Group 12 - Summer 2012 - Fall 2012

August 3, 2012

Shenmin Lo, EE
Joseph Lunder, CpE
Siarhei Traskouski, EE
Robert Wadsworth II, EE

TABLE OF CONTENTS

Index of Tables

1 EXECUTIVE SUMMARY

Chess is a game of strategy that has been played for hundreds of years, even in modern day chess is a popular game. Two particular trends have developed in the world of chess, correspondence chess and advanced chess playing algorithms run by computers. Correspondence chess is a form of chess that began years ago and has only grown more popular in the modern day. In correspondence chess two players play a game against one another while separated by a distance, sometimes halfway around the world. The moves in correspondence chess are placed by contacting the opposing player through some form of long-distance communication. In the modern day this form of contact has shifted from sending letters through the postal service to sending emails and using correspondence chess servers which cater to the trend. At the same time that correspondence chess was changing chess algorithms were being created. These programs which are run on the computer are aimed at finding a solution to the game of chess. At first they were not very adept at playing versus a live human but in the past few decades these programs have evolved greatly. There are now algorithms that can beat almost anyone one the planet.

At UCF there is a project known as Deep RGB named after the famous chess playing computer Deep Blue. The aim of this project is to develop something unique in the chess world which is a device that combines the two aforementioned trends into one physical product. This item is a chessboard unlike any other it will allow two players to play on it like a standard board. However it will also allow a single player to either play a chess computer or to play correspondence chess over the internet. The board itself will operate the opponent's pieces whether it is the chess algorithm or an opponent across the country. The board will use magnets to manipulate the pieces moving them when a new move is received from the computer or the opponent and correcting mistakes made by the player, re-centering pieces detecting illegal moves etc. It will allow multiple users to use one board by up-linking to a server in order to access stored user information.

This user information will contain a wealth of preferences allowing the player to set up the board in his or her own special way. It will allow them to set anything from their preferred color of the LED array stored beneath the playing surface, to the type of music they would like to listen to while playing. Deep RGB will not just use the audio system to play music selected by the user it will use it to alert players when certain events occur. Such as when a player is placed in check or when an opponent has made a move, alerting a correspondence chess player that it is their move. It will help those who are new to the game by illuminating potential moves of pieces with the LED array as well as moving pieces to their previous position if an illegal move is made. The board will be able to reset the board for another game with the press of a button using magnets under the surfaced and placed within the pieces to move them to their positions. The server will also store saved games so that the board can be set to the last saved state of

the user's game. This combined with the many features that are not listed here combine to make Deep RGB one of the greatest products to affect the chess community.

2 PROJECT DESCRIPTION

2.1 MOTIVATION

To a few of the members of the group, the game of chess has been a present, but non-overwhelming, part of life for years. Chess is the ultimate in training to think in strategic and flexible ways and is therefore taught to many children who show even a slight aptitude for the game. Though none of the members of the group developed the focus for chess that is required to become a true master of the game, each of them can enjoy a nice game every once in a while. This casual interest in the game of chess, added to a healthy obsession with robotics, is what allowed for the formation of the idea behind DeepRGB.

Chess is a game that can take a long time to play. If the players involved are not actively participating in a sit-down game, each move of the game can take hours, if not days, to make. Correspondence chess is a very popular way to play the game that involves two players potentially never meeting each other in person, yet competing in a slow, but determined manner. Chess games have been played remotely through the use of standard mail, telegraph, and email, and people more recently have even moved to connecting to remote servers to play games via a graphical user interface program. Though this last method reaches the epitome of convenience in remote, lag-free play, it loses a crucial part of the feel of playing chess, that of being able to lift a piece in your hand and decisively place it in its new position.

Chess, as a game that is easy to explain and play but difficult to master, is a perfect match for creation of an automated system to run it. The rules of chess are fairly clear-cut and therefore easy to implement on a small microcontroller, and that same microcontroller allows for communication out to a server to handle the problem-space search and return a new move.

2.2 OBJECTIVES

The objectives for Deep RGB were decided among all the members of the group over the course of several meetings and are intended to support functionality that is believed to be required for a game of chess and also that which will make Deep RGB stand out among both other chess board systems and other senior design projects.

- Main Control Unit
    - Receive information from Hall Effect Sensor Grid

    - Processing of grid for chess piece locations
    - Process possible locations of lifted pieces for LED system
    - Communication with LED and Sound systems for events
    - Communication with Wi-Fi system for sending and receiving board state
- Power Supply
    - Provide stable power to the MCU and each of the subsystems that communicate with it
    - Provide power continuously for the duration of the game of chess
- Hall Effect Sensor Grid
    - Scanned regularly for positions of pieces removed and placed on the board
    - Scanned for pieces not located centrally within the squares
- Magnetic Piece-Moving System
    - Manage grabbing a single chess piece via magnet, moving it, and leaving in place without disrupting any other piece on the board
    - Handle correction of piece location when a human player places a piece towards an edge of a square instead of in the middle.
    - Pieces mounted with small magnets allowing for triggering Hall Effect sensors
    - Magnets in pieces strong enough to be grabbed easily, but not so strong as to affect pieces around them.
- Audio System
    - Receive information on events from MCU
    - Read requested song information from internal SD card and play over integrated speakers
- LED System
    - Light each players pieces in their preferred color
    - Listen for chess piece movement events from MCU and light calculated possible move squares
    - Simulate combat when a piece is taken by varying color of LED in contested space
- Board-based Wi-Fi System
    - Connect to a public or private Wi-Fi network to facilitate communication with server

- Handle making requests to the server to change or view the current state of the board

- Server-based Software
  - Listen for connections from board or web interface and process updates to game states
  - Allow for secure connections from the board and web interface.
  - Store the game state and players for a large number of possible games
  - Recognize when a game is waiting for a computer player and send the board state to the chess engine for processing

2.3 REQUIREMENTS

The requirements for each section were decided upon by the group as a whole after suggestion by the individual group member in charge of that section. The requirements for the body were decided by the entire group together.

Table 1 - Body Requirements

Title	Requirement
BOD01	Weight of entire unit no more than 5kg.
BOD02	Dimensions of playing field no larger than 40cm by 40cm.
BOD03	Smooth playing surface that obfuscates the LED and Hall Effect sensor grid.

The requirements for the Main Control Unit and Wi-Fi System were decided upon by Shenmin Lo and were agreed upon by the other members of the group.

Table 2 - Shenmin Requirements

Title	Requirement
MCU01	Maximum 7 volts input required.
MCU02	Minimum 16KB flash memory space.
MCU03	Minimum 16MHz clock frequency.
MCU04	Minimum 1 SPI and 1 UART bus.
MCU05	Open source schematics and libraries preferable.
MCU06	Minimum 25 digital input and outputs.
MCU07	Free Integrated Development Environment.

WLS01	Minimum 0.1 Mbps transfer rate.
WLS02	Minimum 15 meter range.
WLS03	UART interfacing.
WLS04	Maximum 5 volts input.
WLS05	Low power usage possible.

The requirements for the LCD Display, Magnetic Piece Positioning System, and Power Supply were submitted by Siarhei Traskouski and were agreed upon by the other members of the group.

Table 3 - Siarhei Requirements

Title	Requirement
LCD01	Will display relevant information from the system.
MPP01	Grabber capable of accessing every square of the play area, including the board and graveyard.
MPP02	While moving, the grabber will not affect other pieces on the board.
PSS01	Takes input of American standard 60Hz, 120V AC.
PSS02	Provides stable output power of 3.3V, 5V, and 12V DC.

Robert Wadsworth decided on the requirements for the LED Matrix, Hall Effect Sensor Grid, and Audio System. They were discussed and agreed upon by the other members of the group.

Table 4 - Robert Requirements

Title	Requirement
LED01	8 by 8 grid underneath each square of the chess board.
LED02	Each LED is programmable in 255 steps of each red, green, and blue.
HES01	Coverage of 8 by 8 chess board and two 2 by 8 graveyards for recognition of movement of chess pieces.
HES02	Capable of measuring magnetic forces of up to 5 Tesla.

HES03	Hall Effect sensors in use will not suffer damage from the magnetic fields of the various chess pieces.
HES04	Linear Analog output to Main Control Unit.
AUD01	Play .wav files of 6-36KHz sample rates.
AUD02	Play audio through speaker of at least 8 Ohm.

The requirements for the server software, including the Management Application, Multiplayer Support, Saved Game Support, and Chess Algorithm were submitted by Joseph Lunder and were agreed upon by the group as a whole.

Table 5 - Joseph Requirements

Title	Requirement
MNG01	Allow communication between server and board.
MNG02	Allow for secure, individual access to the server and related games.
MNG03	Respond to move requests within 5 seconds, discounting network lag.
MPS01	Handle connections for both sides of the chess game as well as observer positions.
MPS02	Authenticate users via password.
SGS01	Store up to 100 million individual, in-progress games.
SGS02	Allow for mid-game state save of in-progress games.
CHS01	Interact with one or more chess algorithms to process moves for in-progress games.
CHS02	Run chess algorithm on maximum number of cores available to the system.
CHS03	Run chess algorithm only when processing a move for an in-progress game.
CHS04	Engine will read from Opening Books for initial moves.
CHS05	Engine will utilize Endgame Table Bases for optimum efficiency in endgame conditions.

Title	Requirement
CHS06	Engine will allow for multiple difficulty levels to match various player skill levels.

3 RESEARCH

3.1 SUBSYSTEM RESEARCH

3.1.1 MICROCONTROLLER UNIT

When it comes to microcontrollers, the world of electronics is teeming with various devices that are capable of providing the power and efficiency needed in this project. One of the most crucial decision factors when it comes to selecting an appropriate microcontroller is to opt for the most affordable overall price of the embedded system while satisfying the specifications needed in order to make the DeepRGB functional, efficient and reliable. Another key criterion is the flexibility offered within its programming language. The preferable programming language needed for this project would be C/C++ due to its simple structure and ability to easily interface with libraries needed to control the components of the chess board. Libraries are prewritten subroutines of computer code that help simplify the coding process and ensure that all components are being utilized at the full potential. Custom made libraries take a great deal of time, effort and money to create and would be in our best interest if these were available to us in an open source environment.

3.1.1.1 INTEGRATED DEVELOPMENT ENVIRONMENT COMPARISON

There are numerous microcontrollers capable of handling the functions needed for this project and each come with their own Integrated Development Environment (IDE). Having an open source IDEs would ensure that the costs remain low while providing us with the flexibility to integrate and modify the libraries needed for this project. Below is a comparison of the most popular IDEs used

3.1.1.1.1 ARDUINO IDE

Keeping all the criterions explained in section 3.2.1 in mind, one of the most popular and user friendly is the Arduino IDE. This IDE is used to develop applications for ATmega microcontrollers and has an enormous community providing new and innovative open source libraries that can be applied to solve the needs of this project. It is available for free on the internet and is equipped with many debugging tools. The IDE comes with

many libraries used to control stepper motors and there are templates available that show how to receive instructions from wireless devices. Modified versions of it can be found that add functionality to the programming software. The IDE has a programming language especially similar to C++ called Wiring and is capable of running on multiple operating system platforms such as Windows, Linux and OSX.

3.1.1.1.2 MICROCHIP - MPLAB X IDE

The MPLAB X IDE is used to build applications for Microchip microcontrollers. This award winning IDE also has a large community creating massive amounts of libraries open to public use. Its coding language is defaulted to C, but can also be modified to accept assembly code as well. The compiler has a built in debugger and comes with multiple example. It is capable of running on multiple operating systems such as the ones described above and is available for free.

3.1.1.1.3 TI MSP430 - IAR EMBEDDED WORKBENCH

There are various IDEs available to code TI's MSP430, one of which is the IAR Embedded Workbench. It utilizes a C and C++ compiler and like most compilers comes with a debugger. The IDE comes with a few examples and templates, but none that are relative to our project. The library used to control a stepper motor or receive wireless data from a device would get very complicated to build ourselves and we would prefer an open source alternative. The IDE also has problems installing on other operating systems and has no current support for Linux nor OSX computers.

3.1.1.2 MICROCONTROLLER UNIT SELECTION PROCESS

Taking into consideration the information gained from comparing the different microcontroller IDEs available, we can now compare their hardware counterparts to match the needed specifications. With all the features present in DeepRGB, the microcontroller will need to be able to execute many instructions per second and also provide many analog and digital inputs and outputs. The ratiometric linear Hall Effect sensors will output an analog signal and would need to be converted to digital in to be quantifiable by the microcontroller's software. This means that the microcontroller used must have analog to digital converters present.

Many of the other components in Deep RGB also require a Clock, Pulse Width Modulation (PWM) and Transfer/Receive (TX, RX) pin-outs to accommodate a wireless data transmission device. The goal is to have the exact amount of pin-outs needed on the microcontroller board, too few would prevent the microcontroller from performing its needed tasks, and too many would unnecessarily increase the cost of the entire project.

A crucial part of the selection process would be the availability of the development board's schematic. Having the schematic for the development board would allow us to gain a deeper understanding of the microcontroller's pin-outs and the components needed to assist all the functions present on the development board e.g. power managements, status LEDs, etc.

To prevent DeepRGB from wasting power while not in use, it will need to exhibit low power consumption and the possibility of a standby option that would switch off components that are currently not being used after a certain amount of time. Having a low power microcontroller also allows the final product to have a small power supply that would not radiate excessive heat. Below are a few microcontroller unit choices selected for comparison.

3.1.1.2.1 ATMEL CORPORATION ATMEGA 2560

Most if not all Arduino compatible microcontrollers designs are available on the Arduino website and are created under a Creative Commons license. Having this wealth of information at our hands allows us to custom make our own Printed Circuit Boards (PCB) with all components integrated onto a single board. Having one microcontroller in charge of the other components in DeepRGB would be ideal for dependability and cost reasons, but daisy chaining is possible with the majority of microcontroller boards.

The price of most Arduino compatible development boards are affordable and provide us with the ability to purchase these premade boards to setup the initial development process and later integrate this onto our own custom PCB. With the piece positioning system taking up most of the space inside the chess board, the size of the Arduino PCB is not much of a concern.

The ATmega 2560 R3 shown in Figure 1, is one of Arduino's largest microcontrollers available on the market today and will be the prime example when choosing the correct microcontroller development board. Other Arduino based microcontrollers are basically scaled down versions of the Arduino Mega 2560 R3. Having a development board allows us to start programming DeepRGB and ensure all specifications are met before creating our own PCBs.

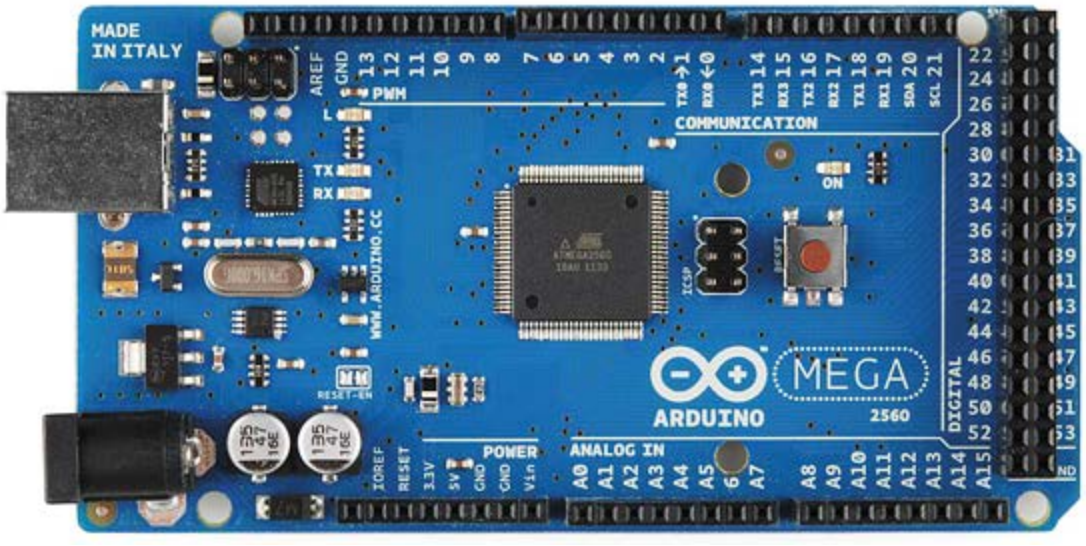


Figure 1 – The Arduino Mega 2560 development board reprinted with permission from Sparkfun.

The microcontroller development board itself is based on the Atmel Corporation ATmega2560 and utilizes Arduino's open source IDE, this allows it to be backwards compatible with other Arduino microcontrollers. With its open source schematic, we can gain a greater understanding of its components and allow us to easily remanufacture our own version of the microcontroller. Since the Arduino Mega 2560 is capable of providing us with many digital input and output ports, it can be modified to suit our specific amount of ports needed, whereas to add more ports to a microcontroller would pose a greater challenge. It also has the convenience of uploading code to it via USB whereas other microcontrollers may use a serial port. This would allow us to program the board from virtually any computer available.

The microcontroller has a total of 54 digital input and output pins of which 14 can be utilized as PWM outputs if necessary. It is also capable of reading 16 analog inputs and can receive or transmit data from 4 hardware serial ports.

Featured below, is a general summary of the Arduino Mega 2560 specifications and characteristics as well as its open source datasheet.

- Microcontroller ATmega2560
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V

• Digital I/O Pins	54 (of which 14 provide PWM output)
• Analog Input Pins	16
• DC Current per I/O Pin	40 mA
• DC Current for 3.3V Pin	50 mA
• Flash Memory	256 KB of which 8 KB used by bootloader
• SRAM	8 KB
• EEPROM	4 KB
• Clock Speed	16 MHz

Most Arduino microcontrollers can be either powered by an AC-to-DC adapter, external power source or via the USB cable used to program the board and is automatically selected when the board receives power. If the AC-to-DC adapter option is chosen, it can be plugged into the board via its equipped 2.1mm center-positive power plug to provide the board with the necessary 7-12 volts needed. DeepRGB will be utilizing an AC-to-DC adapter, but the idea of battery power did rise as a valid means of powering the system. This can be done by connecting the leads of the battery to the Gnd and Vin pin headers on the board. The recommended input voltage range for the Arduino Mega 2560 is 7-12 volts, using an adapter would supply the board with the constant regulated input voltage recommended while using a battery could cause this voltage to drop over time and cause the microcontroller to become unstable if the input voltage drops below 7 volts. The benefit of using a battery would have made DeepRGB more mobile, but would have made the entire enclosure larger in size, heavier and would require the user to plug it into a wall socket for charging every couple of hours.

The ATmega2560 has a built in flash memory of 256 KB used for storing code and the bootloader. The bootloader is a small piece of software (8 KB) that is preloaded onto the microcontroller and allows us to upload our code without the need of extra hardware. The bootloader is only active during the first few seconds after the microcontroller is reset and provides us with a quick startup time. The code that gets uploaded onto the flash memory of the Arduino is called a sketch; these sketches are coded and compiled in Arduino's open source compiler. The compiler is written in Java and based on Processing, avr-gcc and additional open source software.

The 54 on board digital pins on the can be utilized as both input and output ports using the following built in functions: pinMode(), digitalRead() and digitalWrite() where pinMode() assigns a specific pin to be used as an input or output, digitalRead() requests

a digital input from a specified pin and digitalWrite() outputs a digital signal to a pin. These pins operate at 5 volt and are able to supply or receive 40 mA. They're also fitted with a pull-up resistor ranging from twenty to fifty Kilo Ohms.

The Arduino Mega 2560 also provides us with many analog inputs; these are extremely crucial in order to make DeepRGB aware of the positioning of chess pieces and provide 10 bits of resolution which allows us to have 1024 values while measuring individual Hall effect sensors. Along with the default digital pins, the Arduino Mega 2560 has designated pins with specialized functions that provide added functionality and flexibility such as:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).
    - Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).
    - These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 2 to 13 and 44 to 46.
    - Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).
    - These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- LED: 13. There is a built-in LED connected to digital pin 13.
    - When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library.

AREF. Reference voltage for the analog inputs.

Used with analogReference().

Reset. Bring this line LOW to reset the microcontroller.

Typically used to add a reset button to shields which block the one on the board.

Most Arduino boards are capable of communicating with other microcontrollers and other devices. The Arduino Mega 2560 is equipped with four hardware Universal Asynchronous Receiver/Transmitters (UART) that can be used for Transistor-Transistor Logic (TTL) serial communications. These are crucial for DeepRGB, we will be making use of a wireless data transmission device in order to transmit and receive chess piece positions from a server. This data need to be transmitted and processed by the Arduino

and saved in an array. The wireless data transmission device will utilize these UARTs in order to gain access to the Arduino's flash memory. These UARTs are also equipped with a LED that flickers when data is transmitted from and to the Arduino, this will help us with troubleshooting any wireless transmission problems that may arise.

Almost all Arduinos come with some sort of overcurrent protection system to prevent any damage from occurring while programming. The Arduino Mega 2560 is equipped with a resettable polyfuse that shields the computer's USB port from any short circuits that may occur. If the current in the USB port exceeds 500mA, the fuse will automatically sever the connection from the computer until the short circuit or overload is removed from the Arduino.

3.1.1.2.2 MICROCHIP PIC18F46K80

The PIC family of microcontrollers designed and built by Microchip is known to be exceptionally efficient and very popular amongst embedded systems developers. These microcontrollers are known for their low cost, broad accessibility, enormous user base and its IDE. There are many examples of projects that have similar functionality as DeepRGB that use a version of the PIC microcontroller. Microchip's PIC18F46K80 microcontroller was selected to be considered for this project due to the following specifications:

- Program Memory Type — Flash
- Program Memory (KB) — 64
- CPU Speed (MIPS) — 16
- RAM Bytes — 3,648
- Data EEPROM (bytes) — 1024
- Digital Communication Peripherals — 2-A/E/USART, 1-SSP(SPI/I2C)
- Capture/Compare/PWM Peripherals — 4 CCP, 1 ECCP
- Timers — 2 x 8-bit, 3 x 16-bit
- ADC — 11 ch, 12-bit
- Comparators — 2
- CAN — 1 ECAN
- Temperature Range (C) — -40 to 125
- Operating Voltage Range (V) — 1.8 to 5.5
- Pin Count — 44
- XLP — Yes
- Cap Touch Channels — 11

The PIC18F5K80 has many core features that allow it to be known for its power efficiency. The microcontroller is able to switch between many types of modes that save power and wake back up when more power is needed. The microcontroller is capable of running in an idle mode that disables its CPU core but allows peripherals to maintain an active state. The microcontroller also offers a sleep mode called nanoWatt XLP that is

ideal for low power applications and requires only 20nA during this state. The microcontroller also provides 64KB for the application code and is rated for up to 10,000 erase/write cycles while having data retention for up to 20 years.

The development board used to develop and prototype with a PIC 18 series microcontroller is known as the Explorer Board. The development board is capable of handling the entire PIC18 MCU family of processors and would come in handy if we ever needed to replace a faulty processor or decide to upgrade to a better processor. The PIC18 Explorer Board features:

- Multiple PIC18 processors, both a PIC18F8722 on board (128KB Flash, 80 pins, superset of traditional PIC18 family), and a PIC18F87J11 Plug-In Module (128KB Flash, 80-pins, superset of J-series, PIM adjusts to accommodate 3V device). A switch selects the desired processor.
- Supports many other PIC18 devices with Plug-In Modules, supporting 28 to 80-pin PIC18 devices
- PICtail ™ daughter board connector for connection to standard expansion boards such as Ethernet, speech playback, and the many different sensors
- Expansion connector accesses full device pin-out and breadboard prototype area
- Convenient connection for MPLAB PICkit 3, ICD 3 or REAL ICE for in-circuit programming and debugging
- Alpha-numeric LCD display
- USB interface for USB to RS-232 communication
- 25LC256 SPI EEPROM
- Crystal oscillator
- Potentiometer (connected to 10-bit A/D, analog input channel)
- Analog output temperature sensor
- LEDs
- RS-232 port
- Power supply connector and programmable voltage regulator, capable of operation from 2.0V to 5.5V
- Demo software including temperature sensor demo included (illustrates Microchip's analog temperature sensor MPC9701A) and 32 kHz crystal for Real Time Clock demonstration

This microcontroller has a documented example available of a wireless transmission device connected to its UART communication port. The microcontroller is capable of connecting wirelessly, but its 3,648 bytes of RAM hinders this microcontroller and wouldn't allow us to enable all the features needed to make DeepRGB.

3.1.1.2.3 TEXAS INSTRUMENTS MSP430FR5739

The Texas Instruments MSP430FR5739 low-power microcontrollers is made up of several embedded devices with qualities such as FRAM nonvolatile memory, a 16-bit CPU, and different built in peripherals directed for several types of projects. These

features in conjunction with seven low-power modes are able to achieve a low power solution to the project. This microcontroller features:

- Embedded Microcontroller
    - 16-Bit RISC Architecture up to 24-MHz Clock
    - Wide Supply Voltage Range (2 V to 3.6 V)
    - -40°C to 85°C Operation
- Optimized Ultra-Low Power Modes
- Ultra-Low Power Ferroelectric RAM
    - Up to 16KB Nonvolatile Memory
    - Ultra-Low Power Writes
    - Fast Write at 125 ns per Word (16KB in 1 ms)
    - Built in Error Coding and Correction (ECC) and Memory Protection Unit (MPU)
    - Designed to Support Energy-Harvesting Applications
    - Universal Memory = Program + Data + Storage
    - 1015 Write Cycle Endurance
    - Radiation Resistant and Nonmagnetic
- Intelligent Digital Peripherals
    - 32-Bit Hardware Multiplier (MPY)
    - Three-Channel Internal DMA
    - Real-Time Clock With Calendar and Alarm Functions
    - Five 16-Bit Timers With up to Three Capture/Compare
    - 16-Bit Cyclic Redundancy Checker (CRC)
- High-Performance Analog
    - 16-Channel Analog Comparator With Voltage Reference and Programmable Hysteresis
    - 14-Channel 10-Bit Analog-to-Digital Converter (ADC) With Internal Reference and Sample-and-Hold 200 ksps at 100-µA Consumption
- Enhanced Serial Communication
    - eUSCI_A0 and eUSCI_A1 Support: UART With Automatic Baud-Rate Detection, IrDA Encode and Decode, SPI at Rates up to 10 Mbps
    - eUSCI_B0 Supports: I2C With Multi-Slave Addressing, SPI at Rates up to 10 Mbps
- Power Management System
    - Fully Integrated LDO
    - Supply Voltage Supervisor for Core and Supply Voltages With Reset Capability
    - Always-On Zero-Power Brownout Detection
    - Serial On-Board Programming With No External Voltage Needed
- Flexible Clock System
    - Fixed-Frequency DCO With Six Selectable Factory-Trimmed Frequencies (Device Dependent)

- Low-Power Low-Frequency Internal Clock Source (VLO)
- 32-kHz Crystals (LFXT)
- High-Frequency Crystals (HFXT)

The development board used to develop and prototype with a MSP430FR5739 microcontroller is known as the MSP-EXP430FR5739 Experimenter Board. The development board is capable of handling the entire MSP430FR57xx microcontroller family of processors and would come in handy if we ever needed to replace a faulty processor or decide to upgrade to a better processor. The MSP-EXP430FR5739 Experimenter Board features:

- Integrated MSP430FR5739 :
  - 16KB FRAM / 1KB SRAM
  - 16-Bit RISC Architecture up to 8-MHz
  - 2x Timer_A Blocks, 3x Timer_B Block
  - 1x USCI (UART/SPI/IrDA/I2C) Blocks, 16Ch 10-Bit ADC12_B, 16Ch Comp_D, 32 I/Os
- 3 axis accelerometer
- NTC Thermister
- 8 Display LED's
- Footprint for additional through-hole LDR sensor
- 2 User input Switches
- Connections
  - Connection to MSP-EXP430F5438
  - Connection to most Wireless Daughter Cards (CCxxxx RF)

As seen in the specifications of the developer board, this microcontroller has a documented example on how to connect it with a wireless device. Since DeepRGB is going to need to be wireless, this board would is showing to be up to the task.

3.1.1.3 MICROCONTROLLER UNIT SELECTION

After researching various components needed for the other features of DeepRGB, Table 6 was made in order to represent the pins needed to control every component in the system. This table will be cross-referenced with a selection of microcontrollers in the list below and will help choose a correct unit.

Table 6 – The pins and busses needed to connect the peripherals.

DeepRGB Device	Control pins needed	Pin type
Liquid Crystal display (LCD)	1 Register Select (RS) 1 Enable (EN) 4 Digital pin outputs to interface with the LCD	Digital I/O pins
Red, Green, Blue Light emitting diode matrix	1 Serial clock (CLK) 1 Chip select (CS1) 1 Master output, slave input (MOSI)	SPI I/O pins

Hall effect sensor matrix	1 Serial clock (CLK) 1 Chip select (CS2) 1 Master output, slave input (MOSI) 1 Analog input	SPI I/O pins Analog read pin
Wireless data transmission device	1 Transmit pin 1 Receive pin	UART I/O pins
XY grid stepper motors	4 direction control	Digital output pins
Electromagnet	1 Enabler 1 Potentiometer control	Digital I/O pin PWM pin
Audio-Sound Module	1 Serial clock (CLK) 1 Chip select (CS3) 1 Master output, slave input (MOSI) 1 Master input, slave output (MISO)	SPI I/O pins

With all the features of the microcontroller obtained, a comparison of the three microcontrollers is shown in Table 7:

Table 7 – Comparing the three microcontrollers in order to make a final decision.

	PIC18F46K80	**MSP430FR5739**	**Atmel Corporation ATmega 2560**
Operating voltage	1.8 - 5.5 V	2 - 3.6 V	2.7 - 5.5 V
Digital I/O pins	35	33	54
Anlaog input pins	11	14	16
UART & SPI busses	3	3	4
Program memory	64 KB	16 KB	256KB
Clock speed	64 MHz	24 MHz	16 MHz
Experience with product	None	None	Very experienced
Price per microcontroller	$4.30	$6.35	$17.97
Price per development board	$165.00	$29.00	$58.95

The PIC18F46K80 may seem like the logical choice due to its high clock speed, but due to the high cost of its development environment and small RAM size makes it less favorable amongst the three. The MSP430FR5739 comes in second place due to its small program memory a, lack of an open source schematic and insufficient experience with the product. A MSP430 LaunchPad was bought in order to try to gain more experience with the product and further research lead to an absence of libraries needed to control all of the devices for this project. This lead to the final conclusion of using the Atmel Corporation ATmega 2560 microcontroller and its Arduino ATmega 2560 development board.

3.1.2 HALL EFFECT PIECE DETECTION SYSTEM

When Deep RGB began it was unanimously decided that the pieces on the board would be moved via a magnet under the board, as to avoid the clumsiness that a robotic arm would bring to playing a game of chess. With this decision came a new challenge in the form of a detection system that must be integrated into the board in order for it to be able to move the pieces. The board could not simply "remember" where the pieces are by storing the positions in memory since a human would be moving some of the pieces themselves. Therefore the board had to have a way to actively detect where pieces were. There are several ways this could be done including RFI, the pieces could have transmitters installed and the board could be equipped to triangulate their position. This however would not only be expensive but difficult since each piece would need to broadcast a specific signal.

After further thought it was decided that the board could remember what pieces were where by using Hall Effect sensors to detect which pieces were moved and where. This would also not only be an appropriate way for the board to keep track of the user input and continue the game, but would allow the board to add new feature. The main feature that became available was the ability to correct for human error and allow the board to move pieces the user had moved previously. The board could re-center pieces that were placed too close to the edge of a square or move them back to their previous position if the user placed them on a line and the appropriate square could not be determined. It added the ability to move user pieces to the graveyard when they were taken by a piece on the side of the computer. This makes Deep RGB that much sleeker since it isn't hassling the user to remove pieces from the board and the user no longer has to be careful about placing pieces in the exact center of the tile since Deep RGB can take care of that for them.

There are several types of Hall Effect sensors such as switches, latches and ratio metric. There are other variations but these three are rather common. To start switches, like their name sounds switch on when detecting a magnetic field. This is hardly useful for a chessboard since it is either on or off and while telling us that there is definitely a piece in the area of the sensor or that there is none around whatsoever it does not tell us much more than that. A latch turns on in the presence of a magnetic field but will stay on until an opposite magnetic field is applied. Like the switch the latch stays on at a constant voltage so all it can tell us is whether or not a piece has moved by it and if a piece moved by it on the opposite side. This like the switch is useless for the chessboard. A ratio metric Hall Effect sensor not only turns on when in the presence of a magnetic field it also changes the voltage output to correspond with the distance of the field’s source from the sensor. This is exactly what is needed for the Deep RGB project to monitor the pieces on the board.

If a net of ratio metric Hall Effect sensors were placed in/under the board they could be used to calculate the position of every piece on the board. This would also need to be

implemented in the graveyards so that the board can place taken pieces appropriately, making an automatic board reset and the promotion of pawns possible.

To begin since the project is rather large it would be more cost effective to create the net with through hole components eliminating the need for costly PCB manufacture. The next parameter of concern is the sensitivity of the sensor. It cannot be too sensitive for it will sense fields from too far away and will reach maximum output before the piece is in position. This can also be solved by using a component with a wider operating range so that the maximum output is higher and therefore it can sense larger/stronger magnetic fields. The magnets that are being considered now can create a magnetic field of about 3,498 Gauss at the surface of the magnet. Most Hall Effect sensors will reach maximum output before the magnet is close to the sensor.

This problem has been considered and two solutions have been devised both are appropriate and only requires a small amount of recoding to the detection software to switch from one to the other. It is also possible that the program could be written with both algorithms and then all that is needed is a simple input to switch from one to the other.

3.1.2.1 TRIANGLE ARRANGEMENT

The first solution to the problem of maxing out the sensors is triangulation which creates a space about half an inch or more between the sensors and the magnets when the pieces are centered in the tile. This means that there will be three sensors for each square on the board so that triangulation can occur. It is possible to use fewer sensors and simply triangulate on a wider scale but this would raise problems. The magnets would need to be much stronger which could affect how the movement of the pieces worked. The positioning program would also need to calculate whether or not the piece is in the center of the appropriate square using not only less information but less accurate information and doing so would slow down the game play. It is much simpler to have three sensors equal distances from the center that way the program only needs to make sure each sensor is supplying outputs that are similar to each other. The three sensors would never read the same field strength since the polarity relative to the face of each sensor would be different. The main flaw with this strategy is the fact that there would be a very large amount of sensors used. This would not only cost more money but would cost more power as well as provide the challenge of finding a way to supply the data to the MCU without using a vast amount of I/O ports. A situation where this method should be used is when the sensors that are available would be damaged by the magnetic fields produced by the pieces and the movement device as it maintains a distance from the places these magnets will be.

3.1.2.2 ALL CORNERS ARRANGEMENT

The second solution to the sensor problem is using one sensor for every corner created by the tiles. This gives four data points per tile and therefore gives the program more data from which to extrapolate the information needed about each pieces position. However if a sensor is used at every corner the typical sensor could be detecting anywhere from 0 to 4 pieces at one time. In light of this fact a register with the previous data taken from each sensor should be implemented so that the program has a way to test the change in the readings. One major benefit of this design is that less sensors are used, reducing cost and need for input ports on the MCU. This method should be used when the sensors available will not be damaged by the high magnetic field created when a piece is moved down the line to a new position. For reference, the approximate field strength of our magnets at various distances is available in Table 8.

Table 8 - A table showing the approximate magnetic field strength from the magnet being considered at different distances.

Distance From Center of Magnet(inches)	Magnetic Field Strength(Gauss)
0.0	3,498
0.5	1,707.69
.75	505.982
1.0	213.461

The board will have two graveyards in addition to the field (playing area), bringing the total number of tiles up to ninety-six. Considering the two arrangements of sensors above that could mean either 288 sensors (triangle method) or 117 sensors if one is used at every corner. To solve this problem analog multiplexers can be used as switches to allow a single sensor signal through at one time. This can be done for both arrangements and the setup is quite similar the primary difference is that there will naturally be less registers for the corners method. To begin every cathode of every sensor in a row should be connected by a bar. For the triangle arrangement this only pertains to sensors of the same position i.e. top, left, right. Then each anode of each sensor on a row should be connected to a bar. Each cathode bar will feed into a different input on an analog MUX. For the triangle arrangement each position will go to its own MUX so there will be a top MUX, as well as a left and right MUX. Each anode bar will feed into a different input on an analog MUX. By doing this each sensor can be turned on by selecting the right input on the corresponding cathode MUX and the anode MUX. This can be done quickly as a sensor can normally power up in about 30 microseconds. Every output will feed into the same analog input on the MCU since the sensors will only be on one at a time there is no need for more than one input.

However even though there is only one analog input needed to read the data there can be either 8 or 13 digital outputs needed to control the MUXs depending on the arrangement used. This can be reduced by using shift register which need only 3 digital outputs to be controlled bringing the total I/O pins used by either arrangement down to 4.

3.1.3 MAGNETIC PIECE-MOVING SYSTEM

We began to research our project with investigating X and Y positioning tables. The XY rails will be located underneath the chess board and it will be capable of moving from one X-Y coordinate to another simultaneously in order to move certain chess pieces. This is done by applying two magnets to the system. Each chess piece will have a magnet attached to the bottom of it and under the board we will have one strong magnet installed on the moving positioning system. Choosing the right positioning system is a crucial part of this project and is what makes this project seem so magical. We were considering tree approaches of how we would create such a complex positioning system. The next step would be choosing the type of magnets for the chess pieces, the strong magnet for XY rail system and the motors that are capable of providing the accuracy we require.

Most of the XY rail systems we were researching had similar components: they all had two motors and gear system which transfer rotating movement of motors into liner movement in X and Y directions. First approach, as seen in Figure 2, we considered for moving the system was based on an internal ring gear. Motors with cylindrical shaped gear placed on a sliding racks and the same size and shape gear is on the one side of the board. They are connected with a rubber internal gear. Then motor is rotating the system is moving in one of the X directions. To move system in the Y direction we need to send command from the microcontroller to rotate the other motor and system will move in Y direction.

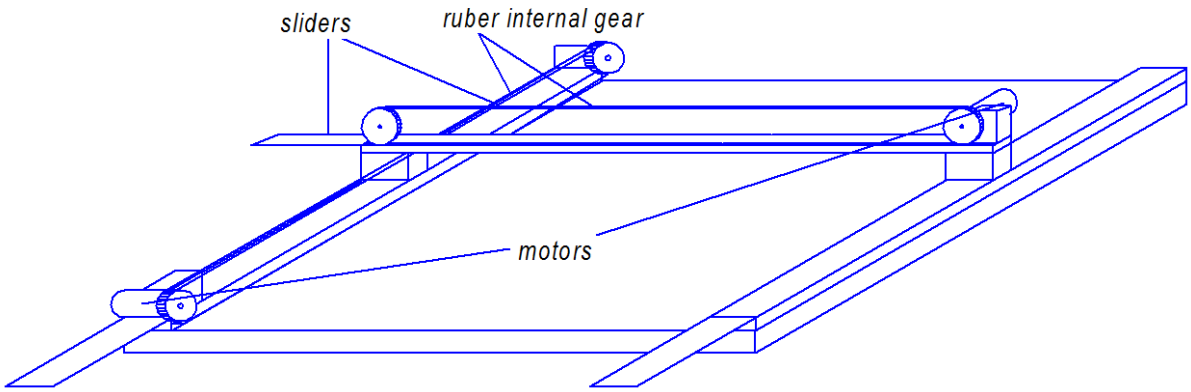


Figure 2 - The first choice for the system

The second approach was a moving system as seen in Figure 3, is based on a worm gear. Instead of motors being installed on sliding racks as it was on the first XY moving system, the second approach system is equipped with four aluminum rods; two in X direction and two in Y direction. This system also utilizes one worm gear between each pair of aluminum rods. The motors are attached to one end of the worm gear and they have a

cylindrical shaped gear which rotates the worm gear and system thus making the positioning system move in the X and Y directions

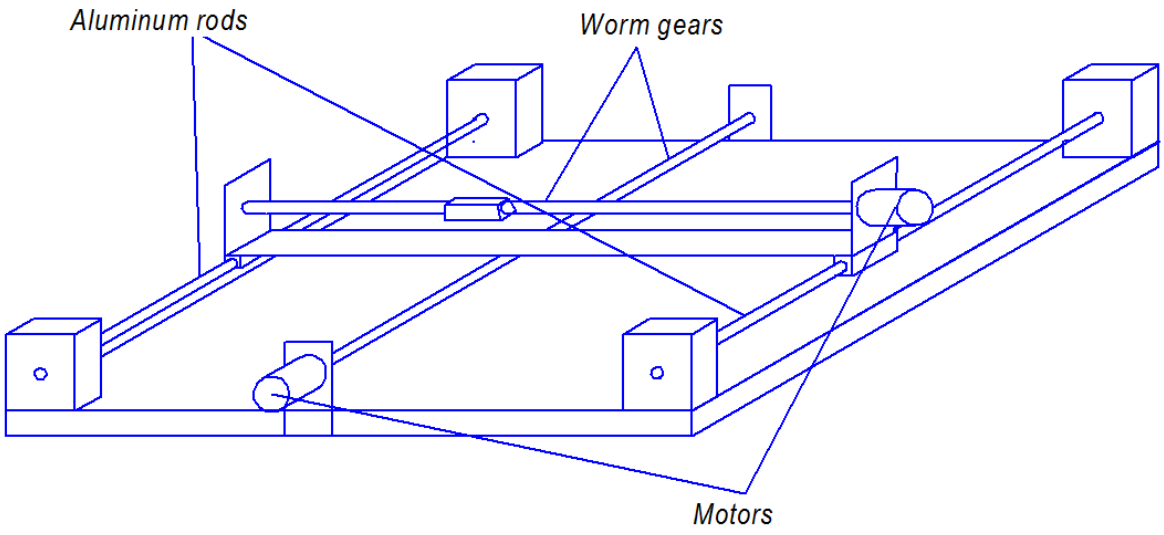


Figure 3 - The second choice for the system.

Our next approach was a system which is based on gear racks. It has a pair of sliding racks in Y direction and one sliding rack in X direction as depicted in Figure 4. It has one motor to move in X and one motor to move in Y direction. The Y direction pair of sliding racks has one motor installed on the top of one of the racks. The X sliding rack is installed perpendicular to the pair of Y sliding racks. One motor is attached to the top of X direction rack. Each motor has a cylindrically shaped gear installed on a shaft of the motor. Along one of the sliding tracks in X and Y directions, we can attach gear tracks and by rotating cylindrical gears motors can move our system in both the X and Y direction.

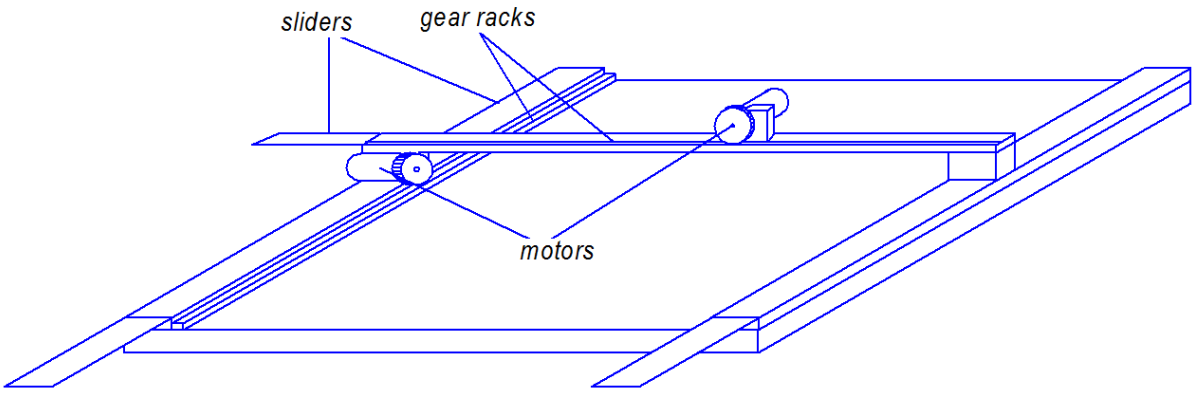


Figure 4 - The third choice for the system

By researching those systems we determined the pros and cons of each system. The most important properties for us were cost of the system, level of difficulty during the installation, reliability and level of noise it makes when it operates. The first system has many moving parts and also it has internal ring gear made from rubber which can slip off or break, so reliability of this system will be less than that of other systems. As far as noise level, the first system has higher level of noise as well because it has many moving parts. Third system is more reliable because it has the least amount of moving parts and it has rack gears instead of a ring. Noise level of the second system is almost the same as the level of the second system. The most reliable from all those systems is the second system with a worm gear because it has aluminum poles and a worm gear in the middle which makes system more balanced compared to all other systems and also produces the lowest noise level. Cost wise however, the second system is the most expensive because it consists of technologically advanced parts such as linear bearing rings which can be quite expensive and aluminum rods that are hard to manipulate. The first system will be in a middle price range and the cheapest will be the rack gear system. A comparison of the positioning system possibilities is shown in Table 9.

Table 9 - Summary Characteristics of XY Positioning Tables

	Noise level	Difficulty of installation	Reliability	Cost of the system
Internal gear	High	Middle	Low	Middle
Worm gear	Low	High	High	High
Rack gear	Middle	Low	Middle	Low

The rack gear system has the best set of qualities we need for our project that is why we decided to move forward with this system for our project. The only big disadvantage of a rack gear system compared to a worm gear system is the sliding racks. During movement when the motor moves the positioning magnet to the end of the platform those racks are sliding out, this can take a lot of extra space when we will try to install this XY table inside a box. During our discussions with the group we decided to consider creation of the hybrid system which includes best qualities of worm gear system and rack gear system.

On the top of X Axis we need to install the magnet system to attract the magnets above the chess board which are mounted in each chess piece. This will allow us to move each figure by moving our magnet system underneath the board. One way to create this magnet system is to use a servo motor which raises and lowers a powerful magnet. This approach requires the purchase of an additional motor and it may increase the total cost of the system. The extra motor will need extra inputs from a microcontroller and in our case when we have a lot of sensors actively reading, it may reduce the reaction time of the microcontroller. Such a system might also require additional height to move the rod with the magnet. Another approach is to create a magnet system where electromagnet will be attached to the top of the X Axis. In this case we will not need to

use a motor to lift or raise rode with the magnet, we just will need to position electromagnet under the right chess piece and then pull the piece into the next X and Y position and release Pieces by turning off electromagnet. In addition this approach will save us extra outputs on a microcontroller because to control the electromagnet we only need to send an on or off signal.

Electromagnets typically come in two types; Horizontal Electromagnets and Vertical Electromagnets. Since we need to attract chess Pieces from above of the electromagnet we need to use a vertical electromagnet. Vertical electromagnets can have different shapes such as cylindrical, rectangular or cubical they also can have opposite poles and parallel poles. Due to the fact that base of chess pieces have a round form; we need to look into using a cylindrical shape electromagnet so it would not attract neighboring pieces.

The approximate diameter for our electromagnet should be no larger than 3.81cm and no smaller than 2.54cm. Our first choice was electromagnet R-1207-12 Company "Magnetech Corporation". This electromagnet has a small diameter of 1-1/4" and a big holding value of 45 lb although holding value was calculated in case there is no air gap between mounting surface and a magnet we can still attract other magnets at a distance because magnet force decreases exponentially. Practical field ratio versus size of an electromagnet is that for about every 4 units of electromagnet diameter we receive one unit of magnetic field distance. Since electromagnet diameter is 3.175cm, our magnetic field distance will be approximately 0.95cm. Due to the fact that we also had magnets on the top of the chess board our magnetic field distance will be bigger than 0.95 cm.

The EM 137 Electromagnet Company "APW Company" was our second choice. This electromagnet has 3.47cm diameter and 33-44lb holding value. The voltage supply for this magnet is 12V. Also it has standard thru hole mounting and includes brackets to hold it in place.

Our last choice was ER2-103 Electromagnet Company "Industrial Magnetic Inc." The magnet has the same diameter as our previous two choices. Different voltage supply 24 V and smaller holding force value. This electromagnet also is twice as expensive as another two electromagnets. We found during our research that there are not many companies that are selling small, powerful and cheap electromagnets. We think the reason is that not many DIY projects involved small and strong electromagnets. But on the other hand there are a lot of inquiries on the Internet from people about how to create electromagnets at home. A comparison of the electromagnet types is made in Table 10.

Table 10 - Electromagnet Specification Comparison

Model of electromagnet	R-1207-12	EM 137	ER2-103
Voltage, V	12	12	24
Current	DC	DC	DC
Duty	Continuous	Continuous	Continuous
Watts	3.3	5	4.2
Amps	0.28	0.41	n/a
Holding Force, Lbs	45	33	22
Weight, Lbs	0.24	0.24	0.4
Diameter, cm	3.175	3.493	3.175
Height, cm	1.905	2.06	n/a
Price, $	40	29.24	76.06

The size of the positioning system is also one of the important factors for us. We didn't want to create big and bulky chess board. Based on specifications boards have approximate size of not bigger than 70cm by 70cm and height of a board should not exceed 30cm. Based on this data we need to choose optimal size and strength of the motors.

In addition to the electromagnet we need to choose some magnets which needed to be installed into chess pieces. After some research, we realized that there are many places there we can purchase magnets. The smallest magnets and the strongest one were neodymium magnets. Those magnets are very strong and the prices for them are almost the same in every store we checked. Finally we chose to purchase magnet from the web site amazon.com.

Stepper Motor

Our research came to the point where we need to research motors and motor controller for our positioning moving system. Block diagram below (Figure 3.2.3.1) shows basic path from sensors to motors in order to move electromagnet under specific location. Positioning sensors register the chess piece position on the board and send this information to the MCU. After it has processed the data from sensors, it sends feed back to the motor controller in a form of different voltage and current levels. The motor controller then sends certain signals to motors in order to move electromagnet in a specific location. This process is depicted in Figure 5.

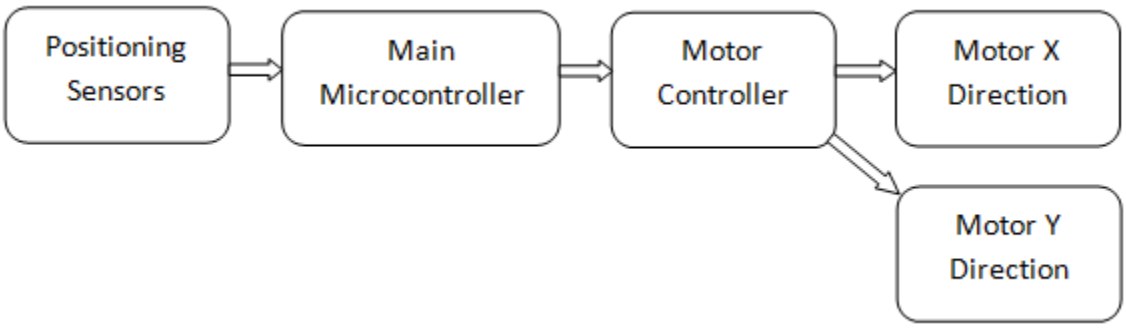


Figure 5 - Block diagram about motor controlling

There are many different types of motors that are used for different applications. There are also many motor controllers used to control different types motors. In our research we decided to consider all possible motor types we can use for our design in order to find the best solution for our Deep RGB project. Of all available motors variations, there are basically two main motor types –these are DC and AC motors. Our research included motors such ass AC motors, liner actuators, DC brushed motors, DC brushless and stepper motors.

First type of motors we were considering was AC motor. The two main categories in which AC motors are divided are the synchronous type and the induction type. Synchronous AC motors use AC line frequency where they use a permanent magnet to generate the rotor magnetic field and allow it to rotate. In this type of motors magnetic field is motionlessly relative to the rotor. Synchronism with line frequency allows motors to run very smoothly and quietly since sine waves allow smooth transition from one phase to another. Due to the fact that AC synchronous motors run at the same frequency as line frequency gives them a low torque value. As a result these motors regularly come with a gear box to increase the torque. The presence of a gear box can increase cost of the system. Figure 6 shows how to control an AC motor with a PIC microcontroller

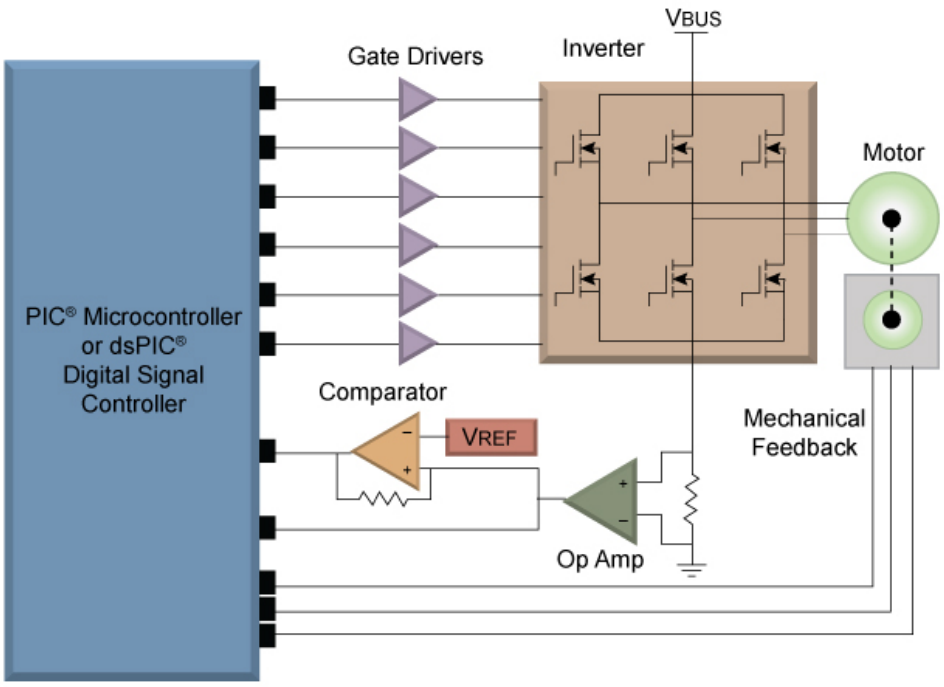


Figure 6 - Block diagram of controlling AC motor by microcontroller

Induction type AC motors are cheaper compared to synchronous motors due to less complicated production process. This type of motor acts as an asynchronous AC motor where the electrical power is transferred to the rotor from stator by an electromagnetic field. The noise level of these motors is lower compared to asynchronous motors and they are easy to start. However induction AC motors are less power efficient than synchronous motors. There are signals from MCU sent to the power inverter which converts them into AC power to supply three phases of the motor. To control motor movement feedback os sent to the MCU in the form of two currents from two phases and information from a tachometer about the physical movement and position of the AC motor. During our research we decided to connect only one motor, therefore we need approximately five inputs and five outputs from the MCU. Since we need at least two motors the amount of I/O pins needed from the MCU will double. Overall AC motors are not very popular to use in small electronic projects. There are also not many motor drivers available for such motors.

The next type of motors we were researching were linear actuators. These types of motors belong to the category of DC motors. Linear actuators transfer different energy such as pneumatic, hydraulic, and electrical into the movement. In our case we were interested in the usage of DC power to transfer energy of rotation to the linear movement. DC linear actuators are quiet, reliable and fast. They tend to already come with preinstalled gear boxes and use a lead screw. They also tend to be driven by a permanent magnet DC motor. The Basic actuator comes only with two wires one for

negative and one for positive voltage. Different types of actuators need 5, 6, 12 or 24 volts from the power supply. To start an actuator all you need is just to provide power to its wires and the actuator will start moving.

When power is disconnected an actuator will hold its position, this option might be useful for our project because we need our positioning system to stay in a given position until we need to move electromagnet to the next position. The speed of the linear actuator is regulated by applying different voltage level so in case we need to decrease speed we need to lower supplying voltage level. To do so we can use a PWM which we can adjust to a required level so we will have speed we need for our application.

Due to the fact that linear actuators have wide use in a positioning system tables we can use them for our movement system. After we did our research on the actuators pricing, the cheapest actuator we found was from www.Firgelli.com at the price of $80. Since we need two of them for our XY positioning table total cost of actuators will double so this becomes too expensive for our budget. Another disadvantage is that price rises dramatically when actuator length increases, it makes the physical size of the chess board too big to be carried around.

DC motors come in two main categories: brushed DC motors and brushless DC motors. Both motor types use DC current to run but since DC has a constant voltage, we will need to make use of the MCU's PWM pins. We need to know exact position of DC motor, to be able to have this requirement, we need to have feedback from the motor to the motor control system so that microcontroller can adjust position of the DC motor based on received information from sensors and feedback. We can see basic connection of DC Brushed motor from Figure 7.

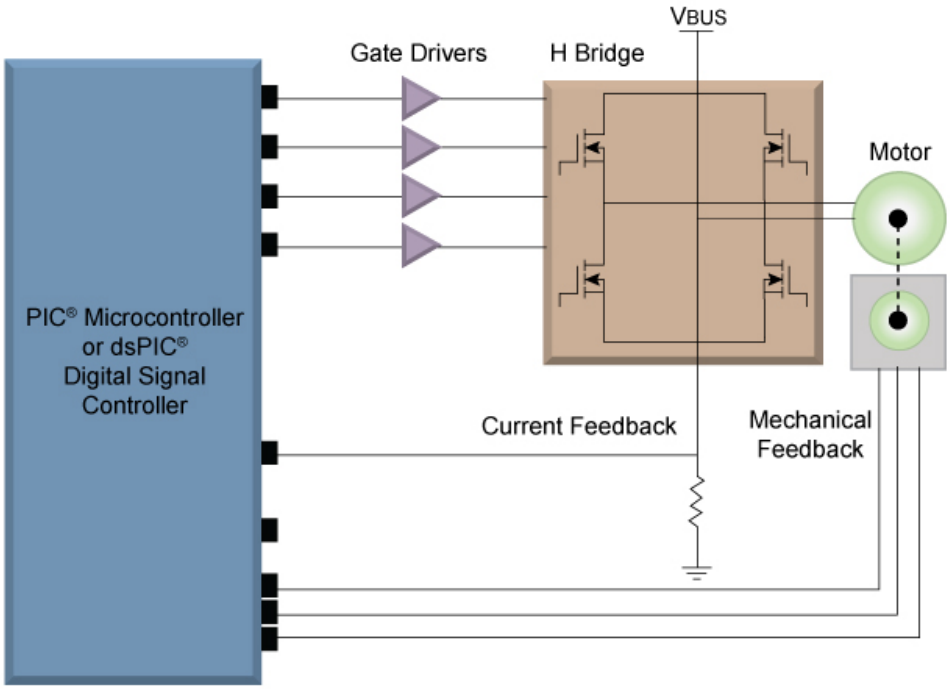


Figure 7 - DC motor control block diagram

Brushed DC motors have permanent magnets installed in stator and coils on the rotor through brushes. On the other hand, brushless motors do not have brushes because rotor has a permanent magnet and starter has coils. Typically, Hall Effect sensors are used to activate one stator's coil after another. For our project we compared brushed and brushless motors to decide which type is more suitable for us. We found that each type of motors has its own advantages and disadvantages. We created a table were we described advantages and disadvantages of those motors. Table 11 is where we first describe brushed DC motor.

Table 11 - Advantages and disadvantages of brushed and brushless DC motors

Advantages	Disadvantages
Brushed DC Motors	
Brushes are replaceable	Maintenance required
Low construction cost	Poor heat dissipation
For fixed speeds not required controller	Lower speed range
Two wire control	Higher electronic noise
Simple and inexpensive control	
Brushless DC Motors	
No voltage drop across brushes	Complex control
Small size and high output	Higher construction cost
Require much less maintenance since	

there are no bushes	
Higher speed range	
Low electronic noise	
Commutation based on Hall Sensors	

Two most common types of DC motors used in the projects are stepper motors and servo motors. We compared two types of motors to choose the motor type which would suite our project the best. Servo motors are more expensive compared to stepper motors of the same size and level of power. Versatility wise, stepper motors are better because they are used in anything from clocks to scanners while servo motors are used more often in larger projects. Servo motors come in wider variety of frame sizes but stepper motors still can be found in small and medium size frame. Stepper motors are easier to setup, all you need is just to connect wires from a motor to the stepper motor driver. The noise levels are better in servo motors; however our motors will be inside of the chess board which reduces noise level. Moreover, stepper motors are most of the times used in a direct drive mode so you can just attach the shaft of the motor directly while servo motors have high RPM and usually requires more gearing ratios compared to stepper motors. As we could see stepper motors are better cost, easier to setup, we can connect them directly, they are very accurate in positioning of their shifts and they have optimal proportion power and size which is enough for our project. Because of all of those characteristics we decided to use stepper motors for our project.

Stepper motors do not have brushes like servo motors. Full rotation of the motor is divided into a number of steps. Most common stepper motors are manufactured with 180, 200 or 400 steps per revolution which is creating stepping angels 2, 1.8 and 0.9 degree per step. For our project we chose to use motor with 200 numbers of steps because it has better combination of the number of steps and cost. The motor can also be driven with or without a stepper motor driver. A stepper motor driver allows us to increase manufactured amount of steps by dividing step number into smaller portions. We decided to use stepper motor driver because we need to have ability to be very precise in positioning of our moving system. There are three types of stepper motors: permanent magnet stepper, variable reluctance stepper and hybrid step motor. Last type is a combination of two other types but it gives us better performance with respect to power, speed and a number of steps. Thus the hybrid stepper motor was chosen for our project.

There are two basic types of stepper motor based on a coil connection. One is the unipolar motor and the other is bipolar. Hybrid step unipolar motors have 5 or 6 wires which are wired to the end of two windings with a center tap on each of them. Hybrid step bipolar motor have 4 wires which are wired two the end of two windings. Based on this, bipolar motors offer bidirectional current flow while unipolar motors allow having the current flow in half of windings since torque is directly related to winding current bipolar motors can create much more torque compare to unipolar motors. Wires in unipolar motors are also thinner and this directly relates to increasing heat loss in unipolar motors and also requires more wires while on the other side bipolar motors

require more complicated circuitry, typically with H-bridge connection. Since motor step driver is quite cheap using bipolar motors will not increase the cost of our moving system. All these facts lead us to the decision that we need to use bipolar step motors for our moving system design. Figure 8 illustrates the basic connection of the stepper motor with a microcontroller.

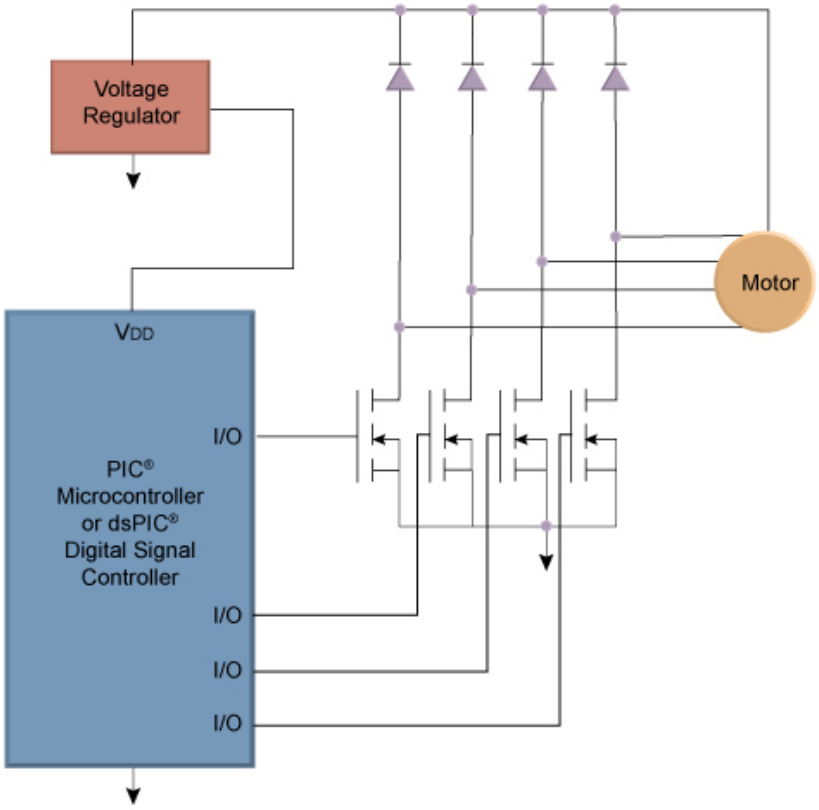


Figure 8 - Stepper motor control block Diagram

From this figure we can see that to connect stepper motor we do not need to have feedback loop due to the specifics of the construction of the stepper which were discussed earlier in the paper.

During our research about bipolar stepper motors our first choice became Stepper Motor -200 steps/rev, 12V 350mA. This motor has many features that we are looking for. It has 4-wire bipolar stepper with 1.8 degree per step for more precise positioning and has a good holding torque 28oz-in. Max current is 350mA and is suitable for our motor driver and can be easily supplied by a power adapter

Our second choice motor was Unipolar/Bipolar, 200 steps/rev, 4V 1200mA. This step motor is a hybrid and can be used as bipolar or unipolar stepper motor. It has 6 wires and 1.8 degree step angle. Every phase consume 1200mA at 4V which lead for a high holding torque 3170g-cm (44oz-in). This motor has a bigger frame size (42mmx42mm)

but it has also quite big holding torque which is almost twice as much as our first choice motor and frame size is only little bit bigger.

Another potential motor was Applied Motion – 5017 – 009 Bipolar Stepper Motor. This motor has medium level of torque 31.4oz-in compared to previous two motors, 6 leads which allow us to use this motor as bipolar and as unipolar also it has frame size NEMA 17 (42mmx42mm). Motor consumes 570mA current per phase that gives it medium power consumption. Weight of motors is not as important a factor in our project as a frame size, torque, and power consumption.

The Mercury ROB-09238 Bipolar Stepper Motor was our next choice. This motor has only 4- wires which make it simple and powerful at the same time. It has 1.8 degree step angle, 2 phases and it consumes 330mA current and 12V voltage. Holding torque is 2300g-cm (31.9oz-in). Frame size is standard. Parameters for this motor are similar to a previous motor the difference between them is that this motor has 4 lead wires instead of 6 so it can be used only as a bipolar motor when the other motor can be used as unipolar as well as bipolar. We displayed data of all stepper motors we have to choose from in Table 12 below.

Table 12 - Motor Specifications Comparison

Motor Model #	**Stepper Motor -200 steps/rev**	**Unipolar/Bipolar, 200 steps/rev**	**Applied Motion –5017–009 Bipolar Stepper Motor**	**Mercury ROB-09238 Bipolar Stepper Motor**
Motor type	Bipolar	Unipolar/Bipolar	Unipolar/Bipolar	Bipolar
Step Angel, degree	1.8	1.8	1.8	1.8
# of Wire Leads	4	6	6	4
Leads length, mm	230	300	305	1200
Drive Shaft Diameter, mm	5	5	5	5
Rated Voltage, V	12	4	6	12
Rated Current, mA	350	1200	570	330
Holding Torque, oz-in	28	44	31.4	31.9
Winding Resistance, Ohm	34	3.3	15	34

Frame Size, mm	42.3 x 42.3	42.3 x 42.3	42.3 x 42.3	42.3 x 42.3
Weights, g	200	350	n/a	200
Price, $	14	19.95	12.95	14.95

We chose stepper motor to move our moving system for its accuracy. For example our stepper motor has 1.8 degree per step so it is totally 200 steps for a full turn of the motor rotor. To move motor to a certain number of steps controller should send same number of impulses to the motor and it regulates the speed of the rotation through the frequency of pulses from the controller.

Stepper Motor Controller

During our research we realized that there are many different motor controllers available on the market. From all available variety of controllers it was hard for us in the beginning to choose a controller for our project. However, after we realized that all stepper motors controllers can be divided into simple categories which created more simple understanding of stepper controllers. Motor controllers can be divided into two basic categories.

First type of controllers has constant voltage. This type of controllers supplies small voltage to each coil. In order to make motor run they just turn on or turn off voltage from specific coils by using transistors. It creates certain limitations on using such controllers during high speed since when a motor starts turning it creates counter voltage which can be as high as the supply voltage. However price of those controllers are very low and they can be used in applications were speed of the stepper is not bigger than 2 or 3 turns per second. This speed is relatively low for our project but it still can be considered since we were planning to use cylindrical shaped gear attached to the stepper motor shaft which increase length of one turn.

The second type of controllers has constant current. In this case stepper motor would use much bigger voltage compared to the constant voltage controller. Due to higher voltage, the motor will also receive a much higher current. One way to protect coils is to use large resistors. Another way is to turn voltage quickly on and off by using pulse width modulation (PWM). This technique is more complicated compared to usage of large resistors since we need to adjust current to low or high levels. However the advantage of this method is a low power leakage.

Simple way to control bipolar stepper motor is to provide specific impulse sequence from the microcontroller to the motor. To do so we need to supply specific voltage to each coil so coils get energized and not energized in a specific way. In Table 13, we show the sequence of voltage needed to be applied to bipolar stepper motor coils to perform full step.

Table 13 - One step voltage sequence is applied to the bipolar stepper motor.

Sequence	Coil 1	Coil 2
1	High+	Low
2	Low	High+
3	High+	Low
4	Low	High-

We can provide this sequence from Table 3.2.3.5 to the microcontroller through the H-bridge. H-bridge is working as a set of switches which is turning coils on and off and they also do reverse polarity so motor can move in a reverse direction as needed.

The usage of a controller built on an H-bridge was under serious consideration by our group. One of the most reasons was that it is the most inexpensive motor controller which can be built from simple electronic parts. One coil requires a full H-bridge for control. It consists from a set of 4 transistors and resistors to limit the current. It can also be built by using MOSFET instead of transistors. We provide a simple diagram of H-bridge on the Figure 9. Since we have two coils in the bipolar stepper motor to control it we need to have two full H- bridges. We need to have a total of 16 transistor and 16 resistors to control two motors. Since building motor controller was not our priority task in this project we decided to search for available solutions on the market.

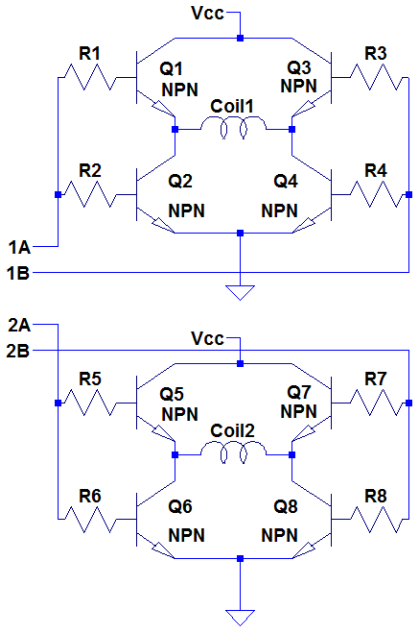


Figure 9 - Full H-bridge to control the stepper motor

During our research we found out that the L293D dual H-bridge chip has two build in H-bridges. Texas Instruments is one of the companies that produced this chip. According to their datasheet this chip has wide voltage supply from 4.5V to 36V. It is a good option since it gave us ability to use it with motors that have a different level of voltage supply. Another option is to use an output current that can be up to 600mA which might not allows us to use full range of permitted current level for stepper motor. Presence of thermal protection in this chip will protect controller from overheating was considered

by our group as a big advantage compared to building a controller from scratch since we can accidently burn one of the transistors during fine-tuning the controller.

After we checked the same type of chip from the other companies we found out that they all have the same basic construction and data specification. For example chip from the company STMicroelectronics L239D has supply voltage 4.5V to 36V abd also has the same output current 600mA as a chip from Texas Instruments.

The next chip we were researching was L298 Dual H-bridge. This chip has basically the same structure compared to the L239 the main difference is bigger power dissipation. It has operating supply voltage up to 46V and output current per channel is up to 2A which is double the current from L239. Higher current level allows us to use stronger stepper motors or have higher torque at the same speed.

Our next step was to find a motor controller based on L298 chip. After we did our research, we found three main motor driver opponents based on L298. The first motor driver we researched was ROB-09670 Motor Driver 2A Dual L298 H-Bridge. It has four directional LED's, eight Schottky EMF-protection diodes. The motor driver requires a 6V to 35V voltage supply and has up to 2A output current per channel. An important feature is also the presence of the heat sink due to high operating temperature of the L298. Another convenient option is presence of 5V voltage regulator with power output which allows us to supply additional logic component from this driver. One disadvantage of this driver is the price of $34. Since we need two of those drivers total price will be up to $70 which is too high for our budget.

The next opponent was Solarbotics L298 Compact Motor Driver from solarbotics.com. This motor driver has very similar characteristics. The price makes a huge difference between two of them. The motor driver from Solarbotics costs only $18.95, this would save us about $30 if we buy two of those motor drivers. But big disadvantage of this driver is that it does not have heat sink attached to the chip. This can lead to overheating problem during motor driver operation. One of the ways to avoid it was to buy heat sink separately and attach it to the motor driver but it will create additional expense and require extra work. We chose to continue our research for a better motor driver.

Last opponent which is based on L298 was L298N Dual h-Bridge DC Stepper Motor Controller Module for Arduino from oddwires.com. Motor driver from this company has the same parameters as two other motor drivers. But it combines two best futures from other two motor drivers. It has a built in heat sink which prevents motor driver from overheating and it costs $12.95 which is the lowest price compared to its opponents.

Bipolar stepper motor has 4 wires which need to be connected to the motor driver. In the Figure 3.2.3.7 4 wires from a microcontroller are also required to be connected to

the motor driver so we can send sequential signals individually to every wire from a stepper motor. Two bipolar motors require 8 I/O ports from a microcontroller. Since we have many other sensors to be connected to the microcontroller we were looking for a better solution to reduced amount of I/O ports required to connect microcontrollers and motor controller. After we did our research we found circuit created by Sebastian Gassner depicted by Figure 10 which is based on the idea that during every step sequence two wires always have opposite polarities.

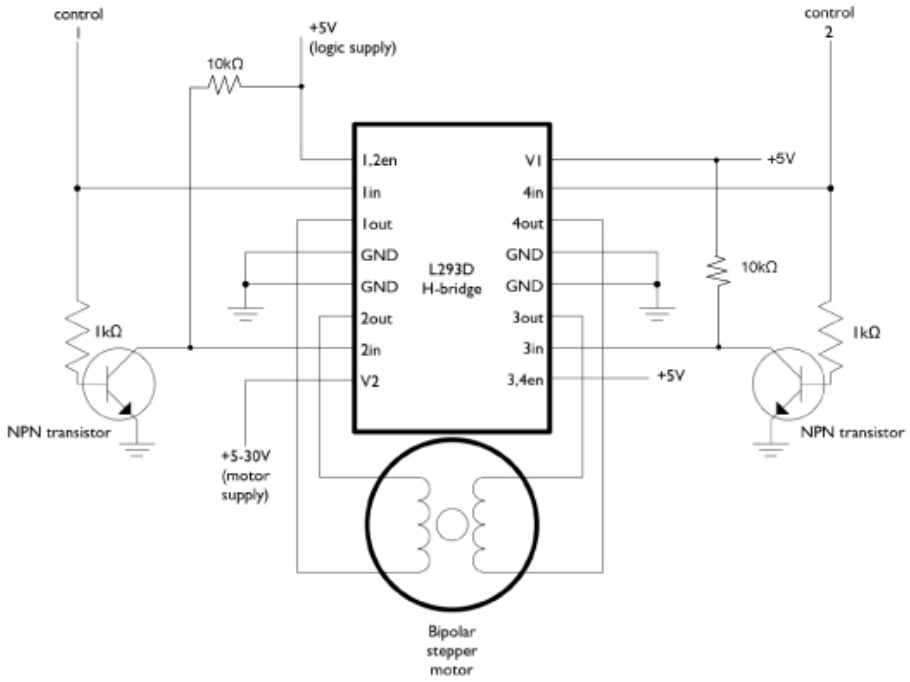


Figure 10 - Two wire bipolar stepper motor connection

This allows us to create a circuit that includes extra switches. Those switches switch polarities of the two wires during every step sequence and it reduces the required amount of I/O ports from a microcontroller for our project from 8 to 4.

The two wires to control stepper motor are definitely a big advantage, however the usage of a motor driver based on L298 chip will allows us to use stepper motor only in full step mode. In Figure 11, we illustrated a full step operation.

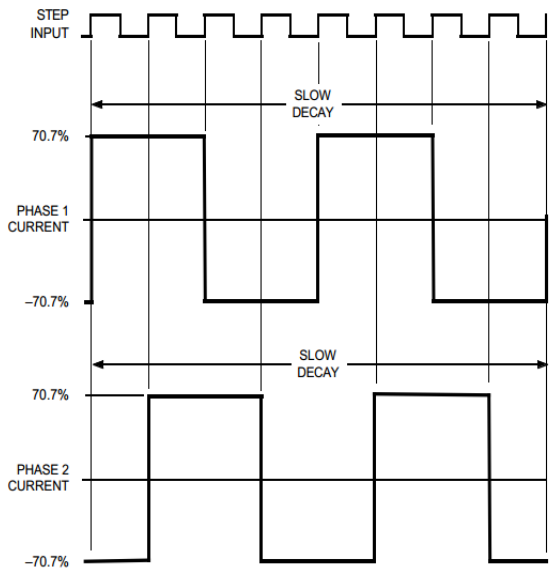


Figure 11 - Full step operation

A stepper motor operated in full step mode during high speed creates certain drawbacks. Couple of them is resonance and vibration. The controller sends a high impulse to one of the coils of the stepper motor and it makes rotor to move one step. Next rotor should stop and wait for the next impulse to move to the next step. But the rotor cannot stop immediately and it moves back and forth before it stops. Such motor behavior results in vibration. Resonance is created when the input motor frequencies from the motor controller match the natural frequencies of the motor. As a result those drawbacks create extra heat in a stepper motor, make the motor skip the steps during high speed and create noise while the motor is in operational mode. During our research, our group looked for a way to eliminate such negative phenomenon of the stepper motor movement. One of the ways is to increase the load on motors to reduce vibration; as a result the rotor has less extra energy to move back and forth. Another more sufficient way to reduce the negative consequences of stepper motor operation is to use microstepping. Eight step microsteps are shown in Figure 12.

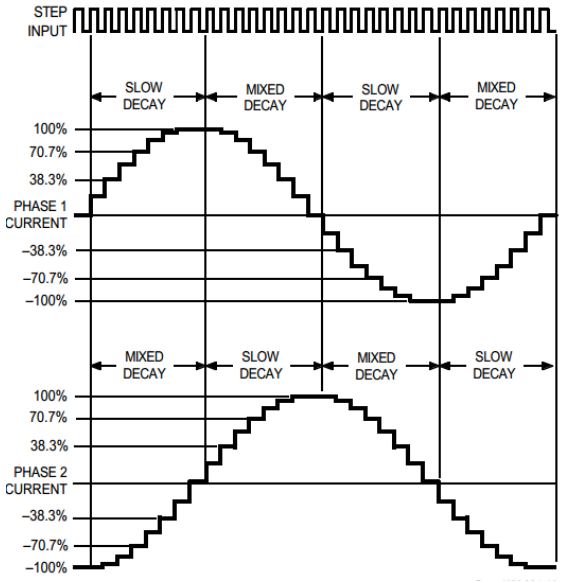


Figure 12 - The 8 microstep operation

Usage of microsteps allows us to have better positioning in our moving system. Due to the fact that the rotor receives impulses which sequence shape is much closer to sin wave compared to a full step mode. It creates a smooth transition from one coil to another. The rotor of the step motor does not make jumps and as a result reduces significantly the noise level and vibration of the motor. This increased accuracy of our positioning system is an important factor for our project. Due to this fact we decided to research for a motor driver that has a microstep feature incorporated into the motor driver.

The first simple stepper motor driver we found was EasyDriver Stepper Motor Driver from the sparkfun.com. This stepper motor driver is capable of driving one 4 wires bipolar stepper motor or a 6/8 wired unipolar stepper motor connected as a bipolar. It has microstepping option in which it can drive motor in full, half, quarter and eight microstep operation mode. It based on A3967 chip from Allegro MicroSystems, Inc. and requires power supply from 7V to 30V. Finally this stepper motor driver can supply from 150mA to 750mA per phase.

The second stepper motor driver we found during our research was Big Easy Driver from the same company sparkfun.com as our first driver. This driver is the latest version of the EasyDriver and it can be used for stepper motors that require more power. Motor drivers become very hot while they are operating and using driver which is suitable for higher operating current and can allow us to operate at lower temperatures compared to the driver with lower operating currents. Current control for this motor driver is up to 2A per phase which is twice as big as a EasyDriver. It can also drive at 16 microstep operating mode and is based on newer chip A4983. Even though price of this motor driver is higher availability of it features makes Big Easy Driver competitive to EasyDriver based on our requirement for the stepper motor driver.

The final motor driver we were considering during our research was A4988 Stepper Driver Carrier with Voltage Regulators from the website pololu.com. As we can see from the name of this motor driver it is based on the newest, compared to the previous two motor drivers, chip A4988. Characteristics of this driver are almost the same as at the Big Easy Driver. It operates from 8V to 35V and can deliver current up to 2A per coil. It also has voltage regulators which allow us to supply any logic device which requires 3.3V or 5V power supply. There are still some disadvantages of this motor driver compared to the Big Easy Driver. Firstly, it does not have an open source as the Big Easy Driver does. Secondly, it does not have mounting holes. However it does have same advantages compared to the Big Easy Driver. One of the advantages is that size of the board is smaller. Another advantage is the price A4988 Stepper Driver Carrier costs $19.95 compared to Big Easy Driver which costs $22.95. In Table 14, we compare all three stepper motor drivers together.

Table 14 - Compares stepper motor drivers

	Easy Driver	Big Easy Driver	A4988 Stepper Motor Driver
Power supply range, V	From 7 to 30	From 8 to 35	From 8 to 35
Output power, V	3.3/5	3.3/5	3.3/5
Microsteps	1/2; 1/4; 1/8	1/2; 1/4; 1/8; 1/16	1/2; 1/4; 1/8; 1/16
Chip	A3967	A4983	A4988
Current control per phase, mA	Up to 750	Up to 2000	Up to 2000
Board size, in	0.6 x 1.9	1.25 x 1.75	0.7 x 1.4
Source	Open	Open	n/a
Price, $	14.95	22.95	19.95

Based on the information from Table 14, we made our decision of what stepper motor driver we are going to use in our project.

3.1.4 AUDIO SYSTEM

When the project Deep RGB was first coming together the idea was tossed around that the player should be able to listen to music while playing. This system would not only play music it would also be used to send alert sounds for different situations such as check and illegal moves. The system would also allow the user to set preferences as to what music they like and the board would play the preferences when the user logged in. Another idea for a feature that arose was the ability for a user to plug in their IPod, iphone or other listening device and play music directly from there.

Since the MCU is more than likely not capable of mp3 decoding a peripheral device will have to be used to make the audio system possible. The MCU that is going to be used is an Arduino Mega 2560 a common MCU used by hobbyists for a wide variety of applications including sound control. In fact the Mega 2560 is so popular for this application that a special audio version of it has been created to make a powerful controller for many audio applications. Since the MCU being used does not have an mp3 decoder on it a shield must be used to convert the information into sound. This shield will take data from the MCU and decode mp3 files outputting the sound through various I/Os usually a 3.5mm headphone jack at the very least. The audio shield will most likely not be able to play the sounds on its own and therefore must be connected to powered standalone speakers or to an amplifier that will then drive speakers. If the project were better funded it would be recommended to use integrated speakers and a small amplifier for the best results.

In order to play music directly from an external device like an IPod an input like a 3.5mm jack would need to be used and the MCU would need to be able to switch the input to the speakers from the audio shield to the input port. Should Deep RGB ever be produced for commercial use this feature could be scaled to include all manner of input ports such as USB, serial connections, SD cards and even blue tooth. For the purposes of this project a simple 3.5mm jack will suffice for the input as to keep it simple. Even so this feature is not of the utmost importance and should it become too difficult to incorporate into the end product it will be cut from the design.

The audio shield must also be able to receive interrupts from the MCU so that it may stop itself in the middle of a sound playback so that it may play an alert sound. If the auxiliary input is included in the final project then the MCU will have to switch the connection to the speakers from the external device back over the mp3 shield then if necessary return the connection to the external device.

3.1.5 LED SYSTEM

To begin designing the LED subsystem that will be in our chessboard we had to decide what type of LED to use. The decision really boils down to two different options single color LEDs or RGB LEDs. The other variables such as mounting, size and power costs all hinge on this first decision.

The benefits of a single color LED are that it has only one input and output; it can be turned on with a single bit of data. However there are some downsides to it such as the fact that it has only one color limits the information that it can give out. It is always the same color for every application it is used for; this makes it a bit less desirable for a chessboard where multiple lines of information need to be presented to the player. Such information as moves available, check indications and move errors.

The benefits of an RGB LED include the fact that it can be changed variably to include a wide range of colors. This is a useful asset in a chessboard since each type of

information can have its own color assigned to it. When each type of information broadcast to the player is color coded it creates the opportunity to make a hierarchy of information sent. For example if a player is in check it could show up as a red line between the piece that has them in check and their king, while moves available to the piece being raised can show up as blue. The downsides to having this ability are the extra programming that will go into it, the program will have to run through several more operations to calculate the LEDS that need to be lit and light them in the right color. The other downside is that RGB LEDs require three inputs which make it difficult to wire to a small board.

After considering the pros and cons of both types of LEDs the group decided that the RGB was the way to go. It would make the board look that much more appealing to the user and would add an extra bit of flare that would set Deep RGB apart from the competition. As an added bonus this decision is what gave the project its catchy name.

This decision led to a few new challenges and questions that needed to be answered. To start off what type of RGB would be used surface mount pin mount etc. Also how would they be powered and would the power be variable to reach the maximum potential of the LEDs or would it be fixed. Finally how would these LEDs be controlled would there need to be multiple boards or just one.

To start off there had to be a decision made on the type of LED used. Pin mounts are inherently bigger than surface mounts which can be a problem with applications that need to be smaller. However they are easier to use than surface mounted units since they can be routed in many ways and do not necessarily be connected to a board.

Surface mounted components are neater and provide a more refined look to a project. They are also thinner which can be a benefit when trying to move objects with magnets through a space. However they must be connected directly to a board to use and that creates a whole new challenge of creating a PCB.

In the end the decision was made to use pinned components. The issue wasn't aesthetic since the components will be hidden it all boiled down to cost. The vastness of project Deep RGB would have made it very costly to create a PCB that would contain all the components necessary to run the LEDs. The larger component size was deemed irrelevant since the overall size of an LED is about 5mm and the magnets being used would have no problem reaching through that amount of space. As stated this decision was made primarily on a fiscal basis and if finances permitted a PCB could be used to make a cleaner product.

RGB LEDs have three inputs which can be made variable to create multiple colors. The three inputs typically have different max voltages as well; usually the voltage controlling red has about half the max voltage as green and blue. There are two possible ways to treat the LED as a variable component or as a fixed component. Thus there must be a decision as to whether the LEDs should change or simply be turned on or off.

The RGB LEDs do not have to be completely variable to work in Deep RGB. There are only three possible uses for the LEDs movement indication, check calls, and errors. Thus the LEDs can be fed three voltages and by turning each one off and on seven colors can be made. This is sufficient for a chessboard but lacks the prestige of having a chessboard that can display a greater variety of colors.

The problem with making an LED change its color is that it is in fact impossible with a regular output from a MCU. Since the MCU is digital and the LED input needs to be analog this creates a problem. There is a rather elegant solution to this problem called pulse width modulation or PWM. To do this a PWM output must be used from the MCU so that it can change the duty cycle of the inputs of the LEDs. Consideration must be made for the clock speed of the MCU that will be used due to the fact that PWM works better when the controller is able to change the output values quicker. The faster the controller can change the outputs the more control it has over the duty cycle and thus it can create more colors than a slower MCU.

After considering both ways of powering the LEDs it was decided that Deep RGB deserved to be a bit flashier and therefore it should have variable colors. Some alternative applications of this ability that have been discussed but not officially added are the ability for players to choose their own colors so as to diverge from the ordinary black and white. Another possible use of this ability is to create a randomizer that will change the colors that will be displayed when showing possible moves so as to liven the board up a little.

It should be noted that since the decision has been made to use RGB LEDs the power consumption of the board will be greater. An average LED has a max forward current of thirty milliamps and the RGB essentially has three LEDs in it so its max power consumption is ninety milliamps. This should be accounted for when deciding on a power source and is covered in the corresponding section of this report. To gain a better grasp on the power used by the LEDs a chart is provided below the values have been taken from a blue LED (since blue requires more power) model YSL-R1047B5D-D2 and an RGB model YSL-R596CR3G4B5C-C10. Both LEDS are made by the China Young Sun LED Technology Corporation this will help ensure that both LEDS being compared are similar in both quality of manufacturing and materials used. Looking at Table 15 it becomes apparent how much of a difference in power consumption there is when an RGB LED is used over a single color LED.

Table 15 - Power Consumption of LED types

LED	V1	A1	V2	A2	V3	A3	Watts
Monochromatic	3.4V	30mA	N/A	N/A	N/A	N/A	102mW
RGB	3.4V	30mA	3.4V	30mA	2.2V	30mA	270mW

This is not a great amount of power but the cumulative effects of multiple LEDs can be significant as shown in Table 16 below. This must be accounted for when selecting a power supply and is discussed there. Another issue is that the MCU cannot supply this

much power to the LEDs so one of two things must happen. The LEDs could be multiplexed so that only one is turned on at a time and the display is quickly written over and over at such a rate that the human eye cannot notice. The other solution would be to include transistors to amplify the power from the MCU and thus power all the LEDs at once. The group has yet to decide on this matter and it will be more thoroughly discussed in the design section of this report.

Table 16 - Example Power Ratings for all LEDs in Project

	Power Per Unit	Total Power For Project
Monochromatic LED	.102W	6.528W
RGB LED	.270W	17.28W
Difference (RGB- Mono.)	.168W	10.752W

One of the more challenging problems of creating an 8X8 array of RGB LEDs is how to control them all. Since each LED has 3 inputs that turns comes to 192 inputs in all. This is a problem since the average MCU does not have that many output pins. This can be remedied by using multiple MCUs but that would raise the cost of the project greatly and frankly is a rather inelegant solution to the problem. The best way to do this is to use shift registers to reduce the code to either three or four digital outputs.

To control the display with three bits of code from the MCU four shift registers are required. There is one register each for red, green, and blue as well as one for the anodes. This means it requires a total of thirty two bits of information to light the entire LED display. The actual design will be covered in the hardware section of this report. Using these shift registers the information can be passed down each one from a single output and stored in the appropriate place. The other two outputs needed from the MCU are the shift clock and the store clock. The shift clock controls when the information is shifted down while the store clock controls when the information is stored and thus outputted. There is one flaw with this design and that is that the PWM is controlled by the store clock which means every time the display refreshes which is about fifty times a second the registers are told to store the information again. This is cuts down on the need to use an enable display bit but creates the risk of having data errors if the data output inadvertently changes so will the display.

There is a way to subvert this problem and that is by creating another output that will control the LED display. This output would be an output enable which just like the name sounds enables the output and thus powers the LED display. PWM is controlled via this output which will turn the display on and off fifty times a second creating the illusion that the LEDs are on continuously. The difference however is that it is not telling the registers to collect new data so if the data input should become corrupted in any way it will not affect the current display.

3.1.6 LCD DISPLAY SYSTEM

Another subsystem research for our project is researching and choosing a LCD display to satisfy all our needs in such system. The basic purpose of LCD in our chess board is to supply user with all possible information he or she needs. This information might include displaying user name and password during the connection to the internet and login to the server. It can also display status of connection such as system connected or not to the server. We can also display information about previous or current moves of chess pieces. Last and but not the least option is that display can show us current status of the game played such as user vs. computer or user vs. user and display can have setup menu where we can pick a certain light and music mode for our chess game. Finally, choosing a correct display should bring to the balance such factors as usability, cost of the system and technical level required for system setup.

One of the ways to display information is to use LED arrays with different colors LED. Each color can inform a user about the process going in the system. For example, red color can represent that chess board system is connected to a power supply; light can stay on if there is a power in the supply or light stays off if there is no power in the system. To show connection of the board to the server we can use green color LED. This LED stays blank when the board is not connected, it can also flash with the constant time frame then chess board is connecting to a server and stay green when the board has established connection with the server. We can use blue color LED to inform user about different errors in the chess board by using different flashing intervals for each separate error.

The simplicity of an LED display system is a big advantage and big disadvantage of this system at the same time. Advantages are that this display requires low technical level and short time to setup; this display also has lowest cost level. On the other hand, it would have lack of ability to show some significant information to a user such as showing some information characteristic for example password, user name or displaying different game information and game status. To summarize the LED system display we used Table 17.

Table 17 - Advantages and disadvantages of LED system displays

Advantages	Disadvantages
Low cost	Lack of ability to display characters
Low power consumption	Need to study first to be able to understand error massages
Easy to implement	
High level of brightness	
Reliability	

Based on this information, the display we need to use should be able to show system status and display some text information about game status. Next our option for display which can display text characters was LCD display. Our research shows that based on the complexity of the arrangement of crystals in LCD display, there are three main types

of LCD displays to be considered. Each feature makes them different from each other and enables them serve different applications.

The first type of LCD display considered was a segment LCD. After we researched this type of display we found that it has a few key disadvantages when compared to other types of LCD displays. Since each character there is implemented by using from 7 to 14 or 16 segments it creates a limitation in displaying all information our user will need. The segment LCD display is also available on a marked those days but it not created to display different characters. Some of them are used only to image numerical information which is definitely not enough for our requirements. The other can display numerical, alphabetical and symbolic information but those displays only have predetermined and limited amount of symbolic information for example the display in a thermostat for air condition. There is a possibility to order customized segment LCD but it will increase cost of such display significantly. Since for our project we have certain budget limitations cost of customized display makes it too expensive for our project. Advantage of segment uncustomized display is easy implementation in the system, low power consumption and lowest cost in family of different types of LCD displays. Characteristics of segment LCD we collected in Table 18.

Table 18 - Advantages and disadvantages of segmented LCD displays

Advantages	Disadvantages
Simple programming process	Lack of characters display ability
Low cost	Small amount of symbols per line
Low power consumption	Lack of preinstalled controller
Reliability	
Easy to integrate	

The next type of LCD display we researched was a dot matrix LCD. The versatility of this type of display is bigger compared to segment LCD because dot matrix LCD uses block of 5 by 7 dots to create a character which can be a number, an alphabetic letter or any symbol that you can draw using this matrix of dots. Dot matrix displays also have built in controller in most of the cases it is Hitachi HD44780. Wide usage of this controller in different projects allows us to find all necessary information on how to implement this display in Deep RGB project. This gives an advantage to this display compared to the other options. Even though there is many information on how to setup other types of display usage of dot matrix display would significantly facilitate its implementation in Deep RGB.

The power supply for most of dot matrix displays is regularly 5V. This beneficial aspect makes it very convenient to implement dot matrix display together with a microcontroller by using the same voltage level for both of them. We also did not plan to use Deep RGB in any harsh environments; the board we are creating should be operated only indoors. However taking Deep RGB to play outside doesn't harm dot

matrix display since they have operating temperature range from 0 to 50C. Another useful feature for us that can be found in dot matrix display is backlights. It increases readability of display in harsh light environment such as bright light or when there is not enough light to read display. We post in Table 19 the advantages or disadvantages of dot matrix LCD displays

Table 19 - 3.2.8.3 Advantages and disadvantages of dot matrix LCD displays

Advantages	Disadvantages
Low power consumption	Unable to use for graphics
Low cost	Limitation on dot matrix size
Integrated controller	Display size limitation
Preinstalled characters	
Ability to create own characters	
Wide variety of sizes and colors	
Easy to find information on how to incorporate display with microcontroller	
Backlights	

A graphic LCD was another option for Deep RGB project. This type of display has significantly increased connection complexity compared to the previous types of display we were considering. It consists of literally thousands of pixels which are used to create images in monochrome or color. This type of display is most moderate compared to the other types of LCD displays and it is typically used in high tech projects which require high resolution graphics and images. It is very popular and as a result, graphic LCD displays have a lot of available information on how to incorporate them with most popular microcontrollers. However the cost of graphic displays is significantly higher compared to the dot matrix displays even if they have the same physical size. There is also a big difference in power consumption for graphic display compared to character based display. Since graphic display has more components than character based it uses more power than alphanumeric displays. Just like alphanumerical displays, graphic displays have a backlight. Most of graphic displays need 5V power supply which also makes them easy to use with microcontrollers. Operating temperature for most graphic displays is similar to operating temperature of alphanumerical displays from 0 to 5C. Information about graphic displays is summarized in Table 20.

Table 20 - Advantages and disadvantages of graphic LCD display

Advantages	Disadvantages
Ability to display graphics and images	High cost
Wide variety of different sizes	High power consumption
Reliable	Hard to program
Backlight	

We didn't even consider touch screen types of LCD display since usage of such display will unnecessarily increase complexity of incorporating this display in Deep RGB project. Because the LCD is used more like an addition to the project concept, an increment of

complexity of the display will not facilitate faster creation of the first prototype. The cost of this display is also much higher compared to the amount we allocate for the display in our budget.

After all previous information where we considered advantages and disadvantages of different types of display system, we decided to use dot matrix LCD display. Those modules are not the latest generation modules with full colors and large display size but characters-based module still used in a large variety of commercial equipment and DIY projects where fewer requirements for displaying information with simple messages make these displays a perfect choice. Even though character-based modules have certain limitations they can still be found in wide variety of lengths and widths. Standard line sizes are 8, 16, 20, 24, 32 and 40 characters with 1, 2, or 4 lines. Simpler one line display wouldn't be enough for our project because minimum requirement for Deep RGB display was to have the ability to see previous line. As a result for our project we need to have at least a 2 line LCD and minimum 16 characters long due to the size of longer information we need to display.

The first part we picked for our display was 16 x 2 Character LCD (Parallel Interface) from pololu.com. It is a simple two line display with 5V power supply which uses interface for HD44780. One negative side of this display is that it doesn't have a backlight which might be a big issue for us since we have RGB backlighting on a chess board and users might try to play in a dark which can make the display uncomfortable to use. The price for this LCD display was low $9.95. The same size and type display only with a backlight costs couple dollars more (total price $12.95) but it gives us ability to fully use it in the dark. The supply current for an LCD without backlight is also only 2mA while for versions that include a backlight only need an extra 120mA in order to operate.

Next we chose LCD16 x 2BL-3V3 from futurlec.com. This is regular LCD display with size 16 x 2 with a backlight. It has build in controller, low 3.3 V power supply and 1/16 duty cycle. The price for this LCD is $8.90 which is almost 30% cheaper than our previous choice. Size of display 16 x 2 meets our minimum requirements for our LCD module on the other hand bigger display will increase cost of the system. The power supply for this display is also 3.3V which is different from 5V power supply used by most popular microcontrollers.

Our third choice was BLUELCD20x4BL with 20 x 4 characters display which makes this display more than double size of the two previous choices. This display looks more modern since it has blue color of characters. It also includes backlight and has 5V power supply. Cost of this display is twice as big compared to our second choice and it costs $16.90.

Finally after doing more research and discussions with our team we decided to choose smaller LCD which should include the same blue color as a previous 20 by 4 characters display, has lower cost, backlight and needed 5V power supply so we can easily incorporate it with our microcontroller. It was BLUELCD16x2BL 16 characters per line display with two lines. Color of this display is blue and it has backlight. It also needed 5 V power supply. Finally its operating temperature ranges from 0C to +50C is completely suitable for indoor environment.

3.1.7 POWER SUPPLY SYSTEM

One of the last phases of our research constituted the research of the power supply module. Our team saw many available solutions and options offered on the market. The goal of the power supply research was to determine the best ratio of quality, reliability and price. During power supply research, we identified two main parts; one is the different ways to supply power to the board and another is the energy distribution between different board components.

In the first part we considered different options to supply power to our chess board. One of the options was to supply power to the board through batteries. Ability to use batteries has certain advantages. It will allow us to use the board in any place where there is no wall socket available. Two types of batteries were considered; NiMH (Nickel Metal Hydride) and Lithium (Li-ion). These two types of batteries are mostly used in small and medium size projects.

NiMH batteries were popular couple years ago and because this technology has just passed its mature peak of development stage, we knew very well their strong sides and their weak sides. One of the advantages of these batteries is that you can recharge them as many times as you want. Their price compared to Lithium batteries is also cheaper. You can get one AA size battery 1.25V for around $2 for 2500 mAh of charge capacity. The disadvantages of NiMH batteries are that they have high rate of discharge while they are not in use and it also takes them about 10 hours to charge depending on different conditions which can create limitation of time when you can use chess board powered by those batteries. For example on a Web site www.pololu.com we found rechargeable NiMH AA batteries. These batteries had 1.2V voltage output, 2200 mAh capacity current rate. For our project we need at least 10 of those batteries to create 12V power output for our stepper motors. The rest of information about this battery we enter in the TABLENUM.

Lithium batteries become more popular for usage in medium size projects compared to NiMH. It happened because they have more advantages than NiMH batteries. One of the Lithium batteries advantages is that they have more power density. Another advantage is that lithium battery which has same energy capacity rate weights 20%-30% lighter than NiMH battery. Even though the price of the lithium batteries is twice as high as NiMH batteries, the ability to use battery of much lighter weight is considered by our

team as a big plus toward lithium batteries usage. They also are environmentally friendly since they do not have toxic material as NiMH batteries. One of the lithium batteries examples was Polymer Lithium Ion Battery from the web site starkfun.com. This battery has current capacity rate 2000 mAh and it has nominal output of 3.7V. To create the minimum 12V output we need to have at least 3 of those batteries. The rest of the information we put in the comparable Table 21.

Table 21 - Compare of NiMH and Polymer Lithium

	NiMH	Polymer Lithium
Nominal capacity, mAh	2200	2000
Nominal voltage, V	1.2	3.7
Minimum amount of batteries to supply 12V	10	3
Weight of each battery, oz	0.97	1.27
Total batteries weight, oz	97	3.81
Price for each battery, $	2.15	16.95
Total price, $	21.5	50.85

From Table 21, we can see usage of NiMH batteries gives us more power and it is about twice cheaper than usage of lithium batteries. On the other hand usage of lithium batteries adds much less weight to our chess board whereas use of NiMH batteries makes the board much heavier. Overall, usage of batteries in our design can create some portability for our chess board project. However our first prototype would not be very portable and the incorporation of batteries in our project will add extra weight to the chess board and also take significant amount of money from our budget. Due to all these facts, we decided to not incorporate usage of batteries in our project and hold this idea as a future upgrade option for the next generation of Deep RGB systems.

The next option to supply power to the chess board was to incorporate a generator in Deep RGB system. Generator would generate electrical power from transferring muscular energy of rotation pedals to electrical power. We got this idea by trying to improve time quality when people spend time playing chess so they can also exercise by rotating generator to provide enough electrical power to play chess.

During our research on power generation option we found couple of available solutions on the market and also some information on how to create power generating machine by ourselves. Available product we found was Powerplus Gazelle – Pedal powered PowerBank Generator from web site amazon.com. This generator costs $238.99 and it can supply 12V DC electrical power. Since the cost of the generator was significantly bigger than it was allowed by our budget we decided to look for DIY option. After we did some research about it we realized that cost of pedal generator even if it were self made would still be around $230 based on the information from the [10]. It also adds extra complexity to implementation of our project. Since this option was not a priority for our project we decided not to consider the future development of this option. But it might be a possible option for the next generations of Deep RGB system.

Our last option to supply power to the chess board was to use 110V AC wall outlet. This option eliminated most of the disadvantages from the other two power supply options. It did not require extra money from the budget to implement it. It also does not add extra weight to the Deep RGB system. Since most of the time Deep RGB system has indoor usage it is easy to connect it to the wall outlet in any room or in case of outdoor usage near the house user can use an extension cord.

After we completed the research about power supply options to the chess board our next step was to research different methods of power distribution to the different components in the Deep RGB chess board. For many years creation of power supply has not undergone many changes. It still consists of some major units such as transformer, some diodes bridge and couple voltage regulators. In spite of the apparent simplicity it is very important to create an efficient and reliable power supply and that all elements of the supplying system work efficiently and for a long time without any hazards and magic smoke.

First we need to determine the minimum amount of voltage for each component in our system. To do so we created Table 22 were we illustrated minimum required voltage needed to supply each component.

Table 22 - Minimum voltage requirements from the power supply for each element

Element	Voltage DC level, V
Microcontroller	From 7 to 12
RGB led lights matrix	5
Hall effect sensors	8
Display	5
Led backlight for display	5
Stepper motor driver	From 8 to 35
Stepper motor	12
Electromagnet	12
Wi-Fi unit	3.3
SD card reader	3.3
Music unit	3.3

From Table 3.2.9.2we can conclude that we have at least 3 or 4 DC voltage levels. One way to deal with all of those different voltage levels is to buy power AC/DC adapter so it can transform 110V AC power to 12V AC power use rectifier to make 12V DC power. After that we can use different voltage regulators to switch to the different DC voltage levels. Another way is to buy transformer which can transform 110V AC to 24 V AC so we can have higher voltage level for our stepper motors and after that we can lower power level for the rest Deep RGB components.

There are many available AC/DC power adapters available on the market. After we have done our research about it we chose two different power adapters which suite our requirements better than other adapters. First power adapter we chose was external

power supply from the web site hobbyking.com which can use 110V and 240V power input. Usage of such adapter has certain advantages. One is that 110V/240V adapters have ability to works in USA and in Europe. Another advantage is that those adapters have low price due to its wide usage. This adapter delivers constant DC 15V and 5A output. That should be enough to supply our stepper motors, electromagnet and the rest of the electronics. External usage of this adapter was considered by our team as an advantage. This can save space inside the chess board and it is also safer for our project to keep power adapter outside of the Deep RGB board in case if any hazard happened to the adapter it does not damage the board. This adapter also has low price of $9.49 and it is CE approved which gives us assurance of quality of this adapter.

Output of 15V from the adapter enables us to use it directly to supply stepper motor drivers. To supply the rest of electronics we need to use voltage regulators to step-down voltages to different levels such as 12V, 5V and 3.3V.

The second power supply adapter we were considering was Adapter Power Supply Balancer Charger from the web site ebay.com. The price of this adapter with shipping is $8.87 which is cheaper than our first choice. It has slightly different characteristics than the other adapter as well. This adapter has 110V/240V power input and the output of this adapter is constant 12V DC and 5A. From one point of view it is preferable for our project the usage of this power adapter since we had to use fewer amounts of voltage regulators. We can supply directly stepper motor drivers, microcontroller and electromagnet. On the other hand, this adapter has smaller voltage level and it might supply less energy to the stepper motors. After discussion with our team we decided to use adapter with 12V and 5A output. Due to the fact that we have 5A and if we need to have higher than 12V power level to supply stepper motor driver we can use step-up voltage regulator to increase voltage level to required amount. Also this adapter on the output has DC male connector 5.5 by 2.5mm which can be easily plugged into 5.5mm female connector which is available to mount on the PCB board.

The input voltage from the wall outlet to our external power adapter would be 110V 60Hz. On the output from the power adapter we receive is 12V 5A DC. After we do some current limitation, we can supply it to the stepper motor driver, microcontroller and electromagnet. Next we decided that the initial output of 12V has to be dropped to 5V and 3.3V. One 5V DC voltage regulator needs to be used. The first 5V DC voltage regulator we were considering was the AP150650T5L-U from digikey.com. This regulator is manufactured by Diodes Inc. The input voltage for this regulator is from 4.5V to 22V, which is suitable for us since we have 12V input from the adapter. Output has a fixed type voltage 5V and 3A current and the operating temperature range is -20C~85C and the highest unit price is $2.98. Another possible regulator we were considering was LM2576T-5.0/NOPB from digikey.com and manufactured by National Semiconductor. This regulator has mostly the same characteristics as the first regulator, the only difference is it has operating range temperature of -40 C to 125 C and unit price is $2.83. For our project usage of voltage regulator from National Semiconductor is preferable

because it has higher operating temperature range and we have two stepper motor drivers which have high operating temperature.

When we were choosing 3.3 V voltage regulators, we were using the same method as for 5V regulators. Our first choice was LM2576T-3.3/NOPB from digikey.com and manufactured by National Semiconductors. It allowed for input voltage from 4V to 40V, which is suitable for us since we can apply 5V or 12V. The output voltage is fixed and has 3.3V. It also has a 3A current output which could be lowed if necessary. The mounting type of this voltage regulator is through hole as the additional heat sink could be added in case the voltage regulator exceeds its operating temperature range which is from -40 C to 125 C. Unit price for this regulator is $2.83.

Another regulator we were considering to use for 3.3 V DC voltage regulator was the LM2576T-3.3GOS-ND, from digikey.com and manufactured by ON Semiconductor. This voltage regulator has very similar parameters as the 3.3 V DC regulator from National Semiconductor, but parameters differed when it came to the input voltage range starting from 7V to 40V which meant we could only use it with a 12 V input. It also has a different price. $2.05 per unit. However, since we were considering using a 3.3 V DC voltage regulator only with 12V input which is more than enough for our second choice voltage regulator and it also has cheaper price.

3.1.8 WIRELESS DATA TRANSMISSION SYSTEM

3.1.8.1 WI-FI

Wi-Fi indicates a device or product that is capable using radio waves in order to transmit and receive data via a computer network. Wi-Fi devices are certified by the non-profit Wi-Fi Alliance and are centered on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards set forth in 1985. A Wi-Fi network is similar to any conventional Ethernet network but is categorized into a variety of subsections based on the signal's transfer rate and operating frequency. The 802.11 categories are as followed:

- 802.11a has an operating frequency of 5 GHz with an indoor range of approximately 100 feet (depending on antenna size). It is capable of a data transfer rate of 25 Mbps to 54 Mbps by using an orthogonal frequency division multiplexing method that encodes digital data on several carrier frequencies.
- 802.11b has an operating frequency of 2.4 GHz with an indoor range of approximately 100 feet (depending on antenna size). It is capable of a data transfer rate of 6.5 Mbps to 11 Mbps by using Complementary Code Keying (CCK) modulation method. This method allows for a higher data transfer rate while but lowers the overall range due to narrowband interference. This standard seems to be the most affordable choice, but lacks some long range capabilities.

- 802.11g has an operating frequency of 2.4 GHz with an indoor range of approximately 100 feet (depending on antenna size). It is capable of a data transfer rate of 25 Mbps to 54 Mbps by using an orthogonal frequency division multiplexing method that originated in the 802.11a standard. Since this Standard uses the same operating frequency as the 802.11b standard, most devices are capable of accepting both types of wirelss signals.
- 802.11n has an operating frequency of 2.4 GHz and 5 GHz with an indoor range of approximately 160 feet (depending on antenna size). It is capable of a data transfer rate of 54 Mbps to 600 Mbps by combining multiple antennas and using a Spatial Division Multiplexing (SDM) method that combines multiple signals allowing the device to reach high transfer rates.

These standards allow us to pick from a wide array of Wi-Fi capable devices to implement in DeepRGB. These devices will allow our system to wirelessly connect to the chess algorithm server through a secure and strong connection. Wireless connections are usually secured by a security protocol to ensure that the data isn't corrupted or stolen by hackers. There are various types of these security protocols and have evolved to provide a virtually undecryptable connection. The most commonly used security protocols utilized in modern wireless routers are WEP, WPA and WPA2 and since most wireless routers have a long cycle lives, the wireless module in DeepRGB needs to be compatible with these types of protocols.

3.1.8.1.1 WIFLY GSX BREAKOUT BOARD

The WiFly GSX Breakout board as shown in Figure 13 offers a radio module that is a complete embedded Wireless Local Area Network (WLAN) device. This device only requires 4 pins and can transmit serial data to and from the Arduino microcontroller using its UART interface. The two pins used in UART for serial transmission are the TX and RX pins that can be used to write data to the microcontroller's flash memory.

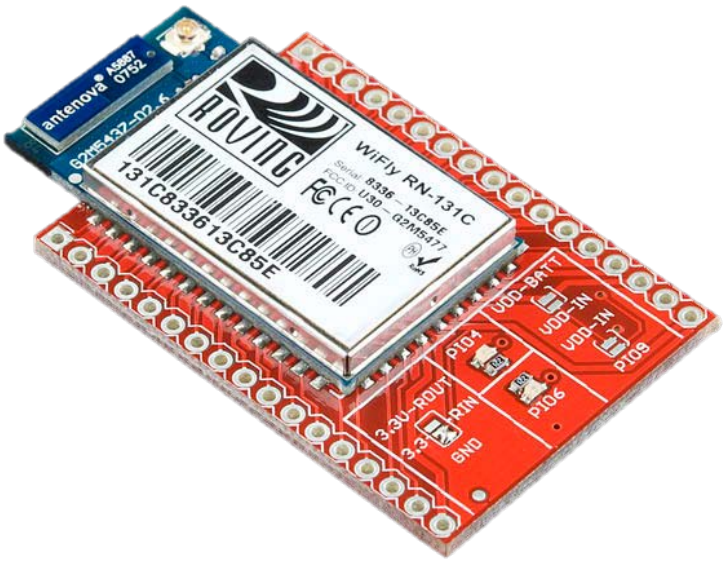


Figure 13 – Sparkfun's WiFly GSX development board reprinted with permission from Sparkfun.

It is capable of connecting to the 802.11b/g standard and can access the most used types of security protocols. It utilizes a embedded Wi-Fi module designed and built by Roving Networks. The Roving Networks RN-131G's features include:

- Qualified 2.4GHz IEEE 802.11b/g transceiver
- High throughput, 1Mbps sustained data rate with TCP/IP and WPA2
- Ultra-low power - 4uA sleep, 40mA Rx, 210mA Tx (max)
- Range: Up to 100m
- On board ceramic chip antenna and U.FL connector for external antenna
- 8 Mbit flash memory and 128 KB RAM
- UART hardware interface
- 10 general purpose digital I/O
- 8 analog sensor interfaces
- Real-time clock for wakeup and time stamping
- Accepts 3.3V regulated or 2-3V battery
- Supports Adhoc connections
- On board ECOS -OS, TCP/IP stacks
- Wi-Fi Alliance certified for WEP-128, WPA-PSK (TKIP), WPA2-PSK (AES)
- FCC / CE/ ICS certified and RoHS compliant.
- Industrial (RN-131G) and commercial
- (RN-131C) grade temperature options
- Price: $84.95

The WiFly GSX Breakout board allows us to connect to our server from extreme distances. Unlike with the XBee about to be discussed, this doesn't require the server to be in the vicinity of DeepRGB.

3.1.8.2 XBEE

Xbee embedded modules deliver wireless end-point connectivity to devices, such as microcontrollers. There are numerous different kinds of Xbee modules that can provide the necessary connection for the microcontroller unit in DeepRGB, but most of them share similar pin-outs thus making the modules interchangeable depending on our environment and needs.

All Xbee modules use the IEEE 802.15.4 standard networking protocol and provide both Point-to-Multipoint (P2MP) and Peer-to-Peer (P2P) networking solutions. The modules are designed for application requiring low latency, low power and reasonable transfer rates that utilize Zigbee centered protocol. Zigbee is an 802.11 alternative and follows the IEEE 802.15.4 standard that offers an affordable, power efficient, open wireless mesh networking environment as shown in Figure 14.

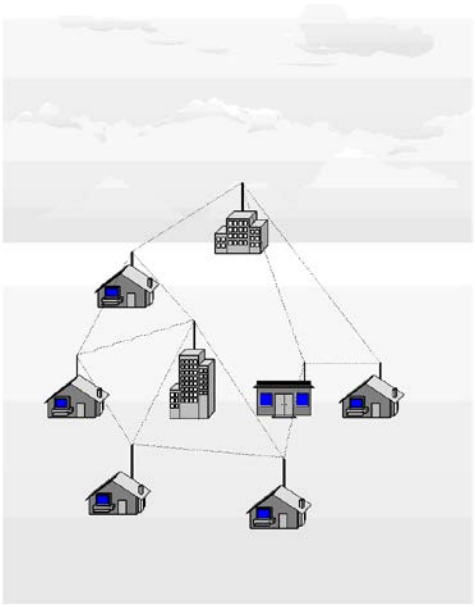


Figure 14 – A representation of an open wireless mesh network reprinted with permission from Wikipedia.

This standard is known to be exceptionally power efficient and makes it desirable in most embedded systems. The mesh system ensures that data can be transferred to an out of range device by using one or multiple Zigbee radio devices in order to relay the information to the required destination. The data gets routed by nodes that receive and forward data from each other, thus ensuring that the data will always find the quickest route to the destination. If a node becomes unavailable, the protocol used will seek out

another route in order to maintain the connectivity between the two or multiple peers needed.

Most modules that incorporate the 802.15.4 standard dictate the use of Direct Sequence Spread Spectrum (DSSS), a modulation procedure where the carrier signal inhabits the full spectrum of the bandwidth. The data signal gets multiplied by a noise signal with a much higher frequency than the original data signal. This entire signal gets multiplied once more and gets deconstructed at the receiving end. This also provides the module with a built in resistance to jamming, ability to share a single channel with multiple users and fixes any timing issues that may occur during transmission

There are numerous Xbee modules that get categorized by their specifications. The following are just a few examples of the modules available on the market today:

3.1.8.2.1 XBEE 1MW CHIP ANTENNA - SERIES 1

The Xbee series 1 module is the simplest and easiest to work with devices for a P2P connection (but is also capable of P2MP). These have limited mesh networking capabilities but make up for that in the ease of setup and flexibility.

The series 1 is capable of connecting to virtually any device with a serial port and has the following features:

3.3V @ 50mA

250kbps Max data rate

1mW output (+0dBm)

300ft (100m) range

Built-in antenna

Fully FCC certified

6 10-bit ADC input pins

8 digital IO pins

128-bit encryption

Local or over-air configuration

- UART hardware interface
- AT or API command set
- Price = $22.95

As seen in the summary above, the Xbee isn't quite as fast as a Wi-Fi connection, but its power consumption is a fraction of the amount needed in a Wi-Fi module. Its range

surpasses large scale Wi-Fi modules and has a small form factor which is desirable in most embedded systems.

3.1.8.3 BLUETOOTH

Bluetooth uses low-power radio waves to transmit data over short distances on a frequency of 2.45 GHz. This frequency spectrum is also known as the Industrial, Scientific and Medical devices (ISM) band. In order to prevent interference with all the other devices within this band, Bluetooth devices tend to send out weak (approximately 1mW) signals. Bluetooth signals don't require a clear line of sight in order to connect to a device; this causes it to be exceedingly popular in small embedded systems. It has the capability of connecting up to eight different devices all together without causing any interference with one another by a method called Frequency-Hopping Spread Spectrum (FHSS) that quickly swaps the carrier between seventy-nine different frequency channels using a classification recognized by both the receiver and transmitter. This technique also ensures that most Bluetooth devices will not interfere with one another due to the constant switching of carrier frequencies.

3.1.8.3.1 BLUESMIRF SILVER

The BlueSMiRF silver is meant to replace serial cables with a wireless solution in most embedded devices. It uses an RN-42 Bluetooth module capable of a 60 foot (18) range from the module while providing a low power and fast data transmissions. It is capable of using the Arduino's UART pins in order to access the microcontroller's flash memory. Further specifications include:

FCC Approved Class 2 Bluetooth Radio Modem

Extremely small radio - 0.15x0.6x1.9"

Very robust link both in integrity and transmission distance (18m)

Hardy frequency hopping scheme - operates in harsh RF environments like WiFi, 802.11g, and Zigbee

Encrypted connection

Frequency: 2.4~2.524 GHz

Operating Voltage: 3.3V-6V @ 45mA

Serial communications: 2400-115200bps

Operating Temperature: -40 ~ +70C

Built-in antenna

UART hardware interface

Price: $39.95

The BlueSMiRF is incredibly small compared to the other previously discussed wireless solutions, but it lacks the range needed to keep DeepRGB somewhat mobile. Nonetheless it needs to be considered against the two other options since it is very power efficient and has the capability of decreasing the overall size of the build.

3.1.8.4 WIRELESS MODULE SELECTION

With the numerous ways of wirelessly connecting the microcontroller unit to a server, Table 23 compares the specifications of the summaries discussed in the above sections.

Table 23 – Comparing the three wireless connectivity devices in order to make a final decision.

Device Name	WiFly GSX	XBee Series 1	BlueSMiRf Silver
Data transfer rate	1Mbps	0.24Mbps	0.1Mbps
Range	100m	100m	18m
Required input voltage	3.3V	3.3V	3.3V
Required input current	210mA	50mA	45mA
Microcontroller interface	UART	UART	UART
Price	$84.95	$22.95	$39.95

The BlueSMiRf Silver Bluetooth option cannot be used for our project due to its limited range. Using Bluetooth would require us to have a computer within 18m to provide the necessary chess piece movement updates and that would end up being a hassle. Both the XBee Series 1 and WiFly GSX module have the same wireless connectivity range, but the Wi-Fi option is preferable due to its higher data transfer rate and wouldn't require a computer to be within 100m. Using multiple XBee Series 1 modules to create a mesh for longer range would end up being inefficient, so the choice of using the WiFly GSX as our development solution was made. The RN-131C wireless module used in the WiFly GSX will be incorporated into the PCB design when the development process is completed.

3.1.9 PRINTED CIRCUIT BOARD

In order to bring all of the modules together and ensure connectivity between them, a PCB would be the ideal choice, but can be quite expensive. There are many options that allow us to create our main circuit board such as:

3.1.9.1 STRIPBOARD

A popular and cost effective method of connecting circuits is using a strip board. Strip board is widely used during prototyping and with good soldering techniques can provide a professional looking circuit board with all of the components on one board. The name stripboard automatically explains the layout of the board, it is made up out of strips of copper pads that conduct electricity. Most of the devices needed for DeepRGB have through-hole counterparts, except the MCU. Since the MCU uses 100 surface mounted pins, a strip board is nearly impossible to implement into our designs.

3.1.9.2 PERFBOARD

Perfboard is a board used to prototype circuit boards. It is very similar to strip board, except it is made up out of small squares of copper instead of strips. These copper pads canbe locatedon both sides of the board in order to utilize both sides for prototyping. But perfboard suffers the same downfall as stripboard; surface mounted components cannot be soldered onto their perfboard due to their small size.

3.1.9.3 PCB

Since the previous options have been ruled out due to their size and impracticality, a PCB seems like the best choice for this project. In order to keep the size of the physical chess board small, a PCB would be ideal since it allows us to use as many surface mount components as possible. Choosing the option of PCBs also allows us to gain more knowledge into professional circuit board making. PCBs need to be designed by using applications such as Eagle Cadsoft, ExpressPCB and DipTrace. Using these applications also allows us to move around our modules and components in order to achieve the smallest circuit board possible. PCB manufacturers tend to set their prices on size, so having the smallest circuit board will not only save space, but will lower our overall costs as well. Like perfboard, PCBs have the possibility to utilize both sides of the board. This can make the circuit board even smaller and give the project a more professional looking design.

3.1.10 SERVER SOFTWARE

The first part of designing the software side of the system was in fact choosing the hardware that would run the software. In that particular decision, there were only two options, running the chess engine on-board, using either the single main processor or adding another processor to handle only that, or offloading the chess engine and its related functions entirely to a separate server.

The benefits of running the chess engine on-board included having one singular piece of hardware that contained the entirety of the project, letting us focus on one physical thing and allowing tighter integration of the algorithm with the board itself. The

downside of this was that any on-board processor was going to be severely underpowered compared to the highly powerful CPUs in modern PCs and servers, and chess engines are entirely processor limited operations which left any on-board solution very slow at computing moves relative to the speed capable otherwise.

Alternatively, offloading the computationally intense portions of the code to a separate server allowed us to compute deep searches of the chess problem space in a fraction of the time required by an on-board solution. The powerful, multi-cored CPUs in modern computers let the complex, threaded searches by chess engines operate efficiently in a very small amount of time. On the other hand, moving to an off-board solution required the addition of networking hardware to the otherwise self-contained unit, adding this new complexity of the board's main controller while it removed the hurdle of computing chess moves.

The decision was made in a group discussion early in the project. In order to add variety (and a certain "wow" factor) to our project, we would design the system to include connection to an off-board server for the purpose of handling game save states and computer-player chess moves. A Wi-Fi module would be added to the board to enable communication to and from a centrally located server via public or private Wi-Fi networks.

This decision also led to a bundle of new ideas for features since the server architecture allows for easy networking and centralization of information. As mentioned, the early idea of game saves was an easy target for redeployment to the server. In addition, the ease of networking brought up the possibility of multiplayer, which came to add another "wow" factor to the project. The possibility of tying into social networks like Facebook and Twitter was also discussed.

Once the decision was made, real research into the possibilities for a server-based architecture could be started. There were several questions to tackle, including what language to write the server software in, which protocols to use, and, of course, what chess engine to use.

3.1.10.1 SOFTWARE LANGUAGE

There were several options as far as programming language was concerned. Most current, high-level languages were capable of tackling the problem presented, but each would allow a different approach to the problem.

C#.NET - C# was the first choice of language research because of experience and familiarity with the language the group had. C# and Microsoft's .NET environment would have allowed for incredibly easy debugging through the use of their integrated Visual Studio.NET IDE. C# was an obvious research area as it was an immensely powerful language with libraries and extensions already available to handle many tasks that would be required of the software system. Primarily, C# was incredibly easy to get

working with the command line chess engines researched. In addition, C# was a personal favorite language of the group's resident Software Engineer, Joe.

As a downside, however, C# was not known for its ability to interact well with non-Microsoft standards. C#'s interactions with MySQL, from Joe's experience, were clunky at best and using C# would have locked the group into using Active Server Pages (ASP) for delivery of web content. In addition, while Visual Studio was not the only IDE capable of compiling and running C# code, it was the most well-known and supported, potentially locking the group's programmer into a single IDE choice without as much flexibility as provided by other languages' IDEs.

PHP - PHP was looked at because of its nature as a web-focused language. Writing the server in primarily PHP would have allowed for a native web experience instead of the second-hand job that would have existed with C# and .NET. PHP was also very open to platforms such as MySQL that allow the group to get production-level quality without paying for an enterprise license. The number of frameworks available for PHP also left the group with many options as far as design and implementation.

On the other hand, PHP was not as flexible on the command line as C# and was not quite as easy to interface with command line chess engines. As a web-based language, this was not surprising, but the benefits of utilizing the simple web-interfaces in exchange for a more difficult experience on the backend had to be weighed carefully.

Java - Java is usually an acceptable alternative for C# anywhere the latter would perform well, and this fact remained true for this project. All the benefits of C# (in this project's scope) would also apply to Java, but the since Java is slightly slower than C# on Windows machines, there was little reason to use it over C#.

Python - Python was another powerful language choice. It had several of the advantages of both PHP and C#.NET, but a severe downside that would likely prove untenable. Like PHP, Python was incredibly easy to implement on the web front and, like C#, was very powerful on the backend, easily tying into the command line input and output from chess engine programs. The downside was that Joe was not nearly familiar enough with Python to write the server efficiently without devoting a large amount of time to learning the language first.

Perl - Perl was looked at as an afterthought. As a scripting language, Perl would have been easy to implement both the web front-end and chess engine back end. However, Perl was also limited by the platform that it would be performing on. To maximize the capabilities of the language, it would have to run on a Unix-based system, something that was not available for free to the group. Since every other programming language option had the benefit of working fine (or only, as the case with C#) on the Windows machines that the group had access to, Perl was dropped as a language choice early.

C++ - C++ was looked at as well, as it was an enterprise-level language that had robust and deep support spanning years. C++ was adept at port-based communications and would have actually allowed for direct communication with the few chess engines that provide non-command line interfaces, as many algorithms are themselves written in C++. C++ was considered a solid backup choice for each other language as, with work, C++ was almost limitless in its scope. While other language options provided benefits in certain areas, C++ had no downside that could not be corrected with time spent.

3.1.10.2 CLIENT-SERVER PROTOCOL

There were two primary methods investigated for the communications between the board and the server: direct port communication and web-server based communication.

Direct port communication had the benefit of being easy to work with on the low-level controller in the chess board. It would also have been relatively easy to implement in C++, C#, Perl, and Python. The downside was that, without significant infrastructure time investment, port-to-port communication was generally synchronous, leaving the server able to communicate with only one client at a time and requiring it to close and re-open the connection incredibly often as a result.

Web server based communication would have been easy to set up in almost any programming language that was explored, and that was why it was the primary focus of research for the project. Port-to-port communications were the basis of web-based communication, of course, but good open-source options exist for every platform considered that handle the problems that exist with port-to-port, including multi-threading and queue systems to handle several simultaneous asynchronous connections and always available information.

3.1.10.3 SERVER-ENGINE PROTOCOL

The research into which chess engine began with researching the two primary chess protocols, the Chess Engine Communication Protocol (CECP) and the Universal Chess Interface (UCI), along with the possibility of implementing a chess engine through original code and therefore using a proprietary, original protocol. The idea of implementing an original chess engine was quickly discarded though as the depth of modern chess engines was found to be such that years could be spent on an algorithm, refining it's AI and search patterns, only to have minor improvements in overall play.

The CECP and UCI both utilize the concept of piping input and output from the engine to whichever device or program is using them. Interfacing with these protocols programmatically therefore requires the ability to spawn a new process on the host machine and redirect it's standard input and output to the server process. As this involves direct access to the host machine's operating system, C#, C++, Python, and Perl excel at it while PHP suffers from its singular problem.

The primary difference between the two protocols is that the UCI offloads some duties from the chess engine itself and onto the program that is implementing the interface. As such, chess engines designed to function with the UCI have the opportunity to lack certain features that would be useful to have, such as support for opening books and endgame Table Bases.

3.1.10.4 CHESS ENGINES

There are countless chess engines available for use in DeepRGB, far more than could be researched fully in a reasonable amount of time. For this reason, only cursory research was done on only a handful of engines based on the overall skill level of the engines. This research is detailed in

Table 24 - Chess Engines, shown below.

Table 24 - Chess Engines

Engine[1]	Maximum Elo Rating	Interface	Multiple Skill Levels?	Endgame Table Bases?	Opening Books?
Houdini v2.0c	3209	UCI	Yes	Yes	Yes
Houdini v1.5a	3203	UCI	No	Yes	No
Stockfish v2.2.2	3163	UCI	Yes	No	Yes
Rybka 4.1	3162	UCI	Yes	No	No
Critter v1.2	3159	UCI	No	No	Yes
Vitruvius v1.11c	3125	UCI	No	Yes	Yes
Komodo v3	3115	UCI	No	No	Yes
Deep Junior v13	3040	UCI	Yes	Yes	Yes
Spike v1.4	3033	Both	No	Yes	Yes

To meet requirements CHS04, CHS05, and CHS06, the system must incorporate at all three functionalities listed in

Table 24, columns 4, 5, and 6. One engine alone is not required to meet all requirements, only the system as a whole, so it is possible to incorporate the use of multiple engines so as to cover all requirements.

[1] All information was obtained by checking the engine's command line output. UCI interfaces are required to share their capabilities when started.

4 DESIGN

4.1 HARDWARE DESIGN

4.1.1 MAIN CONTROL UNIT

The main control unit will need to be able to handle all of the input and outputs needed to make DeepRGB functional. It was decided in section 3.2.1.3 that the ideal microcontroller choice would be an Atmel Corporation ATmega 2560. Below is a brief explanation of main components within the microcontroller unit and their respective functions

4.1.1.1 VOLTAGE REGULATOR

The voltage coming from the main power regulator may not be as stable as we want it to be. To ensure the microcontroller unit doesn't get damaged in the case of an over-voltaging problem, a second voltage regulator will be put in place. Figure 15 shows the circuit for the power supply of the microcontroller unit.

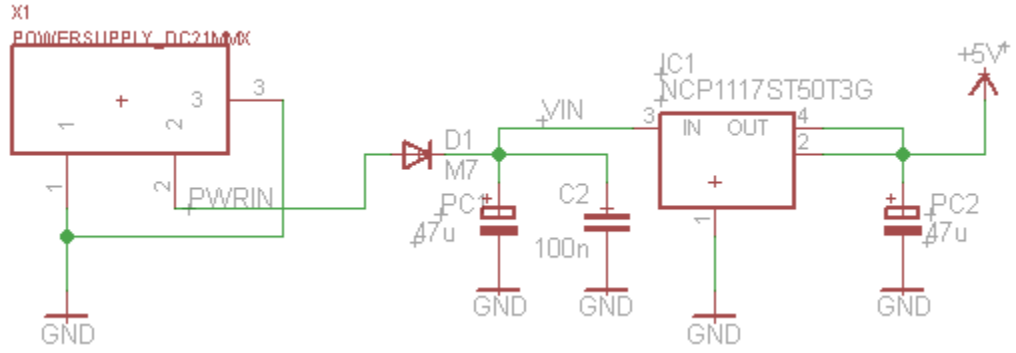


Figure 15 – The voltage regulator needed for the microcontroller reprinted with permission from Arduino.

The component labeled IC1 is an adjustable output low dropout voltage regulator. This component will ensure that the microcontroller unit will only receive the necessary 5 volts as stated in section 3.2.1.2.1.

4.1.1.2 THE ATMEL CORPORATION ATMEGA 2560

The Atmel Corporation ATmega 2560 will be the component that receives and transmits data from the other devices nested within DeepRGB and will process this using the programming code in its built in flash memory.

Figure 16 shows the Atmel Corporation ATmega 2560 with each of its pins referenced. The most important pins used for this project will the SPI pins used for the serial bus. SPI is one of the fastest ways the Atmel Corporation ATmega 2560 can communicate with other devices.

Since a total of three devices share this serial bus, a slave select pin must be used for each one in order for that device to be activated when needed. The LCD display will use four digital I/O pins to receive its data instead of using the serial bus. The wireless data transmission device will need to make use of the UART bus, this bus allows the wireless device to receive data from the Atmel Corporation ATmega 2560 and transmit it to the server via a web interface. The movement motors will need eight digital pins and one PWM pin. These pins can be arbitrarily chosen since they only send out a high or low signal.

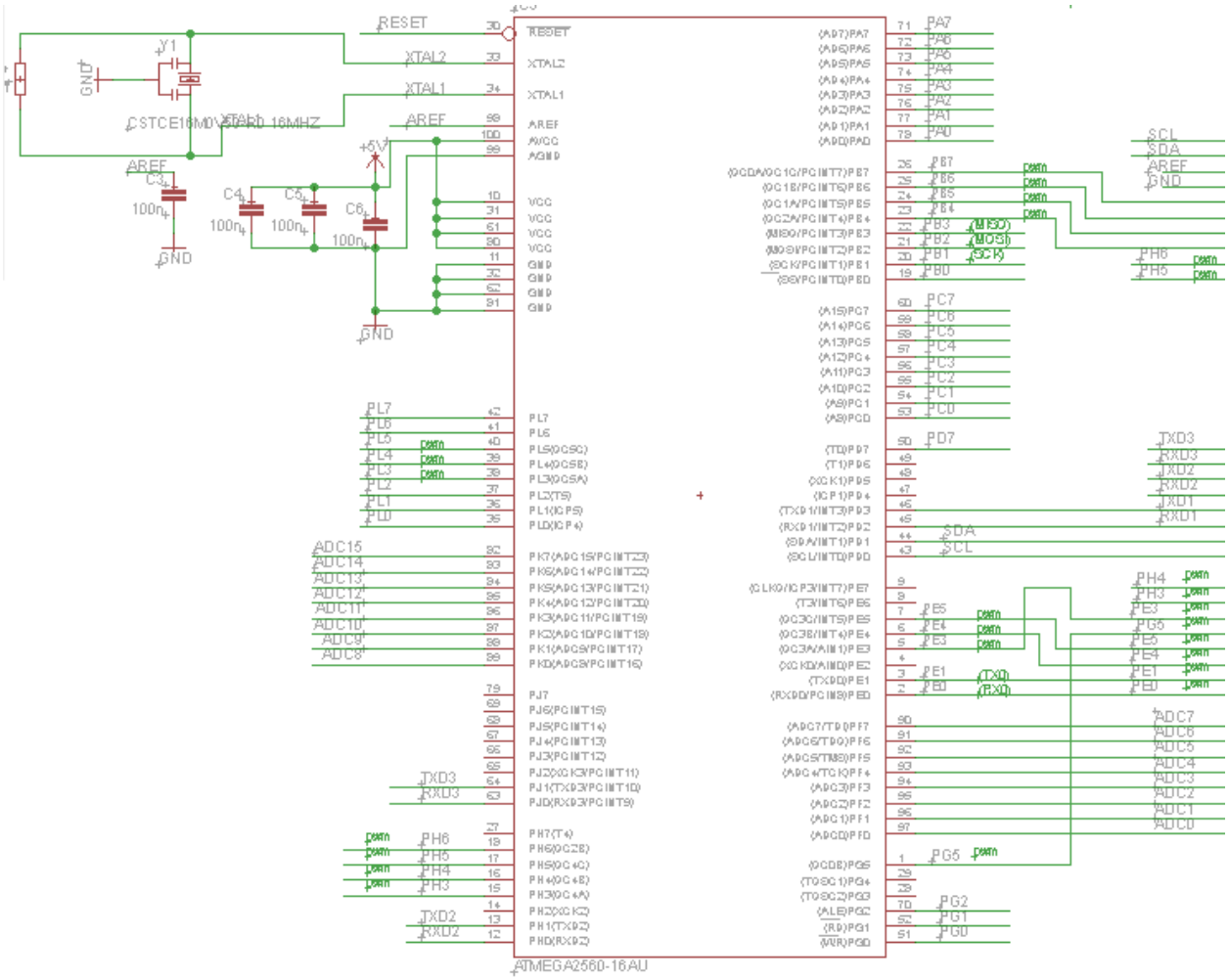


Figure 16 – The ATmega 2560 and its pin labels used for mapping reprinted with permission from Arduino.

Table 25 below shows a summary of which pins will be assigned to transmit and receive data from every component within the subsystem.

Table 25 – Labeling the pins used for each module

DeepRGB Device	Control pins needed
Liquid Crystal display (LCD)	Register Select = PD7 (T0) Enable = PG0 (WR) Data = PG1 (RD), PC0 (A8), PC1 (A9), PC2 (A10)
Red, Green, Blue Light emitting diode matrix	Serial Clock = PB1 (SCK/PCINT1) Chip select 1 = PC3 (A11) MOSI = PB2 (MOSI/PCINT2)
Hall effect sensor matrix	Serial Clock = PB1 (SCK/PCINT1) Chip select 2 = PC4 (A12) MOSI = PB2 (MOSI/PCINT2) 1 Analog input
Wireless data transmission device	RX = PE0 (RXD0/PCINT8) TX = PE1 (TXD0)
XY grid stepper motors	Directional controls = PC5 (A13), PC6 (A14), PC7 (A15), PA7 (AD7)
Electromagnet	Enabler = PA6 (AD6) PWM = PE3 (OC3A/AIN1)
Audio-Sound Module	Serial Clock = PB1 (SCK/PCINT1) Chip select 3 = PA2 (AD2) MOSI = PB2 (MOSI/PCINT2) MISO = PB3 (MISO/PCINT3)

These pins will be connected via pathways on the final manufactured PCB. The stepper motor control on the other hand will require some headers and long wires to be in place in order to ensure that its range of movement is not hindered in any way.

4.1.1.3 USB TO SERIAL INTERFACE

The Atmel Corporation ATmega 2560 will need to be able to interface with a computer to receive its programming code. This device will be able to connect to USB and transmit data to the flash memory of the microcontroller. The Atmel Corporation ATmega16U2-MU as shown in Figure 17 will need to have a DFU bootloader programmed onto it. The bootloader software is capable of ignoring malformed data and can also reset the Atmel Corporation ATmega 2560 if needed.

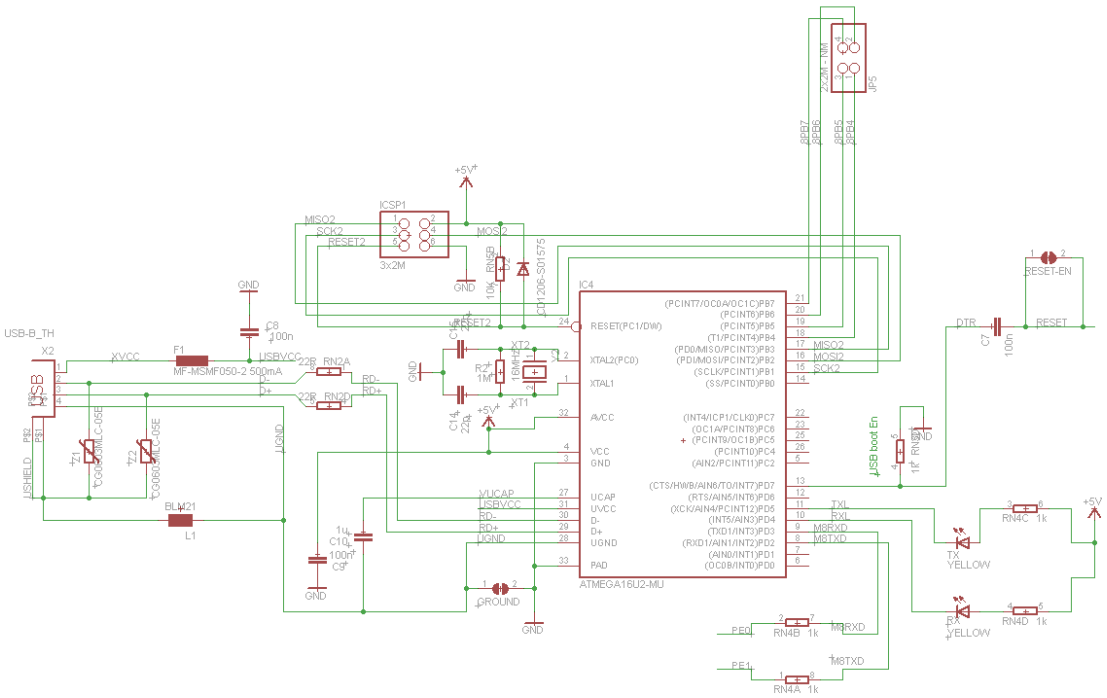


Figure 17 – The USB to serial interface used to upload the programming code reprinted with permission from Arduino.

4.1.2 MAGNETIC PIECE-MOVING SYSTEM

Design of the magnetic piece-moving system is similar to the design of an XY positioned table. The system should have the ability to move an electromagnet to any required location under the chess board and it should be able to be activated and deactivated during specified a time frame so figures can be moved in the order determined by MCU.

For the research part, we decided check the combination of two types of positioning systems together. We combined a rack gear system with a worm gear system which allows us to create an inexpensive, yet high accuracy, positioning system. We used worm gears travelling along aluminum rods in the Y-axis, then we used a gear track for Y-axis movement from the left side of the base board to the center along the Y-axis. From the rack gear system, we used gear tracks to move stepper motors along the path determined by the MCU. Movement along the Y-axis is made by a stepper motor which is located under the X-axis plane. The X-axis plane is based on two liner bearings and it can smoothly slide along the Y-axis aluminum rods. We also set up one aluminum rod on the top of the X-axis plane , where it goes parallel to the plane, together with gear racks. The stepper motor moving along the X-axis is located on the top of the X plane and is connected to the aluminum rod by one linear bearing. The electromagnet is the highest

point of our moving-piece system and it is attached to the X-axis stepper motor. We put basic illustration of the magnetic moving-piece system in the Figure 18. As an option we were considering to use modified sliders instead of aluminum rods. This will reduce price of the positioning system. Because in this case we do not need to buy linear bearing which is quite expensive and cost of the aluminum rod is also higher compared to the sliders cost. At this time we do not know if we can modify sliders but we can only find out when we are going to start experimenting with the construction of our moving system.

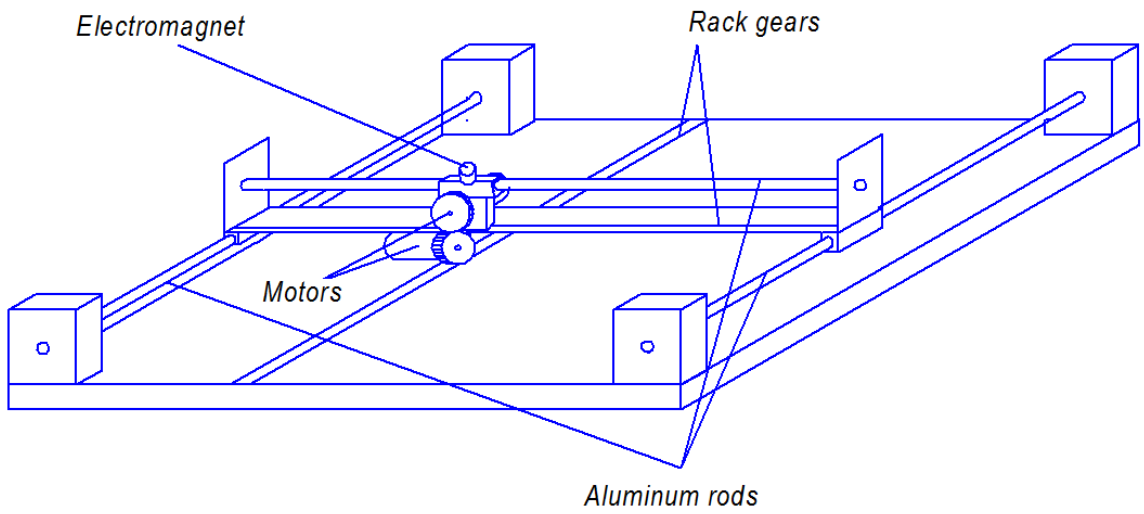


Figure 18 - Basic design of the moving-piece system

To control the electromagnet we created a simple circuit where we have one switch which turns the electromagnet on or off through a command sent by MCU. For the switch we used an NPN-transistor which opens if the pin out from MCU stays at the high level and closes when the pin goes low. We also needed to have a diode to protect the transistor from back voltage when we turn the electromagnet off. Finally, we added a couple of capacitors to reduce signal noise from electromagnet. As an option, we connected a potentiometer to analog an I/O pin of the MCU. It gave us the ability to control the level of power through the electromagnet so we can make perfect adjustment to the electromagnet's pulling force. In Figure 19, we see the basic connection from the electromagnet to the MCU.

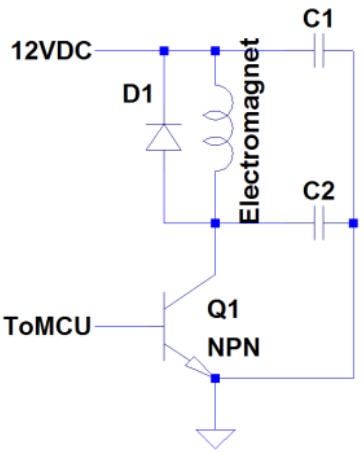


Figure 19 - Connection of the electromagnet to the MCU

The stepper motors we chose were the Mercury ROB-09238 Bipolar Stepper Motor; they have a light weight at only 200g, but it is still a powerful motor with 31.9 oz/in torque. The motor has also the longest wires of the group, which allow us to comfortably connect the motor to the motor driver without worrying about building a bridge. The stepper motor has a standard diameter shaft size of 5mm which allows us to connect a cylindrically shaped gear. The cylindrically shaped gears and rack gears are sold at vexrobotics.com.

4.1.3 PIECE DETECTION SYSTEM

As discussed in the research section of this report Deep RGB will employ a piece detection system powered by Hall Effect sensors. These sensors will detect the magnetic fields generated by the magnets placed inside the pieces and send the information to the MCU which will use it to extrapolate the positions of the pieces. This feature allows the board to reset the board after a game as well as re-center pieces moved by the user and store piece positions including the graveyard. The program which is described in the software section of this report will store the initial piece positions as well as update the positions when the board moves a piece. The Hall Effect sensors will help the board determine if a piece has been picked up by sensing a drop in the magnetic field. The board will then show available moves of that piece using the LED subsystem. Once the piece is moved the board will determine where using the sensor grid and then cross reference that information with the position index to make sure it was a legal move as well as to determine if a piece was taken or not. If the move is invalid either by the player placing the piece on a line or in an invalid tile the board will move the piece back to its original position before the move was made.

In the research section two arrangements of sensors were considered the triangle arrangement and the all corners arrangement. Both of these rely on ratiometric Hall

Effect sensors, the difference is the positioning and therefore the potential distance from a magnetic field as well as the possibility of sensing more than one field at a time. Before either method can be applied there must be a decision on what sensor to use. It has already been stated that the sensors should be ratiometric but they must also be able to endure the magnetic forces created by the movement device and the pieces. In the triangular arrangement the sensors will be approximately half an inch away from the center, the piece should never be farther from a sensor since the tiles will be only an inch and three quarters square. Therefore if the triangular arrangement is used the sensors should be able to at the very least detect a piece from a half inch away. However if the all corners method is used then the sensors will be one and one eighths inches away from the center. Once again this is the farthest a piece can be from a sensor so if the corners method is used the sensors should be able to detect a piece from an inch and an eighth away.

Another factor that must be considered while deciding on the appropriate sensor is the cost of the component. While the cost is not the most important factor it is certainly worth careful consideration, due to the sheer amount of components needed and the potential for cost to grow to high very quickly. It should be noted that when comparing the sensors below the prices will be based on the special pricing for buying components in bulk. Deep RGB requires at least 117 sensors, if the corners arrangement is used, and therefore the price shown will be the lowest price for which 117 units may be purchased. This will help in the decision by showing the absolute worst case scenario price wise for each sensor being compared. The previously discussed requirements as well as some not mentioned are listed below and will be used as the criteria for selecting the correct sensor.

- Cost: Less than $4.00 per unit
- Type: Linear Ratiometric
- Package: Through hole
- Sensitivity: Less than 2mV/G
- Operating range: 5000G preferable, above 3500G acceptable
- VCC(recommended): 5-6V
- VCC(max): Less than 10V
- I(max): Less than 30mA

Using these parameters to find the appropriate Hall Effect sensor can be more difficult than apparent at first glance. The problem stems from the operating range which in turn relies on the sensitivity. Most Hall Effect sensors available are built for precision therefore having a higher sensitivity allowing them to detect smaller fields. While many a programmable allowing users to set the sensitivity lower many still had a relatively high sensitivity i.e. 3mV/G and even more in some cases. This in turn limits the operating range, and if they were used in Deep RGB they would max out and possibly become damaged before the pieces were positioned. However there are some

promising components available and four stand out for their potential to work in Deep RGB.

4.1.3.1 A1302UA

The first of the four candidates is the A1302 which is a member of the A1300 family that is produced by Allegro Microsystems Incorporated. This sensor is well known and used frequently by hobbyists for a wide variety of applications. The 1302 comes in the same package as the 1301 which is available in both surface mount and through hole packages. However it has less sensitivity making it the preferred sensor of the 1300 family for potential use in Deep RGB. The specifications for the A1302UA are listed below for ease of reference.

- Cost: $1.54 per unit
- Type: Linear Ratiometric
- Package: Through hole
- Sensitivity: 1.3mV/G(typical), 1.0mV/G(min)
- Operating range: 0-2307G(typical), 0-3000G(max)
- VCC(recommended):6V
- VCC(max): 8V
- I(max): 11mA

According to the specifications above the 1302 could be used in Deep RGB, however this is not advisable. As the data shows the operating range only reaches the requirements when at its absolute maximum under the recommended settings. This is with VCC set to 6V and the sensitivity considered at its minimum value. This however would not work since the 1302 is not programmable and therefore there is no way to ensure that the sensor would be at its minimum, it would be luck of the draw. The potential hazard of using the 1302 is the chance of a sensor turning on when a piece is either too close or even on top of it. This would max out the output of the sensor making it almost useless and potentially damaging the component.

4.1.3.2 MLX90215

The second sensor to be considered the MX90215 is a programmable Hall Effect sensor manufactured by Melexis Integrated Systems. The 90215 comes in a through hole package much like the other sensors except that it has a fourth pin used for programming the sensitivity and other parameters. Using this component would force a change in the design however it is sturdier than other sensors and the ability to set the sensitivity would ease the construction process.

- Cost: $2.38 per unit
- Type: Linear Ratiometric
- Package: Through hole

- Sensitivity: 0.5mV/G (guaranteed minimum)
- Operating range: 0-4000G
- VCC(recommended):5V
- VCC(max): 5.5V
- I(max): 6.5mA

The 90215 certainly lies within the parameters required to create the sensor net. It can handle the magnetic forces created by a piece placed directly on top of it. However the fact that there is an extra pin required for the component to function as well as the price bars the 90215 from being a number one choice. That said if the budget for the project were larger and time was taken to redesign the schematic and the program itself the 90215 would work well with Deep RGB.

4.1.3.3 A1384

The A1384 is a member of the 138X family of Hall effect sensors and like the A1302UA is manufactured by Allegro Microsystems Incorporated. The 1384 like the 90215 is a programmable sensor though unlike the 90215 it is programmed before reaching the consumer. This keeps the package down to three pins rather than four but does not allow for the ability to program the sensor on the go. The 1384 is an economical substitute for a programmable sensor costing less than the 90215, though the reduction in cost does correlate to a relative drop in performance when compared to the 90215.

- Cost: $3.16 per unit
- Type: Linear Ratiometric
- Package: Through hole
- Sensitivity: 2.0mV/G (guaranteed minimum)
- Operating range: 0-1750
- VCC(recommended):5V
- VCC(max): 5.5V
- I(max): 8mA

After reviewing the specifications in detail it is obvious that the A1384 is not sufficient for use in Deep RGB. The operating range is too low due to the sensitivity being as high as it is and the maximum input voltage cannot be set high enough to increase the operating range enough for use. This sensor like the A1302UA would simply reach its maximum output far too early.

4.1.3.4 A1360

The A1360 is part of the 136X family and is the least sensitive of them all, like the A1302UA and the A1384 the 1360 is manufactured by Allegro Microsystems Incorporated. The 1360 is a programmable Hall effect sensor like the 90215. It is available in a four pin through hole package and can be programmed to be both

unipolar and bipolar. Unlike the 90215 the 1360 is far less expensive and can pose an effective substitute for the 90215.

- Cost: $2.40 per unit
- Type: Linear Ratiometric
- Package: Through hole
- Sensitivity: 0.7mV/G (guaranteed minimum)
- Operating range: 0-3571G
- VCC(recommended):5V
- VCC(max): 8V
- I(max): 12mA

The A1360 does not perform as well as the 90215 in consideration to the project. However the 1360 is much less expensive costing nearly a dollar less per component. The 1360 like the 90215 can endure the magnetic field created by a piece sitting directly on top of it. Another feature like the 90215 is the package, in that the component has four pins. While this is still a problem it is one that must be overcome for the sake of the project. There simply is no linear ratiometric Hall effect sensors found that can withstand the forces of the magnets and comes in a three pin package. Therefore the next best option is the A1360 from Allegro and shall be the Hall effect sensor used in Deep RGB.

Since the Hall effect sensor to be used comes only in four pin packages the sensor grid design must be altered somewhat. Upon further study of the datasheet for the 1360 it is apparent that changes to the design will not be that drastic. The fourth pin controls the filter and allows the user to set the bandwidth and eliminate noise. This is accomplished by using a capacitor between the fourth pin and the ground. Since the parts have not been purchased it is impossible to tell whether or not the filter will need to be set or not as it is set to 50kHz by default. Until experiments are done there is no way to tell if the noise levels will be too high and if so what capacitor will be needed to fix the error.

The number of pins used to control the sensors will be the same as the components will be programmed before being installed. This is accomplished by sending a pulse through the Vout pin which then programs the component. The 1360 has two modes which allow the user to test the features of the component. The first is called the Hold command which sets two of the programmable parameters and holds them until the VCC is reset. This allows the user to evaluate how the two parameters affect each other and how they perform together. The other mode is called the try mode which allows the user to set one parameter which is stored temporarily until the VCC is reset. There are also two modes which are used to permanently program the part. One is called blow mode this mode is used to program each parameter by blowing internal fuses in the part, several parameters can be programmed sequentially by using blow mode more than once. Once the parameters are set the user can use lock mode which blows a device-level fuse and prevents the parameters from being changed any further.

In order to activate the Hall effect sensors several analog MUXs must be used. In the corresponding research section of this report it was deduced that depending on the configuration used the sensor network would require 3 8-input MUXs and one 16-input MUX for the triangle configuration or 2 16-input MUXs for the corners arrangement. Additionally the shift registers necessary to run either setup is different, the triangle arrangement requires a 16-bit shift register while the corners arrangement requires an 8-bit shift register. This in conjunction with the fact that the sensor being used is able to manage the stresses involved with the arrangement the all corners method should be applied to the board. This reduces the cost of components as well as simplifies project construction considerably.

In response to the decision above the listed components from this point onward will be specifically geared toward the corners method. This decision could only be made once the sensor was chosen. The triangle method can use the components that will be reviewed though the exact placement and number of parts needed varies a little.

The first part that must be chosen after the sensor is the analog MUX which is used to turn the sensors on and off. There are several types of analog multiplexers which include analog to digital as well as analog to analog. Analog to digital uses an ADC to sample the analog signal selected and convert it to digital so that it is easily read by controllers. Analog to analog like its name sounds merely acts as a switch and allows a signal to pass from the pin chosen to the output. Until now the MUX has always been assumed to be analog to analog this is true for the MCU portions of this report as well. This is because the signal that is to be read is not going to pass through the MUX, the signal that turns the sensors on will be what is multiplexed and therefore it must remain analog. This said analog to digital MUXs can work although a DAC would need to be used in series with the output so that the circuit could be connected.

The MUX must be able to take in 16 inputs and output one signal, preferably an analog signal. The package should be through hole like the other components so as to fit on the board being used. Cost is not a big factor when deciding which MUX to use as there will only be two needed. These specifications as well as others are listed below for easy comparison to the components being considered.

- Cost: Less than $5.00 per unit
- Multiplexed inputs: 16
- Output(s): one analog
- Package: Through hole
- Voltage input(max):5.5V
- Iin(max): 12mA

Another consideration should be made so that the input and output are as similar as possible though this may be best determined through experimentation

4.1.3.5 CD74HC4067

The CD74HC4067 is a 16 input single source analog to digital multiplexer manufactured by Texas Instruments. It comes in both surface mount and through hole packages at a low cost. The 4067 offers high speed switching using silicon gate CMOS technology.

- Cost: Less than $0.95 per unit
- Multiplexed inputs: 16
- Output(s): one digital
- Package: Through hole
- Voltage input(max):6V
- Iin(max): N/A(output is digital)

The 4067 is an affordable 16-input analog multiplexer that can operate in the ranges needed by Deep RGB. However this particular component is an analog to digital MUX and therefore causes problems when trying to use it as a digital switch like it would be in the sensor grid. It was previously stated that it is possible to convert the digital signal back to analog which is true but unless a more preferable choice presents itself this should not be done.

4.1.3.6 ADG406

The ADG406 a member of the ADG4xx family of analog multiplexers manufactured by the Analog Devices company. The 406 is a 16:1 double source analog to analog multiplexer and is available in a 28 pin DIP package. The package is equipped with 4 extra pins that are not connected when used and are marked NC in the datasheet. This is because the ADG406 shares the same package as the ADG426 which contains on chip address and control latches to make interfacing with an MCU easier.

- Cost: $0.51 per unit(available for sample purchasable in tubes of 13 for $6.60)
- Multiplexed inputs: 16
- Output(s): one analog
- Package: 28 Pin DIP
- Voltage input(max):27V(cutoff at 25V)
- Iin(max): 20mA

After reviewing the specifications above it is apparent that the 406 is a potential match for Deep RGB and meets all the requirements except cost. Fortunately there is a version called the ADG406BNZ available for sample. The BNZ is actually better than the regular 406 in one way and that is that it is compliant with the restrictions on hazardous wastes and contains a negligible amount of lead. This said the BNZ version of the 406 performs the same as the regular and should work just fine for Deep RGB’s sensor grid.

4.1.3.7 ADG426

The ADG426 is another member of the ADG4xx family and comes with a few features that the 406 does not have. While it is still a 16:1 double source analog to analog multiplexer the 426 comes with onboard address and control latches making it easier to control with an MCU. Otherwise the 426 has the same exact specifications as the 406 the only difference is the cost.

- Cost: $0.53 per unit(purchasable in tubes of 13 for $6.91)
- Multiplexed inputs: 16
- Output(s): one analog
- Package: 28 Pin DIP
- Voltage input(max):27V(cutoff at 25V)
- Iin(max): 20mA

The ADG426 is a much better choice over the 406 when using an MCU due to its ability to easily interface with a microcontroller. This is done by using the four pins that went unused in the 406 to set the address and latch it so that it cannot change until the WR signal is lowered again. Therefore the signal incoming from the MCU does not have to remain constant, therefore the microcontroller can move on to other tasks and save power. Since we will be using shift registers to input the data into the multiplexer and therefore the latched address feature is unnecessary for this application.

4.1.3.8 ADG506A

The ADG506A is a 16:1 dual source analog to analog multiplexer in the ADG5xx family manufactured by Analog Devices. The 506 is a CMOS monolithic analog multiplexer in comparison to the 406 which uses LC^2MOS technology. The main difference between the 506 and the 406 is the resistance created by the device while on the 506 has roughly ten times the impedance as the 406. This amounts to less than three hundred ohms of impedance to the 406's thirty ohms.

- Cost: $0.51 per unit(purchasable in tubes of 13 for $6.91)
- Multiplexed inputs: 16
- Output(s): one analog
- Package: 28 Pin DIP
- Voltage input(max):27V(cutoff at 25V)
- Iin(max): 20mA

While the 506 shares the same specifications, at least those of concern, as the 406 it does create more impedance than the latter. Since the 406 and the 506 cost the same this puts the 506 at a disadvantage and is thus a less sound component for use in this application.

After comparing the four multiplexers above it is quite clear which one should be used in Deep RGB. The analog MUX that will be used is the ADG406BNZ which as stated before is the lead free version of the original 406. This component stood out over the

others for its availability to be sampled from the manufacturer, potentially eliminating the cost of this component altogether. The 406 is a dual source component and the sensor net will be run on positive DC voltage the negative source of the MUX should be grounded reducing the range of the output to 0V-25V.

Since the multiplexers being used will require 4 digital inputs each to control them an 8-bit serial in/parallel out shift register should be used to link them to the MCU. This will not only reduce the total pins needed from the MCU to control the grid to 3 it will also provide the added feature of an address latch. Once the register is loaded and the store command is enabled the address will be saved in the register and constantly outputted until the register is changed. This frees up the MCU to perform other tasks without having to keep the output to the multiplexers at a steady level. Shift registers were used in the design of the LED subsystem and since there will be a quantity of 8-bit shift registers purchased for that application the same component may be used for this one as well.

4.1.4 AUDIO SYSTEM

As discussed in the corresponding research section of this project the chessboard will be able to play alert sounds when certain events occur during a game. In addition the board will play music based on the preferences set by the user which are loaded onto the board from the server when the user logs in. To do this it is necessary to use an audio shield to decode the mp3 files stored in memory and output them as signals to the speakers to make the appropriate sound. There are several of these peripherals available on the market several of which are very reputable and well known amongst hobbyists who use Arduinos.

A criterion must be established by which to select the proper shield to begin with the audio shield must be able to play mp3 files so that the sound quality is not too low like MIDI or WAV files. Secondly it should be able to output to either a speaker or a 3.5mm jack although both would be preferable. The ability to store data on the board itself either by removable drives or memory integrated into the board itself should be available. These criteria and others are listed below to facilitate in the comparison of the different components.

- Cost: Less than $50.00
- Decoding capabilities: Mp3 or better
- Outputs: 3.5mm jack or speaker output
- Data Storage: Removable drives or onboard memory
- Input: SPI interface
- Voltage input: 6V or less
- Current input: 100mA or less while running
- Interrupt capable: Yes

- Arduino-019 Audio shield

The Arduino-019 is an audio shield based on the popular VS1053b mp3 decoder chip and specifically designed by Seeed Studio to interface with Arduino microcontrollers. The 019 is a cost effective way to decode mp3 and other audio files while being driven by an Arduino. The 019 can output through a 3.5mm jack it is also capable of output to external devices through the I_2S serial connector and has an input 3.5mm jack for a microphone. Not only is it capable of decoding mp3s it is also capable of encoding Ogg Vorbis files from the microphone input. With this capability the 019 can take in a signal from an external device such as an mp3 player encode it as an Ogg file then replay it through the speakers.

- Cost: $27.50
- Decoding capabilities: Mp3, Ogg Vorbis
- Outputs: 3.5mm jack and line out
- Data Storage: micro SD card up to 2Gb
- Input: SPI interface
- Voltage input: 5V
- Current input: Unspecified
- Interrupt capable: Unspecified

The 019 boasts some impressive specifications although it has a few flaws the most concerning of which, is the lack of information provided by the datasheet available. Due to these missing details it is not possible to determine the current needed to operate the board nor is it apparent whether or not the music can be interrupted at a moments notice although the likelihood that it can is high. Therefore the 019 should be used only as a last resort if a better candidate cannot be found.

rMP3

The rMP3 is one of the newer Arduino peripherals created by Rogue Robotics. It is compatible with some of the more popular data structures such as FAT32 and is capable of reading a micro SD card of up to 32Gb. It is also capable of supporting higher bit rate mp3 files than most audio shields as it can decode files with bit rates up to 320kbps. It also has the capability to read and write audio files from an external source to memory rather than just read and decode them from memory. However this does not come cheap as the rMP3 costs almost three times the average audio shield.

- Cost: $64.99
- Decoding capabilities: Mp3
- Outputs: 3.5mm jack
- Data Storage: micro SD up to 32 Gb
- Input: SPI interface
- Voltage input: 5V
- Current input: 60mA

- Interrupt capable: Yes

The rMP3 is an impressive piece of hardware although it does have its drawbacks, mainly the price is higher than the budget allows. The unit does more than it needs to for this application and therefore the extra cost is unnecessary. If the project were ever to be turned into a consumer product the rMP3 could add a few features to the board making it more desirable. However as the project does not require such a sophisticated piece of equipment it must be dismissed as a viable part.

DEV-10628

The DEV-10628 is an audio shield designed around the VS1053b mp3 decoder and is geared towards use with Arduino microcontrollers. It is capable of decoding mp3s as well as Ogg Vorbis and other popular audio formats. It is run using SPI between it and the MCU the MCU loads the buffer on the shield with the information that it needs to run for a small fraction of time. While the shield is performing the tasks on the buffer the MCU is free to perform other tasks. This buffer is relatively short so the MCU could wait until the end of the buffer to load an alert sound and most users would not notice.

- Cost: Less than $39.95
- Decoding capabilities: Mp3, Ogg Vorbis
- Outputs: 3.5mm jack and L, R, GBUF hardwire
- Data Storage: micro SD
- Input: SPI interface
- Voltage input: 5V
- Current input: N/A
- Interrupt capable: Yes

The 10628 is an inexpensive audio file decoder for use with an Arduino MCU, it is based on the most popular decoding chip used for these applications today. Capable of decoding some of the most popular and best quality sound files used today the 10628 is also capable of outputting the sound to an amplifier and used to drive large speakers. Thus the 10628 is the ideal audio module for use in this project for its simplicity and lack of frivolous features.

After the audio shield has been purchased it will be connected to the MCU using 8-Pin headers to create a fast and reliable connection. When the player logs in the MCU will receive the data from the server and command the audio shield to play the music that the user prefers. Like the auxiliary input capability this feature may not make it into the final prototype however it should be included if at all possible. The MCU should also receive data from the server on the conditions of the match. If a notable event occurs then the MCU will receive an alert from the server where the chess algorithm is being run. The MCU will then write the data onto the audio shield's buffer so that it stops playing the music and sounds the appropriate alert which will be stored on the micro SD card. For the purposes of this project the speakers that will be used will be a pair of

powered computer speakers so that the signal does not have to go through an amplifier to be played.

Error! Reference source not found. is a flowchart that shows how the audio system will generally work it must be noted that the external device and user blocks are there for the possibility that they will be included in the final prototype. They however are not guaranteed to be included and therefore are outlined and connected by dashed lines. It shows that the server communicates with the MCU this is through Wifi which is discussed elsewhere in the project documentation. The server sends the MCU user data as well as information such as alerts which the MCU sends to the audio shield via headers where it is decoded and output through the 3.5mm jack into a switch and then into the speakers where it is output as sound. The switch is controlled by the MCU which allows the appropriate signal through based on the user input and current alerts sent to it by the server. As with the user and external device blocks the switch block is a dashed line because it may or may not be included in the final product. This is also why there has not been any research into what switch to use for this application.

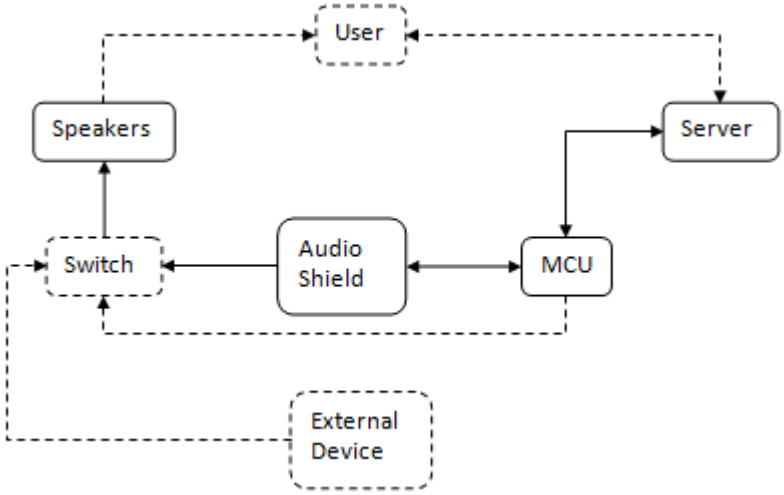


Figure 20 - A flow chart describing how the audio components will interact with each other

4.1.5 LED SYSTEM

To start off the LED system the first thing that must be done is to select an appropriate LED to use. It has already been decided that the LEDs used should be a pin mount RGB LED but there are many different components that fit this description. To narrow it down further the LED selected should fit the following criteria.

- Cost: $2.00< per unit
- Luminosity: between 500 and 5000 mcd (They can't be too bright or they will pollute other squares
- Pin Configuration: Common anode

- Height: Less than 45mm from tip to tip
- Diameter: 5mm
- Voltage(max): 5V or less
- Amperage(peak): 50mA or less
- Viewing angle: less than 80 degrees greater than 20 degrees

Once this criterion has been used it narrows the search down to a more reasonable level. After looking through several sites from which to purchase the LEDs it came down to four possibilities. The four are listed below along with their relevant specifications.

4.1.5.1 RL5-RGB-C-2

The first of the four LEDs being considered is the RL5-RGB-C-2 (hereon referred to as the C-2). It is available from the company Super Bright LEDS. It is a cost effective RGB LED that is available in both common cathode and common anode. The rest of the specifications are as follows.

- Cost: $0.79 per unit
- Luminosity: 1200-2200 mcb
- pin configuration: common anode/cathode
- height: 35.1mm
- Diameter: 5mm
- Voltage(max): 2.4V,3.5V,3.5V
- Amperage(peak): 50mA
- Viewing angle: 60 degrees

The C-2 is within all of the parameters but the luminosity is lower than the other LEDs presented. On the other hand its viewing angle is larger than the other components presented. This could help make it more viable since the wider the viewing angle the more lumens or photons the component projects therefore making it able to light up a larger area. Another benefit of the C-2 is that it is shorter than most of the other LEDs being considered making it possible to create a skinnier board.

4.1.5.2 RL5-RGB-C

The RL5-RGB-C (hereon referred to as RGB-C) is in the same family as the C-2 LED. It is available from the same company Super Bright LEDs and has similar characteristics to the C-2. The full list of relevant specifications is given below.

- Cost: $1.59 per unit
- Luminosity: 1000-5000 mcb
- pin configuration: common anode/cathode
- height: 35.1mm
- Diameter: 5mm

- Voltage(max): 2.6V,4.0V,4.0V
- Amperage(peak): 50mA
- Viewing angle: 15 degrees

The RGB-C is relatively expensive to the other LEDs costing over twice as much as the C-2. However it does have a greater luminosity range than the C-2. This extra luminosity is a benefit but is severely limited by the viewing angle which is rather small. Thus the RGB-C is more suited for focused lighting applications such as flashlights and lamps rather than for illuminating a chessboard. If it were used the RGB-C could have the potential for lighting up a small portion of the squares rather than the hole area.

4.1.5.3 YSL-R596CR3G4B5C-C10

This particular LED (referred to as the C10) is made by a Chinese company called China Young Sun LED Technology Corporation and is available on the Sparkfun website. It is available in packages of 100 which drive the cost per unit down. These are still more expensive to buy than the C-2 but since there are thirty-six extra there is room for error. The specifications of the C10 are as follows.

- Cost: $0.60 per unit (when bought in packages of 100)
- Luminosity: 800-4000 mcb
- pin configuration: common anode/cathode
- height: 38.6mm
- Diameter: 5mm
- Voltage(max): 2.2V,3.4V,3.4V
- Amperage(peak): 30mA
- Viewing angle: 40 degrees

The C10 is an excellent choice to use in Deep RGB. It is inexpensive and comes in packages large enough to account for any mistakes made while soldering the LEDs in place. It has the right pin configuration available and it short enough for the purpose it is fulfilling. The light output while less than the RGB-C is still backed by the larger viewing angle. Another perk to the C10 is the lower max amperage which helps lessen the strain on the power supply.

4.1.5.4 YSL-R596CR3G4B5W-F12

This LED (referred to as the F-12) is manufactured by the same company that makes the C10 and is very close to the C10 in its specifications. One of the differences it that the F12 is a diffused lens LED rather than a clear LED like the previous components. This helps the light scatter and become less focused than that of a clear LED which is ideal when trying to light up an area rather than a single point like on the chessboard. The specifications of the F-12 are as follows.

- Cost: $0.60 per unit (when bought in packages of 100)

- Luminosity: 1200-6500 mcb
- pin configuration: common anode
- height: 38.5mm
- Diameter: 5mm
- Voltage(max): 2.2V,3.4V,3.4V
- Amperage(peak): 30mA
- Viewing angle: 40 degrees

The F-12 while costing the same as the C10 has a brighter luminosity which combined with the same viewing angle will make it easier to light an area. The diffused lens is beneficial as well since the light will not be as focused and won't appear as a single point in the middle of each square. These attributes make it nearly perfect for this application however before a final decision could be made one more test had to be conducted.

The parameter 'luminosity' is given in mcb or millicandelas; this is an adequate way of measuring the brightness of a source but does not account for what happens to the light after it leaves the source. The unit mcb is used to mainly describe LEDs that are being used in focused applications such as lighting and flashlights. These units can be converted into lumens which measure the light intensity over an area; it is proportional to both mcb and the viewing angle. The higher the lumen count the brighter the lighted area will be. Below is a chart with the lumen count for each LED and should be helpful in determining which of the LEDs is best suited for the task of lighting the board.

As the table shows the two LEDs that give out the most light over the largest area are the C-2 and the F12. Out of these two the C-2 is less expensive if only sixty four are bought, that being said it was decided to go with the F12 due to the fact that it comes with spares. The F12 might put out less light per area but this is a small difference and in the end it will not matter.

Table 26 - A chart comparing the lumen values of the LEDs being considered for Deep RGB

LED	C-2	RGB-C	C10	F12
Red	1.18lm	0.07lm	0.31lm	1.07lm
Green	1.86lm	0.27lm	1.52lm	2.47lm
Blue	1.02lm	0.06lm	0.35lm	0.46lm
Total	4.06lm	0.40lm	2.18lm	4.00lm

After deciding which LEDs to use it had to be decided what shift registers to use in the project. The purpose of the shift registers as stated before will be used to reduce the amount of outputs needed from the MCU. The LEDs will be arranged in a grid and called by activating the anode and cathode of each specific LED color. To do this we link each row with a common anode which results in 8 bits of data. Then each color in each column is connected by a common bar thus creating another 8 bits of data for each

color. In total there are 32 bits of data which require 4 8bit shift registers to regulate them.

The schematic in Figure 21 shows how each of the LEDs would be wired so that only four 8 bit registers are needed to run the entire matrix. Now that the need for these components has been established more parameters must be defined so that a shift register can be chosen.

- Cost: $5.00 < per unit
- Size: less than 2 inches in length
- VCC: 5V or higher
- Iout: greater than 30mA
- Clock: greater than 16MHZ

The most commonly used family of shift registers for this application are the 595 series. The 595 is a serial in parallel out register which is perfect for the LED matrix because it reduces the data signal down to one bit. There are several of these 595 series registers and two of them came under consideration while trying to find the right parts for the project.

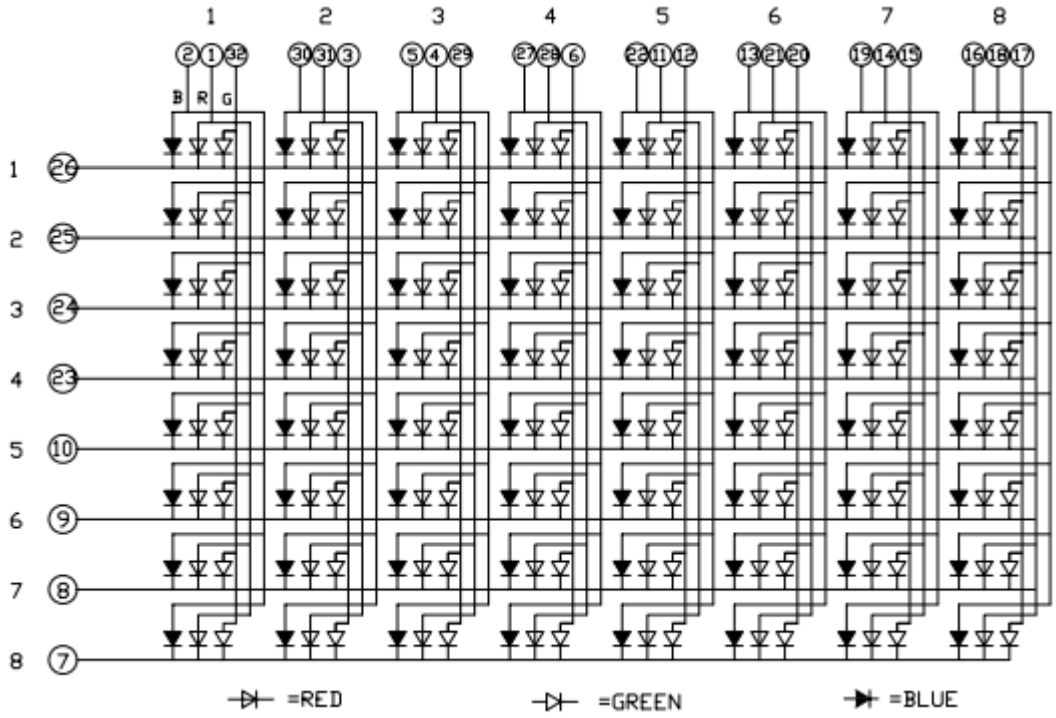


Figure 21 - An example schematic of an LED matrix awaiting reply to request for permission to reprint from the China Young Sun LED Technology CO. Picture taken from the YSM-2388CRGBC datasheet.

4.1.5.5 74HC595

The 74HC595 is a low current 8 bit shift register that is made by Texas Instruments. It is commonly used for latching data to run LED matrices and other large data applications to free pins on the microcontrollers being used. The specifications of the 74HC595 are as follows.

- Cost: $1.50
- Size: 0.785 inches in length
- VCC: Maximum of 6 volts
- Iout: 35mA
- Clock: 29MHz at VCC=6V

From the data it is easy to see why the 74HC595 is a commonly used component for LED matrices. It is relatively inexpensive costing well under half the allowed price per unit. It is compact measuring under an inch in length. The register meets the power needs of the LEDS being able to withstand up to 35mA of current passing through the component. Finally one of the most important features of this component is the clock cycle which is greater than that of the MCU being used to the registers should not slow the MCU down.

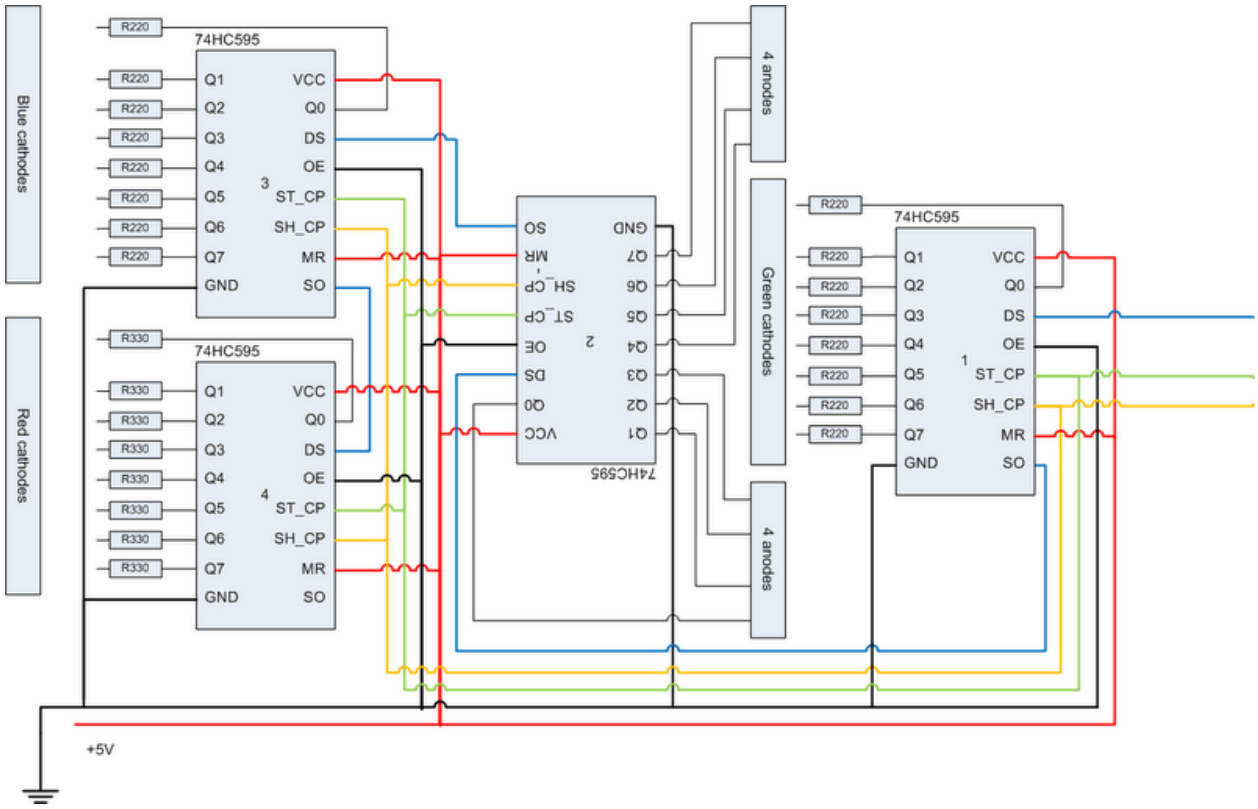


Figure 22 - A schematic of the four shift registers connected together awaiting response to request to reprint from Francis Shanahan

http://francisshanahan.com/index.php/2009/how-to-build-a-8x8x3-led-matrix-with-pwm-using-an-arduino/

The other candidate for the shift register is the TPIC6B595 which is a high power register. This register while fitting inside the parameters for the shift register it would simply be overkill and so the 74HC595 shift register should be used in the project. The four shift registers should be connected together such that the data stream flows through one register and out to the input of the next register. This way there is a continuous stream of data and each register can be loaded from the same output pin. The other two inputs absolutely necessary for the matrix to function are the shift clock and the load clock. Each register should output to one of the busses green, red, blue and anode. Once completed it will resemble the schematic provided in Figure 22.

Once the matrix is completed and connected to the MCU via the shift registers the MCU will simply load a bit onto the first register then the shift clock will shift it over one and the MCU will load the next bit. By this process the data is pipelined down the registers until all are loaded and then the load clock is activated which stores the data in the registers. This is a total of three outputs that the MCU must calculate and output the details on how the MCU does this are discussed in the MCU software section of this report.

4.1.6 LCD DISPLAY SYSTEM

The display has not been the main part of DeepRGB. The main purpose of the display in Deep RGB is to provide extra information about the game such as status of the game, as well as information required to log on to the system at the start. At the start of the game, the display of shows this information and assists the user in setting up an internet connection. It needs to be capable to display all this required information without clipping.

The display should be able work in internal areas. This requires we operate correctly inside a temperature range from 5 C to 40 C. The cost of display is the same as for other elements, and it should be able to complete its task as well as to be as small as possible. From our display research, we found that the alphanumerical display model, BLUELCD16x2BL, meets all our requirements. It has 2 lines with 16 symbols in each line, allowing us to display all necessary information. The resolution of each symbol is 5 by 8 pixels, which allows us to use a wide selection of preinstalled symbols and even create some of our own symbols if required. The full size of the module is 80mm by 36mm while the actual display size is 64.5mm by 16mm.

The display BLUELCD16x2BL requires 5V DC supply voltage and supply current in the range of 2mA to 4mA. However, it has supply voltage range for logic from 2.7V to 5.5V, which in our case can be used successfully with either the 3.3V supply line or the 5V supply line. Supply voltage ranges for the LCD from 3V to 13V, which allows us to use

this LCD with any supply line in our project. Operational temperature ranges from 0 C to 50 C which satisfies our temperature requirements.

Due to the fact that alphanumerical display BLUELCD16x2BL has a preinstalled display driver, the Hitachi HD 44780, our task of connecting the display to the MCU became far more simple. To connect the display, we need only know the pin configuration of the driver. The BLUELCD16x2BL has 16 pins, with numbering going from left corner around to the right. In the Figure 23, we illustrate the basic connection of the BLUELCD16x2BL display to MSU.

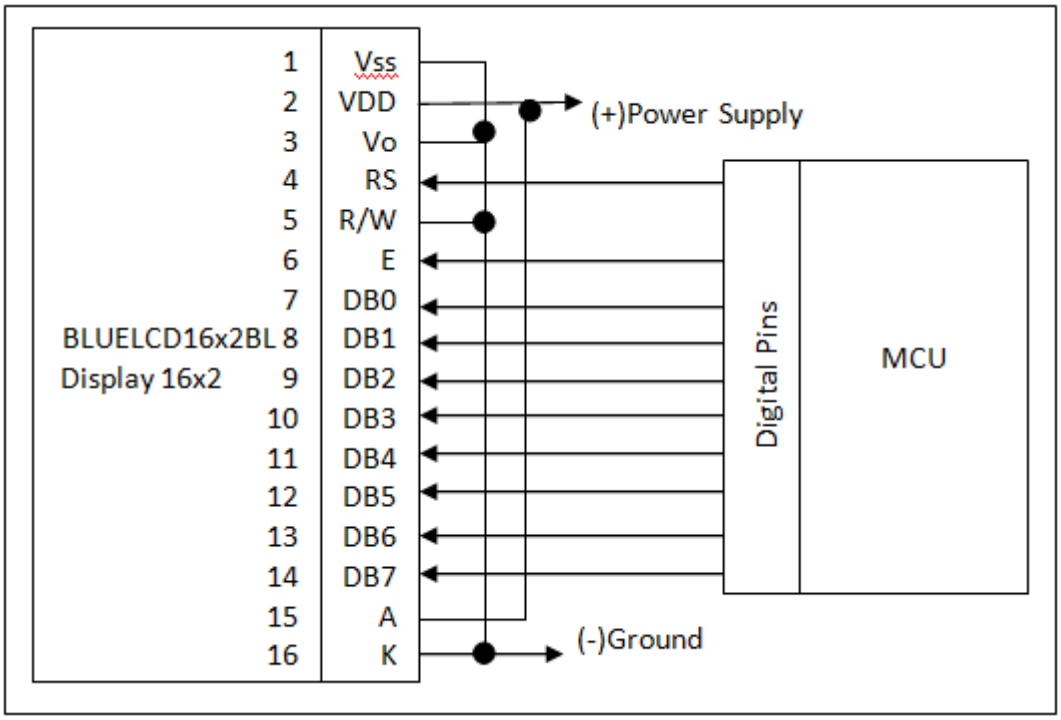


Figure 23 - Display connection diagram to the MCU

The V_o connection is used for contrast adjustment. R_s is used to register selection, and by connecting to ground, we can send it a low status. The R/W stands for read and write; we also connected this to ground. Display DB6 is used to enable or disable the signal. Pins DB7 through DB14, inclusive, are used to send data bits from the MCU to the display. For the regular text information we can use only 4 data pins, those from 11 to 14 inclusive. To activate a backlight we use pins 15 and 16, where A stands for Anode (which needs to be connected to V_{cc}) and K stands for Cathode (which needs to be connected to the ground).

4.1.7 POWER SUPPLY

The implementation of power supply system in Deep RGB project is very important because the appropriate function of all components of the Deep RGB system relies

squarely on the quality of the energy provided by the power supply. There are two important components of the power supply system: the source of the power supply, and the power distribution method within the system.

In the first component of the power supply system we included power outlet and AC/DC power adapter. Power outlet supplies power at 110 V and 60Hz to the adapter. The adapter is lower, rectified and supplies 12 V DC voltage output to the Deep RGB system. Selection of the place where adapter can be located inside of chess board or outside is an important decision because adapter has a transformer in it which can get very hot. Since we need to decrease the amount of heat inside the chess board, we decided on the external adapter placement. Laptop adapter ideally suited our requirements due to its external usage and high quality power output.

In the second component of the power supply system we included element which needed to be implemented on the PCB board. Those elements are 5V DC and 3.3V DC voltage regulators. It also included some capacitor to reduce noise and some resistor to supply appropriate current level to the components of Deep RGB chess board.

Another approach to the design of the second component of the power supply was to have an individual voltage regulator for each component of the chess board. An advantage of this approach is that we can protect every other component in case a hazard of sort arises in one single element. The disadvantage of such an approach is higher cost of implementing the power supply system. We would also need to have surge protection implemented in the power supply itself. Incorporation of overheat protection is not very important in this case because all major elements have a built in overheat protection. In Figure 24, we demonstrate the implementation of the first approach to the whole power supply system.

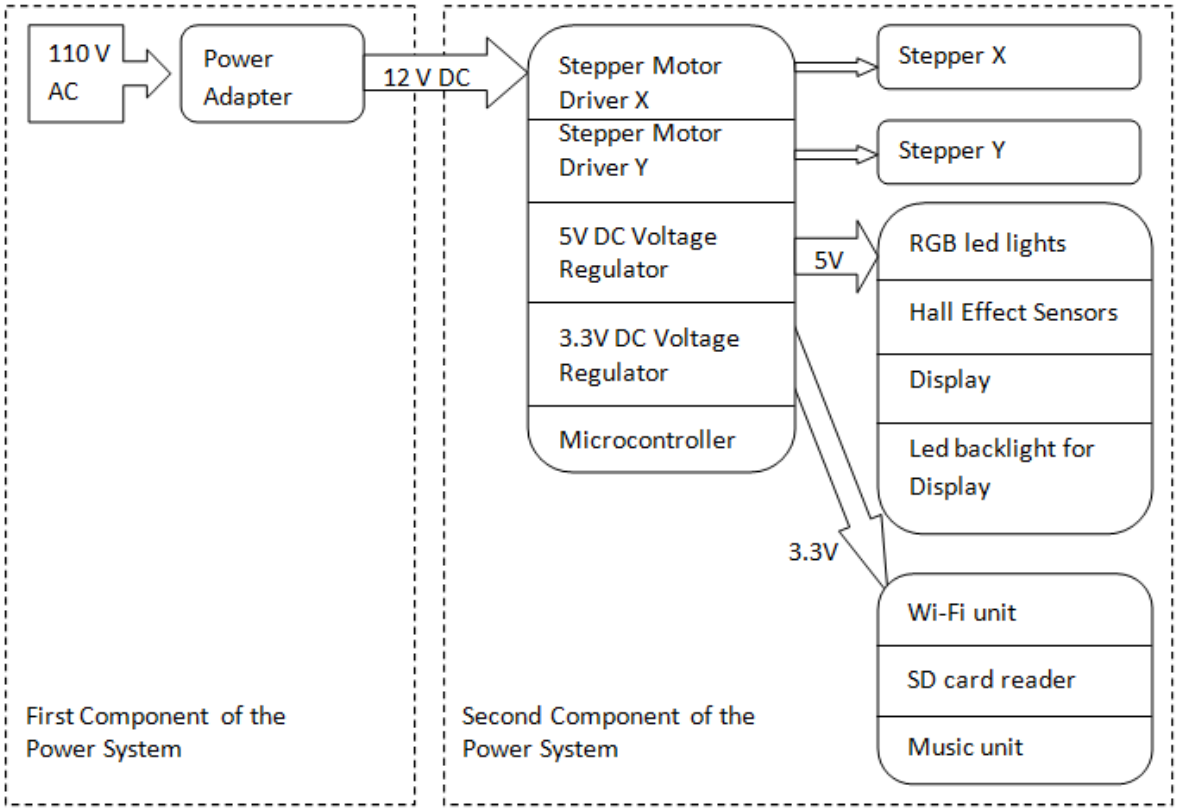


Figure 24 - Power Supply Block Diagram

4.1.8 WIRELESS CONNECTIVITY SYSTEM

The wireless data transmission device will be placed on a PCB alongside the other devices within the physical board. As chosen in section 3.2.6.4, the Wi-Fi module chosen for use is a Roving Networks RN-131G as shown in Figure 25. Since it only requires four wires to operate (PWR, GND, TX and RX), this module can be surface mounted onto the PCB and connected to the Atmel Corporation ATmega 2560 via pathways.

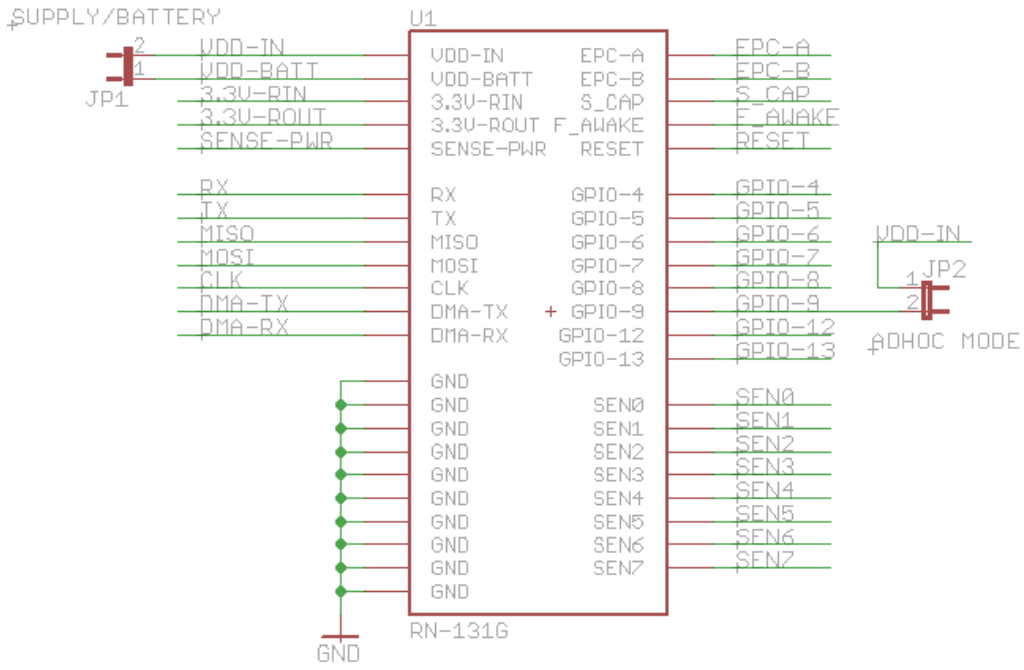


Figure 25 – The Roving Networks RN-131G and its pin labels for mapping reprinted with permission from Sparkfun.

Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)

The Roving Networks RN-131G is also equipped with ten general purpose digital I/O and eight analog sensor interfaces. Since the analog sensors will be connected to the Atmel Corporation ATmega 2560 directly, these inputs and outputs will remain unconnected. This module is capable of establishing a connection using UART or SPI, the UART option was chosen in order to free up the SPI bus during heavy data flows. This module also has the capability of having an antenna attached to allow for a longer connectivity range.

4.1.9 PCB DESIGN

Since strip and perfboard cannot be used for our project, the PCB layout for our circuit will need to be decided upon. Using Eagle Cadsoft, we can design our PCB and add the components onto the design to ensure that each device is obtaining a connection to their required pins. Using Eagle Cadsoft will allow us to move around modules before making a final decision. The placement of the modules is crucial, for example; the Wi-Fi module is best placed as close to the housing as possible in order to maintain high signal strength through the physical chess board casing. A prototype layout for our PCB can be seen in Figure 26

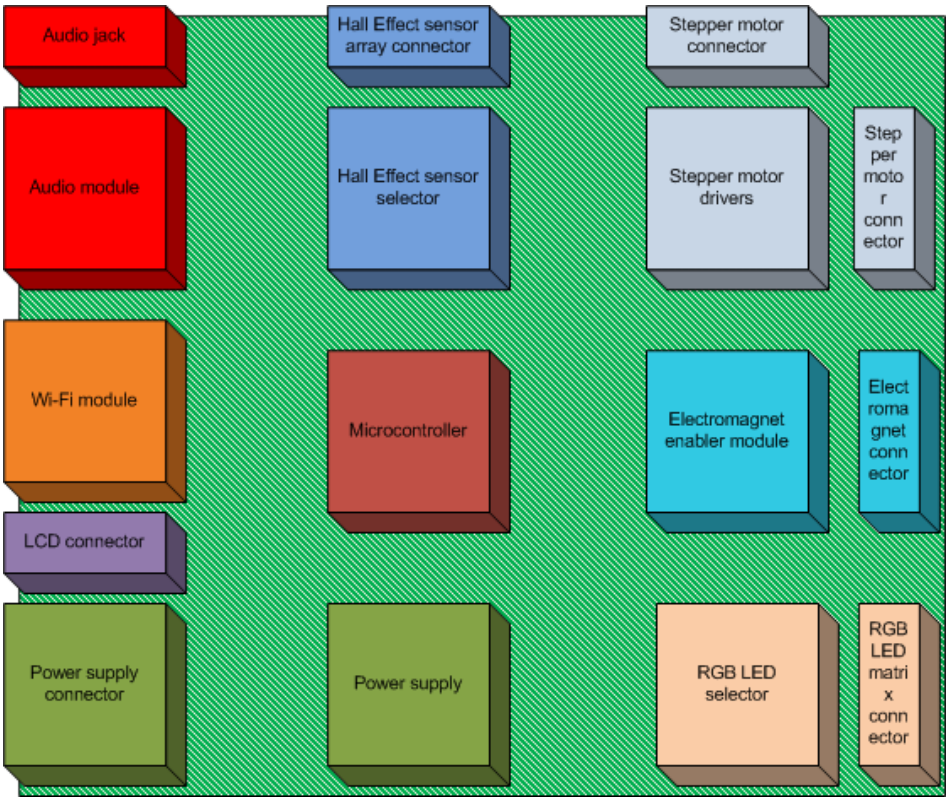


Figure 26 – The general layout of the PCB design.

The modules used for this project will need to be placed in a certain way that it maintains a small footprint and keeping crosstalk to a minimum. An easy way to reduce crosstalk and ensure that pathways don't cross over each other is to make a two layer PCB. Having multiple layers allows us to shrink the total size of the PCB but would increase the price per board, although the benefit of a multi-layered PCB greatly outweighs the increase in cost.

4.2 MAIN CONTROL UNIT SOFTWARE DESIGN

We will be using the Arduino IDE to code the Atmel Corporation ATmega 2560. Most of the devices already have open source libraries associated with them. This allows us to have every device up and running as quickly as possible. The libraries need to be installed in the same folder as the Arduino IDE in order to have them function as intended. These libraries are provided for public use and have a large community following that can provide feedback if the need for troubleshooting arises.

4.2.1 HALL EFFECT SENSOR ARRAY CODE

The hall affect sensor array code will need to send data to the shift registers in order to scan specific squares through the SPI bus and the analog input assigned to the array.

The array will be made active when the slave select pin goes low. There will be for loops designed to quickly scan the values coming from the Hall Effect sensors. These values need to be converted in a quantifiable number using the A/D converters on the microcontroller. The values in the array will be compared to see if any changes have occurred, if the piece isn't centered on the square or if an illegal move is made, it will then need to correct the position of the piece to the center of the square and report back the new position via the Wi-Fi module.

4.2.2 RGB LED ARRAY CODE

Using the SFRGBLEDMatrix library, the RGB LED array will be able to display several different of colors through PWM. The array uses the SPI bus to select specific squares that need to be lit and write the color values to those squares. The array will be made active when the slave select pin goes low. When a piece is picked up by the user, the code will need to light up the blocks where a possible move can be made. Since chess has a strict set of rules on movement, this can easily be implemented within the microcontroller's code. A player can also assign a color to their account and this color must be displayed whenever the user is logged in and in a game.

4.2.3 AUDIO MODULE CODE

The audio module utilizes the SPI bus in order to select and play an MP3 file located on a built in SD card. The SFEMP3Shield library is capable of selecting a song from the SD card and loading it into the audio module's buffer. The buffer is capable of holding up to 100ms worth of audio data while the microcontroller performs other tasks. The code will also be able to control the volume and save that value for each account.

4.2.4 MAGNETIC PIECE POSITIONING CODE

The code needed to control the 2 stepper motors uses a very simple stepper library. The library known as AccelStepper only needs two inputs in order to move a stepper motor. The two variables are speed and number of rotation degrees. The chess board will resemble a Cartesian coordinate system and the microcontroller will have specific movements saved for each type of chess piece. If a piece is in the way, the code will need to move the obstructing piece and allow the new piece to freely move to its new location. Since an electromagnet is used, the code will need to enable the device when it needs a piece moved and disable it when it's not in use in order to save power.

4.2.5 WIRELESS DATA TRANSFER CODE

The Roving Networks RN-131G will be using the WiFly Serial open source library. In order for this library to work, some extra libraries must be installed in order to simplify

the coding even further. Libraries such as Pstring (a lightweight class for printing to buffers), Streaming (a method to simplify print statements), DateTime (a library for keeping track of the current date and time in software) and NewSoftSerial (an improved version of the SoftwareSerial library) must also be included in the same directory of libraries since the WiFly library will need access to these.

Only the Service Set IDentifier (SSID), encryption type (if any) and password (if any) are needed in order to secure a connection between the wireless network module and the network. Other options include port number, but that won't be accessible to the end user. The port number will default to 80 since HTTP is TCP/IP on port number 80.

The values needed to connect to an access point are entered in by the user via a makeshift keypad designed using the squares of the chess board as a keyboard. This information can be saved to prevent the user needing to enter in their settings every time they want to connect to their network.

4.2.6 LCD CODE

The code for the 16x2 LCD will use a library known as LiquidCrystal. This library greatly simplifies the code needed to display text on the LCD. Its main function is lcd.print(), where you only need to write a string of text in between the parenthesis and it will display this on the display. This will be used to prompt the user for account information. When not prompting the user, the LCD could display useful information such as time and amount of moves placed so far.

4.3 SERVER DESIGN

After considering all possible software options presented in Section 3.2.7, it was decided that a web-based system would be constructed to handle the server-side processing of the chess engines and remote-play options. A web-based approach was chosen for many reasons including facilitation of alternatives, portability, and maintainability.

By avoiding direct port-to-port communication, it is immensely easier to create alternative interfaces for the system, which was a design intention from the beginning. While Deep RGB will include only a website interface for remote play, it was intended to be able to accommodate the creation of iOS or Android applications or even native Windows, Mac, or Linux applications created by other users at other times in a graceful manner. The use of a web-interface combined with the portability of the Internet allows for almost any programming language to interface with the server and thus provide access to the system.

Portability is important to the project because it facilitates the creation of alternative interfaces as explained above by being available to a wide range of other systems. As a web-based system, Deep RGB's server will be available to any machine that can log onto the Internet, which in this day includes desktop and laptop computers, tablet devices, smart phones, even ordinary smart phones, and the list grows by the year. By interfacing via the Web, Deep RGB will be available to all of these platforms if a corresponding front-end system is available. For most system, the included website interface will be perfectly suited to on-the-go use of the system.

As for maintainability, a web-based system is the height of simplicity since all code for the system can be housed on a single server and the majority of it is available in non-obfuscated plain-text, not the unreadable byte code of Java or the machine code of pure compiled languages.

The Deep RGB server system will be designed in C#.NET and ASP.NET using Microsoft Visual Studio 2010. As the multitude of in-progress games will need to be stored in a compact manner, a Structured Query Language (SQL) database will be created using Microsoft SQL Server as it integrates with Visual Studio very easily and allows use of the very powerful LINQ (Language Integrated Query) to SQL implementation of SQL interaction. The chess website as well as the game progress update and request pages will be implemented with ASP.NET frontends and C# backends. There will also be a single management thread created in C# that will run continuously checking for games that need to be played by a computer opponent and will send those games off to the chess engine to be processed. A visualization of the integrated server system is available in Figure 27 - Visualization of Chess Server System.

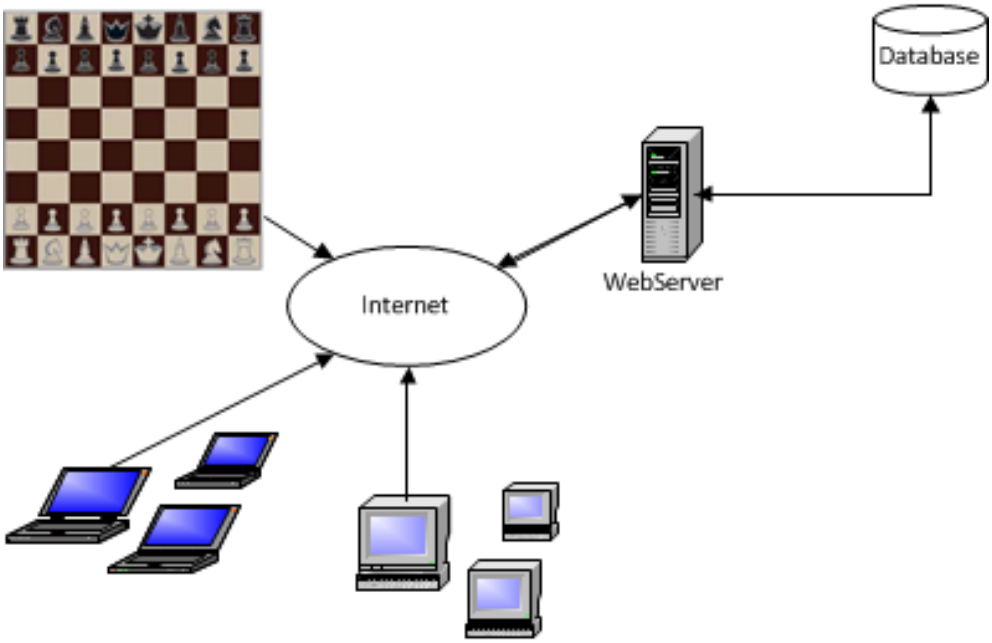


Figure 27 - Visualization of Chess Server System

4.3.1 DATABASE

The server system is driven by a Microsoft SQL Database. This database will consist of five tables, *users*, *games*, *moves*, *sessions*, and *reset*. The database will keep information gathered via the various management pages described in Section 4.3.3 and store it in an organized manner. The database will not be directly accessible from outside the server system. The only way to view information in the database or update the values that are stored there is through the management pages. Even though this is true, the database will be encrypted in memory.

4.3.1.1 TABLES

The ***users*** table will be created to store relevant information about each individual user in the system. It will have columns for user id number, username, the user's real name, LED color, and audio theme, as well as a hashed password field and fields for the date and time the user's account was created along with when the user last made a move.

The ***games*** table will store the relevant state information about each of the games the system is tracking, in-progress or completed. It's list of columns includes game id number, white player id number, black player id number, observer id number (if applicable), whether or not observers are allowed at all, the current state of the board, the turn number, which player's turn it is, the date and time when the game was created, and the date and time of the last move made in the game.

The ***moves*** table holds a single record for every move taken in every game. Whenever a move is made via the update move management page, a record is inserted into this table, tracking the id number of the move, the id number of the game that is being played, the id number of the user that made the move, a bit representing which side made the move (black or white), what the turn number was when that move was made, and the time and date that the move was made.

The ***sessions*** table is a temporary record table that is designed to store the authentication keys of users who have logged in via the log in management page. The only records in this table will be user ids and the hashed authentication key that goes with them. This table will be occasionally cleaned out if the user the authentication key is assigned to has not made a move in more than an hour.

The final table is the ***reset*** table, another temporary record table that holds the password reset keys generated for the password reset system. The records in this table contain a userid, a password reset key, and the date and time that key was created. This table is cleared out on the same schedule as the *sessions* table, even though the password recovery system itself will not allow use of the keys if they are more than an hour old.

The overall structure of the database is seen in Figure 28 - Database Structure below. The four main tables are depicted with their corresponding columns. Important to note are the key mappings (depicted by dotted lines) in the diagram. The mappings from table to table allow the system to connect data in one table (for example, a move record in the moves table) with related data in another table (such as the game the move was made in, stored in the games table).

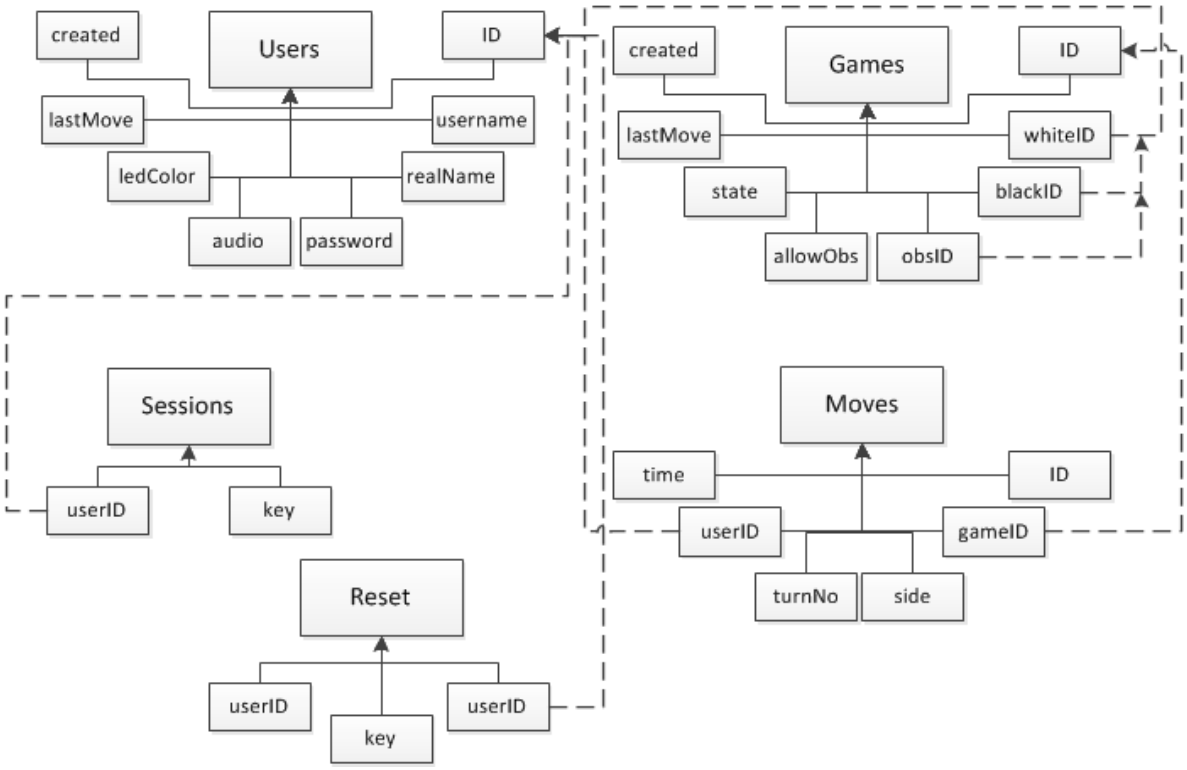


Figure 28 - Database Structure

4.3.1.2 SQL FUNCTION DESCRIPTIONS

The Structured Query Language (SQL) that Microsoft SQL Server implements describes functions that are to be used when constructing a database query. TABLE XXX below describes each function, along with its inputs and outputs.

Table 27 - SQL Function Descriptions

Function	Inputs	Outputs	Description
INSERT	Variable	None	Create a new record in a given table, with values as provided.
SELECT	Variable	Variable	Return zero or more rows from one or more tables in the database, optionally subject to filters

Function	Inputs	Outputs	Description
UPDATE	Variable	None	Change the contents of zero or more records in a given table to values as provided.
DELETE	Variable	None	Delete zero or more records from a given table, optionally subject to filters

4.3.2 CHESS ENGINE INTEGRATION

The chess engines described in Section 3.2.7.4 offer a variety of functions and skill levels, but since the features required by the server are set in stone there was only a single engine that would work. This engine is the Deep Junior engine, version 13. While Deep Junior is the only program that fulfills requirements CHS04, CHS05, and CHS06, another engine fulfilled objective CHS05 and was also the most advanced (in terms of Elo rating) engine available for free. Since Houdini v1.5 will utilize endgame table bases to optimize the endgame, it is possible to utilize that engine for maximum computer opponent difficulty after the opening book period.

The chess engine will be utilized by the C# management thread to handle processing all moves made by computer opponents. Since the UCI protocol does not require engines to implement opening books, we had to find an engine that supported that on its own in order to not have to implement them in the server software. Being as that chess strategy itself is not a primary goal of this project (and because we are not particularly skilled at it, and would therefore bring the overall skill level of the system down), we attempted to leave all algorithm programming to those who spend time optimizing it. Because of that, not implementing opening books or endgame table bases ourselves was important.

The chess engine, since it implements the UCI interface, must respond to standard UCI commands and reply with a standard format. That allows for the creation of a C# class that will be interchangeable between the two engines, Deep Junior v13 and Houdini v1.5, that we are using for the computer brains. This class will be called from a delegate method that will be spun off into a new thread by the management program whenever the chess engine is needed.

The chess engines will be delegated by a management process that runs whenever the server is active. This process is a very simple construction that manages scanning the database on a regular basis and spawning off a thread to deal with the chess engine when a game requires a computer move to be calculated. In addition to checking for computer moves to be made, the management process also does a simple aging on the *reset* and sessions tables in the database every hour to make sure the records in those tables are fresh. A simple flowchart for the process is available in Figure 29 -

Management Process Flowchart and the UML diagram for the classes presented is shown in Figure 30 - Management Process UML Diagram.

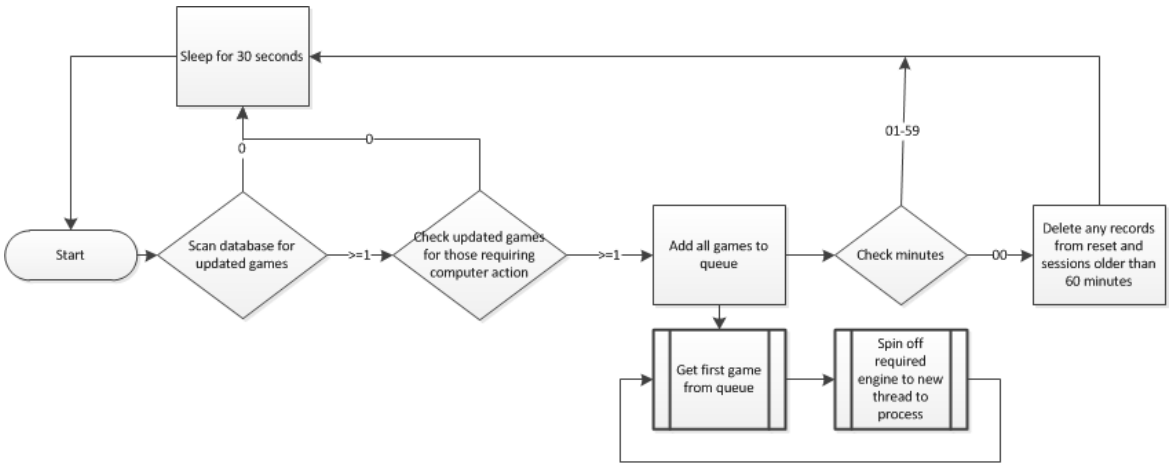


Figure 29 - Management Process Flowchart

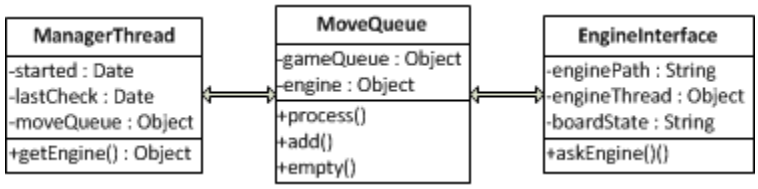


Figure 30 - Management Process UML Diagram

4.3.3 MANAGEMENT PAGES

Both the physical chessboard and the web interface described in Section 4.3.4 will utilize the same management pages to control the progress of the game and update the system's database with the current state of the game after a move has been made. There will be a separate page for each of logging a player in, getting the logged in user's profile information, requesting the list of games available, requesting the current state of a specific game, creating a new game and updating the database after a move has been made. In addition, there will be four administration pages that will handle creating new accounts, generating reset keys for users who have forgotten their passwords, resetting the passwords for those users, as well as adding a logged in user as an observer to an in-progress game.

Each page is kept separate for two reasons. Firstly, it keeps the backend of each page constrained in what it should be dealing with, limiting the number of unintended side effects of given algorithms. This is primarily a design benefit, as it allows the programmer to keep a more defined idea of the purpose of a single process when each process is divided into separate files. Secondly, separate pages for each function limit the amount of information any single page returns which makes it easier on the physical board's RAM-limited microprocessor to process entire requests at once.

None of the management pages are made to be human-readable; each is designed instead for machine readers and, therefore, sends and receives information in the JSON format. JSON, which stands for JavaScript Object Notation, is a lightweight, platform independent, text-based format. It is human-readable to an extent, which is a bonus for development, but it's format is very formulaic and is therefore easy for programming languages to interpret. The format is described in Figure 31 - JSON Object Definitions and Figure 32 - JSON Type Definitions.

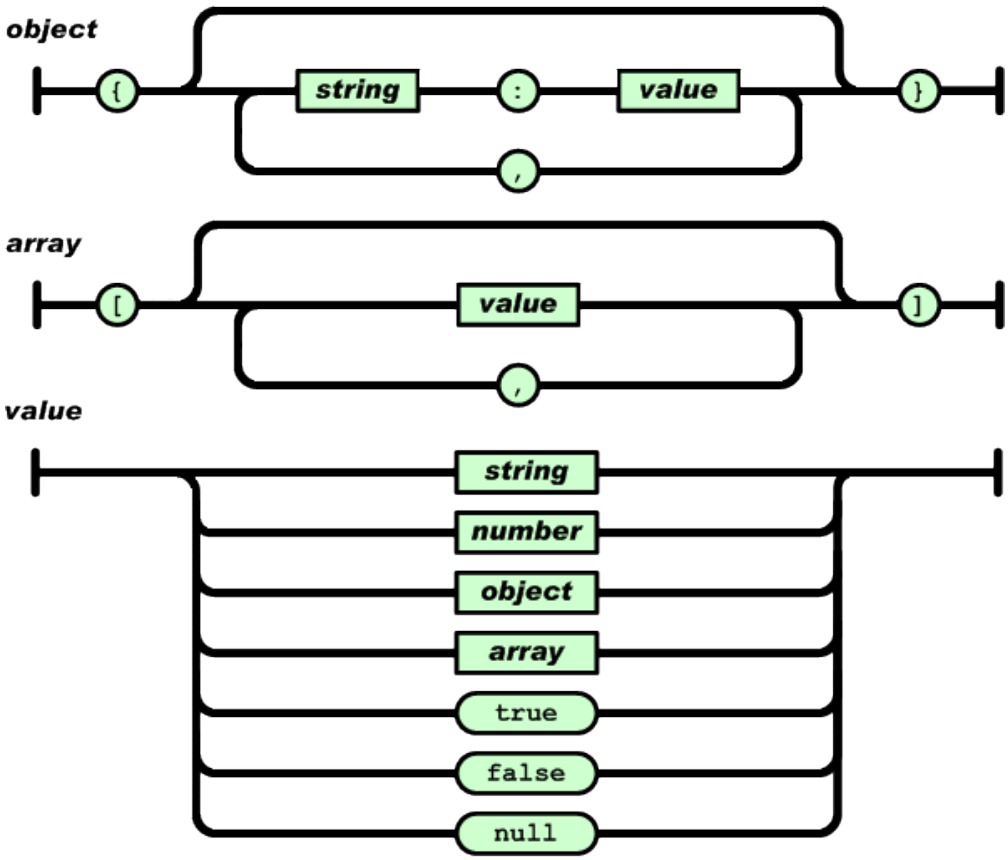


Figure 31 - JSON Object Definitions

A summary of the inputs and outputs of the management pages can be found in Table 28 - Management Page Descriptions below.

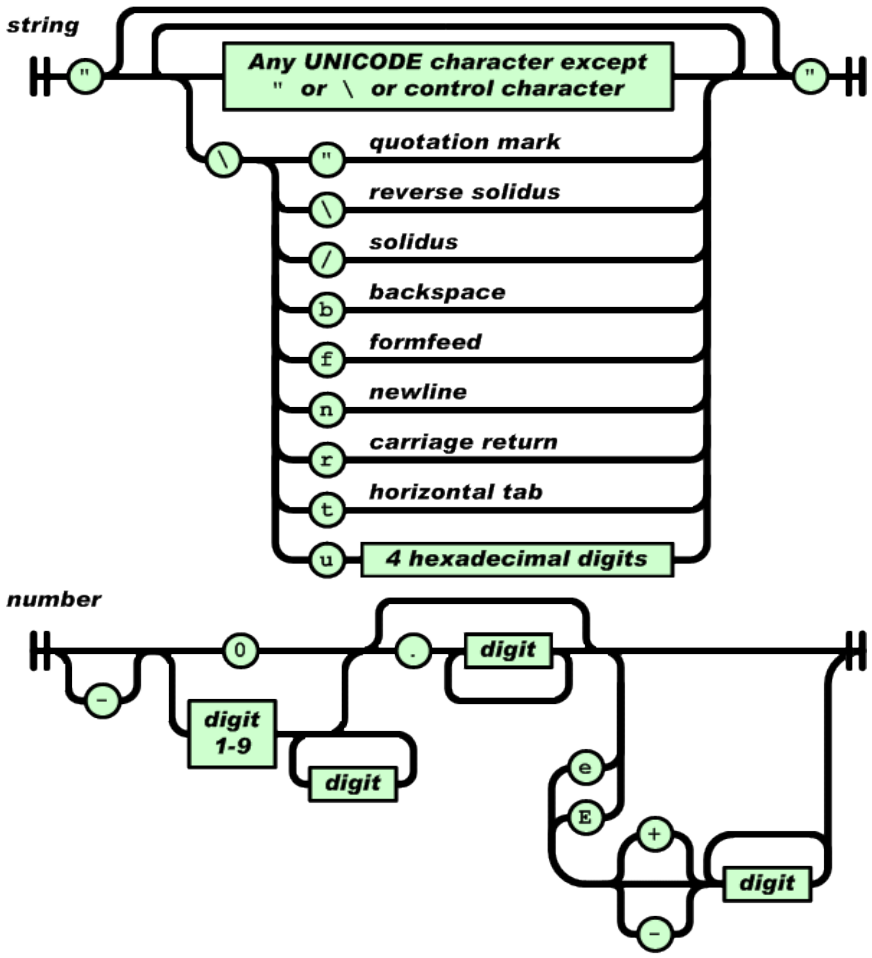


Figure 32 - JSON Type Definitions

Table 28 - Management Page Descriptions

Page Name	Required Input	Optional Input	Output	Description
Log In	• Username • Password (SHA-1 hashed)		• Authentication Key • User ID	Logs in a user and creates a session on the server
User Info	• Authentication Key • User ID		• Username • LED Color • Audio Theme	Get information about the currently logged in user.
Games List	• Authentication Key • User ID	• *showObserver* • *page_count* • *page_number*	• Games IDs • Player usernames	Get a list of all games the user can interact with.

Page Name	Required Input	Optional Input	Output	Description
Game Info	• Authentication Key • User ID • Game ID	• *moves* • *move_limit*	• Game ID • Players • Board State • Turn Number • Active Player • Turns (Optional)	Get the full state of a game, given its ID.
Create Game	• Authentication Key • User ID • Opponent Username	• *allowObserver*	• (Success) o Game ID o Players o Board State o Turn Number o Active Player • (Failure) o Error Code o Error Message	Make a new game, given a logged in user and an opponent.
Update Move	• Authentication Key • User ID • Game ID • Board State • Move Made		• (Success) o Game ID o Players o Board State o Turn Number o Active Player • (Failure) o Error Code o Error Message	Submit a move in an ongoing game.
New Account	• Username • Password • Real Name • Email Address	• LED Color • Audio Theme	• (Success) o Success Code • (Failure) o Error Code o Error Message	Create a new account using the information provided.
Pass Lost	• Username		• Email containing reset key	Request a reset key for a user who has lost their password

Page Name	Required Input	Optional Input	Output	Description
Pass Reset	• UserID • Reset Key		• (Success) o Success Code • (Failure) o Error Code o Error Message	Choose a new password for the user
Add Observer	• UserID • Authentication Key • GameID		• (Success) o Success Code • (Failure) o Error Code o Error Message	Add a user to a game as an observer.

4.3.3.1 LOG IN PAGE

The first management page requested by both the physical chess board as well as the web interface will be the log in page. The log in page takes as input a POST request from a web client with the parameters to include a plaintext username and a SHA-1 hash of that user's password. It will take the plaintext username and apply a SHA-1 hash to it as well before concatenating the result with the password's hash. The salted password hash is then hashed once again with SHA-1. The resulting string is compared to the contents of the user's password as it is stored in the *users* table of the database. If it matches, a response is created containing the user's user id number and an authentication key that must be resubmitted with each other request. This authentication key is concatenated with the user's remote IP address and is then hashed and stored in the table. Further requests to management pages must submit the authentication key correctly and be coming from the same IP address, or else the management page being requested will return a status code 403 - Forbidden error. The system cannot use the more standard method of storing authentication via client-side cookie relating to server-side sessions because the chess board is incapable of storing cookies.

4.3.3.2 USER INFORMATION PAGE

The second page that is requested by both clients is the user profile information page. This page takes as input the authentication key provided by the log in page and the logged in user's user id and returns the user's profile information, including display name, LED color choice, and audio choice. The decision to store this information or request it again when it is needed again is left up to the client. The user profile information is of such small size that the bandwidth involved in serving it is miniscule.

Because the user information page must return the user's profile, it must make a direct connection to the *users* table of the database. It requires only read (*SELECT*) access to the table, and is therefore a relatively safe operation.

4.3.3.3 GAMES LIST PAGE

The first of the real workhorse pages is the games list page. It replies to a request containing an authentication key and a user id number with a list of games that user is involved (observing or playing either black or white) and the players of those games. The page will also support a query string parameter, *showObserver*, to show all games that are open to observers which will not be utilized by the physical chess board but will be used by the web interface to allow for setting up observer status for a game currently in progress.

The games list page accomplishes this task by connecting directly to the *games* table of the database and issuing a *SELECT* query with *WHERE* clauses searching for the user's user id in the *whiteID*, *blackID*, and *obsID* columns. Every game that is returned by that query is packaged into a JSON object and returned as the response to the initial request. Since this is potentially a very long response, another two query string parameters are available for use (primarily by the physical board), *page_size* and *page_number*, allowing for paged reading of the games list.

4.3.3.4 GAME STATE PAGE

The game state page requires input of an authentication key and user id number from an authenticated user, as well as a game id number for a specific game. The page then queries the database's *games* table for information related to the given game. Finally, it returns the game state, turn number and active player, as well as the display names for both players.

The page also offers other options accessible through query string parameters. With the *moves* parameter set to true, the page also queries the *moves* table and returns the series of moves that have occurred to date in the given game. With *move_limit*, the client can limit the number of turns returned if, for example, only the last five turns are desired.

4.3.3.5 NEW GAME PAGE

The new game page requires an authentication key and the user id that corresponds with it, a bit representing the user's starting side, as well as another username. The second username is the name of the second player that is to take part in the game, who will then take the side of the match opposite the creating user. The *users* table is queried for the user id number of the user who's username was given.

If no user id number is found, then the user submitted a fictitious username, and a status code 200 response with a JSON object consisting of only an "error" object with a string value explaining the error in plaintext and an error code representing the lack of user. If, however, the user is found in the *users* table, then a new game is created by inserting a new row into the *games* table with the given users' id numbers. By default, the *allowObs* column is set to false, however a query string paramater, *allowObserver*, can be set to true to force the new game to allow observers. The new game page then requests the game state page for the newly created game and sends the information back as the response.

4.3.3.6 UPDATE MOVE PAGE

The final workhorse page is the update move page. This page takes as input the authentication key and user id number of the logged in user, as well as a game id number, and finally the new state of the board and the move actually made. The page will first check to see if the move being submitted is a legal move according to the rules of chess, and then it will insert a new record into the *moves* table with the relevant info and then update the *games* table to advance the turn (if necessary), switch active player, and set the last played field to the current time. It will then return the response from the game state page for the updated game, similarly to the new game page.

If the move is illegal for any reason, an error code is generated along with a plaintext description of the error and this object is returned to the user.

4.3.3.7 NEW ACCOUNT PAGE

The new account page is the first of the administrative management pages. The input is all of the information required by the *users* table, namely username, real name, email address, password, and optionally LED color and audio theme. The password submitted to this page will have to have already been hashed using SHA-1 (as the web-based interface will do) as a security precaution. The username, email address, LED color, and audio theme are all added in plaintext to the database, but the username is also hashed with SHA-1 as a salt, then the password hash and username hash are concatenated before being hashed a final time with SHA-1. This doubly hashed, salted string is then stored in the database. This is the only point in the server system that the hash in the user's password file is created, and along with the password reset page, is the only place it is changed at all. The page responds with either a success message or error message in a JSON object.

4.3.3.8 PASSWORD LOST PAGE AND PASSWORD RESET PAGE

The password lost page is part of the backend to the web client's password recovery system, as described below in Section 4.3.4.2. The page's function is to create the

password reset keys that are required by the password reset page and store them in the *reset* table along with the user id number of the user that requested the reset, and the time the request was made. The reset key provided by the password lost page will only be active for 30 minutes. The reset key is sent in an email to the email address stored in the database for the user. The email contains a link to the web-client's password reset page that is the front end for the password reset management page.

The password reset page is the management page that takes the reset key provided by the password lost page and the user's new submitted password and determines if the they key is valid. If it is, the new password is hashed as in the new account page and stored into the *users* database in place of the old password.

4.3.3.9 SET OBSERVER PAGE

The final administrative page is the set observer page. This page asks for an authentication key and user id number from the logged in user, as well as a game id number. The page then checks to see if the game in question is set to allow observers, and if it is, adds the user to the game as an observer. The page responds with either a success message or error message in a JSON object.

4.3.4 WEB INTERFACE

The only non-board interface provided for Deep RGB in this project will be the web-based board. The web application will allow a user to connect via username and password and will then access the database to display all games the user is involved in. The games will be listed in an easy to navigate manner allowing the user to select which game he would like to play. A table area will be created displaying, by default, the games the user is involved in that are currently in-progress. A small area to the side of the interface will allow the user to instead display all games that he has been involved with, in-progress or not, as well as sort the games listed by opponent, time the last move was made, time the game was started, or total number of moves. This area will also link to the profile editor page described below.

Once a game is chosen, the user will be navigated to a page displaying the current state of that game on a chess board graphic. If it is the user's turn to make a move, there will be a notification of this and the user's chess pieces will be interactive. Selecting a piece will display that piece's available, legal moves by highlighting the squares to which it can be moved. The user can then select a square to make the given move, or re-select the piece to de-select it and return to the default state of the board. Once a move has been chosen, the page will submit this action to the server by making a request to the same page the physical board uses.

A second page will be available from the main log in page allowing the user to edit his or her profile. The profile will store the user's display name and real name, as well as the

color the physical chess board uses to identify that user and the audio the board plays when that user is playing. The display name is a simple (less than 15 character) nick name that will be displayed to the other player in any games the user is involved with, as well as observers to those games. The user's real name is requested so as to greet the user on log on to the website or physical board. The color used to identify the player is editable on the profile editor as well by selecting hexadecimal values for each of red, green, and blue to choose a specific color and hue. A color selector will be available to assist the user in choosing a color. Finally, the profile editor page will allow for a dropdown choice of audio themes for use on the physical board.

4.3.4.1 LANDING PAGE

The landing page for the web application will be the initial log in page. Log-ins will be simple username and password combinations with both username and password chosen by the user. The elements of the landing page will include text fields to allow the user to enter their username and password, a submit button to send the log in information to the server, along with links to create a new account and reset the password, if the user has forgotten theirs. Unless a user has logged in using this page, every other page in the system will redirect the user to this landing page to request that he or she logs in. A mock up of the page is found below, in Figure 33.

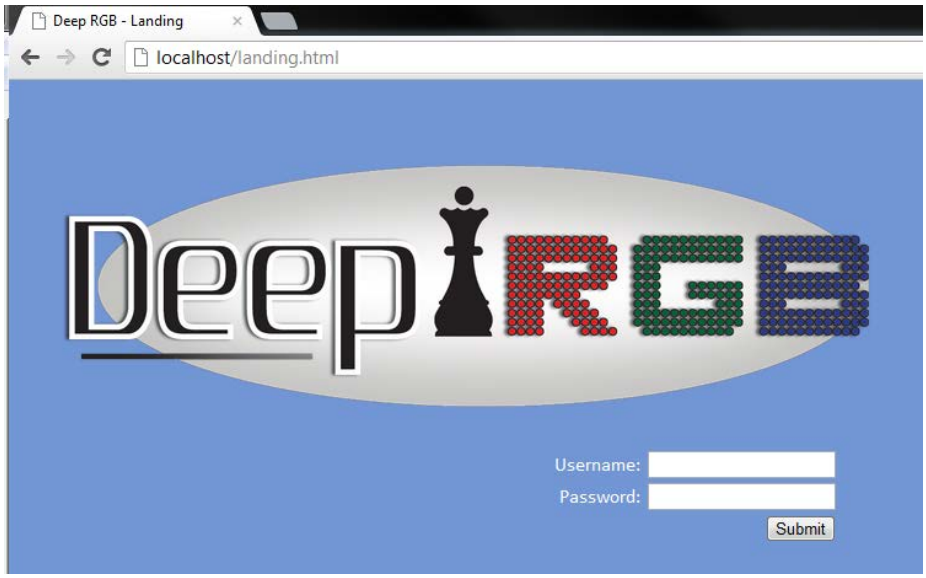


Figure 33 - Landing Page Mockup

The landing page backend will incorporate the secure sign-on algorithms to allow for a user to input his or her username and password only once per session and maintain a connection to the server for the remainder of his or her play time. When the page is submitted, the username and password will be sent to the server via the log in management page, described in Section 4.3.3.1. An authentication key is created and

stored in the user's session, allowing him access to the remainder of the web application.

4.3.4.2 PASSWORD RECOVERY AND RESET PAGE

The link from the landing page to recover the user's password takes him or her to the password recovery page. This page requests the user's username and email address. If the data provided by the user matches a record in the users table of the database, an email is sent to the user's email address with a link to another page that contains an authentication key that is good for only one use and only 30 minutes. If the user does not use the link to navigate to the password reset page within the time limit, the authentication key will no longer function and they will be unable to reset their password without requesting another password reset email. If the user does access the password recovery page within the given time limit, they will be offered a single text field and allowed to submit a new password. That password will be hashed and added to the user's record in the users table of the database and the user will be redirected back to the landing page where they will be able to log in with their username and this new password.

4.3.4.3 ACCOUNT REGISTRATION PAGE

A new account will only be creatable from the web interface. A potential user of the physical chess board will, then, be required to access the web interface at least once to establish a profile. The account registration page will be accessible from the landing page and will allow an un-registered (or registered, if they desire a new username) user to create an empty user account for use on the system. All relevant user information (username, password, email address, LED color and audio theme) are available as fields to fill out, with only username, password, and email address being required. If submitted, the LED color and/or audio theme will be stored in the user's profile, otherwise a default color will be chosen for the LED System and no audio theme will be used for that user. A mockup of the account registration page is seen in Figure 34.

The account registration page utilizes the LGPL-licensed JSColor library provided freely at http://jscolor.com/. This library allows for the attachment of JavaScript color pickers onto text fields that allow for the choice of a hexadecimal color value by visually inspecting the color spectrum allowed. The library is incorporated into the page through the use of a single JavaScript file (jscolor.js) that is installed into the website root along with a few helper files and is used by appending a special class onto any <input> tags that require the color picker.

The profile edit page will be a derivative of the account registration page. It will request the same information (except username and password), and submit that information to the new account management page with the authentication token received from

logging in. In this case, the new account management page will only update the corresponding row in the *users* table instead of creating a new one.

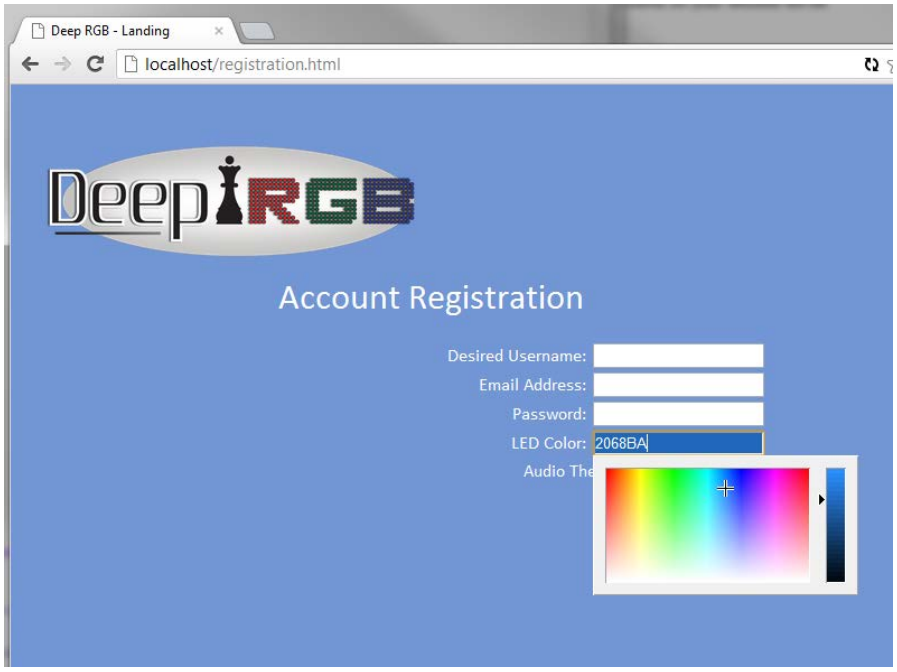


Figure 34 - Account Registration Mockup

4.3.4.4 GAMES HOME PAGE

The games home page is the primary landing screen for a logged in user. The main part of the screen displays a list of games that are active and including the logged in user, by default. A filter checkbox is available in the sidebar to also display all games the user is not involved with but are open to observers. The games home gets this list from requesting the games list management page with appropriate query string. Each item in the list will be linked to open the game that it is referring to, as either a player or an observer, as is appropriate. If the Show Observable filter is on and the user clicks on a game they are not involved with currently, they will automatically request to be added as an observer and will be taken to the game page thereafter. A mockup of the games home page is available below as Figure 35 - Games Home Mockup.

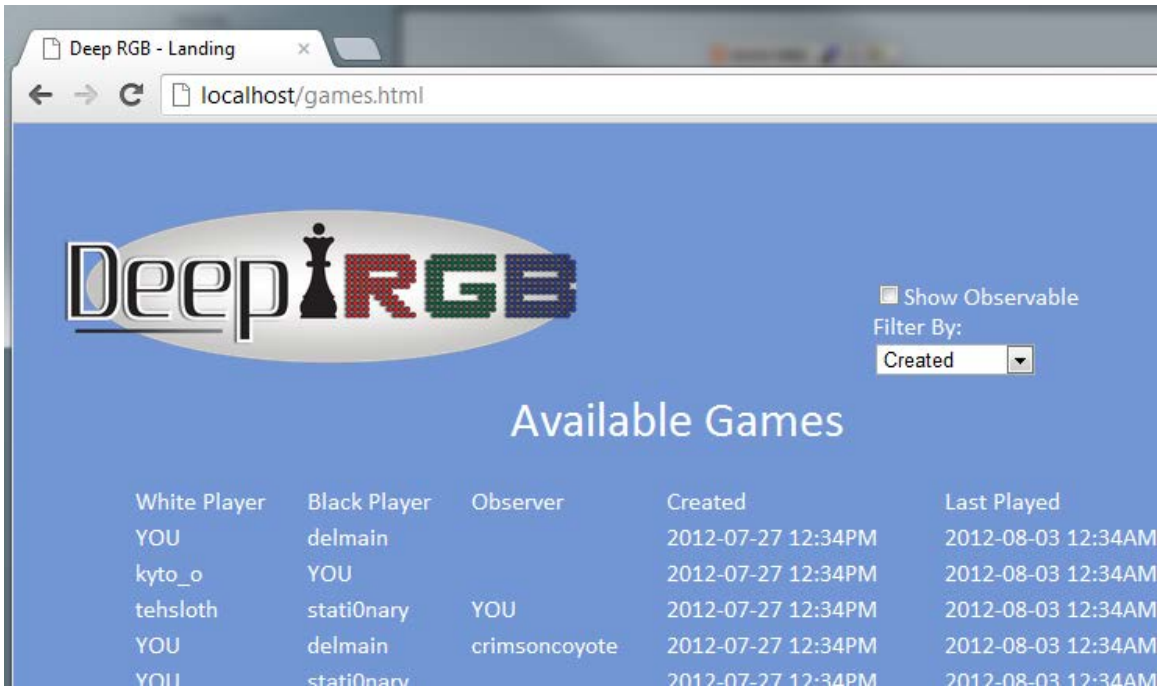


Figure 35 - Games Home Mockup

4.3.4.5 ACTIVE GAME PAGE

The active game page will be the workhorse page of the web client, handling all the actions involved in playing a game of chess. This page will user primarily AJAX (Asynchronous JavaScript and XML) requests to get updates from the server on the state of the game as well as send updates to the server. JavaScript functions will be assigned dynamically to squares on the virtual chess board so as to have a fluid and well-mannered interface to the chess game. The variables in use are defined in TABLE YYY, and the functions used are defined in TABLE XXX below.

Table 29 - Game Page Variables

Variable Name	Type	Description
currentState	JSON Object	Holds the current state of the board in the same format as is returned by the game state management page.
board	string	The board's current state, expressed in Forsyth-Edwards Notation (FEN).
activePlayer	bool	The player who's turn is active. True is white, false is black.
turnNum	int	The current chess turn number.
activeCellX	int	Coordinates of the currently active cell. Both set to -1 when no cell is being selected.
activeCellY	int	
activePiece	string	The piece located in the currently active cell.

Variable Name	Type	Description
refreshTimer	int	Number of seconds between calls to getState() when user is not active. Defaults to 30.

Table 30 - Game Page Functions

Function Name	Inputs	Returns	Effect
getState()			Requests a full state update from the game state management page. When the request is fulfilled, it sets the *currentState* variable, calls *unpackState*() to unpack the JSON into their respective local variables, and calls *displayState*() to refresh the view.
unpackState()	currentState	board activePlayer turnNum	Read the JSON object stored in *currentState* and extract the board state, active player, and turn number and store them in local variables for easier manipulation.
packState()	board activePlayer turnNum	currentState	Reverse of *unpackState*(). Take the local variables and pack them into a JSON object for return to server.
displayState()		View	Read the state saved in the *currentState* variable and sets the page up correctly to display that state at the beginning of the current player's turn
selectCell()	Coordinates	activePiece activeCellX/Y	The default action on every square containing the active player's chess pieces to start. Activating sets the active variables and disables *selectCell*() on all other cells. Enables *deselectCell*() on *activeCell*. Calls *showMoves*()
deselectCell()	Coordinates		Resets the active variables and places *selectCell*() back on all cells containing the active player's chess pieces.
showMoves()	activeCellX/Y activePiece		Highlights all cells the active piece can move to from the active cell. Does not display moves that are illegal (leaving yourself in check). Activates *makeMove*() on highlighted squares.

Function Name	Inputs	Returns	Effect
makeMove()	Coordinates	success (boolean)	Attempts to make the move selected by submitting a request to the update move management page. Will first call *packState*() to package the current state and chosen move into a JSON Object to be sent. Will receive the success or error code returned from the management page and handle properly.
refreshState()	refreshTimer		Active when user is not active player. Will call *getState*() every *refreshTimer* seconds.

A very rough mockup of the game page is available as Figure 36 below.

Figure 36 - Game Page Mockup

4.3.5 SERVER CONFIGURATION

The Deep RGB server will be running on a temporarily donated PC that is owned by a group member. As such, it's hardware specifications and some software specifications are set. As mentioned previously, the platform is a Windows machine, running Windows 7 Service Pack 1, specifically. The software necessary to run the server is explained in Table 31 - Server Hardware and Software below, along with the hardware specification of the server.

Table 31 - Server Hardware and Software

Item	Manufacturer	Version/Value	Description
Server Operating System	Microsoft	b7601	Windows 7 with Service Pack 1
Http Host	Microsoft	7.0	Internet Information Services 7.0
.NET Interpreter	Microsoft	4..3039	.NET Framework 4
Central Processing Unit	Intel	Q9400	Intel Core2-Quad. Four cores operating at 2.66GHz
Random Access Memory	OCZ	8GB	8GB OCZ Gold Series DDR2 RAM

4.3.6 CLIENT CONFIGURATION

As a web-based application, the client configuration required to interact with the Deep GRB web interface is very basic. Almost any modern device will be able to connect to the web interface and play a game. The server for Deep RGB will be hosted externally in Orlando, FL through a dynamic DNS based URL, allowing for connection through a simple URL (for example: http://deep-rgb-ucf.com/). Any user wishing to use the web interface to Deep RGB would need only enter the URL into their web browser and they would be sent directly to the landing page, ready to log in.

The minimum specifications for access to the Deep RGB web interface are very lax, as it is designed to be compatible with the majority of users. Most users interested in the system will already be running on a somewhat modern computer. Most modern desktop and laptop computers should have all the software necessary to access DeepRGB without any further installation. The minimum support specifications are listed below.

Supported Operating Systems:

- Windows 2000/XP
- Windows Vista/7
- Mac OSX v10.5 "Leopard" or newer

Supported Desktop/Laptop Web Browsers

- Microsoft Internet Explorer 6 or later
- Mozilla Firefox 3.6 or later
- Opera Software Opera 7 or later
- Mac Safari 3 or later
- Google Chrome 3 or later

Support Mobile Browsers: (All mobile browsers must be entirely up to date for phone compatibility issues)

- Google Android Browser
- Google Chrome Mobile
- Opera Software Opera Mobile

5 TESTING AND PROTOTYPING

5.1 HARDWARE TESTING

5.1.1 MICROCONTROLLER UNIT

To test the microcontroller and it's IDE we will need to write code for subroutines that test every pin available. In order to perform these tests it is recommended that the microcontroller be connected to the PC. This is to allow any error codes to return to the computer and be displayed on the screen. To test the IDE, we can enter some incorrect segments of code to ensure the debugger is working as intended. A quick and easy way to test the digital and PWM pin-outs is to write a code that will turn on or fade an LED depending on the pin-out. If the LED turns on or fades, the pin-out should be working correctly. As a second test, the output of the LED can be measure by an oscilloscope or a multimeter to ensure the proper output voltage is being supplied to that pin. If the multimeter measures 5V for digital HIGH and 0v for digital LOW, we can rest assure that the digital pin-outs work as intended. Since PWM has 256 levels of sensitivity, the output for every level should increase by approximately 0.0195 volts and show 5 volts output for a level of 255. The UART pins can be tested in conjunction with the wireless connection device since they use the UART pins to communicate. As for the SPI bus, the audio module can be used to test these pin-outs since it uses the complete SPI bus in order to send and receive data from the device.

5.1.2 HALL EFFECT SENSOR TESTING

First a Hall Effect sensor should be acquired and placed in a breadboard in order to be tested. Once the sensor has power it should be programmed to the specifications given in the design section and a piece should be moved around it at measured distances while recording the output of the sensor. Once the sensor is sure to work for this application it should be arranged in a small matrix with several other sensors the same way as they would be connected in the final configuration. The MUXs should be connected to the matrix and should be used to turn each one on and off when the sensor is on it should be recorded how long it takes to get an accurate reading so that can be programmed into the MCU as well as how they react to multiple piece in the area around them. The MCU should then be connected to the same matrix and used to run the same tests to assure that all components are functioning properly together. Every sensor should undergo a test to assure that they all perform within acceptable limits and do not give bad readings or readings unproportional to the other sensors. Finally the matrix should be fully constructed and every sensor should be tested to make sure they all come on and are not giving false readings after this the system is ready for use.

5.1.3 AUDIO TESTING

The audio shield should be connected to the MCU so that it can be run and tested to assure that all the commands from the MCU are being followed. Then the SD card should be loaded with a sound file and run by the audio shield and output to speakers to demonstrate that the audio shield is working in conjunction with the MCU and that the SD card is functioning properly. Now the server should run a scenario where each type of alert is called while the music is playing to ensure that the alert sounds and interrupts are functioning correctly. Finally if it is to be used the switch should be placed to separate the speakers from the audio shield and the auxiliary 3.5mm input jack. This switch should be tested by giving the server a certain user command to switch from the external input to the audio shield all the while performing the previous test to make sure all components are working together smoothly. After this test the audio system is ready to be installed into the board.

5.1.4 LED TESTING

In order to test the LED array in and make sure that it works correctly, sufficient knowledge about its components is necessary. To begin the LEDs should be tested by placing one into a breadboard and connecting a resistor to each pin. Then power concurrent with the levels referenced in the datasheet should be applied to each leg of the circuit to see how the LED responds. Once the characteristics of the LED are familiar there should be several set up on a breadboard so that they should all be able to function like the original test LED. Then one of the shift registers should be placed on the board and each leg of the LEDs should be wired to an output on the register with all the grounds connected to one common. Then the shift register should be connected to

the MCU which should be programmed to make the LEDs operate in a specific pattern. Using this method the code in the MCU can be changed so that the effects of PWM are more familiar allowing for better coding of the MCU. Finally the LEDs should be set up as described in the design section and the shift registers should be connected as well. Once the matrix is connected in its final configuration the MCU should be connected to it and a program run to test each LED to make sure the matrix is functioning properly.

5.1.5 POWER SUPPLY TESTING

The main purpose of the power supply test is to make sure that every component of Deep RGB system received appropriate amount of voltage, with an appropriate amount of current, at an acceptable level of stability. Our test procedure consists of determining power level of the power supply after the DC conversion. Next, we need to check voltages from each supply line. After that, we need to connect all subsystems to the power supply lines and re-check voltage level in each line again, with loads applied. This provides certainty that all elements can work together simultaneously without overloading the power supply.

5.1.6 DISPLAY TESTING

The display we are using is not connected directly to the MCU because the display has a pre-installed display driver which mostly takes care of all operations required to display text on the LCD screen. All we need to do to make display work is to connect display unit data outputs to the MCU and connect the remaining display pins to the power supply and to the ground, as directed by the display driver instruction set. During the display testing procedure, we need to make sure that display is correctly connected to the MCU and to the power supply. The final test is to check if the display is showing the information the MCU sends it.

Our first step of the display test procedure was to connect display to the MCU and to the power supply. We need to make sure that all connections are made in a specified way. Next we needed to turn on power supply so display unit and MCU have power. After that we need to establish communication between display and MCU, as a result MCU should send activation command to establish connection with display unit. We needed to check ability of the display to show determined words to do so we sent commands from the MCU to the display to show “hello world “and as a result we should see “hello world” on the display. After that we can test backlight option in the display as a result and we should be able turn backlight on and off. At the end of the display test procedure we needed to turn off power supply.

5.1.7 WIRELESS MODULE

The Wi-Fi module comes with a testing procedure to ensure that a secure and strong connection is made to a server. After connecting the UART cables to their proper location, a test code can be downloaded from the manufacturer's website. While connected to a computer, this test will check the firmware of the module to ensure that is it updated to the most recent version. The test code will then dump the memory of the wireless module and enters a command mode. When the command mode is activated, we can allow it to scan for a network and try to connect to one. If the connection is made, the network's information is printed along with the module's IP address and MAC address. If the connection gets timed out or fails for any reason, a corresponding error will be printed that explains the reason for failure. When a connection is made, the module can be commanded to ping an IP address to ensure packets are being sent and received.

5.1.8 MAGNETIC PIECE-MOVER TESTING

A properly functioning magnetic piece-moving system is an important part of Deep RGB. We need to make sure that this system satisfies all requirements, and this requires that we use a detailed test procedure wherein we check all working elements of the magnetic piece-moving system. First, we needed to check all mechanical elements of the system. This requires moving the stepper motors to every X and Y coordinate, during which time we need to make sure that all parts move smoothly. Next we need to check all the connections from the stepper motors to the motor drivers and MCU. After that we need to establish a connection with MCU. The next test required is to determine if stepper motor moves at the same speed and makes the same number of steps as it was specified by microcontroller. Finally, we need to check the electromagnet to be sure it is turning on and off according to the MCU's directions.

We start the magnetic piece-moving system testing by making sure that mechanical mechanism moves each of the X and Y-axiss. It works if mechanism movement is smooth along both axes. Our next step is to connect stepper motors to the power outlet and to the MCU, ensuring all connections are in appropriate order. After we check all connections we need to turn on power supply and it should supply 12 V DC for stepper motors. Next we need to setup communication between stepper motors and MCU. As a result, the controller should recognize connection to the stepper motor drivers. Our next step is to test the stepper motors ability to follow a specified number step commands from the MCU, ensuring each motor makes one full revolution for each step. An important part of the test procedure is to test the positioning system's ability to follow a specified path, allowing the positioning system should move the electromagnet in a predetermined path set by MCU. Before we move to the next test, we need to turn off the power supply. After we do that we need to connect the electromagnet to the power supply and to the MCU and make sure all wires are connected in appropriate order. To start the next step, we need to turn power back on to the electromagnet and stepper motors. For the last test, we test the electromagnet behavior. The result should

be that the electromagnet turns on at the determined location and turns off after we move it into next location.

5.1.9 PCB

Since our PCB will be designed using Eagle Cadsoft, we can implement testing pads. Testing pads are small conductive circles that are used to ensure that a pathway is conductive and unobstructed. These conductive test pads as seen in Figure 37 allow us to use a multimeter with a short circuit function in order to check if the circuit is being completed. This is a primitive way of testing a circuit, because a pathway could show up as a short but still warp a signal passing through it. PCB manufacturers tend to test for these faults by using two probes and sending a signal pulse through the pathway. If the same signal is received on the receiving probe, then the pathway is unobstructed. If the signal gets warped along the way then there could be some sort of interference in the pathway.

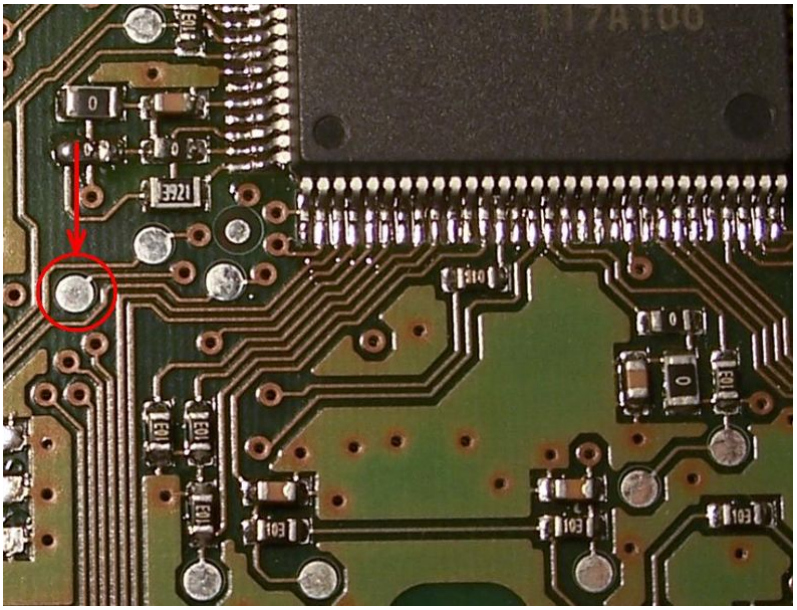


Figure 37 – A close-up on the testing pads used to test pathways on PCBs reprinted with permission from Wikipedia.

5.2 SERVER TESTING - 1 PAGE

Software testing of the server system is both incredibly important and incredibly easy because of the management page set-up utilized. Since each component is already sub-

divided to a single task for a single file, there is no real risk of cross-contamination between processes since each occurs in its own space.

5.2.1 MANAGEMENT PAGES

Each management page can be tested separately through the use of custom built HTML pages supplying the required info through an HTML form with the POST action making a request to the management page that is being tested. For every page that requires an authentication key and user id number (all but Log In, New Account, Pass Lost, and Pass Reset), the first test to be performed is that of submitting to the page without a working key-id combo.

5.2.1.1 LOG IN PAGE

The log in page has a simple test procedure. Three tests must be made. The first test is to submit a correct user name and password hash and check to ensure the page returns a code 200 - OK response along with the provided user's user id number and an authentication key. The second test is the reverse, sending an incorrect username with an incorrect authentication key, checking for the expected code 403 - Forbidden response from the server. The final test is supplying a username which is correct and an authentication key that is also correct but for a different user. The expected result is another 403 - Forbidden response.

5.2.1.2 USER INFO PAGE

The test for the user info page is simply requesting the page. If the JSON object returned contains the correct profile information for the logged in user, the test is a success.

5.2.1.3 GAMES LIST PAGE

There are two tests required for the games list page. The first is requesting the page while logged in, without passing the optional showObserver parameter, but with using the page_count and page_number parameters. If a paged result is received containing the predefined number of games of which the user is already a part, then it is a success; any other result is a failure. The second test is for the showObserver parameter; for that test, the games list page must return a list of all games the user is involved in as well as all games for which allowObs is true and there is not yet an observer.

5.2.1.4 GAME INFO PAGE

The test for the game info page is also two-fold. The first test is to check for the info being returned correctly without either parameter turned on. If the game state is received as expected without any moves array, the test is a pass. The second part of the test is to send the same request but with the moves parameter set to true and the moves_limit parameter set to a value higher than the number of moves made; if the

same game state is received along with a moves array containing each move, the test is a success.

5.2.1.5 CREATE GAME PAGE

There are two tests for the create game page. The first is an expected failure by sending a non-existent username. If the server responds with an error code, we know the test is passed. The second request to the create game page should be for a username which is correct. The expected response then is a game state JSON object containing a brand new game.

5.2.1.6 UPDATE MOVE PAGE

There are several tests required for the update move page. Each parameter in turn must be set as a failure condition with the rest correct to ensure any failure point returns the correct error code and message. Finally a valid move must be POSTed to the server to make sure the page can accept a valid move when one is presented.

5.2.1.7 NEW ACCOUNT PAGE

Only a single test is required. If all the required information is sent to the management page, then a new account should be created in the database and a success code should be returned.

5.2.1.8 PASS LOST & PASS RESET PAGES

A set of combined tests must be performed for the password recovery system pages. A user must first request a reset key before attempting to reset his password within 30 minutes, then again before attempting a reset after 30 minutes have passed. After that, a user must request a reset key, and then a second user must attempt to use that key to reset their account. Finally, a user must try to reset his password using an invalid reset key and ensure the attempt fails.

5.2.1.9 ADD OBSERVER PAGE

The singular test for the add observer page is to attempt to add the user to an available game as the observer, then check the games list page for that user to ensure that game appears properly.

5.2.2 CHESS ENGINE PROCESS

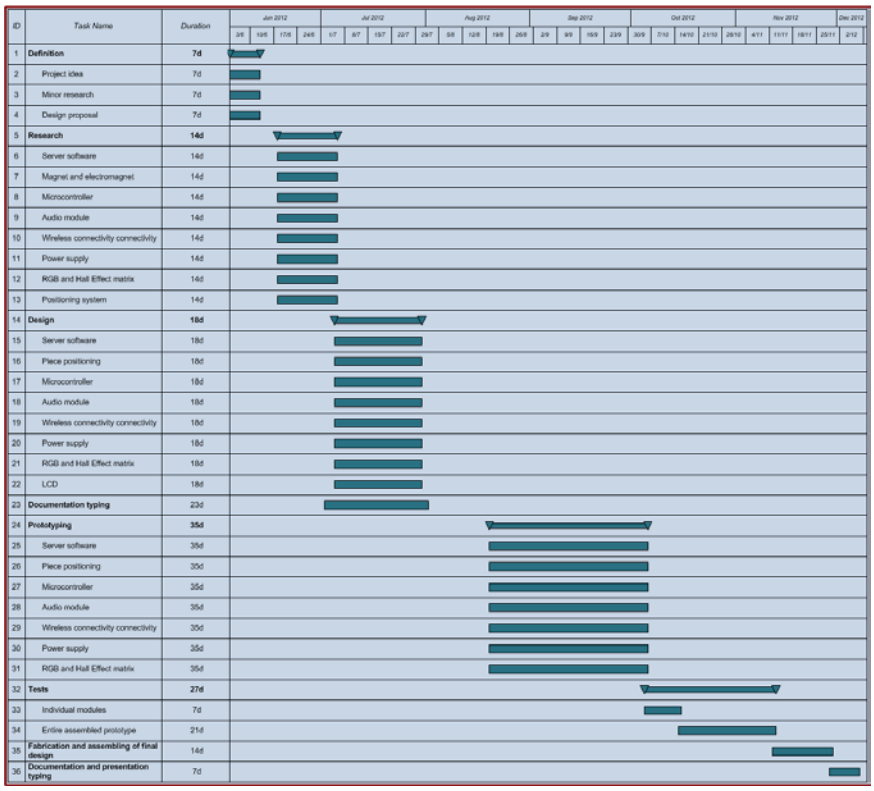
The test for the chess engine process involves simply sending a board state to the chess engine thread and asking it for a response. If a move is returned from that call, the chess engine thread is communicating with the chess engine correctly.

5.2.3 WEB CLIENT

Testing the web client is an integration test of the entire server system. Several full games of chess must be played against computer opponents while attempting each of the various chess moves and illegal moves, ensuring that the system allows legal moves (even unusual ones) and discards illegal moves.

6 ADMINISTRATION

6.1 MILESTONES



ID	Task Name	Duration
1	**Definition**	**7d**
2	Project idea	7d
3	Minor research	7d
4	Design proposal	7d
5	**Research**	**14d**
6	Server software	14d
7	Magnet and electromagnet	14d
8	Microcontroller	14d
9	Audio module	14d
10	Wireless connectivity connectivity	14d
11	Power supply	14d
12	RGB and Hall Effect matrix	14d
13	Positioning system	14d
14	**Design**	**18d**
15	Server software	18d
16	Piece positioning	18d
17	Microcontroller	18d
18	Audio module	18d
19	Wireless connectivity connectivity	18d
20	Power supply	18d
21	RGB and Hall Effect matrix	18d
22	LCD	18d
23	**Documentation typing**	**23d**
24	**Prototyping**	**35d**
25	Server software	35d
26	Piece positioning	35d
27	Microcontroller	35d
28	Audio module	35d
29	Wireless connectivity connectivity	35d
30	Power supply	35d
31	RGB and Hall Effect matrix	35d
32	**Tests**	**27d**
33	Individual modules	7d
34	Entire assembled prototype	21d
35	**Fabrication and assembling of final design**	14d
36	**Documentation and presentation typing**	7d

Figure 38 - Gantt Chart

ID	Task Name	Start	Finish	Duration
1	**Definition**	**6/4/2012**	**6/12/2012**	**7d**
2	Project idea	6/4/2012	6/12/2012	7d
3	Minor research	6/4/2012	6/12/2012	7d
4	Design proposal	6/4/2012	6/12/2012	7d
5	**Research**	**6/18/2012**	**7/5/2012**	**14d**
6	Server software	6/18/2012	7/5/2012	14d
7	Magnet and electromagnet	6/18/2012	7/5/2012	14d
8	Microcontroller	6/18/2012	7/5/2012	14d
9	Audio module	6/18/2012	7/5/2012	14d
10	Wireless connectivity connectivity	6/18/2012	7/5/2012	14d
11	Power supply	6/18/2012	7/5/2012	14d
12	RGB and Hall Effect matrix	6/18/2012	7/5/2012	14d
13	Positioning system	6/18/2012	7/5/2012	14d
14	**Design**	**7/5/2012**	7/30/2012	**18d**
15	Server software	7/5/2012	7/30/2012	18d
16	Piece positioning	7/5/2012	7/30/2012	18d
17	Microcontroller	7/5/2012	7/30/2012	18d
18	Audio module	7/5/2012	7/30/2012	18d
19	Wireless connectivity connectivity	7/5/2012	7/30/2012	18d
20	Power supply	7/5/2012	7/30/2012	18d
21	RGB and Hall Effect matrix	7/5/2012	7/30/2012	18d
22	LCD	7/5/2012	7/30/2012	18d
23	**Documentation typing**	7/2/2012	8/1/2012	23d
24	**Prototyping**	8/20/2012	**10/5/2012**	**35d**
25	Server software	8/20/2012	10/5/2012	35d
26	Piece positioning	8/20/2012	10/5/2012	35d
27	Microcontroller	8/20/2012	10/5/2012	35d
28	Audio module	8/20/2012	10/5/2012	35d
29	Wireless connectivity connectivity	8/20/2012	10/5/2012	35d
30	Power supply	8/20/2012	10/5/2012	35d
31	RGB and Hall Effect matrix	8/20/2012	10/5/2012	35d
32	**Tests**	**10/5/2012**	**11/12/2012**	**27d**
33	Individual modules	10/5/2012	10/15/2012	7d
34	Entire assembled prototype	10/15/2012	11/12/2012	21d
35	**Fabrication and assembling of final design**	11/12/2012	11/29/2012	14d
36	**Documentation and presentation typing**	11/29/2012	12/7/2012	7d

Figure 39 - Gantt Details

6.2 BILL OF MATERIALS

Table 32 – Bill of Materials for development stage.

Item	Distributor	Part Num.	Quantity	Total Price
Arduino Atmega 2560	Sparkfun	DEV-11061	1	$58.95
WiFly GSX	Sparkfun	WRL-10050	1	$84.95
Neodymium magnets	Amazon	B001ANVAHI	1 (pack of 100)	$4.59
16x2 LCD display	Donated by team member	LM1602C	1	$0.00
Stepper motors	Sparkfun	ROB-09238	2	$29.90
Rack Gearbox Bracket (2-pack)	Vexrobotics	275-1188	1	$9.99
Advanced Gear Kit	Vexrobotics	276-2184	1	$19.99
Amico Electromagnet	Amazon	B005F79TIW	1	$6.82
Sparkle power supply	Newegg	N82E16817103064	1	$37.99
MP3 Player shield	Sparkfun	DEV-10628	1	$39.95
Stepper motor driver	Pololu	1183	2	$39.90
Analog multiplexer	Analog	ADG406BNZ	4	$6.60
Allegro Hall Effect Sensor	Digikey	A1360LKTTN-T	117	$281.43
Shift registers	Digikey	296-14857-1-ND	10	$4.94
RGB LEDs	Sparkfun	YSL-R596CR3G4B5W-F12	1 (pack of 100)	$59.95
Custom PC functioning as server	Donated by team member	None	1	$0.00
Server OS	Donated by team member	None	1	$0.00
Visual Studio 10	Donated by team member	None	1	$0.00
SQL Server Browser	Donated by team member	None	1	$0.00
Chrome Developer Toold	Google Inc.	None	1	$0.00
Total price without tax or shipping				$692.36

Table 33 - Bill of materials for final design stage.

Item	Distributor	Part Num.	Quantity	Total Price
PCB	4PCB	None	1 order	$60

ATmega 2560	Digikey	ATMEGA2560V-8AU	1	$19.97
ATmega16-U2	Digikey	ATMEGA16U2-AU-ND	1	$3.71
Various main board components(USB socket, headers, etc)	Digikey Sparkfun Ebay	None	1	$75
Rover RN131C	Digikey	740-1036-ND	1	$40.15
Stepper motor driver IC	Digikey	620-1140-2-ND	2	$7.02
MP3 and MIDI Codec	Sparkfun	VS1053B	1	$19.95
Total price without tax or shipping				$225.80

7 APPENDICES

7.1 BIBLIOGRAPHY AND CITATIONS

7.1.1 WORKS CITED

Arduino. (n.d.). *Arduino Mega 2560*. Retrieved 7 15, 2012, from Arduino: http://arduino.cc/en/Main/ArduinoBoardMega2560

Arduino. (n.d.). *Arduino-mega2560_R3-schematic.* Retrieved 7 25, 2012, from Arduino: http://arduino.cc/en/uploads/Main/arduino-mega2560_R3-schematic.pdf

Kgrr. (n.d.). *Wikipedia.* Retrieved 7 2, 2012, from Wireless mesh network: http://en.wikipedia.org/wiki/Wireless_mesh_network

Microchip. (n.d.). *PIC18 Explorer Board*. Retrieved 7 21, 2012, from Microchip: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en535770

Microchip. (n.d.). *PIC18F46K80* . Retrieved 7 19, 2012, from Microchip: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en550197

Networks, R. (n.d.). *WiFly GSX/EZX.* Retrieved 6 29, 2012, from Roving Networks: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/WiFi/WiFly-RN-UM.pdf

RSparkfun. (n.d.). *WiFly GSX Breakout Board*. Retrieved 8 3, 2012, from Sparkfun: https://www.sparkfun.com/products/10050

Sparkfun. (n.d.). *Arduino Mega 2560 R3*. Retrieved 7 21, 2012, from https://www.sparkfun.com/products/11061: https://www.sparkfun.com/products/11061

Sparkfun. (n.d.). *Bluetooth Modem - BlueSMiRF Silver.* Retrieved 7 6, 2012, from Sparkfun: https://www.sparkfun.com/products/10269?

Sparkfun. (n.d.). *RN-131G Breakout-v13.* Retrieved 7 5, 2012, from Sparkfun : http://www.sparkfun.com/datasheets/Wireless/WiFi/RN-131G%20Breakout-v13.pdf

Sparkfun. (n.d.). *XBee 1mW Chip Antenna - Series 1 (802.15.4).* Retrieved 7 3, 2012, from Sparkfun: https://www.sparkfun.com/products/8664?

TI. (n.d.). *MSP430FR5739*. Retrieved 7 16, 2012, from TI: http://www.ti.com/product/msp430fr5739#description

TI. (n.d.). *MSP-EXP430FR5739 Experimenter Board*. Retrieved 7 21, 2012, from TI: http://www.ti.com/tool/msp-exp430fr5739

Wikepedia. (n.d.). *Printed circuit board.* Retrieved 7 22, 2012, from Wikepedia: http://en.wikipedia.org/wiki/Printed_circuit_board

C-2. (n.d.). *RL5-RGB-C-2*. Retrieved 7 13, 2012, from Super Brigh LEDs: http://www.superbrightleds.com/moreinfo/component-leds/5mm-rgb-clear-tricolor-led-wide-angle/976/

RGB-C. (n.d.). *RL5-RGB-C*. Retrieved 7 13, 2012, from Super Bright LEDs: http://www.superbrightleds.com/moreinfo/component-leds/rl5-rgb-clear-tricolor-led/298/

C10. (n.d.). *YSL-R596CR3G4B5C-C10*. Retrieved 7 13, 2012, from Sparkfun: https://www.sparkfun.com/products/105

F-12. (n.d.). *YSL-R596CR3G4B5W-F12*. Retrieved 7 13, 2012, from Sparkfun: http://www.sparkfun.com/datasheets/Components/LED/YSL-R596CR4G3B5W-F12.pdf

74HC595. (n.d.). *74HC595*. Retrieved 7 13, 2012, from Sparkfun: https://www.sparkfun.com/products/733

1302. (n.d.). *Allegro A1302UA*. Retrieved 7 20, 2012, from Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?wt.z_cid=ref_hearst_0211_buynow&mpart=A1302KLHLT-T&v=620&cur=USD

90215. (n.d.). *Melexis MLX90215*. Retrieved 7 20, 2012, from Melexis: http://www.melexis.com/ProdMain.aspx?nID=12

1384. (n.d.). *Allegro A1384*. Retrieved 7 20, 2012, from Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?wt.z_cid=ref_hearst_0211_buynow&mpart=A1384LLHLT-T&v=620&cur=USD

1360. (n.d.). *Allegro A1360*. Retrieved 7 20, 2012, from Digikey: http://www.digikey.com/scripts/DkSearch/dksus.dll?wt.z_cid=ref_hearst_0211_buynow&mpart=A1384LLHLT-T&v=620&cur=USD

4067. (n.d.). *Texas Instruments CD74HC4067*. Retrieved 7 21, 2012, from Texas Instruments: http://www.ti.com/lit/ds/symlink/cd74hc4067.pdf

406. (n.d.). *Analog Devices ADG406*. Retrieved 7 21, 2012, from Analog Devices: http://www.analog.com/en/switchesmultiplexers/multiplexers-muxes/adg406/products/product.html

426. (n.d.). *Analog Devices ADG426*. Retrieved 7 21, 2012, from Analog Devices: http://www.analog.com/en/switchesmultiplexers/multiplexers-muxes/adg406/products/product.html

506. (n.d.). *Analog Devices ADG506A*. Retrieved 7 21, 2012, from Analog Devices: http://www.analog.com/en/switchesmultiplexers/multiplexers-muxes/adg506a/products/product.html

019. (n.d.). *Seeed Studio Arduino-019 Audio shield* Retrieved 7 24, 2012, from Micro4you: http://www.micro4you.com/store/arduino/arduino-shield/arduino-mp3-shield.html

rMP3. (n.d.). *Rogue Robotics rMP3*. Retrieved 7 24, 2012, from Cutedigi: http://www.cutedigi.com/arduino-shields/mp3-shield-for-arduino.html

10628. (n.d.). *DEV-10628*. Retrieved 7 24, 2012, from Sparkfun: https://www.sparkfun.com/products/10628

[1]Wabbot,"Stepper Motors (2)". [Online]. Available:

http://www.societyofrobots.com/member_tutorials/node/120

[2]"Stepper Motors". [Online]. Available:

http://www.tigoe.com/pcomp/code/circuits/motors/stepper-motors/

[3] Jones W. Douglas, "Control of Stepper Motors a Tutorial".[Online]. Available:

http://homepage.cs.uiowa.edu/~jones/step/index.html

[4]"Motor Control and Drive". [Online]. Available:http://www.microchip.com/pagehandler/en-us/technology/motorcontrol/motortypes/brushed.html

[5]"Stepper Motors". [Online]. Available: http://www.epanorama.net/links/motorcontrol.html#stepper

[6]"Electromagnet". [Online]. Available: http://www.geeplus.biz/FTPROOT/Electromagnets_technical_overview.pdf

[7]Stefan Holodnick, "Playing with my Arduino (Board)". [Online]. Available: http://blog.dailyinvention.com/playing-with-my-arduino-board/

[8]”LCD Display”. [Online]. Available:

http://www.electronics-project-design.com/LCD_Display.html

[9]”Schematics –Robot Power Regulation”. [Online]. Available: http://www.societyofrobots.com/schematics_powerregulation.shtml

[10]”Pedal Powered Prime Mover – DIY Plans”. [Online]. Available:

http://www.los-gatos.ca.us/davidbu/pedgen/plans.html