

		Wi-Fi/Bluetooth	variant
Power Efficiency	Moderate; supports deep-sleep modes	Low power; suitable for battery use	Efficient but limited wireless options
GPIO Pins	30+ configurable I/O pins	14 digital I/O, 6 analog inputs	26 multi-function GPIO pins
Programming Language Support	C, C++, MicroPython	C, C++	C, C++, MicroPython
Advantages	Excellent performance; built-in wireless; strong library support	Simple, reliable, easy to prototype	Low cost; fast and power efficient
Disadvantages	Slightly higher power draw and complexity	Limited memory and connectivity	Requires external Wi-Fi module
Project Evaluation	Selected for superior performance, built-in wireless communication, and multitasking support	Insufficient memory and connectivity	Suitable backup if wireless integration fails

Table 5-3: AI generated table (Sensor Comparison). Prompt in Appendix E

Sensor Type	How it works	Advantages	Disadvantages	AI Evaluation for Project
Hall Effect Sensor	Detects magnetic fields from small magnets placed in each chess piece	Reliable detection, fast response, easy to connect to ESP32.	Requires a magnet in each chess piece.	**Best Option** – Accurate and consistent for chessboard grid detection.
Capacitive Sensor	Measures small changes in capacitance when a piece is near.	Non-contact sensing, sensitive	Easily affected by board material and humidity.	**Possible Option** – Works, but less reliable on wooden surfaces
Piezoelectric	Generates	Simple and low	Needs physical	**Not Suitable** –

Sensor	voltage when pressed or tapped.	power.	contact; cannot detect stationary pieces.	Only detects movement or pressure.
Ultrasonic Sensor	Uses sound waves to detect nearby objects.	Detects presence over a distance.	Low precision, bulky for small grid layout.	**Not Suitable** – Too coarse for detecting individual squares.

AI platforms were particularly valuable in comparing programming languages suitable for the ESP32 microcontroller—primarily C++, Python, and C. ChatGPT provided detailed comparisons of each language's performance characteristics, ease of use, and compatibility with embedded systems. C++, as used in the Arduino IDE, was recommended for its balance of high-level abstraction and efficient execution, supported by a vast ecosystem of libraries and community documentation. It offered simplicity in syntax and strong integration with sensor libraries, making it ideal for implementing the Smart Chessboard's move detection, LED control, and communication routines. Python was noted for its fast development cycle and ease of debugging, making it an excellent choice for rapid prototyping and testing algorithms before full implementation. However, it suffers from slower runtime performance and higher memory consumption, which may limit scalability for complex multitasking or timing-sensitive operations. In contrast, C (using the ESP-IDF framework) provided the most direct access to hardware and the highest execution efficiency. It enabled precise timing control and resource optimization but required more extensive setup, a steeper learning curve, and longer development times. Based on this comparison, the team adopted C++ for the main implementation, reserving Python for preliminary tests and recognizing C as a potential option for future optimization if higher performance were required. This AI-assisted comparison streamlined the decision-making process, helping the team select a language that balanced development speed, maintainability, and hardware efficiency.

Table 5-4: Programming Language Comparisons

Option	Advantages	Disadvantages	Project Application
C	Fast execution; direct hardware access; memory-efficient.	Requires manual memory management; less abstraction.	Used for low-level ESP32 control and real-time data handling.
C++	Object-oriented; integrates libraries easily; supports modular	Slightly more resource-heavy than C; steep learning curve for embedded	Used for developing structured firmware and modular logic.

	development.	systems.	
Python	Easy to read and debug; Wide AI/ML library support	Requires interpreter; slower and not ideal for real-time microcontroller control.	Used for initial testing and algorithm prototyping before firmware deployment.

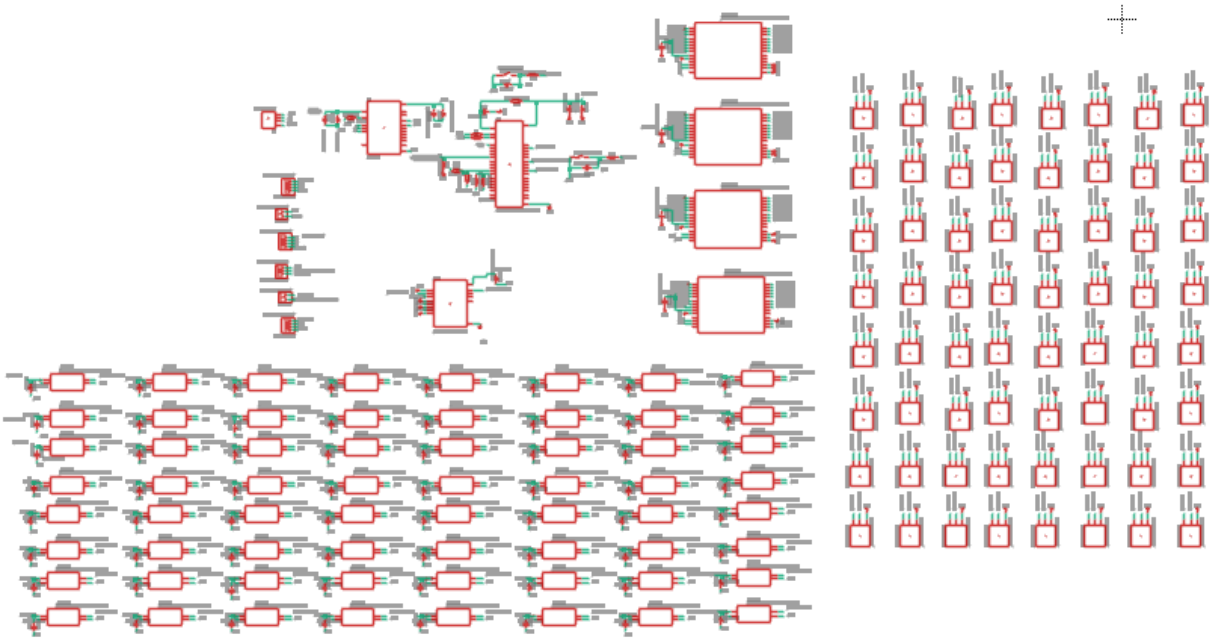
AI platforms also proved beneficial during software development, particularly in implementing move-detection algorithms, board state management, and communication between components. ChatGPT provided algorithmic guidance for sensor polling, validation of chess moves, and real-time updates to the graphical interface. It suggested code templates, debugging strategies, and modular programming structures that allowed faster integration of hardware and software while reducing errors.

Overall, these examples demonstrate that AI platforms significantly enhanced the efficiency and effectiveness of the Smart Chessboard project. They accelerated research, supported coding, informed programming language selection, improved technical writing, and provided conceptual design guidance. However, every AI-generated recommendation required careful validation and adaptation to ensure compatibility with the project's specific hardware, software, and user-interaction constraints.

Chapter 6: Hardware Design

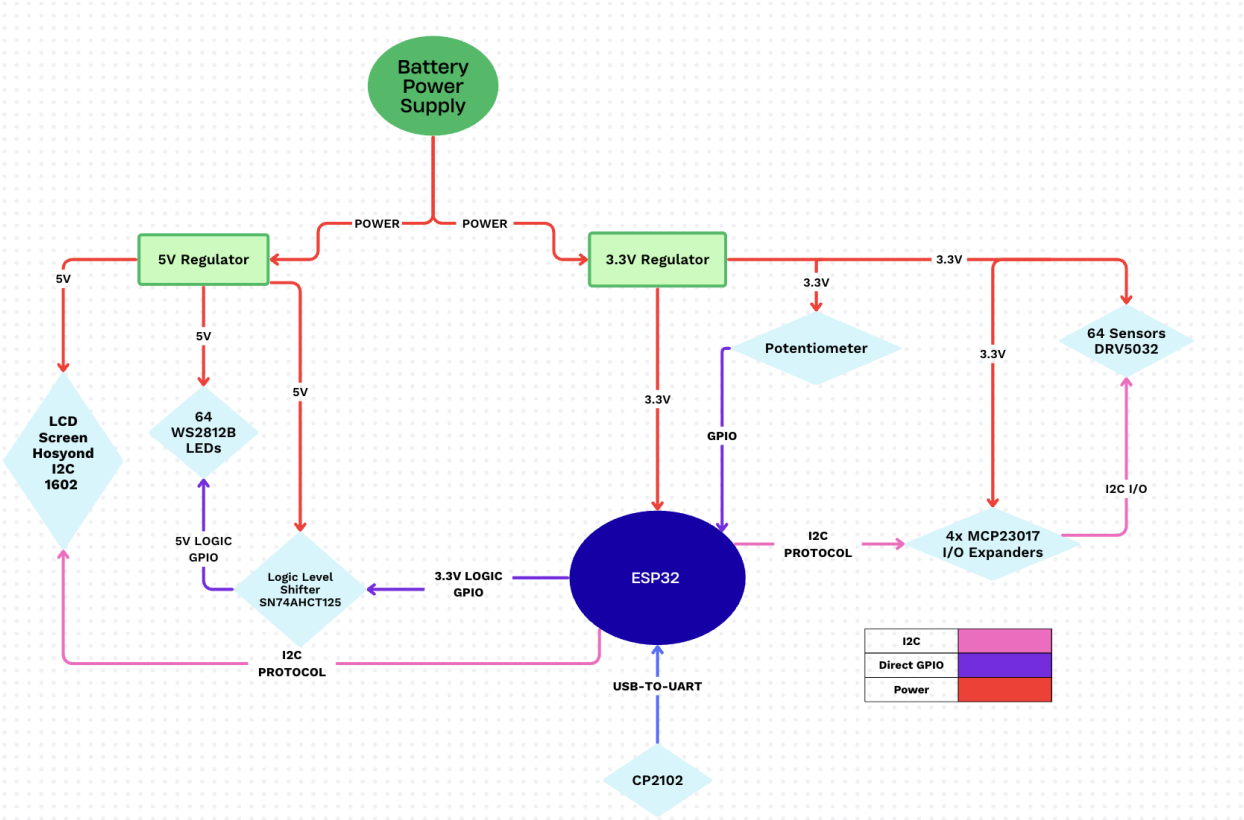
This section details the hardware design of the entire system. More specifically how each component interacts with one another. The system can be divided into 6 subsystems. One of the core subsystems being the power delivery/electrical system that explains how each individual component receives their required input voltage. Another critical design schematic to examine is the Microcontroller and how it will send and receive signals in order to interact with peripherals and other components on/off the PCB. Components directly connected to the microcontroller include the I/O extenders, logic level shifter and male header pins for the peripherals. The I/O extender and logic level shifter specifically correlate to the hall effect sensors and the LEDs respectively.

Figure 6-1: Overall Schematic Diagram



The hardware block diagram details how the subsystems within the Pathfinder Chessboard interact with one another. More specifically it simplifies the correlation details between each of the subsystems within the schematic. These details of the hardware block diagram adds clarity to the overall schematic allowing us to fully explain the bigger picture without getting lost in the intricate details of each individual subsystem.

Figure 6-2: Subsystem block diagram

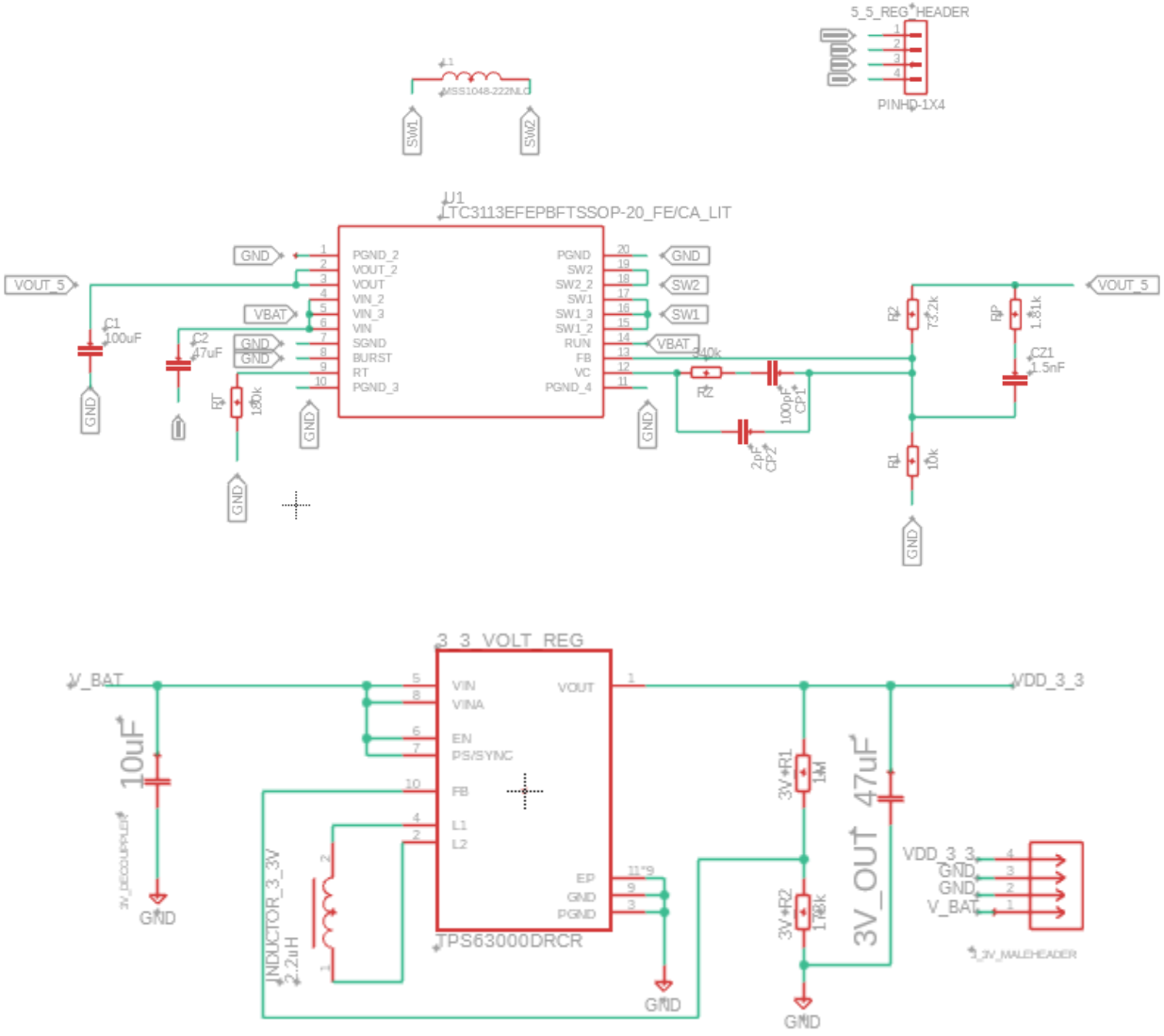


6.1 Power Delivery/ Electrical Power system

The power delivery/Electrical Power system is one of the key subsystems within the Pathfinder Chessboard. It ensures that the power delivered to each component is adequate and ensures functionality throughout the system. For the power system we have to take into account the voltage and current of each system inputs/outputs in order to create a proper schematic of the subsystem. This section will talk in detail how each of the components are powered and how the power is managed.

The power delivery system is powered via 4 D NiMH batteries in the Pathfinder Chessboard side compartment. The switch can complete the circuit when on, allowing for power to flow through. Power then flows through the batteries to both the LTC3113 buck-boost and TPS63000DRCR buck-boost (configured for buck on this design) regulators. These converters are responsible for the 5 V and 3.3 V regulation needed throughout various components throughout the circuit. Specific components such as the potentiometer, ESP32, hall effect sensors, and I/O expanders all required the regulated 3.3 volts from the TPS63000DRCR converter for this design. The LTC3113 on the other hand was used to step up the battery voltage to 5 V in order to power the LEDs, logic shifter, and LCD screen.

Figure 6-3: Bottom: 3.3 V Buck converter schematic, Top: 5 V Buck-Boost converter schematic



6.2 Microcontroller & CP2102

The microcontroller controls the software that coordinates the entire system and in turn is responsible for sending and receiving signals from connected components such as the LCD, potentiometer, LEDs, and sensors. In order to achieve this specific GPIO pins are labeled i and configured for either transmitting or receiving data. The ESP32-S3-WROOM-1 offers a significant amount of freedom when assigning GPIO pins. Due to this flexibility most of the pins on the microcontroller can be assigned to different peripheral interfaces.

In this design I2C communication lines are mapped to GPIO pin 8 (SCL) and GPIO pin 9 (SDA). Because I2C operates using active-high signaling, pull-up resistors are used on these lines to ensure proper communication.

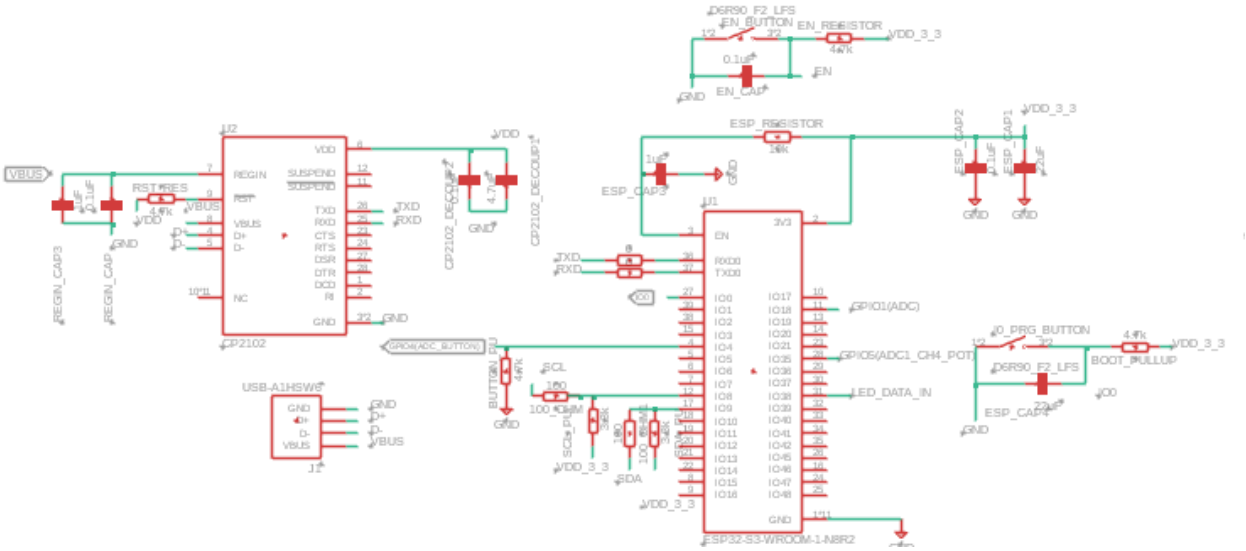
Other GPIO connections include the enable (EN) pin for controlling the microcontroller's operation, a dedicated data input pin used to transmit signals to the WS2812B LEDs, and the UART interface pins (IO0, RX, and TX), which are used for boot configuration, programming, and serial communication.

In order for the microcontroller to function properly, it must be booted correctly. The components on the left side of the schematic include the CP2102 and USB-A connector, which together provide the interface used to program the board. The CP2102 acts as a USB-to-UART bridge, enabling communication between a host computer and the microcontroller through the RXD0 and TXD0 pins.

On the right side of the schematic, a push button is connected to the IO0 strapping pin, which is active low. This button is used to place the ESP32 into bootloader mode, a necessary step when uploading firmware. Additionally, an enable (EN) button is located above the microcontroller and is used to reset or power-cycle the device as needed.

Together, these components support both programming and normal operation of the microcontroller by providing reliable boot configuration and serial communication.

Figure 6-4: Microcontroller Schematic



6.3 Peripherals

Components not included on the PCB board are mounted on the casing of the Pathfinder Chessboard for user easy access.These components include the LCD screen, potentiometer and the ON/OFF switch. In order to connect these peripherals to the main PCB block terminals and

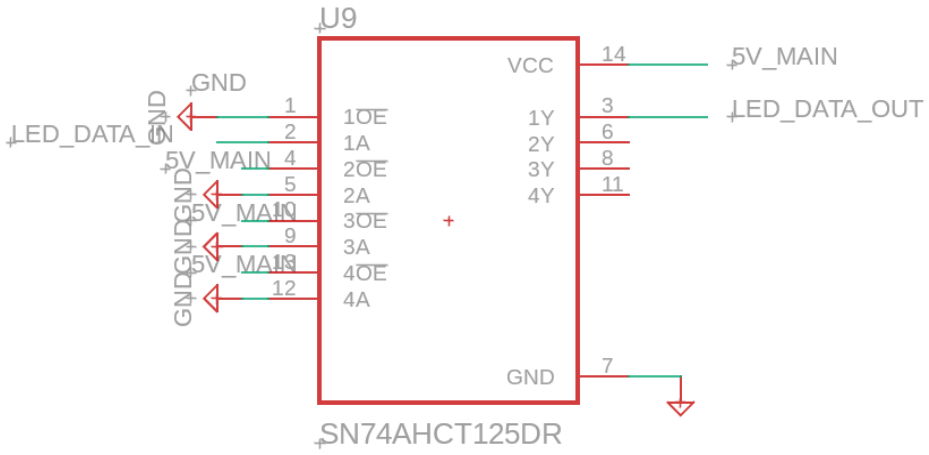
male header pins were soldered onto the main PCB which could now connect to the peripherals using wires. The wires are then attached to the block terminals or the male header pins in order for complete functionality

6.4 Logic level Shifter

The SN74AHCT125 is a crucial component for our project because the WS2812B requires the dataline to be within a narrow voltage range of Vdd-0.5 to Vdd+0.5V. Since our LEDs are powered by 5V, the data signal must be close to 5V. The ESP has a logic output of only 3.3V, which is not enough to meet the LEDs specifications.

The SN74AHCT125 acts as a logic level shifter for the ESP32's data line and shifts it from 3.3V to 5V. The SN74AHCT125 is an active component that takes a 5 V Vcc supply. It works by taking the logic signal given to its input and mirroring it at the output at the Vcc voltage level without losing any logic state. To ensure the buffer is always driving the signal its $\overline{OE}$ (output enable) must be tied to ground due to this pin operating in an active-low mode. The remaining three unused channels have their $\overline{OE}$ pins tied to Vcc(5v) to keep them disabled and preventing float inputs.

Figure 6-5: Logic level shifter schematic

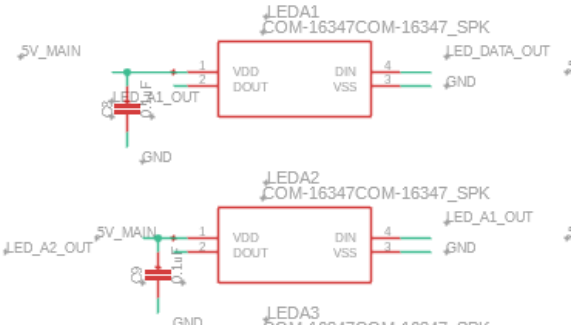


6.5 LED Design

Each of the 64 WS2812B LED's are powered in parallel by the 5v rail. Each LED has a dedicated 0.1µF for decoupling, which is essential to prevent voltage instability caused by the rapid current draw of the LED circuit. The 5V logic data signal required by the LEDs comes from the SN74AHCT125 level shifter. This single data line is connected to the Din pin of the first LED. The WS2812B are then connected in a daisy-chain architecture, where the Dout pin of one LED connects directly to the Din pin of the next pin which allows the ESP32 to individually address and control the color and brightness of all the squares on the chess board whilst using only one GPIO pin. Below is the image of the first two LEDs out of the 64 LEDs, with the first

LED receiving the LED_DATA_OUT line and sending the LED_A1_OUT line to the next LED.

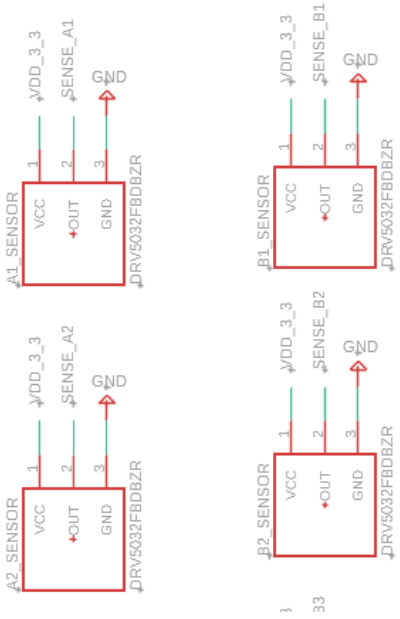
Figure 6-6: LED Schematic



6.6 Hall Effect Sensors

The hall effect sensors are another integral part in indicating where a piece is on the chessboard. The sensors require a specific magnetic threshold to be crossed in order to send a signal. Each hall effect sensor must be assigned to one of the 64 tiles on the Pathfinder Chessboard. With the MCU not having enough pins to support 64 different components I/O extenders must be used. In order to be integrated with the I/O extenders the sensors are first given a label that correlates to a GPIO pin on the I/O extenders. For example if a piece is put on the C5 square the sensor would output a data signal to GPA5 of the I/O extender. Each capacitor is also powered by the 3.3 volt rail. Below is the image of sensors labeled A1, A2, B1 and B2 out of the 64 sensors.

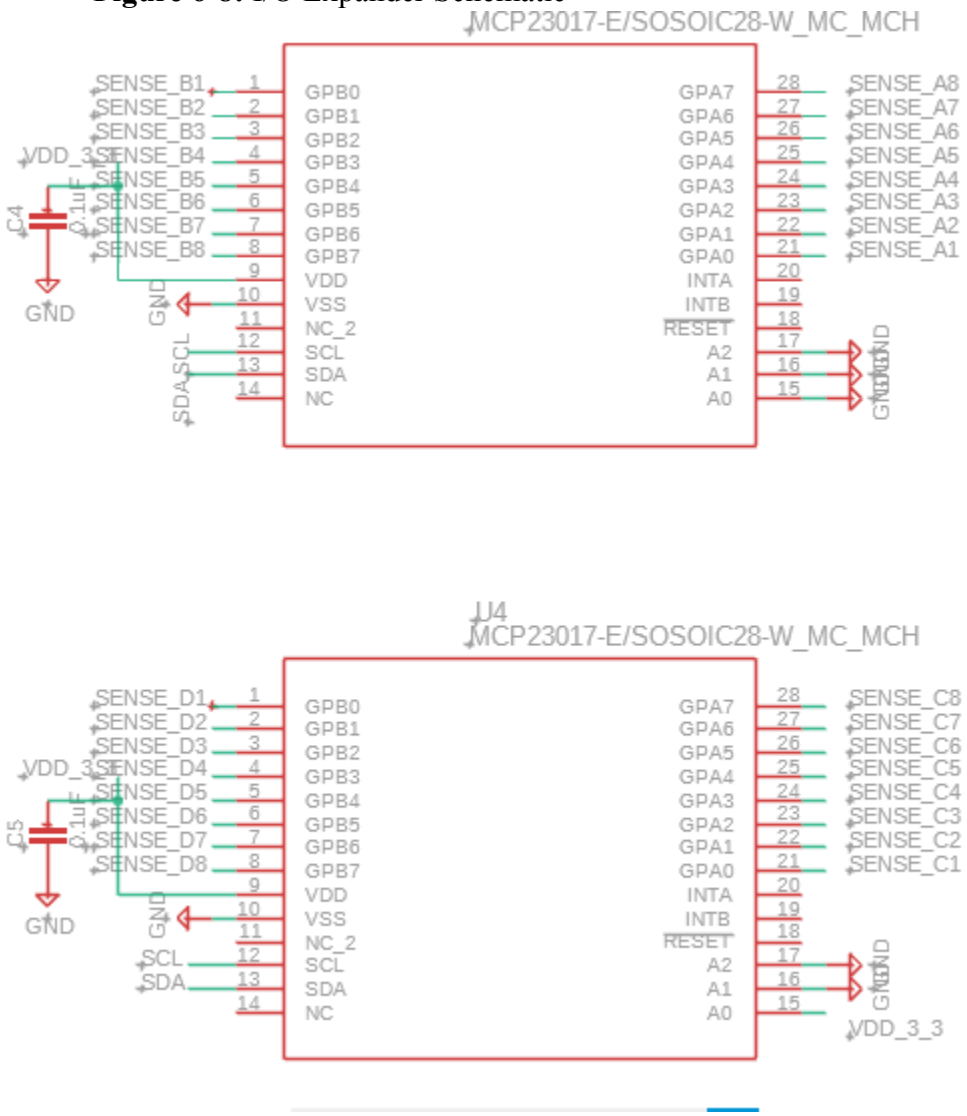
Figure 6-7: Sensor Schematic



6.7 I/O Expanders

The I/O expanders are used to expand the amount of GPIO pins the MCU can control. This was done due to our project needing to control 64 hall effect sensors and therefore requiring 64 GPIOs. In order for the MCU to tell which I/O expander is which, the I/O expanders must have a unique address. The addresses are configured with the A0-A2 pins. These pins are either set to ground or the voltage source. Below is the image of the first two I/O expanders.

Figure 6-8: I/O Expander Schematic



Chapter 7: Software Design

This chapter provides an overview of the software architecture for the Pathfinder Chessboard, emphasizing how the chess engine interfaces with the system hardware to deliver a responsive and rule-compliant gameplay experience. The software is responsible for managing all aspects of player interaction, including game setup, move validation, turn management, and visual feedback through the LED interface and LCD screen. The design ensures that the system reacts efficiently to physical interactions while maintaining the integrity of the game rules.

At the heart of the system is an ESP32 microcontroller, which serves as the CPU for integrating real-time sensing, rule enforcement, and LED-based user feedback. The chessboard will be equipped with 64 Hall-effect sensors to detect the presence and movement of pieces, along with 64 WS2812B addressable LEDs that provide dynamic visual guidance and event-based signals. The software must continuously monitor all sensor states while controlling precise LED patterns, requiring careful management of timing, interrupts, and I²C communication with MCP23017 I/O expanders to expand the number of available GPIO pins.

To manage the complex interactions between hardware and gameplay logic, the software employs a structured data model representing the chessboard state. This model supports real-time validation of moves according to chess rules, turn tracking, and interaction with the integrated chess engine, Micro-Max. By combining reliable sensor input, accurate game-state modeling, and responsive visual output, the software design achieves a seamless interface between physical gameplay and computational logic, providing both an intuitive user experience and strict rule enforcement.

The overarching goal of the software architecture is to maintain low-latency responsiveness while supporting scalable and maintainable code. This ensures that players experience immediate feedback, accurate move validation, and engaging visual cues, all of which contribute to a high-quality, interactive chessboard experience.

7.1 System Architecture

The Pathfinder Smart Chessboard software architecture is built around an ESP32 microcontroller that executes the full game logic, monitors board state using an array of Hall

effect sensors, and drives WS2812B LEDs for visual feedback. All processing occurs locally, without reliance on external connectivity, ensuring that the system can enforce chess rules. The ESP32 interfaces with two MCP23017 I/O expanders over the I²C protocol to acquire magnetic field readings from all 64 DRV5032 Hall effect sensors. These readings enable the system to determine when and where a piece is picked up or placed. Once a potential move is detected, the chess rule engine will validate legality, update the internal board representation, and trigger appropriate LED behavior to guide the players.

7.1.1 LED Configuration

The LED subsystem provides visual feedback by controlling a daisy-chained strip of 64 WS2812B addressable LEDs, one under each board square. These LEDs will be indexed linearly in a row to row mapping pattern so the software can translate a board coordinate (row,column) into a LED address. The subsystem highlights valid destinations when a piece is lifted, uses distinct patterns for illegal moves, and transitioning players' turns. Additionally, states such as check or checkmate should trigger a unique lighting sequence to alert players. All LED operations will be synchronized with gameplay to ensure intuitive and responsive interaction.

7.1.2 Sensor Configuration

The system is responsible for sampling sensor data and detecting piece movement events. Each Hall effect sensor corresponds to a specific board square and outputs a signal indicating whether a magnetic chess piece is present.The ESP32 reads these signals by polling two MCP23017 expanders via I²C, allowing access to all 64 input states. A motion detection algorithm compares the current sensor matrix with the previously recorded matrix to identify piece removal (source square) and placement (destination square). This subsystem abstracts raw sensor data into high-level actions, enabling the game logic to process movement without directly handling hardware-level details.

7.2 User Interaction Logic

The LED interface acts as the primary method through which players interact with the system, providing real-time visual feedback for every stage of gameplay. When a player lifts a piece, the LEDs illuminate all legal move options for that piece. This feature helps new or inexperienced players learn valid movement patterns while also allowing more experienced players to navigate the board quickly without needing to mentally track all possibilities. By making move options visually obvious, the system supports both learning and efficient play.

In addition to guiding valid actions, the LED system plays an important role in preventing and correcting errors. If a player attempts an illegal move or places a piece on an invalid square, the board responds with flashing lights or a specific corrective animation. These visual warnings are designed to be noticeable without interrupting the natural flow of the game. This ensures that players immediately understand the mistake while the system maintains control over game legality and state consistency.

The LEDs may also communicate turn ownership by using distinct color schemes for each side. This minimizes confusion, especially during fast-paced gameplay or when players

momentarily step away from the board. Event-based lighting patterns, such as special animations for captures, checks, or checkmate add an extra layer of clarity and engagement. These patterns provide instant feedback about important game events, making the overall experience more intuitive and visually informative.

By combining guidance, correction, and event signaling, the LED interface significantly enhances overall usability and accessibility. Players can quickly understand board conditions, recognize errors, and follow the game's progression without needing to reference external displays or rule reminders. This interaction logic helps create a seamless, user-friendly experience that supports both casual and competitive play.

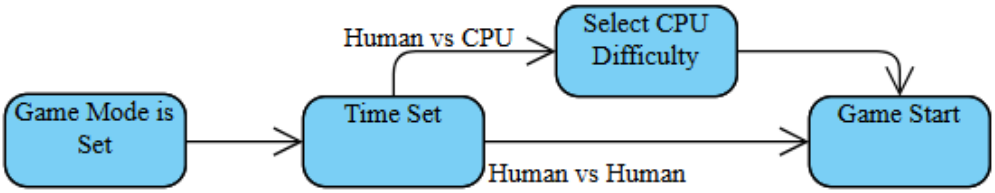
7.2.1 Chessboard Initial Setup

When the system is first powered on, the user is presented with a sequence of configuration options to customize the gameplay experience. The initial prompt requires the user to select the game mode, choosing between **human vs. human** or **human vs. computer**. This selection is made using an onboard potentiometer, which serves as an intuitive analog input device. A potentiometer reading above 50% corresponds to human vs. human mode, while a reading below 50% selects human vs. computer mode. Once the desired mode is chosen, the user confirms the selection by pressing a dedicated push-button, ensuring that the system does not proceed until the user intentionally locks in the choice.

After the mode selection is confirmed, the user must specify the time limit for the match. This setting determines the maximum duration of the game and is configurable up to one hour. The potentiometer is again used for this selection, allowing users to smoothly adjust the time control to their preference. If the potentiometer is set to its minimum position, the system interprets this as a request for **no time limit**, enabling unrestricted play without countdown pressure. This flexible timing system supports a variety of playstyles ranging from fast, competitive matches to relaxed, open-ended games.

For games played in human vs. computer mode, the system requires one additional configuration step: selecting the computer difficulty level. The same potentiometer input is reused to determine the desired challenge level, with increasing potentiometer values corresponding to more advanced CPU difficulty. This scalable design allows players to tailor the opponent's strength to their skill level, accommodating both beginners and experienced users. The selected difficulty is recorded once the user finalizes the configuration, ensuring consistent behavior throughout the match.

Figure 7-1:



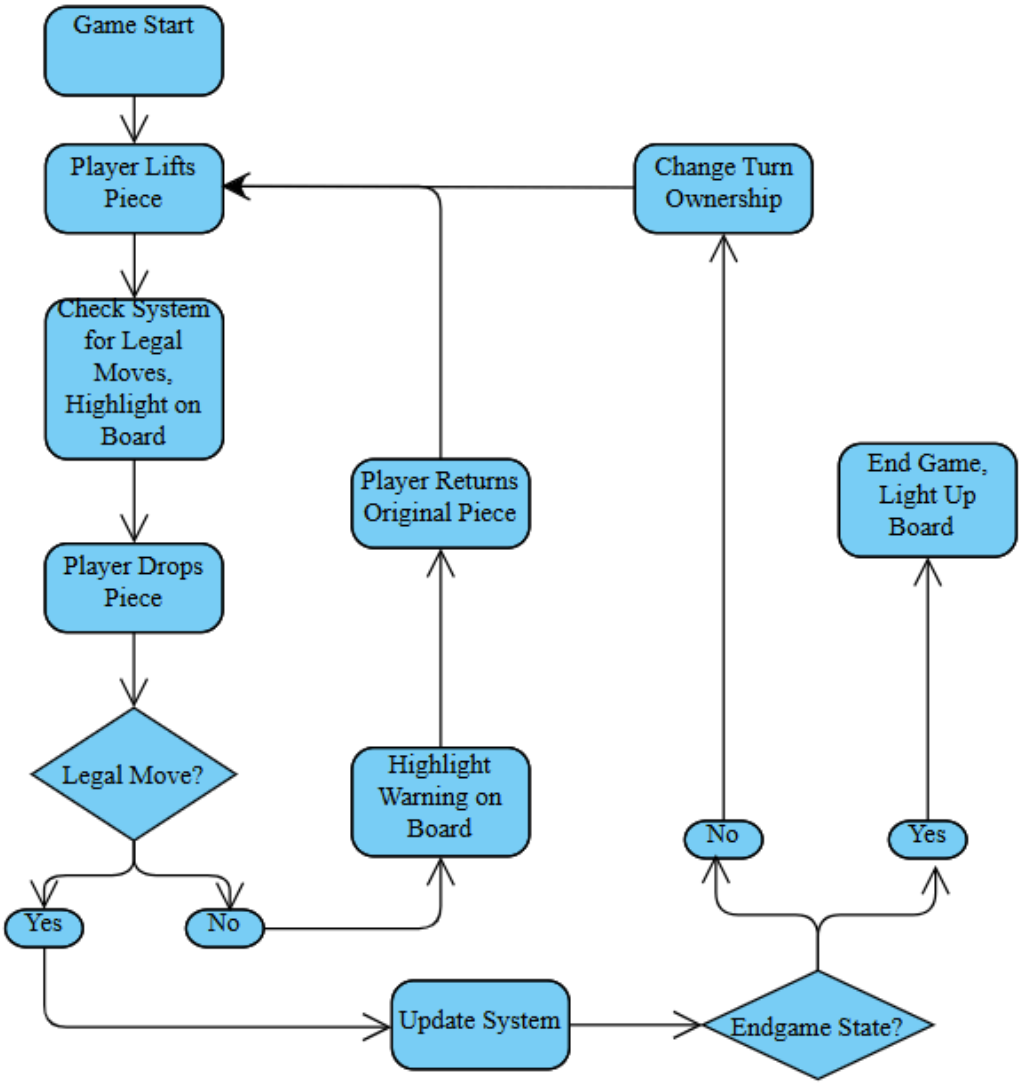
7.2.2 Game State Handling

The game logic operates through a finite-state machine designed to ensure the system responds predictably to every user interaction. At startup, the board enters a ready state, a neutral condition in which no piece is selected and the system waits for the player to initiate an action. When the user lifts a chess piece, the Hall-effect sensors detect the piece removal and identify which square is being interacted with. If the lifted piece belongs to the player whose turn it currently is, the system transitions into a selection state, during which all legal destination squares for that piece are illuminated. This visual cue provides real-time feedback and prevents accidental or uninformed moves.

Once the legal moves are displayed, the system enters the placement state, where it waits for the player to place the piece onto a new square. When the piece is set down, the firmware verifies whether the chosen destination square corresponds to a legal move generated by the chess engine. If the move is valid, the system updates the internal game state, including board position, move history, and turn ownership. It also triggers checks for special conditions such as check, checkmate, or stalemate. After completing these updates, the system returns to the ready state to await the next player action.

If the player places the piece onto an illegal square, the system recognizes the invalid move and immediately reverts to the ready state without updating any internal records. To ensure the player is made aware of the mistake, the board displays a visual warning pattern, indicating that the attempted move did not comply with the rules. This prevents the game from entering an inconsistent or ambiguous state and reinforces rule-compliant interaction.

Figure 7-2:



When playing against the computer, the state machine operates similarly but includes additional logic to accommodate computer-generated moves. After the player completes their turn, the chess engine calculates the optimal move for the computer. Instead of illuminating all legal squares, the system highlights only the destination square of the computer's chosen move, guiding the player to manually move the CPU's piece. This preserves the tactile nature of over-the-board chess while incorporating computer strategy.

When the player lifts the computer's piece, the system enters a validation state to ensure the move is executed correctly. Once the piece is placed on the highlighted square, the sensors confirm that the final position matches the move selected by the engine. If the placement is correct, the game state updates as usual and transitions back into the ready state for the player's next turn. If the piece is placed incorrectly, the board triggers the same visual warning pattern used for illegal human moves, preventing the CPU's state machine from advancing until the correct square is chosen. This process maintains full rule compliance while ensuring the user remains engaged in physically executing every move on the board.

7.3 Micro-Max Explanation

For this project, we use an external chess engine to handle all game logic instead of creating our own. Building a full chess engine from scratch would require a significant amount of time, especially because we need support for a CPU opponent. To stay within our project schedule and still provide a reliable computer player, we chose to integrate an existing engine rather than develop one ourselves.

The engine we selected is **Micro-Max**, a very small and efficient chess engine that still provides all the essential features needed for standard chess play. Even though it is compact, Micro-Max can generate legal moves, evaluate board positions, detect check and checkmate, and choose moves for the computer opponent. Its lightweight design makes it easy to integrate with our system without requiring large amounts of memory or processing power.

Micro-Max determines its moves by looking at all legal possibilities and evaluating which outcome is best for the current position. The engine can adjust its difficulty by changing how deeply it searches through possible moves, allowing us to scale the CPU strength depending on the user's chosen settings. Using Micro-Max gives our project a dependable and efficient way to run chess logic while allowing us to focus on other aspects of the system, such as sensing, lighting, and user interaction.

7.3.1 Chess Game Logic Explanation

The Micro-Max chess engine starts with a 10 by 12 board to create its chessboard, though only squares 21 through 98 are available as actual chess squares. The rest of these squares lie outside the main square, and are used to detect illegal moves. This makes it so when an illegal move is performed outside the standard 8x8 board, instead of halting the program with an array out of bounds exception we can just check if it is in bounds. Each piece is coded as an integer where they hold two aspects, the piece type and color. The piece starts with a value, 8 for white pieces, then 16 for black pieces. The piece then adds a number, 0-5, which represents a pawn to a king, so the pieces range in values from 8 to 13 and 16 to 21. This is effective as in binary this represents 1000 to 1101 and 10000 to 10101, meaning the library can use the AND operation with the piece to determine its color and rank. For example, a black bishop has integer 18 (10010) and if we AND this with 7 (111) we get 2 (010) the number which corresponds to a bishop. If we AND 18 with 24 (11000) we get 16 (1000) indicating this is a black piece.

These two systems combine to a very efficient lookup table to determine legal moves. Using the piece rank the chess engine stores what each piece can dom move diagonally, horizontally, or even special moves. Once the engine knows the applicable movement directions, it begins to generate moves until either it encounters a legal square, square outside range, a friendly piece, or an enemy piece at which point it will mark the move accordingly or remove it if not legal. Special moves are also handled, for example moves with a pawn such as en passant or a knight's irregular movement.

7.3.2 Chess AI Logic Explanation

The chess engine Micro-Max comes with a prepackaged AI logic system which allows the user to play against a computer opponent. It is built around a compact alpha-beta negamax searcher. The way the alpha-beta searcher works is by holding two values, the alpha, the best score found so far, and the beta, the lower score bound that is tolerable. This is done to avoid having to search through every possible move, if the system encounters a move that is worse than a previously discarded alternative, it stops exploring further moves in that branch. This eliminates the vast majority of move combinations and allows the engine to look several moves ahead, mimicking how a player may act.

Another component to this engine is the quiescence search, which, in short, allows the engine to evaluate turbulent positions by looking ahead, not just focusing on gaining small victories to get out of volatile situations but by “calming down” and thinking ahead, which helps it avoid blunders. The engine also contains a small transposition table, a hash-based memory of positions it previously evaluated. This saves time and effort as the engine spends less time doing calculations on moves it already knows the value of.

To calculate the value of a move, the engine first simulates the move on the board, and assigns it points based on a variety of factors. The biggest factors are capturing a piece or being attacked by an opponent's piece, which increase the points and decrease it, respectively. The engine also rewards points for improving piece placement or board control. The engine then looks at how the opponent would respond, and assumes they would counter with their best possible move, and if this move is considered to be a strong move, points are decreased, but if the opponent only has a weak move, then scores end up higher. Using the alpha-beta search and quiescence search described earlier, the engine then prunes the weak moves until it is left with its best possible move.

The strength of the AI is determined by the search depth and the node amount. The search depth amount controls how many half-moves, a single move by only one player, the AI is allowed to search though. For example, if this number were 3, the chess engine would simulate its move, the opponent's move, then its response to the opponent. The higher the values the more in depth the AI can search, leading to better planning and development, though at the cost of time efficiency and resource use. The node amount represents the amount of branching paths the engine can make for each piece, whether it be during the alpha-beta search or quiescence search. If this number is small, then the engine can only afford to search shallow lines and may miss out on longer moves or ones which pay off later, but it will increase the efficiency and decrease the time spent on each move.

Chapter 8: System Fabrication/Prototype Construction

8.1 Demo Testing

Prior to PCB fabrication, all critical components and subsystems were validated using a breadboard setup. This preliminary testing phase ensured that the circuit logic, signal flow and component interactions functioned as intended before committing to a permanent design. Conducting early validation reduced the risk of costly design errors and minimized the need for post fabrication revisions.

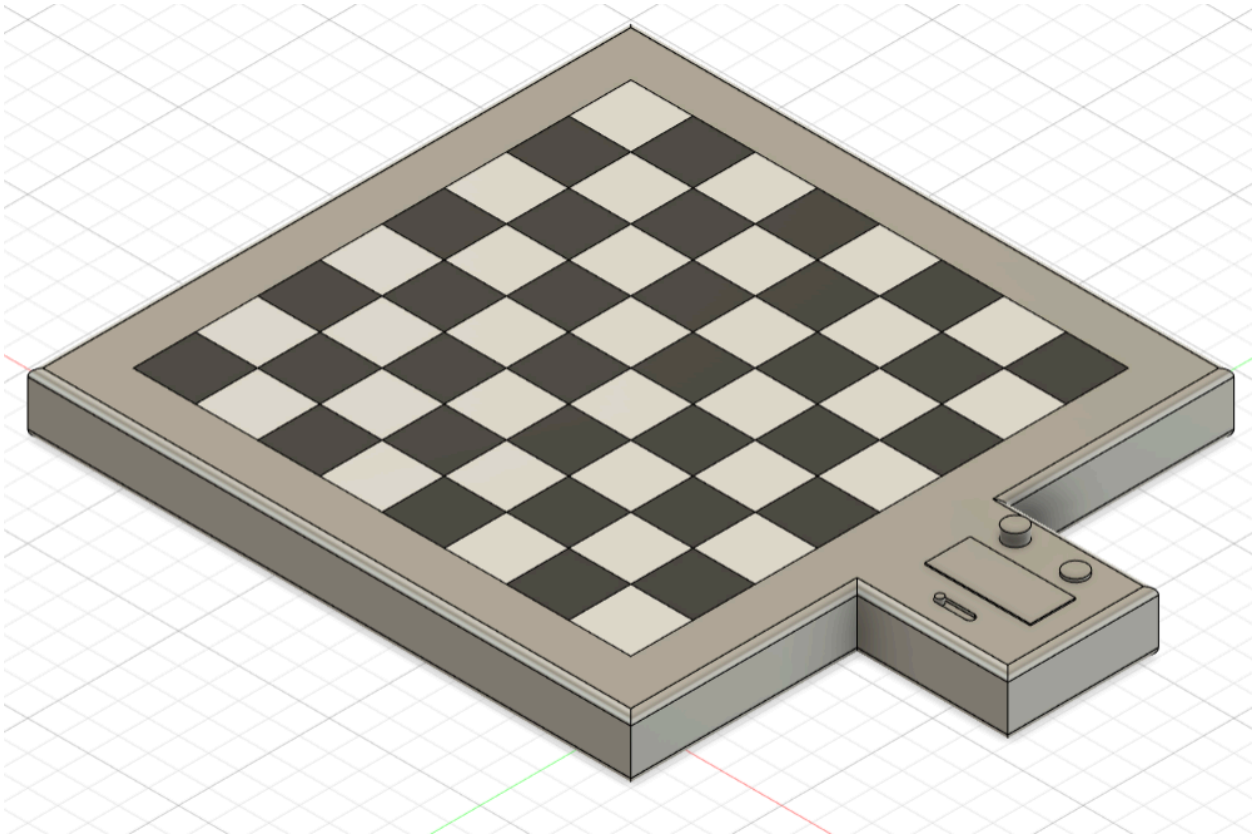
8.2 Mockup 1

The first mockup of the chessboard, shown below, was developed using Autodesk Fusion 360 and is based directly on the dimensional specifications outlined in the original project proposal. This initial design serves as a visual and structural reference for the planned system, allowing the team to verify spatial requirements, assess component placement, and ensure that the overall form factor aligns with the intended user experience. By creating a detailed 3D model early in the design process, the team is able to evaluate feasibility before committing to fabrication or material procurement.

From this mockup, several improvements have already been identified for the next iteration. One of the primary modifications under consideration is reducing the overall footprint of the board to more closely resemble the dimensions of a standard tournament-sized chessboard. This adjustment not only enhances realism from a user perspective but also contributes to better proportional balance between the playing surface and the embedded electronics.

Another planned modification involves expanding the side compartment that houses the electronic components. In the initial design, this compartment appears more compact than anticipated, leaving limited clearance for wiring, microcontrollers, power delivery hardware, and supporting circuitry. Increasing its size will provide more comfortable routing space, reduce thermal and layout constraints, and prevent the internal design from looking visually cramped. These refinements will guide the next version of the model as the team works toward a more polished and functional final enclosure.

Figure 8-1: 3d model of the Pathfinder Chessboard



8.3 PCB Layout

The final stage of PCB design involves translating the schematic into a physical layout. Components are arranged and routed to create a manufacturable board, which is then submitted for fabrication.

Several design considerations were taken into account during this process, including trace width, component placement, and power integrity. For example, decoupling capacitors used to mitigate voltage fluctuations were placed as close as possible to their corresponding components to ensure effective noise suppression.

The layout of sensors and LEDs was designed to match the standard 8×8 chessboard configuration. Each of the 64 tiles contains both an LED and a sensor, ensuring accurate position detection and visual feedback. To maintain signal integrity and minimize parasitic effects, supporting components such as decoupling capacitors were placed on the underside of the PCB directly beneath their associated LEDs.

Additional components—including the ESP microcontroller, header pins for external peripherals, and the voltage regulation circuitry—were positioned along the right side of the board, adjacent to the compartment designated for battery storage. This placement simplifies power routing and improves overall system organization.

In the PCB layout visualization, red indicates components on the top layer (e.g., sensors, microcontroller, and LEDs), while blue represents components on the bottom layer (e.g., decoupling capacitors and I/O expanders).

Special attention was given to the layout of the voltage regulators, as their performance is highly dependent on PCB design. The regulator circuits were implemented in close accordance with datasheet recommendations to ensure stable and efficient operation. Critical factors included minimizing parasitic inductance and capacitance through tight component placement, particularly for the inductor and input/output capacitors, which directly impact output stability and noise.

Component selection also played a key role, as appropriate inductor types and capacitor values were required to meet the electrical and transient response specifications of the design. Additionally, the physical size of the regulator circuitry was carefully considered to ensure it could be accommodated along the side of the PCB without interfering with other components. Proper trace width and short, low-impedance routing paths were used to handle current demands and further improve power integrity.

Figure 8-2: Main PCB layout

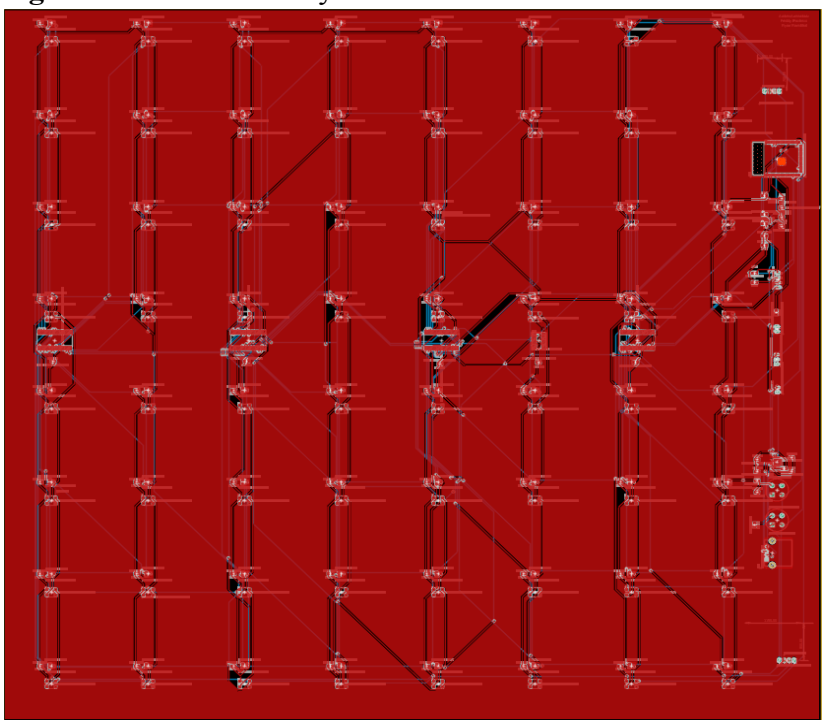


Figure 8-3:LTC 3113 Regulator PCB Layout

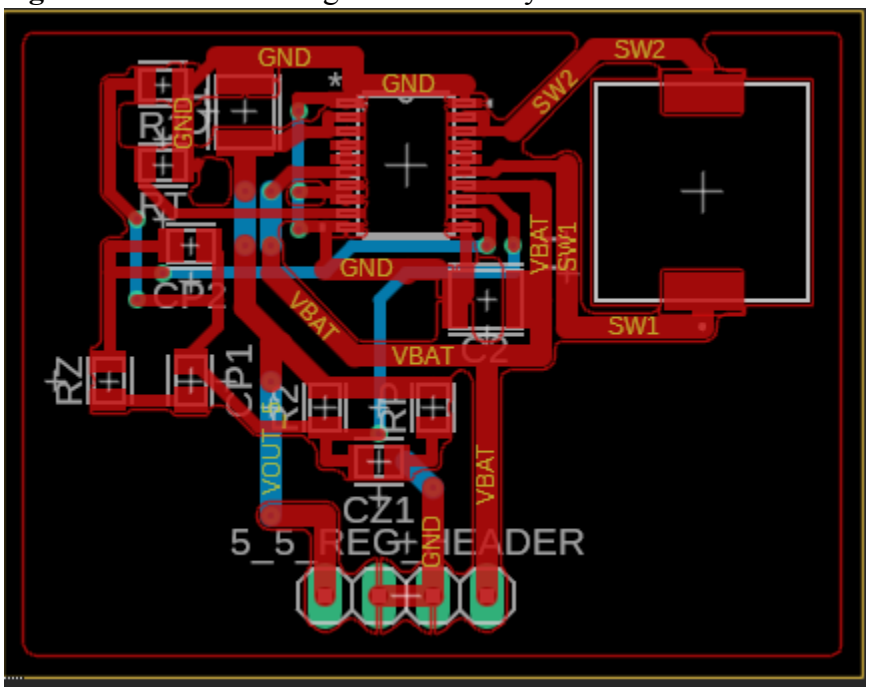
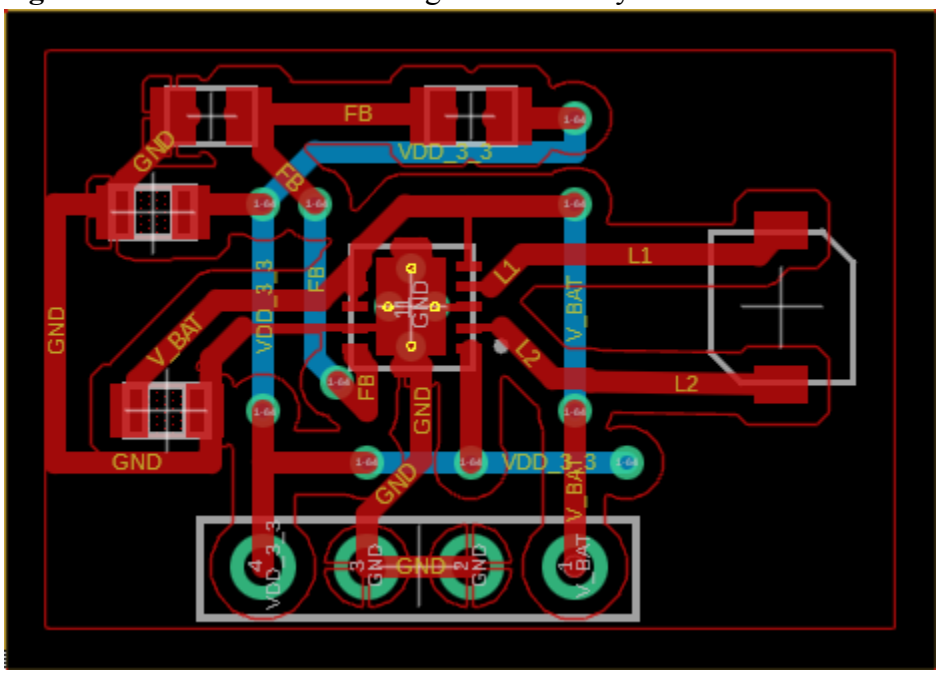


Figure 8-4: TPS63000DRCR Regulator PCB layout



8.4 PCB Fabrication

The main and regulator PCBs were fabricated using JLCPCB, selected due to prior experience and familiarity from previous coursework. This manufacturer provides flexibility in

customizing key parameters such as board thickness, copper weight, and surface finish, allowing the design to meet project specific requirements.

Additionally JLCPCB offers cost efficient services, including discounts that help reduce overall project expenses while maintaining acceptable manufacturing quality.

8.5 Housing

The housing for the Pathfinder Chessboard was constructed using a shadow box approach to provide both structural support and a clean, enclosed presentation. The main enclosure for the chessboard was built using a 16×17 shadow box frame, selected to accommodate the full board layout along with internal components such as the sensor grid, LED matrix, and wiring. This frame provided sufficient depth to house the electronics while maintaining a compact overall form factor.

To integrate supporting hardware, a secondary 6×6 shadow box frame was attached as a side compartment. This compartment was specifically used to house user-accessible and power-related components, including the potentiometer for system adjustments, a rocker switch for power control, and the battery pack for portable operation. Separating these components from the main board improved organization and reduced electrical interference with the sensing system, while also making them easily accessible to the user.

The top surface of the chessboard was enclosed using a transparent acrylic sheet, which serves both as a protective layer and a stable platform for piece movement while allowing LED illumination to remain clearly visible. The side compartment was enclosed using a plastic sheet, ensuring protection of internal components while maintaining a lightweight and cost-effective design. Overall, this housing approach balances durability, visibility, and ease of integration for the system's hardware.

Chapter 9: System Testing and Evaluation

9.1 Overview of Testing Approach

In the mini demonstration setup, the ESP32-S3 directly reads sensor states through its available GPIO pins. This configuration is intentionally simplified, allowing the team to validate the core functionality of the chessboard system without the added complexity of I²C expanders, which will be necessary for the full-scale project to accommodate all 64 sensors. The primary objective of hardware testing at this stage is to ensure that each sensor reliably produces the expected digital HIGH or LOW signal corresponding to the placement or removal of a chess piece.

Testing procedures include verifying correct wiring for each sensor to prevent misreads, ensuring that no floating inputs exist that could lead to unstable readings, and confirming that the software correctly interprets sensor activations to update the internal board state. Additionally, the piece movement logic is evaluated to ensure that the system correctly identifies valid and invalid pawn moves, updates the board grid representation accordingly, and triggers the appropriate LED feedback for visual verification. This early-phase testing provides a controlled environment to validate foundational system behavior, identify potential hardware or software issues, and ensure the architecture will scale successfully when transitioning to the full 64-square implementation using MCP23017 I/O expanders.

9.2 Hardware Testing Plan

Hardware testing focuses on verifying sensor accuracy, signal integrity, communication reliability, and the correct operation of the LED visualization system. This testing ensures that the physical components of the chessboard reliably interact with the software and produce consistent outputs under various operating conditions. To systematically validate the hardware, testing is divided into two phases: the SD1 mini demonstration and the full-scale chessboard implementation using MCP23017 I/O expanders.

Table 9-1: Hardware Testing Summary

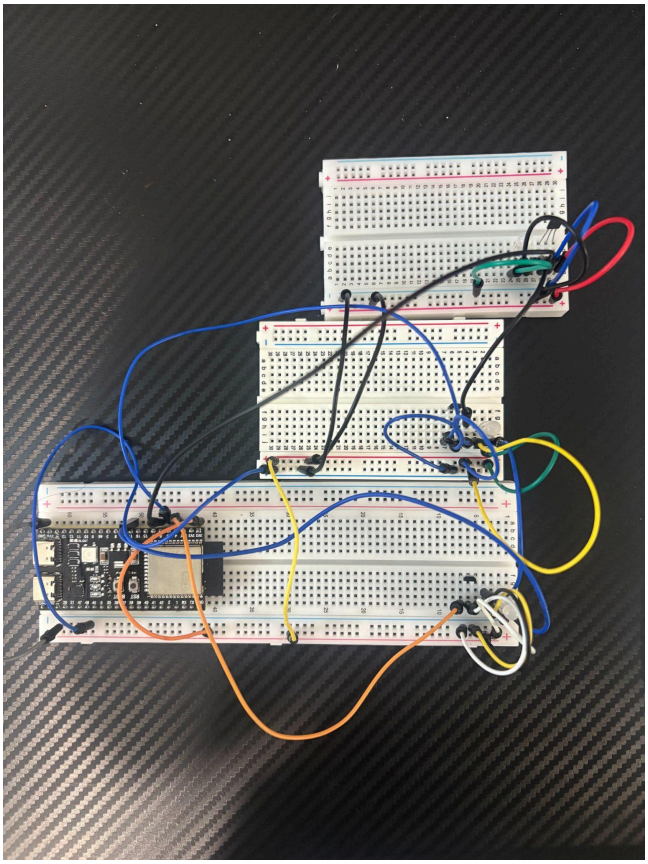
Hardware Component	**Test method**	**Expected Outcome**	**Notes**
ESP32	Sensor toggling	Stable HIGH/LOW readings	Mini Demo
MCP23017 Expanders	I²C bus scan + register check	All addresses are detected, correct configuration	Final System
Sensors	Magnet detection	No missed activations	All phases
LED Chain	Color + timing test	Correct visual output, no flicker	NeoPixel timing critical
Power Distribution	Multimeter, load test	Stable 5V and 3.3V rails	Ensures LED reliability

9.2.1 Mini Demo Hardware Testing (Direct GPIO Inputs + LCD Display)

The SD1 mini demonstration focuses on validating the fundamental sensing and LED-response logic using the ESP32-S3's GPIO pins. Since the mini demo uses only a small subset of sensors, each wired directly to the microcontroller, this phase emphasizes electrical

correctness, timing behavior, and stable digital signal detection without the added complexity of I/O expanders. Testing begins by individually verifying each sensor's ability to produce a clean HIGH or LOW signal when a chess piece with a magnet is placed on or lifted from its corresponding square. This includes confirming proper pull-up configurations, ensuring that no pins float during idle states, and measuring the consistency of sensor output across multiple scenarios. Variations in magnet size, magnet distance, and square alignment are also evaluated to ensure that the sensing mechanism is not overly sensitive to small positional changes.

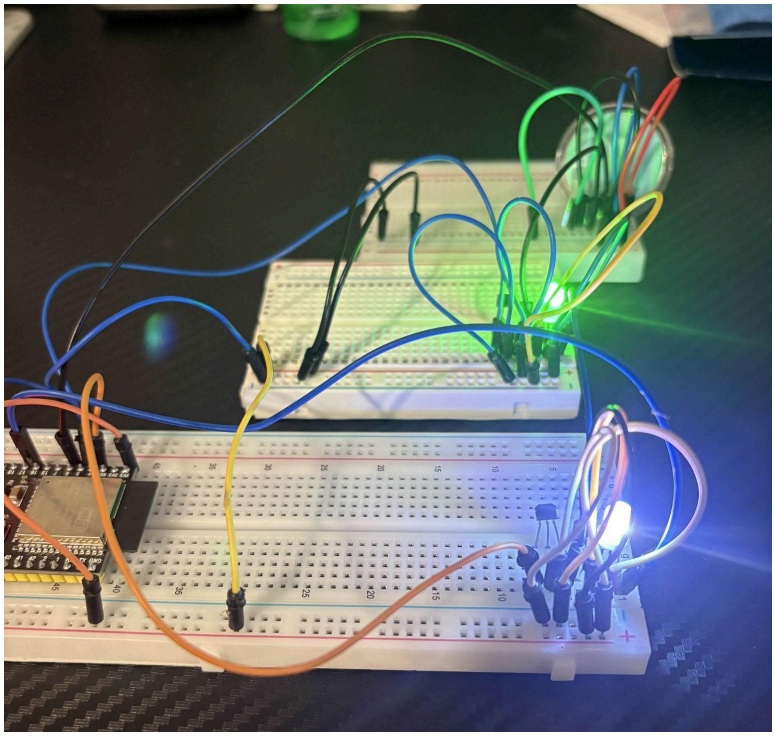
Figure 9-1: Mini Demo Breadboard Design



Wiring validation is another core component of this test phase. The image above shows our demo layout, which closely mimics the structure and signal flow of the final full-scale PCB. The WS2812 LEDs are daisy-chained and driven from a single ESP32-S3 GPIO pin, allowing the team to validate data propagation, power stability, and signal integrity along the LED chain. Each Hall-effect sensor requires a dedicated GPIO input; in the demo, these sensor outputs are wired directly to the ESP32-S3, while the final system will employ four I/O expanders to accommodate all 64 sensor inputs. Each sensor line is checked against the GPIO pin assignments defined in the software's board-mapping array to confirm correct routing and ensure that wiring errors or loose connections do not compromise real-time detection.

After all sensor inputs are validated, the LED subsystem undergoes functional testing through progressive lighting patterns, color-change routines, and move-feedback indicators. These tests confirm that the daisy-chained LEDs respond reliably to control signals and that the expected colors and patterns propagate correctly across the chain. Timing behavior—particularly during animations or rapid color transitions—is also analyzed to ensure that LED update operations do not interfere with continuous sensor polling or overall system responsiveness.

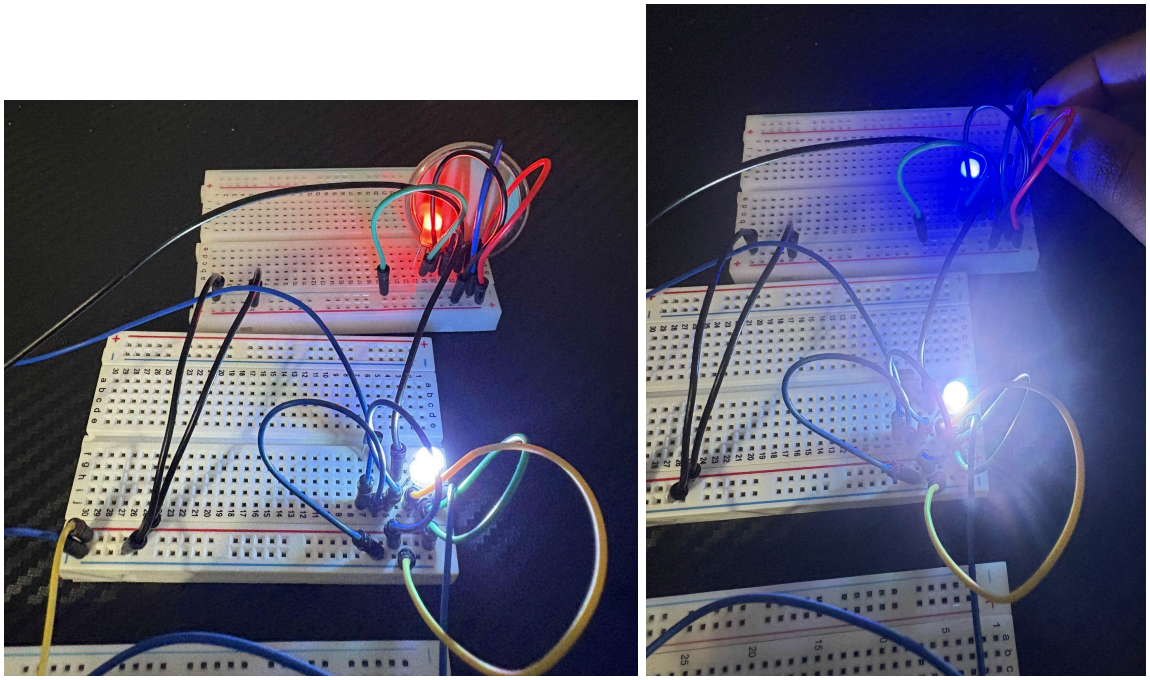
Figure 9-2: Pawn lifted scenario



The image above shows the demo scenario where the pawn on a2 is lifted. The system detects the lift and lights a2 white to mark the origin square. It then computes the legal move to a3, lighting a3 green to indicate the valid destination. This demonstrates correct lift detection, move validation, and real-time LED feedback.

Finally, the mini demo undergoes integrated interaction testing, where real chess interactions—such as lifting a pawn, placing it on a new square, and performing simple captures—are executed to verify that sensor inputs and LED outputs operate cohesively. These tests form the foundational verification stage, ensuring that all core hardware behaviors function reliably before introducing the 64 GPIO connections through the I/O expanders used in the full-scale system.

Figure 9-3: Capture Scenario

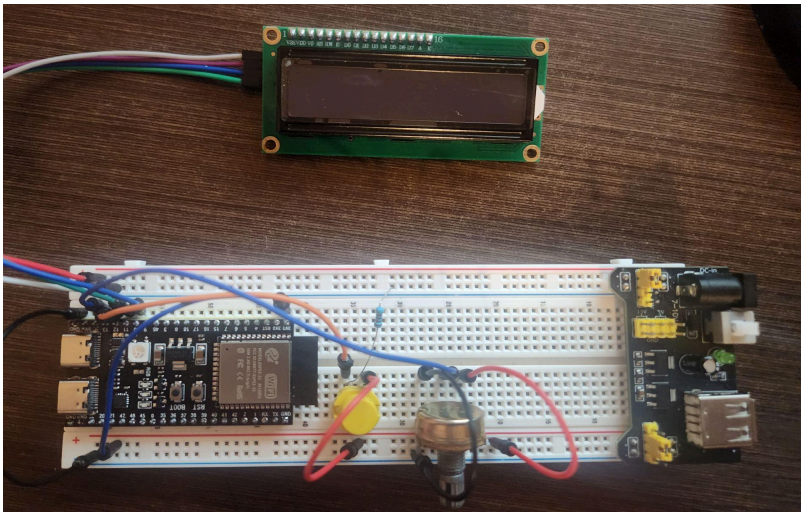


Demo: capture available (left), **capture completed** (right)

The image above (left) shows the mini demo where a pawn is lifted from a3 and an opposing piece is present on b4, which is illuminated red to indicate a capturable enemy square. The image above (right) shows the pawn moving from a3 to b4, where the LED blinks blue to signify that the capture action has been successfully completed.

Another aspect of the mini demo is the demonstration of the LCD response and user selections to first set up their game. We want to validate the I^2C bus which is being used with the LCD in our test and then the MCP23017 I/O expanders in our full product. Testing begins by first making sure we have implemented the LCD screen, potentiometer, and button correctly. We have applied an active low resistor to our button to ensure the value reads LOW when not pushed. We also ensure the potentiometer is set up correctly, reading from ground from one side, and from VDD from the other, outputting through the middle to the board. We configure the SDA and SCL pins on the board to our selected ones to ensure the LCD screen works correctly.

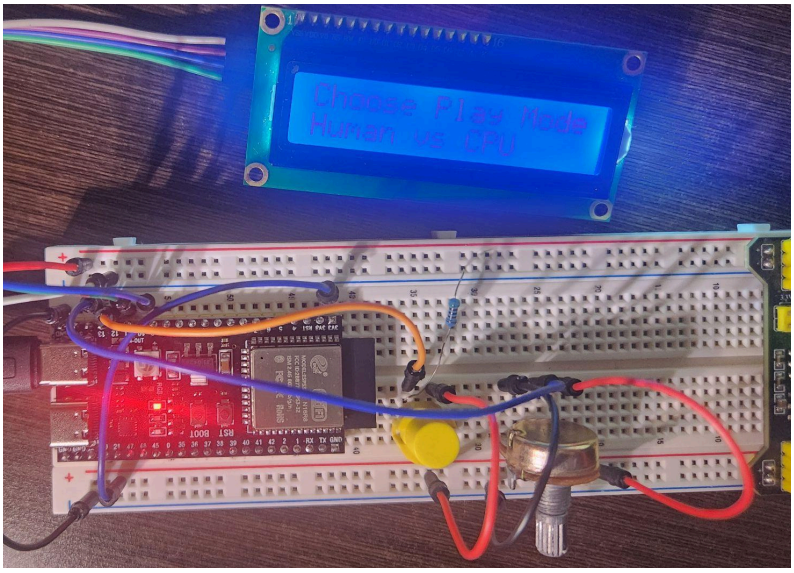
Figure 9-4: Mini Demo Breadboard Design 2



The image above shows the breadboard layout for the LCD, button, and potentiometer. The button and potentiometer are connected to VDD and one global I/O pin each. We also attach a resistor to the button where it connects to the board to function as a pull down resistor. For the LCD screen we connect it to ground, and for VDD we need to apply 5 volts, which in our final product will be done using a logic level shifter. To connect the SDA and SCL pins to allow for I^2C communication we first map those pins in the software as the ESP32 S3 allows for custom placement for the I^2C communication pins.

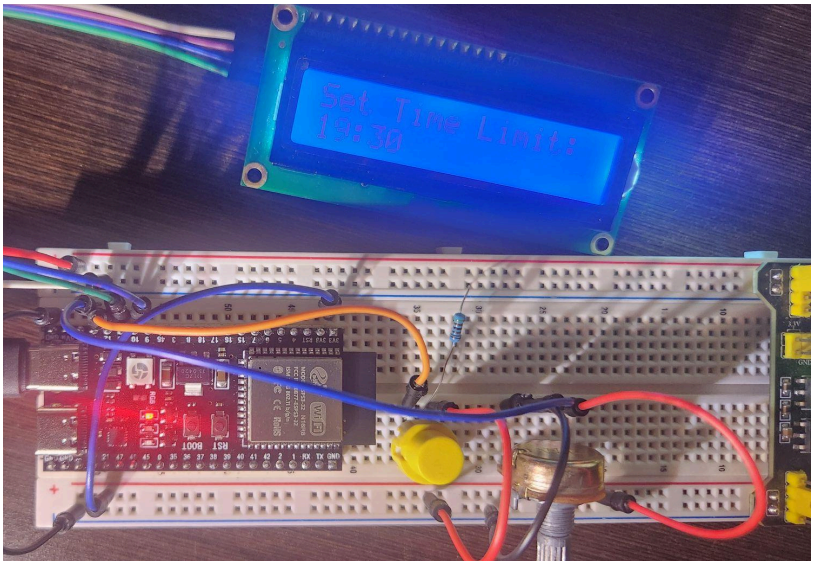
After we confirm that we receive digital input from the button and analog input from the potentiometer, we then confirm the LCD screen acts accordingly when we update it, and can then move on to further testing.

Figure 9-5: Initial Setup



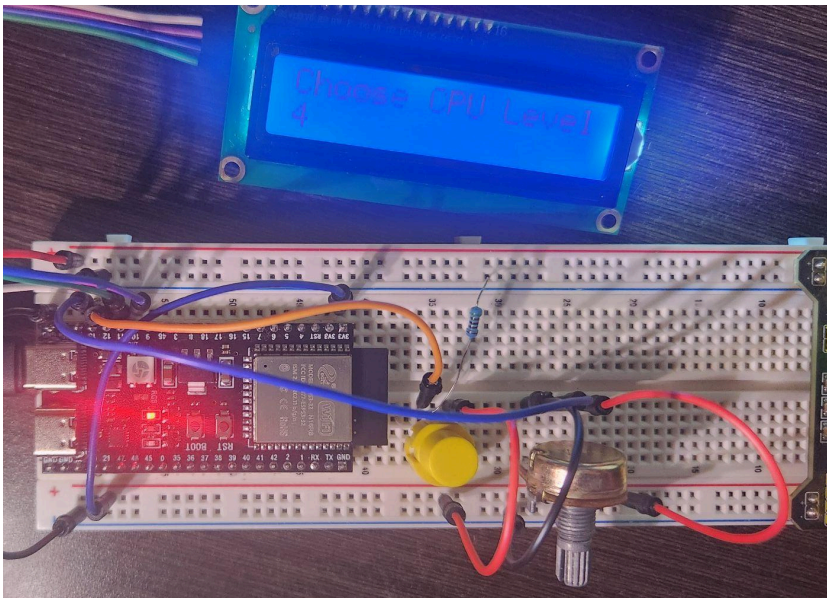
The above image depicts the beginning state shown to the user. The user is presented with two options which are controlled by the potentiometer, if it is set to over half then the user is set to “Human vs. CPU” and if under then “Human vs. Human”. The user then presses the button to select that mode and move on to the next screen.

Figure 9-6: Time Select Screen



The image above shows the second screen the user will see regardless of their choice between a human or computer opponent. Here the user uses the potentiometer again to choose the allotted time for the game. The timer increases in increments of 15 seconds, and if maxed out to the highest voltage, the timer will show “99:99” indicating no limit set for the current game.

Figure 9-7: CPU Level Selection



The above image shows the selection the user must make if they have selected to go against a CPU. The user uses the potentiometer again to select the level, with the higher the voltage level the higher the CPU level given. When selected, the user presses a button and the system will continue to start the game.

9.2.2 System Hardware Testing for Final System (I2C + MCP23017 Expanders)

The SD2 full-scale hardware implementation will introduce the MCP23017 I/O expanders, which will provide the 64 GPIO inputs required to monitor every chessboard sensor. Testing during this phase will focus on ensuring stable I²C communication, validating correct device addressing, and confirming that each expander pin reliably reads the associated sensor state. Because this stage represents the complete hardware architecture of the final system, the tests will prioritize robustness, consistency, and performance across all components operating simultaneously.

The first step in this phase will involve verifying that each MCP23017 properly appears on the I²C bus and responds to configuration commands issued by the ESP32-S3. Once communication is confirmed, all 64 sensor input pins will be assigned and tested according to the system's row–column indexing scheme. Each sensor will be activated individually to ensure that the detected board position matches the intended mapping in software. Signal stability will be evaluated by observing whether sensor readings remain consistent during repeated activations and under varying chess logic conditions.

Since the full system will require the ESP32-S3 to poll sensor states continuously, timing analysis will be performed to confirm that the microcontroller can read all 64 sensors quickly

enough to track real-time piece movement without latency. While these I²C operations are taking place, the LED subsystem will be tested concurrently to verify that LED animations and color updates remain smooth, responsive, and free from flicker or delays. This will ensure that I²C bus activity does not interfere with the timing-sensitive LED control protocol. Ultimately, the full-scale hardware testing will ensure that the complete SD2 chessboard functions reliably under real-world usage and supports accurate, responsive piece detection during gameplay.

9.3 Software Testing

Software testing ensures that the firmware accurately reads sensor inputs, maintains board state, validates movement rules, and drives LEDs. The software is written in Arduino C and represents the board as a two-dimensional array, which allows visualization of piece movement and legal moves. Testing is performed for both SD1 and final configurations.

Table 9-2: Software Component Testing

Software Component	**Testing Method**	**Expected Behavior**
Board Matrix	Sensor reads	Accurate 8x8 mapping
Movement logic	legal/illegal move tests	Correct rule interpretation
Sensor Debouncing	Rapid toggling	No false triggers
LED output	Color and timing	Correct LEDs
I2C Data handling	Register polling	Accurate readings

9.3.1 Sensor Grid Logic Testing

The software will maintain an 8×8 two-dimensional array chess engine representing the full chessboard, where each element corresponds to a specific sensor position. This grid will function as the central data structure for tracking piece presence, detecting movement, and triggering LED visualizations. Sensor grid logic testing will ensure that any change in a sensor's

physical state, whether a piece is lifted or placed, is accurately and consistently reflected in the software's internal representation.

For testing the sensor our main issue to test was making sure they are triggered only when they need to and the software responds accordingly. When using the hall effect sensor, bringing a magnet close to it causes it to activate, sending a signal to the controller to activate the respective functions. One problem we may face, however, is that when the sensor is activated, it may in fact bounce between high and low signals for a bit. We look to solve this using debouncing.

Debouncing is a technique to ensure that a function or event is not called too quickly in succession, and with our sensors we need to make sure they are activated once when the magnet applies to them. If not it could lead to errors in the game logic, causing problems for the player such as false moves. We implement debouncing by using timers. First we set our specified delay time, that being the amount of time we want between activations. We then check the time when a signal is received against when the previous signal was received, if it is greater than our delay time we can be sure this is a separate call and run the function.

We then test this feature by setting up the serial monitor, which allows communication between our development board and the computer where we program the board. We rapidly toggle the sensor high and low, and have the computer output when the sensor is activated. If we have implemented our system correctly then the monitor should output only once, and we know we can move on to integrating this system with the chess logic and LED.

9.3.2 Movement Logic and Rule Evaluation Testing

Movement logic and rule evaluation testing will focus on verifying that the system correctly interfaces with the integrated chess engine, which will serve as the authoritative source for determining legal and illegal moves. Instead of implementing rule evaluation directly in firmware, the software will transmit the current board state—represented by the 8×8 sensor grid to the chess engine, which will return the set of valid moves for the selected piece. This approach ensures rule accuracy, simplifies firmware design, and guarantees compliance with standard chess behavior.

The returned move set will be validated through unit tests to ensure correct interpretation and mapping to LED indicators. Legal moves will be illuminated in green, illegal squares will be highlighted in red if selected, and the origin square will appear white. These outputs will confirm that the system accurately reflects the engine's decision-making.

Functional testing will simulate movement sequences where the software updates the engine's internal board state after each completed move. Automated test scripts will feed predefined scenarios—such as attempted captures, blocked pawns, and edge-of-board conditions into the system to verify that the engine provides consistent and correct evaluations. Each result will be cross-checked with expected engine output to confirm system alignment.

These tests will also verify that debouncing logic prevents noisy sensor transitions from generating unintended engine queries. Overall, this testing phase will ensure that the software correctly utilizes the chess engine to manage rules while maintaining accurate LED visualization and real-time responsiveness.

9.3.3 LED Visualization Testing

LED visualization testing will verify that the NeoPixel lighting system operates reliably, accurately, and responsively under real-time gameplay conditions. Since the LED strip will provide immediate visual feedback for move validation, square highlighting, and error indication, it is critical that the system maintains a consistent color output and timing performance. Testing will evaluate the color accuracy, update reliability, and the system's ability to handle rapid state changes.

The subsystem will be stress-tested by generating fast, successive update calls that simulate typical gameplay scenarios, including multiple piece movements and recalculated valid-move indicators. These tests will ensure that the microcontroller can maintain correct LED data sequences even when other tasks, such as I²C polling or grid-state updates occur concurrently. The goal will be to confirm that the LEDs do not flicker, shift colors unexpectedly, or experience synchronization errors during high-frequency updates.

Latency between sensor detection and LED response will also be measured. This will involve lifting and placing pieces in controlled intervals while recording the time required for LEDs to update the highlighted squares. The system must maintain near-real-time responsiveness to provide intuitive feedback to the user. Performance will be validated both in simulation during development and again once the full hardware system is available.

Additionally, long-duration stability tests will be conducted to ensure that LEDs maintain accurate color representation and consistent brightness during extended operation. This will help identify any timing drift, power inconsistencies, or overheating that could degrade visual accuracy. Together, these tests will confirm that the LED visualization subsystem meets the responsiveness and reliability required for a functional interactive chessboard.

9.4 Integration Testing

Integration testing will evaluate how the hardware, firmware, chess engine, and LED indication system will operate when combined into a unified platform. This phase will verify that subsystem interactions will be reliable, synchronized, and capable of supporting real-time chess rule evaluation.

9.4.1 Integration Testing in Mini Demo

Integration testing for the mini demo focused on validating the interaction between the sensors, firmware, and LED feedback system before the full chess engine and complete hardware were introduced. During this phase, individual sensors were tested to ensure they correctly detected piece lifts and placements. The firmware was then evaluated to confirm that raw sensor inputs were interpreted as valid board-state events.

The LED indication system was tested to verify that the origin square illuminated white, legal squares illuminated green, and illegal squares illuminated red based on simplified movement logic. Although the full chess engine was not yet integrated, a basic ruleset was implemented to confirm that the data pipeline from sensor detection to firmware processing to LED output operated correctly.

This stage demonstrated that the core subsystems worked together reliably on a small scale. It confirmed that the system could successfully capture sensor events, process them through firmware logic, and produce accurate real-time visual feedback using the LED system.

9.4.2 Integration Testing for Final System

Integration testing for the final system will expand on the mini demo by integrating the full board, all sensors, and the chess engine responsible for complete rule evaluation.

In this phase, the microcontroller will transmit board-state updates to the chess engine, which will determine legal moves based on the full rules of chess. Integration tests will verify that engine feedback will be received correctly by the firmware and translated into LED commands without delay. The LED grid will be evaluated to confirm that legal squares will illuminate green, illegal squares will illuminate red, and the starting square will illuminate white for every type of move, including captures, promotions, en passant, castling, and other complex scenarios.

System-wide stress tests will be performed to confirm that the engine will handle rapid state changes, that debouncing algorithms will prevent false inputs, and that communication between all components will remain stable. The final integration stage will ensure that the entire system, hardware, software, and chess engine will function together seamlessly under real operating conditions.

9.5 System-Level Evaluation

System-level evaluation will focus on validating the complete end-to-end performance of the final chessboard under realistic operating conditions. This phase will measure how well the sensors, MCP23017 expanders, ESP32-S3 firmware, LED visualization system, and chess engine operate together as a unified system.

Latency will be evaluated by measuring the time between a sensor activation and the corresponding LED update. Reliability testing will involve repeatedly cycling through simulated and physical sensor activations to ensure that no events are missed, duplicated, or corrupted during I²C communication or software processing.

Stress testing will simulate rapid or erratic gameplay to confirm that the I²C bus remains stable under high transaction loads, and that the software correctly processes continuous position updates without logic failures or visual glitches. The LED subsystem will be monitored for

timing accuracy, ensuring that simultaneous updates across the board do not introduce flicker or timing violations.

Finally, power and thermal performance will be evaluated under extended operation. The ESP32-S3, MCP23017 expanders, and LED array will be tested to confirm that each device operates within safe electrical and thermal limits during prolonged gameplay. This ensures that the completed system remains stable, responsive, and reliable in real-world use.

9.6 SD2 Preparation and Transition to Final System

The SD2 demonstration served as the final, fully functional presentation of the system. In that phase, the complete hardware, firmware, and LED visualization subsystems are integrated to showcase the finished smart chessboard. Unlike earlier demonstrations that used a reduced number of sensors, the SD2 version incorporates the full set of chessboard inputs through MCP23017 I/O expanders, allowing all 64 squares to be monitored simultaneously. This final configuration highlights the system's scalability, reliability, and readiness for real-world use.

The demonstration showcases accurate piece-lift detection across the entire board, ensuring that each sensor correctly identifies when any chess piece is lifted or placed. The system fully integrates the chess engine, enabling it to evaluate legal and illegal moves based on actual board state rather than predefined logic. This includes validation of standard pawn movement rules, diagonal captures, blocked paths, and illegal moves, providing a comprehensive representation of real gameplay conditions.

LED visualization is also demonstrated in its final form. When a piece is lifted, the origin square will illuminate white, legal move destinations will illuminate green, and illegal squares will display yellow when attempted. The LED chain will respond in real time across all 64 tiles, confirming that the NeoPixel data stream remains stable under full-load operation.

In SD2, it also highlights the system's ability to maintain fast processing speeds, stable I²C communication, and consistent LED output during stress scenarios such as rapid square activations or consecutive piece movements. This demonstration verifies the system's final performance, ensuring that the hardware integration, movement evaluation logic, and LED feedback operate reliably and cohesively as a completed product.

In our conclusion of SD2, the team has validated that the system meets all operational expectations for accuracy, responsiveness, stability, and user interaction, representing the finalized implementation of the smart chessboard project.

Chapter 10: Administrative Content

10.1 Bill of Materials

Component:	Part name:	Quantity:	Unit Price:	Total Price:
Microcontroller	ESP32-S3-WROOM-1-N8R2	4	$5.71	$22.84
Button	TS02-66-70-BK-160-LCR-D	3	$0.10	$0.30
Rocker Switch	EG5619-ND	1	$6.39	11.80
Potentiometer	09806 Trim Pot	1	$1.25	$1.25
Batteries	TENERGY 1.2V NiMH Rechargeable D Batteries 4 Pack	1	$24.99	$27.98
Battery Holder	4-D battery holder	1	$4.49	$4.49
Development Board	ESP32-S3	1	$20	$20
I/O Expander	MCP23017	8	$1.62	$12.96
3.3 voltage regulator	TPS63000DRCR	3	$2.61	$7.38
5 Volt regulator	TPS61022RWUR	1	$1.68	$1.68
5 Volt Regulator	LTC3113	3	$12.08	$36.24
Logic shifter	SN74AHCT125DR	3	$0.39	$1.17
LCD screen	I2C 1602 LCD Display	1	$8.95	$8.95
LEDs	WS2812B addressable LEDs	128	$0.267	$34.18
64 Hall effect	DRV5032	100	$0.276	$27.60

Sensors				
Frosted acrylic top	1	1	$39.99	$42.59
32 chess pieces	Chess Pieces	32	$8.73	$8.73
Magnets	Neodymium magnets	32	$0.07	$6
Frame Casing	16x17 Shadow Box frame	1	$67.95	$72.37
Custom PCB	PCB	1	$366.29	$366.29
Misc:				$124.51
TOTAL COST:				$839.31

10.2 Project Milestones (SD1 & SD2)

Senior Design 1 Task	**Expected Completion**	**Progress**
Divide & Conquer Document	9/5/2025	Completed
First Meeting with Mentor	9/10/2025	Completed
Research Chess Libraries (Solo and CPU)	9/20/2025	Completed
Start Gathering Materials	10/01/2025	Completed
Midterm Report	10/31/2025	Completed
Finish Breadboard Prototype	11/03/2025	Completed
Finish PCB Prototype	11/15/2025	Completed
Final Report	11/25/2025	Completed
Mini Demo Video	11/25/2025	Completed
Committee Meeting	December 2025	Completed

Senior Design 2 Task	**Expected Completion**	**Progress**
Order PCB	December 2025	Completed
Construct PCB	December 2025	Completed
PCB and Software Testing	January 2026	Completed
General Testing and Bug Fixing	January 2026	Completed
Meeting with Mentor	February 2026	Completed
Project Completion	April 2026	Completed
Final Demo Video	April 2026	Completed
Final Check Before Presentation	April 2026	Completed

Final Presentation	April 2026	Completed

10.3 Work Distribution

Andres Armendariz	Hardware Research PCB Design + Construction Hardware Testing
Freddy Pacheco	PCB Construction Software Research Hardware Testing
Ryan Ramdihal	Software Research Hardware Testing Software Integration

Chapter 11: Conclusion

This paper describes the development process and progress throughout Senior Design I and Senior Design II for the Pathfinder Chessboard, an interactive electronic chessboard designed to assist players by illuminating legal moves and providing real time move validation. The system uses hall effect sensors to detect piece positions and an LED matrix to guide gameplay. The Pathfinder chessboard serves as both an educational tool for beginners and an accessible aid for players who benefit from virtual assistance. Senior Design I focuses on the discovery, planning and validation phases of creating a working PCB project. Throughout the Senior Design I semester, this team identified key constraints, planned details, and prototyped the foundational design needed for the successful implementation in Senior Design II.

The development process began with extensive planning and research. The team conducted reviews over past similar projects, created detailed hardware and software block diagrams, and evaluated components needed for a reliable system performance. After researching and picking the components we were able to establish some groundwork for the Pathfinder Chessboard project. Building on this groundwork, the team also created the initial schematics which focused on laying out the electrical groundwork for the MCU, power system, LEDs, sensors, etc ensuring that each subsystem could operate reliably within the board's specifications and constraints.

In conjunction with researching components the team also researched and planned out the software needed for the project. With multiple different approaches available we choose to use

the external chess engine Micro-Max to handle game logic for displaying the best moves. The team also created our own custom game state handling logic in order to provide real-time feedback and prevent accidental or uninformed moves. Together these decisions established a solid foundation for the Pathfinder Chessboard.

In order to validate these hardware and software decisions, the team built a breadboard prototype on the completed schematic. This prototype allowed for testing key subsystems, including hall effect sensor detection, microcontroller signal handling, LED behavior, and peripheral subsystem management. Not only did the prototype allow for testing of the hardware but in conjunction with the software we were able to implement a working prototype of the Pathfinder Chessboard. The successful breadboard implementation provided confidence in the electrical and software design decisions and confirmed readiness for PCB development in Senior Design II.

The hardware and software design process for the Pathfinder Chessboard involved careful planning and execution with a focus on power, micro controller integration and overall system reliability. The successful prototyping of these components allowed for the completion of the prototype PCB design. When designing the PCB the team had to take into account signal integrity, trace length, and board dimensions along with other manufacturing and performance constraints.

Senior Design II focused on transforming this groundwork into a complete and operational system. The team finalized and assembled the custom PCB, integrating the ESP32-S3, I/O expanders, hall effect sensors, and LED matrix into a unified design. Careful attention to power regulation, signal routing, and physical constraints resulted in a stable and reliable hardware platform capable of supporting all required functionality.

A major focus of Senior Design II was the seamless integration between hardware and software. The team refined the embedded software to ensure accurate real-time detection of chess piece movements using hall effect sensors, while also providing immediate visual feedback through the LED matrix. The implementation of custom game state logic, combined with the Micro-Max chess engine, enabled the system to validate moves, prevent illegal actions, and suggest optimal gameplay decisions. These features collectively enhanced the responsiveness and usability of the system, ensuring that it met both functional and performance requirements.

However, the development process was not without its challenges. Throughout both semesters, the team identified several challenges and areas requiring refinement before moving on to our final product. These included power distribution concerns, microcontroller pin-mapping limitations, and software-hardware integration complexities. These challenges reinforced an important lesson in that building a project from scratch is not a linear process but instead iterative refinement. Key decisions such as the inclusion of the LTC3113 5v regulator, TPS63000DRCR 3.3 voltage regulator, MCP23017 I/O expanders and ESP32-S3 are some examples of parts that replaced other components due to not fitting the specifications of the system.

The software development process followed a similarly iterative and nonlinear path. One challenge we faced when developing the code is that the magnets were not inherently tied to each of the pieces. This meant that additional logic had to be implemented in order to accurately map sensor activations and validate moves accordingly. By adding tracker arrays and by the use of interrupts with the chess engine and sensors, we were able to overcome logic coding concerns for our game state and increase response time for our project.

The final implementation of the Pathfinder Chessboard represents a fully functional and integrated system that achieves its original design objectives. The completed project demonstrates the successful application of embedded systems design, real-time processing, and hardware-software co-design principles. By providing illuminated legal moves, real-time move validation, and interactive feedback, the system serves as both an educational tool for beginners and an assistive device for users who benefit from enhanced visual guidance.

In addition to meeting its technical goals, the Pathfinder Chessboard offers significant value in educational and accessibility contexts. Its interactive design promotes engagement, reinforces proper gameplay mechanics, and reduces the learning curve for new players. The system also establishes a foundation for future enhancements, such as wireless connectivity, mobile application integration, and expanded user interface features

Overall, the progression from Senior Design I to Senior Design II demonstrates a complete engineering lifecycle, from concept development and validation to full system implementation and refinement. With a fully operational prototype, validated PCB design, and integrated software system, the team has successfully delivered a robust and reliable Pathfinder Chessboard. This project not only fulfills its intended purpose but also exemplifies the effectiveness of structured engineering design processes and collaborative development in creating complex embedded systems.

Appendix A: References

Matthew (MisterCutie). "History of Chess: The Basics." Chess.Com, 28 Jan. 2009,] https://www.chess.com/article/view/the-history-of-chess.

DIY Machines. (2021, March 25). DIY Super Smart Chessboard | Play Online or against Rasp Pi. Hackster.io. Retrieved September 3, 2025, from Hackster.io website. https://www.hackster.io/diy-machines/diy-super-smart-chessboard-play-online-or-against-rasp-pi-b57daa

*Andres Perez. "UCF Fall 2023 Senior Design 2 Group 13 Final Demo Presentation." *YouTube* 29th November 2023,* https://www.youtube.com/watch?v=n4JKfAw31GQ.

Stockfish. Stockfish – Strong open-source chess engine, 2025–present, stockfishchess.org
"MaxBotix Inc." MaxBotix, 2025, maxbotix.com/blogs/blog/how-ultrasonic-sensors-work?srsltid=AfmBOooGXP07a7AdD1RTegGyavo3t_QaIfDNGfV8tS8VfYCJigrSlzx

ada, lady. "Li-Ion & LiPoly Batteries." Adafruit Learning System, 29 July 2012, learn.adafruit.com/li-ion-and-lipoly-batteries?view=all&gad_source=1&gad_campaignid=21079267614&gbraid=0AAAADx9JvQD5IPiIQa2pFQlS-Vs3ztXA&gclid=CjwKCAjw04HIBhB8EiwA8jGNbaz1kNg1SjsYcyBJ5XvQOCFceWa4CR4p_HTy6p7sq50ctTDigjlvQhoCSyMQAvD_BwE

SSZT164 Technical Article
Www.ti.com, 2023, www.ti.com/document-viewer/lit/html/SSZT164.

"Client Challenge." Monolithicpower.com, 2025, www.monolithicpower.com/en/learning/resources/hall-effect-sensors-a-comprehensive-guide?srsltid=AfmBOopazyS9MtK2FmSj4oXThAQgzdocOpD0SQXyt29u3KYptMvs71LO

What Is a Capacitive Sensor?
*Balluff.*https://www.balluff.com/en-us/blog/what-is-a-capacitive-sensor

What is a piezoelectric sensor?
https://www.electronicsforu.com/technology-trends/learn-electronics/piezoelectric-sensics
Stykemain, Adam . “Inductive Sensor Explained | Different Types and Applications - RealPars.” *Www.realpars.com*, 30 June 2021, www.realpars.com/blog/inductive-sensor

IEEE Standards Association. (n.d.). *What are standards? Why are they important?* IEEE. Retrieved October 20, 2025, from https://standards.ieee.org/beyond-standards/what-are-standards-why-are-they-important/

International Organization for Standardization. (2018). *Information technology — Programming languages — C* (ISO/IEC 9899:2018). https://www.iso.org/standard/74528.html

IPC, Generic Standard on Printed Board Design, IPC-2221A, Jan. 2012. [Online]. Available: https://www.ipc.org/

Underwriters Laboratories. (2024). ANSI/UL 8750: Standard for safety — Light emitting diode (LED) equipment for use in lighting products (Ed. 2). ANSI/UL. https://www.intertek.com/siteassets/resources/sun/2024/ul-8750-sun-rev-12-7-2022-ed-5-1-2025.pdf

International Organization for Standardization, International Electrotechnical Commission & Institute of Electrical and Electronics Engineers. (2021). ISO/IEC/IEEE 29119-4:2021 – Software and systems engineering — Software testing — Part 4: Test design techniques. https://standards.iteh.ai/catalog/standards/sist/20536fd0-e74a-4b3c-8285-af54a8441204/iso-iec-ieee-29119-4-2021

IEEE Standards Association. (2015). IEEE recommended practices for modulating current in high-brightness LEDs for mitigating health risks to viewers (IEEE Standard 1789-2015). https://standards.ieee.org/ieee/1789/4479/

Institute of Electrical and Electronics Engineers. (2024). IEEE Standard for Information Technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (IEEE Std 802.11-2024). https://standards.ieee.org/ieee/802.11/10548/

Appendix B: Copyright Permissions

Micro-Max (© 2002–2009 H.G. Muller) is used in accordance with the original license included in the Micro-Max source code.

Appendix C: Datasheets

WS2812B LED

https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

ESP32-S3-WROOM-1

https://documentation.espressif.com/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

LCD Screen 1602

https://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf

Buck-Booster Converter

https://www.ti.com/lit/ds/symlink/tps63002.pdf?ts=1761772102180&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS63002

Arduino Uno

Arduino. *Arduino Uno Rev3 Technical Specifications.* 2024. https://store.arduino.cc/products/arduino-uno-rev3

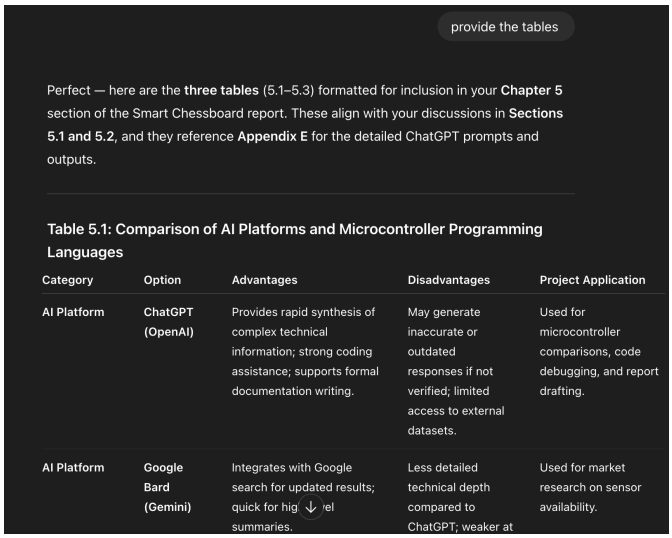
Texas Instruments. *MSP430G2553 Datasheet.* 2024. https://www.ti.com/product/MSP430G2553

Appendix D: Software Code

Initialization of LED colors, sensor pins in direct GPIO, and LED addresses.

```
// --- COLORS ---
uint32_t BLUE;
uint32_t GREEN;
uint32_t WHITE;
uint32_t RED;
uint32_t OFF;

void setup() {
  Serial.begin(115200);

  strip.begin();
  strip.show();

  BLUE  = strip.Color(0, 0, 255);
  GREEN = strip.Color(0, 255, 0);
  WHITE = strip.Color(255, 255, 255);
  RED   = strip.Color(255, 0, 0);
  OFF   = strip.Color(0, 0, 0);

  pinMode(SENSOR1, INPUT_PULLUP);
  pinMode(SENSOR2, INPUT_PULLUP);

  strip.setPixelColor(0, OFF);
  strip.setPixelColor(1, OFF);
  strip.show();
```

Appendix E: ChatGPT Prompts and Outcomes

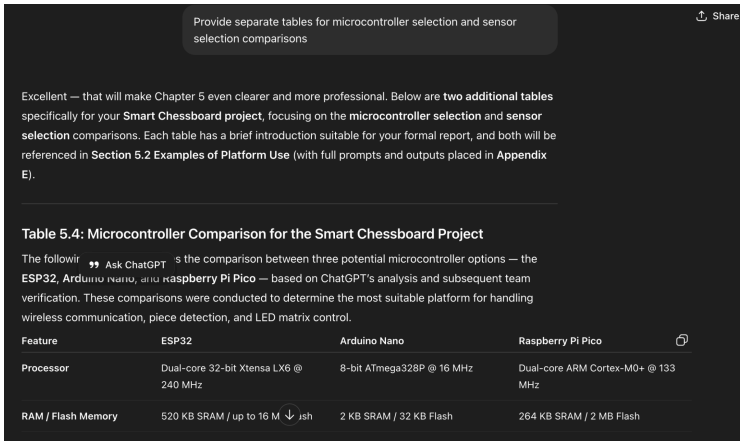


provide the tables

Perfect — here are the **three tables** (5.1–5.3) formatted for inclusion in your **Chapter 5** section of the Smart Chessboard report. These align with your discussions in **Sections 5.1 and 5.2**, and they reference **Appendix E** for the detailed ChatGPT prompts and outputs.

Table 5.1: Comparison of AI Platforms and Microcontroller Programming Languages

Category	Option	Advantages	Disadvantages	Project Application
AI Platform	ChatGPT (OpenAI)	Provides rapid synthesis of complex technical information; strong coding assistance; supports formal documentation writing.	May generate inaccurate or outdated responses if not verified; limited access to external datasets.	Used for microcontroller comparisons, code debugging, and report drafting.
AI Platform	Google Bard (Gemini)	Integrates with Google search for updated results; quick for hig ↓ el summaries.	Less detailed technical depth compared to ChatGPT; weaker at	Used for market research on sensor availability.



Share

Provide separate tables for microcontroller selection and sensor selection comparisons

Excellent — that will make Chapter 5 even clearer and more professional. Below are **two additional tables** specifically for your **Smart Chessboard project**, focusing on the **microcontroller selection** and **sensor selection** comparisons. Each table has a brief introduction suitable for your formal report, and both will be referenced in **Section 5.2 Examples of Platform Use** (with full prompts and outputs placed in **Appendix E**).

Table 5.4: Microcontroller Comparison for the Smart Chessboard Project

Ask ChatGPT

The followir s the comparison between three potential microcontroller options — the **ESP32**, **Arduino Nano**, and **Raspberry Pi Pico** — based on ChatGPT's analysis and subsequent team verification. These comparisons were conducted to determine the most suitable platform for handling wireless communication, piece detection, and LED matrix control.

Feature	ESP32	Arduino Nano	Raspberry Pi Pico
Processor	Dual-core 32-bit Xtensa LX6 @ 240 MHz	8-bit ATmega328P @ 16 MHz	Dual-core ARM Cortex-M0+ @ 133 MHz
RAM / Flash Memory	520 KB SRAM / up to 16 M ↓ ash	2 KB SRAM / 32 KB Flash	264 KB SRAM / 2 MB Flash

Chapter 5 Prompt for Tables.