



Design and Implementation of a Smart Automated Chess Board

Abdelrahman Thabet^{1*}, and Judhi Prasetyo¹

¹ Middlesex University Dubai

*email: AA5746@live.mdx.ac.uk

ARTICLE HISTORY

Received: 21 February 2025

Revised: 3 April 2025

Accepted: 8 September 2025

KEYWORDS

Smart Automated Chess Board

CoreXY Motion System

Chess Engine Robot

Chess Piece Detection

Chess Gantry Robot

ABSTRACT– This research presents the design and implementation of an innovative automated chessboard. The system autonomously moves chess pieces using a CoreXY motion system with an electromagnet end-effector, while real-time move detection is achieved through Hall Effect sensors. Powered by the Robot Operating System (ROS) on a Raspberry Pi running a competent chess engine, the board allows the users to play against an intelligent opponent with varying difficulty levels. This project bridges the gap between traditional and digital chess, offering an innovative and interactive platform for entertainment and education.

© Thabet et al.



This is an Open Access article distributed under the terms of the [Creative Commons CC-BY-4.0 License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Introduction

Chess has captivated players and enthusiasts for centuries with its deep strategic complexity and intellectual challenge. Traditionally played on a physical board with physical pieces, chess also embraced the coming of the digital age, migrating onto mobile phones and computers. However, this migration comes at the cost of the tactile feeling of playing chess, which players and enthusiasts value. There has been a growing interest in intelligent chess boards, bridging the gap between chess's digital and physical realms.

Robotics involves designing, manufacturing, operating, and applying robots across various domains. From industrial automation to space exploration, robots have transformed necessary tasks, enhancing precision, efficiency, and safety. In all sectors of industry robots have transformed how humans carry out tasks.

Integrating robotics into chess introduces a new fusion of physical and digital chess. An intelligent automated chess board brings home this fusion, leveraging a robot mechanism to move pieces autonomously, sensors to detect the player's moves, and software to dictate the rules of the game and Artificial Intelligence strategies. Such a system would merge the digital and physical domains of chess.

This research focuses on designing and implementing an intelligent automated chess board that seamlessly interacts with chess pieces by leveraging established robotic design and manipulation strategies. By featuring autonomous piece movement, real-time movement detection, and gameplay against artificial intelligence, the system aims to bridge the

gap between traditional board games and modern robotic technology, offering an innovative platform for entertainment and education.

Literature Review

With the integration of robotics increasing in our every day lives, so has integration with board games such as chess. This literature review explores recent advancements in this field. The two main sections of advancements in this field are in detection and methods of moving pieces.

Naturally, as we observe a chess game to identify various pieces and strategies, it made sense to give eyes to robots (cameras) to perform object detection, which performs the same job as human eyes—identifying pieces and their positions on the board using object detection algorithms [1]. However, there are downsides to using cameras for detecting pieces, as they require almost ideal conditions to operate as expected. With the inherent similarity between chess pieces, the slight details that differ could be lost to slightly dim lighting. No chess set is like the other, which creates a difficult job for the vision-based detection system. However, the issue of detecting the square edges could be remedied by overlaying a grid on the image to identify the various positions [2] distinctly. Approaches for not relying on detecting the specific piece include a comparison that compares a snapshot taken after a move is made with the previous snapshot to identify which piece has moved and to which square by the change of the square color distribution (going from a white square to a white square with a black object) [3]

Other approaches are considered for piece detection, such as using various sensors to detect the pieces. One method is to use reed switches to detect magnets on the pieces identifying a change in position on the chessboard [4]. This method forgoes complete issues of a changing environment and depends entirely on magnetic field detection. However, it introduces a problem of identifying the piece itself, as with no vision-based system, there is no direct manner of placing a piece's type. This method also relies on the comparative method as seen in [3]; by tracking changes in data input, the system can identify which piece moved to the new location. Another detection method includes LDRs (Light Dependant resistors), which change value based on the available light [5]. Holes are made above the sensor; therefore, when a piece is covering the hole, the value of the sensor changes significantly, indicating the presence of a piece. However, this method is trampled by the same problem as a vision-based system, as in dim lighting, and it would be challenging to ensure all sensors are not triggered with no pieces covering them.

The method of manipulation also differs significantly from one implementation to the other. Though all methods aim to achieve the same goal, they vary greatly. The first method uses industrial robot arms to manipulate and move the pieces [6]. This method provides precise movement but requires extensive path planning to pick and place the piece correctly, as planning a varying movement for every turn safely is no easy task. Another method of manipulation is the use of XYZ Cartesian motion systems like those on 3D printers, which means that with some modifications, any 3D printer could be a formidable chess opponent [5]. Finally, XY gantry-based systems shift the board entirely as they lie below the board, completely obscured. This approach is safer for users as they cannot interact with the moving parts [7].

Adding to manipulation methods are added methods of interacting with the system, such as voice commands for move designation. This method ensures accessibility to differently able people, allowing the manipulation method to move both sides of the board with verbal instructions given by the player verbally [7].

Material and Methods

A. Methodology

System Design Guidelines

To successfully implement an intelligent automated chessboard, the following guidelines were put in place:

1. Present an intuitive graphical user interface.
2. Perform localized computation for Chess AI and robot control.
3. Use readily available parts to ensure ease of maintenance.
4. Integrate proven robotic mechanisms.
5. Use ROS for real-time communication and control.
6. Design the system to be scalable.
7. Perform autonomous piece movement.
8. Implement real-time move detection.

9. Implement chess rules.
10. Operate with a singular power input.
11. Ensure a fully enclosed design with no external moving parts.

B. Hardware Architecture

Motion System Selection

The selection of the motion system was critical in achieving the goals of speed, precision, and repeatability. Several options, such as Cartesian XY, H-bot, and CoreXY, were evaluated based on the following:

- Speed: A lighter load on the moving end-effector reduces the momentum needed for acceleration and deceleration.
- Accuracy: The CoreXY design minimizes vibrations and drifts by having no motor attached directly to the moving parts.

After reviewing references [8] and open-source projects like the Jubilee 3D printer developed by Joshua Vasquez [9], it was clear that the CoreXY motion system would meet the requirements for the intelligent chess board, offering both mechanical simplicity and precision [10].

Sensor Selection

Detecting the chess pieces is crucial, leading to considering two sensors: Hall Effect sensors and Reed switches. Both are capable of sensing magnetic fields. However, Hall Effect sensors were selected for the following reasons:

- Durability and reliability are stemming industrial use-cases.
- Easy integration into a PCB, unlike the fragile glass or plastic casing of Reed switches.

Although Hall Effect sensors require an extra pin compared to Reed switches, their robustness and compatibility with the overall system outweighed this minor complexity.

End-Effector Design

The motion system required an end-effector capable of moving chess pieces within the fully enclosed system. An electromagnet was selected as the optimal tool for this task due to its ability to interact with the magnetic base of each chess piece without the need for external, mechanical contact.

Power Supply and Enclosure

To streamline the design, a single input powers the system, and all components are housed in a fully enclosed device with no external moving hardware.

C. Software Architecture

The Robot Operating System (ROS) was selected as the communication and control framework due to its support for real-time robot applications. ROS establishes communication between various controllers. ROS's ability to interface with Python and C++ simplifies the integration of any proposed system.

D. Control System

Given the system's complexity, multiple microcontrollers and microprocessors were chosen to handle distinct tasks:

- Raspberry Pi: Runs the chess AI and interfaces with ROS for high-level control.
- Arduino R4: Provides increased processing power and memory for managing sensor data.
- Arduino Mega: Handles motor control using a Ramps board with stepper motor drivers.
- Arduino GIGA: Powers the graphical user interface on a high-quality display.

This combination of controllers ensures scalability, processing efficiency, and ease of integration.

E. Testing and validation scenario

Testing the board's performance will be evaluated and validated quantitatively and qualitatively. The testing criteria for each are outlined below:

Qualitative Criteria:

- The chessboard responds correctly to player moves and inputs.
- The gantry accurately reaches desired positions and moves pieces successfully.
- The interface should be straightforward, clearly indicating win/lose status, move validations, and the current playing color.
- The board should be able to move pieces to the correct positions.
- The system effectively detects invalid moves—such as illegal captures, moves to invalid squares, or illegal castling.

Quantitative Criteria:

- The chessboard is expected to reliably place pieces over the sensors and detect its placement.
- The interface should update rapidly and responsively.
- The system recognizes valid moves and initiates bot moves almost instantly.
- The board powers up rapidly after being turned on.
- Bot moves are expected to be executed rapidly with no delay.
- Moves on the board should be repeatable consecutively and non-consecutively with no discrepancy.
- The system is expected to play multiple consecutive games without performance degradation or crashes.

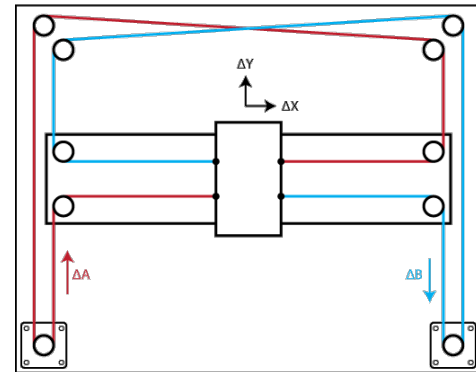
Implementation

The development process was divided into three phases to successfully implement the smart, automated chess board: Mechanical Design, Electronic Design, and Software Design. Following the completion of each phase, the components

were carefully integrated to ensure they function harmoniously and cohesively.

Mechanical Design

Starting with Mechanical Design, having studied the operation principle of CoreXY, we started designing a gantry based on the following schematic[5] (Moyer):



Equations of Motion:

$$\Delta X = \frac{1}{2}(\Delta A + \Delta B), \quad \Delta Y = \frac{1}{2}(\Delta A - \Delta B)$$

$$\Delta A = \Delta X + \Delta Y, \quad \Delta B = \Delta X - \Delta Y$$

Figure 1. CoreXY Technique

and also referencing the CoreXY implementation in Jubilee [8](Vasquez), we were able to create the following design:

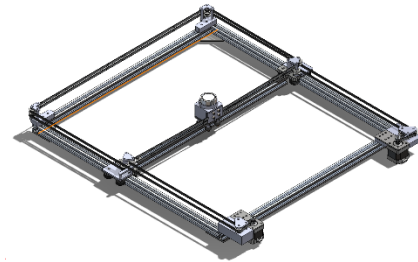


Figure 2. Gantry Design

This overall design operates with the CoreXY principle with two stepper motors working harmoniously to move the end effector in the X and Y directions. The gantry is built out of 2020 aluminum V-slots brought together using OpenBuilds Corner plates. This gantry consists of the following parts:

- 1) Motor Mount (Figure 3)
- 2) End-Effector (Figure 4): The second crucial part of the mechanical design is the end effector that will traverse the board and interact with the chess pieces. The end effector consists of four main parts:
 - Electromagnet 12 volt
 - Belt Clamping plate
 - Endeffector Body
 - OpenBuilds Mini V gantry kit

The end effector assembly provides enough clamping force for the belts to move the end effector. It is also stable enough

on the V slot to prevent vibrations or fast movements from interfering with its operation.

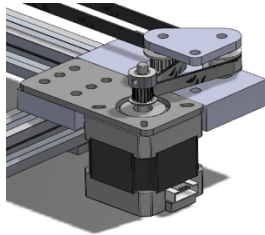


Fig. 2. Motor Mount

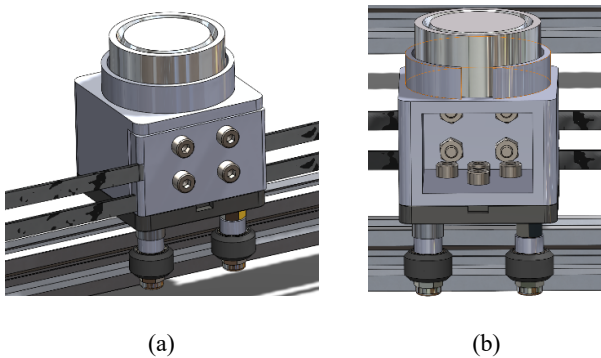


Figure 4. End-Effector. (a) Design 1, (b) Design 2.

3) Corner Stabilizers (Figure 5): The corner stabilizers are attached to the far corners of the Gantry using T-Nuts and have two staggered pulleys for the driving belts pathing around the Gantry to the end effector. Stabilizing plates and columns provide stable movement for the belts. Both corner stabilizers are identical mirrors.

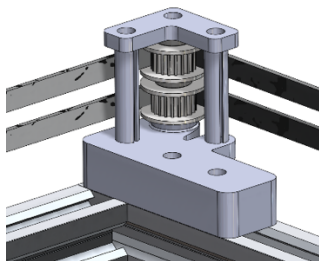


Figure 5. Corner Stabilizers

4) Left Carriage (Figure 6): The left carriage houses one pulley and an idler placed in specific locations to allow the belts to connect to the end effector in a parallel fashion. It is attached to an OpenBuilds V-Slot Carriage kit. The design features supports and a stabilizer plate to ensure no vibrations or inaccuracies are produced due to loose belt paths.

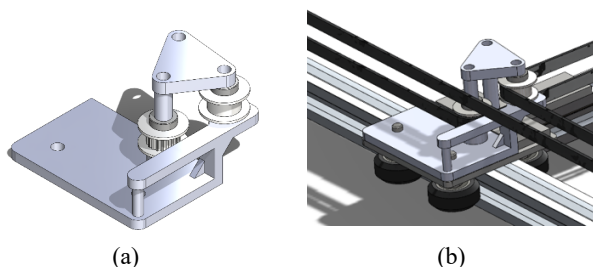


Figure 6. Left Carriage (a) Part, (b) Assembly.

5) Right Carriage (Figure 7): The Right carriage is similarly equipped; however, it features a unique design to accommodate the different pathing required for the belts to be secured to the end effector. However different it looks, it possesses an identical stabilized layout and carriage kit.

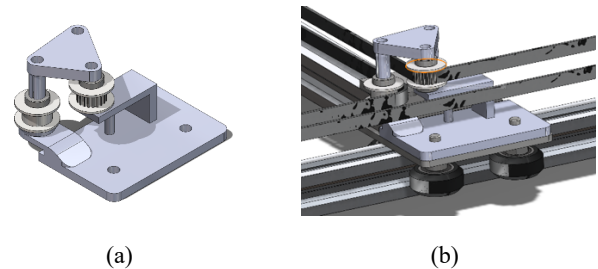


Figure 7. Right Carriage (a) Part, (b) Assembly.

6) Final Implementation (Figure 8): The Gantry was fully assembled to spec as per the initial design shown in the following figure. After testing, it proved fully capable of traversing its two degrees of freedom.



Figure 8. Assembled Gantry

7) Enclosure Design (Figure 9): After designing the Gantry, it was necessary to design an enclosure to fit the Gantry along with the additional electronics without obstructing movement or functionality. Multiple materials were considered, from 3D printing the enclosure to acrylic, and after a few revisions, it was deemed acceptable for its affordability and reasonable strength. Following is the final iteration:

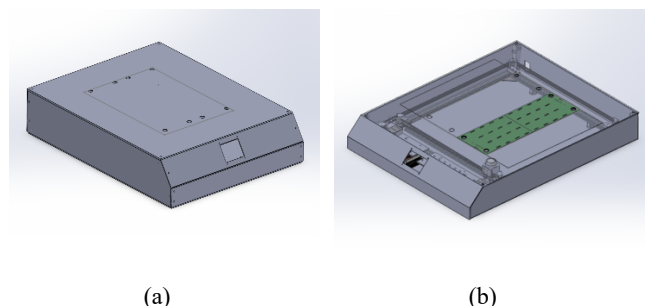


Figure 9. (a) Enclosure Assembly, (b) Transparent view.

The enclosure breaks down into five distinct panels that screw into the supports mentioned in Mounting Parts

subsection. The top front panel houses the mounting points for the HMI (human-machine interface).

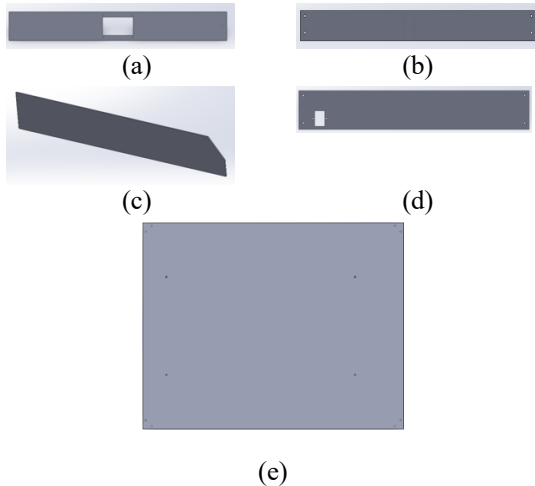


Figure 10. Enclosure Panels (a) Top Front Panel, (b) Bottom Front Panel, (c) Side Panels, (d) Back Panel, (e) Top Panel

The Bottom Front and side panels protect the parts inside while giving a view of the board's operation.

The back panel, though serving a similar purpose to the others, houses the power input to the system. It uses a standard fused 3-pronged input with a switch.

The bottom plate is the most structurally loaded panel in the system. Required to bear the full weight of the Gantry and accompanying parts, It was favorably decided not to be machined out of acrylic but rather wood for its strength.

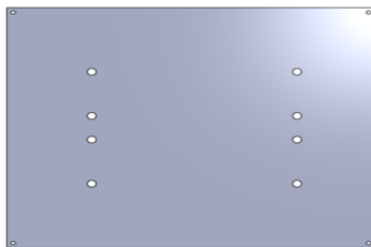


Figure 11. Bottom Plate

The top panel, however, stressed it needed to be made out of acrylic to maintain the lowest thickness so as not to obstruct the functionality of the system. Electromagnets are mainly designed to operate directly on the object; therefore, increasing the distance further could impact its performance. The top panel is also designed to be mounted with magnets to the structure for easy maintenance and accessibility to the various components.

8) Mounting Parts: Various mounting parts and supports were needed to attach the panels and maintain a rigid structure. Tower-like supports are designed for the panels to screw into. They're designed with a hole in the top for a magnet to attach to the top panel.

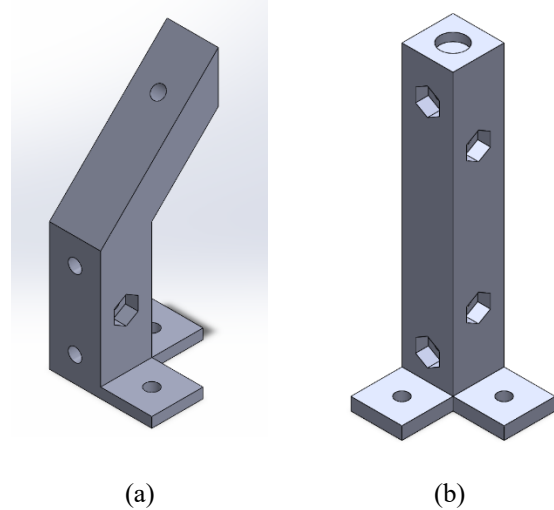


Figure 12. Support (a) Front, (b) Back

Due to the angled structure of the front supports, an added part was needed for front magnets to attach to the top panel; these magnet enclosures are held in place using screws to the side panels.

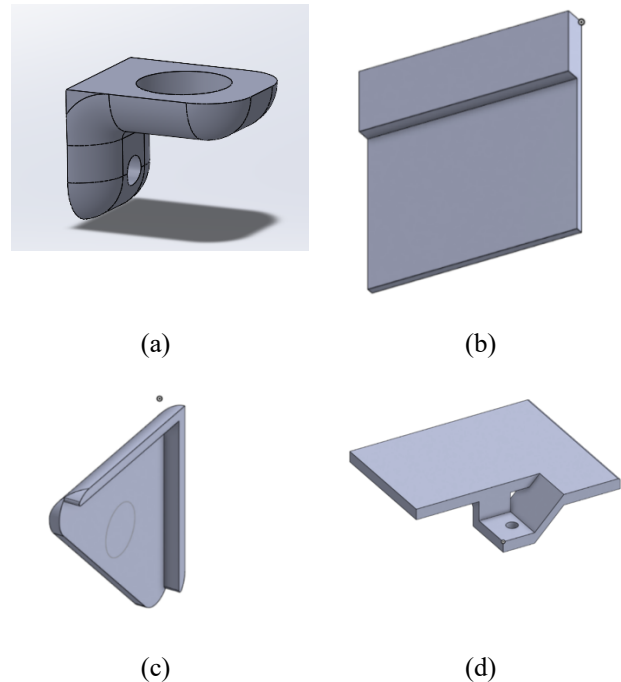


Figure 13. Mounting Parts (a) Top panel magnet mount, (b) PCB clip, (c) Top Panel Magnet attachment, (d) Front Top panel Additional Support

The following support adds additional support to the front top panel to withstand the added weight of the screen and maintain a rigid structure when the user is pressing the screen.

Panel supports bolted to the bottom panel were needed to add rigidity to all the vertical panels. Without them, the inherent flimsy nature of acrylic panels presents itself, making the structure impractical.

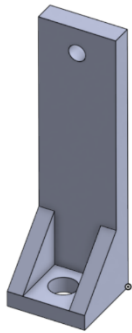


Figure 14. Side panel support

Bringing all of these parts together with a chess sticker on top results in the following assembly:



Figure 15. Fully Assembled Chess Board

9) Electronic Mounts: The following PCB mounting parts are oppositely attached from the top panel to the underside of the PCBs and held together by accompanying nuts and bolts. These mounts maintain the PCBs in position but allow some freedom, mitigating any bending that might occur, damaging the PCBs.

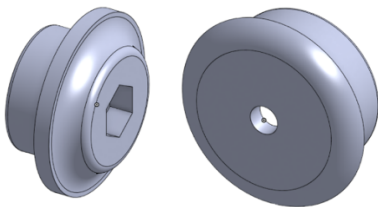


Figure 16. PCB mounts

Each limit switch required a distinct mount to prevent shorting with the aluminum V-slots. They attach to their respective sides of the Gantry using T-Nuts.

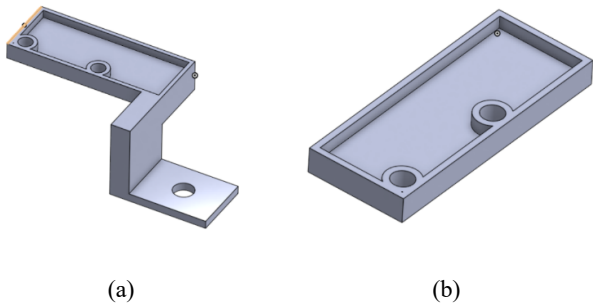


Figure 17. (a) X limit Switch mount, (b) Y limit Switch mount

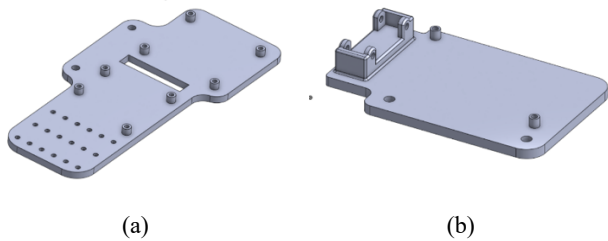


Figure 18. Controller Panel, (b) Ramps Panel

The controller and Ramps panel house all controlling components in the system. They shield exposed soldered pads from the V-slots. Also added to the Ramps panel is a slot for a fan to mitigate overheating in the system.

In the following image are all the panels mounted and secured to the Gantry:

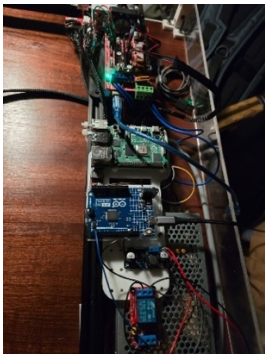


Figure 19. Controller Panels

Electronic Design

1) Circuit Design: After deciding that the chess pieces would be detected using Hall Effect sensors, the manner of connecting the 64 sensors puzzled me, as connecting 64 pins for data only was not feasible. However, using a 16 by four-channel multiplexer, specifically the CD74HC4067, reduces the required pins for data from 64 to a mere nine pins. Going one step further and soldering the sensors to meticulously designed PCBs helps achieve pinpoint precision of piece alignment and positioning. The proposed schematic and PCB are seen below:

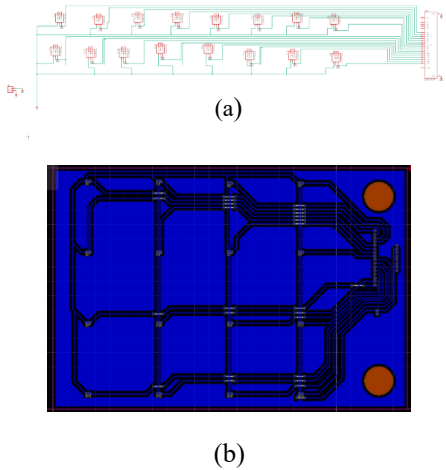


Figure 20. (a) PCB Schematic (b) PCB Design

Each PCB holds 16 sensors that the multiplexer can access and pass through to the microcontroller. With that sorted, the four identical PCBs were connected to the microcontroller without compromising more than nine pins. This situation is where the enable pin available on each multiplexer proves powerful, as exploiting this feature makes it possible to chain common cables, such as the selection and signal pins, together without compromising the data integrity. A schematic of the electronic circuit is shown below:

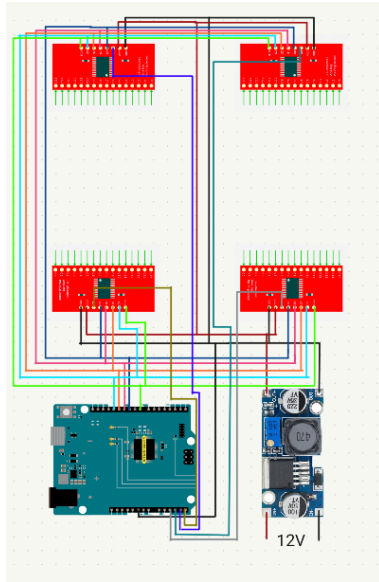


Figure 21. PCB to Microcontroller Circuit Schematic

The PCBs are connected to a Step-down voltage regulator (12V to 5V) through a controlled relay to accommodate their power requirements and decrease interference from other devices. However, they share a ground connection with the Arduino for a common voltage reference. The schematic shows the Selection pins (S0, S1, S2, S3) and the Signal pin (SIG) on all PCBs present on common wires, respectively, and connected to the same pin on the Arduino. However, the enable pins are connected to discrete control pins on the Arduino. The Signal pin is connected in a Pull-up configuration as, naturally, hall-effect sensors provide floating values when inactive; this ensures that the pin output is high (5V) when inactive and produces a low (0V) signal when active.

2) Power Delivery: Moving on from circuit design, the overall power delivery system, according to the guidelines set prior in section Methodology, the Smart Chess Board should be powered by a single power source. Following that guideline, we settled on using a 12V power supply connected to 220V mains to power the system. That selection was based on the fact that the Ramps board with the stepper motors requires 12V to operate and adequately drive the Nema-17 Stepper Motors. To accommodate the Voltage requirements of the other components, we used a 12V to 5V USB step-down voltage regulator to provide power to the Raspberry PI, which would, in turn, use USB power delivery to power the Arduino GIGA and R4, respectively. This allows the components to communicate over USB and maintain the use of their voltage regulators to prevent components from failing <https://doi.org/10.59097/jasae.v3i2.52>

due to an accompanying voltage regulator failure. This connection strategy secures the reliability and safety of our components.

3) Gantry Components: Moving the Gantry are two NEMA-17 stepper motors that operate harmoniously to access movement in the X and Y axes. These specific motors were selected for their ideal accurate step movement, reliability, and sufficient torque.

The Gantry requires at least two limit switches to execute a homing function correctly. Therefore, attached to the Gantry are two Maker Bot limit switches identifying the zero location of the X and Y axes, respectively. Maker Bot limit switches were selected due to their availability and simplicity.

Software Design

The Software bringing this is based on a discrete system, Higher and lower level control. However, to designate the exact functions of each level, the features to be implemented were identified as follows:

- Move Validation
- Piece movement detection
- Chess engine with variable difficulty
- Implementation of chess rules
- Accurate movement of chess pieces
- Present the user with useful data on the interface reflecting the piece movements on the board

We developed the following flowchart as a general guide for the operation of the chessboard:

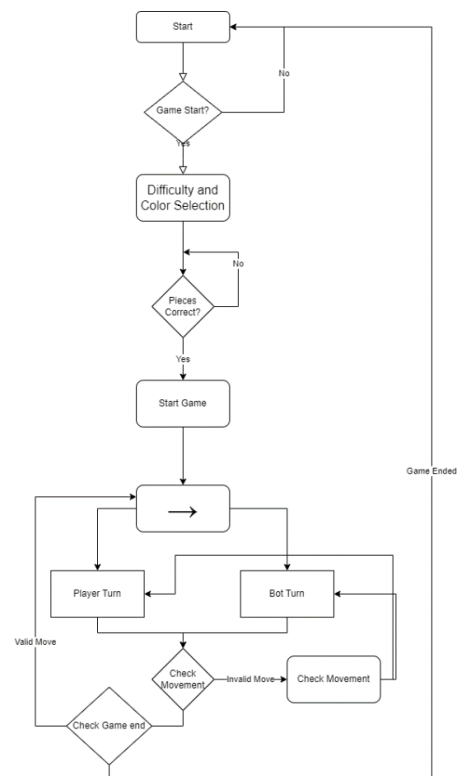


Figure 22. Chess Board Flow Chart

1) Lower Level Control: The lower level control is reserved for the Microcontrollers (Arduinos) connected to the

actuators and sensors. These were mentioned previously in section Methodology. Starting with the Arduino R4 receiving and relaying the sensor data through a ROS topic to the Raspberry PI. Following that, the Arduino mega with the Ramps 1.4 Shield Receives positional and magnet operational data through a ROS topic from the Raspberry PI and relays sufficient feedback to indicate its status. Finally, the Arduino GIGA receives a plethora of data on multiple ROS topics, feeding live data on the chess game displayed to the user. It also handles the player's requests (color selection, ending turn, forfeit, difficulty). It sends the data on the appropriate topics to be handled by the scripts present on Raspberry PI.

Higher level control on the Raspberry PI houses all three scripts, each with a unique ROS node that communicates over topics to relay key information about the board and game status. The following breakdown explains briefly what each node does:

- Chessboard_move detector: This node handles all sensor data and tracks each piece in the 8x8 chessboard. It identifies captures and moves of unique pieces and extracts UCI moves to be validated and captured by the Python Chess library in the stockfish_node.
- Stockfish_node : Using the chess library, this node validates moves and keeps track of ongoing games. It generates moves using the Stockfish engine and communicates with the gantry_controller to execute piece movement.
- gantry_controller: All gantry movement is handled by this node. It identifies movement patterns (diagonal, vertical, horizontal, etc.) and designates the ideal path. It sends positional and magnet data to the Ramps board to produce the desired piece movement and then provides feedback to confirm successful movement.

This system proves sufficient to detect and identify moves with redundancies to prevent foul play and implement chess rules correctly. It also satisfies the player's needs with a live preview of the chess board and sufficient configuration of game settings.

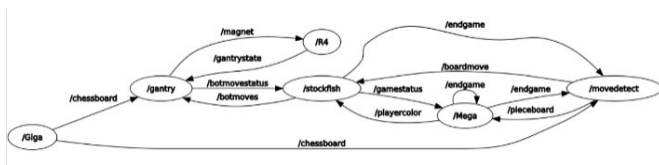


Figure 23. RQT Graph

The RQT graph above presents all the previously mentioned nodes and topics connecting them.

Results and Discussion

The smart chess-playing board, the result of this project, not only plays the game but exhibits its functionality through measurable and observable behaviors of the chessboard, both qualitatively and quantitatively.

A. Qualitatively

- The Chessboard responds to player moves and inputs in an expected manner.
- The Gantry reaches the desired positions correctly and moves the pieces successfully.
- The interface is straightforward.
- The Knight movement sometimes clips neighboring pieces.
- The chess board, given correct placement by the player, detects the player moves correctly every time.
- The chess board, given correct placement by the player, detects the player moves correctly every time.
- The interface indicates win or lose status and reasoning correctly.
- The interface indicates validation of move correctly.
- The interface identifies which color is playing correctly.

B. Quantitatively

- The chessboard places the chess piece correctly over the sensor 8/10 times, measured over five consecutive games played with the bot.
- The chessboard updates its interface view of the board every 500ms.
- The chessboard recognizes moves and starts bot moves almost instantly if valid.
- The chessboard starts up within a minute of turning on the switch
- Each bot move, depending on the distance needed to be traveled, takes between 20 to 50 seconds.

The board must first detect moves correctly and when foul play is executed. Testing was done by attempting to capture in an invalid move, moving a piece to an invalid square, and castling when it was not possible. In all three cases, the board reported the invalidity of the move successfully in all stages of the game. The board reports that the moves are invalid to the player through the interface. Following that, the board could always detect correct moves, given the piece placement was accurate. This was tested by playing multiple games with the robot.

Additionally, the board needed to execute diagonal and straight moves successfully with no deviations. With careful observation, it maintained direct paths while executing those moves. The board was also required to perform moves and reach positions in a repeatable fashion, testing whether the board could repeat the same series of movements, repeatably reaching the same designated squares over 20 iterations. The sequence of movements moves the bot to each corner of the board consecutively, followed by a diagonal traversal across the board. However, knight movement causes minor

collisions when passing between squares, but this is due to the offset piece placement.

Finally, the system must play multiple games consecutively without degrading performance or crashing. This was observed by playing five consecutive games with varying player sides and bot difficulties, with no visible degradation observed to the board's performance.

Given the previous observations, the board can complete full games consecutively. However, careful placement of the pieces in their correct positions is required for sensor detection. Sometimes, when the bot misses its place point, the player would need to intervene and ensure the piece is detected correctly.

These issues are primarily due to the small proximity area of detection by the sensors and the small magnets on the pieces. However, the magnet size was required to prevent any interference between pieces, as when larger sizes were tested, they would repel each other during movement.

The user testing results highlighted several key areas for improvement. Users found that the board was extremely challenging to beat suggesting the addition of an easier bot to create a more beginner-friendly experience. They also commented on the substantial size of the board, which made it feel impractical for daily use and transport. Additionally, users observed that the sensors' limited detection radius introduced a learning curve, as players had to be particularly precise when placing pieces. Users also suggested adding more feedback mechanisms from the board to enhance the playing experience, such as lights to show the valid squares when moving a piece. Some users even favored a more traditional chess clock interface where screen interaction is optional, emphasizing a desire for a simpler, more streamlined playing experience.

Conclusion

Design and implementation of a working prototype of a Smart Automated Chess Board has been successfully realized. It performs the required actions correctly and efficiently. However, it lacks some accuracy in some cases, which could be remedied in multiple ways. The project would benefit greatly from a revised gantry design to reduce size and weight. It would also benefit from different Hall Effect sensors for larger proximity detection. Furthermore, integrating all the control boards into a more straightforward setup would reduce the space and connections used in the project. Finally, adding features such as online gameplay, puzzles, and lessons would transform this prototype into a convincing product for chess enthusiasts.

References

- [1] R. Zhang, J. de Winter, D. Dodou, H. Seyffert, and Y. B. Eisma, "An open-source reproducible chess robot for human-robot interaction research," arXiv preprint, 2024, doi: 10.48550/arXiv.2405.18170.
- [2] P. K. Rath, A. M. Pitale, and S. N. Shah, "Autonomous chess playing robot," in 2019 28th IEEE Int. Conf. Robot and Human Interactive Communication (RO-MAN), 2019, pp. 1–6, doi: 10.1109/RO-MAN46459.2019.8956389.
- [3] G. Ranganathan, "An economical robotic arm for playing chess using visual servoing," Journal of Innovative Image Processing, vol. 2, no. 3, pp. 141–146, Jun. 2020, doi: 10.36548/jiip.2020.3.004.
- [4] K. H. Zurange, "Professional AI chess board autonomous chess board (human vs. robot)," International Journal of Engineering Research & Technology (IJERT), vol. 9, no. 5, 2020, pp. 135–138, doi: 10.17577/IJERTV9IS050139.
- [5] V. Gupta, A. Kumar, S. Jaiswal, and S. Agrawal, "Autonomous chess playing robot," International Journal of Engineering Research & Technology (IJERT), vol. 4, no. 3, Mar. 2015, doi: 10.17577/IJERTV4IS030620.
- [6] J. Golz and R. Biesenbach, "Implementation of an autonomous chess playing industrial robot," in Proc. 2nd Int. Conf. Engineering Science and Innovative Technology (ESIT), Phuket, Thailand, Apr. 21–23, 2016. [Online]. Available: <https://www.researchgate.net/>
- [7] N. U. Pilueta, H. D. Grimaldo, M. F. Jardiniano, and M. B. Garcia, "Chessbot: A voice-controlled chess board with self-moving pieces," AIP Conference Proceedings, vol. 2502, no. 1, Article 040001, 2022, doi: 10.1063/5.0108986.
- [8] I. E. Moyer, "CoreXY | Cartesian motion platform," CoreXY.com. [Online]. Available: <https://corexy.com/theory.html>. Accessed: Sep. 18, 2024.
- [9] J. Vasquez, "machineagency/jubilee: Jubilee source files," GitHub repository. [Online]. Available: <https://github.com/machineagency/jubilee>. Accessed: Sep. 18, 2025.
- [10] B. Gurav, "CoreXY 3D printer: All you need to know," All3DP, Jul. 1, 2023. [Online]. Available: <https://all3dp.com/2/corexy-3d-printer-is-it-worth-buying/>. Accessed: Sep. 18, 2024.