Deep RGB

Shenmin Lo, Joe Lunder, Siarhei Traskouski, and Robert Wadsworth II

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — This report has been created to delineate the design process followed in the creation of an automated Chess board with a unique array of capabilities. Once complete the product will allow for users to compete in a game of Chess with either another physical player or to pit themselves against a powerful Chess computer. Should the operator choose to play against the computer or with an opponent who is not physically present the board will utilize a solenoid placed under the board to move the pieces for them. This design results in a truly unique product that provides a particular playing experience that cannot be found in other products.**

***Index Terms* — Automation, Electromagnetics, Hall effect Devices, Light emitting diodes, Microcontrollers, Wireless LAN**

I. INTRODUCTION

Developed to make use of modern technology in order to place a fresh spin on the classic game of Chess, Deep RGB is a complex and challenging electrical and computer engineering design project. Complete and functioning in a portable package, Deep RGB is an automated chess board with a unique combination of design features. Making use of an internal Atmega 2560 to tie the entire project together Deep RGB allows two players to compete in a physical game of Chess even if separated by a great distance. To do this Deep RGB communicates with a server using original code and a wifi connection, once linked with the server the user may update any of his/her games which have been stored on the server. Using a solenoid, which is imbedded in the unit, Deep RGB will update the player's move on their opponent's board allowing for the experience of playing a real game of Chess. This same system is also used to reposition pieces on behalf of the computer. In order to receive player input a Hall Effect sensor array is used to detect and actively track pieces as they are moved around the playing surface. This allows the board to update the chess algorithm so that it may respond appropriately while simultaneously providing an avenue to detect illegal moves. In addition to these technical amenities the project includes an RGB LED array from which it earned its name. This array is used to add a decorative flare to the entirety of the project making it more appealing to the eye, as well as contributing to the scalability of the project allowing for features such as user customization and move assistance.

II. COMPONENTS

The complexity of Deep RGB necessitates many different components to make the entire project function properly. In the following subsections these elements will be analyzed in greater detail than could be afforded in the introduction.

A. Microcontroller

The microcontroller forms one of the most critical building blocks that form the entirety of Deep RGB. It ties everything in the project together from driving the LED array to communicating with the server and delivering the coordinates to the piece positioning system. Therefore it was decided to employ the Atmega 2560 by Atmel for its capacity to manipulate and control all of the other components inside the board. The chip has a 16MHz clock frequency which is necessary to run the LED and Hall Effect sensor arrays as well as communicate with the server via a WIFI module. Both the Arduino IDE and visual studios with the visual micro plug-in were used to program the microcontroller.

B. Movement system

The movement system consists of a set of racks and gears which act as guides for two stepper motors allowing them to move a solenoid in the XY plane. The movement system receives its coordinates from the MCU via a stepper driver module, once it has moved into position it uses the solenoid to reposition the pieces routing them along the lines separating tiles to avoid collisions.

C. Piece Detection System

A1325 linear ratiometric hall-effect sensors are placed in the center of each tile allowing the MCU to determine if a piece has been moved to that tile. This enables the system to actively search for pieces and ensuring that the solenoid is on target when interacting with a piece.

D. Wi-Fi

The communication between the Atmega and the server is taken care of by the WiFly GSX module from Sparkfun, an embedded WLAN device capable of WIFI communication up to 100m in IEEE 802.11b/g standard. The WiFly was chosen for its adherence to the IEEE communication protocols as well as its small footprint and the ability to interface with the MCU using only four pins. Using code tailored to this project the Wi-Fi module will transfer data to and from the server supplying the controller with the necessary data to successfully carry out many of the necessary functions in this project.

E. Server

Custom built to meet the demands of Deep RGB the server's primary function is to house the Chess algorithm so that the MCU may remain free to direct other functions onboard the product. The server has been arrayed with several other functions throughout its creation, serving as a web interface for players who wish to store their game states or desire to play while away from their board. Other features include user profiles/preferences and secured logins.

F. LED Matrix

Under the playing surface an array of RL5-RGB-C-2 common cathode RGB LEDs create the LED matrix. Through pulse-width-modulation the color of the LEDs can be changed to any shade desired and are used to illuminate the tiles that contain a piece allowing for easy distinction between the two sides. To minimize the amount of inputs used to control the array each pin on the LEDs are connected to a corresponding bus which are then routed into a shift register reducing the pin footprint on the MCU to three. This system adds to the scalability of the project allowing for player customization and move assistance in future revisions of the product.

C. LCD

An LCD is used to create a cost effective method of displaying pertinent data to the user. It is not a major part of the project but one necessary in order to supply an indication that the board is on and working.

III. Idealization

This section contains the ideal operating characteristics of the system. It is placed here to give the reader an idea of how the project works as a whole. To begin the flowchart in Fig. 1 shows a simplified flowchart of the project during a game.

As can be seen from the flowchart the system is a loop that runs until there is a checkmate at which point the board is reset.

To begin the user must power on the system and log into the server from there they may choose whether to load a saved game or start a new one. The board then sets the field accordingly, it should be noted that for the board

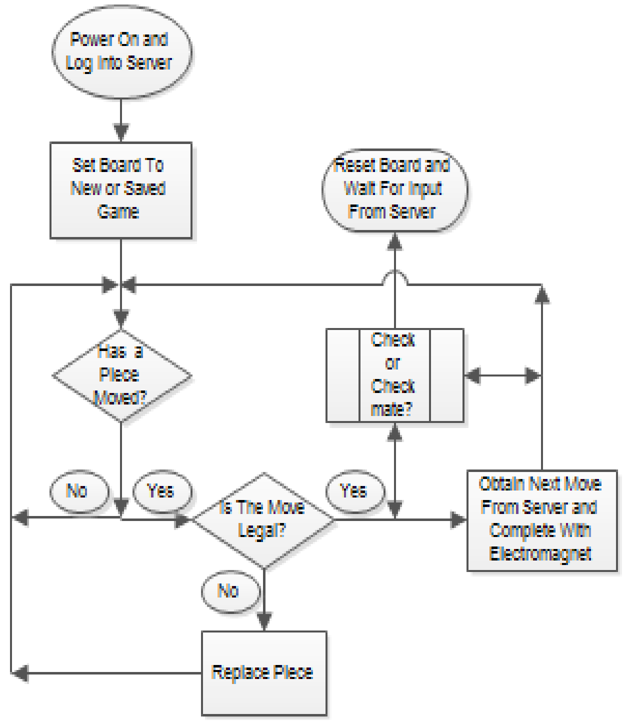
Power On and Log Into Server
Set Board To New or Saved Game
Reset Board and Wait For Input From Server
Has a Piece Moved?
Check or Check mate?
No
Yes
Is The Move Legal?
Yes
Obtain Next Move From Server and Complete With Electromagnet
No
Replace Piece

Fig.1 Flowchart showing the typical operation of the system during a game.

to load a game the system must play through the entire game move by move until it reaches the saved state. This could be avoided but it would take longer to code and since it might be helpful and is common practice to review past moves it was decided to leave the code as is.

After the board has been set the system is placed in a loop and begins using the hall-effect network to detect if a piece has been moved or not. If no piece has moved the system simple continues to gather input from the network. Once the MCU detects that a piece has been moved it determines if the move is valid or illegal by using the Chess algorithm stored in the server. If the move is illegal the piece is returned to the original tile and the player may move again. Should the move be legal the MCU relays the information to the server which in turn either sends the information to the opponent's board or calculates the next move for the compute using the algorithm. Before the next move is made the server calculates whether or not the game has a check or a checkmate, if a check is made then certain new rules discussed later will come into effect until the check is broken or the game is over. If there is no checkmate the game continues and the next move is either calculated with the algorithm or received from the opponent and the pieces are rearranged. Before the loop is closed the system once again determines if the there is a check or checkmate in the game and responds accordingly either by ending the game or returning to the beginning of the moved piece loop.

IV. Hardware Implementation

This section will discuss the actual physical components that make up the entire project some information may be redundant on items such as the Atmega 2560 however new information that is pertinent to the project is provided in each of the sections. To facilitate this discussion a block diagram has been provided in Fig. 2 which shows the physical components of the project and groups them based on their location in the overall project.

A. Microcontroller

The Atmega2560is the center of the PCB which was modeled after the Arduino Mega 2560 Rev. 3 an open source embedded MCU that is very popular with hobbyist. Using the Arduino schematics to create a PCB allows the Atmega 2560 to control 54 I/O digital output pins, 15 of which are capable of pulse width modulation, and receive input from 16 analog pins. It also run at 16MHz and can power both three and five volt products. Having the range of capabilities that using the Arduino schematic offers allows Deep RGB to run all on board functions with a single MCU. The controller will have to communicate with every other component in the project and therefore the various modes of contact between parts will be discussed. The WiFly GSX comes equipped with UART capability and it is through this serial communication format that it is controlled and used to relay information between the server and MCU. The shift registers which are used to manipulate the LED matrix are driven using three SPI pins. The Hall-Effect sensor grid is controlled using four standard I/0 pins and one analog to digital converter input pin to read the data from the sensors. There is more to the microcontroller than just the way it contacts the other components however these features have been discussed in other sections and will be omitted to avoid redundancy.

B. Server

To be implemented using an external computer the server contains a large amount of the software side of the project. It is therefore discussed with greater detail in the software section of this report.

C. Led Array

The LED array is an 8X8 array of RGB LEDs that is used to distinguish the pieces on one side of the board from the other. It is controlled by the MCU through four, eight-bit, serial in parallel out shift registers. The registers are fed a stream of data and load the registers one bit at a time when one register overflows the overflow is fed into the next register. Once all the registers are full they are given the load command which stores the 32-bit string of

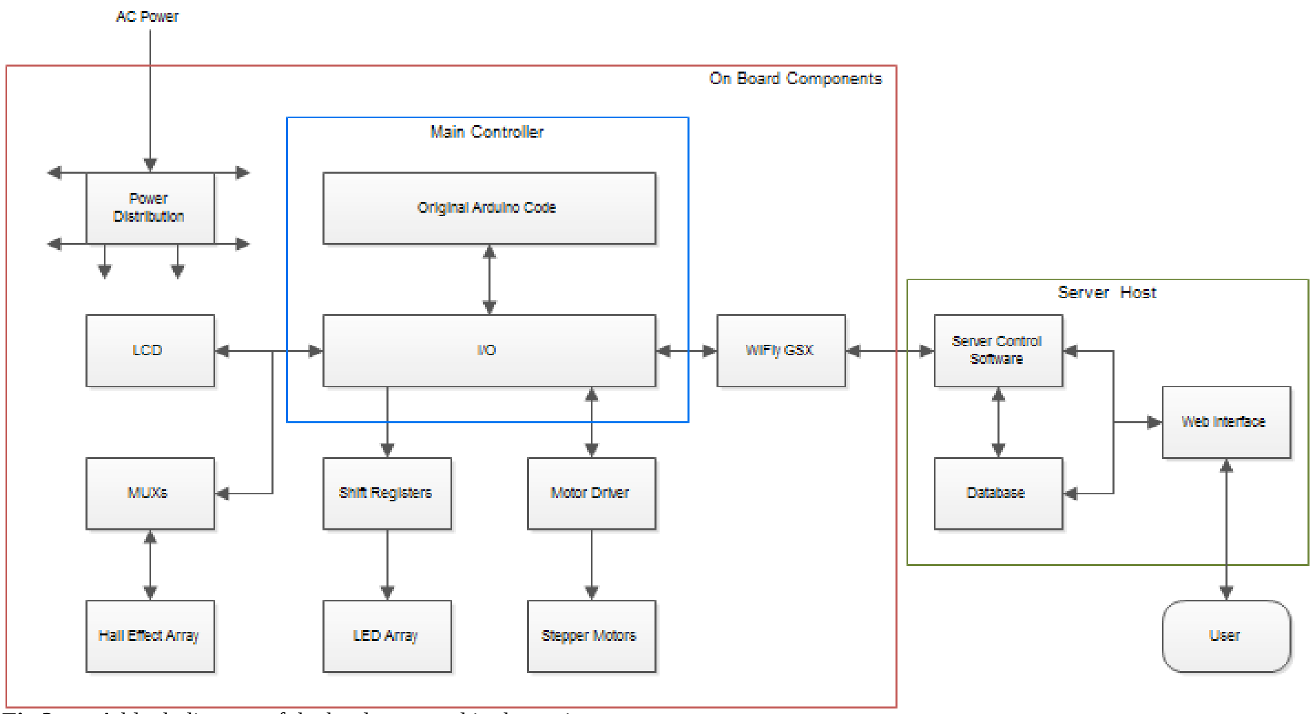

Fig.2 A block diagram of the hardware used in the project

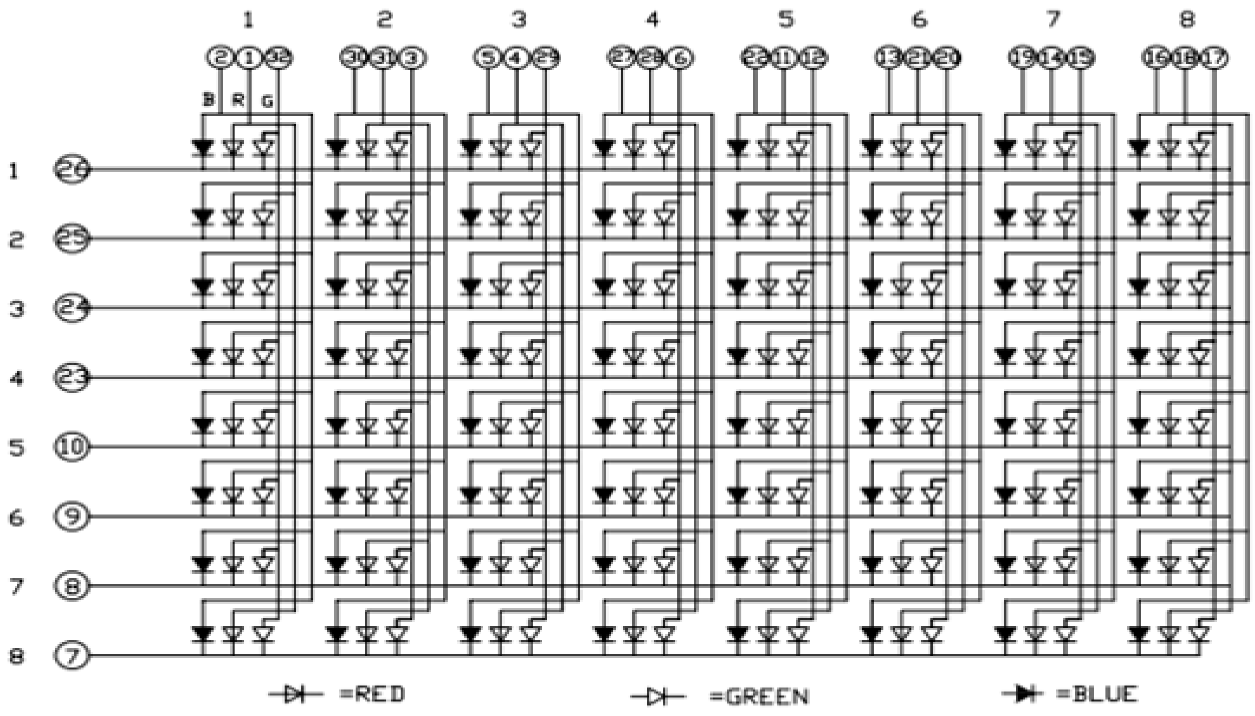

Fig.3 A block diagram of the hardware used in the project

binary in the registers. There is an active low output enable pin on each register which is grounded so that the LEDs are engaged as soon as the load command is given. How the LEDs are connected to the registers is described earlier in this report however Fig. 3 depicts the setup of the LEDs visually in case any questions remain.

D. Hall Effect Grid

Another 2D array the Hall-Effect sensors are arranged much like the LEDs however the sensor array is a 12X8 array, bringing the sum of sensors to ninety-six. The sensors used are linear ratiometric sensors which mean that they can not only detect magnetic fields they can be used to detect both the distance and polarity of the field. The sensors under each tile are used to detect if a piece has been placed on the tile or not in order to let the MCU update the server with the current position of every individual chess piece. To select a single input from a sensor four pins from the MCU control MUXs that direct a single sensor's output into an input pin on the MCU that is capable of analog to digital conversion. The A/D converter converts the analog voltage from the sensor output pins into an integer between 0 and 1023, allowing us to obtain 10 bits of resolution. Once the analog output of the sensor is converted to a digital signal, the MCU can use the data to determine where pieces are anywhere on the board and use this data in order to operate the keyboard as well.

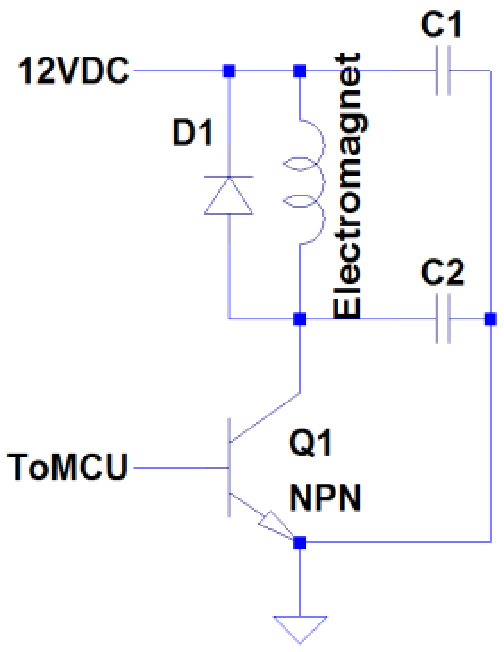

Fig.4 Circuit diagram depicting the solenoid control circuit

E. Movement System

In order to move the solenoid accurately the stepper motors must be controlled precisely with a motor controller. Deep RGB uses the A4988 stepper motor driver which is a very small embedded circuit which is capable of very small movements. Since there are two motors two drivers are needed, the A4988 is directed by the MCU using 5 general I/O pins allowing the motors to be controlled with 10 pins. The driver uses four outputs to move two Mercury ROB-09238 bipolar stepper motors by increments as small as 1/16 of a rotation. There are also sensors that tell the MCU where the motors are creating a feedback loop to prevent the system from trying to send a motor beyond its limits. When the motors have moved the solenoid into position the solenoid can be simply turned on by supplying power to the NPN transistor of the control circuit shown in Fig. 4.

F. LCD

The liquid crystal display is used to relay important information to the user. It is completely operated by the MCU through three general I/O and 3 PWM pins. Information displayed on the LCD will mostly be directed there from the server by the MCU. Messages displayed will include login verifications and status updates on the game. Its main use is a tool to assure the operator that the unit is functioning correctly.

G. Power System

The power system in Deep RGB is required to convert AC electricity from a standard U.S. wall socket into DC usable by the components in the product. Using a common laptop charger is not the simplest way to achieve this however it is much more cost effective than the original plan to use a blade server power supply. However this complicates the matter since the laptop power supply has an output of 12 volts at 6 amps it must be stepped down before it can be used. This was achieved using an ATX power supply (picoPSU-160-XT) that would step down the input voltages to 12, 5 and 3 volts. The stepped down power can then be used by the MCU which in turn supplies power to the LEDs, modules, motors, LCD and sensors.

V. RULES

Deep RGB is programmed to follow the standard rules that apply to most versions of Chess governing piece movement and taking an opponent's piece. However there are some rules that have always been up for debate as to whether they are valid or not. Therefore the following is a list of those rules that will be included in the project but may not be directly assumed to be.

A. Castling

Castling is probably the most recognized rule that falls under this category and is actually a valid move in most tournaments. It is used to avoid a check by switching the position of the king and a rook. For this move to be performed legally there are some prerequisites that must be met. The King and Rook involved must not have been previously moved, nor may the King be in check. The two pieces may not have a piece placed between them while castling and the act of castling may not place the King in check, which is true of all moves [1]. If these prerequisites are met the player may simply place the Rook on the King's tile and the King on the Rook's.

B. Promotion

Promotion occurs when a pawn reaches the final rank (row on the board) and is promoted to a higher ranking piece. This is mandatory once a pawn reaches the final rank on the board due to the fact that the pawn cannot move backwards. To complete a promotion once the pawn has reached the eighth rank it is simply swamped with a piece of choice from the player’s graveyard.

C. En Passant

En Passant is a move that was added after the rules were changed to allow pawns to move two spaces when moving from the original starting position. It became clear that with this new rule paws could pass right by their opponent without ever being threatened by their pawns and so En Passant was created. This rule allows a player to take a pawn with a pawn as though it had not passed; it is restricted to ranks four and five. En Passant requires a pawn to move two spaces forward and land alongside one of the opponent's pawns. The next move is the one and only chance for the opponent to enact this rule and take the pawn by simply removing it from the board and placing their pawn as if it had taken a piece from the rank behind the opponent's pawn. Implementing this into the chess program was slightly more difficult, but in the end necessary in order to provide a complete chess playing experience.

VI. MAIN CONTROLLER SOFTWARE

This section serves to cover the different software that is used in many of the functions throughout the project. This particular section focuses on the code that is stored and runs on board the main controller.

A. Hall Effect Sensor Array Code

This code is a derivative of the SFRGBLEDMatrix library which was developed to control LED matrixes and make them create different patterns according to the users wishes, it is the same library used for the Deep RGB LED matrix. Since the two matrices are very similar it was easier to modify the LED code rather than create a new one. The code is responsible for controlling the multiplexers that funnel data back into the MCU it then converts the raw data into digital data that is then used to determine the locations of the pieces. The actual state of the board is not saved by this code but is stored in the server to be discussed later. It was created this way because the piece detection system has no way of telling one piece from the next, therefore it was easier to have the server keep track of all the moves in a game and simply use the sensor data to update the game state.

B. LED Array Code

The LED matrix is controlled by code written using the SFRGBLEDMatrix library created specifically for RGB LED matrices. This library comes with a multitude of built in functions from running patterns across the array to turning on one LED, so many functions in fact that much of the library was deleted to save space on the main controller. This portion of the MCU code is responsible for turning the LEDs under each piece on and changing the color using PWM. Each tile is assigned a value by the code which determines the color of the tile which is used to control the duty cycle in the PWM and therefore the color. This feature of Deep RGB has proved to be the most taxing on the MCU due to the needed refresh rate. This has been solved by utilizing the Atmel’s extensive RAM as well as opting for a slightly dimmer LED color.

C. Magnetic Piece Positioning Code

Controlling stepper motors is something that is well documented and has been honed over the years. Therefore it wasn't difficult to find AccelStepper, a code library geared toward stepper motors. AccelStepper simplifies the process of driving a motor by reducing the needed information to two variables, the speed of the movement and the degrees of rotation. This information is derived from the data provided by the sensors on the racks and the saved state of the game which is located on the server. This code guides the motors so that the solenoid can move a piece along the lines between tiles in order to move past other pieces which may lay in the way. It also turns the solenoid on and off when necessary in order to take hold of the pieces.

D. Wireless Data Transfer Code

To communicate with the server via WIFI using the WiFly serial open source library would prove too much of a challenge therefore several libraries were added to ease the pain creating the communication code. Pstring and Streaming are two libraries added in order to assist with printing strings and print statements. The NewsSoftSerial and DateTime libraries were added because the WiFly library relies on functions stored in them. The code is written so that only the SSID, encryption type and password are needed to create a connection between the server and the WiFly board. Once the connection is made all necessary data may be passed between the two units in one simple string reducing the complexity of the entire system. This data includes game state updates, login information, board reset information and more.

E. LCD Code

The 16X2 LCD requires functions found in the LiquidCrystal library in order to function properly. The main function is the lcd.print() call which allows a string to be written to the LCD just by entering it in between the parenthesis. The cursor can also be placed anywhere on the 16x2 LCD by using the lcd.setCursor() function in order to keep the spacing between words correctly. This makes inquiring about login information and supplying other data to the user a simple process.

VII. SERVER AND WEB INTERFACE

Since the code to make the server function is somewhat large it warranted its own section. Before beginning it is necessary to outline the sub sections that make up the server as a whole. The database containing all the information used by the server is made up of the five tables users, games, moves, sessions, and reset. For the user to interact with the server a web interface was created that cover a wide array of functions such as user creation, keeping track of active games and also allowing players to observe other user games.

A. Database

The database is where all important information needed by the server to operate is stored. Created using Microsoft's SQ lite it is constituted by five tables containing the information. The first table is the users table which stores data on each user such as passwords and usernames in an array of structs. Games is the second table which is a list of columns containing the player ids, state of the game, date and time the game was created as well as other information. Moves is the third table which is simply a list of all the moves made in every game along with the move id, game id and the id of the user who made the move as well as other useful data. The sessions table is a list of the authentication ids of every player that has made a move in the last hour, if a move has not been made for over an hour the authentication id is recycled. The final table is the reset table which is also temporary like the previous table although it is used to store the password reset ids that are requested by users.

B. Web Interface

The server interface consists of several web pages created using htmlserver language. This allows for server to remain a relatively simple entity and therefore achievable using a single PC.

The list of pages is too long to place in this report so only a few of the more important pages will be discussed in much detail. When a user enters the web interface the first page they will encounter is the login page which assigns the user an authentication id once a matching username and password are entered. This authentication id is placed in the sessions table of the database so that the server software may access it whenever data needs to be transferred; Fig. 5 shows the login page of the web interface.

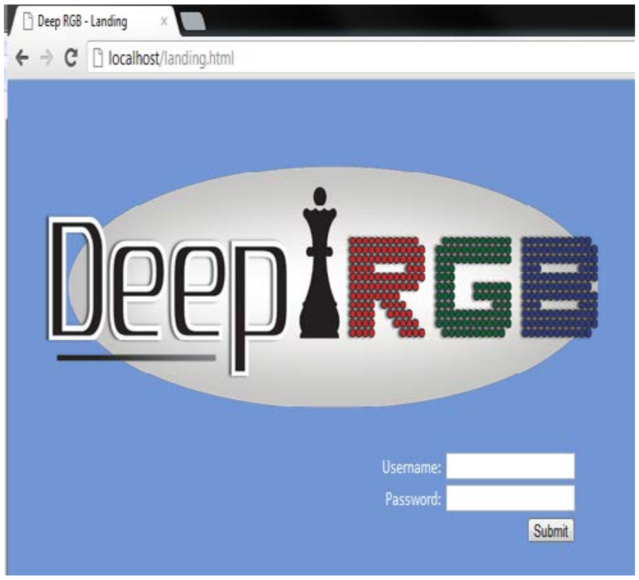

Fig.5 A screenshot of the login page used to access the web interface.

If the user cannot remember their password or have not registered yet then there are two links on the login page which allow them to either recover their password or sign up for an account. The password recovery page makes use of both the user and reset tables in the database in order to verify that the person trying to enter the site is who they claim to be. The registration page modifies the user table adding the information entered into the array of structs where it is stored for future use. Once past the log in screen the web interface directs into the Games Homepage which allows the user to select from available games to either start or continue, there are filters which allow the user to narrow the search for a game and reduce time spent on this page. After a game has been chosen the user is taken to the active game page which allows them to view the status of the game using a virtual board as shown in Fig. 6. From this page moves may be made and the server will take care of sending the updated state to the board so when the player returns to their board the game will still be current and there will be no need to physically move the pieces. Since this can be done from the web any web enabled device may be used to play, in theory someone without a Deep RGB board may play against someone who owns one from their cell phone.

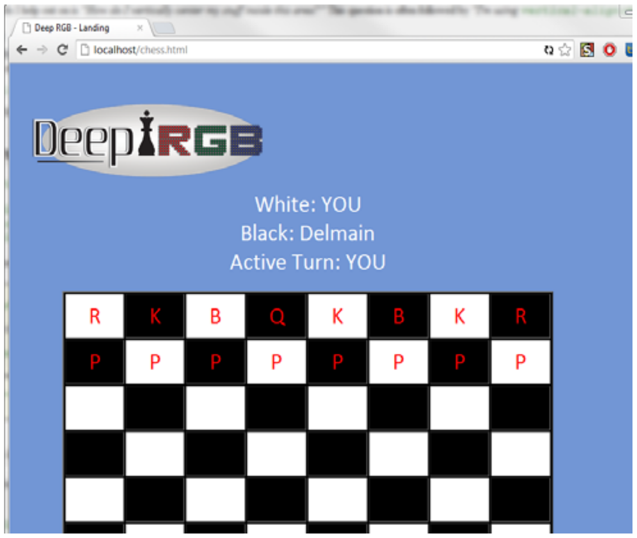

Fig.6 A rough example of the active game page used by the web interface.

VIII. Completion

To complete Deep RGB and integrate and house all of the parts a frame was made from wood to keep costs low. At the base of the wooden frame the racks, gears and stepper motors were mounted along with vibration dampening material to reduce noise. The PCB along with the WIFI module and other necessary components were tucked away along the side of the housing due to problems with the solenoid which limited the space available between the solenoid and the playing surface. The solenoid was mounted to a car driven by one of the stepper motors and above it the LED and Hall Effect Sensor arrays were placed. Since PCB design and manufacture can be both costly and time consuming the LED and Sensor matrices were implemented using prototyping board. To be exact three large perforated prototyping boards were mounted together to create a surface large enough, the boards had to be non-clad due to the fact that the copper would interfere with the solenoid being able to move the pieces. The playing surface a simple frosted pane of glass so as to diffuse the light from the bright LEDs sits atop the unit. The LCD was mounted to the same side of the frame as the PCB so as to reduce the distance between it and the controller. Once the pieces are added, clear acrylic with neodymium magnets embedded in the bottom, the board itself is ready for use. The server can simply be run by any PC available at the time, for demonstration purposes a group member will supply their personal computer to run the server.

References

[1] FIDE. (2012) Handbook E. Miscellaneous Retrieved 11 November 2012, World Wide Web: http://www.fide.com/fide/handbook.html?id=124&view=article

[2] China Young Sun LED Technology CO. (2012) YSM-2388CRGBC datasheet Retrieved 10 November 2012, World Wide Web: http://www.sparkfun.com/datasheets/Components/YSM-2388CRGBC.pdf

Authors

Shenmin Lo a graduating Electrical Engineer Shenmin hails from Aruba and is hoping to find a job back home once he has completed his bachelor's.

Joe Lunder a senior Computer Engineer preparing for graduation Joe maintains a job as a programmer and is looking forward to a bright future designing and coding various programs

Robert Wadsworth II is graduating from the university of Central Florida with a bachelor's in Electrical Engineering he is seeking a position with a company that will lead him abroad.

Siarhei Traskouski looking forward to graduating from the University of Central Florida as an Electrical Engineer Siarhei is originally from Belarus and would like to find a job working for an international engineering firm once finished.