Chess Playing Robotic Arm Using Stockfish

Pranav Dhakate, Onkar Gaikwad, Raman Gandewar, Prathamesh Ghalsasi, Divij Gujarathi, Anil Kadu
Vishwakarma Institute of Technology, Pune

Abstract— **With the implementation of hardware control using an Arduino Uno microcontroller as well as the software functionality of the Stockfish chess engine, the author reports the build-up and installation of a visionless robotic arm to play chess and move pieces by itself. A gripper is powered by an MG90 servo motor, whereas a three-phase MG995 servo motor drives base, shoulder, and elbow motion of the robotic arm. Such a cost-saving approach demonstrates the potential of smart robots in interactive games using minimal hardware as calculations of inverse kinematics ensure the accurate placement on the board without the use of a camera.**

Keywords— Chess-playing robot, Stockfish engine, Arduino Uno, MG995 servo motor, inverse kinematics, robotic arm, autonomous systems.

I. INTRODUCTION

One of the most significant advances in robotics and artificial intelligence is the development of robotic systems capable of competing against human players in gaming environments, such as chess. Chess, with its high strategic complexity, has been at the forefront of demonstrating AI capabilities since the mid-20th century, with milestones such as IBM's Deep Blue's defeat of a reigning world champion in 1997. An additional level of complexity is introduced when computation-based intelligence is mapped onto a physical robotic platform, where fine-grained mechanical control and sensing of the environment become essential.

This study avoids the need for vision systems employed in chess-playing robots through a cost-effective but efficient methodThough extremely robust, vision systems are extremely expensive and demand a lot of computational power and are thus not suitable for use at home or in small educational settings. This robotic arm uses user input as well as pre-programmed board points to define and execute movements.

The robotic system performs fantastically due to the maximal move generation exploitation through the Stockfish chess engine. Meanwhile, the hardware configuration, powered by an Arduino Uno driving MG995 and MG90 servo motors, illustrates how cheap sensors can be leveraged to yield accurate and consistent motion.

This research seeks to close the divide between theoretical AI and real robotics by illustrating that the potential exists for table software and basic hardware to be coupled to provide smart, interactive capabilities. The main goals are to build a structurally stable and functionally efficient robotic arm, create a skilled control algorithm, and include the system's ability to play a full game of chess.

II. LITERATURE REVIEW

To design a robotic arm, it’s important to understand how its movements work including its degrees of freedom, how easily it can reach different areas, and how much space it operates in. A study showed that testing different setups is crucial for a Chess-Bot, where the arm must follow game rules within a fixed space. [1] Robots often struggle in environments where they can’t fully “see” everything. In a test, a 6-jointed robotic arm with a gripper still performed well, showing that even general-purpose robots can handle tasks like playing chess if game rules and safety are considered. It achieved around 92% success during its turns. [2] A good Chess-Bot needs reliable mechanical parts. Tools like linear sliders and four-bar linkages help with piece movement and capturing. Smart programming is equally vital, and using algorithms like Minimax and alpha-beta pruning helps the robot play smarter. [3] To test the bot’s intelligence, chess engines like Leela Chess Zero (LCZero) and Stockfish are compared. Stockfish often outperforms LCZero, especially in complex puzzles like Plaskett's, due to its efficient algorithms. Bellman’s equation also helps improve the bot’s chances of winning. [4] For the robot to understand the board, it needs solid vision. A computer vision method using edge and shape detection was found to work well even in low light, allowing the robot to adjust and respond better during the game. [5] In short, designing a Chess-Bot brings together lots of different fields like how things move (kinematics), how to build strong parts (mechanical design), how to make good decisions (smart algorithms), and how to “see” (computer vision). This review of past research gives a strong starting point for building a smarter and more adaptive chess-playing robot. It’s all about combining smart programming with engineering to create robots that can play well, think smartly, and react to the world around them.

III. METHODOLOGY

System Overview:-
The proposed robotic arm chess-playing system consists of two main parts:
Software System: for move generation and coordination using the Stockfish chess engine.
Hardware System: Robotic arm and control electronics, responsible for executing the moves.

Hardware Design:-

Robotic Arm Configuration:

Degrees of Freedom (DOF): The arm consists three DOF for movement (base, shoulder, and elbow) and an additional DOF for the gripper.

Servos:
Base Joint: MG995 servo for rotational motion to align against columns on the chess board.
Shoulder and Elbow Joints: Two MG995 servos for vertical and horizontal positioning of the arm.
Gripper: MG90 servo for gripping and releasing chess pieces.
Material: Lightweight and rigid materials for the arm ensure durability while maintaining Maneuverability.

Chessboard Setup:
A standard chessboard with predefined square dimensions (e.g., 50 mm per square) is used.
The robotic arm is placed at a fixed position relative to the chessboard (15 cm by 11.5 cm in your project).
Electronics:
Arduino Uno serves as the main microcontroller, interfacing with the servos and the chess engine.
A power supply unit provides sufficient current to drive the servos without voltage drops.

Software System:-

Stockfish Integration:
Stockfish, an open-source chess engine, is employed for calculating optimal moves based on the current board state.
The engine runs on a connected computer, and moves are communicated to the Arduino via serial communication.

Inverse Kinematics Calculations:
The location of every square on the board is translated into joint angles for the servos through the inverse kinematics method. In determining the actual angles required for every position, a potentiometer recorded and measured servo angles during manual calibration.
This method provided precision through the association of calculated angles and real-world positions. Inputs used are: arm base to target square distance, shoulder joint vertical displacement from the floor, and arm segment lengths. As a reference from potentiometer readings, well-calibrated angles for every square (e.g., A1, B2, etc.) were calculated and stored for ready use in actual gameplay.

Control Logic:
User enters the opponent's move through a serial monitor.
Stockfish computes the response move, which is converted to servo angles.
The Arduino makes the move by sending PWM signals to the servos.

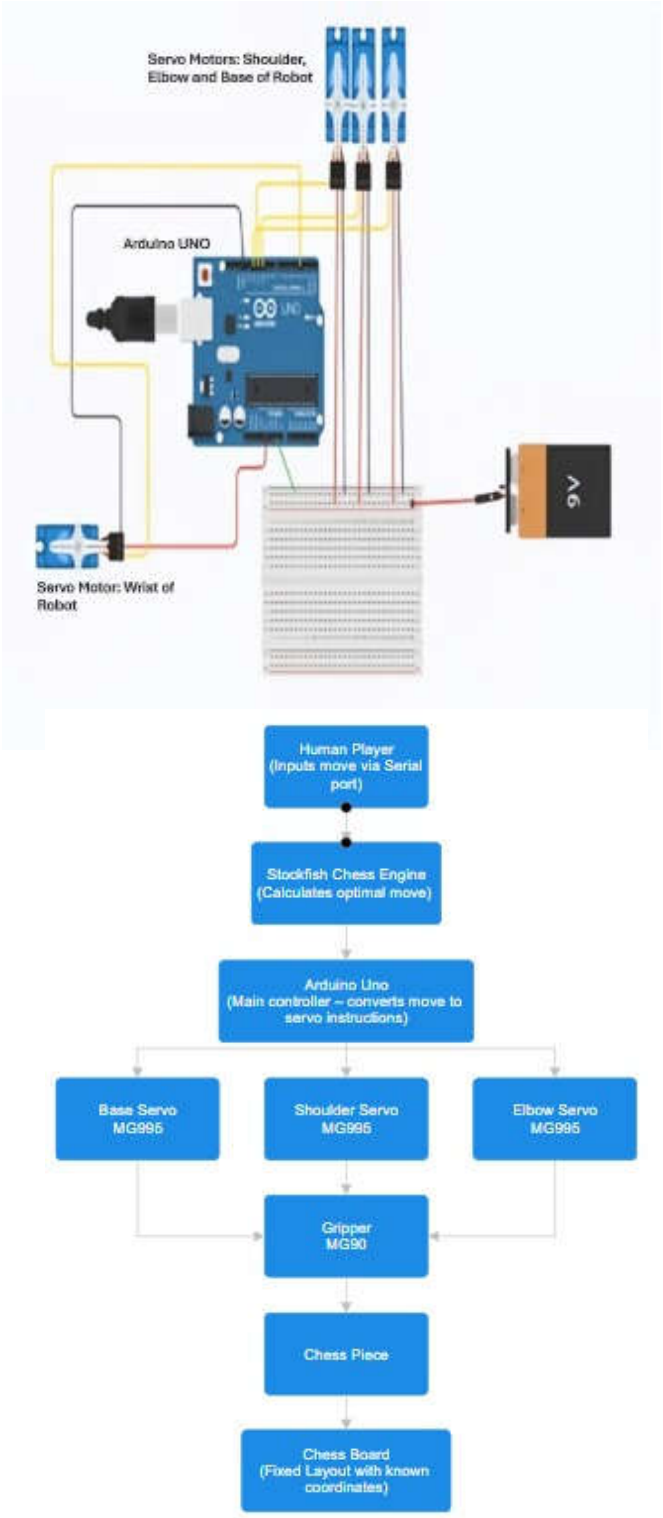

Figure 1: Circuit Diagram and Flowchart

Integration Process

Calibration:
Potentiometers were employed at calibration to determine the exact servo angles for each chess board square.
Fine-tunning ensured accurate placement for all the pre-defined board coordinates.

Testing:
The arm is lso tested on different chess positions, ranging from opening moves to mid-game complexity and endgame precision. Edge cases like piece collisions or invalid inputs are managed with error correction mechanisms.

IV. IMPLEMENTATION

The installation process of the chess-playing robotic arm system involved integrating hardware and software components to achieve a functional and efficient system. This section describes the hardware assembly processes, software programming, calibration, and testing processes.

Hardware Assembly:

The robot arm was built using light and strong materials for stability and minimal weight on servo motors. The arm was built using three (DOF)base, shoulder, and elbow and one additional (DOF) for the gripper. The base rotation was accomplished using an MG995 servo and the shoulder and elbow using two MG995 servos. An MG90 servo was utilized for the gripper to lift and release the chess pieces was driven by a MG90 servo.

The arm was positioned in the normal position with respect to the chess board. The arm was positioned 11.5 cm vertically and 15 cm horizontally from a corner of the board. The arm was able to reach all tiles because to this positioning.

The servos were controlled by the Arduino Uno microcontroller, which also handled hardware-software communication. The Arduino was the main controller, interpreting chess engine input into servo movement. The system was powered by an external power supply that provided sufficient current to power the servos during run-time in a stable and reliable way.

Software Development:

The program was written to communicate with the Stockfish chess engine, which determined the optimal chess moves based on the position of the game. Stockfish was executed locally on a computer, and the chess moves were sent to the Arduino via serial communication. The Arduino program then translated these moves and translated them to precise servo angles that the robotic arm would use to make the move.

The robotic arm moved under the control of inverse kinematics.

Inverse kinematics is an algorithm to compute the joint angles (shoulder, elbow, base) required to reach a target end effector position of the arm. The chessboard coordinate squares were systematically translated to joint angles for the three big servos.

The Arduino control program generated Pulse Width Modulation (PWM) pulses for every servo, thus actuating the servos to the proper positions for every move. The robot base was first rotated towards the target square, then the shoulder and elbow joints adjusted to place the gripper on top of the piece. The gripper closed to pick up the piece, and the arm lifted it upwards. Positioning over the target square, the arm opened the gripper, and the chess piece would fall.

Calibration:

Calibration was an important step to find out the consistency of the robotic arm's movement. A potentiometer was mounted on every servo angles at different locations on the chessboard. The arm was manually positioned at every square on the chessboard, and the respective potentiometer values were noted.

These values were employed to realize the maximum servo angles in every chessboard position. Using calculated angles as a reference point compared to real angles produced by the potentiometer in the real world enabled the pre-stored values for angles to be calibrated to compensate for any difference.

Preparatory experiments were performed to determine the accuracy of the robotic arm against board positions A1, B2, and C3. Inaccuracies were corrected through the process of incrementally tuning the servo angles to the point that the robotic arm precisely arrived at the targeted positions.

System Integration:

Once the hardware and software components were assembled and calibrated, the system was integrated to function as a single system. The user input the opponent's moves through the serial monitor in algebraic notation, e.g., "e2 to e4." The moves were then interpreted by the Stockfish chess engine, which determined the best move given the current state of the board. The move was then sent to the Arduino, which operated the robotic arm to perform the action.

Testing and Validation:

The system was tested to determine its response under actual game conditions. The robotic arm was positioned on the board in different opening, mid-game, and complex positions to determine its precision, accuracy, and response time. The arm performed the mandatory moves effortlessly, taking an average of 10 to 15 seconds per move, depending on the complexity of the move.

Stress testing was also conducted to determine the robustness of the robotic arm and its performance in cases of prolonged usage. The system was also subjected to prolonged game playing sessions to verify that the servo motors showed consistent performance.

V. OUTPUT

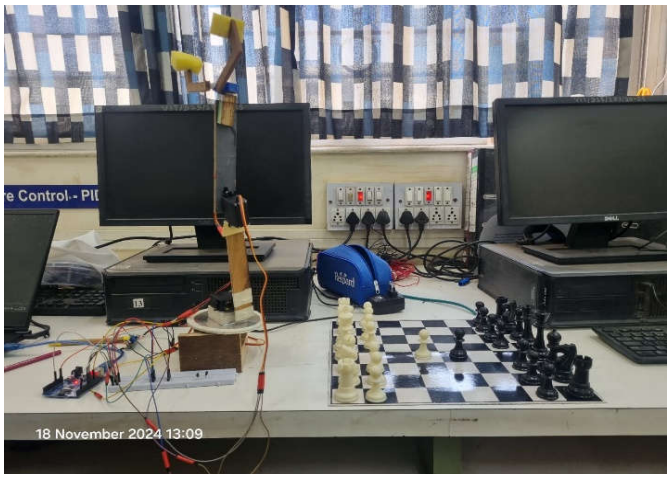

Figure 2: Working Model of Chess-Bot

The robot arm, which was specially made to play chess, was thoroughly tested under different conditions to prove its performance, accuracy, and speed. The findings are described as follows:

Precision in Move Execution:

The robotic arm moved to the position of chess pieces with precision.

Position mistakes for the coordinates provided were 2 mm, and that is okay for playing chess.

Potentiometer-based calibration integrated introduced consistency to accuracy in play.

Move	From → To	Target Coord. (X, Y)	Placed Coord. (X, Y)	Position Error (mm)	Success (Y/N)	Remarks
1	E2 → E4	(40, 20)	(41, 21)	1.41	Y	Slight offset
2	G1 → F3	(60, 80)	(60, 79)	1.00	Y	Good placement
3	B1 → C3	(20, 80)	(23, 82)	3.61	N	Needs angle tuning
4	D2 → D4	(30, 20)	(31, 19)	1.41	Y	Minor error
5	H7 → H5	(70, 10)	(69, 9)	1.41	Y	Within tolerance
6	E7 → E5	(40, 10)	(40, 12)	2.00	Y	Slight overshoot
7	F1 → C4	(25, 70)	(26, 69)	1.41	Y	Acceptable accuracy
8	D1 → H5	(70, 70)	(68, 71)	2.24	Y	Slight drift
9	H2 → H4	(70, 20)	(70, 20)	0.00	Y	Perfect
10	G8 → F6	(50, 90)	(52, 88)	2.83	N	Elbow servo error

Table 1: Precision Table (distance in mm)

Gameplay Simulation:

It was tested on simulated games against human opponents using the Stockfish chess engine.

The robotic arm successfully played a complete game of chess with no hardware or software failure.

Stockfish moves were played smoothly with an average move completion time of 10-15 seconds.

System Stability:

The servo motors performed continuously over prolonged durations of time, providing uniform torque and speed.

The gripper mechanism, controlled by the MG90 servo, lifted and released chess pieces safely without loss or damage.

User Interaction:

The system needed input for opponent moves, and this was transferred appropriately through the serial monitor.

Error-handling procedures were put in place to handle input that was in error or under abnormal conditions, e.g., trying to make moves with non existent pieces.

VI. CONCLUSION

This study is a demonstration of an affordable, visionless robotic chess-playing system solution by using the Stockfish chess engine along with inverse kinematics and servo-based control. The system is highly accurate and reliable with the removal of costly vision systems. Inverse kinematics enables the robotic arm to determine joint angles for precise piece placement on the chessboard to enable smooth interaction with the game.

The project demonstrates the viability of integrating software intelligence into mechanical systems for interactive devices, providing the basis for future research in recreational and educational robotics. Through the use of inexpensive hardware, robotic systems are made available to a broader audience, ranging from schools and hobbyists.

Overall, this research paper provides a key to making greater progress in robotics so that low-cost effective learning and playing systems can be developed.

VII. REFERENCES

[1] **Pichai, K.** (2023). A Retrieval-Augmented Generation Based Large Language Model Benchmarked on a Novel Dataset. Journal of Student Research, Volume 12, Issue 4.

[2] **Gobet, F., & Lane, P. C. R.** (2012). Chunking mechanisms and learning. In N. M. Seel (Ed.), Encyclopedia of the Sciences of Learning. New York, NY: Springer.

[3] **Webster, J. J., & Kit, C.** (1992). Tokenization as the Initial Phase in NLP. Proceedings of the COLING-92: International Conference on Computational Linguistics, Nantes, France, August 23-28, 1992. Association for Computational Linguistics. doi: 10.3115/992424.992434.

[4] Gongde Guo, Hui Wang, Yaxin Bi, David A. Bell, KNN Model-Based Approach in Classification, August 2004, https://www.researchgate.net/publication/2948052.

[5] Kovan Mzwri, Turcsányi-Szabó Márta, Chatbot Development using APIs and Integration into the MOOC, Central-European Journal of New Technologies in Research Education and Practice, DOI:10.36427/CEJNTREP.5.1.5041