Networked Physical Chessboard for Remote Play

ECE 445 Design Document - Spring 2026

Project # 51

Danny Guller, Payton Schutte, Quinn Athas

Professor: Arne Fliflet
TA: Wenjing Song

Table Of Contents

1. Introduction

1.1 Problem

Chess remains one of the most globally recognized strategy games, played recreationally, competitively, and educationally. While digital chess platforms offer unmatched convenience, they fundamentally alter the nature of play. Screen-based interaction removes the tactile feedback of handling physical pieces, reduces spatial awareness of the board, and weakens the sense of presence traditionally associated with face-to-face games. For many players this results in a less enjoyable experience.

The absence of affordable solutions that combine physical play with remote connectivity presents a meaningful gap. Existing commercial electronic chessboards capable of online synchronization are often expensive, limiting accessibility to enthusiasts or specialized institutions. Consequently, players must choose between convenience and the authentic physical experience of the game.

This problem connects to broader societal considerations. Chess is widely associated with cognitive benefits, including improvements in concentration, problem-solving, and strategic thinking. However, increasing reliance on screen-based entertainment contributes to digital fatigue, eye strain, and reduced physical interaction. Systems that encourage tangible interaction while preserving remote connectivity may help mitigate these concerns. Furthermore, enabling richer remote interaction supports social well-being, particularly for individuals separated by distance, mobility limitations, or remote learning environments.

From an economic perspective, reducing barriers to entry through lower-cost technological solutions promotes wider adoption and educational accessibility. Technologically, the project reflects growing interest in hybrid physical-digital systems, telepresence interfaces, and embedded sensing technologies.

1.2 Solution

This project proposes a networked physical chessboard system that enables two remote players to participate in a fully physical game of chess. Each board will detect piece placement and movement using an array of Hall effect sensors embedded beneath the playing surface. Each piece will have a magnet embedded into its bottom. Sensor readings will be processed by a microcontroller to reconstruct board state, infer player moves, and enforce chess move legality.

Once a move is confirmed locally on the digital display, it will be transmitted over Wi-Fi to a centralized server and relayed to the opponent’s board. Visual indicators, including LEDs and an integrated digital display, will guide players in replicating moves and provide system feedback such as connectivity status, turn indication, and move validation. The system is designed for minimal screen usage to maintain the tactile feel of chess.

1.3 Visual Aid

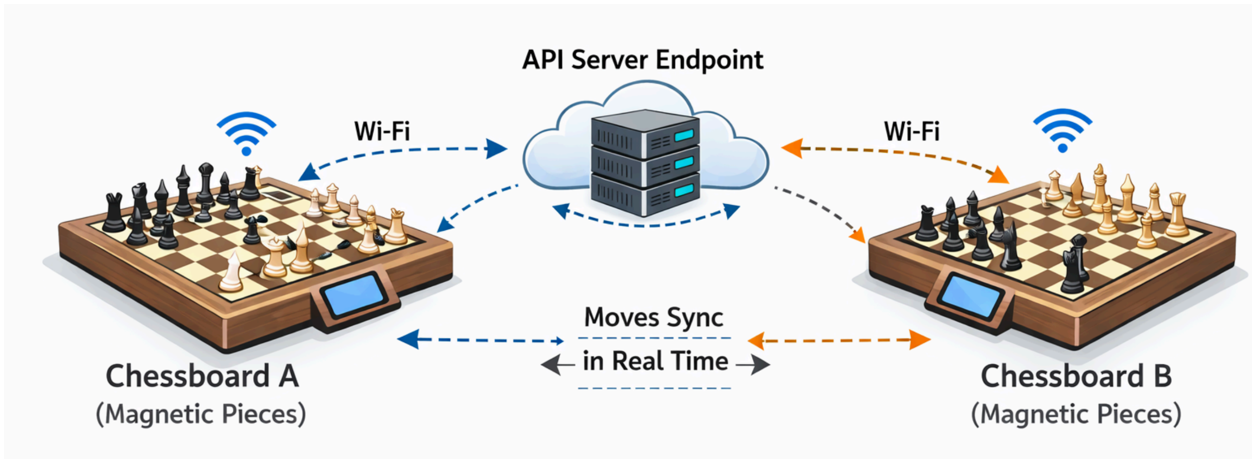


Figure 1: Visual aid of what a networked physical chess setup

The visual aid shows how the networked chessboard system is used in a real-world scenario.

Each player has their own chessboard, complete with magnetic pieces and a built-in display. When a move is made on one board, the system detects the piece movement and sends that information over Wi-Fi to a central API server. The server then forwards the updated move to the other chessboard.

As a result, both boards stay synchronized. A move made by one player is reflected on the other board, allowing two people in different locations to play as if they were sitting across from each other.

1.4 High-Level Requirements

To consider our project successful, our chessboard must fulfill the following:

A. The system shall detect chess piece presence on all 64 squares with a minimum accuracy of **95%** under normal operating conditions.

B. The system shall transmit and receive confirmed moves with an end-to-end latency not exceeding **5 sec** over a standard Wi-Fi connection.

C. The system shall maintain board synchronization during gameplay and recover from temporary network interruptions of up to **30 seconds** without requiring a manual reset.

2. Design

2.1 Block Diagram

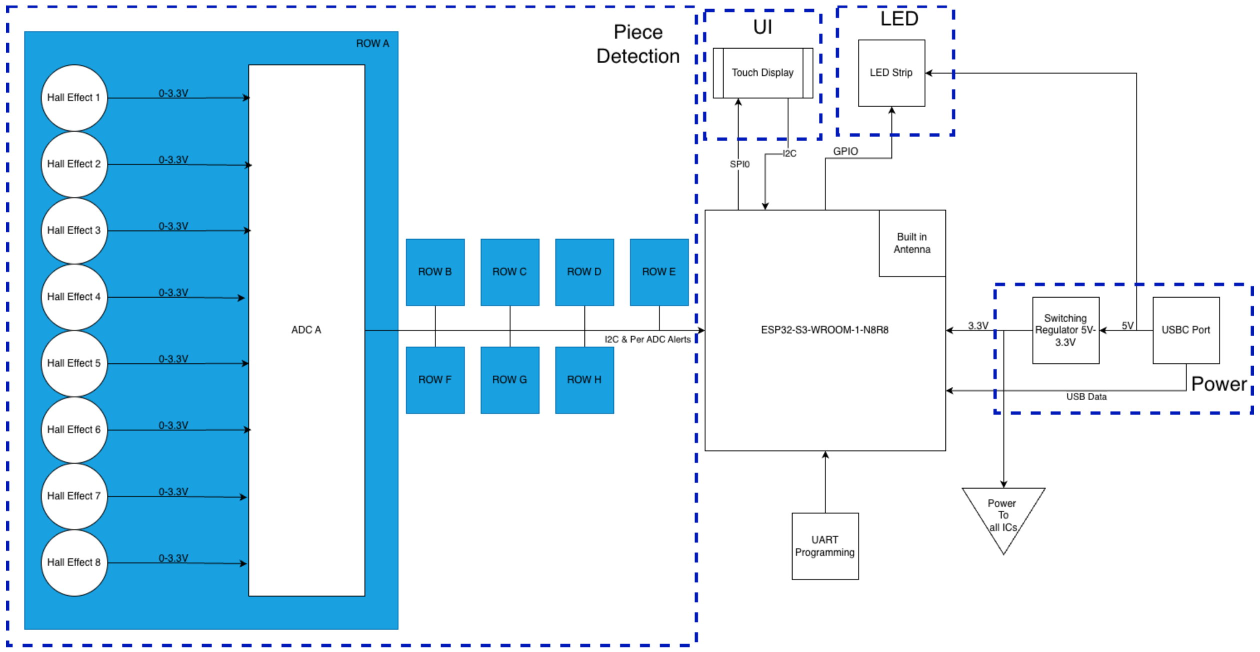


Figure 2: Block diagram of single chess board in system

The system consists of several key subsystems that work together to enable remote chess gameplay. The sensing subsystem uses Hall effect sensors to detect the presence and movement of magnetic chess pieces, allowing the board to accurately determine piece positions. The processing subsystem interprets the sensor data, identifies player moves, and maintains the local board state. The communication subsystem manages wireless connectivity via Wi-Fi, transmitting move information between the chessboard and the API server. The API server serves as a coordination layer, updating the overall game state and managing the selected game mode. Finally, the visual feedback subsystem uses LEDs to display moves and board updates, providing clear, immediate guidance to the players while preserving the physical playing experience.

2.2 Physical Design

The physical design of this project consists of the chess board, the electrical area under the chess board, and the compute/display enclosure. The layout will be a classic chess board of 64 1.5-2” squares with a 4” wide box connected to the side. The enclosure will be about 2” tall in total.

Underneath the playing space, breadboards will be mounted to the bottom plate. This is where the hall effect sensors will be placed so that they are directly underneath their associated square. Breadboards will be used to decrease mess cabling, and provide a way to daisy chain power and ground to each sensor.

Breadboards will also ensure that the sensors are positioned correctly and have no chance of moving, causing inability to read parts of the board.

On the right side of the chess board (from the white players perspective), there will be a 4” extension where the display and main PCB will be housed. The display sits in the middle of this extension so that, no matter the player's color, they will be able to read the display. Directly under the display, the main PCB will be housed. It will stand off the bottom plate with M3 screws and sandoffs.

All wires coming off the main PCB will be 2.54mm pin header connections. This will make wiring to the bread boards under the chess board and to the display easiest. The display already has pin headers for the necessary signals.

Programming can be done through either the UART programming headers on the board or through the USB port located on the opposite side of the hall effect sensor headers. This provides multiple ways to program if one were to fail or if one is unreachable and the PCB must be reprogrammed.

2.3 Hardware Subsystems

2.3.1 Piece Detection Subsystem

This subsystem is responsible for detecting chess piece presence across all 64 squares of the board. Each square is instrumented with a DRV5055A4QLPG linear Hall-effect sensor positioned beneath the playing surface. Chess pieces contain embedded permanent magnets which generate a measurable magnetic field when placed on a square. The Hall-effect sensors produce an analog voltage proportional to the sensed magnetic flux density.

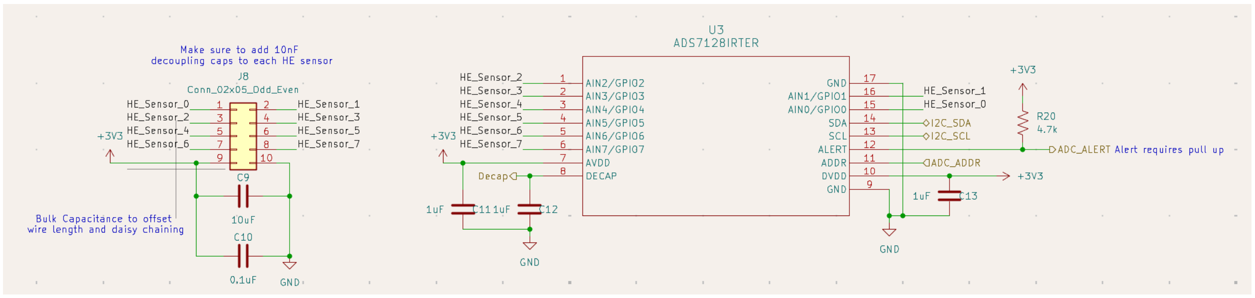


Figure 3: Piece Detection schematic for one row

Sensor outputs are digitized using eight ADS7128 eight-channel analog-to-digital converters, providing a total of 64 sensing channels. The sensing subsystem continuously samples each sensor and provides the board microcontroller (ESP32-S3-WROOM-1) with digitized magnetic field measurements. These measurements are used to determine square occupancy and detect piece placement or removal events.

This subsystem directly contributes to Requirement A, which specifies that piece detection accuracy must be within ±1 milliTesla. Based on the DRV5055A4QLPG sensitivity at 3.3 V operation, a magnetic field variation of 1 milliTesla corresponds to approximately 8.25 milliVolts. Reliable detection therefore requires the subsystem to distinguish voltage variations on the order of single-digit millivolts. The ADC resolution and sampling strategy were selected to ensure stable detection performance.

The ADS7128 and DRV5055A4 are electrically compatible on a shared 3.3 V rail. The ADC uses AVDD as its reference, providing a 0–3.3 V input range. The DRV5055A4 operates from 3.0–3.63 V and produces a ratiometric output centered at VCC/2 (≈1.65 V at 3.3 V), with a linear range of approximately 0.2–3.1 V. This ensures the sensor output remains fully within the ADC's input range without additional conditioning. Because both devices share the same supply, voltage variation is inherently canceled through ratiometric measurement.

With 12-bit resolution, the ADC LSB size is approximately 0.806 mV. Given the sensor sensitivity of 7.5 mV/mT, this corresponds to about 0.11 mT per LSB. Sensor noise (~1.5 mV peak-to-peak) equates to roughly 0.2 mT magnetic noise, which remains well below the ±1 mT requirement. Offset calibration at startup can further improve measurement accuracy.

Table 1: Board Control Subsystem – Requirements & Verification

Requirements	**Verification**
• Taking the Hall-effect sensors as input, the sensing subsystem must determine whether a chess piece is present on an individual square.	• Ensure the board is in an idle state with no chess piece placed on the selected square. Record the sensor reading as reported by the board microcontroller via serial debugging. • Place a chess piece on the selected square. Record the sensor reading via serial debugging. Confirm that the reading differs from the idle measurement by a detectable margin. • Remove the chess piece. Record the sensor reading. Confirm that the measurement returns to the baseline value.
• The sensing subsystem must detect magnetic field variations corresponding to ±1 milliTesla.	• Record the baseline sensor voltage for an empty square. • Place a chess piece on the square and record the sensor voltage change. • Confirm that the observed voltage variation exceeds the minimum detectable voltage corresponding to 1 milliTesla (~8.25 mV). • Repeat measurement across multiple squares to verify consistency.

• The sensing subsystem must maintain stable square occupancy readings without false toggling.	• Place a chess piece on a square and record sensor readings over a fixed observation period (e.g., 10 seconds). • Confirm that the occupancy state does not oscillate between occupied and unoccupied. • Repeat for an empty square and confirm stable baseline readings.
• The sensing subsystem must detect chess piece presence on all 64 squares.	• Sequentially place a chess piece on each square. • Record sensor responses via serial debugging. • Confirm that each square produces a detectable voltage variation when occupied.
• The sensing subsystem must update square occupancy readings within an acceptable detection latency.	• Record sensor readings while placing and removing a chess piece. • Measure the time required for the occupancy state to update. • Confirm detection latency falls within system timing requirements.

2.3.2 Move Processing and Legality Subsystem

This subsystem is responsible for interpreting sensor-derived square occupancy data to reconstruct the board state, detect changes, and determine player moves. The subsystem operates as a software module executed on the ESP32-S3-WROOM-1 microcontroller. By continuously monitoring square occupancy updates provided by the Piece Detection Subsystem, the software maintains a real-time representation of the chessboard configuration.

Board state reconstruction is achieved by storing sequential occupancy maps and identifying state transitions between samples. Changes in square occupancy are interpreted as piece placement or removal events. Using this information, the subsystem generates candidate move hypotheses, including standard moves and capture events. For example, a transition from occupied → empty followed by empty → occupied is interpreted as a potential piece movement, while simultaneous transitions may indicate captures.

Once a candidate move is detected, the subsystem verifies move legality according to standard chess rules. This includes enforcing movement constraints, turn order, and piece interaction rules. Illegal actions such as moving a pawn three squares, moving a piece through another piece, or performing an invalid capture are rejected. Legal moves result in an updated board state, while illegal moves trigger corrective feedback through the User Interface Subsystem

Table 2: Move Processing and Legality Subsystem – Requirements & Verification

Requirements	**Verification**
• Taking square occupancy data as input, the subsystem must reconstruct the current board state.	• Ensure the board is initialized with a known piece configuration. • Record the reconstructed board state via serial debugging output. • Confirm that the reported board state matches the physical piece placement.
• The subsystem must detect piece movement events based on occupancy transitions.	• Place a chess piece on a square and record the board state. • Move the piece to a new square. • Record the updated board state via serial debugging. • Confirm that the subsystem reports a move event.
• The subsystem must detect capture events.	• Place two chess pieces on adjacent squares. • Move one piece onto the square occupied by the other piece. • Record the board state via serial debugging. • Confirm that the subsystem reports a capture event and updated occupancy state.
• The subsystem must recognize valid chess moves.	• Execute known valid moves (e.g., pawn forward movement, knight movement). • Record subsystem outputs. • Confirm that the subsystem accepts the move and updates the board state.
• The subsystem must reject illegal chess moves.	• Attempt invalid moves (e.g., pawn moving three squares, bishop moving through a piece). • Record subsystem outputs. • Confirm that the subsystem rejects the move and provides an illegal move indication.
• The subsystem must maintain a consistent board state following valid moves.	• Execute a sequence of valid moves. • Record board state after each move. • Confirm that board state updates reflect expected piece positions.
• The subsystem must transmit validated	• Execute a valid move.

move events to the Networking Subsystem.	• Monitor transmitted messages via serial debugging or network logging. • Confirm that a move event is generated and transmitted.
• The subsystem must provide feedback for illegal moves.	• Attempt an illegal move. • Observe User Interface behavior. • Confirm that an illegal move indication is displayed.

2.3.3 Networking and Synchronization Subsystem

This subsystem manages communication between remote chessboards through a centralized server hosted on AWS. It operates as a software module on the ESP32-S3-WROOM-1 microcontroller and utilizes the onboard 2.4 GHz WiFi radio to connect to the server over TCP/IP using HTTPS requests.

The subsystem receives validated move data from the Move Processing and Legality Subsystem and transmits that data to the AWS-hosted API. The API stores the authoritative game state and provides opponent move updates when requested. Incoming move data is forwarded to the Move Processing Subsystem for board updates and to the User Interface Subsystem for display guidance.

Move ordering is enforced at the server level. The server maintains the authoritative sequence of moves for each board number and game style combination. When a board submits a move, the server updates the stored state. When the opponent board polls the API, it retrieves the most recent move state.

Table 3: Networking and Synchronization Subsystem – Requirements & Verification

Requirements	**Verification**
• The Networking and Synchronization Subsystem shall establish a WiFi connection and successfully communicate with the AWS-hosted API.	• Power on the board and connect it to a configured WiFi network. • Monitor serial debugging output to confirm successful IP address assignment. • Execute a test API request to the AWS server. • Confirm receipt of a valid HTTP response code.
• The subsystem shall transmit validated move state, board number, and game style to the AWS API in correctly formatted JSON.	• Execute a valid move on the board. • Capture the outgoing API request using serial debugging output or AWS API logs. • Confirm that the JSON payload contains the correct move state, board number, and game style fields. • Confirm that the server stores the updated move state.

• The subsystem shall retrieve updated move state data from the AWS API.	• Inject a move using a second board or the AWS web debugging interface. • Trigger an API poll request from the local board. • Monitor serial debugging output. • Confirm that the received move state matches the value stored on the server.
• The subsystem shall complete board-to-server communication in under 2.5 seconds.	• Execute a valid move on the board. • Record the timestamp at move validation. • Record the timestamp at server acknowledgment (via serial output or AWS logs). • Compute the elapsed time. • Confirm that the elapsed time is less than 2.5 seconds.

2.3.4 User Interface Subsystem

The User Interface Subsystem utilizes a 3.5-inch IPS capacitive touch display with a 480×320 pixel resolution. The display connects directly to the ESP32-S3-WROOM-1 using SPI (Serial Peripheral Interface) for display data.

The IPS panel provides wide viewing angles and consistent color reproduction, which improves visibility during gameplay from multiple positions around the board. The 480×320 resolution provides sufficient pixel density to clearly render chess notation, confirmation prompts, network status indicators, and error messages without excessive graphical complexity.

The capacitive touch interface allows users to confirm moves and interact with session configuration menus. Touch detection is handled via an integrated touch controller that communicates with the ESP32 over a dedicated I²C (Inter-Integrated Circuit) bus. This enables accurate position detection while minimizing processor overhead.

Table 4: User Interface Subsystem – Requirements & Verification

Requirements	**Verification**
• The display shall initialize successfully and render graphical output upon system startup.	• Power on the board. • Monitor serial debugging output to confirm successful display initialization. • Observe the screen. • Confirm that a startup screen or system status message is rendered correctly without graphical corruption.

• The display shall render text and graphical elements at the full 480×320 resolution.	• Load a test screen containing text, icons, and geometric shapes spanning the entire display area. • Visually inspect the display. • Confirm that the rendered output fills the full screen and that no clipping or distortion occurs.
• The display shall register touch input used for move confirmation.	• Execute a valid move that generates a confirmation prompt. • Touch the confirmation button on the display. • Monitor serial debugging output. • Confirm that the touch event is detected and that confirmation is registered by the system.
• The display shall update visual feedback within acceptable system latency.	• Trigger a move confirmation prompt. • Record the timestamp at move validation. • Record the timestamp at which the prompt appears on screen (via debug logging). • Confirm that display update latency is less than 200 milliseconds.
• The display shall visibly indicate network connection status.	• Establish a normal WiFi connection. • Confirm that the display shows a connected status indicator. • Disable WiFi connectivity. • Confirm that the display updates to show a disconnected status indicator.

2.3.5 LED Subsystem

The LED Subsystem provides immediate visual feedback by illuminating board squares associated with detected or received moves. LEDs are embedded beneath each square of the chessboard and are individually addressable by the ESP32-S3-WROOM-1 microcontroller.

When a valid local move is detected, the Move Processing and Legality Subsystem signals the LED Subsystem to illuminate the source and destination squares. When a remote move is received through the Networking and Synchronization Subsystem, the corresponding squares are illuminated to guide the user in replicating the opponent's move.

Each square's LED is controlled independently to ensure precise indication of board positions. The system must guarantee 100% addressing accuracy so that only the intended LEDs are activated. Incorrect LED activation would result in move ambiguity and gameplay errors.

Table 5: LED Subsystem – Requirements & Verification

Requirements	Verification
• The LED Subsystem shall allow the MCU to individually address each board square.	• Implement a diagnostic test mode that sequentially activates each LED. • Activate LEDs one at a time in numerical square order. • Visually confirm that only the intended square illuminates during each step.
• The LED Subsystem shall achieve 100% square addressing accuracy.	• Execute the sequential LED diagnostic test across all 64 squares. • Record square index and observed illuminated square. • Confirm that each command activates the correct physical square with no unintended illumination.
• The LED Subsystem shall illuminate the correct source and destination squares for a valid local move.	• Execute a valid move on the board. • Observe LED behavior. • Confirm that exactly two LEDs illuminate corresponding to the source and destination squares.
• The LED Subsystem shall illuminate the correct squares for a remote move received via the Networking Subsystem.	• Inject a remote move using a second board or AWS debugging interface. • Observe LED behavior on the receiving board. • Confirm that the illuminated squares match the transmitted move.
• The LED Subsystem shall update LED state within acceptable system latency.	• Execute a move or inject a remote move. • Record the timestamp of move validation or receipt. • Record the timestamp when LEDs illuminate. • Confirm that LED activation occurs within 200 milliseconds.

2.3.6 Power Subsystem

The Power Subsystem regulates wall power to stable voltage rails required by the system hardware. The subsystem receives 5V input through a USB-C connector (USB4215-03-A) and distributes power across two primary rails: a regulated 3.3V rail for low-voltage peripherals and a 5V rail for higher-current components including the LED array and the microcontroller input.

The USB4215-03-A connector provides a fixed 5V supply from a wall adapter. Proper configuration channel (CC) resistors are implemented to request default 5V operation. The 5V input rail directly powers components requiring 5V operation, including the LED subsystem and the 5V input stage of the ESP32-S3 module (via onboard regulation).

To generate the 3.3V rail, the subsystem uses the TLV62569PDDCR synchronous buck regulator. This switching regulator converts the 5V input to a regulated 3.3V output with high efficiency. The device supports input voltages from 2.5V to 5.5V and can supply up to 2A of output current, providing significant margin above the required 500 mA system load.

The 3.3 V rail directly impacts magnetic measurement accuracy because both the Hall sensors and ADC operate ratiometrically from this supply. Shared supply operation cancels proportional voltage variation, but excessive ripple or deviation beyond ±0.1 V could degrade resolution. Maintaining a stable, low-ripple 3.3 V rail preserves the calculated 0.11 mT resolution and ensures compliance with the ±1 mT detection requirement.

Table 6: Power Subsystem – Requirements & Verification

Requirements	**Verification**
• The Power Subsystem shall accept 5V input through the USB4215-03-A USB-C connector.	• Connect a 5V USB-C wall adapter to the board. • Measure voltage at the input to the TLV62569 regulator. • Confirm voltage is within 4.75V–5.25V.
• The Power Subsystem shall provide a stable 5V rail to support the LED subsystem and MCU input.	• Power on the system with LEDs active at maximum brightness. • Measure the 5V rail using a multimeter or oscilloscope. • Confirm that voltage remains within 4.75V–5.25V without significant droop.
• The Power Subsystem shall regulate the 3.3V rail to 3.3V ±0.1V.	• Power the system under nominal load. • Measure the 3.3V output of the TLV62569 using a calibrated multimeter. • Confirm voltage remains between 3.2V and 3.4V.
• The Power Subsystem shall operate within safe thermal limits at full load.	• Operate the board with LEDs, WiFi, and display active for at least 15 minutes. • Measure regulator temperature using a thermal probe or IR thermometer. • Confirm temperature remains within limits specified in the TLV62569 datasheet.

2.4 Software Design

The software for the system is implemented on the ESP32-S3-WROOM-1 microcontroller and follows a modular, state-driven architecture. System behavior is governed by a centralized finite state machine (FSM) that coordinates piece detection, move validation, user interaction, LED control, and network communication.

The system transmits game state to the AWS API using FEN (Forsyth–Edwards Notation). Instead of sending incremental move commands, the full board configuration is serialized into a FEN string and transmitted along with the board number and game style. This approach ensures complete state synchronization and simplifies server-side logic.

2.4.1 Software Architecture

The software is divided into the following functional modules:

- Piece Detection Interface
- Move Processing and Legality Engine
- Networking and Synchronization Module
- User Interface Controller
- LED Control Module
- Game State Manager (Finite State Machine Controller)

The Game State Manager controls system flow and ensures that only one major operation (validation, transmission, or synchronization) occurs at a time.

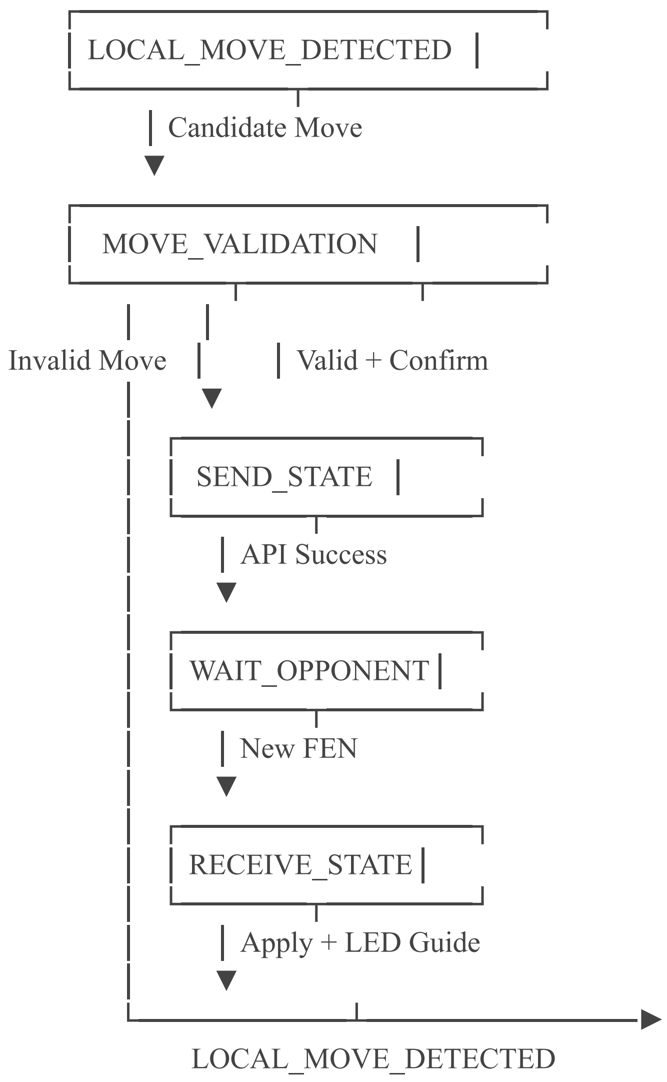
2.4.2 Internal Board Representation

The internal board state is maintained as an 8×8 array structure storing piece type and color. After every validated move, the board array is serialized into a FEN string.
Example starting position: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR
The FEN string is used as the authoritative state transmitted to and received from the AWS API.

2.4.3 Finite State Machine Design

The system operates using the following primary states:

```
┌─────────────────────────┐
│ WAIT_FOR_GAME_START │
└───────────┬─────────────┘
      │ Start Game
      ▼
┌─────────────────────────┐
│ GAME_INITIALIZATION │
└───────────┬─────────────┘
      │ Init Complete
      ▼
```



WAIT_FOR_GAME_START

The system remains idle while the user configures board number and game style. No move detection occurs.

Transition:
User selects “Start Game” → GAME_INITIALIZATION.

GAME_INITIALIZATION

The internal board array is initialized to the standard starting position. The corresponding FEN string is generated.

Transition:
If it is the local player’s turn → LOCAL_MOVE_DETECTED.

If waiting for opponent → WAIT_FOR_OPPONENT.

LOCAL_MOVE_DETECTED

The Piece Detection Subsystem monitors square occupancy. When a valid occupancy transition pattern is detected, a candidate move is generated.

Transition:
Candidate move detected → MOVE_VALIDATION

MOVE_VALIDATION

The Move Processing and Legality Engine verifies:

- Correct player turn
- Piece movement constraints
- Collision rules
- Capture legality
- Check conditions

If the move is invalid:

- The User Interface displays an "Illegal Move" message.
- The internal board state remains unchanged.
- No FEN update occurs.
- The system returns to LOCAL_MOVE_DETECTED.

If the move is valid:

- The internal board array is updated.
- A new FEN string is generated.
- The User Interface prompts for move confirmation.

Transition:
User confirms move → SEND_STATE.

SEND_STATE

The updated FEN string, board number, and game style are formatted into a JSON payload and transmitted to the AWS API.
Transition:
Successful API response → WAIT_FOR_OPPONENT.

WAIT_FOR_OPPONENT

The system polls the AWS API at fixed intervals to retrieve the most recent FEN state. During this time, the display indicates that the board is waiting for the opponent.

Transition:

If received FEN differs from local FEN → RECEIVE_STATE.

RECEIVE_STATE

The received FEN string is parsed and converted into an 8×8 board array. The system computes the difference between the previous and new state to determine the source and destination squares.

The LED Subsystem illuminates the relevant squares to guide the player in replicating the move. Once piece detection confirms that the physical board matches the received FEN state, the LEDs deactivate.

Transition:
Board synchronized → LOCAL_MOVE_DETECTED.

GAME_END

The system transitions to GAME_END upon detection of checkmate, stalemate, or manual termination. The display presents the game result and disables further move processing.

Transition:
User selects “New Game” → WAIT_FOR_GAME_START.

2.4.4 Design Rationale

The software is implemented as a deterministic finite state machine to ensure controlled and predictable system behavior. Each state performs a single logical function, preventing conflicting operations such as validation and transmission from occurring simultaneously. Clearly defined transitions improve reliability and simplify testing and debugging.

FEN is used as the communication format to transmit complete board state rather than incremental moves. This ensures consistent synchronization between boards and simplifies server implementation. Because FEN strings are compact and human-readable, they also support efficient transmission and straightforward debugging.

The modular architecture separates detection, validation, networking, UI, and LED control into independent components. This structure supports future expansion without requiring major redesign, allowing additional features to be implemented while preserving core system stability.

3. Cost and Schedule

3.1 Cost Analysis Overview

This estimate assumes two complete boards (one per player) plus a small relay server for remote matchmaking. It also includes a few extra PCBs for rework and debug.

3.2 Labor (per partner)

A reasonable starting salary for a UIUC Computer Engineering bachelor's graduate is about $77,653/year [2].

That is about **$37.33/hour** ($77,653 over 40 hours per week for 52 weeks).

Formula required by the rubric:

$$(\$/\text{hour}) \times 2.5 \times \text{hours} = \text{total}$$

Assume **60 hours** per partner for design, build, test, integration, and demo prep.
Hourly rate: $37.33/hr
Per partner labor: 37.33 × 2.5 × 60 = $5,599.50→ **$5,600**
3 partners total labor: 3 × 5,600 = $16,800
Total labor (all partners): $16,800

3.3 Non-standard Parts, Services, and Equipment

A. Non-standard parts
   a. Hall sensors used to detect and identify individual pieces (many sensors).
   b. Per-square RGB LEDs for board feedback (many LEDs).
   c. WiFi MCU module and multi-channel ADC.

B. Shop services (as needed)
   a. Laser cutting for an acrylic top plate or enclosure panels.
   b. Simple CNC routing for a wood base.

C. Lab equipment (assumed available in the lab)
   a. Soldering station
   b. Hot air
   c. Bench supply
   d. Multimeter
   e. Oscilloscope

3.4 Parts Cost Table (estimate)

Item	**Manufacturer**	**Part #**	**Qty**	**Unit Cost**	**Estimate Total**
WiFi MCU module	Espressif	ESP32-S3-WROOM-1	2	$5.38	$10.76
8-ch ADC	Texas Instruments	ADS7128IRTER	2	$6.61	$13.22
Hall sensors (piece detection)	Texas Instruments	DRV5055 (variant)	128	$0.85	$108.80
Addressable RGB LEDs	WS2812B class	WS2812B	128	$0.037	$4.76
OLED display module	SSD1306 class	SSD1306 0.96"	2	$2.10	$4.20
USB-C receptacle	HCTL	HC-TYPE-C-16P-01A	2	$0.0798	$0.16
Buck regulator	Texas Instruments	TPS62160DGKT	2	$1.98	$3.96
Passives + connectors bundle	Various	Various	1	$15.00	$15.00
Custom PCBs (prototype runs)	JLCPCB	2-layer PCB	2 runs	$25.00	$50.00
SMT stencil	JLCPCB	≤ 100 × 100 stencil	1	$3.00	$3.00
Enclosure materials	Various	Acrylic/wood + fasteners	2	$35.00	$70.00
Magnets pack	Various	Neodymium	2	$12.00	$24.00
Chess piece sets	Various	Standard pieces	2	$20.00	$40.00
Wiring + misc supplies	Various	Various	1	$15.00	$15.00
Relay server hosting	TBD	Cheaper plan	3 months	$5.00/mo	$15.00

Table 7: Parts and Costs

[3] [4] [5] [6]

3.5 Grand Total

Labor: $16,800
Parts/services/hosting: $377.86

Estimated Grand Total: $17,177.86

3.6 Schedule

A. Week 1
    a. Freeze requirements and interfaces
    b. Finalize sensing method and per-square electronics count
B. Week 2
    a. Schematic complete
    b. Firmware skeleton running on the MCU
C. Week 3
    a. PCB layout complete
    b. Order PCBs and stencil
D. Week 4
    a. Bring-up plan written
    b. Server relay and protocol draft complete
E. Week 5
    a. Assemble first PCB
    b. Power, USB-C, regulation, and basic GPIO tests
F. Week 6
    a. Sensor readout working
    b. LED feedback working
G. Week 7
    a. Piece identification working on a subset of squares
    b. Calibration procedure drafted
H. Week 8
    a. Assemble second board
    b. Board-to-board sync working over WiFi
I. Week 9
    a. Reconnect logic, state resync, and error handling
    b. Long run testing
J. Week 10
    a. Full demo script
    b. Packaging, cable management, and final safety checks

4. Discussion of Societal Impact, Engineering Standards, Ethics, and Safety Considerations

4.1 Positive Contribution to Public Health, Safety, and Welfare

The Networked Physical Chessboard for Remote Play contributes positively to public health, safety, and welfare across economic, environmental, social, cultural, and global dimensions.

A. Social and Cultural Impact

Chess is known to improve memory, focus, and strategic thinking. Many players prefer the feeling of real pieces instead of tapping on a screen. This project supports that experience while still allowing remote play.

The system benefits families and friends who live far apart. It also supports students in remote learning environments and older individuals who may not travel easily. Chess has cultural value in many countries. Making this technology more affordable helps more people participate in that tradition.

B. Public Health Considerations

Digital chess platforms require long periods of screen time. Screen overuse can lead to eye strain and fatigue. By keeping the game physical, this system reduces screen exposure. Players still communicate remotely, but the board interaction remains hands on. Physical piece movement also supports fine motor engagement. This creates a more natural experience compared to using a touchscreen.

C. Economic Impact

Commercial smart chessboards are often expensive. This project shows that a lower cost system can be built using embedded sensors and standard components. Lower cost designs increase accessibility. They also encourage competition and innovation. Affordable products help schools and families gain access to educational tools that might otherwise be out of reach.

D. Environmental Considerations

The system operates at low voltage. It uses efficient switching regulation to reduce wasted power. The USB C interface also avoids proprietary adapters. The design does not rely on disposable components or chemical batteries. Power consumption is small compared to most consumer electronics. The environmental impact is thus very minimal.

E. Global Impact

This project demonstrates hybrid physical and digital interaction. The same concepts could apply to other remote learning tools or devices. It supports global communication while preserving real world interaction.

4.2 Engineering Standards

A. Electrical and Communication Standards [1]
   a. IEEE 802.11 governs wireless local area network communication. The WiFi module used in this system must comply with IEEE 802.11 physical and MAC layer requirements.
   b. IEEE 802.3 concepts for data integrity and structured communication inform reliable network transmission practices.
   c. IEEE 802.15.4 is referenced for general low power wireless system design principles, even though WiFi is used instead of WPAN.
   d. IEEE C95.1 provides safety levels for human exposure to radio frequency electromagnetic fields. The low power WiFi module operates well below these exposure limits.
   e. IEEE 11073 principles for device communication safety inform structured data exchange practices, though this is not a medical device.

Although this project is a prototype, awareness of these standards reflects professional engineering practice.

B. Software and Cybersecurity Standards [1]
   a. IEEE 12207 outlines software life cycle processes. Basic structured development practices are followed.
   b. IEEE 29119 provides guidance on software testing standards. Functional testing and validation are performed on move detection and synchronization logic.
   c. IEEE 1686 addresses cybersecurity capabilities in embedded devices. While simplified for this project, secure communication and controlled access are considered.
   d. ACM Code of Ethics and Professional Conduct governs responsible computing practices.

The project does not collect personal data. Only move information is transmitted between boards.

4.3 Ethics and Professional Responsibility

The IEEE and ACM Codes of Ethics emphasize honesty, safety, and avoidance of harm. These principles apply directly to this project.

A. Cheating Concerns

The board could be used alongside chess engines. This creates the possibility of unfair play in competitive settings. The system does not claim to prevent cheating. It is intended for casual and recreational use. Marketing and documentation will clearly state the system's limitations. We will not imply features that are not technically guaranteed.

This aligns with the IEEE Code of Ethics requirement to be honest and realistic about system performance

B. Transparency

Unlike simpler magnetic boards, this system identifies individual pieces and tracks their positions in software. This improves accuracy and reduces ambiguity during gameplay. However, users must understand how the detection works and what its limits are.

Sensor errors, magnet misalignment, or unexpected piece placement could cause incorrect board states. The system includes legality checks and synchronization routines to reduce these issues. Clear documentation explains how the board interprets moves and how to correct mismatches if they occur.

C. Data Privacy

The system transmits move data over WiFi. It does not store personal information. Communication will use secure methods when possible. No hidden data collection will occur.

4.4 Electrical and Mechanical Safety Concerns

The system operates at 5 volts input and 3.3 volts internally. This is considered low voltage. Some safety concerns do still exist.

A. Electrical Risks
   a. Short circuits from solder errors
   b. Overcurrent conditions
   c. Heat from voltage regulation
   d. Electrostatic discharge damage

B. Mechanical Risks
   a. Small magnets inside pieces
   b. Sharp solder joints
   c. Structural weakness if enclosure fails

All the risks of this project are low but must still be addressed to ensure safe usage of this product.

4.5 Safety Mitigation Procedures

Many precautions are taken to ensure the safety of the user during games.

A. Electrical Protection
   a. Use of USB C 5 volt input only
   b. Switching regulator with thermal and current protection

    c. Proper PCB grounding
    d. Decoupling capacitors near integrated circuits
    e. Insulated enclosure covering electronics

A current limited bench supply will be used during early testing. Output voltages will be verified before connecting sensitive components.

B. ESD Protection
    a. Anti static handling during assembly
    b. Grounded workstations
    c. Careful soldering procedures

These are standard industry practices used during the design and manufacturing of electrical products.

C. Mechanical Protection
    a. Magnets secured permanently inside pieces
    b. Smooth enclosure edges
    c. Reinforced PCB mounting
    d. No exposed conductive traces

This will also ensure that small pieces do not fall off the board or pieces, thus reducing risk of choking hazards as well.

D. Software Safety
    a. Move legality checking before transmission
    b. Network timeout detection
    c. Board state validation during reconnection

This will ensure the software side of the product remains safeguarded. This will prevent cheating or other types of exploitation of the system.

4.6 Justification of Safety

This design protects both users and developers. It operates at low voltage. It contains no high power components. It uses no lithium batteries. There are no moving actuators. There are no exposed high temperature surfaces.

The system does not involve high voltage, chemicals, aerial devices, or human body interfacing. Risk level is low under course safety guidelines.

Proper testing, enclosure design, and transparent documentation ensure safe operation. Thus, the project meets professional ethical standards and prioritizes user safety.

5. References

[1] DigiKey | Electronic Parts and Components – Fast Shipping, Global Support, www.digikey.com/. Accessed 27 Feb. 2026.

[2] ESP32-S3-WROOM-1-N16 Price & Stock | Digipart, www.digipart.com/part/ESP32-S3-WROOM-1-N16. Accessed 27 Feb. 2026.

[3] Grainger Engineering Office of Marketing and Communications. "Become an Electrical or Computer Engineer." *Illinois*, ece.illinois.edu/admissions/ece-majors. Accessed 27 Feb. 2026.

[4] "IEEE Standards Association." *Wikipedia*, Wikimedia Foundation, 9 Feb. 2026, en.wikipedia.org/wiki/IEEE_Standards_Association.

[5] "PCB Manufacturing & Assembly." *JLCPCB*, jlcpcb.com/. Accessed 27 Feb. 2026.

[6] "Usage & Billing." *Heroku Dev Center*, devcenter.heroku.com/articles/usage-and-billing. Accessed 27 Feb. 2026.