Autonomous Chess Playing Robot

Prabin Kumar Rath, (prabinrath123@gmail.com), Neelam Mahapatro (neelammahapatro36@gmail.com), Prasanmit Nath (prasanmit.nath@gmail.com), Ratnakar Dash (ratnakar@nitrkl.ac.in) National Institute of Technology, Rourkela, 769008, Odisha, India.

***Abstract*— Chess is an ancient strategy board game that is played on an 8x8 board. Although digital games have become attractive today, chess still retains its popularity in the onscreen version of the game. There has also been considerable development in the chess game engines to play against a human counterpart. The objective of this work is to integrate these chess engines with an actual board game experience and create an autonomous chess player. The system is designed around the use of an open source chess engine and a computer numeric control (CNC) controlled magnetic moving mechanism for moving around the chess pieces. The moves from the human counterpart are taken through an overhead computer vision system. The robot makes the game much more interactive and builds a link between the human and computer system.**

Keywords*— Chess engine, ROS, Arduino, Computer vision, CNC control.

I. Introduction

We live today in an increasingly digitized world and technology has developed tremendously due to which computer chess games and online chess hubs have become very popular. Unfortunately, they do not give the feeling of playing a real game of chess with chess board and pieces. A lot of mobile and desktop applications are available that allow users to play chess with a computer through an user interface (UI) but they do not consider the interactive domain of the game.

Motivation behind the development of an autonomous chess robot comes from the goals that can be achieved with it. The robot as an industrial product can help in conducting professional chess matches remotely from two different places. Chess being one of the most popular strategy games requires tremendous practice and training. Chess training centers around the globe can use these robots for giving remote training sessions to children from experts around the world.

Automation of the game of chess has been previously attempted through different robotic models. The most common approaches used for handling the autonomous movements of chess pieces is using a mobile manipulator[1]. The design and development of such a manipulator which can carry out accurate and precise onboard movements is difficult and expensive. Use of passive electronic sensors such as reed switches and hall sensors have been used previously by Sušac and Filip [2] to keep track of the piece positions on the physical chess board but they are fragile magnetic switches which are susceptible to permanent damage in case of any mechanical jerk on the robot.

The objective of this work was to create a simple and durable automated chess playing robot that will be able to interact with a human chess player and play full-length matches. It uses available chess engines for predicting best possible moves. The engine used in this work is an open source chess engine called Stockfish [3]. A CNC [4] controlled XY slider mechanism is designed to move the chess pieces around the board. It utilizes an electromagnetic head to latch on to magnetic bottom ends of the chess pieces and drag the pieces. The system uses an overhead camera to track the moves made by the user. When a user makes a move on the board, the current locations of the pieces are fed to the system after being detected by the camera. Stockfish gives the best move according to the resulting board configuration. The move performed by the Stockfish is carried out physically by the CNC slider such that no two pieces collide with each other during the movements. This gives a very real experience of chess in a more fascinating and interactive way.

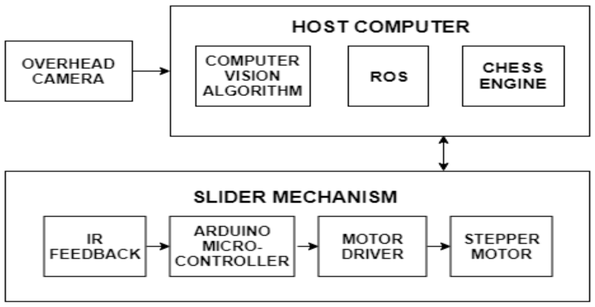


Fig. 1: System block diagram

In this work, the focus is on sensing of the chessboard through camera feed and the autonomous movement of the chess pieces. The purpose of this work was to build a working prototype and establish a link between a human player and computer chess engines in an interactive manner. Besides that, software stack developed for the robot contains add on features of virtual chess games such as load saved game in Portable Game Notation (PGN) [5] format, historic games etc. to make chess more interesting. The biggest challenge was to create a fully functional and working interface between the physical world and the available chess engines and make its mechanical unit work in sync. The basic system architecture of the proposed system can be seen in Fig. 1. As an additional feature, a voice command gaming setup was also developed using the method described by P. Nath [6] using a mobile android app and Bluetooth connection. This allowed the system to take inputs from the user for the required moves and use the setup as shown in wizard chess from Harry Potter movie series.

II. System Architecture

A. Slider Mechanism

The mechanical unit of the robot primarily contains a three-layer design for implementing an XY slider that can maneuver an electromagnet to each and every position of the chess board. Two railings are used in the Y direction and one in the X direction. Linear gear along each axis and motors with circular gears, mate to drive the sliders along both the axis. The chess pieces have small neodymium magnets attached to their base for coupling with the electromagnet. Fluorescent yellow papers are attached on top of the pieces for detection using image processing. The physical board is realized with a flex printed with the chess board attached on a glass plate.

Position accuracy is achieved by implementing IR feedbacks from both the axis of motion. Slider displacement is quantized by using black and white strips of width 3mm. The IR circuit counts the strips and moves according to the demanded count. A Low Pass Filters (LPF) has been used to filter out the noise in the feedback due to mechanical vibrations. This feedback signal is further processed with a time-based filter in the microcontroller to get accurate position feedbacks. This setup is can be seen in Fig 2.

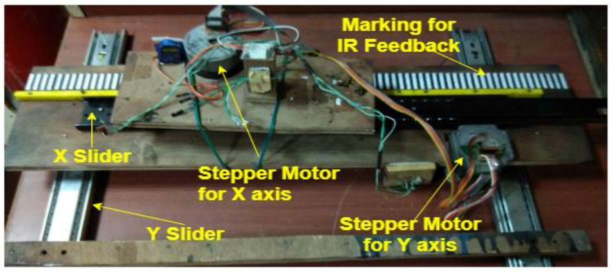


Fig. 2: Slider Mechanism

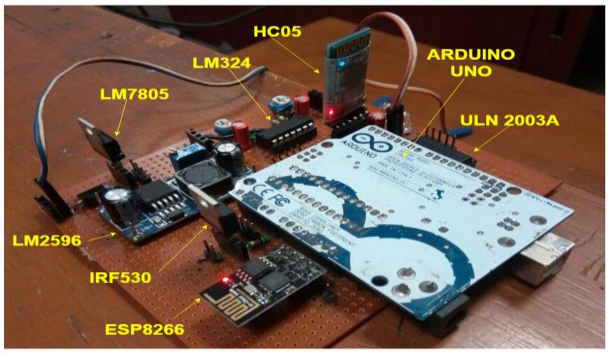


Fig. 3: Control Circuit

The use of LPF for filtering the interrupt signals is adapted from the method described by Alain and René in their paper [7] on DC voltage filtering. The time-based filter simply maintains a minimum threshold time between two successive interrupts which are considered as true positive feedback.

The circuit for driving the unit consists of an Arduino[8] UNO R3 microcontroller, ULN2003A [9] as the motor driver, LM324[10] with HPF and IR pairs for position feedbacks, ESP8266-01 [11] and HC-05 [12] for wireless communication. The unit can be controlled from web as well from Bluetooth enabled devices remotely, for giving the user hustle free playing experience. Fig 3 shows the final control circuit.

B. Vision system

The user's physical moves are fed to the system using a Logitech C270 720p [13] overhead camera which takes real-time images of the board at a rate of 40 Frames per Second (FPS). Three copies of each frame are made for three different tasks consisting up color segmentation for chess piece detection, edge detection for square center tracking and hand detection for process control. The acquired square centers are mapped on the binary frame generated after piece detection to produce an 8X8 binary integer matrix representing the occupancy of each square in the image of the chess board. The generated matrix is considered for further processing only if there is no hand detected in the hand detection output, thus preventing any ambiguous input to the system. The setup of the overhead camera fitted on a stilt can be seen in Fig. 4.

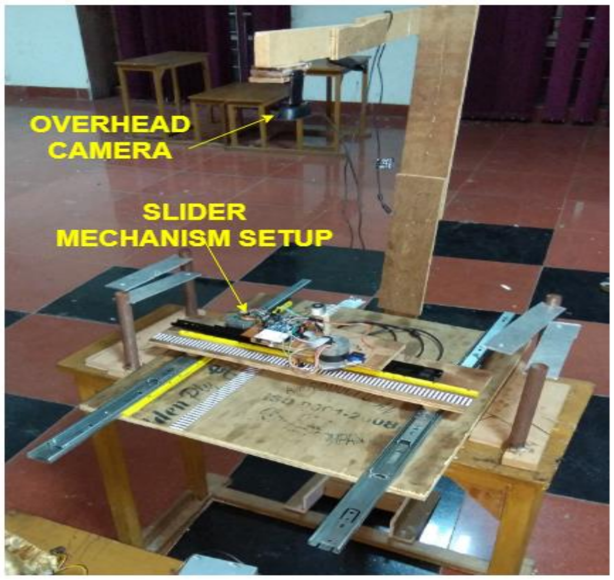


Fig. 4: Complete setup (without glass board) of the robotic chess player

Irregular lighting conditions in an indoor environment makes edge detection difficult thus creating problems in accurate square center detection. To solve this issue the tracked square coordinates are latched to a buffer and gets updated each time when all 32 white squares are tracked. This is feasible because the board remains static with respect to the camera during the entire game-play.

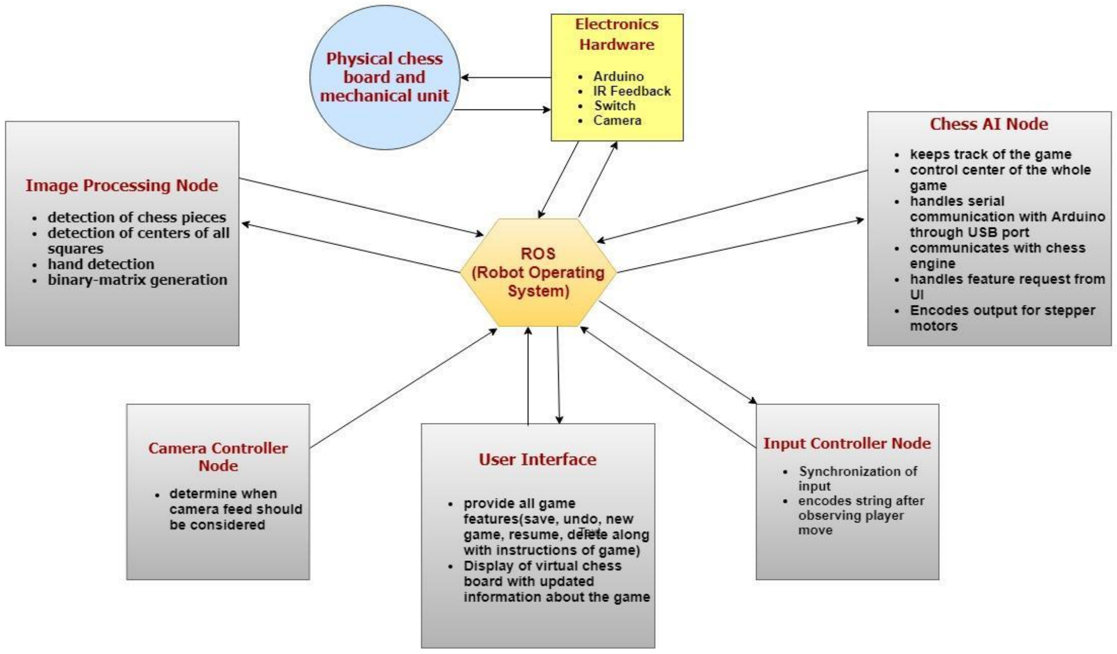

Physical chess board and mechanical unit
Electronics Hardware
Arduino
IR Feedback
Switch
Camera
Chess AI Node
keeps track of the game
control center of the whole game
handles serial communication with Arduino through USB port
communicates with chess engine
handles feature request from UI
Encodes output for stepper motors
Image Processing Node
detection of chess pieces
detection of centers of all squares
hand detection
binary-matrix generation
ROS (Robot Operating System)
Camera Controller Node
determine when camera feed should be considered
User Interface
provide all game features(save, undo, new game, resume, delete along with instructions of game)
Display of virtual chess board with updated information about the game
Input Controller Node
Synchronization of input
encodes string after observing player move

Fig. 5: ROS node map showing data flow between different modules

These 32 white squares can then map to the rest 32 black squares and thus track the entire chess board in every frame. The vision algorithms are developed with the OpenCV [14] libraries in C++.

C. Control system and chess engine

The software for the robot is powered with a chess engine which processes the user input and gives the corresponding best move as the output. Currently, the open-source Stockfish engine has been used by default which is also the World's best chess engine. The 8X8 matrix is processed to determine the onboard position changes to generate the user input in the form of a string according to the Universal Chess Interface (UCI)[15] format which is recognized by all the chess engines globally. Once the engine gives its output as another UCI string, it is encoded into a move sequence string which is sent to the microcontroller through serial communication. The microcontroller then reads this string to carry out on board changes with the XY slider.

The user is assisted with an interactive UI during the entire game-play to keep him/her updated with the game which provides all the virtual game features including save, restart, load, undo and delete options. This makes the game more user-friendly and interesting as it includes both real and virtual features of the chess game. The control system is developed with the Robot Operating System (ROS) framework in C++ and Python assisted with UI (User Interface) developed with Qt in C++. The interactions with the chess engines and game state tracking are handled by python chess module. Serial communication with the microcontroller is achieved with pyserial python module[16]. The interaction between individual nodes in ROS can be visualized with the node map in Fig. 5.

III. IMAGING AND TRACKING

Chess board pattern detection methods are based on corner detection algorithms. These have two basic drawbacks.

- The number of false positive outputs increase with increasing noise in the captured image.
- Outliers detected are difficult to segregate from the correct set of points.

The algorithm used in this work for center detection of the chess squares requires further filtering of the output to get the approximated position of the chess pieces. The algorithm assumes static configuration between camera and the chess board in real-time, it is a simple yet effective approach to solve the problem of chess board square center detection. This algorithm uses a latched vector approach for consistent center detection based on HSV image segmentation and canny edge detection. The robustness was tested by using prints of the chess board for giving tough detection challenge to the algorithm. The algorithm works in 6 steps.

A. Cropping the Image

The image obtained is cropped to get the Region of Interest (ROI) that is the chess board. This removes unnecessary noise from the surroundings during detection of the centers.

B. Cloning the frame

The obtained cropped frame is cloned thrice to make a total of four frames. Each frame is used for a different purpose in the subsequent steps.

C. *Chess piece position detection*

A cloned RGB frame is converted into a corresponding HSV frame. This frame is threshed with respect to the color of the yellow markers attached on the top of the pieces, to obtain a binary image highlighting the position of the pieces in the frame. The result can be seen in Fig 6.

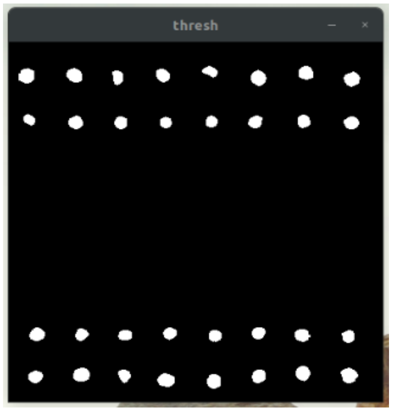


Fig 6: Threshed image for chess piece position detection

D. *Detection of the square Centres*

A cloned RGB frame is converted into a grayscale image. This frame is then blurred by Gaussian Blur convolution with a kernel size of 7X7. The blurred frame is then processed with the canny edge detection algorithm [17] to get the contours present in the image. This can be seen in Fig. 7. The obtained contours are filtered with respect to their shape and area to get the contours corresponding to the white squares on the chess board image. Each of the obtained contours is processed with respect to their moments to get the center of area, which in the real world would correspond to the center of the white square on the board. The center coordinate vector (32X2) obtained is not in an accessible format as the contours are randomly detected during the contour detection process. Hence this vector is sorted in ascending order according to the sorting criteria given below.

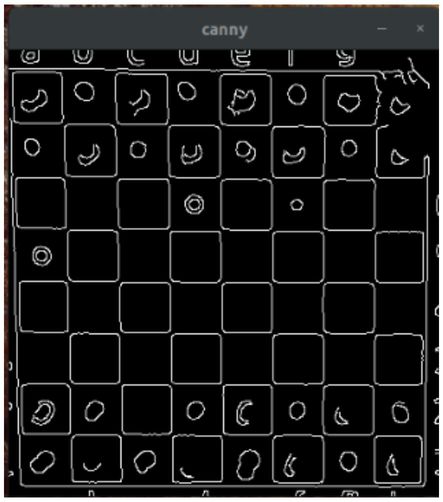


Fig.7: Canny Edge Detected output

A vector of 32 points having each point represented as (Xi, Yi). First, we sort the vector with respect to the Y coordinates. Then we divide the 32 element vector into 8 equal parts. Each part is sorted with respect to X coordinates. The algorithm is illustrated in Fig. 8. This converts the vector into a systematic format for accessing the position of a particular square on the image of the chess board.

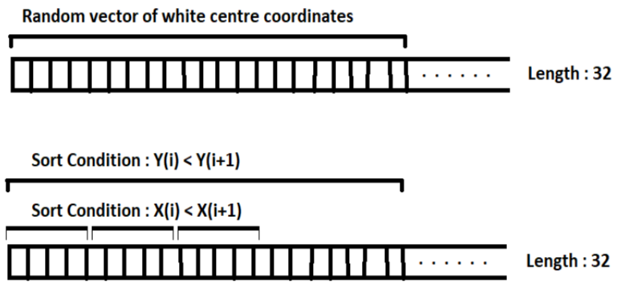

Random vector of white centre coordinates
Length : 32
Sort Condition : Y(i) < Y(i+1)
Sort Condition : X(i) < X(i+1)
Length : 32

Fig. 8: Sorting Algorithm pictorial representation

The black square centers are mapped with respect to their corresponding white square. This mapping is done with a pixel offset which depends on the vertical distance of the camera from the chess board. Fig. 9 represents the above mapping technique. As the vertical distance of the camera increases the offset value decreases and parallax error reduces. With this step, we have obtained the coordinates of all the 64 square centers present on the board image. The vector of 64 coordinates is latched in the memory for reference until a new set of coordinates is detected by the process. This step is essential because edge detection algorithms are sensitive to lighting conditions of the environment. As the position of the camera remains fixed with respect to the board throughout the game, the latched coordinates can be assumed to be reliable for referencing the center of the squares.

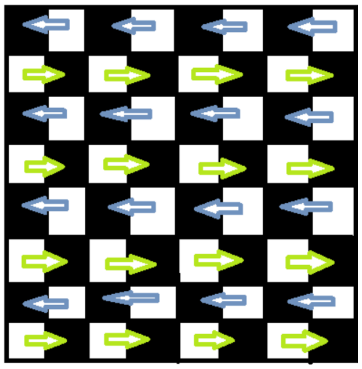
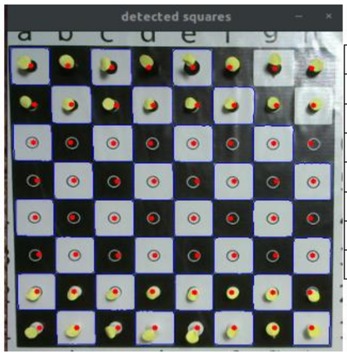


Fig 9: Mapping of the black squares from the detected white squares

E. *Generating the binary position array*

The obtained coordinate vector is accessed to get the coordinates of each square. The HSV threshed frame obtained in Step-C has the same dimensions as that of the frame processed in Step-D. Hence a white pixel cluster in the HSV threshed frame at a coordinate obtained from the vector will represent the presence of a piece on the real board corresponding a specific square. This method when applied to all the 64 coordinates in an iterative manner gives the board configuration in the form of a 2D binary array of 1s and 0s. 1 correspond to a position being occupied by a chess piece and 0 corresponds to an unoccupied position. Fig. 10 shows the binary matrix corresponding to a board configuration at the start of the game.



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

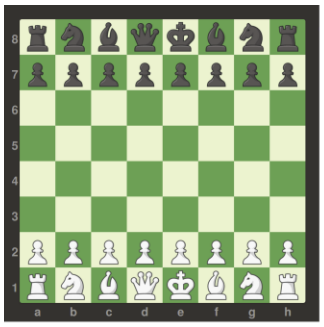
Fig 10: Tracked coordinates of all 64 squares and Binary matrix representation of the initial chess board configuration

F. *Hand Detection*

The hand detection is a simple procedure for preventing any input errors to the system during the user's move. The user is asked to wear a fluorescent color glove while playing the game. These gloves are easy to detect in the image using the same HSV thresholding methods as described in step-C. The binary image obtained after thresholding is checked pixel by pixel to determine the presence of the gloves in the frame in real-time. The computation of the binary matrix for piece position detection is done only in the absence of the gloves thus preventing any possible input errors.

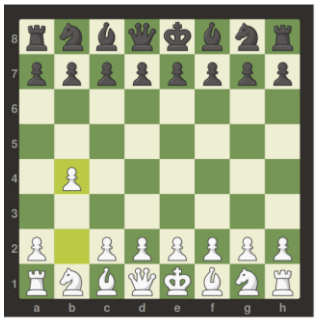
IV. Chess Move Tracking

The onboard moves are tracked by the overhead camera using the computer vision algorithm described in the previous section to generate binary matrices at the frame sampling frequency. About 15 FPS(frames per second) is required in order to track all the intermediate changes happening on board. The computer maintains a current instance of the binary matrix in the memory. Once a new matrix arrives from the next frame it is compared with the current matrix to determine any intermediate change in piece positions. After completing a move human player confirms it by pressing a physical switch or through UI. The tracked intermediate moves are processed to determine the coordinate changes and the type of the move.



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Fig 11: Chess board and binary matrix configuration before b2 to b4 move



1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Fig 12: Chess board and binary matrix configuration after b2 to b4 move

For the move shown in fig 11. and fig. 12, the algorithm will produce intermediate step string "b2_10,b4_01". b2 and b4 are the affected coordinates. 10 is an occupancy pair which represents that a piece is present in the current frame and is absent in the next frame, similarly 01 represents vice versa. The occupancy pairs associated with each intermediate move are appended to generate a move trace. Move trace for the above move is 1001. This binary representation of a move is unique to a move type in the game.

Similarly, a killing move will involve three occupancy pairs, 01 for the killed piece being removed from the board, 01 for the killing piece leaving its position and 10 for the killing piece occupying the killed piece position. So the resulting move trace is 010110 which is unique for a killing move in the game. Different types of moves with their move trace and associated conditions to determine type of the move is shown in Table 1.

TABLE I. TABLE OF MOVES

Types of Moves in Chess	**Move trace on addition condition for type recognition**
Normal	1001
Killing	10101
Castling	10011001 with 2 pieces as king and rook at 1st or 8th row
Respawn	1001 A pawn at its opponent's end row
En passant	101001

V. Output Move Sequence Generation

Once the move and its type is recognized, it is fed to the engine. The output of the engine is in UCI format and is the best move corresponding to the current board configuration. The associated coordinates and the type of move is determined from the engine's output to generate a character string encoding the required tasks to be carried out by the XY slider to realize the move physically on the board.

The electromagnet stays at a reference position (below the d5 coordinate) before the control circuit receives any move command from the computer and returns to this reference after every move. The computer calculates the path as a sequence of steps that has to be carried out by X and Y stepper motors to move the pieces from one position to another. The command string generated by the computer is sent to the microcontroller via serial communication. It has three basic information in a sequential format.

Format: (x/y) (numeric value) (0/1) (H/L)

- The motor to use: (x/y)
- Number of steps to count: (numeric value)
- Direction of rotation CW/CCW: (0/1)
- Keep electromagnet ON/OFF: (H/L)

1 step count is defined as feedback counts required to drag a piece half the square side length on the physical chess board.

Example: - "y10x30y50x10Hx10y41x11Ly11x31y11x11"

The above string encodes the sequence for the move 'b2b4'-Normal. The first 3 characters "y10" of the above example command tells the control unit to move the Y stepper in clock-wise direction until one step count is complete. An occurrence of H/L in the command changes the state of the electromagnet. The path planned by the computer corresponding to the example move is shown in Fig 13.

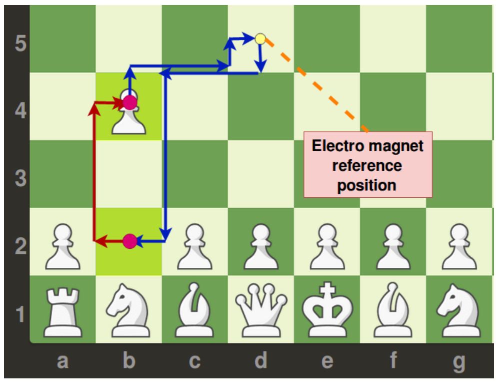


Fig 13: Path planned for the move "b2b4". Blue lines - Electromagnet OFF, Red lines – Electromagnet ON

CONCLUSION

This robot is aimed towards giving the user a realistic experience of playing chess with a computer. The challenge was to create a fully functional and working interface between the physical world and the engine. The robot was tested in different indoor environments with variations in lighting conditions and the results of the position tracking were robust and consistent. The robot could play full length matches with more than 50 moves independently. The position accuracy that could be achieved with the feedback mechanism in the slider was up to 3mm. Cumulative error after each move was eradicated using regular offsets at the reference position.

This system can be expanded to include Wi-Fi/LAN capabilities for enabling remote matches. Further improvements need to be done in the hand detection algorithm and using machine learning algorithms will be a good option to enable the usage in different lighting conditions.

REFERENCES

[1] Golz, Jens, and Rolf Biesenbach. "Implementation of an autonomous chess playing industrial robot." In *Proc. of 2015 16th International Conference on Research and Education in Mechatronics (REM)*, pp. 53-56. IEEE, 2015.

[2] Sušac, Filip, Ivan Aleksi, and Željko Hocenski. "Digital chess board based on array of Hall-Effect sensors." In *Proc. of 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1011-1014. IEEE, 2017.

[3] Golz, Jens, and Rolf Biesenbach. "Implementation of an autonomous chess playing industrial robot." In *Proc. of 16th International Conference on Research and Education in Mechatronics (REM)*, pp. 53-56. IEEE, 2015.

[4] Yan, Wen, Tan Ji-wen, Zhan Hong, and Sun Xian-bin. "Fault Diagnosis Model Based on Multi-level Information Fusion for CNC Machine Tools." *International Journal of Hybrid Information Technology 9*, no. 8 (2016): 367-376..

[5] PGN(Portable Game Notation), chess game recording format [online] available:https://en.Wikipedia.org/wiki/Portable_Game_Notation

[6] P.Nath and U. C. Pati. "Low-cost android app based voice operated room automation system." In *Proc. 0f 3rd International Conference for Convergence in Technology (I2CT)*, IEEE, 2018.

[7] Tessier, A., Le Sage, R. and Leblanc, R.M, in *Applied Spectroscopy* "Simple low-pass filter to reduce noise in direct current signals: application to the fluorescence of chlorophyll a at low concentration.," 34(2), pp.226-227, 1980.

[8] Shudhanshu, A. Avinash Kumar, B. Amit Garg, C. Raj Kumar, D. Sadashiv Raj Bharadwaj, and E. Gaurav Singh. "Innovation in Basic Science Laboratory through Sensors and Open Source Technology." In *International Journal on Recent Trends in Engineering & Technology 11*, no. 1 (2014): 313.

[9] Saha, Somnath, Tushar Tyagi, and Dhananjay V. Gadre. "ARM (R) Microcontroller Based Automatic Power Factor Monitoring and Control System." In *Proc. of Texas Instruments India Educators' Conference*, pp. 165-170. IEEE, 2013.

[10] Lopez-Calle, I., F. J. Franco, J. G. Izquierdo, and J. A. Agapito. "Two-Photon Absorption (TPA) backside pulsed laser tests in the LM324." In *Proc. of European Conference on Radiation and Its Effects on Components and Systems,* pp. 138-143. IEEE, 2009.

[11] Chandramohan, J., R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. A. Gopinath, and S. Ranjithkumar. "Intelligent smart home automation and security system using Arduino and Wi-fi." In *International Journal of Engineering And Computer Science (IJECS) 6*, no. 3 (2017): 20694-20698..

[12] Cotta, Anisha, Naik Trupti Devidas, and Varda Kalidas Naik Ekoskar. "Wireless Communication Using HC-05 Bluetooth Module Interfaced With Arduino." *In International Journal of Science, Engineering and Technology Research (IJSETR) 5*, no. 4 (2016).

[13] Jay Geater (2014), Logitech - C270 HD Webcam. [online] available: https://www.logitech.com/en-in/product/hd-webcam-c270.

[14] Culjak, Ivan, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. "A brief introduction to OpenCV." In *Proc. of the 35th international convention MIPRO,* pp. 1725-1730. IEEE, 2012.

[15] Rudolf Huber and Stefan Meyer-Kahlen (November 2000), [online] available: https://en.wikipedia.org/wiki/Universal_Chess_Interface

[16] Goradiya, B. H. A. R. G. A. V., and H. N. Pandya. "Real time Monitoring & Data logging System using ARM architecture of Raspberry pi & Ardiuno UNO." In *International Journal of VLSI and Embedded Systems-IJVES 4*, no. 06118 (2013): 513-517.

[17] Hocenski, Zeljko, Suzana Vasilic, and Verica Hocenski. "Improved canny edge detector in ceramic tiles defect detection." In *Proc. of 32nd Annual Conference on IEEE Industrial Electronics (IECON)*, pp. 3328-3331. IEEE, 2006.