February 21st, 2020
Professor Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, BC V5A 1S6

RE: ENSC 405W/440 Design Specification for IntelliChess

Dear Professor Scratchley,

The following document contains the design specifications for our product IntelliChess by Company 10, as part of the course ENSC 405W. At Company 10, our goal is to make learning chess a more fun and interactive experience, while maintaining the originality and simplicity of normal chessboards. IntelliChess is a smart chessboard that aims to bring the computer opponent from your screen to a physical chessboard.

Whether you are a chess Grandmaster or just starting out, IntelliChess has something for everyone. The board offers an invisible opponent that moves its own pieces while also providing hints during the match and feedback after the match through the included mobile app.

This document will outline the different design specifications our product will follow throughout the various phases of this project (proof of concept, engineering prototype, final product), as well as the reasoning as to why these design decisions were made. Multiple appendices will be included, containing User Interface and Appearance Design choices, Supporting Test Plans and Supporting Design Options.

Our company consists of 5 passionate engineering students coming from diverse backgrounds, united under the goal to demonstrate and apply the knowledge acquired from SFU throughout the years. We would like to thank the TAs Shervin, Mike and Chris for their ongoing help and guidance throughout the project. And thank you for taking the time and effort to review our requirement specification document. If you have any questions, please contact me at cwliang@sfu.ca

Sincerely,

Marco Liang
CEO, Company 10

IntelliChess

Design Specification Document

Company 10 Members:
Marco Liang
Malcolm Mckean
Husam Mohammad
Taha Ahmed
Eric Park

Document ID: C10-IDS

Revision Num	Reason For Revision	Date
1.0	Initial Creation of Document	3/20/2021

Abstract

This document contains the design specifications for the Intellichess Smart Chess Board. The system consists of a mobile application used for game analysis and emulation, and the board itself which utilizes a microcontroller to incorporate the piece movement sub-system and the piece detection sub-system. Multiple designs were considered and simulated, and the optimal solution for each component was determined after careful review. The movement sub-system was identified as the most problematic and most likely to encounter failure as it was an implementation of new technology. Appropriate test plans were developed to thoroughly test if the system met all the requirement (and ultimately the design) specifications. The final system consists of a mobile application which handles the chess engine logic, allowing us to operate the movement and piece detection system using an Arduino Mega. The piece detection itself is primarily done using anti-metal RFID tags whereas the piece movement utilizes a grid of solenoids that move a loaded magnet. The document lists the specifications, approach and results for all three phases.

Table of Contents

List of Figures

List of Tables

Glossary

Term	Definition
Rank	A term used for the 8 rows on the chess board, running from 1-8.
File	A term used for the 8 columns on the chess board, running from a-h
BLE	Bluetooth low energy
UART	Universal asynchronous receiver-transmitter
SPI	Serial Peripheral Interface

TTL	Time to live
IoT	Internet of Things
GUI	Graphical User Interface
UHWM	Ultra High Molecular Weight - a subset of the thermoplastic polyethylene.
FEN	Forsyth-Edwards Notation - a standard notation for describing chess board position
UCI	Universal Chess Interface - communication protocol for chess engines
POC	Proof of concept
GPIO	General Purpose Input Output
GUI	Graphical User Interface
UI	User Interface

1 Introduction

Easy to learn, yet almost impossible to master, the game of Chess is a timeless masterpiece. With the relatively recent ability to play the game online, the popularity of the game has seen explosive growth[1]. However, most players have been playing virtually as virtual chess has a lot more assistance features and options to customize your game. Recognizing the likely demand for a physical version of the same feature packed system, our company decided to take on the challenge of designing a smart board. The board will incorporate all the available virtual chess features allowing users to enjoy the tactile aspect of the board game without any compromise on functionality.

The smart board consists of many intricate and complicated design components. However, the movement system in particular is challenging to implement. We could not find any reference for using solenoids to drive a loaded magnet. There is a possibility that the movement system might encounter failure while trying to overcome friction when dealing with the full load. The team is also paying special attention to the electromagnet choice as there can be potential issues with strength and piece attraction. The main obstacle is trying to find the balance between strength and the weight of the electromagnet (and the friction it generates on the pieces).

In this document, we will be going over the complete design specification for our product IntelliChess. We will discuss our design considerations and finalized choices with their justifications, and provide insight on the design challenges we faced and how they were addressed.

Design Classification:

Each Design specification has an associated stage of development, indicated in the Phase column of the design tables. The three phases are:

● **Alpha (A)**, design specifications aimed to be completed as soon as possible for our proof-of-concept prototype, which will be demonstrated at the end of ENSC 405W

● **Beta (B)**, design specifications that will be satisfied in the engineering prototype for demonstration at the end of ENSC 440

● **Final Product (FP)**, design specifications that will be satisfied upon delivery of the final product.

2 System Overview

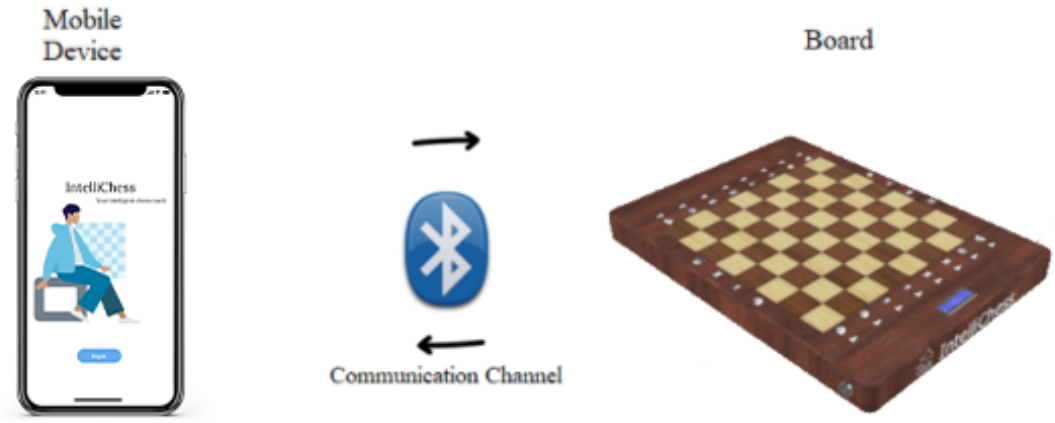


Figure 2.0.1: High Level System Overview

At a high level, our system is made up of two components, a mobile application and the Smart Board, with the two communicating over BLE (Bluetooth Low Energy). The board consists of our unique piece movement and detection systems which are discussed in greater detail later in the document. The mobile application interfaces with a chess engine and sends the engines move data to the board. The mobile application also provides helpful functionalities to the user such as the ability to request hints. The mobile application is explained in greater detail later in the document.

3 Mobile Application

The IntelliChess companion app will serve as the central point for our gameplay. It will host the chess engine that processes and determines the best moves for the computer. The user will be able to set up their games from the app, and also get a live view of the chess board as the game goes on. The board itself will send the piece locations, and the app will use the chess engine to generate the next move, and send that as a command back to the board, so the board will “play” against the user. In the figure below, the basic gameplay communication flow between the app and board is shown.

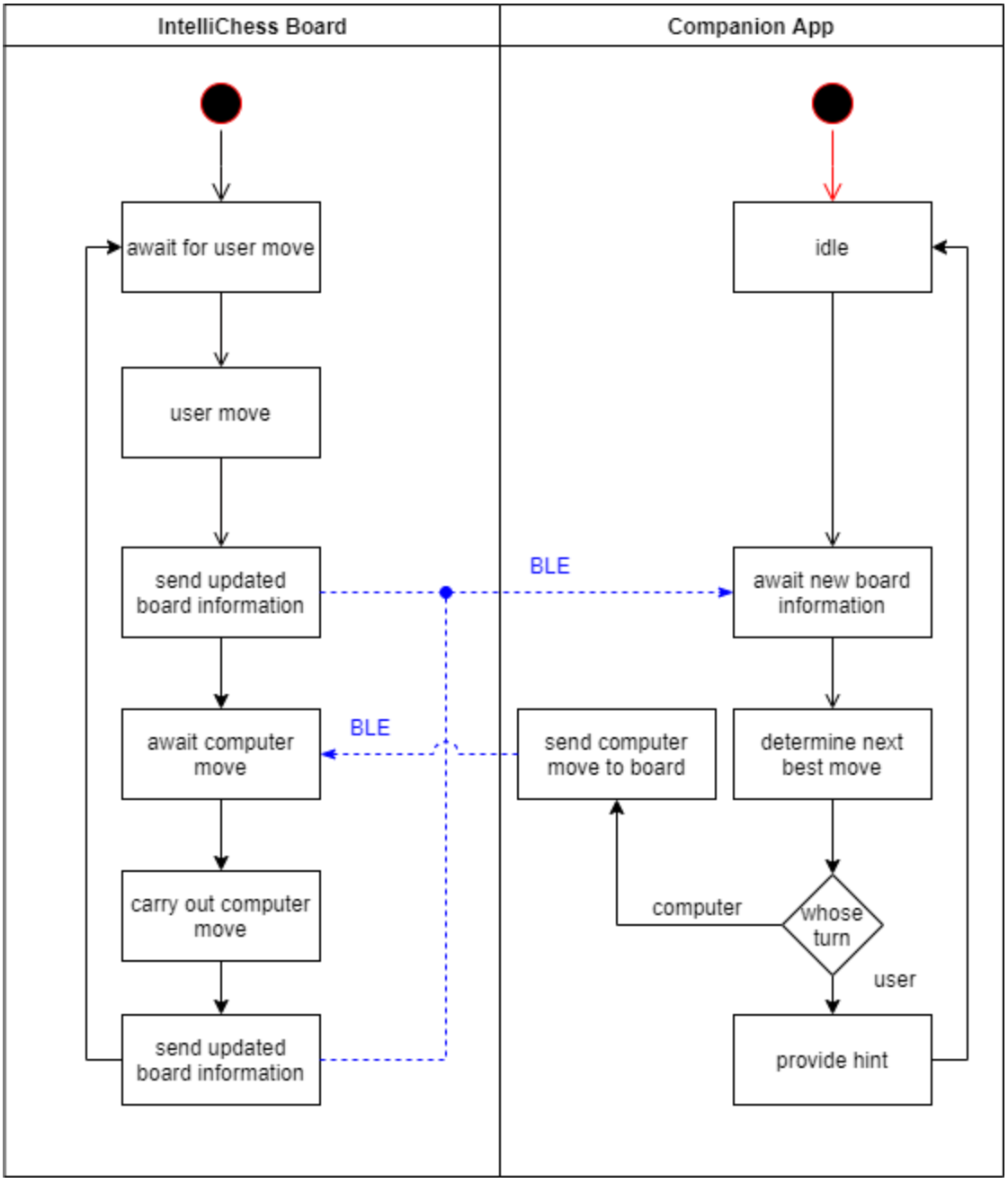


Figure 3.0.1: Communication Flowchart for Companion App

The mobile application will be built using Facebook's React Native framework, which is capable of building and compiling both Android and iOS applications using just one set of source code. The Android application is built for SDK 29 (Android 10), following Google Play's target API level requirements [2], but allows it to run on minimum SDK 18 (Android 4.3), due to the minimum SDK required for BLE support. Based on the App Store Guidelines [3] set out by Apple, applications must run on the currently shipping OS, which would be iOS 14.1, our target deployment level, though lower levels of iOS are supported.

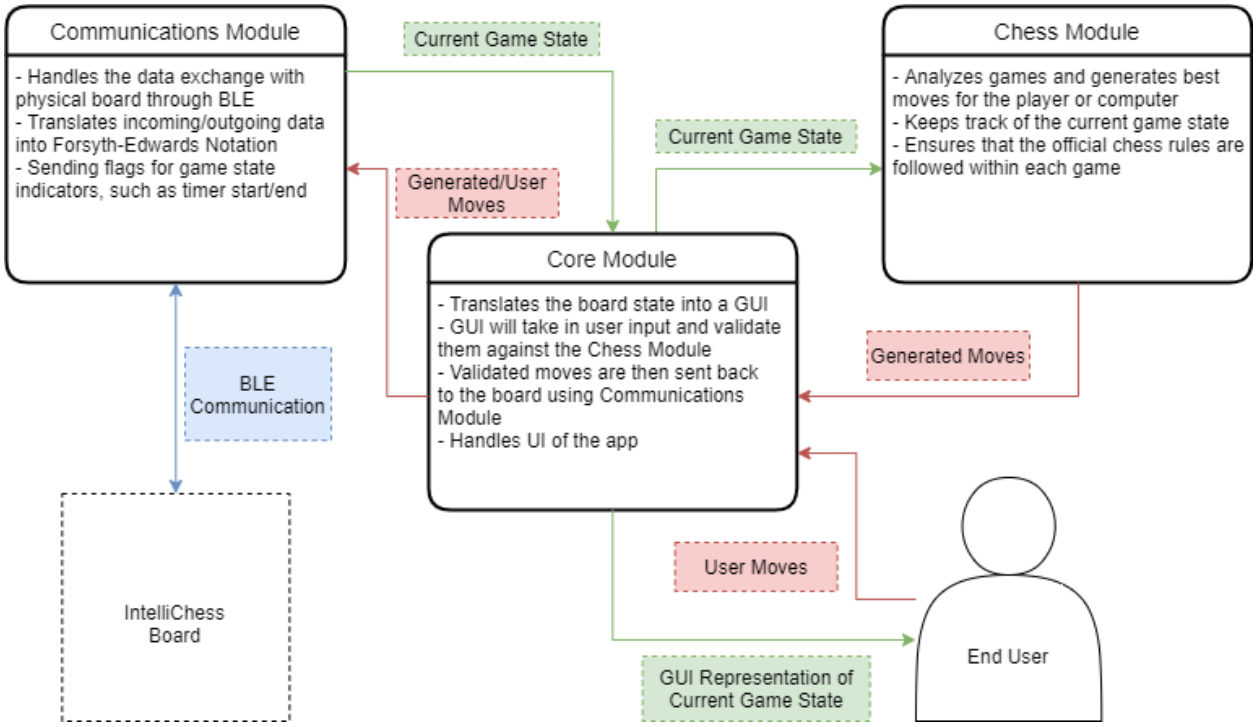


Figure 3.0.2: Module Breakdown of Companion App for Beta Prototype

As seen in *Figure 3.0.2*, for the Beta Prototype, IntelliChess App will be split into three main components:

- Communications Module
- Chess Module
- Core Module

Each module will be introduced in detail below. In the Final Product, more modules will be added to enable online multiplayer.

3.1 Communications Module

The Communication Module handles the information exchange between the IntelliChess board and its companion app. This module will be built on top of the **react-native-ble-plx** library [4], provided by Polidea. This library provides the necessary tools to scan, connect, and send/receive data from a BLE peripheral. A large role of the Communications Module is sending the chosen computer moves to the board, and receiving the current board information after the user makes a move, as well as other smaller indicators such as game start/end, time outs, etc. In order to communicate the information efficiently, another function of the Communications Module would be to translate all incoming and outcoming board data into the proper FEN (Forsyth-Edwards Notation) form, which would enable the Core and Chess Module to read and process the current game state. Since both iOS and Android require explicit user permission before allowing the app

to use Bluetooth, the Communications Module will also be responsible for getting that permission via an alert before establishing the connection.

The data transfer will be done following the GATT(Generic Attribute Profile) protocol, which defines a client-server relationship between the BLE peripheral (GATT server), and the BLE device (GATT client). In the context of IntelliChess, the peripheral would be the board itself, or more specifically, the HM-18 Bluetooth 5.0 Module integrated into the onboard microprocessor, and the BLE device would be the mobile device the companion app is running on.

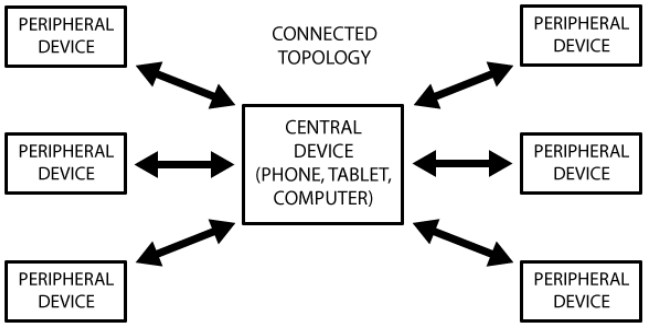


Figure 3.1.1: GATT Server/Client Relationship[5]

For the Alpha phase of development, which should build up to the Proof-of-concept demo at the end of ENSC 405W, the Communication Module will be complete in terms of sending and receiving board states, which would report the states for the six squares during the demo, as well as sending new commands to control the pieces using the electromagnet movement system. For the Beta phase, the capability to encode and decode data into FEN form will be added to the Communications Module, as well as the ability to send/receive status indicators, such as game start/end, timer start/end, invalid move, etc. For the Final Product, no additional capabilities will be needed for the Communications Module.

Design ID - Phase	**Description**	**Corresponding requirement & design specification**
Des 3.1.1-A	The Communications Module will be capable of sending/receiving data through BLE	**RS-2.1.08 to RS-2.1.12 RS-2.1.16 to RS-2.1.17, RS-2.2.11, Des 3.1.1, Des 3.1.3, Des 3.2.1**
Des 3.1.2-A	The Communications Module will ask the user for permission before connecting	**NA**

Table 3.1.1: Communications Module Design Specifications for PoC Prototype

Design ID - Phase	Description	Corresponding requirement & design specification
Des 3.1.3-B	The Communications Module will be capable of encoding/decoding data into FEN form	**RS-2.1.08 to RS-2.1.12 RS-2.1.16 to RS-2.1.17 RS-2.2.11, Des 3.1.1, Des 3.1.3, Des 3.2.1**
Des 3.1.4-B	The Communications Module will be capable of sending/receiving game status indicators	**RS-2.1.04 to RS-2.1.06 RS-2.1.18 to RS-2.1.19 Des-3.2.10**

Table 3.1.2: Communications Module Design Specifications for Beta Prototype

3.2 Core Module

The Core Module will be responsible for handling the incoming data from the Communications module, relaying that information to the Chess Module to process and determine the next best move, and relaying that back to the Communications Module to send back to the physical chess board. It will also be responsible for updating the graphical representation on the screen for the user to track the current game state. This means that the Core Module must be capable of deciphering the board state data in FEN form, and printing it out on the Graphical User Interface (GUI). Due to the fact that the IntelliChess app will not include many UI elements, the graphical interface for the chess board will also be included within the Core Module, as well as the other UI elements within the app, rather than splitting those into a dedicated module.

There are some foreseeable possible difficulties in implementing the chess board GUI, since the engineering team does not have a lot of mobile game development experience. However, there are many open-source libraries available for JavaScript, popular libraries such as **chess.js** will be very helpful in speeding up the development time. There are also examples of chess implementations for React Native available on the following repositories:

- https://github.com/halilb/react-native-chess - uses Lichess as its backend
- https://github.com/venil7/chess-react-native - uses creator's own chess engine
- https://github.com/Hackstr/react-native-chess - a small example of a chess GUI

All three of these made their full source code available to the public, however, only the first one specified its usage license as MIT [X] - fully open-source, giving the permission to use, modify, publish without restrictions. So it has been chosen to be the foundation for the Core Module to be built off of, with the other two repositories kept reference purposes only, citing any usage when necessary, but NO code will be copied.

For the Alpha phase of development, the Core Module will be handling the GUI display for the six-squared PoC prototype, which means it must be able to read in the data that the Communications Module will provide, though the relay to the Chess Module will not be implemented in this stage. The GUI display for the Alpha phase will also allow the user to modify the piece positions on the phone, and relaying that information back to the Communications Module in order to demonstrate the electromagnet movement system. These capabilities will be further expanded in the Beta phase of development, to take in the board information for the full sixty-four chess board, and to communicate with the Chess Module to determine the next best move, to validate user moves, to check game end, to set the game parameters, as well as sending the computer's moves back to the physical board. Other than handling the backend data exchange, the Core Module will also handle the connection screen and game mode selection screen on the front end. In the Final Product, the Core Module will be further worked on to include options for multiplayer gameplay, connecting to an account management and authentication system (Accounts Module - not implemented in PoC or Beta phase), and the UI elements will be separated into their own module (UI Module - also not implemented in PoC or beta phase).

Design ID - Phase	**Description**	**Corresponding requirement & design specification**
Des 3.2.1-A	The Core Module will be able to render a GUI for the PoC six-squared prototype that displays what piece is on which square on the physical board	**RS-2.2.02**
Des 3.2.2-A	The Core Module will be able to read in the data that the Communications Module has received from physical board	**RS-2.2.02, Des 3.1.1**
Des 3.2.3-A	The Core Module will be able to take in user inputs in moving the pieces on the GUI, and send that information to the physical board	**Des 3.1.1**
Des 3.2.4-A	The Core Module will be able to handle the UI flow of the companion app, taking users from connection screen to game screen	**NA**

Table 3.2.1: Core Module Specifications for PoC Prototype

Design ID - Phase	Description	Corresponding requirement & design specification
Des 3.2.5-B	The Core Module will be able to translate a FEN string onto the chess GUI	**RS-2.2.02, Des 3.1.1, Des 3.1.3, Des 3.2.1**
Des 3.2.6-B	The Core Module will be able to send the board state data to the Chess Module and request for a best next move (for hint or CPU move)	**RS-2.1.12, Des-3.2.3**
Des 3.2.7-B	The Core Module will be able to send the board state to the Chess Module and request for validation of the user's move, or for game end, or all possible moves for the piece selected	**RS-2.1.05 to RS.2.1.13 RS-2.2.09 to RS-2.2.11**
Des 3.2.8-B	The Core Module will be able to display hints and possible moves for the current user on the chess GUI	**RS.2.1.12 to RS-2.1.13 RS-2.2.09 to RS-2.2.11**
Des 3.2.9-B	The Core Module will be able to display an error message on the GUI when an illegal move is played by the user, as well as a suggestion	**RS-2.1.08 to RS-2.1.11 RS-2.2.09 to RS-2.2.11**
Des 3.2.10-B	The Core Module will be able to take in user inputs such as end turn, resign, start/stop timer, and relay that information to the Communications Module	**RS-2.1.04 to RS-2.1.06 RS-2.1.18 to RS-2.1.19 Des 3.1.4**
Des 3.2.11-B	The Core Module will allow the user to setup game parameters such as timing, sides, difficulty, and relay those to the Chess Module	**RS-2.1.02-RS-2.1. 04**

Table 3.2.2: Core Module Specifications for Beta Prototype

3.3 Chess Module

The Chess Module will be responsible for handling the processing and analyzing of each chess game, as well as generating the moves for the computer's moves, and hints for the player. Other functionalities of the Chess Module will be to keep track of the game stages, whose turn it is,

checking whether the game has ended, generating the possible moves for each piece, and ensuring that each move made within the games will abide by the official chess rule book governed by the International Chess Federation.

Given the complexity of building a chess engine from scratch, IntelliChess will instead incorporate an open-source chess engine, Stockfish. The reasons that Stockfish has been chosen are:

- It is open-source, and is available to use for free
- It's the second strongest chess engine in the world, behind only Google's AlphaZero, which is an AI powered by a super-computer [6] [7]
- It offers eight different levels of difficulty to choose from
- It's fully compatible with the Universal Chess Interface protocol, which makes it compatible with a wide range of existing chess GUIs
- It's been confirmed working on mobile devices, as it is available on other mobile apps already
- It will be able to analyze games and provide hints/feedback for players
- It's the most popular chess engine, with a large amount of community support

Stockfish will provide the game analysis and move generation capabilities that will be needed for the game play aspect of the IntelliChess app, and the Chess Module will be built around it to provide the necessary interface for the Core Module to generate moves. This means the Chess Module will need to adhere to the UCI protol, and given that the board states are communicated using FEN strings in Stockfish, the Chess Module must be made to be fully compatible with them as well. Essentially, the Chess Module will be a wrapper that enables the Stockfish engine to be used within the IntelliChess app, and the engine itself has already been ported over to work with React Native with the **Stockfish.js** library [8]. The main challenges that will be faced during development will include incorporating the Stockfish engine in a background thread, since the game engine is required to be running at all times in order to process moves on the fly, and the UI thread, controlled by the Core Module, cannot be interrupted. Fortunately, there is a library, **react-native-workers**, that provides background workers for the React Native framework [9].

Development on the Chess Module is reserved until the Beta phase of development, as the PoC prototype will not be able to demonstrate an entire chess game, instead only showing the most fundamental pieces working together. Therefore, all the capabilities of the Chess Module will be specified for Beta phase only. There are also no foreseeable changes needed for the Chess Module to work on the Final Product, as its functionality will remain the same.

Design ID - Phase	Description	Corresponding requirement & design specification
Des 3.3.1-B	The Chess Module will be able to take in board data in FEN form	**Des 3.1.1, Des 3.1.3**
Des 3.3.2-B	The Chess Module will be able to analyze games in real time	**RS-2.1.13**
Des 3.3.3-B	The Chess Module will be able to generate moves that follow the official rules of chess as set out by the International Chess Federation	**RS-2.1.01, RS-2.1.07 to RS-2.1.12, RS-2.1.17**
Des 3.3.4-B	The Chess Module will be able to check if a user's move was legal	**RS-2.1.08-RS.2.1.11, RS-2.2.09**
Des 3.3.5-B	The Chess Module will be able to generate the potential moves for a piece	**RS-2.1.09**
Des 3.3.6-B	The Chess Module will be able to determine when the game has ended and who the winner is	**RS-2.1.04 to RS-2.1.06**
Des 3.3.7-B	The Chess Module will be able to report the current state of the game at any point	**RS-2.2.02**
Des 3.3.8-B	The Chess Module will run on a separate thread from the main thread, and will not interrupt the app UI when it's processing	**NA**
Des 3.3.9-B	The Chess Module will take in game settings from the Core Module and modify itself accordingly	**RS-2.1.02 to RS-2.1.04**
Des 3.3.10-B	The Chess Module can take in user inputs, such as to resign, stop/start a timing clock, pawn promotion...	**RS-2.1.15, RS-2.1.18 to RS-2.1.19**

Table 3.3.1: Chess Module Specifications for Beta Prototype

4 Hardware Design

4.1 Microcontroller: Arduino Mega 2560

The Arduino Mega 2560 will be the primary controller for the system, responsible for collecting and transmitting piece location information, as well as triggering coils for piece movement. The Mega is capable of both analog and digital transmission and has a built in oscillator for time-sensitive operations. With enough pins to support all our required components, the Arduino

Mega was an easy first choice as it has extensive support and documentation online, as well as a dedicated IDE which makes implementation of communication protocols considerably easier.

The following design specifications will be used for all phases:

Design ID - Phase	Description	Corresponding requirement & design specification
Des 4.1.1-A	The microcontroller provides built in support for SPI and UART communication protocols	**Des 4.2.2** **Des 4.2.3**
Des 4.1.2-A	The microcontroller will have 4KB of non-volatile storage memory	**RS-2.2.02-A** **RS-2.2.09-B**

Table 4.1.1: Microcontroller Specifications for PoC Prototype

Design ID - Phase	Description	Corresponding requirement & design specification
Des 4.1.3-B	The microcontroller has a built in crystal oscillator	**RS-2.1.19-B** **RS-2.1.04-B**
Des 4.1.4-B	The microcontroller will be able to supply 5V	**RS-2.1.16-B** **RS-2.2.03-A**

Table 4.1.2: Microcontroller Specifications for Beta Prototype

Figure 4.1.1: Arduino Mega 2560 [10]

Chip: ATmega2560
Clock Speed: 16 MHz
Supply voltage: 5V
EEPROM: 4 KB
Digital I/O pins: 54
Analog I/O pins: 16

Table 4.1.3: Arduino Mega Specifications

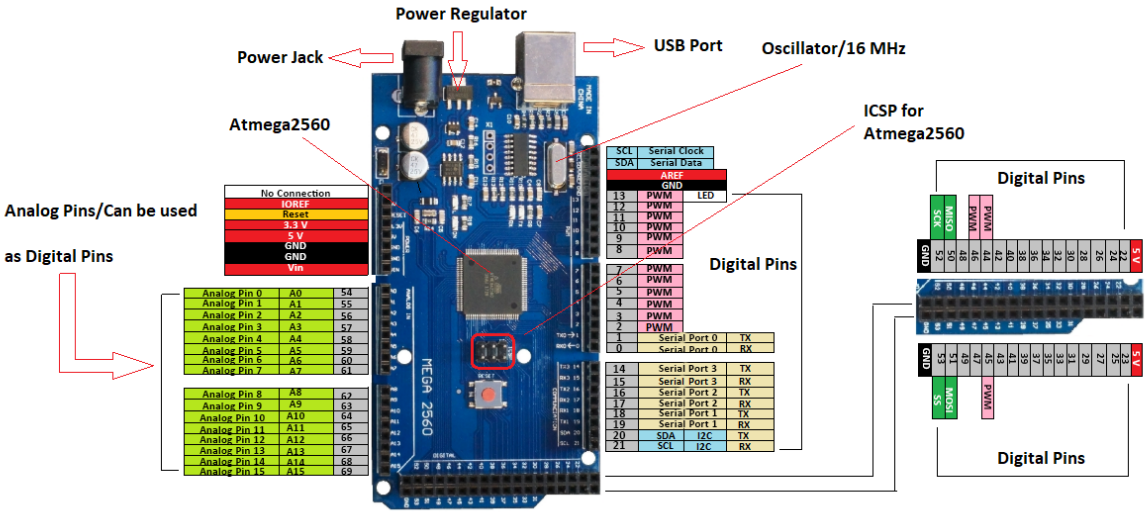


Figure 4.1.2: Arduino Mega pin layout [11]

The Arduino Mega provides a lot of digital and analog pins for multiple connections. However, dedicated pins for various communication protocols including SPI, I2C and UART is what makes it a considerably better option than the rest. The board is also much easier to implement in our system as it handles the power supply for the ATmega2560 chip while also providing a power supply option to the user at 5V with a max current output of 40mA. It also has built in 4KB of EEPROM memory, which allows us to generate and hold our piece location data inputted by the multiplexer and transmit it to the mobile application without needing to worry too much about problems with memory getting overwritten. This setup also fluidly works with our bluetooth module and multiplexer, which use the UART and SPI protocol respectively, for communication.

4.2 Bluetooth: HM-18 BLE Module

Requirements **[RS-2.1.08 to RS-2.1.12]**, **[RS-2.1.16 to RS-2.1.17]** and **[RS-2.2.11]** imply wireless communication between the microcontroller in the chessboard and the mobile application. To establish this 2-way communication, a BLE module will be used. BLE allows for the exchange of small amounts of data periodically while consuming the minimum amount of energy. Unlike traditional bluetooth connections, BLE is ideal for instantaneous seamless connections without the tedious pairing instructions required with traditional bluetooth.

Another reason for choosing BLE as the communication link is because our app will be built using React Native. With React Native, we can make use of the versatile open source react-native-ble-plx library [4] which allows for easy implementation of BLE into our project.

Design ID	Description	Corresponding requirement & design specification
Des 4.2.1	The bluetooth module will support connections to both iOS & Android	**NA**
Des 4.2.2	The bluetooth module will be able to both send and receive data between the microcontroller and mobile application.	**RS-2.1.08 to RS-2.1.12** **RS-2.1.16 to RS-2.1.17** **RS-2.2.11**

Table 4.2.1: Bluetooth Design Specifications

The above two requirements are not labelled with any specific phase because they will be included in all three phases.

For our system, the BLE module we have chosen is the HM-18 Bluetooth 5.0 Module. This module was designed specifically for the arduino, our microcontroller of choice, and supports both iOS and Android.

Figure 4.2.1: HM-18 BLE Module [12]

Chip: CC2640R2F
Bluetooth Version: 5.0 BLE
Protocol: UART
Supply Voltage: 3.3-6V
Working Frequency: 2.4 GHz
Power Consumption: 2.6mA
Range: ~100m

Table 4.2.2: HM-18 Specifications

The HM-18 makes use of one of the Arduino's UART connections for its operation. The pin layout for this module can be seen below in figure 4.2.2. The main pins needed for our application are the VCC, GND UART_TX and UART_RX. Although this module's supply voltage operates at 3.3-6V, the TTL logic level operates at 3.3V exclusively. Since the arduino pins are rated to provide 5V, a circuit is needed to lower the voltage of the transmit and receive pins of the BLE module. This circuit can be seen below in figure 4.2.3 and it consists of a voltage divider that lowers the 5V pinout voltage to 2.5V (Note: figure shows arduino micro with HM-10 module, however, the circuit is the same for the Arduino Mega with a HM-18 module).

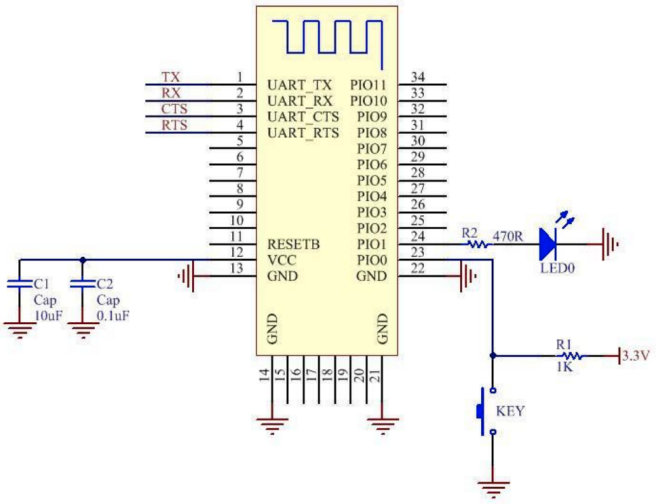


Figure 4.2.2: HM-18 Pinout [12]

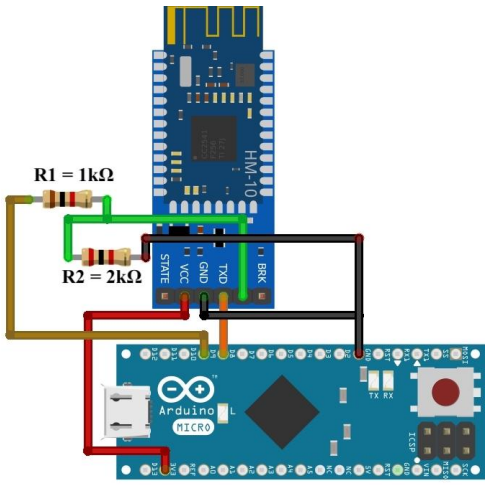


Figure 4.2.3: TTL Logic Voltage Divider Circuit [13]

$$Vin_{HM-18\ RX\ pin} = Vout_{Arduino} \times \frac{R1}{R1+R2} = 5V \times \frac{1k\Omega}{1k\Omega + 2k\Omega} = 2.5V$$

4.3 RFID Piece Detection

Requirements **[RS-2.2.01-A]**, **[RS-2.2.02-A]** and **[RS-2.3.4-A]** specify that the chess board will detect individual chess pieces placed on each grid of the board, to seamlessly detect the board state and player moves and transmit to the companion app. To achieve this, a system using RFID tags and antennas is used to detect individual pieces on a grid-by-grid basis. An RFID antenna will be placed at each grid of the chessboard, and will detect each chess piece that is placed on the board at each specific location. A system overview of components is shown below in figure 4.3.1. Our design uses a combination of Arduino and PN532 RFID module to interface multiple antennas, selected using an analog multiplexer to read each grid of antenna sequentially. For the alpha phase, there will be a total of 9 antennas in a 3x3 grid, but for beta and final product, there will be 64 antennas, each representing every square of the chess board.

Figure 4.3.2 shows the operating state of the piece detection. During the player's turn, the board will enter the piece detection state, continuously polling each antenna for placed chess pieces. When all 64 antennas are read, the state of the board is sent to the companion app through BLE. When a change in board state is detected, it is implied that the player has lifted off a piece, and the board will continue to poll the antennas until a state change is detected again, which indicates that the player has made their move. During the opponent's turn, the piece detection is not active, as the board is aware of the movement to be made. This also prevents any interference from the

electromagnetic movement system while active. When it is the player's turn again, the board will resume polling for the player's moves.

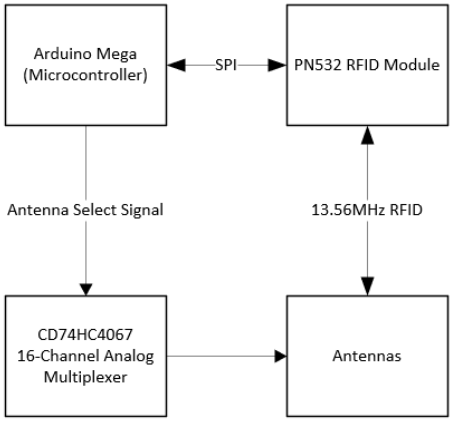


Figure 4.3.1: System Overview of RFID Piece Detection

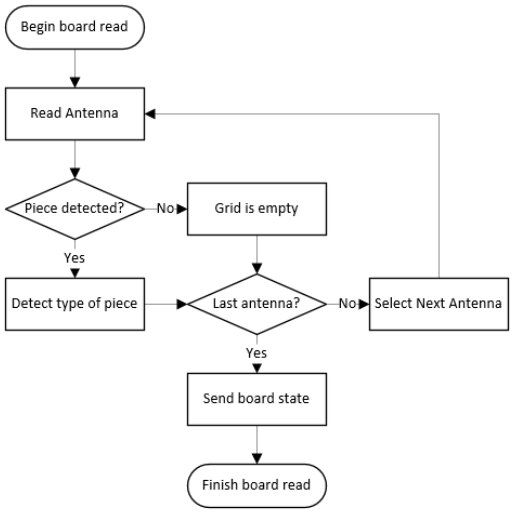


Figure 4.3.2: Board reading state flowchart

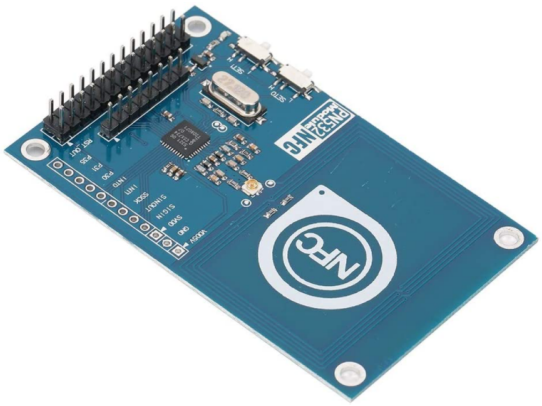


Figure 4.3.3: PN532 RFID Module [14]

Chip: NXP PN532
Operating Voltage: 3.3V
Supply Voltage: 3.3~5.5V
Working Current (Standby): 100mA
Working Current (Read/Write): 120mA
Working Frequency: 13.56MHz
Interface: SPI, RPi 20pin, I2C, UART

Table 4.3.1: PN532 RFID Module Specifications [15]

RFID Module

To read and write to RFID tags from antennas, an RFID module is required to perform this task. The ITEAD PN532 NFC Module was chosen for our design, shown above in Figure 4.3.3. Using

the NXP PN532 IC, this module can interface with the Arduino using a variety of protocols, including UART, SPI, and I2C. In addition, this module is easily modified to allow an external RFID antenna to be connected, rather than using the built-in antenna on the board. NFC, or near-field communications, uses 13.56MHz RFID interface to communicate over small distances of 10cm or less. An advantage of NFC is the short communication distance, allowing accurate detection for each grid of the chessboard, without interference between each other. Also, the target device, or the tags, can be completely passive, ideal for chess pieces requiring no additional power. The module is controlled by Arduino through SPI, using the provided Arduino NFC library.

RFID Tags

For storing unique information of each chess piece, a passive RFID tag will be placed at the bottom of each piece which will be detected by the antenna when placed on top of the chessboard. For meeting the requirements of locality of detection, as well as avoiding interference with the movement system, an anti-metal RFID tag has been chosen for our design. An anti-metal tag consists of an additional layer of ferrite sheet, which prevents interference when placed directly on metal surfaces.

Figure 4.3.4: Anti-metal RFID Tag [16]

For the alpha prototype, a circular anti-metal RFID sticker is used, shown above in figure 4.3.4. These tags contain an NXP MIFARE Classic IC, containing 1kB of EEPROM memory. The diameter of these tags are 25mm, which exceeds the size of the base of the chess pieces, but a smaller tag will be used for beta and final product. Each tag will pertain information of the type of chess piece, for detecting the specific piece that is placed on the board.

Antennas/Multiplexer

To read the RFID tag embedded in the chess piece, each square of the chessboard will have an antenna centered under the grid, for a total of 64 antennas in the beta and final product. However, to address multiple RFID antennas, additional setup is required. The antennas cannot be passively connected in series or parallel, as doing so compromises the antenna working frequency as well as power. Therefore, each antenna must be addressed individually by the RFID

module at a time. This will be achieved with a use of an analog multiplexer controlled by the Arduino to select each antenna at a time.

Figure 4.3.5: PulseLarsen NFC Flex Stamp Antenna [17]

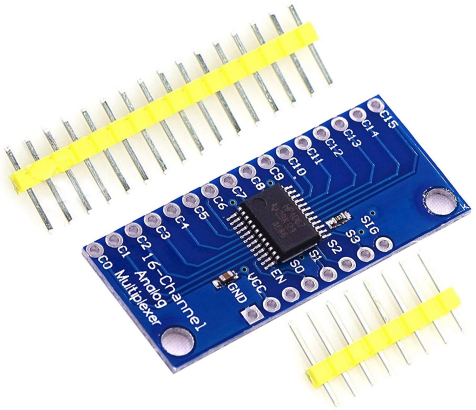


Figure 4.3.6: CD74HC4067 16-Channel MUX [18]

The antenna chosen for our design is the PulseLarsen W7001 NFC Flex Stamp Antenna, shown in figure 4.3.5. With an overall dimension of 25mm x 25mm, and thin flexible build, this antenna was perfectly suitable for our needs. One end of each antenna is connected directly to the NFC module, and the other end is connected to each output channel in the multiplexer, which is connected to the NFC module. From testing, the tag and antenna combination provided detection range up to 7mm offset from the center of the antenna, as well as 10mm of vertical range, meaning cross-interference between antennas were not an issue.

The multiplexer controlling the antennas is a TI CD74HC4067 16-channel analog multiplexer, as shown in figure 4.3.6. With bidirectional signal flow, and support for high-frequency antenna signals, this module is able to seamlessly select and transmit the read RFID signal. For the alpha phase, only one multiplexer is needed to address all nine antennas, using 4 digital GPIO pins from Arduino. For the beta and final product, a total of 5 MUXes will be used in a stacked configuration, using 6 GPIO pins, as shown below in figure 4.3.7.

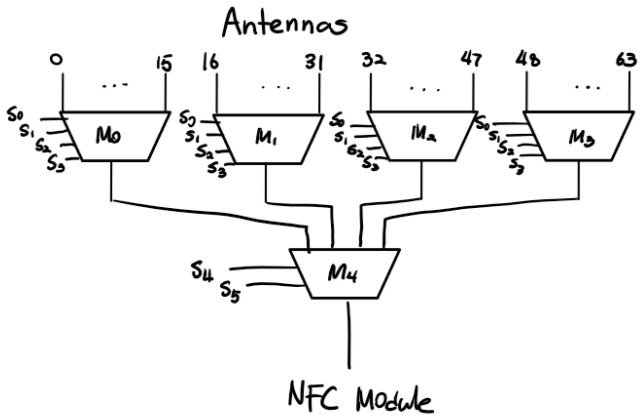


Figure 4.3.7: 64-bit MUX configuration

Design ID	Description	Corresponding requirement & design specification
Des 4.3.1-A	There are 9 RFID antennas in a 3x3 grid	**RS-2.2.01-A, RS-2.2.02-A**
Des 4.3.2-A	The Arduino will continuously detect changes of pieces placed on the board	**RS-2.2.01-A, RS-2.2.02-A**

Table 4.3.2: RFID Piece Detection Design Specifications Specific to alpha phase

Design ID	Description	Corresponding requirement & design specification
Des 4.3.1-B	There are 64 RFID antennas in a 8x8 grid	**RS-2.2.01-A, RS-2.2.02-A**
Des 4.3.2-B	The Arduino will detect changes during player's turn	**RS-2.2.01-A, RS-2.2.02-A**
Des 4.3.3	Each chess piece will have an embedded RFID Tag	**RS-2.3.4-A**
Des 4.3.4	After polling the entire board of antennas, board state is sent through BLE to the app	**RS-2.2.01-A, RS-2.2.02-A**
Des 4.3.5	RFID module communicates with Arduino using SPI	**NA**
Des-4.3.6	6 GPIO pins from Arduino are used to control the multiplexers	**NA**

Table 4.3.3: RFID Piece Detection Design Specifications

4.4 Chessboard & Chess Pieces

Chessboard

Requirements **[RS-2.2.16]** and **[RS-2.3.2 to RS-2.3.3]** imply every square on the chessboard must have a width that is at least twice as big as the diameter of the base of the biggest chess piece, the king. This will allow a piece to move in between two squares without disturbing the neighbouring piece (figure 4.4.1). To accomplish this, we will use chess pieces with a king base diameter of 2.3cm. This will require a square size of 4.6cm and since there are 8x8 squares on a chessboard, the main playing field will have a total area of:

$$Playing\ field\ area = 36.8cm \times 36.8cm = 1354.24\ cm^2$$

However, for the PoC we are only demonstrating the core concepts of our idea like the movement and detection and a full board is not required for that. So, for the PoC the chess board will only be a 3x3 board (Size may change depending on time and resources).

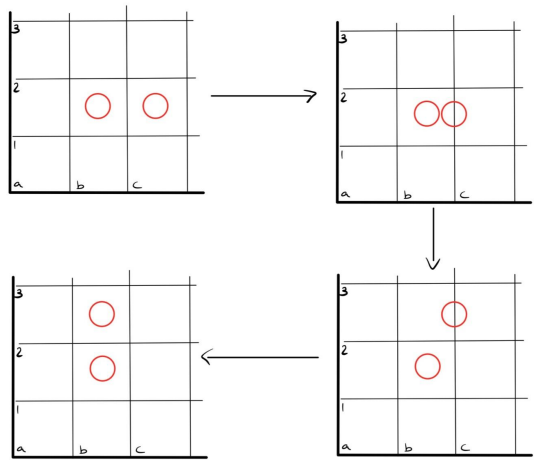


Figure 4.4.1: Example of movement without hitting neighbor piece

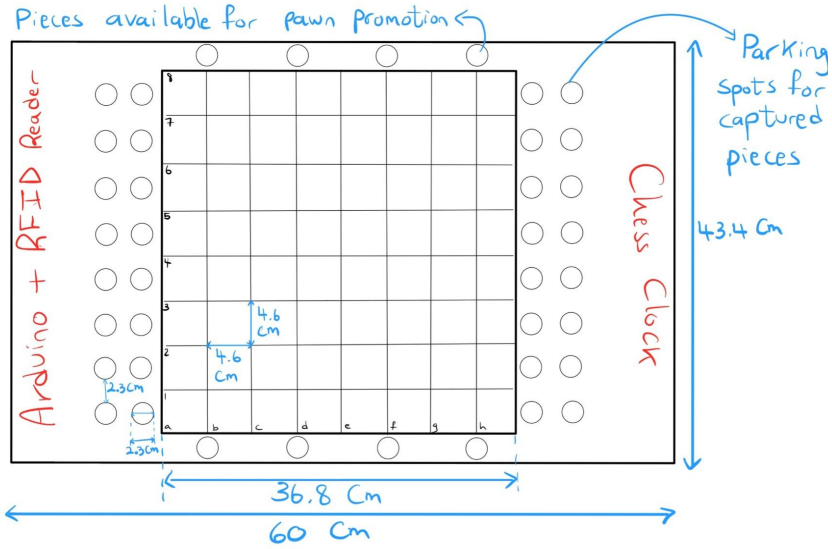


Figure 4.4.2: Top view layout of the board

Design ID	Description	Corresponding Requirements & Design Specifications
Des 4.4.01-A	Board is a 3x3 grid instead of the full 8x8	**NA**
Des 4.4.02-A	Squares have a width of 4.6cm to accommodate piece movement	**RS-2.2.16-B** **RS-2.3.2-A** **RS-2.3.3-A**

Table 4.4.1: Chessboard PoC Design Specifications

Design ID	Description	Corresponding Requirements & Design Specifications
Des 4.4.03-B	Board is a full 8x8 wooden chessboard	**RS-2.1.01-B** **RS-4.0.1-B** **RS-4.0.2-B** **RS-4.0.3-B**
Des 4.4.04-B	Board has markings from 1-8 for the ranks and a-h for the files	**RS-2.2.04-B** **RS-2.2.05-B**
Des 4.4.05-B	Board has a parking spot for every piece indicated by the shape of the piece on the sides of the board	**RS-2.2.06-B** **RS-2.2.07-B** **RS-2.2.08-B** **RS-2.3.5-B**

Table 4.4.2: Chessboard Design Specifications specific to beta phase

Chess pieces

Based on our market research survey, the majority of players prefer wooden pieces. The dimensions of these pieces can be seen in figure 4.4.3 below. To enable movement with the electromagnet, the chess pieces will have a hole drilled in the bottom and a steel metal insert will be placed inside as seen in figure 4.4.4. The metal insert will have a diameter of 0.6cm and a height of 1.2cm. It is made out of steel because of its high magnetism properties. Below the metal insert will be the Anti-metal RFID tag mentioned earlier and the final layer will be a felt sticker to reduce friction.

Figure 4.4.3: Chess piece dimensions [19]

Figure 4.4.4: Metal insert inside chess piece

Friction

The felt layer at the bottom of the chess piece is necessary to reduce the friction forces between the piece and the board. This will also dampen the sound while the piece is moving and eliminate scuffing the wooden board which is mentioned in requirement **[RS-2.2.12-B].** The difference in friction between wood-wood and felt-wood can be seen in the calculations below. When the piece is stationary, it will experience the highest friction force because the static friction coefficient is much larger than the kinetic friction coefficient ($\mu_s > \mu_k$). So, the calculations below consider static friction only as it will be the largest force to overcome by our movement system.

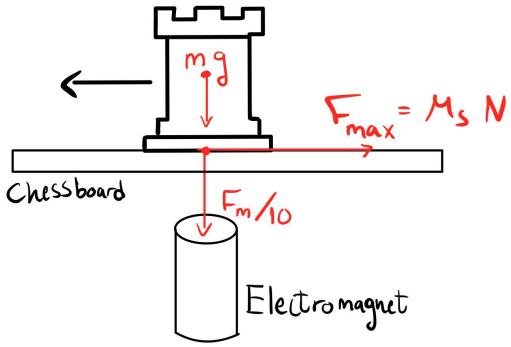
The weight of the chess piece will consist of two elements. The first is the weight of the piece itself, represented by **m**. The second element is the additional weight from the electromagnet pulling the chess piece down into the board. The pulling force of an electromagnet is usually represented by a quantity called the **Holding force (Fm).** The holding force is the maximum force of the electromagnet measured in kg that is required to pry the magnet away from a ferromagnetic surface when they are in direct contact. In real world applications, the true magnetic force is never equal to Fm. Instead, it is usually Fm divided by a factor of 5-10 depending on the environment, how much surface contact there is between the electromagnet and the metal and if there is a gap between them. In our system, we are at the worst possible case of having a gap and material between the chess piece and the electromagnet. So, we will use a factor of (Fm/10) to represent the extra weight on the chess piece when calculating our friction force.

$F_m = 5\ kg\ (Holding\ force\ of\ our\ electromagnet)$

$m = 0.005\ kg\ (weight\ of\ chess\ piece\ with\ metal\ insert)$

$\mu_{s1} = 0.5\ (static\ friction\ coefficient\ of\ wood - wood)$

$\mu_{s2} = 0.29\ (static\ friction\ coefficient\ of\ felt - wood)$



$$M_{total} = m + \frac{F_m}{10} = 0.005 + \frac{5}{10} = 0.505\ kg$$

$$N = M_{total} \times g = 0.505\ kg \times 9.81\ {}^{m}/_{s^2} = 4.954\ N\ (Normal\ force\ on\ chess\ piece)$$

$$F_{max\ friction\ 1} = \mu_{s1} \times N = 0.5 \times 4.954 = 2.477\ N$$

$$F_{max\ friction\ 2} = \mu_{s2} \times N = 0.29 \times 4.954 = 1.437\ N$$

Figure 4.4.5: Illustrations/calculations of friction forces on a chess piece [20]

From these calculations, we see that adding a felt pad under the piece reduces friction by approximately 42%. This method will be incorporated into both the PoC and the final product as it is necessary to achieve smooth and effortless movement of the chess piece.

Design ID	Description	Corresponding Requirements & Design Specifications
Des 4.4.06	Chess piece dimensions will be between 1.8cm to 2.3cm	**RS-2.3.2-A** **RS-2.3.6-A** **Des 4.4.02-A**
Des 4.4.07	Chess pieces are wooden	**RS-2.3.7-B**
Des 4.4.08	Chess pieces have a metal insert with a diameter of 0.6cm and height of 1.2cm to be attractable to the electromagnet	**RS-2.3.1-A** **RS-2.2.03-A** **Des5.2.2**
Des 4.4.09	Chess pieces have a anti-metal RFID tag embedded in them for detection	**RS-2.2.01-A** **RS-2.3.4-A**
Des 4.4.10	Base of the chess pieces have a felt pad to reduce friction	**RS-2.2.12-B** **Des 5.2.1**

Table 4.4.3: Chess Piece Design Specifications

The above table contains general design specifications that will be included in all three phases, and thus, they are not labelled with any specific phase. The table below contains design specifications that are specific to the beta phase.

Design ID	Description	Corresponding Requirements & Design Specifications
Des 4.4.11-B	Extra chess pieces are provided for pawn promotion	**RS-2.3.5-B**

Table 4.4.4: Chess Piece Design Specifications specific to Beta Phase

5 Movement System

5.1 Solenoid based electromagnetic array

Our current implementation of the movement system is based on an array of solenoids. On top of this array of solenoids is a permanent magnet. To navigate this permanent magnet, the solenoids will be individually triggered, forcing the magnet (and consequently the electromagnet) to move

in the process which in turn will move around the pieces when desired.

To determine the characteristics of our solenoid, first the frictional force acting on the loaded permanent magnetic was determined.

$$Total\ mass\ of\ the\ magnet + load = M_{total} = 0.0575\ kg$$

$$F_{friction} = \mu_s \cdot F_{normal}\ ;\ where\ \mu_s = 0.2\ (Steel\ against\ polyethylene)$$

$$F_{friction} = 0.0115N$$

As mentioned in figure 4.4.5, the friction contribution from the electromagnet pulling the piece will be 1.473N.

Therefore the x-component of the attractive magnetic force must overcome 1.4845 N frictional force in order to move the magnet with the load. With our current coil positioning, the attractive force produced by the solenoid must be at least 1.49498 Newtons to meet this requirement.

The magnetic force acting on a ferromagnetic material, produced by a solenoid can be determined by the following equation:

$$F_m = (N \cdot I)^2 \cdot \mu_o \cdot \frac{A}{2 \cdot g^2}$$

$where$:

$N = number\ of\ turns$

$I = current\ passed\ to\ wires\ in\ Amps$

$\mu_o = magnetic\ constant = 1.25663706 \times 10^{-6}\ m\ kg\ s^{-2}\ A^{-2}$

$A = cross-sectional\ area\ of\ solenoid\ in\ meters$

$g = distance\ between\ the\ solenoid\ and\ the\ ferromagnetic\ material\ in\ meters$

As the desired force can be achieved with multiple combinations, a MATLAB script was utilized to find a combination that maximized the magnetic force while minimizing the amount of material and work required to produce the solenoid . It was determined that if 19-gauge copper wire was used, a solenoid with 6 layers of 27 turns each or 162 turns total, will be sufficient to produce 2.7N magnetic force when 2 amps current is passed. This is significantly above our 1.49498 Newton target to liberally account for any error that was not identified.

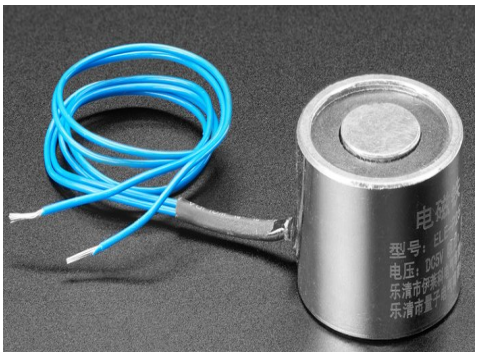
The following design specifications will be consistent for all design phases:

Design ID	Description	Corresponding Requirements & Design Specifications
Des 5.1.1	The solenoids will be able to withstand 2 amps of current	**RS-2.2.03-A** **RS-2.3.1-A** **RS-4.0.4-B**
Des 5.1.2	The solenoid will have a max height of 3cm and a diameter of 1.7cm	**None**
Des 5.1.3	The magnet's height will be 0.3175cm and diameter will be 3.175	**Des 4.4.2** **Des 5.2.2**
Des 5.1.4	The plexiglass sheet will be 0.1cm in thickness	**None**
Des 5.1.5	The solenoid inserts will be 0.6cm in diameter and 3cm long	**Des 5.1.2**
Des 5.1.6	The solenoid will be wound with 0.9mm copper wire	**Des 5.1.2** **Des 5.1.1**

Table 5.2.1: Movement system design specifications

5.2 AdaFruit 5 volt electro-magnet

The electromagnet used is a 5-volt ada fruit electromagnet with a holding force of 5kg. This is the lightest electromagnet available that satisfies our strength requirements. An added benefit is that it also uses only 5 volts to operate, which can be supplied through the arduino itself without needing an external power supply. The combination of strength, weight and ease of use made it the best choice for our design.



Nominal Voltage: 5V
Current draw: 0.3Amps
Holding Force: 5 Kg
Diameter: 25mm
Center Diameter: 12mm
Height: 20mm
Lead Length: 280mm

Figure 5.2.1: AdaFruit Electromagnet [21]

Table 5.2.1: Adafruit Electromagnet Specifications[21]

The following design specifications will be consistent for all the design phases:

Design ID	Description	Corresponding Requirements & Design Specifications
Des 5.2.1	The electromagnet has a holding force of 5Kg.	**Des 4.4.08** **RS-2.2.03-A** **RS-2.3.1-A**
Des 5.2.2	The electromagnet will be 20mm in height and 12mm in diameter	**RS-2.3.2-A**
Des 5.2.3	The electromagnet will operate with a 5V supply at 0.3amps	**Des 4.1.4** **RS-4.0.4-B**
Des 5.2.4	The electromagnet will be 50 grams in mass	**Des 5.2.1** **RS-2.2.15-B**

Table 5.2.2: Movement system design specifications

6 Conclusion

The document highlights our major design choices for the implementation of our chessboard and its prototypes.

The companion app for IntelliChess is designed to both be a simple interface for the user to interact with, mimicking just any other chess game, and utilizes the powerful microprocessors in modern smartphones to host our chess engine. By using React Native as our framework, we're able to develop the app for both iOS and Android simultaneously, maximizing on the amount of users that can take advantage of our product.

After careful consideration, we decided on the Arduino Mega as our microcontroller for our beta and final product as it offers special support for implementing our bluetooth module/multiplexer and has its own IDE which makes low-level programming on it considerably easier.

Our piece detection system utilizes RFID, which is relatively easier to implement than hall sensors, is not prone to interference with metals and is able to assign unique identifiers for every single piece making piece identification considerably easier and the board much more immersive

to interact with.

For the chess pieces themselves, we determined the garosa chess pieces to be optimal for our system since they are lightweight and work well with our board size constraints. The underside of the pieces is tapped with felt as our anti-friction solution. Felt is the cheapest and most widely available anti-friction tape, and also comes in sizes that match our piece bases making their implementation very easy.

The finalized movement system is based on an array of manually wound solenoids. It offers the most flexibility in terms of customization and produces a noiseless system that is much faster than our competitors X-Y mechanical gantry in terms of piece movement. It is also the most consistent in terms of piece positioning. The decided AdaFruit 5 volt electro-magnet for piece engagement, provides the most optimal balance between strength and weight.

7 References

[1] R. Dottle, “2020 Chess Boom,” *Bloomberg.com*, 16-Dec-2020. [Online]. Available: https://www.bloomberg.com/graphics/2020-chess-boom/. [Accessed: 20-Feb-2021].

[2] "Meet Google Play's target API level requirement", developer.android.com, 2021. [Online]. Available: https://developer.android.com/distribute/best-practices/develop/target-sdk. [Accessed: 27- Mar- 2021].

[3] "App Store Review Guidelines - Apple Developer", Developer.apple.com, 2021. [Online]. Available: https://developer.apple.com/app-store/review/guidelines/. [Accessed: 27- Mar- 2021].

[4] "Polidea/react-native-ble-plx", GitHub, 2021. [Online]. Available: https://github.com/Polidea/react-native-ble-plx. [Accessed: 27- Mar- 2021].

[5] "Introduction to Bluetooth Low Energy", Adafruit Learning System, 2014. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt. [Accessed: 27- Mar- 2021].

[6] "CCRL 40/15 - Index", Ccrl.chessdom.com, 2021. [Online]. Available: https://ccrl.chessdom.com/ccrl/4040/. [Accessed: 27- Mar- 2021].

[7] "Chess Engine | Top 10 Engines In The World", Chess.com, 2021. [Online]. Available: https://www.chess.com/terms/chess-engine. [Accessed: 27- Mar- 2021].

[8] "nmrugg/stockfish.js", GitHub, 2021. [Online]. Available: https://github.com/nmrugg/stockfish.js. [Accessed: 27- Mar- 2021].

[9] "devfd/react-native-workers", GitHub, 2016. [Online]. Available: https://github.com/devfd/react-native-workers. [Accessed: 27- Mar- 2021].

[10] "Arduino Mega 2560 Rev3 | Arduino Official Store", Store.arduino.cc, 2021. [Online]. Available: https://store.arduino.cc/usa/mega-2560-r3. [Accessed: 27- Mar- 2021].

[11] A. Aqeel, "Introduction to Arduino Mega 2560 - The Engineering Projects", The Engineering Projects, 2021. [Online]. Available: https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-mega-2560.html. [Accessed: 27- Mar- 2021].

[12] "DSD TECH HM-18 CC2640R2F Bluetooth 5.0 BLE Module Compatible with iOS and Android", Dsdtech-global.com, 2018. [Online]. Available: http://www.dsdtech-global.com/2018/06/dsdtech-hm-18.html. [Accessed: 27- Mar- 2021].

[13] "HM-10 Bluetooth Module Pinout, Applications, Interfacing with Arduino", Microcontrollers Lab, 2013. [Online]. Available: https://microcontrollerslab.com/hm-10-bluetooth-pinout-arduino-interfacing-applications/. [Accessed: 27- Mar- 2021].

[14] Amazon.ca, 2021. [Online]. Available: https://www.amazon.ca/gp/product/B07Y46XKM6/. [Accessed: 27- Mar- 2021].

[15] "ITEAD PN532 NFC MODULE - ITEAD Wiki", Itead.cc, 2021. [Online]. Available: https://www.itead.cc/wiki/ITEAD_PN532_NFC_MODULE. [Accessed: 27- Mar- 2021].

[16] Amazon.ca, 2021. [Online]. Available: https://www.amazon.ca/gp/product/B01FR6M8SS/. [Accessed: 27- Mar- 2021].

[17] Mouser.ca, 2021. [Online]. Available: https://www.mouser.ca/new/pulse-electronics/pulselarsen-antennas-w7001/. [Accessed: 27- Mar- 2021].

[18] Amazon.ca, 2021. [Online]. Available: https://www.amazon.ca/gp/product/B07KBTT5ST/. [Accessed: 27- Mar- 2021].

[19] Amazon.ca, 2021. [Online]. Available:https://www.amazon.ca/Chessmen-Figures-Figurine-Standard-International/dp/B08FFL5XJT/ref\=sr_1_8?dchild=1&keywords=wooden+chess+pieces&qid=1616564266&sr=8-8. [Accessed: 27- Mar- 2021].

[20] "Friction and Friction Coefficients", Engineeringtoolbox.com, 2021. [Online]. Available: https://www.engineeringtoolbox.com/friction-coefficients-d_778.html. [Accessed: 27- Mar- 2021].

[21] A. Industries, "5V Electromagnet - 5 Kg Holding Force", Adafruit.com, 2021. [Online]. Available: https://www.adafruit.com/product/3873. [Accessed: 27- Mar- 2021].

[22] I. Batterbee, “Don Norman's seven fundamental design principles,” *Medium*, 05-Feb-2021. [Online]. Available: https://uxdesign.cc/ux-psychology-principles-seven-fundamental-design-principles. [Accessed: 19-Mar-2021].

[23] "IEEE/IEC 82079-1-2019 - IEEE/IEC International Standard for Preparation of information for use (instructions for use) of products - Part 1: Principles and general requirements", Standards.ieee.org, 2019. [Online]. Available: https://standards.ieee.org/standard/82079-1-2019.html #Additional. [Accessed: 21- Mar- 2021].

[24] "ISO 9241-11:2018", ISO, 2018. [Online]. Available: https://www.iso.org/standard/63500.html . [Accessed: 21- Mar- 2021].

[25] "ISO/IEC TR 11580:2007", ISO, 2007. [Online]. Available: https://www.iso.org/standard/42779.html . [Accessed: 21- Mar- 2021].

[26] “Image: Man Figure,” *Humaaans*. [Online]. Available: https://www.humaaans.com/. [Accessed: 20-Mar-2021].

[27] “Image: Chess Piece Figures”. *Font Awesome*. [Online]. Available: https://fontawesome.com/license. [Accessed: 20-Mar-2021].

[28] A. Medina, “Image: Chess Background Design Free Vector,” *Freepik*, 06-Nov-2019. [Online]. Available: https://www.freepik.com/vectors/background. [Accessed: 16-Mar-2021].

[29] D. Gandy, “Image: Power Button Off free vector icons,” *Flaticon*. [Online]. Available: https://www.flaticon.com/free-icon/power-button-off. [Accessed: 16-Mar-2021].

[30] C. Burnett, “Image: Chess klt45.svg,” *Wikimedia Commons*. [Online]. Available: https://commons.wikimedia.org/wiki/File:Chess_klt45.svg. [Accessed: 17-Mar-2021].

Appendix A: User Interface & Appearance Design

A.1 Introduction

The game of chess has deep historical roots and traditions and has largely remained unchanged for hundreds of years until recently. With the rapid technological advancements made in the 20th and 21st centuries, online chess, powerful chess computer opponents, and game analysis tools have now become a reality. Intellichess aims to integrate some of these technologies together with a real, physical chessboard all while striving to maintain the traditional look and feel of a classic chess board as much as possible.

Purpose

The purpose of this document is to outline the user interface (UI) design of the Intellichess smart chess board, including the mobile companion application as well as the physical board itself. By the end of the document, it should be clear to the reader why certain UI design choices were made.

Scope

This document will cover the following areas with respect to the Intellichess smart chess board system:

- User Analysis
    - An outline of the knowledge required by the user to operate the system
    - Restrictions with respect to the users prior experience with similar systems or devices
    - Restrictions with respect to the users physical abilities to use the system
- Technical Analysis
    - Technical analysis of the system, taking into account the “Seven Elements of UI Interaction” (discoverability, feedback, conceptual models, affordances, signifiers, mappings, constraints) outline in Don Norman’s text (The Design of Everyday Things)
- Engineering Standards
    - An outline of the engineering standards that apply to the system’s user interfaces
- Analytical Usability Testing
    - Details concerning the analytical usability testing undertaken by the design team
- Empirical Usability Testing
    - Outlines the methods of testing required for future implementations of the system
    - Addresses safe and reliable use of the system

A.2 User Analysis

Our board's ability to both train and compete at a very high level, makes it a good choice for both professionals and beginners alike. Therefore, our target audience is simply chess players. However, our board's ability to move its own pieces and mimic an actual user makes it an especially attractive choice for people who value the feel and tactile aspect of the game. Also, the ability to train with an AI that can match your skill level makes this board a good choice for people who are actively looking to improve their game.

To use our product the user is required to have an introductory understanding of the rules of chess. Even though the product incorporates a trainer, the board will simply tell you when you have made an incorrect move, but not mention why the move is incorrect. Likewise, the board will tell you the best move but not tell you why the move is the best. If the user does not have prior understanding of the type of moves each piece is able to execute, they will not be able to utilize the training feature properly.

The user is also expected to have basic knowledge regarding the operation of smart phones. This is because our system incorporates a mobile application that is required to calibrate the board and carry out certain tasks such as selecting sides, starting a match, selecting opponent difficulty etc.

A.3 Technical Analysis

In the book "The Design of Everyday Things" by Don Norman [22], the author lists a set of 7 fundamental design principles that make your product more efficient and user friendly. These principles are discoverability, feedback, conceptual models, affordance, signifiers, mapping and constraints. In this section we discuss how our design will apply to these principles to help the user enjoy an overall better experience with our product.

Discoverability

Discoverability refers to how intuitive different elements of your design are to find. The user should discover any functionality of the product or UI with ease. For our product, IntelliChess, the discoverability will consist of:

- IntelliChess board looks just like a normal chessboard, eliminating any added complexity to learning chess differently from a traditional chessboard.
- A power button placed on the side of the board allows the user to turn on the product (Figure A.7). Using Bluetooth Low Energy (BLE), the board connects instantaneously to the user's smartphone with the click of a button in the app (Figure A.1), so no tedious bluetooth pairing operation is needed.
- The mobile app design incorporates all the functionality the user needs like the hint button and the end turn button so they are accessible at all times (Figure A.3).

Feedback

Feedback refers to how the product communicates the result of the user's actions. It is essential to inform the user that they accomplished what they intended to do. It also refers to the state of the product if any errors are detected. For IntelliChess, feedback will consist of two components, hardware and software.

- Hardware:
    - Once the user plays their turn, the computer opponent responds with a move of their own and the chess piece moves automatically.
- Software:
    - Haptic feedback through the phone vibration indicates to the user when a button is pressed (iPhone only)
    - After every turn, the mobile app updates the digital board with the move that was played. So, the user knows the board detected their correct move (Figure A.3).
    - The ticking clock indicates how much time both the user and the computer opponent have left
    - Once the game is finished, user is presented with an end screen indicating win/loss and how their chess elo score changed (Figure A.4)

Conceptual Model

Conceptual Model refers to the idea that there should be a simple and clear explanation of how something works. For example, visual cues, such as the trash icon, are used to clearly explain to the user how the delete button will work. Part of creating an effective conceptual model includes preventing the user from using the product the wrong way (for example, irreversible USB ports are designed to not allow the user to insert the USB connector in any other orientation)

- Software:
    - All of the buttons on our mobile app are clearly labelled, and only the actions available to the user are displayed on each screen
    - Each button will have three states (enabled, in progress, disabled), and the buttons will be disabled whenever an action is unavailable to a user (for example, when they make an invalid move, the end turn button will be disabled to clearly show that it is something that the user cannot do)

Affordance

Affordance is the idea that the *perceived* action and the actual properties of an object are connected, thus helping the user determine its operation. For example, a round knob suggests to the user that it can be turned and used to tune some sort of the control, and a button suggests to the user that it can be pressed once to perform some sort of action.

- Software:

- We use buttons extensively in our app to show the available actions to the user, actions like ending turns take their inspiration from the end turn buttons in chess tournament
- The virtual chess board allows the user to move each piece around, and to any available squares, much like its real life counterpart
- The game mode selection buttons work like slider buttons, where only one button may be selected at once, clearly explaining to the user than only one of each option may be chosen

Signifiers

Signifiers communicate to the user the specific action each object will perform, whether it's through text labels, color, sizing, or all of them.

- Hardware: Labelled power button with the well known on/off symbol used on almost every modern device
- Software:
  - Our app uses labelled buttons that are clear and concise on their exact actions
  - Whenever appropriate, illustrations are used in addition to the text labels, to further clarify the action of the objects
  - Consistent coloring helps to both tie together the UI design, as well as making it easier for the user to differentiate between different actions (Figures A.2,A.3)
    - Dark grey: Disabled button
    - Blue: Available button
    - Bright yellow: Hint button

Mapping

Mapping refers to the relationship between the action performed and the result of the action. For example, the volume bar on a phone slides up/down according to the user's finger. When a direct relationship does not exist, the design must utilize signifiers to establish this relationship.

- Software:
  - "End Turn" button on the mobile app stops the user's time which is situated directly above it (Figure A.3)
  - "Hint" button displays the hint on the digital board by lighting up the necessary squares needed to accomplish the move.

Constraints

Constraints limit the amount of interactions the user can take, thus reducing the information the user will need to process, reducing the possible points of confusion. For example, a mouse cursor isn't able to move outside of the screen, limiting the user to the physical boundaries and creating a clear area of usable application

- Software
    - Our app keeps the amount of available interactions to the user minimal (every screen has less than 5 points of available interactions)
    - We chose to keep the 3 essential game tweaking settings only (opponent, difficulty, time), and simplified the options to be less than 3 for each of them.
    - Disabled buttons will prevent the user from making illegal moves and performing actions that are unexpected, thus limiting the information they will have to process in order to make a decision on their next interaction.

A.4 Engineering Standards

IEEE/IEC 82079-1-2019	IEEE/IEC International Standard for Preparation of information for use (instructions for use) of products - Part 1: Principles and general requirements [23]
ISO 9241-11:2018	Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts [24]
ISO/IEC TR 11580:2007	Information technology — Framework for describing user interface objects, actions and attributes [25]

Table A.1: Relevant Engineering Standards

A.5 Analytical Usability Testing

The analytical usability testing is a component of our design which will utilize and involve the entire team. For our proof of concept, our main goal is to realize a system where our major design components are working and interacting fluidly. To achieve this task, we will be testing the components both individually and finally as a system to verify correct functionality.

In the first stage of testing, design teams will individually verify the correct operation of the following major elements of the project:

1. Consistent and correct detection of individual chess pieces and their movements using an arduino.
2. Solenoid based movement system and its ability to accurately carry the electromagnet without affecting other pieces.
3. Reliable communication with the mobile application, and the ability to read and write data without failure.

In the second stage of testing, the design teams will cooperate to realize the system as a whole and establish a working proof of concept with the following integrations tested:

1. Automatic transmission protocol initialization when the board is powered on with easy pairing with the mobile application
2. The ability to read the piece positions and transmit the location of each piece to the phone over bluetooth automatically after a single initialization command
3. Passing location information into the chess engine to calculate the next move then sending back the computers move to the arduino, along with displaying it for the user on the phone to follow.
4. The arduino successfully receiving the move directions and triggering the optimal solenoids to move the permanent magnet and carry out the move
5. Correct and timely triggering of the mounted electromagnet (using the arduino) to engage the correct piece and disengage only when at the correct location

A.6 Empirical Usability Testing

The empirical usability testing component of our design is a crucial element which will involve not only the input of the team, but also the input of potential users to rectify our existing design.

In the first part of our empirical usability testing (PoC testing), the team members will individually use the chessboard and report any encountered problems in terms of usability. This will include moving the chessboard around, setting up the chessboard as a first time user, connecting to the board to initialize a match, customizing the computer's skill level and personalizing match conditions. The responses will then be looked at to determine if there is something that needs to be addressed, the amount of work needed to fix the problem and note its impact on the potential final product.

For the engineering prototype and the final product, we will incorporate external usability testing. This will involve friends and family members who are not involved in the project but interested in the game of chess. They will be asked to use our product with minimal guidance and then asked to provide feedback in the form of a survey, based on their experience with the product.

The following questions will be included in the survey:

1. How portable is the IntelliChess board?
2. What did you think of the appearance of the board?
3. How easy was it to connect to the mobile application?
4. How easy was it to navigate the mobile application?
5. Does the board incorporate all the features you expected?
6. Did you run into any issues with piece movement or detection?
7. Would you recommend this board to someone else at the current price point?

A.7 Graphical Presentation

A.7.1 Companion Application Examples

Figures 1-4 below show our proposed app design which will help the user interface with the physical board and will also display a digital version of the chessboard that updates in real-time.

Upon opening the app, the user will be greeted with a very simple **Landing Page** (Figure 1), which only includes one button that will initiate the connection to the board. This is also where the Bluetooth permissions popup will show if the user has not previously given our app permission. Any errors in connection and basic troubleshooting steps will also be shown on this page, preventing the user from advancing to any other page.

If the connection is established successfully, users will be brought into the **Main Menu** (Figure 2), where they will choose the game mode, the difficulty, and the time restraint for their game. Once the game is set up, they will be brought into the **Gameplay Screen** (Figure 3), where a simple chess board will show them the current state of the IntelliChess board, and the amount of time each player has left for their game, it also includes two buttons, one that will allow them to end their turn (disabled until a valid move is made), and another that will show a hint for their next best move (highlighted on the app's virtual board, as well as the IntelliChess board). Once the game is finished, the **End Screen** (Figure 4) is shown, and the user is given the option to return to the **Main Menu** to set up their next game.

Figure A.1: Landing Page [26]

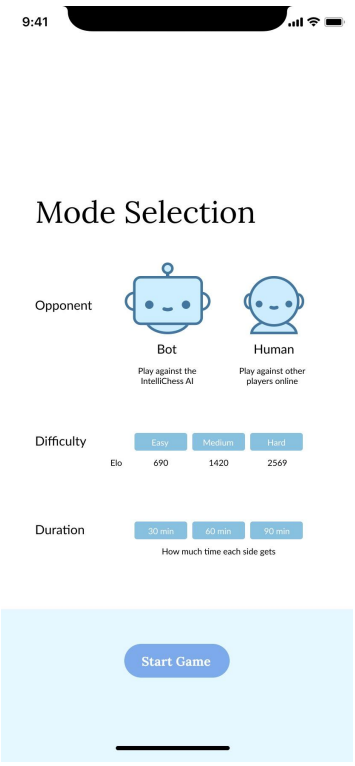


Figure A.2: Main Menu

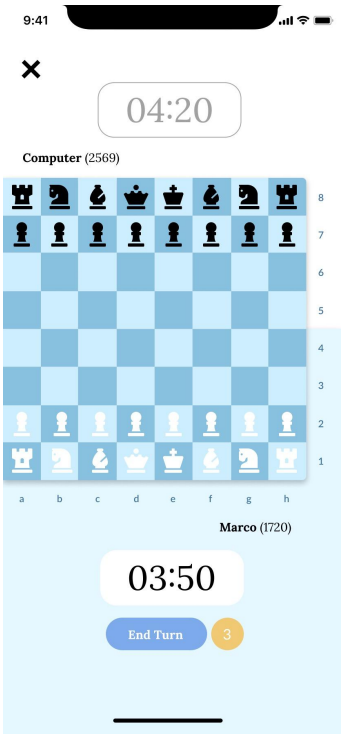


Figure A.3: Gameplay Screen [27]

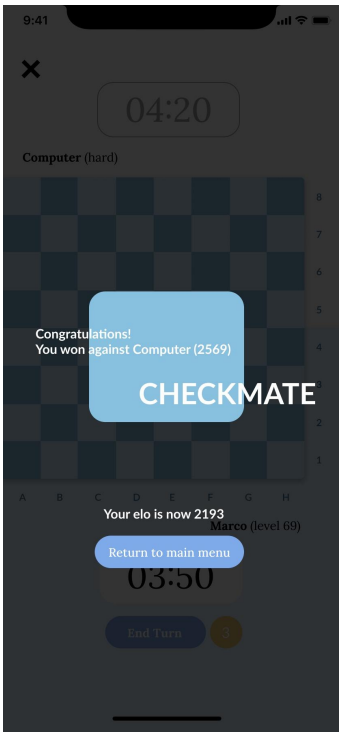


Figure A.4: End Screen

A.7.2 System Design Examples

Figures 5-7 below details our concept and overall design using a 3D model designed in Fusion 360. All units seen in any of the figures are in centimeters.

Figure A.5: Intellichess Smart Board 3D model [28, 29, 30]

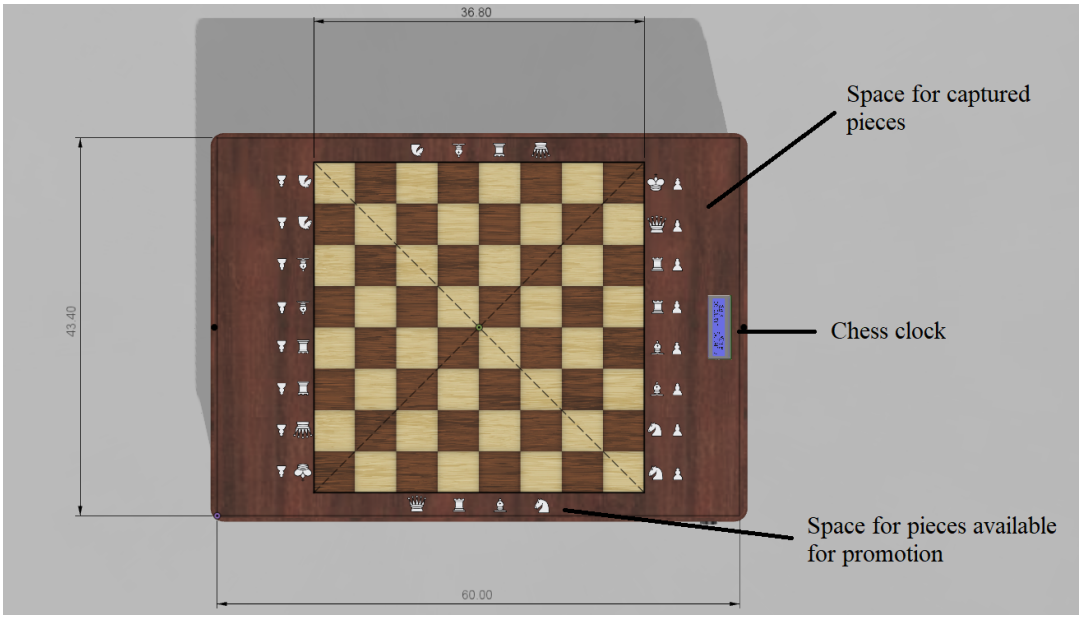


Figure A.6: Intellichess Smart Board Top View (units: cm)

Figure A.7: Intellichess Smart Board Front View (units: cm)

A.8 Conclusion

The UI and appearance appendix highlights our main design choices. We identify and recognize our main users to be general chess enthusiasts with a special appeal to chess players who value playing on the physical board. Inspired by "The Design of Everyday Things" by Don Norman, we use the author's seven fundamental design principles to organize our technical analysis. The main engineering standards have also been mentioned in the appendix report and cover both IEEE and ISO. Solid consideration has been given to the analytic usability of the system, with three fundamental design goals realized and listed. The empirical usability section builds on the analytic usability considerations by incorporating external users, and highlights the proposed survey that we will use to mitigate risk, reconsider design choices and organize execution of the final product. Finally the graphical presentation section presents the main components of the system as envisioned by the team for the final product. Some technical design details are also discussed in this section.

Appendix B: Supporting Test Plans

POC Specific

Associated Requirement ID	Test Description	Pass/Fail
RS-2.2.01-A	3x3 board initially empty, chess GUI shows empty board	☐ P ☐ F
RS-2.2.02-A RS-2.3.4-A	Place chess pieces on the board and indicate that they have been detected through the chess GUI	☐ P ☐ F
RS-2.2.02-A RS-2.3.4-A	Replace one of the pieces with another, and show that chess GUI has updated	☐ P ☐ F
RS-2.2.03-A RS-2.3.1-A	Remove all pieces except one, showcase piece movement in all directions (forwards, backwards and diagonal)	☐ P ☐ F
RS-2.2.03-A RS-2.3.2-A	Repopulate board and showcase a piece moving in between other pieces to the desired square	☐ P ☐ F

Table B.1: PoC tests

Mobile Application

Associated Requirement/ Design IDs	Test Description	Pass/Fail
Des 3.1.2-A	The user is asked for permission to enable bluetooth if not already enabled.	☐ P ☐ F
Des 4.2.1	The user is able to initiate a bluetooth connection between the board and the app, from the app.	☐ P ☐ F
Des 4.2.1	Connection between mobile application and board is successfully established after being initiated by the user.	☐ P ☐ F
Des 3.1.4-B	The user is notified whether the connection was successful or not.	☐ P ☐ F
Des 3.1.3-B	Application is able to recognize when the game has been set up correctly.	☐ P ☐ F

Des 3.1.4-B	The user is notified when the application recognizes that the game is ready to start.	☐ P ☐ F
RS-2.1.03-B	Application allows the user to select a side (black or white).	☐ P ☐ F
RS-2.1.02-B	Application allows the user to select a difficulty level.	☐ P ☐ F
RS-2.1.04-B	Application allows the user to select a time format.	☐ P ☐ F
Des 3.1.4-B	Application recognizes when the game has been started.	☐ P ☐ F
Des 3.1.4-B	Application emulates the chessboard and updates as gameplay progresses.	☐ P ☐ F
RS-2.1.12-B	The user is able to request hints in the application.	☐ P ☐ F
N/A	Best moves are listed when the user requests a hint.	☐ P ☐ F
N/A	The application notifies the user if the game has ended.	☐ P ☐ F
N/A	The application notifies the user if an error has occurred.	☐ P ☐ F
RS-2.1.08-B **RS-2.1.09-B** **RS-2.1.10-B**	The application notifies the user if they have played an illegal move, and why the move was illegal.	☐ P ☐ F
RS-2.1.18-B	The user should be able to resign in the application.	☐ P ☐ F

Table B.2: Mobile Application Tests

Piece Movement

Associated Requirement/ Design IDs	Test Description	Pass/Fail
RS-2.3.2-A	Pieces are able to be moved without colliding with one another.	☐ P ☐ F
RS-2.3.1-A	Pieces are left on the correct square after a completion of a move, and do not overlap with neighbouring squares.	☐ P ☐ F
RS-2.3.1-A	The correct piece is moved.	☐ P ☐ F
RS-2.2.03-A	Pieces are able to be moved in all directions.	☐ P ☐ F

Table B.3: Piece Movement Tests

Piece Detection

Associated Requirement/ Design IDs	Test Description	Pass/Fail
RS-2.3.4-A	The board is able to distinguish between the 6 different types of chess pieces.	☐ P ☐ F
RS-2.2.01-A	The board is able to differentiate between empty and occupied squares.	☐ P ☐ F

Table B.4 Piece Detection Tests

Appendix C: Supporting Design Options

C.1 Microcontroller

To control the movement of pieces, monitor piece positions, and perform gameplay analysis, using a microcontroller is one way to accomplish these tasks. Designing and implementing a custom controller would be overly difficult, time-consuming and unnecessary as there are a multitude of powerful microcontrollers available on the market. Some of these products that were considered are listed in the table below.

Option	**Pros**	**Cons**
Arduino Uno	- Compact - Multitude of online resources - Supports many additional modules such as Bluetooth and wifi modules	- Only 14 digital I/Os - 32 KB flash memory
Arduino Mega **(Chosen)**	- 54 digital I/O pins - 16 MHz clock - 256 KB flash memory - Supports many additional modules such as Bluetooth and wifi modules	- Larger than the Uno and Pi - Less resources online as UNO is more popular - More expensive than other options
Raspberry Pi Zero	- 40 GPIO pins - Cheapest out of the three options	- Not plug and play, risk of file corruption and other issues. - No on board storage

Table C.1.1 Design Microcontroller Options

Of the three options listed in table C.1.1 above, the Arduino Uno was the microcontroller chosen for the POC. It allows for quick and easy development, is relatively inexpensive, and contains all the functionality necessary for the POC. For the beta and final design however, the Arduino Mega will be used as the number of I/O pins the Uno provides will no longer be sufficient.

C.2 Bluetooth Module

Requirements **[RS-2.1.08 to RS-2.1.12]** and **[RS-2.1.16 to RS-2.1.17]** imply wireless communication between the microcontroller in the chessboard and the mobile application. To accomplish this 2-way communication there are 3 main options: Wifi module, BLE module or Traditional bluetooth. In the table below we discuss the pros/cons of each method.

Method	**Pros**	**Cons**
BLE Module **(Chosen)**	- Low energy consumption - Instantaneous pairing - Small in size - High location accuracy - Very secure connection	- Very slow compared to Wifi - Requires user to turn on bluetooth - Small data packets
Traditional Bluetooth	- Can send large data packets - High quality transmissions ideal for music - Faster than BLE	- Tedious pairing process - Requires user to turn on Bluetooth - Requires access to the Apple developer Bluetooth program
Wifi Module	- Can send the largest data packets - Speeds up to 1.3 Gbps [23]	- Large in size - More expensive than BLE - Susceptible to hacking

Table C.2.1: Different wireless communication methods

Based on our requirements, we decided to choose BLE as our way of wireless communication between the board and the app. Specifically, the HM-18 Bluetooth 5.0 BLE Module. BLE modules are the main communication method in any new IoT device. From smart watches to electric cars, BLE is the best option. It allows for relatively fast and reliable transfer of small amounts of data periodically. BLE also has a major advantage over traditional bluetooth because it eliminates the tedious pairing process we have come to hate over the years. Although wifi modules were also a great option, we decided against them because we will not be sending large amounts of data between the board and the app. Also, since our mobile app is based on React Native, BLE is a much better choice with extensive support options and online resources to implement in React Native.

C.3 Piece Identification

Keeping track of which pieces are on what squares is a major design component and multiple methods were considered before settling on a design. No matter the design choice, requirements **RS-2.2.01-A** and **RS-2.2.02-A** needed to be satisfied, the potential designs considered are listed below.

Method	**Pros**	**Cons**
Pressure Sensors	- Proven to work (implemented by competitors)	- Need to press the chess piece down to activate the sensor before moving. - Hinders users ability to play moves quickly
Hall Sensors	- Inexpensive - Immersive - Unable to uniquely identify pieces	- Potential interference with magnetic movement system - Requires pieces to be magnetic
RFID **(Chosen)**	- Immersive - Not currently implemented by any competitors - Can assign unique IDs to each piece.	- Potential locality issues - Requires 64 antennas - Potential interference with magnetic movement system

Table C.3.1 Piece Detection Design Options

RFID was chosen for piece identification, for both the POC and the final design. The locality aspect of RFID was the biggest issue, as an RFID reader with a range no greater than the area of a single chessboard square would be needed. An RFID antenna was found with locality small enough to ensure that there would be no unwanted reads from pieces on nearby squares, and was tested and confirmed to be suitable for the design. From further research, no other designs utilizing RFID technology have been found, making this method a unique and original approach.

C.4 Chess Pieces

Different styles of chess pieces of various shapes and sizes were considered during the design process. The different sets of pieces considered are listed below, along with their pros and cons. The sets of pieces were also constrained by our requirements, mainly **RS-2.3.6-B** and **RS-2.3.7-B**.

Option	**Pros**	**Cons**
Garosa chess pieces **(Chosen)**	- Made of wood (user preferred material) - Cheap ($15.69 CAD) - Non-toxic material - Piece dimensions allow for a board of appropriate size - Light - less friction between piece and board	- Esthetically not the nicest looking - Cheap craftsmanship
House of Chess boxwood chess pieces	- Excellent craftsmanship - Made of wood (user preferred material)	- Expensive ($119.95 CAD) - Piece dimensions would require too large of a board. - Heavy - increase friction between piece and board
Dioche chess piece set	- Light - Esthetically pleasing - Cheap ($17.09 CAD) - Eco-friendly material	- Material not preferred by users - Piece dimensions would require too large of a board.

Table C.4.1 Chess piece options

Based on our requirements, the first option in table C.4.1 was selected as it satisfies the preferred user chess piece material, is relatively cheap, and allows for our designs chess board to be of an appropriate size.

Another important design consideration involving the chess pieces, is the mitigation of piece friction with the board surface. As the pieces will be slid across the board using magnetism, the lower the resistance caused by friction, the better. Multiple methods of reducing friction, as shown in the table below, were considered, all which involve altering the bases of the chess pieces.

Option	**Pros**	**Cons**
Teflon disks	- Lowest coefficient of friction (0.05 - 0.10)	- Expensive - Need to get machined - Likely unable to get thin enough disks
Low Friction (UHMW) Tape	- Easy to acquire - Cheap ($20 CAD/roll) - Ultra thin - High abrasion and wear resistance	- Doesn't come in shapes that suit the piece base (need to later ourselves) - Coefficient of friction not as low as teflon (although it is close)
Felt **(Chosen)**	- Cheap - Easy to apply - Can be very thin	- Not as slippery as teflon

Table C.4.2: Chess Piece Friction Reduction Options

Felt was chosen for the POC as it is economical, easy to acquire (available at local craft stores), relatively thin, and has a low coefficient of friction. The second option will be used for the final design as it has high abrasion and wear resistance which will ultimately make the pieces last longer.

C.5 Movement System

Three design choices were considered for the movement system.

Apart from the decided standard solenoid configuration, we also considered implementing the entire solenoid grid on a PCB. This was to give us a considerably compact implementation where the 6mil trace of the PCB itself would be wound into coils. Finally, a X-Y mechanical gantry was also looked at to identify its usability.

The table below highlights the pros and cons of each system.

Option	**Pros**	**Cons**
Standard solenoid array **(Chosen)**	-Customizable -Can have metal insert to amplify magnetic field -Easy to design and produce -Very economical	-Requires extra sheet to move magnet -Takes up more space -Requires a lot of manual effort to produce
PCB solenoid array	-Integrated design -Easier to control switching -Much more compact	-Expensive to build -Hard to design -Produces very weak magnetic fields -Can not have metal insert
X-Y Gantry	-Most reliable movement -Can hold a very strong electromagnet easily -High degree of accuracy	-Needs recalibration after each turn -Very big and noisy -Takes more time to operate -Costly

Table C.5.1: 2D movement system implementation options

Standard solenoids were chosen to realize our movement system. We wanted a noiseless implementation with no lag time between each move - this almost immediately eliminated the X-Y gantry system as it required recurrent calibration and produced a considerable amount of noise during its operation. The PCB solenoids were simply not strong enough to move our loaded permanent magnet, and every change required a complete redesign (and ultimately a reprint) which was not economical at all. The standard solenoids were very cheap to make, offered a lot of customization in terms of choice of wire, height, metal insert etc. making it ideal for us as it offered us a lot of room to implement our new technology.

Option	Pros	Cons
AdaFruit 5V electromagnet **(Chosen)**	-5kg holding force	-Takes up less space -Heavier -Will generate more friction
Arduino electromagnet module	-Easy to control -Lighter	-Takes up more space -1kg holding force only

Table C.5.2: Electro-magnet options

After failing to move the chess pieces with the Arduino electromagnet module, we realized that we needed considerably higher holding force to engage the pieces. Therefore, the stronger AdaFruit electromagnet was selected. It is able to utilize the Arduinos built in power supply to produce a much stronger electric field but is also more taxing to the solenoid movement system underneath both because of the friction it produces and its own weight.