System Design for Chess Connect

Team #4,
Alexander Van Kralingen
Arshdeep Aujla
Jonathan Cels
Joshua Chapman
Rupinder Nagra

April 5, 2023

1 Revision History

Date	Version	Notes
2023-01-11	Arshdeep Aujla	Introduction, Purpose, User Interface, Other Considered Designs
2023-01-17	Arshdeep Aujla	Added Design of Hardware
2023-01-18	Joshua Chapman	Scope, Project Overview, System Variables, Timeline
2023-01-18	Joshua Chapman	Hardware Design, Communication Protocols, Wiring Diagram
2023-01-18	Rupinder Nagra	Interface
2023-04-05	Arshdeep Aujla	Add answer to reflection question about limitations
2023-04-05	Arshdeep Aujla	Removed the template text in appendix
2023-04-05	Arshdeep Aujla	Removed extra heading and added reference to SRS Abbreviations and Acronyms

2 Reference Material

Please refer to the SRS document for additional Abbreviations and Acronyms.

2.1 Abbreviations and Acronyms

symbol	description
M	Module
MG	Module Guide
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
Chess Connect	Explanation of program name
UC	Unlikely Change

Contents

List of Tables

List of Figures

3   Introduction

This document outlines the system design portion of this project's design documentation. Design documentation is intended to separate the project into modular components to increase the project's understandability and reusability.
Other useful documents for this project are the following:

- SRS

- HA

- VnV

4   Purpose

The purpose of this document is to outline a detailed system design. The system design includes system variables, user interfaces, a timeline for completion, and designs of many different components such as hardware, electrical components, and communication protocols. This document also includes references to an intensive list of the project's mechanical and electrical components and a reflection in the appendix.
Other documents relating to design are the following:

- Software Architecture Document

- Detailed Design Document

5   Scope

The Chess Connect system includes a physical chess board and associated software application to aid in the learning and sharing of in-person chess games. Mechanical design of the chess board and electrical systems are included in the scope. Design of the user interface and communication between the systems is also included. Not included is the design of the chess AI being used as a learning tool. Exisiting chess websites and platforms are also out of the scope of Chess Connect.
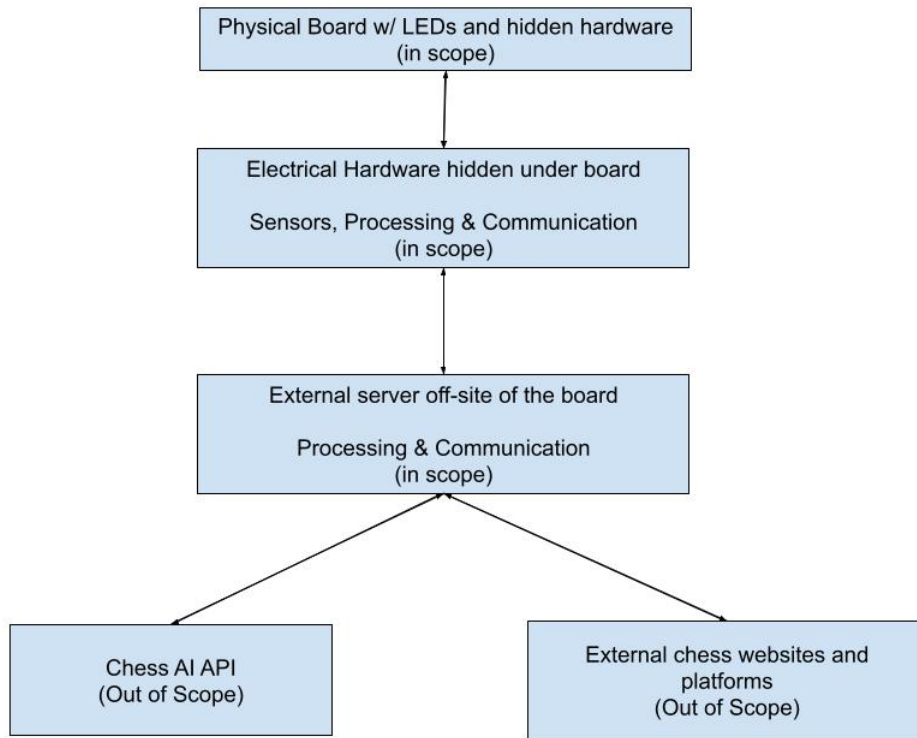
Figure 1: System Scope

6 Project Overview

6.1 Normal Behaviour

The normal operation of the chess board involves configuring game mode and settings, reading the physical positions and identifiers (magnetic strength) of the pieces, lighting up corresponding LEDs and determining legal moves on the micro-controller. There will be three game modes: Normal Mode (no LED feedback), Engine Mode (best moves calculated by a chess engine and displayed by an LCD display) and Beginner Mode (legal moves displayed by LED when a piece is picked up). The micro-controller will also be simultaneously transmitting data to the server via Bluetooth and receiving responses in the form of "best" moves from the server. The server will be calculating these best moves in real time depending on the configuration of the pieces on the board sent from the chess board, and sending it back over Bluetooth every time it is queriedwhile the game is in "Engine Mode". The server will also be communicating all of this information to a web application where users can tune in and watch the pieces and seegame stats in real time.

6.2 Undesired Event Handling

When an unexpected event occurs, chess connect will always return to the most appropriate safe state. When an action during a game, the on-board LCD screen displays the error and provides instructions to correct the mistake and proceed normally. When an error occurs in the communication between sub-systems, all active memory is stored and a default error screen appears on the LCD screen to inform the players.
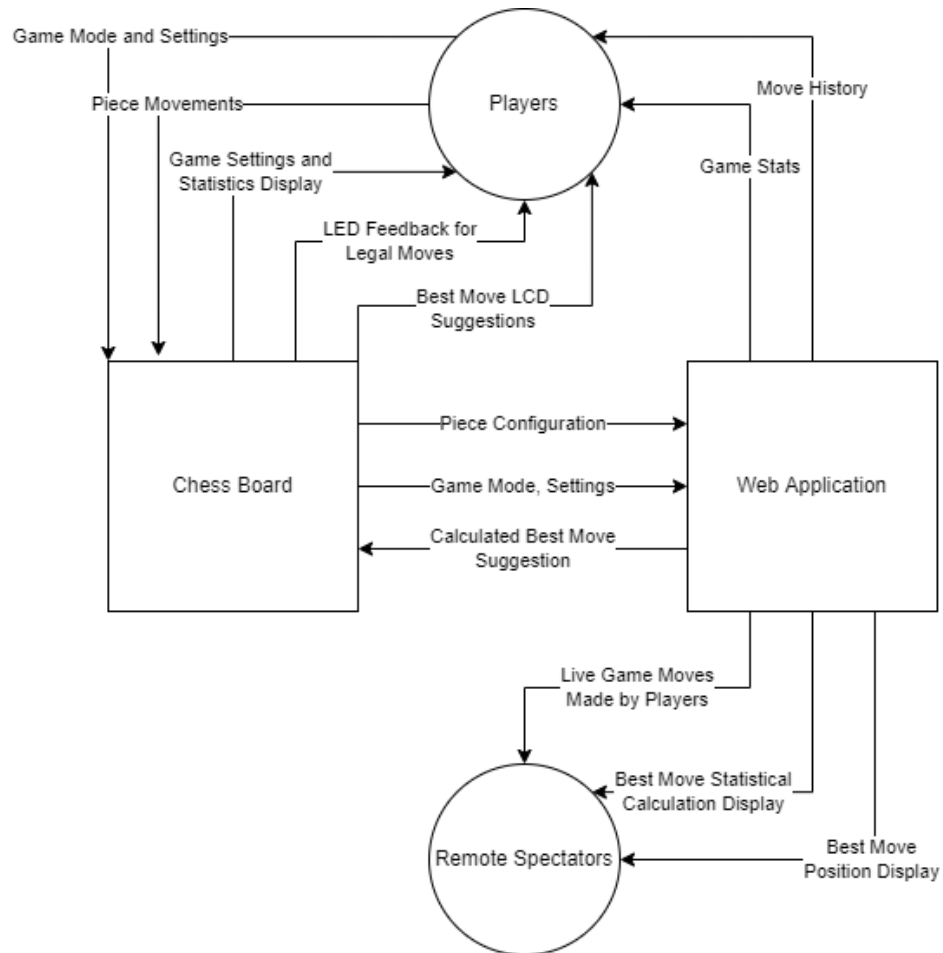
6.3 Component Diagram

Figure 2: Overall System Context

6.4 Connection Between Requirements and Design

Requirements GA1, GA2, GA3, GA4, GI1, GI2, GI3, NB2, NB3, EB2, EB3, EB4, BB3, BB4, BB5 describe a variety of buttons to support game flow and modes. Too many physical but-

tons were required so an LCD screen was placed on the side of the board to accomodate the large number of functions.

Requirements GI5, GI6, ND1, NA1, ED1, ED2, EA1, BD1 dictated decisions for specific information to be communicated from the board to the Web Application. Serial Communication was chosen to display a custom string to communicate the necessary game state, errors and best moves between the board and the application.

Requirements NB1, EB1, BB1 describe the memory performance required of the hardware to store game data of entire games. Design satisfies this requirement with the addition of two processors to share the memory load and processing load.

Requirement BB2 outlined the implementation of LEDs requiring fast response times. Due to communication protocols, screens and sensors are already consuming processor time, a second processor was added with faster processing speed to handle the LEDs and sensors exclusively.

7 System Variables

7.1 Monitored Variables

Variable	Units	Description
s_a{1-8}	Volts	States of tiles a1 - a8 on the board. They are analog signals converted to digital and the state of the tile is determined. The possible states of each tile is empty, black/white pawn, black/white rook, black/white knight, black/white bishop, black/white queen, black/white king.
s_b{1-8}	Volts	States of tiles b1 - b8 on the board. " "
s_c{1-8}	Volts	States of tiles c1 - c8 on the board. " "
s_d{1-8}	Volts	States of tiles d1 - d8 on the board. " "
s_e{1-8}	Volts	States of tiles e1 - e8 on the board. " "
s_f{1-8}	Volts	States of tiles f1 - f8 on the board. " "
s_g{1-8}	Volts	States of tiles g1 - g8 on the board. " "
lcd_x_pos	Volts	Positive X coordinate of the resistive touch screen.
lcd_x_neg	Volts	Negative X coordinate of the resistive touch screen.
lcd_y_pos	Volts	Positive Y coordinate of the resistive touch screen.
lcd_y_neg	Volts	Negative Y coordinate of the resistive touch screen.
bluetooth_rx	Volts	Serial communication receive channel via bluetooth to the web application. Best moves from the chess engine are sent to the chess board via this channel.

7.2 Controlled Variables

Variable	Units	Description
hall_power{1-8}	Volts	Eight sources of power to the eight columns of sensors under the board. The power will be pulse width modulated to multiplex the signals of the hall sensors.
LED_row_pos{1-9}	Volts	Nine power supplies are delivered to the nine rows of LEDs. When a corresponding ground is supplied to a column, an individual LED can be lit.
LED_col_gnd{1-9}	Volts	Nine ground supplies are delivered to the nine columns of LEDs. When a corresponding power is supplied to a row, an individual LED can be lit.
LCD_Display{0-7}	Volts	Eight bits of information are sent to the LCD screen to display a variety of images.
bluetooth_tx	Volts	Serial communication transmit channel via bluetooth to the web application. The current game state and errors are sent via this channel.

7.3 Constants Variables

Constant	Unit	Value
Chess board width	inches	12
Chess board length	inches	12
Chess board tile width	inches	1.5
Chess board tile length	inches	1.5
Supply Power to Board	Volts	110 VAC

8 User Interfaces

8.1 Hardware Interface

The user will interact with two main components of the hardware.

- Magnetic chess pieces

- Physical chess board containing sensors

They will interact with the chess pieces and chess board as they would in a normal chess game. The chess board reflects a standard chess board, with LEDs in the center of each square. They would move the chess pieces on the board and remove them in according to the rules of chess. If the device is set to beginner mode, the LEDs will light up in according to which available moves are available for that chess piece. They would interact with the LEDs by using them as a guide for potential moves to make.

8.2   Software Interface

The users will interact with the software component of this device through a web application. They would need a device with an internet connection and and internet browser to view the application. The user will interact with this interface by visual viewing the chess board status in real time including a visualization of the chess piece locations. They will also be able to turn on and off beginner mode in this interface through clicking an interactive button in the web application.

9   Design of Hardware

To build the custom chess board for Chess Connect we designed a modified chess board to fulfill the requirements of the system. The chess board has visible LEDs from below the board that can be seen in the figure below.
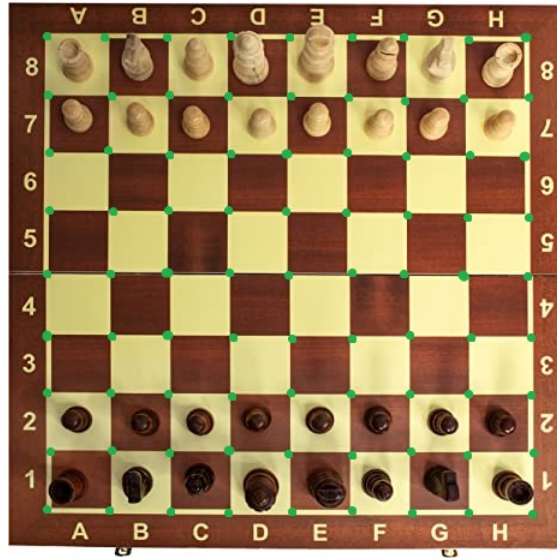

Figure 3: Hardware Design

The chess board will be hollow and contain enough space to contain the electronics. An AC power cable will protrude from the side of the board to connect to an outlet nearby.

9.1 Components Acquired

Please refer to Appendix B for a detailed list of acquired mechanical components.

10 Design of Electrical Components

10.1 Components Acquired

Please refer to Appendix C for a detailed list of acquired electrical components.
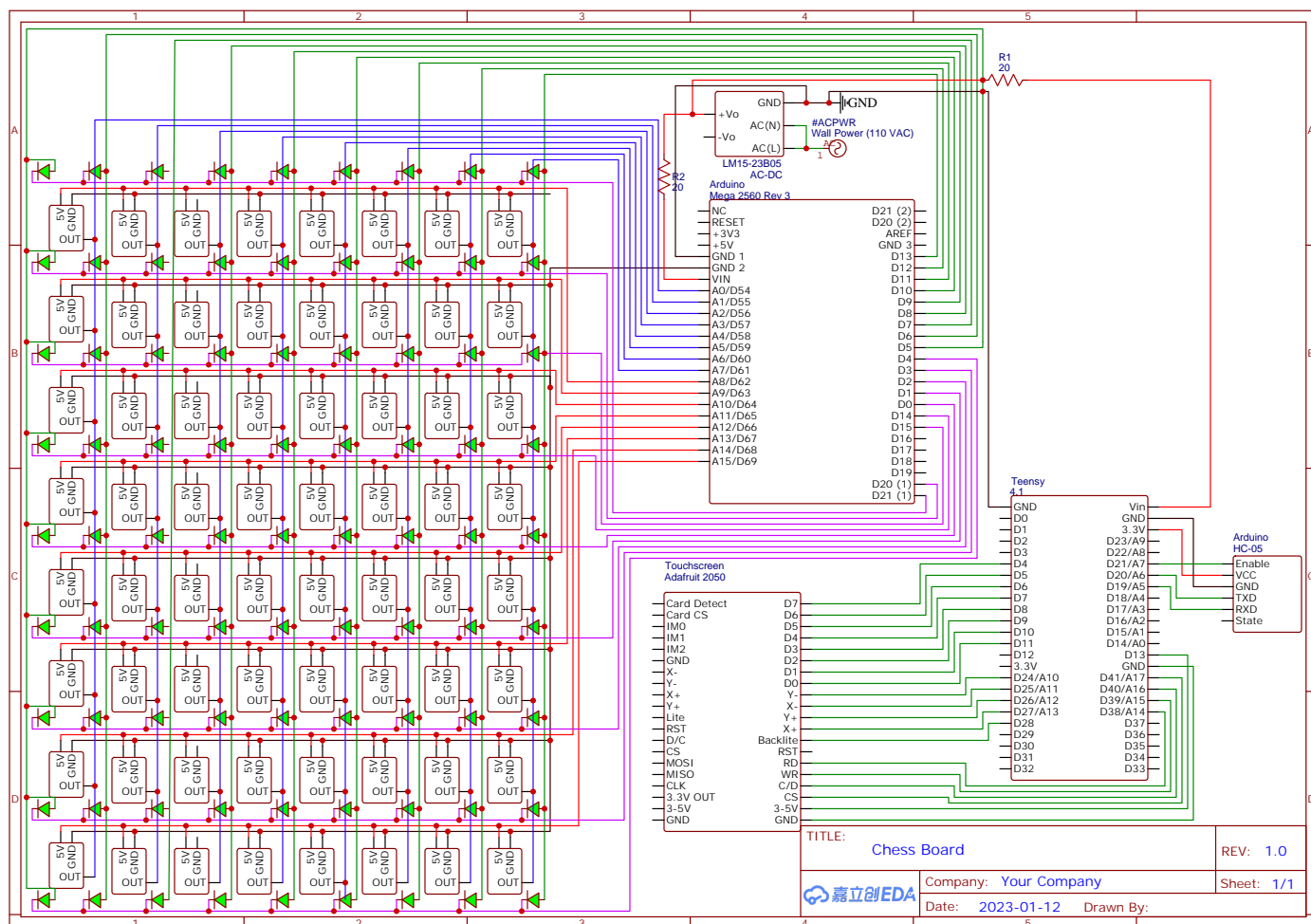
10.2 Wiring Diagram

Figure 4: Electrical Wiring Diagram

10.3 LED and Piece Interaction

Each square on the chessboard will contain a Hall sensor and a green LED. The LEDs are to indicate where a piece can possibly go, depending on which mode the user is in. For example, if the user is in a beginner's mode and picks up the White King, then the LEDs surrounding all the squares where the King is picked up from will light up telling the user that the King can be placed on any of those squares. The logic for which LEDs light up depend on a few things:

- The type of piece. Different LEDs would light up for King, Queen etc.

- If any of the user's own color piece is placed on a square, that LED would not light up as you cannot place two pieces on the same square. However, if an opponent's piece is placed on a square, that LED may light up provided it is a valid move depending on the type of piece the user is making the move with.

- Any square further away from a square where user's own piece is present, will not light up as the pieces cannot jump a piece unless it is a Knight.

Let us see the working of the LEDs considering the example on a 5x5 grid as shown in the figure below. '1-5' are the rows and 'A-E' are the columns. 'a-e' are the twenty-five different LEDs corresponding to each square. The green background squares will indicate the LED to be ON for that square in the grid.

Figure 5: 5x5 Grid

9

- Case 1: King present in the middle of a 5x5 grid chessboard. The King can move only one square in any direction – up, down, to the sides, and diagonally.

Figure 6: Potential moves for the King

The LEDs g,h,i,l,n,q,r,s will light up for the potential moves provided those squares are empty or an opponent's piece is placed on those squares.

- Case 2: Queen present in the middle of a 5x5 grid chessboard. The Queen can move multiple squares in any one direction - forward, backward, sideways and diagonally as far as possible as long as she does not move through any of her own pieces.

Figure 7: Potential moves for the Queen

The LEDs a,c,e,g,h,i,k,l,n,o,q,r,s,u,w,y will light up for the potential moves provided those squares are empty or an opponent's piece is placed on those squares.

- Case 3: Bishop present in the middle of a 5x5 grid chessboard. The Bishop can move multiple squares but only diagonally. Each bishop starts on one color (light or dark) and must always stay on that color.

Figure 8: Potential moves for the Bishop

The LEDs a,e,g,i,q,s,u,y will light up for the potential moves provided those squares are empty or an opponent's piece is placed on those squares.

- Case 4: Knight present in the middle of a 5x5 grid chessboard. The Knight can move two squares in one direction, and then one more move at a 90-degree angle in either direction, just like the shape of an "L".

Figure 9: Potential moves for the Knight

The LEDs b,d,f,j,p,t,v,x will light up for the potential moves provided those squares are empty or an opponent's piece is placed on those squares. Knight is the only chess piece that can skip over a piece and be placed even if it is immediately surrounded by a user's own piece. For example, if there is user's own pieces present at the squares B2, B3 and B4, the user can still place the Knight at A2, A4, B1 or B6 by skipping over the other pieces.

- Case 5: Rook present in the middle of a 5x5 grid chessboard. The Rook can move multiple squares but only forward, backward or to the sides.

Figure 10: Potential moves for the Rook

The LEDs c,h,k,l,n,o,r,w will light up for the potential moves provided those squares are empty or an opponent's piece is placed on those squares.

- Case 6: Pawn present in the middle of a 5x5 grid chessboard. Pawns are unusual because they move and capture in different ways: they move forward but capture diagonally. Pawns can only move forward one square at a time, except for their very first move where they can move forward two squares.

Figure 11: Potential moves for the Pawn

The LEDs c,g,h,i will light up for the potential moves provided those squares are empty

or an opponent's piece is placed on those squares. For pawn, the LED 'c' only lights up if it is the first time a pawn is making a move. After a pawn has been moved once, it can never move two squares at once. Also, a pawn can move diagonally, only if there is an opponent's piece is present on those squares. Otherwise, it can only move in the forward direction. For example, the LED 'g' or 'i' will only light up if there is an opponent's piece present on them.

Truth Table for LEDs

Based on the assumptions above, the Truth Table for the LEDs 'a-y' would as shown in the figure below. The digit '0' means LED is OFF and '1' means LED is ON.

PIECE TYPE	CASE #	LEDs ON/OFF																								
		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
King	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0
Queen	2	1	0	1	0	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	0	1	0	1	0	1
Bishop	3	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
Knight	4	0	1	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0
Rook	5	0	0	1	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0	0	0	1	0	0
Pawn	6	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12: Truth Table

11 Design of Communication Protocols

11.1 Arduino Mega to Teensy

The two processors communicate to share game state, game mode and move information for smooth performance. Arduino Mega collects data and forms the current game state from the sensors given. Once the current game state is formed it is sent via serial communication to the Teensy processor. The string is in the form of FEN@Game Mode@Resign.

11.2 Web Application to Teensy

The processor and the web application communicate to share current game state, game mode and best moves from the engine. The Teensy sends a string in from of FEN@Game Mode to the web application to perform best move calculations. Once the web application finds the best moves, it will send a string in the form of bestMove1@bestMove2@bestMove3 back to the Teensy.

12 Timeline

Task	Member	Date
Learning the library of the LCD screen	Josh	Jan 23
Learning the library of the bluetooth module for web app communication	Jonathan	Jan 23
Installing the hall sensors and LEDs to the bottom of the chess board	Arshdeep	Jan 23
Complete design and implementation of web application	Rupinder	Jan 30
Solder sensors and LEDs to the appropriate Arduino	Arsheep	Jan 30
Configure power supply for arduinos and confirm safety of the system from wall power	Josh	Jan 30
Implementing a chess library into the arduino Mega	Alex	Jan 30
Implement communication protocols for bluetooth and serial communication onto the processors	Jonathan	Jan 30
Complete communication protocol packet from web app to arduino	Rupinder	Feb 7
Run tests with the completed piece tracking code with completed board	Alex	Feb 7
Revision 0 Demonstration. Completed by this time will be piece recognition, LED response, communication between processors, bluetooth communication, web application and LCD screen design	Josh, Arsheep, Alex, Rupinder, Jonathan	Feb 13

A   Interface

An online chess game typically has a user interface that includes a chess board display, where the player can see the current state of the game and make moves. Our web application will only serve as an additional display for the moves being played on the physical board, simply mirroring the moves with minimal latency over Bluetooth. The different modes on the physical board will also be compatible with the moves displayed on the web application. A web application interface aims to be more intuitive for players that are accustomed to online chess and might also allow for a more immersive viewing experience for spectators.

B   Mechanical Hardware

- 20" Chess Board
- Self Contained Box to hide hardware

C   Electrical Components

- Arduino Mega 2560
- Teensy 4.1
- 3.5" Touchscreen Display
- HC-05 Bluetooth Module
- AC-DC converter for
- 81 green LEDs
- 81 1000ohm resistors
- 64 HALL sensors
- 64 1mF capacitors
- 30 ft, 20 AWG wire

D   Reflection

Project Limitations

There are a few limitations in our design. The first one is that the device is not wireless. It requires a wired power supply at all times. This could be improved by adding a wireless

power supply to the device. Then it would only need to be charged every so often, rather than requiring a constant supply of power. Another limitation is that the device always needs to be connected to the Internet to upload the real-time game data. With unlimited resources, we could put an LTE chip in our device to ensure there is always a connection to the Internet. Although this is not feasible for the scope of the project. Lastly, a limitation is the physical size and portability of the device. The final product is expected to be larger than a chess board, as well as approximately 15cm thick. With more money and time, the size could be reduced to improve the portability.

Other Considered Designs

One problem that we had to overcome in our design is that there are not enough input and output pins in one microcontroller for all of the components. One solution we considered was having multiple microcontrollers for this project to ensure there is one input pin for each input sensor and one output pin for each output. This design would be beneficial in the way of simplicity of code. The tradeoff would be the complexity in coordinating the communication between multiple microcontrollers and the web application. A second option to solve this problem is to use multiplexing to reduce the number of input and output pins needed for a large number of components. This option requires more complex code, but only requires one microcontroller device. We chose to implement the multiplexing option because it only used one device which saves money, as well as eliminates the need to coordinate between multiple microcontrollers.