

Smart Chessboard Assistant

Authors: Justin Doi, Sabah Sadik, Kevin Park

Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

Abstract—A system capable of enabling visually impaired chess players to effectively play and learn chess with several previously online-only features.

Index Terms—Computer Vision (CV), Hall-Effect Sensor, Printed Circuit Board (PCB)

1 INTRODUCTION

In recent years, online chess has revolutionized the way many people learn, play, and analyze the game of chess. Some of its features, such as AI assisted game reviews, has given millions of users the ability to study chess at a level that used to require personal coaches. However playing chess online strips away several features of the game that are key to physical chess. For many, such as most local chess groups or the majority of the competitive scene, the act of playing the game with physical pieces is still the status quo. The motivation behind the Smart Chessboard Assistant is to combine the tactile accessibility of physical chess with the game tracking and instructional support found in digital chess platforms. The intended use case is an in-person game played on a physical board, either between two visually impaired players or between a visually impaired player and sighted player. Consider a visually impaired beginner sitting across another player at a tactile chessboard. They can feel the squares, make out the pieces, follow the position of the pieces, but they cannot tell whether a move was legal or if the game state is consistent with how it should be. In a traditional game, this information would come from the other player or an outside source. We want to reduce this external dependence, so that the physical chessboard can be a more independent learning and playing environment for visually impaired and beginner players.

An accessible chessboard can effectively show a visually impaired player the spatial layout of the game through tactile features such as braille coordinates and distinguishable squares. These features let the player physically explore the board and where the pieces are. The gap with these products is that the physical board cannot keep track of the game that is being played. It cannot determine whether a move is legal, confirm if the physical arrangement matches the game state, record move sequences, or provide guidance if a player makes a mistake. Digital chess platforms solve many of these problems since the software always has access to the full game state. It can reject illegal moves, record game history, identify checks or checkmates, and even provide the instructional guidance that would benefit a beginner chess player. However, these useful capabilities

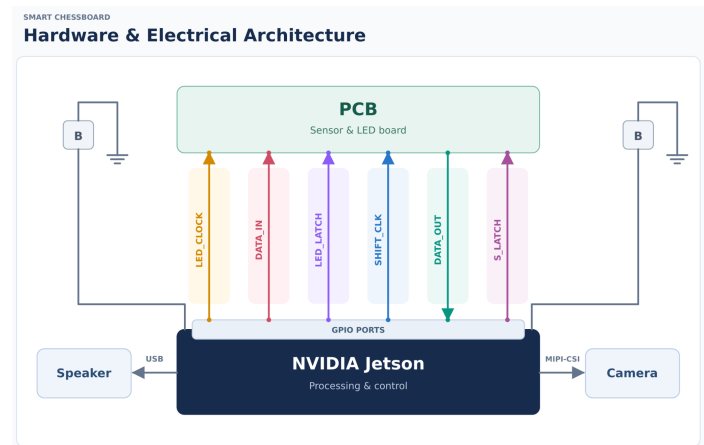
are tied to the digital interface and for a visually impaired player who benefits from physically locating squares and pieces through touch, replacing a board with a screen based interface changes the primary way the game is experienced. The resulting problem is not the lack of accessible chessboards or the software. Both already exist but there are no reliable connections between the two. Our group believes that there is a missing layer between the physical actions from the player and the safeguarded features of correct chess play built into online versions of the game. A physical board still remains important because it allows the visually impaired player to figure out the physical layout of the chessboard through touch. Features like Braille coordinates, tactile differences between squares, and physically distinguishable pieces let the user figure out where the pieces are positioned without having to rely on someone or a visual display. The tactile board serves as the players' primary way of interacting with the game while the digital assistance is there to support the experience rather than replacing it. A visually impaired player should be able to independently locate pieces, make moves, and understand the board position physically, and receive additional information only when needed. This preserves the natural over-the-board experience while providing the helpful features found in online chess. Keeping the game on a physical board also supports the intended use case of the project for in-person play between two players. Whether both players are visually impaired or only one is, the game is to stay centered around the shared physical board rather than online. The goal is to make the physical environment more accessible and informative, not move the experience away from it.

2 USE-CASE REQUIREMENTS

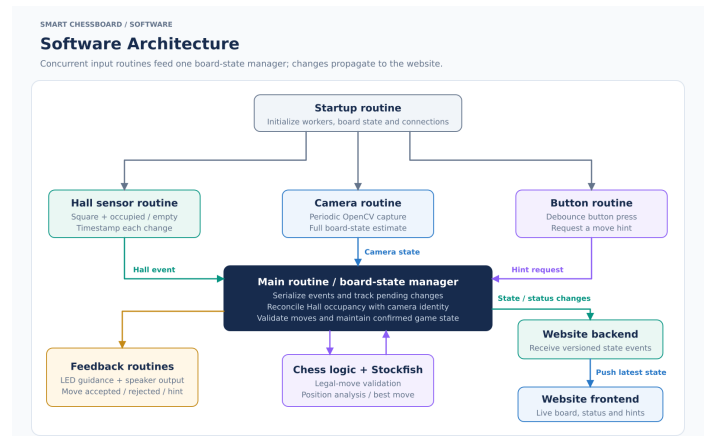
For the reliable move detection aspect, the user should be able to move a piece without having to confirm it through a separate interface. The chessboard sensing system needs to correctly detect 100% of the piece placements and removals across all 64 squares. A detection is considered correct when the PCB accurately identifies the occupancy of every square involved with each completed move. A mismatch can happen if a sensor event is missed, a piece is accidentally moved, or the software state becomes corrupted. If this does occur, then the system should correctly recover the physical board state in at least 9 out of 10 recovery attempts within 10 seconds. Successful recovery means that the system reconstructs the actual arrangement of pieces on the board and updates the internal game state. This is a key part of our requirements because one

incorrect board state can lead to every move after to be interpreted incorrectly. For the response latency, the user should be receiving feedback fast enough so that it does not slow down the pace of the actual game. After a player completes a move, the system should be able to detect the move, validate it, update the digital board state, and give any necessary feedback within 3 seconds. This 3 seconds encompasses all of the time required for physical sensing, software processing, move validation, and user feedback. We were set on 3 seconds because it better follows the pace of a recreational / instructional play. Meeting this requirement makes sure that the assistance is responsive without forcing the player to have to wait after every move. For move validation, the system should correctly identify whether a detected move is legal based on the current state of the game. For correctly detected moves and a correct starting board state, the system should be able to classify 100% of all tested moves to be legal or illegal correctly. If a move is legal, then the digital game state should update and have the move recorded. However, if a move is illegal, then the digital state should not be changed and the player should receive feedback stating that the move needs to be corrected. This is an important requirement since incorrect move validation could lead to the software state to diverge from a valid chess game even if the physical sensing system is working correctly. For accessibility and feedback, the visually impaired player(s) should be able to understand important game events / information without having to rely on an external interface. The system should be providing correct audio feedback for at least 95% of tested feedback events, including illegal moves, check conditions, requested hints, and board-state recovery notifications. While the physical board will still be the primary interactive feature, audio will help provide additional information that cannot be determined through touch alone. This lets the player continue using the tactile chessboard without having to rely on another person or external device for game information.

3 SYSTEM ARCHITECTURE



(a)



(b)

Figure 1: System description. (a) system hardware block diagram. (b) software block diagram. Credit GPT 6 Astra

The hardware architecture showing in figure 1 (a) primarily shows the interconnects between the PCB and Jetson, along with external sensing outputs. More detail of each component in the flow are below.

The software architecture shown in figure 1 (b) will have different threads of execution running the main routine and all other subroutines. The Hall Sensor routine will receive the hall sensor output and reconstruct a matrix representation of which chess squares are occupied or not, similarly the button routine will forward a hint request to the main loop upon a button press. The camera routine will receive and forward JSON packets from the separate process running OpenCV. Output threads will send information about moves to both speaker and LEDs physical outputs, and the current log of moves to the web backend. The main rou-

tine will send and receive data to and from other routines through synchronized channels. For analysis features, it will also make Stockfish API calls after each move, it will upload these measures of which player has the advantage, and be able to output the next best move on a hint request.

3.1 Sensor Layout

The sensor layout consists of an 8×8 matrix of 64 DRV5032FADBZR digital Hall sensors. Each sensor is positioned directly beneath the middle of each chessboard square. Each chess piece will have a magnet embedded on the bottom of the piece. Then each sensor can detect the magnetic field from that magnet, producing an exact voltage that indicates whether the square is occupied (0V for not occupied, and 3.3V for occupied). The eight sensors in each row connect to a 74HC165 parallel-in/serial-out shift register. This resulted in eight total shift registers (one for each row) across the board. These registers are daisy-chained, which allows the NVIDIA Jetson Nano to read all 64 sensor states through a single serial data line, an output from the shift register in the last row. The Nano reads the 3.3V as a 1, and 0V as a 0). Additionally there are 72 capacitors, each placed in parallel with a sensor or register, this is to filter out any high-frequency electrical noise and stabilize the voltage. This arrangement of using shift registers allows the number of outputs from the PCB to GPIO to be just 1.

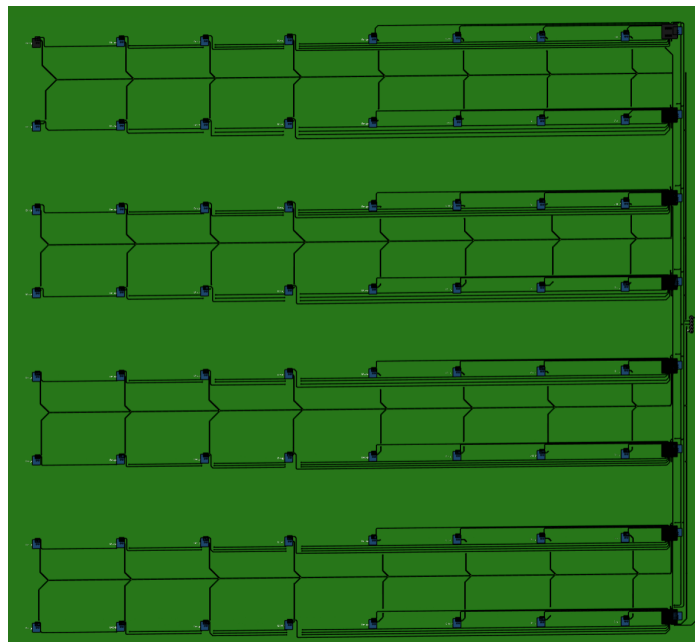


Figure 2: Sensor Layout

3.2 Compute Layer

The NVIDIA Jetson Orin-Nano will be our primary compute device, it houses a 6-core Arm® Cortex®-A78AE v8.2 64-bit CPU 1.5MB L2 + 4MB L3. The processor will enable sufficient parallel execution. And a total of 8 GB of main memory will allow us to run our entire software stack and the underlying lightweight Linux OS with no performance issues from relying on swap. For AI compute it houses an NVIDIA Ampere architecture GPU with 1024 CUDA cores and 32 tensor cores With AI performance of 67 INT8 TOPS, we expect responsive computing on our fine tuned OpenCV model.

The chess software layer is responsible for maintaining the digital representation of the game and interpreting the physical moves on the board. Inputs come in from the Hall sensors and computer vision routines. The players' moves are inferred and validated against legal chess states using the python-chess library before any update is committed to each game state. Valid moves update the digital board state and invalid ones are rejected and passed to the user-feedback subsystem. The module also compares sensor-derived and CV derived states to check for inconsistencies and go into recovery mode if needed. For things like instructional features, the validated board state is passed to a locally stored Stockfish engine to generate responses to hint requests. Valid moves and state changes will be published to a web back end for move tracking.

The user-feedback layer converts validated game events into accessible physical outputs. Audio will be outputted through a board mounted speaker for nonvisual notifications such as illegal moves, recovery warnings, and Stockfish-generated hint responses. LEDs will provide a secondary visual channel for indicating relevant squares for suggested source and destination locations. Feedback commands will only go through if the move and board state validation are confirmed to be correct.

3.3 Computer Vision Layer

The computer vision aspect of the project will use OpenCV directly on the Jetson, leveraging its CUDA cores for increased performance. OpenCV runs on c++; as mentioned earlier we plan on writing our primary software loop in golang. The output of the CV model will be an 8x8 matrix representation of the live board state each entry in the matrix will be a member of the set White Piece, Black Piece X Pawn, Bishop, Knight, Rook, Queen, King, Empty, Covered with a total of 14 possible outputs per square. To get maximum performance we need to leverage CUDA on the Jetson which has been done before on earlier versions of OpenCV; although no reliable solutions exist so far on the latest release. Leveraging the full compute capabilities of the Jetson would enable our system for real-time camera-based board monitoring and better responsiveness to blocked pieces.

3.4 Physical Board Design

The physical chessboard will be custom built by us and include a laser cut wooden casing for the PCB and Jetson underneath the board. There will be speakers along the walls of the encasing that will announce important information about the game to the players. The actual chessboard will be on top of all the hardware. It will be built from colored acrylic with different heights between light and dark squares; underneath each square, we will have a hall effect sensor and a strip of NeoPixel LEDs. The dimensions of the board will be 22x22 inches with 2.5 inch chess squares. Our initial testing of digital hall effect sensors show high detection accuracy even when separated by acrylic. For ease of moving pieces with limited eyesight, in addition to tactile feeling between light and dark squares, a small peghole will be drilled at the center of each square and our chess pieces will have a peg at the bottom that will easily fit into the exact center of a square. This will both ensure the pieces can be moved easily with only tactile response and automatically align the magnets at the bottom of the pegs to the hall sensors, so no inaccurate readings will occur.

4 DESIGN REQUIREMENTS

Here we layout the technical requirements for the design of our system.

4.1 Sensor Requirements

As specified in our user requirements, our users (the visually impaired) are expected to use their hands to navigate the board and hence possibly block the camera. Therefore an overall detection rate of 100% among all 64 squares is essential to notify the software system that a move has been made. To accurately detect each square individually, the sensors must be placed one under each square resulting in the design requirement of 64 total sensors, embedded as a 8x8 matrix. To further insure 100% accuracy, the design requirement specifies a magnetic activation threshold of no more than 3.0 mT. This allows the sensors to detect piece magnets within the magnetic field of 3.0mT, hence it should be able to pick up stronger magnets even through thickness in board material. Reliable detection must be accompanied by a timely response to meet the overall user latency requirement of less than three seconds. Accordingly, any sensor must detect occupancy changes within 100 ms, while the Jetson polls all 64 sensors at 50 Hz, corresponding to a 20 ms polling interval. Hence the max latency of a sensor reading becomes 120ms, which then will be processed using a debounce interval of 100-300 ms. This is to reject brief fluctuations before accepting a change in board occupancy, ensuring that any hovering isn't detected as a piece placed, hence supporting both accuracy and latency user requirements.

4.2 Camera Requirements

To meet our user requirement, at least 95% of processed images with a clear view of the board must achieve 100% detection accuracy. We define 100% detection accuracy as the computer vision (CV) pipeline correctly identifying every piece on the board and its location. To support this target, we have set a design requirement minimum camera resolution of 1920×1080 pixels to preserve the visual details needed to distinguish chess pieces. After the homography transformation in the CV pipeline, the board image is mapped to a 1080×1080 -pixel region, giving each square approximately 135×135 pixels. This is intended to provide sufficient detail for recognizing piece shapes under varying lighting conditions, including shadows cast by users' hands, hence making it the minimum requirement. To further support accurate detection, we require a minimum frame rate of 30 frames per second, which provides a new image approximately every 33 ms. Compared with 15 frames per second, which provides an image every 67 ms, 30fps offers twice as many opportunities to capture an unobstructed view after a hand moves away. Finally, to remain within the overall user requirement latency budget, we allocate 1500 ms to the CV pipeline. The LiveChess2FEN research demonstrated chess-position recognition in less than one second, our design allows additional processing time because the pipeline will share Jetson resources with other system routines.

4.3 Corrupt State Recovery Requirements

In the unlikely event that there is a mismatch between the software and physical board states, we will implement a computer vision based polling algorithm to recover to the true board state. From the hall sensors, the system will know which squares are occupied, then the system will constantly poll the camera to determine which types of pieces are on each of the occupied squares. This will be done repeatedly until they match and the game will continue, even though the move history will be inaccurate. Once completed the system will recover the correct state of the board with greater than 90% accuracy after a maximum polling period of 10 seconds.

4.4 Computer Vision Accuracy Requirement

The computer-vision subsystem will provide an independent estimate of the physical board state for verification and recovery when the Hall-sensor and software states do not agree. The CV model will classify each of the 64 squares into one of 14 possible states: 12 combinations of chess piece type and color, empty or covered. To provide sufficient enough detail for classification, the camera needs to capture at least 1920×1080 pixels. After homography transforms the board into an approximately 1080×1080 pixel top-down image, with each chess square having 135×135 pixels. This resolution is intended to preserve

enough piece-shape information for the model to determine between 14 output classes. The camera also needs to operate at 30 frames per second to produce a new frame every 33 ms so an unobstructed image can be taken after a player's hand leaves the board. For unobstructed images, the CV system needs to reconstruct the board for at least 95% of evaluated positions. A correct board reconstruction requires every identity and location of every piece to match the physical board. CV processing is allocated to a maximum of 1500 ms which leaves enough time within the overall 3-second response requirement for state comparison, chess validation, and user feedback.

4.5 Compute Subsystem Minimum Specifications

The compute subsystem needs to execute the Hall-sensor interface, CV processing, chess logic, networking, and feedback routines concurrently while maintaining the system level latency of 3 seconds or less. The NVIDIA Jetson Orin Nano will be our primary embedded computing platform because its multi-core CPU and GPU allow sensor processing, CV inference, game-state management, and communication tasks to execute in parallel. The total processing time that needs to be satisfied: $T_{total} = T_{sensing} + T_{inference} + T_{feedback} < 3s$. The CV pipeline is allocated up to 1500 ms, while the Hall-sensor subsystem polls the board every 20 ms and detects occupancy changes within 100 ms before debounce processing. The remaining computation time is leftover for the move inference, chess-rule validation, state updates, and feedback.

4.6 Website Requirements

The networking subsystem needs to maintain a synchronized representation of the validated game state on a local website. The website will be hosted on a Jetson and must receive state updates only after the main chess software validates a move. This prevents the incorrect displays of unconfirmed physical board states. For every validated move, the networking layer needs to transmit and record 100% of committed moves in chronological order. This web interface is more so for game analysis after a match completes and not a required interaction method for either player during the game.

5 DESIGN TRADE STUDIES

Here we discuss why and how we decided on several aspects of the design.

5.1 Hall Sensor Comparison

To meet our user requirement of at least 98% Hall sensor detection accuracy, the sensors must reliably identify occupied squares without false detections caused by neighboring

magnets. This led us to a design trade study between analog and digital Hall sensors. When operated at 5 V, the digital sensors provide distinct outputs of approximately 5 V for an unoccupied square and 0 V for an occupied square. However, if the magnetic field from a neighboring magnet exceeds the activation threshold, the sensor may also output 0 V, producing a false positive. Analog sensors instead produce a voltage that varies with magnetic field strength and polarity, potentially allowing software thresholds to distinguish a magnet directly above the sensor from the weaker field of a neighboring magnet. Though this provides greater flexibility, our analog design requires additional processing, two multiplexer stages, and an ADC that communicates with the Jetson over I2C. In comparison, digital outputs can be processed through daisy-chained shift registers connected to the Jetson's GPIO interface simplifying data processing pipeline. To evaluate this tradeoff between our user requirements of detection accuracy and design requirements of the latency of less than 200ms, we purchased and tested both sensor types. The testing procedures and results are presented in the testing section of this report.

5.2 Camera-only vs Sensor-Only vs Hybrid Detection

Another design trade study compared camera-only, sensor-only, and hybrid approaches. The camera-only and sensor-only approaches were estimated to require approximately 1.5 seconds and 1 second, respectively, for detection. However, our target users rely on touch to identify pieces and navigate the board, frequently obstructing the camera's view. A camera-only system could therefore miss board-state changes during these periods, making it difficult to meet the overall system design accuracy requirement of 99%. Continuously processing images would also increase computational demand on the Jetson, possibly adding more latency.

For the sensor-only approach, we considered analog sensors combined with different magnet strengths to distinguish the 12 piece categories: six piece types in each color. However, repeated analog sensor tests produced varying voltage readings for the same magnet. Hence, distinguishing all 12 categories would require sufficiently separated voltage ranges despite the variations in sensor response. This would introduce a lot of calibration in both software and hardware. Additionally the varying magnets were not within our project budget.

Due to these limitations it led us to the selection of a hybrid system combining digital Hall sensors with a camera. Digital sensors provide distinct occupied and unoccupied states but cannot identify piece types, while the camera supplies identification. When the sensors detect a stable state change, they trigger camera processing, which continues until a clear view is available. This allows the system to register state changes even while the board is obstructed and identify the piece(s) change once the user's hand moves away. Triggering computer vision processing through the sensor changes also reduces unnecessary com-

putation. Together, the two subsystems address their individual limitations. Although it adds onto the latency as two subsystems now have to compare to each other. Hence leading to a tradeoff between latency and accuracy.

5.3 Compute Platform

After researching microcontrollers, we decided that the NVIDIA Jetson Orin-Nano would be the best choice for our project, mainly for its CUDA capabilities. One of the main aspects of our system is the CV, and early on in the planning process we needed a compute platform with solid on-board AI compute. Its networking capabilities also enable us to host a website interface, which we realized could be a major aspect of our project to allow for additional analysis and recording of games.

5.4 Computer Vision Methods

While all members of our group have some level of machine learning background, none of us have much experience implementing computer vision models. Our design requirements also need as responsive and correct CV computing as possible. Thus, we intend on building off of OpenCV for our CV layer.

6 SYSTEM IMPLEMENTATION

There should be a subsection for each of the subsystems as shown below.

6.1 Physical Board and Sensors

The sensors are integrated onto a custom PCB consisting of 64 Hall sensors arranged in an 8×8 matrix and eight shift registers, one for each row. The PCB connects to the NVIDIA Jetson through five connections: ground, VCC, load, clock, and serial data. Ground and VCC provide the PCB's power connections, while the shared load and clock signals from the Jetson control all eight shift registers simultaneously. The registers are daisy-chained, allowing the serial output of the final register to transmit all 64 sensor states sequentially to the Jetson through a single GPIO input. The physical board will be manufactured from acrylic. We will additionally use TechSpark tools, including laser cutters, to customize the boards and create the raised black squares and placement holes in the center of each square. NeoPixels positioned beneath the acrylic board will be controlled by an Arduino connected to the NVIDIA Jetson through USB. The Jetson will send commands specifying which squares to illuminate and their colors, and the Arduino will process these commands and generate the signals needed to control the NeoPixels.

6.2 Jetson Compute System

Our Jetson computer has its own provided OS that we have already installed. It has a 40 pin header that will con-

nect to our PCB, the OS makes programming with GPIO relatively simple. It will be located inside the chessboard and connected to a wall outlet. The Jetson also has dedicated connectors for camera inputs which will integrate nicely with the CV program.

6.3 Computer Vision System

We will run our CV model on the Jetson in c++. It will receive photo inputs from the camera and output JSON packets that include a matrix representation of the game state. This will seamlessly integrate with the rest of the software which will maintain the software state of the game from the CV output.

6.4 Digital Board-State and Chess Logic

We plan on using python chess and API calls to a locally stored version of Stockfish to represent the board state in software. The main software routine will determine the move made by and forward it to python chess, which will then respond either with confirmation of the new state or if an illegal move was made. When the main software forwards a request for a hint, Stockfish will respond with the best possible move for the current player.

6.5 Website and Networking Integration

The website will be hosted on localhost from the Jetson. The Jetson will maintain a small database with a server that can maintain multiple clients. The Jetson has networking capabilities, which will be integrated with the rest of the Jetson. Upon a new game, it will begin uploading the sequence of moves to the website and give a live positional advantage similar to Chess.com.

7 TEST & VALIDATION

7.1 Digital/Analog Hall Sensor Comparison Test

The user requirement of at least 100% Hall sensor detection accuracy leads to a comparison between analog and digital sensors. Eight analog sensors and eight digital sensors were ordered and tested on a breadboard using a 5 V supply, the magnets intended for the final product, and a digital multimeter. An acrylic sheet was placed between the magnets and sensors to simulate the physical configuration of the chessboard. The analog sensor produced an output of 2.681 V with no magnet present. With the north pole facing the sensor, the output decreased to 1.466 V, and with the south pole it increased to 3.588 V. These readings represent changes of approximately 1.215 V and 0.907 V from the resting output. In comparison, the omnipolar digital sensor produced 5.00 V with no magnet present and 0 V when a magnet was positioned above it, providing a clear distinction between the two states regardless of magnet polarity. Out of 10 attempts with a single magnet the

digital sensor resulted in complete detection with 0V while the analog sensor resulted in variations of 3.5-3.7V.

To mitigate our concern of cross-talking and low detection ranges, we did distance testing on the x-plane. Distance testing showed that the analog sensor returned to approximately its resting output when the magnet was 0.5 cm away, whereas the digital sensor returned to its resting state at approximately 1 cm. These results indicate that the digital sensor provided a greater detection range under the tested conditions, but is also susceptible to cross talk. Leading us to additional testing with a neighboring magnet positioned 3 cm from the magnet directly above the sensor, which produced no observed crosstalk in either sensor type.

Based on these results, the digital sensor was selected for its greater detection range and clearly distinguishable output states. The analog sensor would require a voltage threshold to determine occupancy. Considering that within 10 tests the variation range of detect itself was not a single voltage, hence it would be difficult to find an accurate threshold. It would also require signals to pass through two multiplexer stages and an analog-to-digital converter before reaching the Jetson, introducing additional latency compared to the digital 20ms sweep.

7.2 PCB Hall Sensor Testing

Once the PCB arrives, we will test whether the sensor subsystem meets our 100% detection accuracy requirement. Usually a single chess game can range from 60-100 moves. So we will test each of the 64 sensors individually three times, producing 192 data points. We will also conduct 32 additional test cases per sensor with a magnet placed over a neighboring sensor to check for cross talk. These tests will be conducted separately from the full game play tests.

7.3 Computer Vision Accuracy Test

For the computer vision accuracy test, a set of predetermined board configurations will be created and the exact identity and location of every piece will be recorded before each image is captured. This will serve as our ground truth and for each image test, the CV model will output an 8x8 matrix in which each square is classified as one of 14 possible states: the twelve combinations of piece type, color, empty, or covered. The predicted matrix will be compared directly against the ground truth matrix. There will be two accuracy metrics that will be recorded. The first will be the square-level classification: $A_{\text{square}} = (\text{of correctly classified squares}) / (\text{total of tested board positions}) * 100\%$. The second will be a complete-board accuracy: $A_{\text{board}} = (\text{of completely correct board reconstructions}) / (\text{total number of tested board positions}) * 100\%$. The passing requirement that we set was that at least 95% of unobstructed test positions produce a completely correct 64-square board reconstruction.

7.4 Latency Test

The latency test will check to see that the integrated system responds fast enough to satisfy the use case requirement of less than 3 seconds from physical move to initiation of the corresponding system response. The test measures the end to end latency and time stamps will be recorded at the major stages of the processing pipeline: Hall-sensor detection, debounce completion, candidate-move inference, move validation, digital state update, and feedback initiation. The total processing time that needs to be satisfied: $T_{\text{total}} = T_{\text{sensing}} + T_{\text{move inference}} + T_{\text{feedback}} < 3s$. The CV recovery latency will be measured separately since it is only used when an inconsistency occurs. The CV design allocates up to 1500 ms for image processing so recovery trials need to ensure that CV processing plus state comparison and feedback remain within the recovery-time requirement.

7.5 Move Validation Test

The move validation test will check if the chess logic subsystem is correctly validating between legal and illegal moves and prevents invalid physical actions from corrupting the digital game state. A set of predetermined board positions and moves will be made and they will have the expected legality of each move known in advance. We will use these as the ground truth and some examples that we will include will be standard piece moves, captures, and other situations where legal move determination depends on the full game state rather than only the movement of a single piece. The primary accuracy metric will be: $A_{\text{validation}} = (\text{of correctly classified legal/illegal moves}) / (\text{total of tested candidate moves}) * 100\%$. The passing requirement is 100% correct legal vs. illegal classification for all correctly specified test cases.

7.6 State Recovery Test

One of the most critical issues our system can experience during use is a mismatch in the digital state of the game and the physical chessboard. Our failure model in this case will only attempt to recover the current physical state and the log of moves kept will be marked as partially corrupt and unreliable. Testing recovery will involve purposely setting an incorrect state in software. Once the system detects a mismatch from the CV outputs, the board neopixel and speaker outputs will announce that the system is trying to recover the current state and that the users should move their hands back to avoid blocking the camera. Successful metrics for testing are for 9/10 correct board state recovery within 10 seconds.

7.7 Full Gameplay Test

The final round of testing will be for full functionality of the system. We will have at least 3 test scenarios: (1) gameplay between two players without any kind of visual

impairment, (2) gameplay between two players with simulated partial visual impairment (not wearing prescription glasses), (3) gameplay between two blindfolded players, (4) gameplay between two real visually impaired chess players, although we can only test this scenario if we are given consent by some visually impaired. For each round of testing, successful metrics include 3/4 games where the system is able to successfully give hints when requested, and log all moves to the speakers, neopixels, and the website.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 4.

8.2 Team Member Responsibilities

Justin will focus on building the software for the Jetson and integrating external components with the Jetson. In addition to the system software, he will be responsible for implementing the web backend and internal database. He and Kevin will develop the computer vision subsystem using OpenCV and will obtain sufficient training data for detection of all pieces. He will also be responsible for using TechSpark's laser cutter for the board and hardware encasing. He will also be responsible for testing the software both during development and final gameplay trials on the full system.

Kevin will focus on the chess software and digital game-state management. He will be responsible for implementing the logic that interprets moves, board state changes, validation of moves, hint generation through Stockfish, and communication of validated game-state updates to the feedback and web subsystems. He will also help test the software modules individually and during full system trials, particularly focusing on the move validation, state synchronization, recovery, and latency.

Sabah will primarily be responsible for designing, testing, and integrating the PCB. Sabah and Justin will primarily design the actual game board as well, although she will mostly focus on the physical design accounting for the dimensions of the PCB and other necessary sensing components.

8.3 Bill of Materials and Budget

Item	Source	Price
NVIDIA Jetson	inventory	Free
Camera for Jetson	https://www.amazon.com/dp/B078G88N88	\$39.99
MicroSD Card	https://www.amazon.com/dp/B078G88N88	\$36
HDMI Cable	https://www.amazon.com/dp/B078G88N88	\$4.39
Testing Hall Sensors (Digital)	https://www.digikey.com	11.87
Testing Hall Sensors (Analog)	https://www.digikey.com	13.85
Chess Pieces		
Magnets	https://www.amazon.com/dp/B078G88N88	12.5
USB speaker	https://www.walmart.com	\$14
PCB		\$400
Arduino	Inventory	Free
NeoPixels		Not Decided
Board Material		Not Decided
Total		532.6

Figure 3: Bill of Material

8.4 TechSpark Usage Plans

We plan on utilizing TechSpark for constructing the chess board and internal encasing for the hardware. This will involve using the laser cutters for the acrylic board and wooden encasing. Justin has authorization to use the laser cutters, so he will do all the necessary laser cutting. We also intend on using TechSpark for general testing of individual components; as we have already done with the digital and analog hall sensors.

8.5 Risk Mitigation Plans

The PCB is a core component of our project, so its failure or delayed arrival poses a significant risk. A delay could slow down testing and integration, while a failure would be particularly challenging because our budget does not allow for a replacement. To reduce the impact of delays, we have already tested the exact sensors and magnets with an acrylic board and will continue testing on a larger scale while awaiting the PCB. To reduce the risk of failure, we are reviewing the PCB design with our TA and experienced professionals to identify potential errors before fabrication. The computer vision subsystem is intended to be a backup for board-state verification and recovery, so low classification accuracy could potentially prevent the system from correctly recovering when Hall-sensor state and software states disagree. Accuracy might also be affected by lighting conditions, temporary hand blocking, or visually similar chess pieces. To reduce such risks, we will col-

lect training data across different board configurations and lighting conditions and test the model against manually labeled ground-truth positions. We also will not be putting sole trust in the CV output but rather as one factor. In the event of unexpectedly low accuracy, we may also opt to give more training samples to the CV model. And if it still has issues detecting the types of pieces, we can color the tops of each piece in which each color maps to the type of the piece which the model should have a much easier time detecting. The Jetson relies on a 128 GB microSD card for persistent storage. This might make storage performance and capacity additional constraints. To reduce this risk, software development will be continued on a separate computer while the Jetson environment is being configured. This can be the case since the major software modules are designed to communicate through common interfaces rather than depend entirely on Jetson-specific functionality. Additionally, we intend to rely on the Jetson heavily for the main software process and hosting the web server to localhost, so it's possible that all of the processes will eventually use swap, making the SD card a hard limiting factor on speed. To mitigate this, we can tune some settings in the OS to never use swap on the essential processes and allow the web backend and frontend to use swap. This should be fine for the web performance as it's not the most essential part of the actual gameplay experience and acts more as a logging mechanism for reviewing games after they are played.

upon was integrating all subsystems while still meeting the project's quantitative requirements. The PCB needs to reliably detect piece placements and removals across all 64 squares, the computer-vision system must accurately reconstruct the board when recovery is needed, and the software must keep the physical and digital game states synced. We also need to ensure that the move validation and user feedback remain accurate while also keeping the total response time below the 3 second limit. While the integration is our primary focus for the project, we will be testing each subsystem thoroughly to ensure that each part works such that each component will meet its individual design requirements before full-system integration. This will help us identify problems within sensing, CV, chess-logic, networking, and feedback independently which will make it easier to isolate failures.

Glossary of Acronyms

- PCB - Printed Circuit Board
- CV – Computer Vision
- OS – Operating System
- API - Application Programming Interface
- JSON - JavaScript Object Notation

9 RELATED WORK

The nature of this project is similar to Spring 2023 Capstone Group A0, Tactile Chess. Their project relies solely on analog Hall sensors for detecting both location and type of pieces. Our project instead implements the computer vision model for piece detection. We also slightly widened the scope of the use case to include all on the spectrum of “visually impaired” as only accounting for the blind can limit the usefulness of the system for some. We also think our project slightly differs in that our system is more geared towards learning chess, demonstrated by the hint request feature.

10 SUMMARY

Overall, our project The Smart Chessboard Assistant combines the tactile accessibility of a physical chessboard with automatic game tracking, move validation, instructional support, and accessible feedback. Hall effect sensors will be the primary way of detecting piece movement while the computer vision will be used for piece identification and backup board-state verification and recovery. Our goal is to have the system reduce a visually impaired and/or beginner player's dependence on another person or separate digital interface while preserving the physical play aspect on a shared chessboard.

The main upcoming challenges that our group agreed

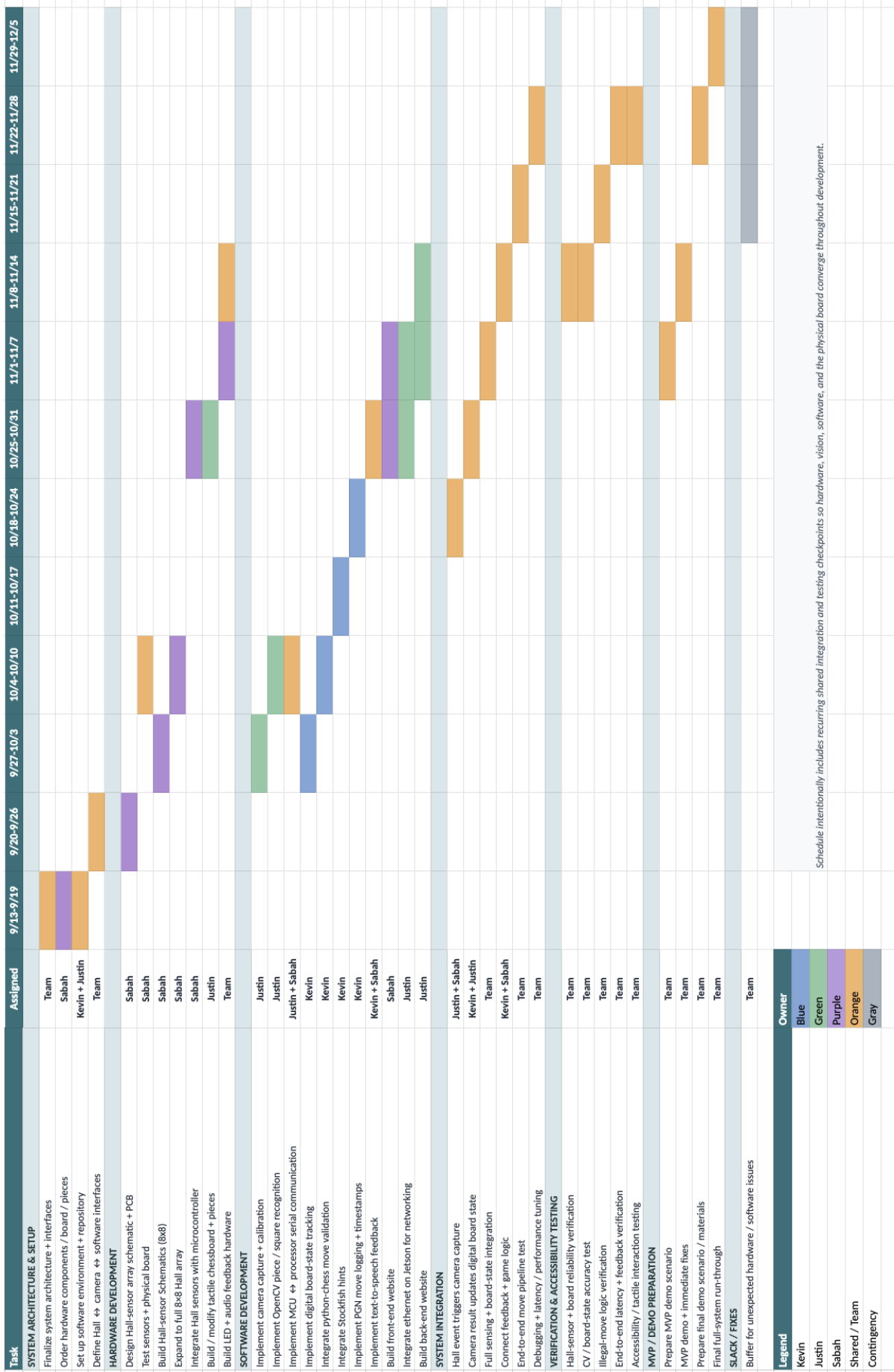


Figure 4: Gantt Chart