AUTOMATIC CHESS BOT

Project- ROBOREX\_CHESS\_MATE

by Team ROBOREX

Team Members:

* Prabin Kumar Rath
* Neelam Mahapatro
* Subhum Sahoo
* Pabitra Dash
* Swostik Acharya

***INTRODUCTION: -***

Chess is an old game which is very popular. But since we today live in an increasingly digitalised world and the technology has developed tremendously, most of the chess games are nowadays played via a computer. It’s a pity because it takes away the feeling of playing a real game of chess

The objective of this project was to create an automated chess bot that could be able to interact with a human chess player using one of the most challenging chess engine STOCKFISH. The game began either with the user’s move or the stockfish’s according to the willingness of user. When user made a move on the board, the current locations of the pieces were fed into the board handler node after being detected by the camera embedded in the system. Stockfish gives a best move according to the user’s pass. The move performed by the stockfish was shown physically on the chess board using some technique such that no two pieces collide with each other during the movement. Electromagnet were used to attract the piece, move along with it as the direction given to stepper motor and reached the final position as instructed. User interface had been made to interact with the player and to give the player a real happiness of playing chess with a robot in a more fascinating and interactive way.

In this project the focus was on sensing of the chessboard through camera feed and the autonomous movement of the chess pieces flawlessly. The purpose of this project was to build a working prototype through some studies and establish a link between human player and computer’s artificial intelligence in an interactive manner. Besides that, this game contains add on features of a real-time game (New Game, Load Saved Game, Instructions, Credits, Delete, Restart, Undo) to make Chess game more interactive and interesting.

***GENERAL HARDWARE DESCRIPTION: -***

***Requirements: -***

1. 25’’ x 25’’ x 2.05” plywood base board

2. 24’’ x 24” glass plate

3. 17.6’’ x 17.6” flex of the chess board

4. 24” x 24” mica plate

5. 2 Railings on Y direction of length 24.63” and width 0.88”

6. 1 Railing on X direction of length 18.31’’ and width 0.88”

7. Racks of length 24’’

8. Plywood

9. 2 Stepper motors

10. Pinion of radius 1cm

11. Plywood level of 2’’ length and 0.75” width for placing electromagnet

12. Electromagnet

13. Camera to capture image of chess board

14. black white strip for IR Feedback

15. 4 C pillars made of iron and steel

16. 32 chess pieces

17. Switch for the detection of completion of user move

18. Photodiode, High pass filter, IR LED for IR feedback

19. Fluorescent paper

20. C clamp stand of wood for fixing camera

At first, a baseboard was made of plywood keeping the dimension 25”x25”x2.05” which played the role of a base for the whole mechanical system. Two parallel Y railings were placed above the base which were of length 24.63” and width 0.88” on an aim for the safe movement of stepper motors in perpendicular direction. A rack of length 24” made of plastic placed near one of these railings and was fixed with the base board for safer movement of y-stepper motor.

Above it, slider made of plywood of 24” and width 5.91” was settled in such a manner that it was exact perpendicular to the Y railings. One of the stepper motors was mounted on the railings through a pinion of radius 1 cm by cutting the upper wood stage accurately. Above that wood stage, another Railing in x direction was constructed. Another rack of length 18.31” formed of plastic placed near the x railing and was fixed with the slider on an aim for the exact movement of x stepper motor. Another plywood stage of dimension 9.84”x3.94”x0.38’’ was created in a shape for the mounting of another stepper motor over a mounting plate.

The stepper motor was mounted on the slider via a mounting plate such that the gear of stepper motor mate with linear gear. X stepper motor was placed for any movement of chess piece over X direction and similarly Y stepper motor was used for any movement of chess piece over Y direction whereas the movement steps were determined by our code and finally fed to the motors.

Another wood level was constructed over the slider for placing the electromagnet. The dimension of this wood level was 2” height, 0.75”x0.75” area of cross section. Over this wood level, the electromagnet was mounted. Thus, the motion of the electromagnet was guided by the motion of the stepper motor. To avoid any mechanical error, we had used two IR feedback system each of which consisted of a circuit containing photodiode, IR led wrapped with black tape and high pass filter placed with a support of small wood piece. This feedback system played a vital role in positioning the magnet in accurate position of the chess board. The black white strip to which the IR feedback is sensible was attached to the wood such that the IR system can sense it. We used two IR feedback circuits on a purpose to robust the motion of electromagnet in both perpendicular direction.

A square glass plate of each side 24” was established, on which the printed chess flex was attached. The flex had a dimension of 26.6”x26.6” such that each square box out of 64 boxes had a dimension of 2.20”x2.20”. Below it, mica plate of 24”x24” was assembled. Then the mica plate and the wood base board were connected by 4 C pillars each of which had length 8.05”, width 2.79”. The materials for C pillars that we chose were iron and aluminium just to make the whole system stable. 32 Chess pieces with fluorescent paper attached above each of them and electromagnets attached to the base of every chess pieces were placed on the glass plate by observing the position in the flex.

A camera was fixed through a C-clamp at a sufficient height from the board such that it could be able to take a flawless image of the chess board.

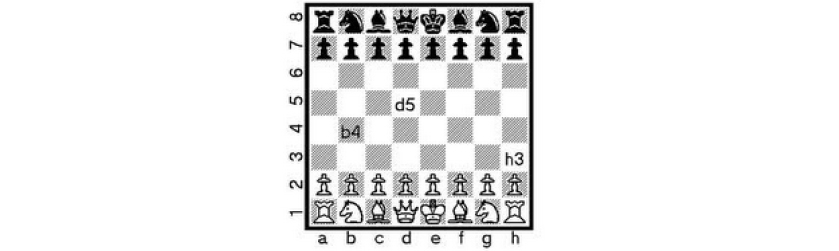


**Our** **Chess Bot View**

***SOFTWARE DESCRIPTION: -***

Before going to the software part of the project, one should have a little idea on Arduino, ROS as this project based on robotics. Basically, Arduino is an open source platform used for building electronics projects. Arduino consists of both a physical; programmable circuit board often referred to as a microcontroller and a piece of software or integrated development environment that runs on computer, used to write and upload computer code to the physical board. Arduino does not need a separate piece of hardware in order to load a new code on to the board it simply needs a USB cable. ROS stands for Robot Operating System which is flexible framework for writing robot software. ROS is a collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotics platforms. ROS is multilingual i.e. multiple language is understandable by ROS and thus ROS act as a way of communication between different languages. ROS helps in establishment of communication between two or multiple file each of which is written in same or different programming language. So, here the message from one node is transferred to the other by a default or customised message file through the ROS. Thus, ROS plays essential role in the intercommunication of different nodes. The nodes we used basically in our projects here are chess ai(Python), board handler(Python), frame pub(CPP), frame handler(CPP), chess ui(CPP), chess ui fen, Ardsimul(Python), Arduino.

A general chess board is labelled in the diagram below. The position is recognised by the computer algorithm according to this conversion as shown here.



Game was started from **chess ui** node. At the starting of game, a well framed user interface was displayed whose Menu bar had the contents New game, Saved Game List, Game Record, Instruction, Credits. The tab Credits contained the name of the tech nerds responsible for the project. On clicking the Saved Game List, all the lists of the game those had been played before was displayed. User could have delete any particular game which he played before or could resume the game from where he left. Before beginning of game, the user was advised to refer the Instruction shown in instruction tab. On clicking the New Game option, there was a facility to enter the name of event, name of white side player, name of black side player and the number of round. The player could choose white side or black side according to his/her willingness. After confirming these data, the player was directed to another UI where the turn of each side player was represented by a dot (Stockfish or human player). Then user was allowed to play with the Stockfish.

The movement accomplished by the user was detected by camera which captured the image of the whole chess board including the pieces. **Frame pub node** accessed that camera image which was in RGB format by OpenCV technology in Mat format and encoded the image to a sensor message and published that image to frame handler node through ROS.

**Frame handler node**, after obtaining the message, decoded it get image OpenCV Mat format which was a RGB format image. This code had the function to crop the image separating all the disturbances outside the chess board those were captured by the camera. Another function was called to convert the obtained cropped image to HSV format. The HSV image was converted to binary image (suppose image 1) by OpenCV inverse threshing and In Range method to detect the fluorescent yellow paper attached to the head of each chess pieces. Another copy of cropped RGB image was also converted to grayscale image by inverse threshing. The grayscale image was blurred for a purpose to apply an OpenCV technology Canny edge detection to find edges of white squares. After finding the edge of white squares, mathematical operations were performed to find the centres of all these 32 white squares and those coordinates were sorted. Applying a technical operation all 32 centres of black squares were also detected. After this operation we had an image (image 2) which contained all the centres of 64 squares. Image 1 and image 2 were combined to acquire all the positions where chess pieces were presented on board. After formulating all the positions, it was encoded into a string format and was subscribed to board handler node through message file.

The published binary string from frame handler node was received by the **board handler** node. After confirmation of the user move (confirmed by player when he pressed the switch), the board handler node had functionality to generate a new string by comparing the changes in the two binary strings out of which one was published by frame handler node recently and the another one was published by the ardsimul node (board condition after the movement of Stockfish, for the 1st move it is the initial matrix). Example: - For b5 to d5 move the string would be like (b5\_10, d5\_01). This string was subscribed to the chess ai node.

Diagram: -

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | h |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | g |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | f |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | e |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | c |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | b |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | a |

1 2 3 4 5 6 7 8

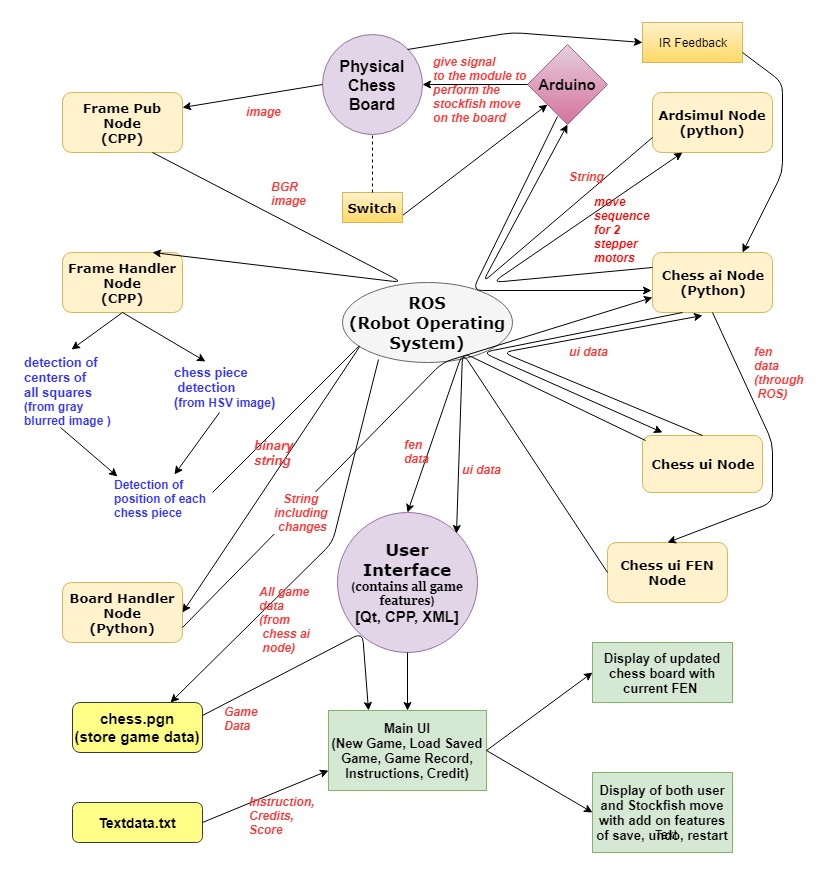
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | h |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | g |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | f |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | e |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | d |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | c |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | b |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | a |

1 2 3 4 5 6 7 8

**Chess ai** node acquired the string from board handler node and converted it into UCI form through an in-built function. UCI is Universal Chess Interface which is a protocol for communicating with the chess engines, here to communicate with stockfish. The UCI form of the above move is simply b5d5. This code had a function to check first whether the move accomplished by the user was a legal move or illegal. If the output came illegal, then a warning message was subscribed to User interface to warn the player about the illegal pass and the user was supposed to correct the illegal move which would be updated in board handler node and now it again came to user’s turn to accomplish a legal move. If the movement came out as legal then in the next step the string(b2d5) was transferred to the ASCII matrix. ASCII matrix is internally stored from which STOCKFISH observed the user’s matrix and generate a best possible move according to the opposite player’s move. The move given by the STOCKFISH was updated in the ASCII matrix and it was also supposed to be updated in the board handler node such that in the next step the user’s pass would be compared with the Stockfish’s. ASCII matrix had the same configuration as that of physical chess board. The ASCII matrix was subscribed to the Ardsimul node which further generated binary string to update the board handler. The current FEN was frequently passed to the user interface and published in which the node **chess ui fen** involved. Basically, User Interface node interacted with the user frequently and flawlessly. From User Interface, we could be able to analyse whether the system was really working and stockfish was playing with the user.

The most important part, the output move given by the stockfish had to be performed physically on the chess board for which we needed to create a XY table of 16X16 matrix, generate a move sequence, and accordingly we had to give electrical signals to the stepper motor for moving the piece on the board with the help of electromagnet. The electromagnet proceeded towards the initial position of targeted chess piece, turned on magnet, picked up the chess piece and placed it in the specified position under the guidance of the stepper motor and IR feedback system.

**Chess ui** node is responsible for all the user interface that we were able to see how the interaction between a player and system was carried out. During the game, every movement was updated in UI. There were also additional functionality like save, undo, delete, loading of saved game in our chess project to have a real feeling of playing chess. All the codes for User Interface were written using **Qt** which is a cross-platform application framework and widget toolkit for creating classic and embedded graphical user interfaces.

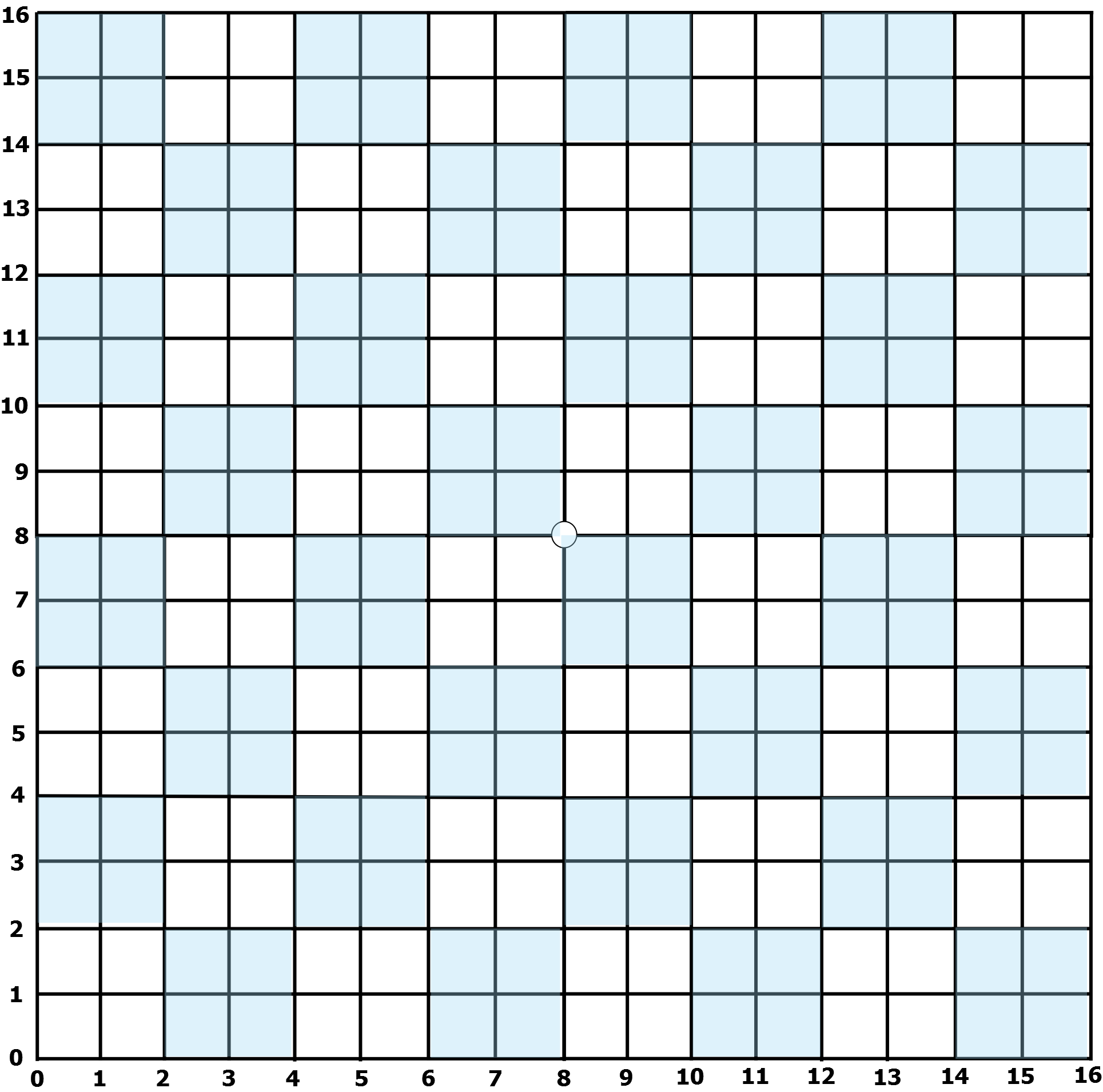


The above diagram shows a sample flow chart how this project functions.

***XY TABLE SOFTWARE DESCRIPTION: -***

It’s generally a 16X16 matrix. The 8X8 matrix was transformed into 16X16 matrix for moving the pieces by the help of electromagnet on the chess board automatically in a flawless manner. So, there was a need of co-ordinate conversion between these 2 matrices. We designed the XY magnet system so that it could be able to transport the pieces along the lines in between the squares so that the pieces did not collide. This was another reason why the board squares needed to be a large. XY magnet system needed to have a finer coordinate system so that it could both go to the location of the piece and pass in between other pieces. The following grid shows the coordinate system of the XY magnet system. The coordinates here range from 0 to 16 in both X and Y directions.

The transformed numerical move coordinates would again need to be changed to fit the granularity of the XY system coordinate system. This involved multiplying each coordinate by two and adding 1: (2x+1,2y+1). Thus, if the instructions were to move piece at (6,2) to space (5,2) received, this would correspond to the XY system needing to move the piece at (13,5) to (11, 5).

*16X16 matrix: -* 

The electromagnet used to start from the centre of the board for each Stockfish move, which is positioned at (8,8) in XY System and returned to the same position after performing the particular movement of chess piece. This was performed on an aim to maintain a constant reference point and to avoid any mechanical error that could have possibly generated. After perfection of all mechanical components, we still got some mechanical error in the positioning the piece due to which we decided to work with IR feedback system which literally showed us the output flawlessly.

So, when the origin coordinates were received, we subtract (8,8) from the coordinates. For example, (13,5) – (8,8) = (5,-3). The Y stepper motor then had to shift down five coordinate steps and X stepper motor then had to shift to Left 3 coordinate steps. After the XY magnet positioner arrived to the initial coordinates, the magnet was then flipped up with the stepper motor to attract the piece. Then the current coordinates were subtracted from the next coordinates. For example, (11,5) – (13,5) = (2,0). Thus, the Y stepper motor needed to shift down 2 coordinate steps. The X stepper motor didn’t need to shift in this case. However, to avoid piece collision we wanted to travel along the lines instead of through the squares. For that reason, we used the concept of offset here. Thus, each time we are moving a piece, we first shift up and right 1 coordinate step. So, we’d go from (13,5) to (14,6). Then we would do the (-2,0) shift, thus going from (14,6) to (12,6). Then we would cancel out the offset shift by going down and left one coordinate, thus going from (12,6) to (11,5). The piece then arrived at the destination coordinates.

According to this technology, the move sequence was generated which was used to determine the number of steps needed to be moved by both the stepper motors separately. Our chess ai node had the function to determine these move sequences and accordingly passed to the Arduino to guide the movement of stepper motor.

This then completes the construction of the XY table. The final part of construction on the XY table involved wiring up the stepper motors. A motor driver was needed for each stepper motor.

***IR Feedback: -***

The main purpose of using IR Feedback system was to avoid the mechanical error generated while positioning the electromagnet as per the move produced by stockfish for a certain chess piece. We had used LM 324, a quad opamp for our feedback system to work flawlessly. This system based on the working principle of photodiode, IR led. The strip we used consisted of alternative black and white strip each of 3mm and a total of 164 strips placed near both the X and Y slider. When IR light falls on the white surface, output voltage of the circuit rises to 5v and this voltage falls to 5v due to the functionality of photodiode.

An interrupt function was constructed to detect the voltage spikes and thus it recorded the position of the electromagnet. More over to say, this system was built to inspect whether the magnet had moved to its specified position or not. If not, a feedback would be given back to the system which would suggest provide further steps to the motor and this feedback would work unless and until the magnet reach its specified position.

***Electronics Description: -***

***Requirements: -***

1. Arduino UNO

2. ULN 2003 - 2no.

3. N-channel MOSFET

4. LM 324

5. Red LED

6. Grun LED

7. IR Pair-2

8. 1uF capacitor-2

9. 0.1uf capacitor-2

10. IN 5408

11. IN 4007

12. LM 2596

13. 330 ohm resistor-4

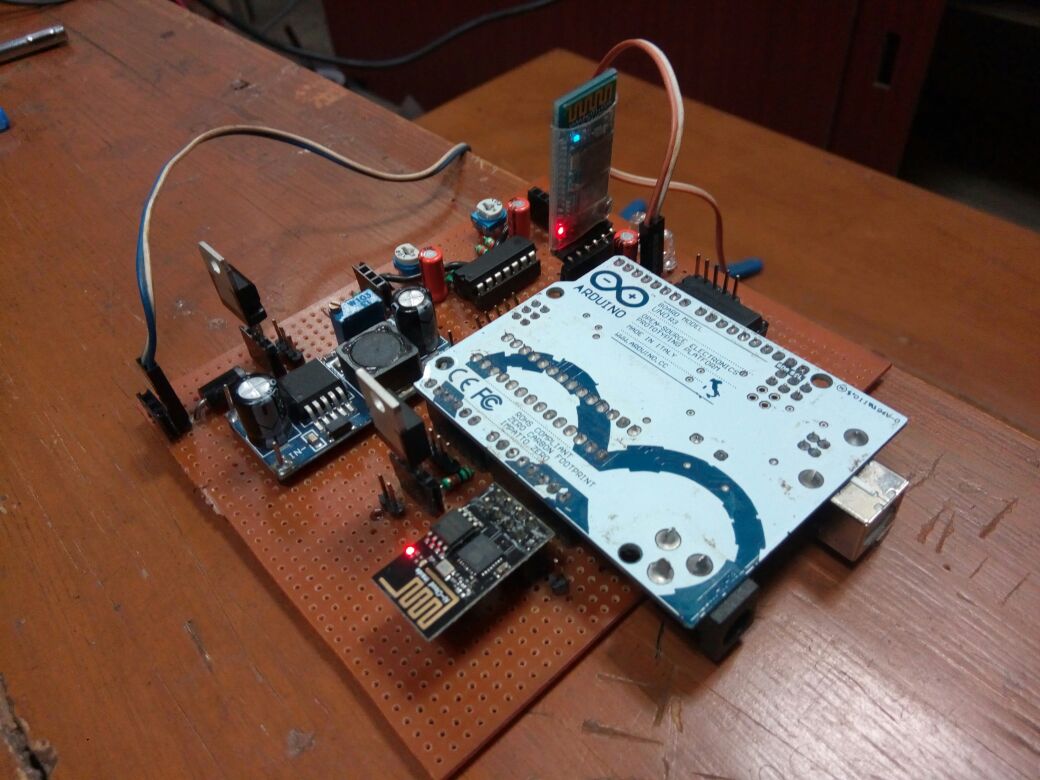
14. 10k ohm resistor-4

15. LM 7805

16. Potentiometer (100k ohm)-2

17. HC05

We supplied 12 volts from SPSN source which was parallelly connected to buck converter, LM 7505, Arduino UNO and the electromagnet. Buck converter provides an output varies from 0 to 12v with a high current. This output was parallelly connected with 2 numbers of ULN 2003A across each of them a capacitor of 0.1uF were connected. The output pins of these ULN2003A provided as input to the stepper motor along with 5volt supply from buck converter. Stepper motor requires a high current to work which was a reason to use Darlington transistor here.



**Final module (Electronics Circuit)**

The output pin of LM7805, a voltage divider provided as input to LM324 and HC05 parallelly which provides an output that is used as input to IR Pair, the feedback system for flawless output. 1uF capacitor and 10k ohm resistor were taken to prepare high pass filter for passing only the low frequency signal. This feedback system consumed a current of 15-20mA that could be provided by LM7805.

The electromagnet was supplied 12 volts because it consumes a current of 300mA. N-MOS connected with 10k ohm resistor in series had been used to act as a switch for the electromagnet. The information when to on or off the switch was obtained from the Arduino, microcontroller.

***Conclusion: -***

All in all, the project has been made to give the user a premium feeling of playing where the dots move by themselves with many tasks going on in the backdrop. The biggest challenge was to create a fully functional and working interfacing between the physical world and the computer’s artificial intelligence. The project really stood up to the mark. We will still be working on the project keeping in mind the new changes that can be appended in it to make it’s working more robust and the further features we can add to it. It was really a great project to work on and the dedication and perseverance shown by the team members really paid off making the project a success.

*Git hub link of our Project: -*

[*https://github.com/prabinrath/Roborex\_Chess*](https://github.com/prabinrath/Roborex_Chess)

***References: -***

1. <https://python-chess.readthedocs.io/en/latest/> We referred this site for gaining idea on Python-chess which is a pure python chess library and then according to our use we applied it.
2. <https://stockfishchess.org/> This is the link to the most powerful python chess engine and we have used this robust chess engine in our project for playing against a human player.
3. <http://wiki.ros.org/> And here is the link to the incredible and most powerful software framework for any Robotics Project i.e. ROS.
4. https://www.dribin.org/dave/keyboard/one\_html/ We used this website for some initial guidance on how to build the XY table. Then we wrote algorithm on our own for proper positioning the chess pieces by guiding properly the two stepper motors through steps.
5. <http://www.instructables.com/> This a very charitable site for building any electronics project. We have studied as per the requirement of project.
6. <https://playground.arduino.cc/> Link to **a wiki where all the users of**[Arduino](http://www.arduino.cc/)**can contribute** and benefit from their collective research.
7. <https://www.qt.io/> This website contains **Qt** which is the faster, smarter way to create innovative devices, modern UIs & applications for multiple screens. We have used Qt Software to build all our user interfaces.
8. <https://opencv.org/> we studied OpenCV and applied for all the image processing codes used in our chess project.
9. <http://python-chess.readthedocs.io/en/latest/pgn.html> We referred to this website and got to know about the PGN files which are used to store the data of a chess game.