Thesis Project Portfolio

Assistive Chessboard

(Technical Report)

Human vs. Artificial Intelligence (AI) Matchups in Strategy Games have Facilitated the Growth of AI Technology

(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Iain Ramsey

Fall, 2022

Department of Electrical and Computer Engineering

Table of Contents

Sociotechnical Synthesis

(Executive Summary)

Human and Artificial Intelligence (AI) Interaction in Strategy Games

"Personally, I rather look forward to a computer program winning the World Chess Championship. Humanity needs a lesson in humility." – *Richard Dawkings*

My technical and STS topics are strongly related because both fundamentally involve human and AI interaction. In my STS topic, I researched three case studies of newsworthy human vs. AI matchups in strategy games and found that development of AI technology increased as an outcome of these events. In my technical report, I designed a chessboard where players can play against an AI, creating a device to facilitate human and AI interaction. I chose my technical report because I'm a chess enthusiast who plays against online chess AI almost daily, but I wanted the opportunity to play against AI in a physical setting as well. I chose my STS topic because strategy games like chess, go, and poker, have always been interesting to me and I wanted to find a research topic that intermixed these games with AI technology in a meaningful way.

For my technical report, I created an electronic chessboard that interfaces with a chess AI to recommend moves to the players. Using an array of magnetic sensors underneath the board, the location of each chess piece is detected, and this information is passed to a computer to be processed. Next, the chess AI calculates one or more recommended moves, and squares underneath the chessboard are illuminated with light emitting diodes (LED's) to indicate to the players which move(s) to make. This process repeats each time a player makes a move until a win, draw, or loss has occurred. In addition, a graphical user interface (GUI) is used to allow players to configure a multitude of settings like AI strength, number of recommended moves, and quantity of time for the AI to think. This assistive LED chessboard serves as a learning tool for hobbyist chess players and allows for humans to interact with an AI in a physical capacity, providing a unique experience that otherwise cannot be achieved online.

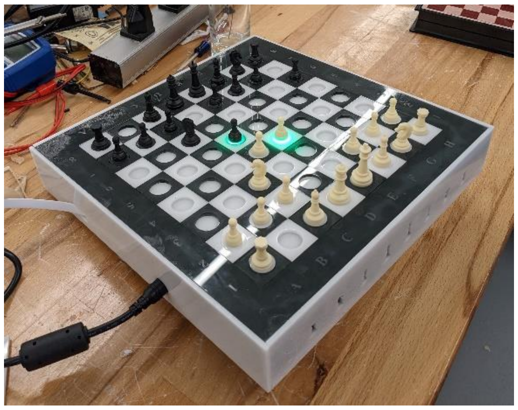
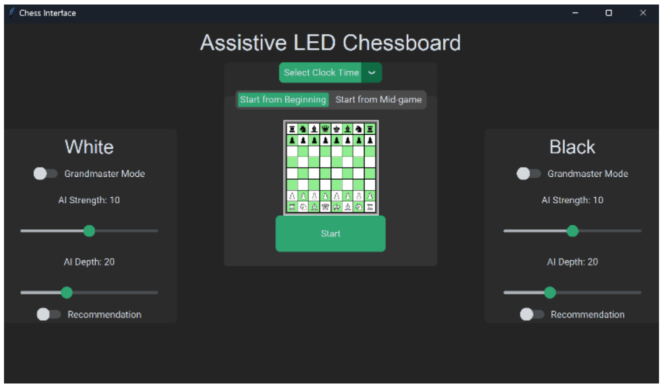


Figure 1. Assistive LED Chessboard



AI Strength: 10
AI Depth: 20
AI Strength: 10
AI Depth: 20

Figure 2. Graphical User Interface

For my STS research topic, I evaluated how newsworthy human vs. AI matchups in strategy games have affected the research and development of AI. I used Actor Network Theory (ANT) and case study analysis to guide my research. First, I analyzed the historic Gary Kasparov vs. Deep Blue chess match in 1997 and discerned how various actors responded to a computer overtaking the best human chess player. Next, I researched the AlphaGo vs. Lee Sedol Go match in 2016 and I speculated on possible implications for computer science theory in the future. Finally, I studied the Libratus vs. professional poker players event that occurred in 2017 and evaluated how skills utilized in poker can be translated into real world problems like business negotiations and diplomacy. The most important takeaway from my research was that successful victories by AI serve to broaden human's minds to the potential possibilities of AI and effectively expedite the development of the technology.

Strategy games and AI have been longstanding passions of mine, and both my STS and technical report provided opportunities to merge these two interests together in a way that also expanded my knowledge. My STS research enriched my technical report because I learned about the history of chess playing AI and how the technology has improved over the past three decades, enabling for a deeper appreciation of the AI that I was programming in code. My technical report enriched my STS research because I was able to replicate a similar environment to the Gary Kasparov vs. Deep Blue game, allowing me to consider things from Kasparov’s point of view and reflect how it feels to lose against a perfect chess playing robot.

Assistive Chessboard

A Technical Report submitted to the Department of Electrical and Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Iain Ramsey

Fall, 2022

Technical Project Team Members

Srikar Chittari

Ramie Katan

James Weeden

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Harry Powell, Department of Electrical and Computer Engineering

Assistive Chessboard

Ramie Katan, Srikar Chittari, James Weeden, Iain Ramsey

12/13/2022

Capstone Design ECE 4440 / ECE4991

Signatures [illegible]

[illegible]

James Weeden

[illegible]

Statement of work:

Ramie Katan:

My responsibilities primarily focused on designing an identification system for the chess pieces. Specifically, I researched Hall Effect sensors that would be both cost effective and act as a switch. As the Hall Effect sensors are digital sensors, being omni polar would create a more reliable system as we only need to detect whether a magnet was placed on the tile or not. I also was in charge of making sure the system was able to receive power as well as designing a way to power the system and control the sensor board using a Raspberry Pi Pico. I created all the schematics and board layouts for the PCBs using the capstone project. The voltage regulator board contains a step-down buck converter which converts 5V to 3.3V. This allows a safe supply voltage level for the raspberry pi pico and sensor boards to operate. I also was in charge of creating a power system for the LEDs so Srikar could interface with them. Throughout the project, I made sure to work closely with Srikar so that the embedded programs would work without significant modifications. I also helped debug issues with the sensor board, LEDs, and voltage regulator board to ensure completion of the project.

Iain Ramsey:

My responsibilities were implementing the serial communication between the Raspberry Pi Pico and PC, developing the finite state machine (FSM) game algorithm, and designing the physical board in CAD and manufacturing and assembling the acrylic parts. For the serial communication, I wrote both embedded C and python code to enable two-way communication between the Pico and PC. For the FSM, I created an algorithm that processes signals sent from the Pico and determines if a legal chess move has been made. The FSM also recognizes when illegal moves have been made, interfaces with the Stockfish chess engine, and writes signals to the Pico to illuminate the LED’s. The FSM serves as the bridge between many different subsystems because it interacts with the embedded software, GUI, and chess AI simultaneously. For the mechanical design and manufacturing, I created CAD files in Solidworks to model the chess board. Then, converting the CAD files into .dxf, I laser cut acrylic sheets using a machine in the mechanical engineering building. Finally, I used acrylic weld-on and super glue to assemble the acrylic pieces. I would like to acknowledge that a friend (Alexa Borden) in the mechanical engineering department provided me access to their laser cutter and advised me on the design of my CAD files. I would not have been able to build as pretty or robust of a chessboard without their help.

Srikar Chittari:

My main responsibilities were programming the firmware of the Raspberry Pi Pico and developing the Graphical User Interface (GUI). I programmed the Pico to read the hall-effect sensor network and send the chess board state to the PC over USB serial. I worked with Ramie to test and debug the signals from hall-effect sensors and multiplexers on the sensor board. Using Progammable I/O from the Pico's SDK, I programmed the Pico to interface with WS2812b addressable LEDs and drive a matrix of 64 LEDs under the chess board. I tested and debugged the protocol and LEDs, and I worked with Ramie to test and debug the logic level translator. I wrote a library in Python to create format recommendations and errors that would be sent to the LED driver code. I also built off James' initial work on the GUI and integrated it with the FSM's real-time updates. I implemented features that would allow the user to start at the beginning of a chess game or in the middle of a game and configure their move recommendations and Stockfish AI settings during the game. I developed James' preliminary chess clock to toggle the active player's clock automatically and support several time limits options. I improved the UI on the GUI to provide an intuitive and responsive experience during the game. As a secondary role, I helped Iain come up with ideas for the FSM implementation and detecting the chess board state.

James Weeden:

For the project my central role revolved around the creation of a graphical user interface (GUI) in Tkinter that allows for the user to configure settings of the stockfish chess AI in game. In completing this role, I built out the main Python framework, setting up widget placement and working to prepare the GUI so that Srikar and Iain could help get it working with the FSM in real time. Furthermore, I designed a menu system to allow users to control sliders that tune parameters of the AI that Srikar himself specifically connected the GUI to regardless of the side that they are playing. I created separate functions to implement countdown timers that were later transferred to more readable clock displays in standard time by Srikar. To give players a clear indicator of the connection between the board and the GUI, I created a live time display of the board that moved pieces in the center of the GUI with respect to actual movements on the board. This included design of a resizable chess set and implementation of custom piece graphics to make the board display easier to modify.

Table of Contents

Table of Figures

Table of Tables

Abstract

The Assistive LED Chess Board is an educational tool that promotes chess enthusiasts, beginners, and advanced players alike, to improve their skills. The interactive chess board illuminates the chess squares showing the user recommendations from a chess engine. Users can configure the recommendations of the chess engine, from engine strength and frequency to the number of recommendations in the graphical user interface. Each chess piece has a magnet at its base, and a network of 64 hall-effect sensors determines the position of the chess pieces. Using a known chess position, either the starting position or a mid-game position, the sensor network is scanned repeatedly by the Raspberry Pi Pico, tracking the movements of chess pieces. This method allows the identities of the pieces to be differentiated in software. Using the board position and the user's recommendation settings, chess move recommendations are generated using Stockfish [12] running on a personal computer (PC). The Raspberry Pi Pico interfaces with light-emitting diodes (LEDs) to illuminate the chess squares involved in the recommendations. This chess board integrates artificial intelligence and human-computer interaction, allowing chess players to study the strategies of a reputable chess engine while building their intuition and skills.

Background

Why we chose this project

The intersection of artificial intelligence (AI) and human-computer interaction (HCI) is one that lends itself to game products. In many instances, AI is used to create an opponent for users, but AI can also help players improve at logic games such as chess. The AI in chess engines can be harnessed to help beginners learn chess and assist advanced players to sharpen their skills. In addition, standard online chess lacks the HCI that can help beginners improve. By letting users interact with physical chess pieces and displaying recommendations visually, beginners are more likely to retain chess strategies. On the other hand, advanced chess players would benefit from a physical chess set as it can emulate a tournament-like experience. An interactive chess board that improves AI and HCI will not only motivate players to improve their craft, but it has the potential to change the status quo for how players practice chess.

The industry standard for assistive chess boards is ChessUp by Bryght Labs [2]. This board can interface with an AI, illuminate chess tiles using capacitive touch, and wirelessly track chess pieces on the board. The product can indicate the optimal move, a mediocre move, and a blunder based on the configured strength of the AI. ChessUp can also wirelessly interface with chess.com [22] to update a player's rankings. However, the steep $370 price tag on ChessUp is a limitation that may cause amateurs to look for alternatives.

Differentiating Factor

This assistive chess board is a popular concept and has been implemented several times in the literature. Specifically, piece detection is an active area of research. Muji et al. designed a chess board using linear hall effect sensors and differing magnets strengths for piece identification [9]. The chessboard was able to achieve a reliability of 80% on piece recognition. Kaufman, Patel, and Sun of the University of Illinois also designed a chess board using magnetic strength and analog hall-effect sensors to differentiate chess pieces [3]. Similar to the product by Bryght Labs, they used LEDs on the chess board to recommend chess moves. While their implementation was successful overall, they experienced unexpected behavior from hall-effect sensors. Another project by Coven of the Massachusetts Institute of Technology implemented piece detection using capacitive touch sensors and found the method to be highly susceptible to slight variations in piece alignment [11]. He found that the chess pieces needed to be placed exactly in the center of each chess square; placing a chess piece off center may cause the sensor to read the piece as if it was on the adjacent square. Existing implementations have used capacitive sensors or magnetic strength to differentiate chess pieces, but these methods often produce unreliable results and are easily influenced by the preciseness of the user's chess pieces.

This capstone project aims to improve on the aforementioned designs with respect to piece detection. Digital hall effect sensors are found to be more reliable because they are dependent on the presence of a sufficiently strong magnet and are not influenced by the strength of the magnet [23]. The hardware will be responsible for detecting the presence of a piece on a chess square, and software will handle differentiating chess pieces. Given a known chess position, either the starting position or a mid-game position, the placement of chess pieces can be tracked in memory as the game progresses. The software differentiating and tracking pieces will also check the validity of chess moves. This strategy will simplify the hardware needed for piece detection and prevent a chess piece from being misidentified by alignment issues between the piece and sensor. In addition, this project looks to improve on past implementations by creating a high degree of configuration for chess recommendation including AI engine strength, AI engine depth, and number of recommendations. This project pairs a simple but elegant hardware design with robust software game logic to create an intuitive and responsive chess board.

Previous Coursework

This project draws on knowledge gained in several engineering courses in the Electrical Engineering and Computer Engineering curriculums. Ramie will design the power supply using concepts taught in ECE Fundamentals I (ECE 2630), ECE Fundamentals II (ECE 2660), and ECE Fundamentals III (ECE 3750). He will also implement the hall-effect sensor network using magnets, logic level sensors, and multiplexers, which is material covered in Digital Logic Design (ECE 2330) and Electromagnetic Fields (ECE 3209). Embedded Systems (ECE 3501/3502) will play a crucial role in providing Iain and Srikar with an understanding of how to design the game's finite state machine (FSM) and Raspberry Pi Pico's firmware respectively. Srikar will also draw on concepts from Operating Systems (CS 4414) to synchronize concurrent tasks on the Pico. The material covered in Advanced Software Development (CS 3240) will help Iain integrate the Stockfish API and allow Srikar and James to develop the graphical user interface (GUI). Iain and Srikar will also draw from other computer science courses such as Program and Data Representation (CS 2150) and Algorithms (CS 4102) to integrate the software and firmware and implement efficient data processing.

Physical Constraints

Design Constraints

One of the biggest design constraints for our project is the computational power needed to run Stockfish to recommend moves in a relatively quick timeframe. The more memory Stockfish has allocated, the fast it will generate recommendations. We opted to use a PC over a Raspberry Pi to run Stockfish due to more RAM and multithreading to yield a more responsive chess board. Furthermore, our PCB would have benefitted from being slightly bigger as the LEDs on the LED light strip are spaced 1.1 inches apart, while the hall effect sensors are spaced 1 inch apart from each other. Unfortunately, due to the manufacturing limitations from Advanced PCBs, a 60 square inch board is the maximum size board. Due to the nature of that constraint, we needed to create two PCBs for this project.

Cost Constraints

One major constraint that we have identified midway through the design process was the amount of money 3W would charge for soldering the components onto the boards. As we have a budget of $500, if we were going to use our original design, it would cost us well over $500 just in soldering the components. Thankfully, we downscaled to a much more reasonable scale. Due to the downscale, we adjusted the number of parts needed for the project. Therefore, neither cost nor availability of parts was a concern for the majority of the project.

Tools Employed

For hardware design, KiCad was used to create the schematics and the board layout [14]. WeBench allowed for an easy way to select the voltage regulator components tailored for our needs [18]. Visual Studio Code was used for software development and debugging the embedded programs, FSM, and GUI [16]. The Arm GCC compiler was used to compile the embedded programs on the Raspberry Pi Pico [31]. CMake was used to manage the embedded build process and produce binaries for the Pico [32]. The embedded toolchain with Arm GCC compiler and CMake was a new workflow and involved a learning curve early in the project. The embedded programming was done in C [17]. The chess engine Stockfish was used to implement the game logic [12]. The FSM and GUI were programmed in Python 3 due its integration with Stockfish API, serial modules, and GUI libraries [15]. Git was used for software version control [33].

For the design and manufacturing of the acrylic board, Solidworks was used to design the CAD files [34]. Acrylic weld-on and super glue were used to assemble the acrylic pieces. Finally, sandpaper was used to polish the acrylic bases of the chess pieces. Since the light from the LED’s must be visible through the chess board, we decided to use semitransparent acrylic for our physical manufacturing. The mechanical engineering’s laser cutter will be used to machine acrylic sheets into squares that will assemble to form the grid of the chess board. Acrylic panels will be used on the sides and bottom of the PCB to provide housing for the electronics and produce an aesthetic project. Laser cutting was chosen over other manufacturing options because it produces very smooth cuts and mitigates the risk of fracturing which is possible when a blade is used [24].

Societal Impact Constraints

Environmental Impact and Sustainability

With the use of a personal computer to power the Stockfish artificial intelligence engine, it was deemed not necessary to use batteries for the power supply, instead relying on power from the computer of choice. This allows the team to avoid sustainability issues with water contamination and low recycling rates associated with battery usage [20]. Furthermore, the Nippon Chemi-Con MVY-series aluminum electrolytic capacitors are given an estimated lifespan of 5,000 hours [21], and, in general, LEDs are given an estimated lifespan of 50,000 hours. Each of these components are RoHS compliant, ensuring that restricted materials including “lead (Pb), mercury (Hg), cadmium (Cd), hexavalent chromium (CrVI), polybrominated biphenyls (PBB), polybrominated diphenyl ethers (PBDE), and four different phthalates (DEHP, BBP, BBP, DIBP)” [29].

Some limitations of the board include the use of acrylic for the board surface and casing, which is not easily recyclable. From an economic standpoint, increased production will allow for the cost of PCBs replacements to decrease due to economies of scale. The use of 8 separate controllable LED strips does allow for only one strip needing replacement in the case that a single LED on a row fails.

Health and Safety

Given the nature of the project revolving around a chessboard and ai engine running on a PC, consumer safety issues are relatively minimal and not a significant challenge posed to the project. However, choking hazards do exist for the integrated magnets within each of the chess pieces for the hall effect sensors to detect. To mitigate this issue, the magnets will be recessed within each of the chess pieces and secured using a strong epoxy. The strength of this adherence will then be tested by attempting to pull the magnets from the chess pieces, modifying the magnet integration within the pieces as necessary either by changing the epoxy or piece design.

Furthermore, the acrylic top to the chessboard containing the PCB and sensors will be fastened using screws to ensure that users are not able to easily access the underlying electronics. Protection against electrical issues including but not restricted to shorts and ground faults will also be taken into consideration in the design of the PCB, additionally following the RoHS, NEMA, and IP safety standards detailed in the standards section of the proposal. The PCB will be designed using nontoxic materials to further ensure safety of the board users.

Ethical, Social, and Economic Concerns

While chess AI can do lots of good when used as a training tool, the technology has potential for harm when used to cheat. In particular, cheating in online chess has increased dramatically in the past decade, with chess.com banning on average 500 accounts per day for cheating [10]. Although our chess engine will run locally and not interface with any online API’s, the assistive chess board could still be used to cheat if a player had two chess games going simultaneously, one physical game on the assistive chess board and one online game. The player could wait for the squares to light up on our assistive chess board and then make the same move in an online game. However, the typical method that players use to cheat online is by having a chess engine running on a separate monitor, and this method is both faster and more accurate than if a cheater used our product. With this in mind, we reason that while our product could potentially be used to cheat in chess, it would be less effective than conventional cheating methods, and therefore would not lead to a significant increase in cheating. In addition, mining silica can have negative effects on the environment [25], and since hall effect sensors use semiconductors with silicon the ethical constraints on the environment must be considered, especially because our project will use 64 hall effect sensors. On the economical side of things, this device costs around $300 to manufacture. This means that economically disadvantaged people will be unable to purchase this device.

External Considerations

External Standards

Given the use of a PCB and additional electrical equipment, an enclosure following the IEC type 1 standard will need to be implemented. This will be designed to protect the user from hazards such as electric shock and protect the devices from outside particulates and light [4]. Regarding the IP safety standards, the product will need to meet the designated IP11 rating, which will “provide a degree of protection to personnel against incidental contact with the enclosed equipment and to provide a degree of protection against falling dirt” [5]. Regarding the use of a PCB, the IPC-2221 standard will be followed. This standard outlines the proper implementation of features such as impedance control, PDN bus layouts, and conductor clearance [6].

With the use of two programming languages, C for embedded design and Python for the implementation of the AI algorithm, certain coding standards are to be held in order to provide cleaner and more reliable code. For C development, the Barr coding standard will be adhered to [7], while the PEP 8 coding standard will be observed for Python development [8].

Intellectual Property Issues

Several patents exist which overlap with the underlying technologies used in our project. These include chess piece detection, on board lighting of the chessboard tiles, and artificial intelligence to dictate potential moves. With the existence of these patents, it is assumed that the assistive chess board does not have potential to be patented. As detailed under patent law in The U.S. Patent Act, “the invention must be statutory, novel, useful, and non-obvious” in order to be patentable [35]. While the piece detection approach may satisfy the requirements of being “statutory, novel, and non-obvious”, it is unclear whether it provides “useful” advantages over boards such as one patented by Bryght Labs that achieves the same effect [27]. Described below are three patents that cover material relevant to our chessboard.

One patent granted to Bryght Labs, Inc. details an "apparatus, system, and method for an electronically assisted chessboard". Under this patent there are several significant overlaps in chessboard goals, including "determining a current location on the chessboard of each chess piece", displaying "the set of valid moves available for the selected chess piece" through an "illumination system to display on the chessboard" the three first most valid moves for the current player. Each of these closely match the intentions of our board, giving a very similar purpose for piece detection and tile illumination for recommending the three most valid moves determined by an artificial intelligence actor. Player recommendations are done through, "selecting a plurality of moves for inclusion in a group of moves based, at least in part, on the skill level of the human player". While this same action is achieved by our project, it is done through the integration of Python StockFish, which is already listed as open source.

Piece detection in the Bryght Labs patent is described as "a capacitive sensing system" that "can include a capacitive sensing array in contact with the playing surface". While our system differs with the use of single hall effect sensors and saved game states to remember piece locations, it is unclear whether this implementation serves as a "useful" improvement to the capacitive sensing system in terms of energy consumption, ease of use, and piece detection interference, since "the capacitive sensing system can include sensing by row and column at an 8x8 resolution or higher resolution grid if desired". In addition, "the lighting array includes color RGB LEDs", mitigating any benefits that our use of driven RGB LED strips may provide. This patent by Bryght Labs depends on work developed in capacitive sensors and tile illumination set out in other patents and is thus a dependent claim on previous patents.

An additional patent reviewed is one for an "electronic chess game" designed such that "each playing piece is encoded in accordance with its identity, and each playing position automatically responds to the encoding when it is occupied by a playing piece" [28]. The means of response is "an electrical circuit associated with the playing position causes other positions to which the playing piece is capable of moving to be illuminated with an appropriate color". Not relying on an AI engine to determine the best move, this circuit is designed to simply show all possible moves a piece can make on the board, not necessarily the most optimal one as determined by an algorithm. This implementation requires a four-bit signal to be sent from the piece, resulting in more complex circuitry. This patent is a dependent claim building off of cited patents in digital encoding and piece detection. Our implementation has the advantage of using a single bit sensor for detecting whether a piece is present or not on a tile and then referring to previous board state as a means for discriminating pieces, providing a "statutory, novel, useful, and non-obvious" improvement to this patent.

Another patent reviewed is a chess game board in which "each of the playing pieces has different codes which can be detected by sensors". These sensors have outputs that "are connected to a signal processing device via which the course of the game is stored and/or evaluated" [26]. The purpose of this board is to "detect which figure is in which place" in a game of chess. Similar to our project, the board uses Hall effect sensors, but instead opts for an analog sensor and an analog to digital converter. The "output of the analog / digital converter is" connected via "a bus with the input of a microprocessor" to decode which piece is at a given location. To discriminate between pieces, "all black pieces or all white pieces are given differently long bar magnets, so that there is a different coding for all figures." This patent is a dependent claim building off of cited patents in magnetic piece detection and chess boards that continuously monitor piece positions. Our team's novel use of digital hall effect sensors allows for a lack of need for different strength magnets and shielding to prevent other sensors from picking up incorrect strengths from magnets on adjacent pieces, only relying on an ON/OFF signal.

Detailed Technical Description of Project

What it is

The assistive LED chess board is an educational tool for chess enthusiasts, from beginners to advanced players. Users can employ this chess board in several ways. One player can play chess against an AI engine of varying difficulties and choose to receive no recommended moves, a recommendation on every move, or recommendations upon request. Alternatively, two users can also play against each other, configuring the frequency of recommendations and AI difficulty to their desire. Due to the varying levels of AI assistance, this chess board can be used to improve the skills of beginners and professionals alike. This chess board integrates AI and human-computer interaction, allowing chess players to interact with physical pieces while learning how to improve their skills.

How it works

The positions of chess pieces are tracked by magnets and hall-effect sensors. Each chess piece has one magnet at its base. Under each chess square, there is a hall-effect sensor that senses the presence of a magnet. With a network of 64 hall-effect sensors, each square on the chess board can be monitored on whether or not a chess piece is at that location. The sensor network is scanned around 20 times a second, which allows it to recognize positional movement. For example, if a chess piece is lifted up and placed in a different position, the sensor network would recognize where the piece was lifted and where it ended up. By continuously scanning the sensors, the chess pieces can be tracked as they move around the chess board. Once the Raspberry Pi Pico determines the current position of the chess board, this data is sent to the PC over serial USB, where the validity of the position is determined. Using the GUI, the user will select their recommendation settings and the PC interfaces with Stockfish API to produce the recommended chess move for the current position. The recommendation will be converted into a matrix that indicates which LEDs need to be illuminated. The matrix will be sent to the Pico via serial USB, illuminating the LEDs of the squares involved in the recommended move. The user can make the suggested move or disregard the recommendation, at which point a new board position is detected causing the process to repeat. This process is more clearly shown in Figure 1, as we can see the 5V input source powering the LEDs and the voltage regulator powering the sensor network.

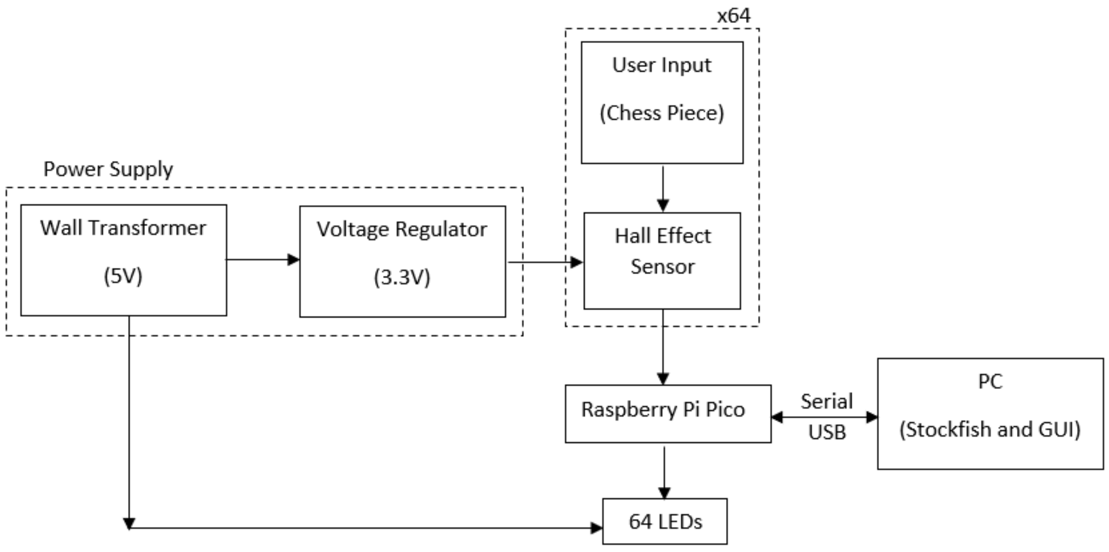


Figure 1: Block Diagram of the Full System

Hardware

The hardware system consists of two PCBs. Of the two PCBs, one acts as a sensor board, and the other one acts as a voltage regulation board which contains a Raspberry Pi Pico microcontroller. The Raspberry Pi Pico is crucial for controlling the logic for the entire system. We chose to split the boards into two distinct sections because freedfm, the manufacturers of the PCB, did not allow a PCB greater than 60 square inches. As the sensor board reached 58 square inches, it was deemed necessary to create a secondary board for the additional functionality. Figures 2 and 3 show the overall schematic of the sensor board and the voltage regulator board respectively.

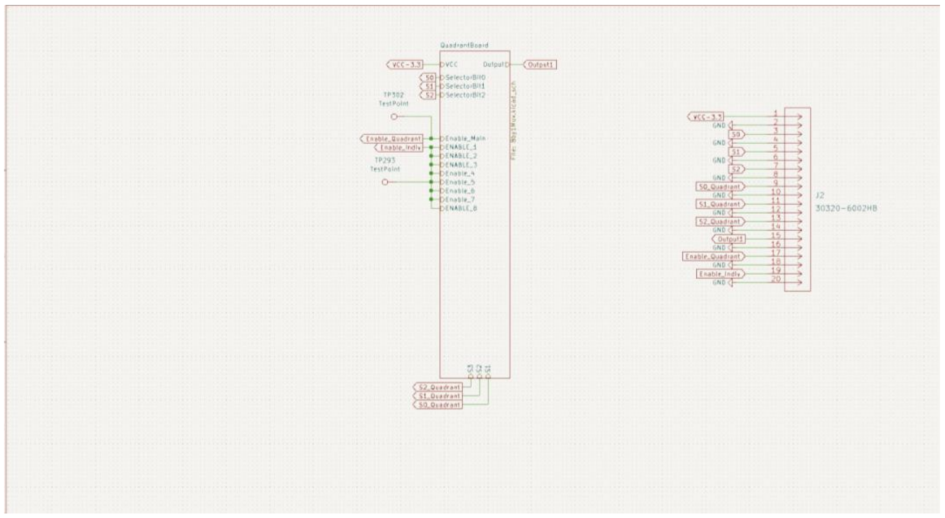


Figure 2: Top level of Sensor Board

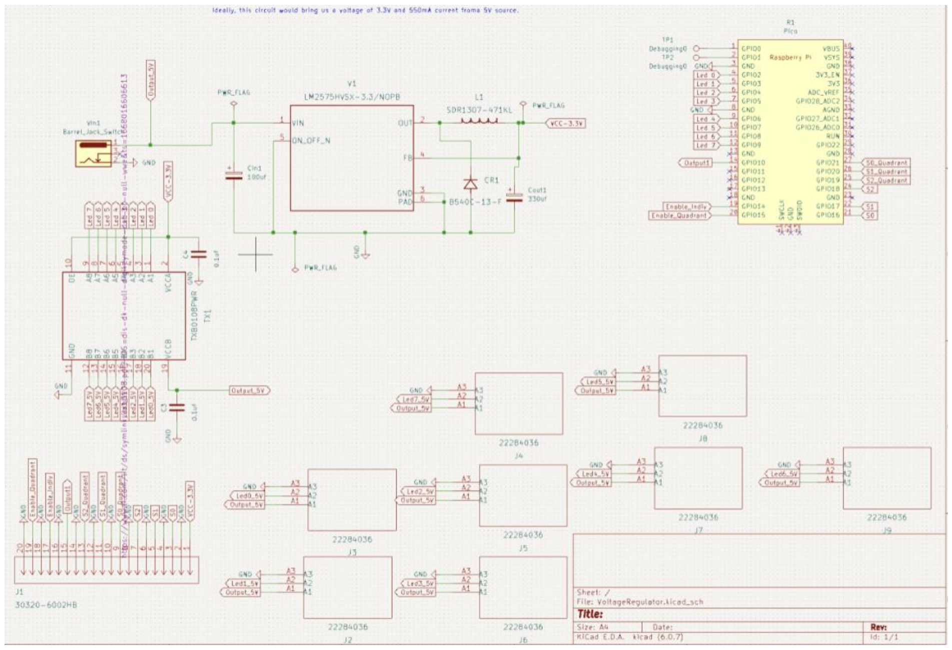


Figure 3: Top level of Voltage Regulator Board

Voltage Regulation and Microcontroller

The assistive chessboard system is powered by a wall transformer that connects to a barrel jack connector on the voltage regulator PCB. Figure 4 shows the components used in the voltage regulator system.

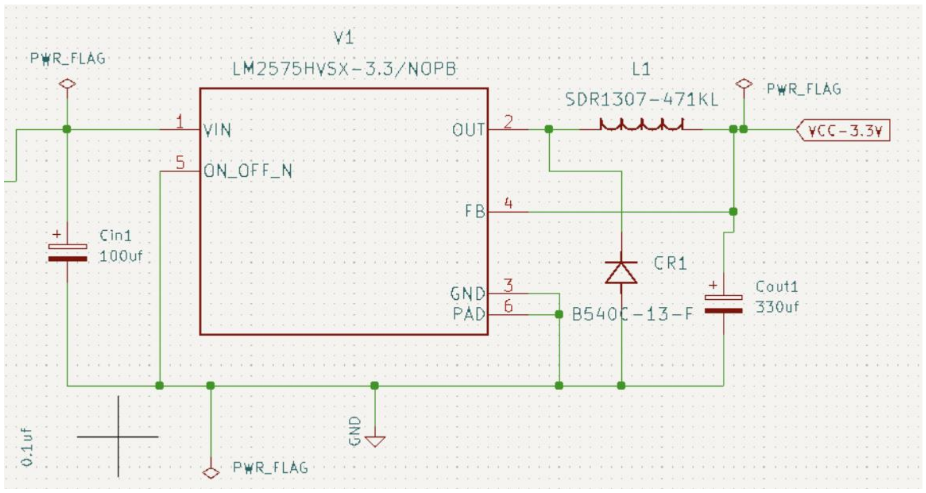


Figure 4: Voltage Regulator

Our board uses this voltage regulator to power the sensor board. It converts a 5V source into a 3.3V source required for use to power the sensor board. Furthermore, the 5V input will also be used to power the LED light strips, as they require quite a significant amount of current to power on. Table 1 will show a list of components our circuit needs to be able to supply power too.

Our board uses this voltage regulator to power the sensor board. It converts a 5V source into a 3.3V source required for use to power the sensor board. Furthermore, the 5V input will also be used to power the LED light strips, as they require quite a significant amount of current to power on. Table 1 will show a list of components our circuit needs to be able to supply power too.

Component	**Current (A)**
Hall Effect Sensors (64)	0.1024
Muxes (9)	0.01024
LEDs (64)	3.84
Total Consumption	3.953

Table 1: Current Consumption of Components

The voltage regulator, designed to take in an input of 5V and output 3.3V, can supply a maximum current of 0.200A. The maximum current consumption of the sensor board is 0.11A. This means that the sensor power requires 0.363W. The voltage regulator is operating at 61% load. The LEDs, on the other hand, will require 19.2W of power while they are on full brightness.

The microcontroller will always be connected to a computer for this project to function as intended. Therefore, the microcontroller will be powered by the computer the chess board is connected with using a Micro-B USB to USB-A cable.

Sensor Board

The hall effect sensors were used to detect if a chess piece was on the given tile. These sensors contained three pins: VCC, Vout, and Ground. The TCS40DPR,LF hall effect sensors we chose were omnipolar, digital hall effect sensors since we only needed to verify whether a magnet was on the tile or not. The hall effect sensors allowed for a voltage of 3.3V, which allowed the hall effect sensors to output into a mux without needing to step the voltage down or step the voltage up. This hall effect sensor worked well with our project, as knowing the polarity of the magnet was not a necessity. Furthermore, as the hall effect sensors are digital, we do not need to worry about the strength of the magnet. As long as the magnet is close to the sensor, the hall effect sensor would work as intended. One major concern about the hall effect sensors was the ability to sense the magnets of their nearby neighbor. To prevent this issue from occurring, we spaced each hall effect sensor 1 inch apart from each other. We also chose smaller, weaker magnets as an extra precaution.

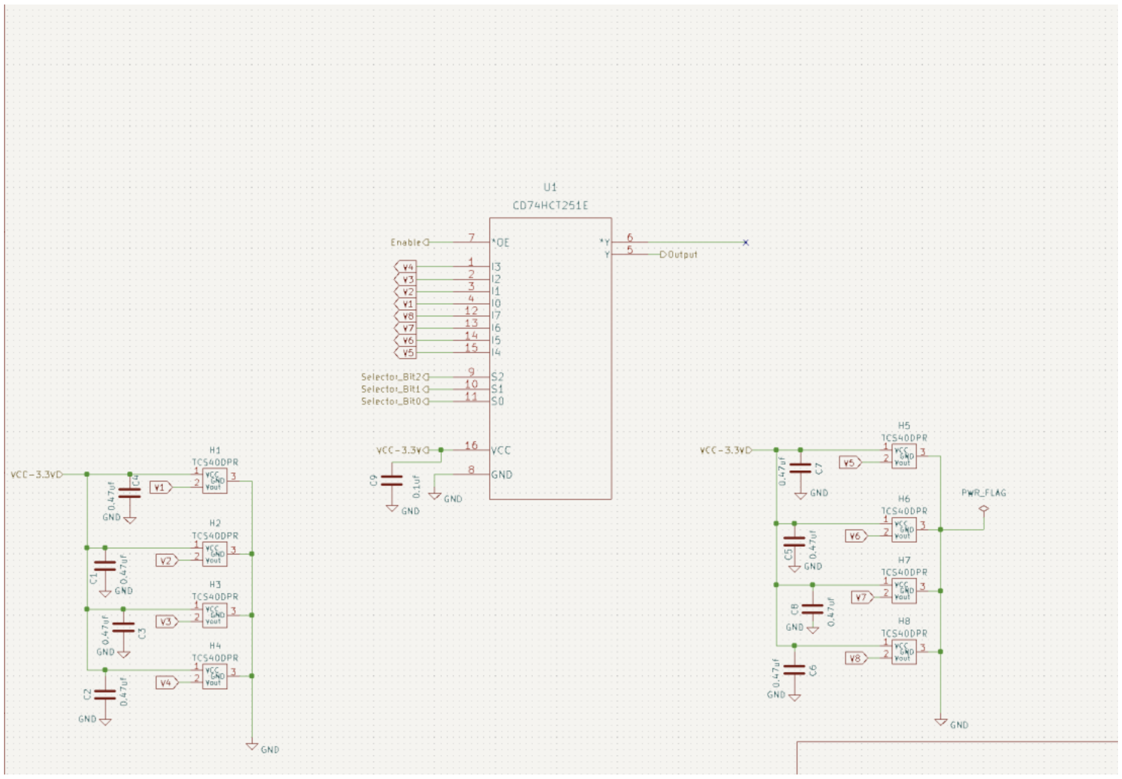


Figure 5: 8 Hall Effect by 1 Mux schematic

In figure 5, we can see that the hall effect sensors used would send their signal straight into a CD74HCT251E multiplexor. This multiplexor is able to handle the 3.3V signal given by the hall effect sensors and safely pass the signal onto the Raspberry Pi Pico. We chose an 8 by 1 digital multiplexor because it was cost effective and allowed us to sense several tiles with one component. As the hall effect sensors and multiplexors used in the project are digital, Furthermore, the signal that was passed through to the multiplexor is a digital signal, which would make the use of an analog multiplexor a wasted effort. Using the circuit from figure 5, we can copy that circuit and use the output of eight multiplexors as an input to a central multiplexor. This central multiplexor's output would subsequently be used as an input to a raspberry pi pico for sensor data collection. This dynamic is more closely seen in figure 6.

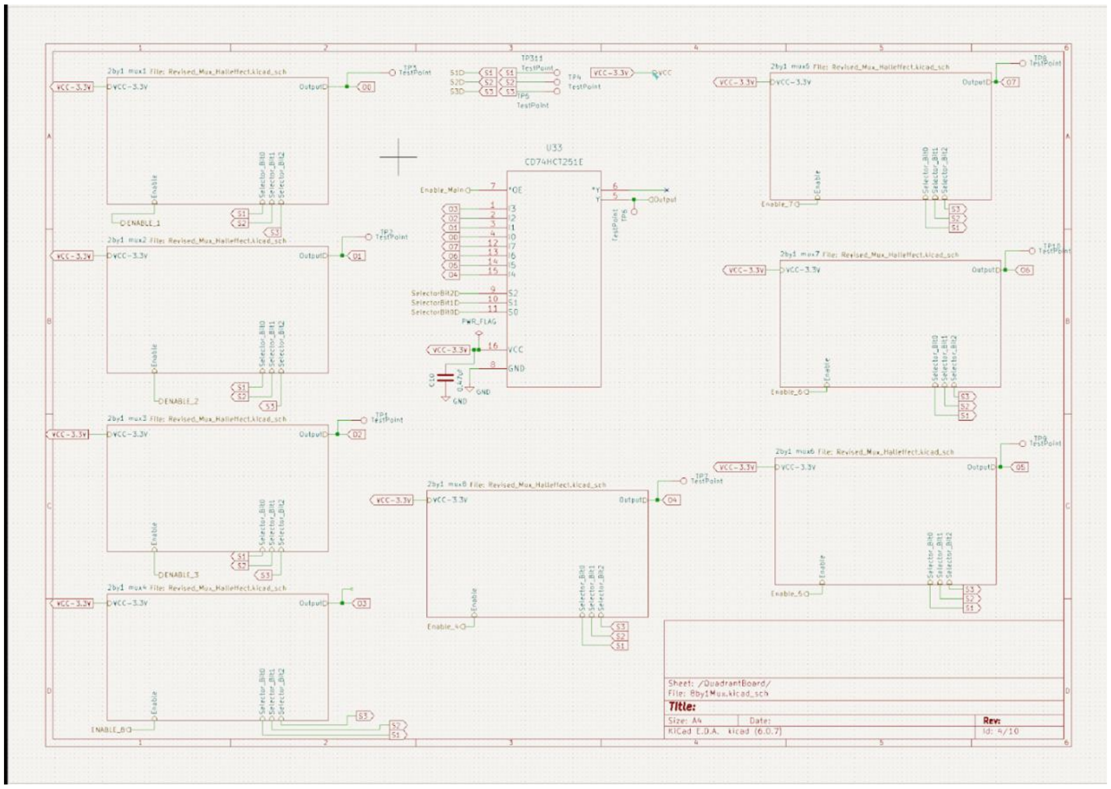


Figure 6: 8 Muxes by 1 Central Mux schematic

Board Layout:

As mentioned previously, we needed to design two boards for our project. Figure 7 showcases the sensor board layout. One major concern, while creating the traces of the board, was to allow the power rails to have many connections with each other. Allowing several connections between the power rail and the ground rail will allow current to flow in all directions and reduce inductance in the wire. An important factor in our design was to reduce inductance in the wire so that the hall effect sensors would not pick up on that magnetic field. The hall effect sensors are spaced 1 inch apart from each other. Every 8 hall effect sensors will be connected to a multiplexor, which in turn will be connected to a central multiplexor. This helped reduce the number of pins needed for input sensing, while allowing the GPIO pins for the Raspberry Pi Pico to be used for other components in the system. As it was important to make sure that the LEDs and the hall effect sensors are under each tile, we needed to make sure that the test points were soldered on the bottom side of the board so that they do not interfere with the pathing of the LEDs.

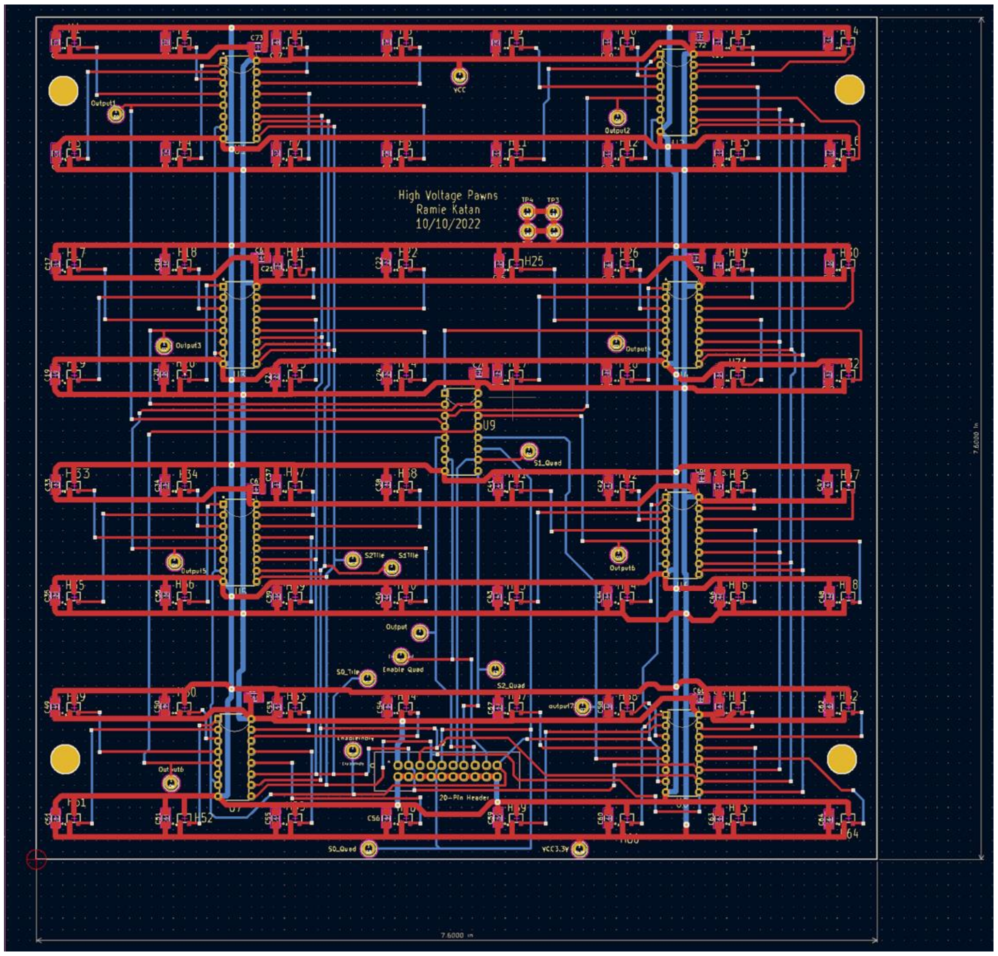


Figure 7: Sensor Board Layout

The next board that needed to be created was the voltage regulator. The layout of the voltage regulator board is shown in figure 8. There are multiple critical components in this board that ensure the functionality of the device. Specifically, this portion of the project is very important as it is the power supply for the sensor board and the LED strips. The voltage regulator board also contains the housing for the microcontroller, which will send signals to the LED light strips and the sensor board.

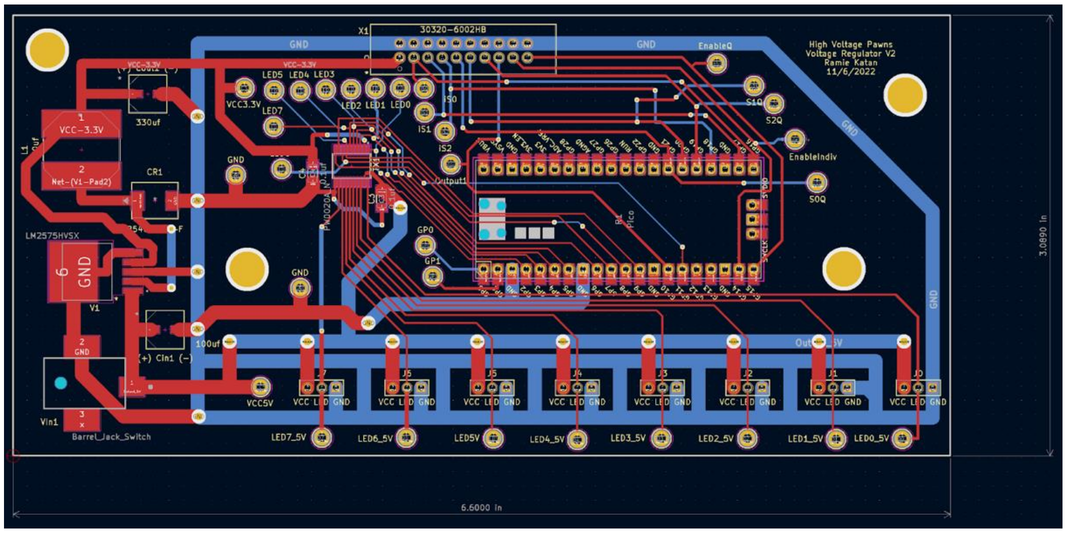


Figure 8: Sensor Board Layout

In the voltage regulator board, we made the traces large enough to handle 4A of current at 5 volts. Specifically, these 100 mil traces are designed to support all the LEDs at full brightness as well as convert the input of 5V to a safe supply voltage that the sensor board can use.

Voltage Level Translator

One critical aspect of the project was understanding how to implement the LEDs. Through countless concept designs, we arrived at the idea of using LED light strips. Each LED contains a mini controller called “WS2812B”. This controller would allow us to easily interface between the LEDs and the Raspberry Pi Pico. Unfortunately, one major problem arises if we use that specific LED controller. The LEDs and the WS2812B controller require the use of a 5V VCC and a 5V data line. Unfortunately, the Raspberry Pi Pico is only capable of supplying a maximum voltage of 3.3V. This means that we will need to use a voltage level translator that will boost the 3.3V signal coming from the Raspberry Pi Pico and convert it into a signal of 5V that is usable for the WS2812B.

Firmware

The Raspberry Pi Pico was the microcontroller used for this project. It was selected due its ability to run on FreeRTOS, interfacing with GPIO pins, integration with the WS2812B LED strip, and built-in serial module. The microcontroller was programmed in C. The FreeRTOS kernel was used to configure interrupts and run tasks concurrently. Three tasks were used: sensor network processing, LED interfacing, and serial communication.

Sensor Network Processing

One FreeRTOS task was used to scan the hall-effect sensors repeatedly (around 20 times a second) and write the board state to the PC using serial USB communication. The sensor network board had 64 hall-effect sensors that multiplexed to one GPIO using nine 8:1 digital multiplexors. There are 8 lower-level multiplexers, and each multiplexer reads the sensors of 8 chess tiles. The outputs of the 8 lower-level multiplexers are inputs to one higher level multiplexer. This allows us to multiplex 64 hall-effect sensors to one GPIO pin that will be used to read sensor data. Table 2 shows the mapping of multiplexers to physical chess squares.

Row 8	0.0	0.1	0.4	0.5	1.0	1.1	1.4	1.5
Row 7	0.2	0.3	0.6	0.7	1.2	1.3	1.6	1.7
Row 6	2.0	2.1	2.4	2.5	3.0	3.1	3.4	3.5
Row 5	2.2	2.3	2.6	2.7	3.2	3.3	3.6	3.7
Row 4	4.0	4.1	4.4	4.5	5.0	5.1	5.4	5.5
Row 3	4.2	4.3	4.6	4.7	5.2	5.3	5.6	5.7
Row 2	6.0	6.1	6.4	6.5	7.0	7.1	7.4	7.5
Row 1	6.2	6.3	6.6	6.7	7.2	7.3	7.6	7.7
	Col A	Col B	Col C	Col D	Col E	Col F	Col G	Col H

Table 2: Hall-effect sensor network multiplexing

In each chess square, the numbers indicate the input of the multiplexers that allow the square to be selected. The format is as follows: <Higher level mux input number>. <Lower-level input mux number>. For example: to read sensor D6, the higher-level mux input 2 must be selected and the lower-level mux input 5 must be selected. A function sets the select bits to read the desired input number from the multiplexer.

The sensor network is read by iterating through all 64 sensors and storing the values in a buffer of 64 chars. Since the hall-effect sensors are active low, the GPIO pin reads the voltage and determines if a magnet is present on that square. A “1” or “0” is stored if a magnet is present or not present respectively. The “0” or “1” character is written to a specified location in the buffer relative to the physical chess square location. In the buffer, each consecutive group of 8 characters represent one row of the chess board. The first 8 characters correspond to row 8 of the chess board (from column A to column H). The next 8 characters correspond to row 7 of the chess board (from column A to column H) and so on. The second to last 8 characters correspond to row 2 of the chess board (from column A to column H). The last 8 characters correspond to row 1 of the chess board (from column A to column H). Once the buffer is full of sensor data, it is sent to the PC via serial USB for further processing by the FSM.

LED Interfacing

The Raspberry Pi Pico interfaced with the WS2812B RGB LED strips using the Pico SDK's Programmable I/O (PIO). The PIO protocol was chosen due to its straightforward integration with the WS2812B controller and the Pico's GPIO pins. It is also valuable because it uses direct memory access to send data to the LED strips, reducing CPU computation and overhead. Eight LED strips were used, one for each column on the chess board, and each strip has eight LEDs. Eight channels were set up using PIO to set the LED colors. The colors were set by inputting a series of hexcodes to the GPIO pins. The hexcodes would get sent as a pulse-width modulation (PWM) signal. For example, a long high followed by a short low would be a logical 1 and a short high followed by a long low would be a logical 0. The hexcode, which has 24 bits, would get sent as PWM through the strip, setting the string of LEDs to the desired color combination.

Several helper functions were used to scale the LED driving code. Ultimately, the LED interfacing was abstracted to a function that would receive a string of length 64 and drive the LED based on the color that was at the particular index. Table 3 shows the mapping of the string input and the corresponding LEDs. The number in each chess square shows the index in the string that sets the particular LED. The char value at the particular index in the input string would determine which hexcode to display on the LED.

Row 8	63	55	47	39	31	23	15	7
Row 7	62	54	46	38	30	22	14	6
Row 6	61	53	45	37	29	21	13	5
Row 5	60	52	44	36	28	20	12	4
Row 4	59	51	43	35	27	19	11	3
Row 3	58	50	42	34	26	18	10	2
Row 2	57	49	41	33	25	17	9	1
Row 1	56	48	40	32	24	16	8	0
	Col A	Col B	Col C	Col D	Col E	Col F	Col G	Col H

Table 3: LED mapping

For more abstraction, a Python library was created to product the format needed to drive LEDs. For example, a function would receive a recommended move from stock and convert that to a format that would illuminate the appropriate LEDs. A similar function was implemented for displaying an illegal move. In addition, an LED start up sequence was created to run through RGB colors through every LED to debug the LED matrix and check for fault connections.

Serial Communication

Serial over USB was used to communicate between the Raspberry Pi Pico and the PC. The Pico SDK can configure standard output (stdout) to be set to serial USB, sending print statements to the PC. A Python serial module was used on the PC to read and write to the serial session. The sensor network data was sent from the microcontroller to the PC. The LED configuration was sent from the PC to the microcontroller. Since the LED configuration is a buffer that the PC writes to and the LED driving function reads from concurrently, a mutex and condition variable were used to synchronize these concurrent tasks. The FreeRTOS synchronization primitives were used to implement this.

Software

FSM

The software is composed of three files: FSM.py, LED.py, and ChessInterface.py. FSM.py is responsible for interfacing with the Raspberry Pi Pico through USB serial connection, executing algorithms to keep track of the state of the chess game, and interfacing with the graphical user interface. Figure 9 shows this algorithmic flow. FSM.py has two primary functions: next State Function and outputFunction. NextStateFunction contains logic that tells the finite state machine when it should transition between possible states. For example, when the Stockfish AI detects that a legal chess move has been made, the checkValid state transitions into the showRecommendation state. The outputFunction contains distinct outputs for each possible FSM state. In the transition state, the output is reading data from the Pico, in the checkValid state, the output is determining if a legal chess move has been made, in the showRecommendation state, the output is writing a signal to the Pico to illuminate LED's, in the error state, an error signal is sent to the Pico, and finally in the end state the game is determined to be either checkmate or stalemate. Figure 10 shows a high-level diagram of the FSM. LED.py contains helper functions that convert chess moves in algebraic notation into a signal that can be interpreted by the Pi Pico. For example, if the Stockfish AI recommends the move "e2e4", the function in LED.py takes that string as input and returns a string of length 64 indicating which LEDs on the chess board should be illuminated.

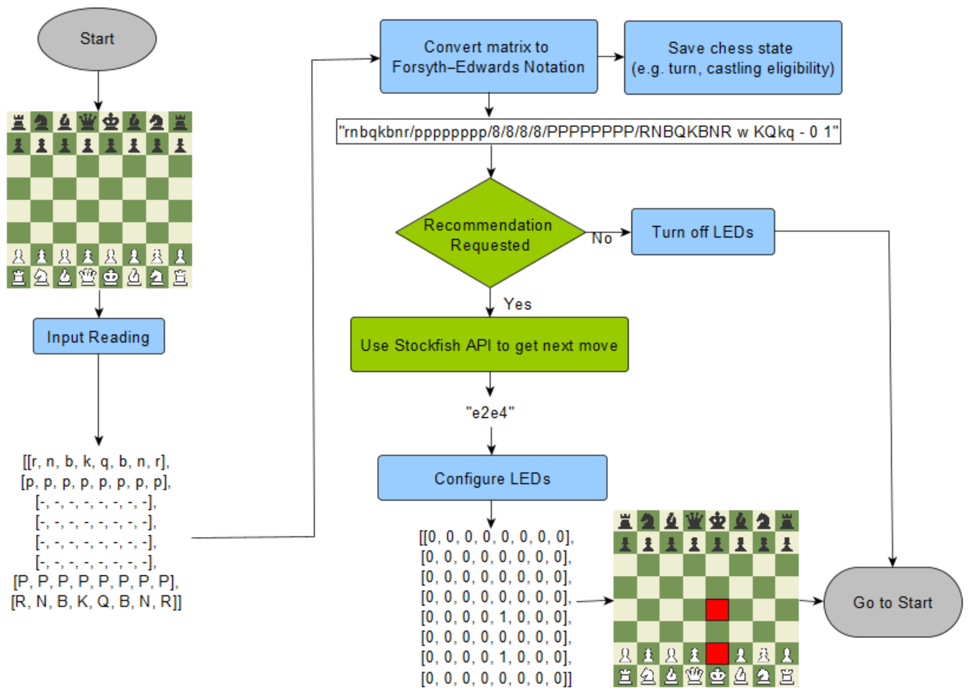


Figure 9: Flowchart of Algorithm

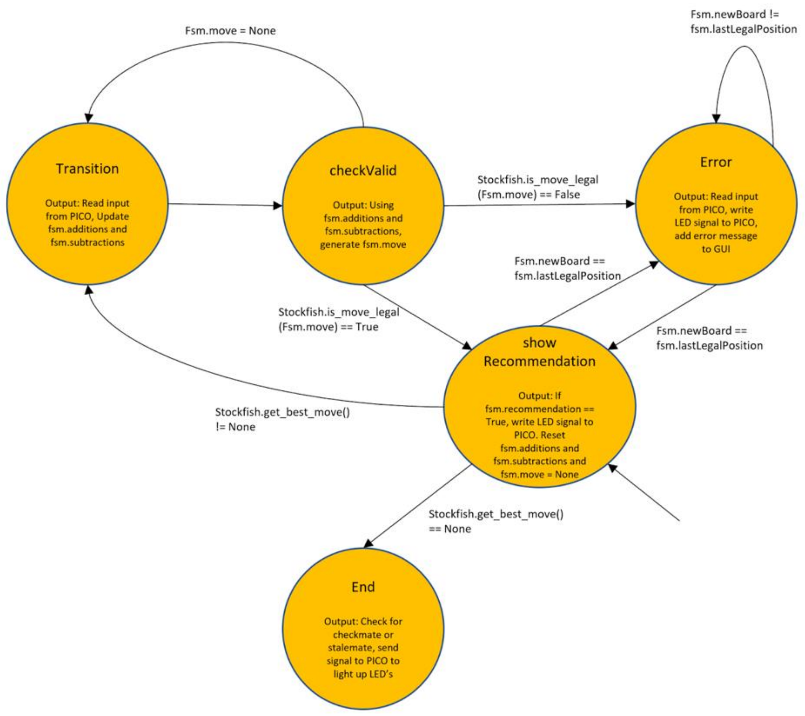

Fsm.move = None
Fsm.newBoard != fsm.lastLegalPosition
Transition
Output: Read input from PICO, Update fsm.additions and fsm.subtractions
checkValid
Output: Using fsm.additions and fsm.subtractions, generate fsm.move
Stockfish.is_move_legal (Fsm.move) == False
Error
Output: Read input from PICO, write LED signal to PICO, add error message to GUI
Fsm.newBoard == fsm.lastLegalPosition
Stockfish.is_move_legal (Fsm.move) == True
Fsm.newBoard == fsm.lastLegalPosition
show Recommendation
Output: If fsm.recommendation == True, write LED signal to PICO. Reset fsm.additions and fsm.subtractions and fsm.move = None
Stockfish.get_best_move() != None
Stockfish.get_best_move() == None
End
Output: Check for checkmate or stalemate, send signal to PICO to light up LED's

Figure 10: FSM Diagram

GUI

The FSM and embedded device are controlled by the GUI. The ChessInterface.py file contains all of the software for the graphical user interface. The GUI was programmed in Python 3 with the Tkinter library and the CustomTkinter library to improve the user interface (UI). The GUI gives the user the ability to modify a range of settings including AI strength, AI depth, number of recommendations, clock time, and starting position. All these parameters modify objects running in FSM.py in real time such as Stockfish engine parameters, board state, and LED recommendations. Figure 11 shows the starting screen of the GUI where users can configure these settings. On this frame, the user can begin the game using the starting chess position. When clicking the "Start" button the program will verify that the chess board is set up correctly. If a piece is out of place, an error message will be displayed alerting the user to fix it.

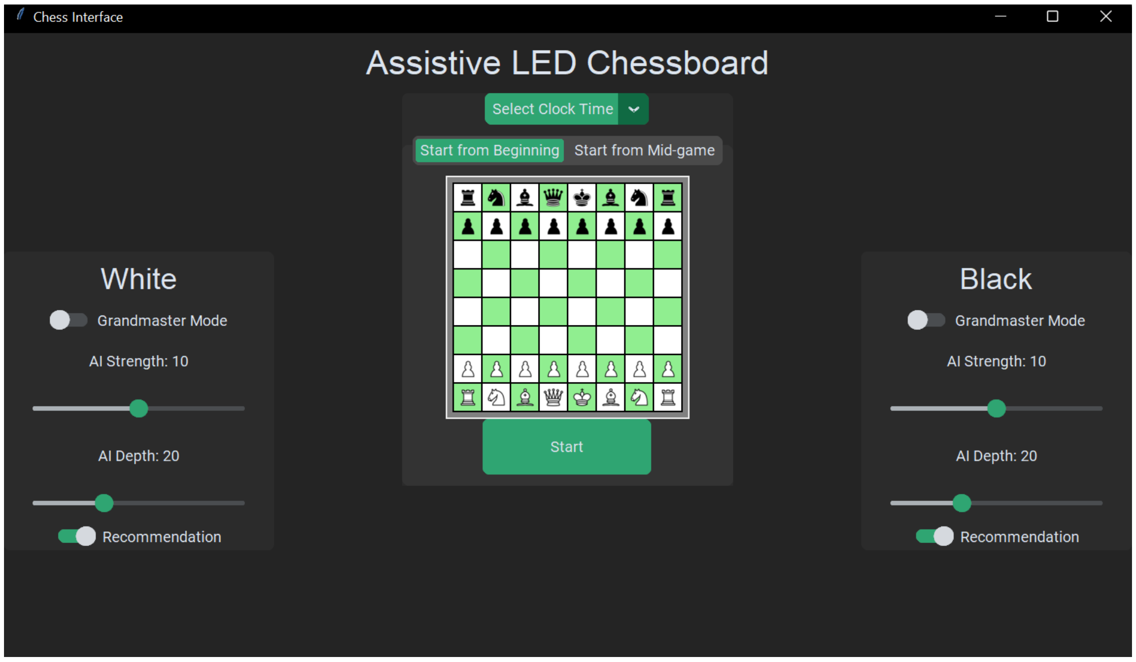
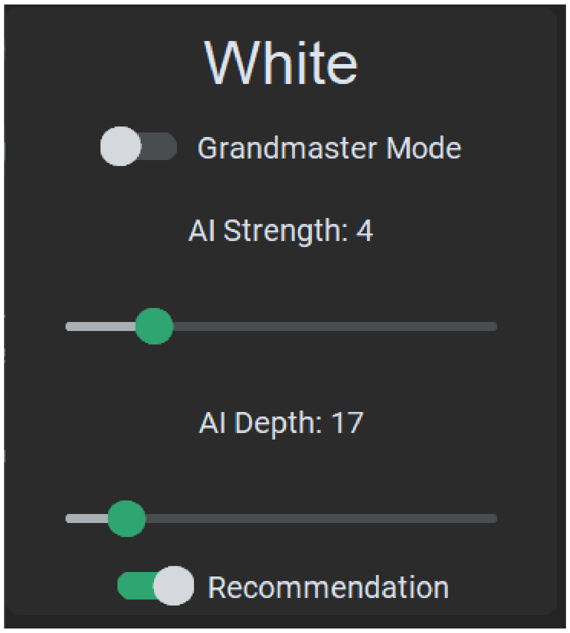


Figure 11: GUI starting screen

There are two modes for recommendations. The default mode is shown in Figure 12. It allows the user to configure the Stockfish AI strength, AI depth, and turn on or turn off a recommendation. In default mode, the user can receive at most one recommendation, but it can be in any AI strength.


White

Grandmaster Mode

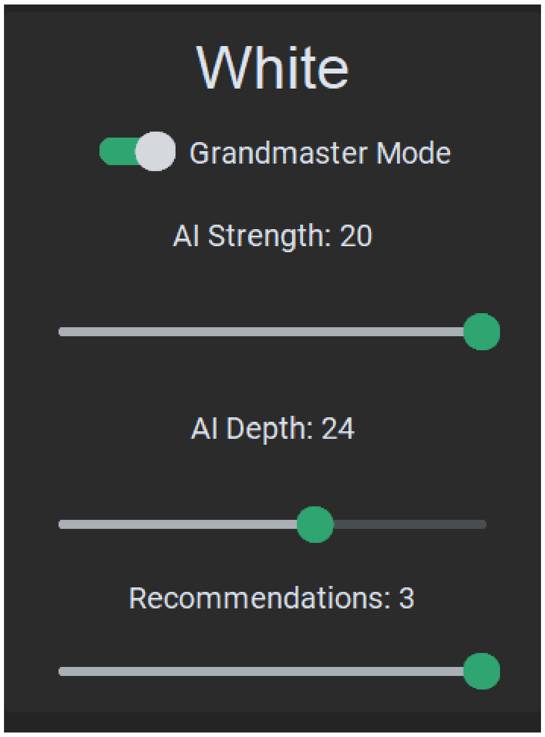
AI Strength: 4

AI Depth: 17

Recommendation

Figure 12: Default Recommendation Mode

The other recommendation mode is Grandmaster mode, which essentially shows the recommendations that a grandmaster would suggest at this point in the game. In this mode, the Stockfish AI strength is fixed at the highest level, but the AI depth and number of recommendations are configurable. This mode is shown in Figure 13.


White

Grandmaster Mode

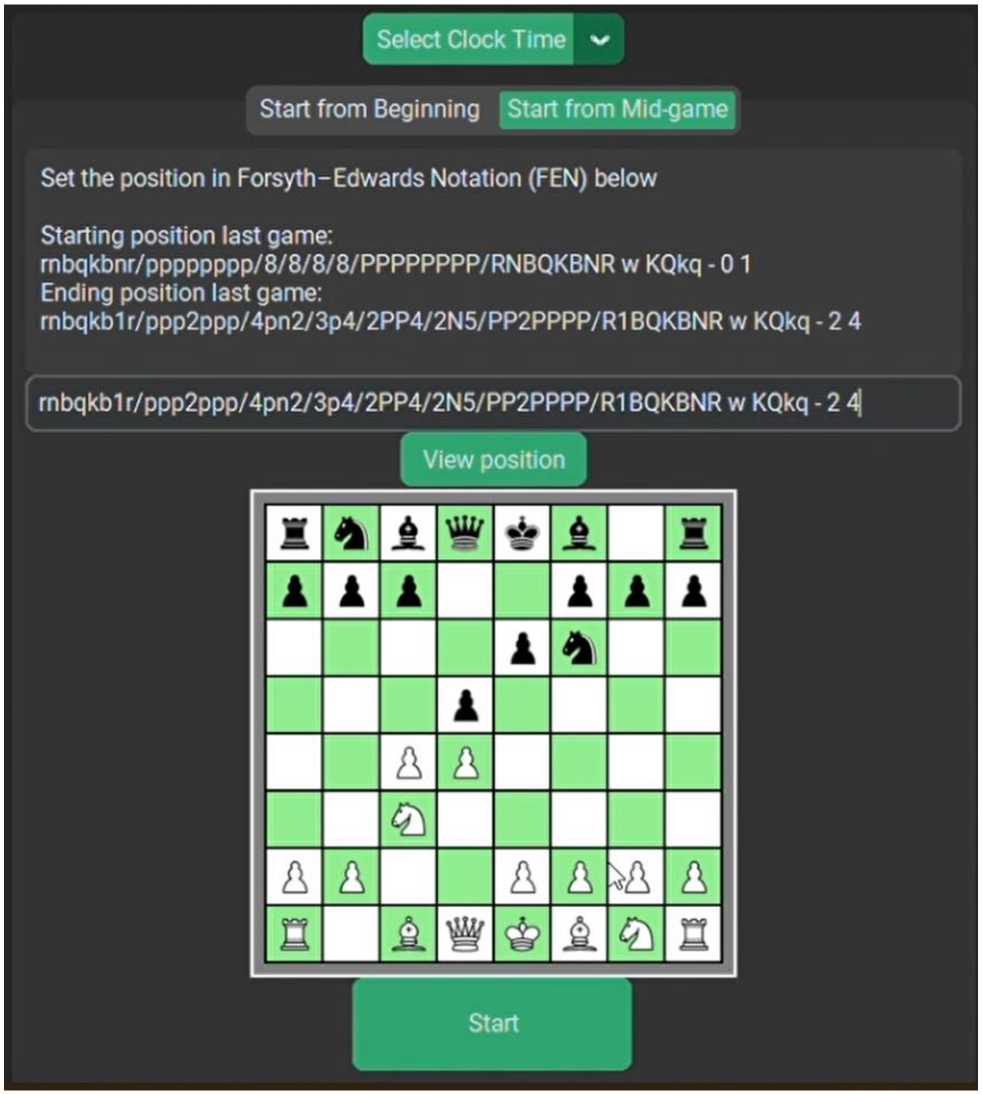
AI Strength: 20

AI Depth: 24

Recommendations: 3

Figure 13: Grandmaster Recommendation Mode

The GUI also allows users to start from a custom, mid-game position. In addition, the user's have the option of starting a game from the starting position of their last game or the ending position of their last game. Figure 14 shows this screen, which shows these two positions. If the user wants to pick up where they left off, they can simply copy and paste the position into the textbox and start the game. The player can also press the "View position" button to see if the supplied position matches where they wish to start.



Set the position in Forsyth–Edwards Notation (FEN) below

Starting position last game:
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
Ending position last game:
rnbqkb1r/ppp2ppp/4pn2/3p4/2PP4/2N5/PP2PPPP/R1BQKBNR w KQkq - 2 4

rnbqkb1r/ppp2ppp/4pn2/3p4/2PP4/2N5/PP2PPPP/R1BQKBNR w KQkq - 2 4

Figure 14: Starting from custom, mid-game screen

Users can select the time limit they wish. The time selected represents the time allocated from each player, not the total time. Figure 15 shows the clock options.

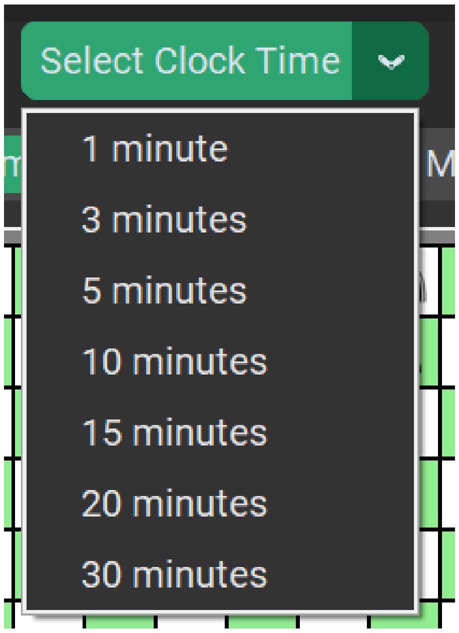


Figure 15: Clock options

The chess clock automatically toggles and counts down the player that needs to make the move. Instead of implementing a timer that the users must manually press, this chess clock interacts with the FSM to read the active player and makes the experience seamless. When a player's clock is active, it turns green and when it is inactive it turns gray. There is also a pause button which the players use to pause and resume the game, stopping and starting the timers. Figure 16 shows this feature.

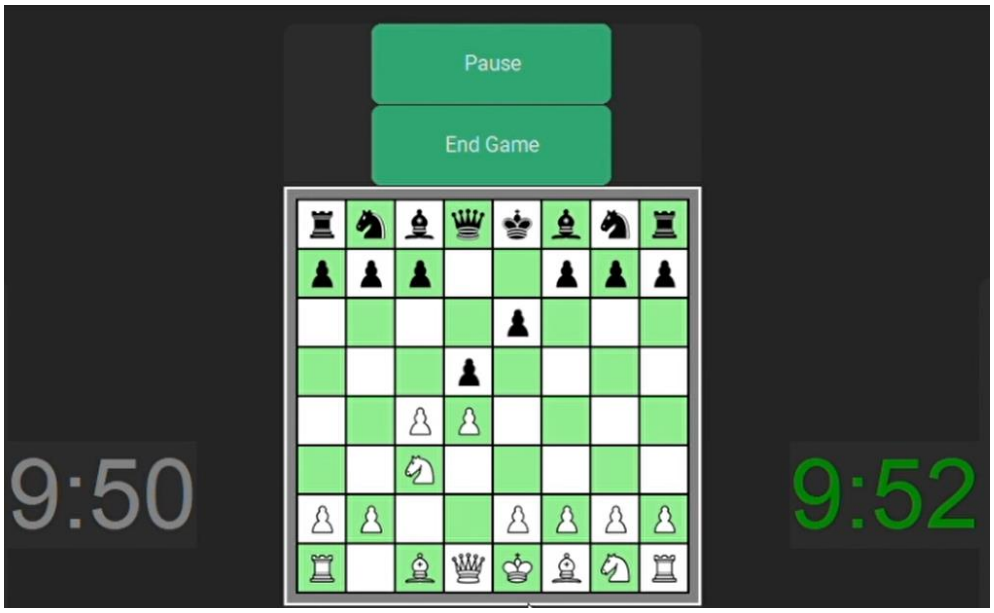


Figure 16: Chess Clock

Mechanical:

CAD Design

The CAD files for the acrylic chess board were designed using a software program called Solidworks. Initially, the design started off by creating the CAD of the top surface of the chess board box. The dimensions of the top surface of the chess board are 10 inches x 10 inches. The top surface includes a grid of squares with circles extruded in the center, as well as the alignment of text on the border to indicate the rows and columns. To ensure the letters are visible on the outside of the box when manufacturing, we will be using laser cutting to slightly etch the letters onto the board. Figure 17 shows the CAD Assembly of the top of the surface with the letters on the top surface as well as the circles extruded in the center. This will allow us to set the pieces perfectly in the center and let the hall effect sensors pick up the magnetic field emitted by the magnetic.

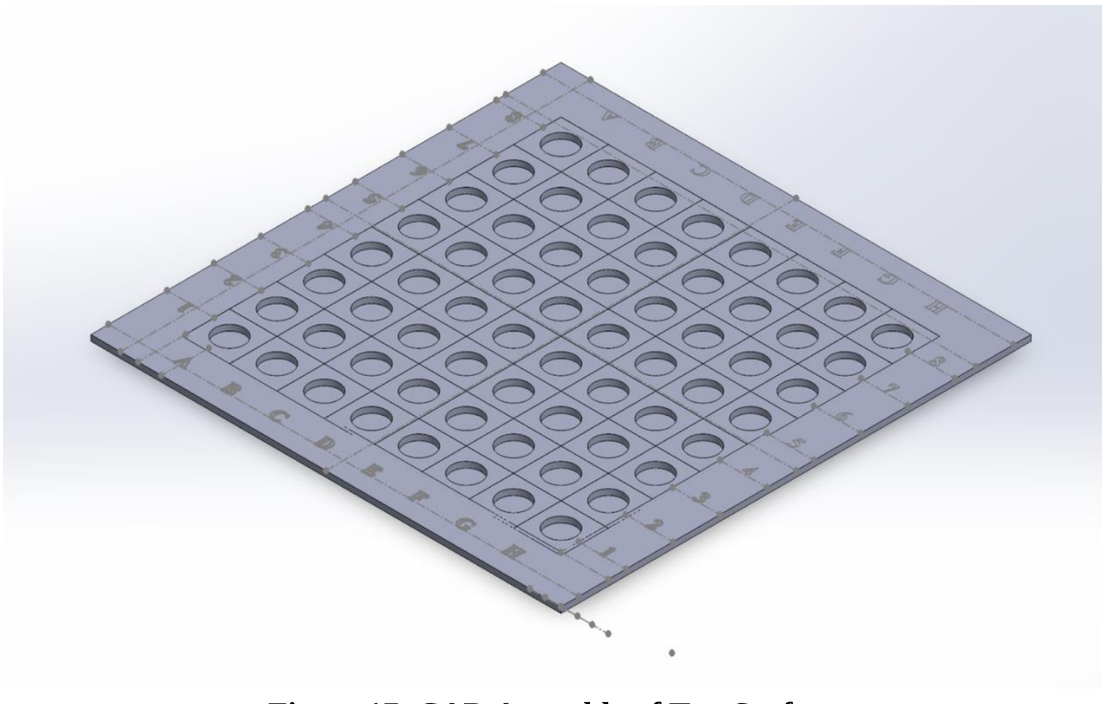


Figure 17: CAD Assembly of Top Surface

After the top surface was designed, the side panels and base were created to fit around the top. The front and the back of the chess board shows where each piece should be placed on the chess board. For example, on the left and right side of Figure 18, we can see the icon of the rook. This gives us an easy way to orient the pieces for those who are new to chess.

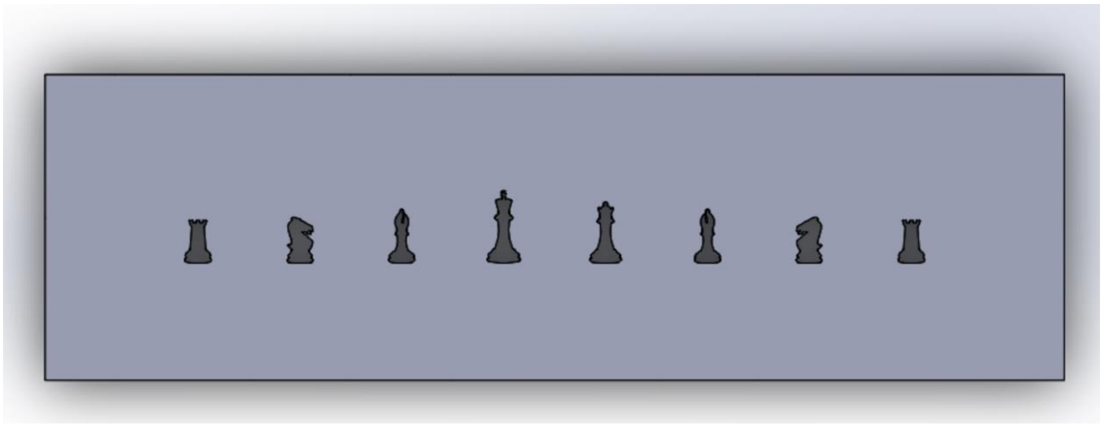


Figure 18: CAD File of Front and Back of the Chess Board

In Figure 19, we can see the left of the board with two holes. The square hole allows us to plug the micro-USB cable from the PC to the Raspberry Pi Pico. On the other hand, the circular hole will allow a passthrough for the power supply barrel jack and seamlessly connect power into the box.

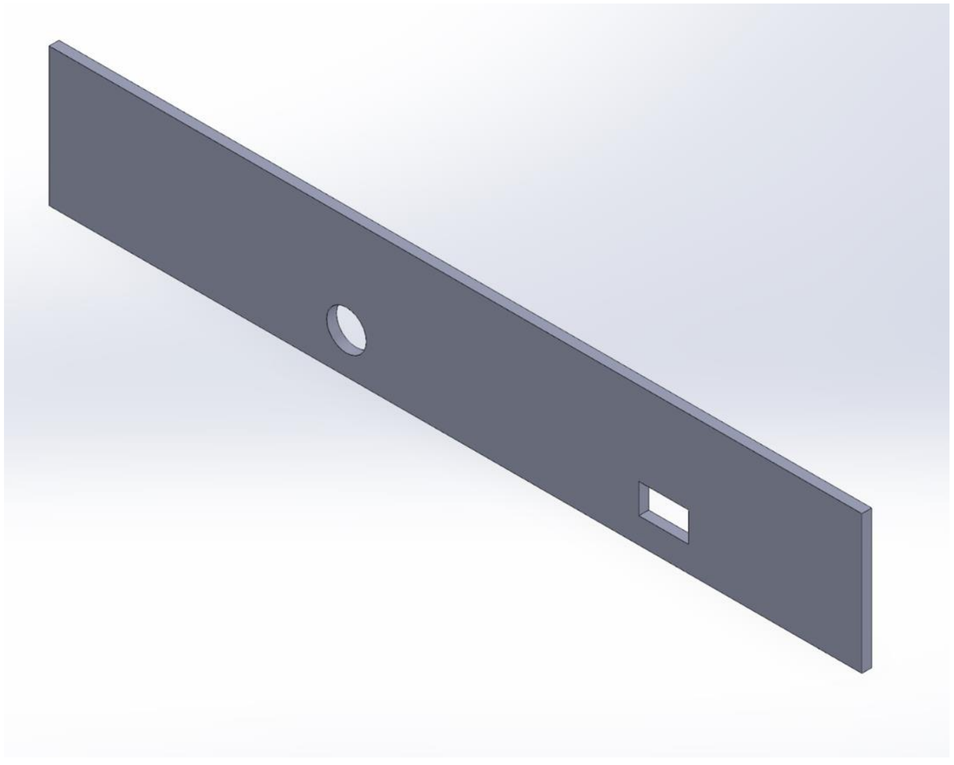


Figure 19: CAD File of Input/Output Ports

The last CAD file is similar to the previous figure, however there are no cutouts in the design. This will act as the final wall in the encasing for the PCBs and wiring.

Assembly

Acrylic sheets were utilized when manufacturing the board. During the process of manufacturing, a laser cutter was used to create fine cuts on the pieces. The laser cutter was strong enough to cut through the entirety of the acrylic. As shown in the CAD, we have circular cutouts for the surface of the chess board. This will allow the chess pieces to smoothly fit within the circular cutout. This ensured that the hall effect sensors would be able to read the pieces everytime the piece was put back down. We needed to create a hole large enough to fit the magnet and hold it in place, so the magnet does not fall off in the middle of the game as shown in Figure 20.

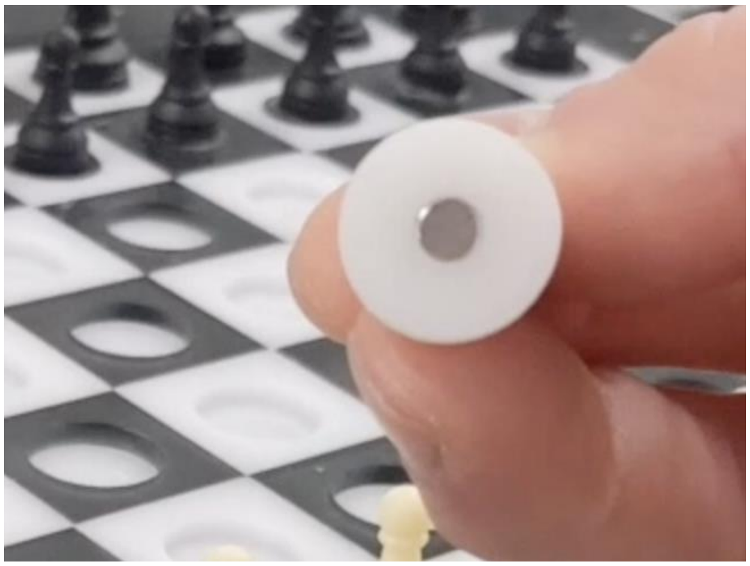


Figure 20: Ring Around the Magnet

Most of the laser cuts went all the way through the material, but the numbers and letters around the border were engraved instead. We wanted to use the laser to etch the letters onto the board to give a seamless look to the surface design of the chess board. Figure 21 shows a more detailed look into how the acrylic sheet was cut for gluing in the later steps.

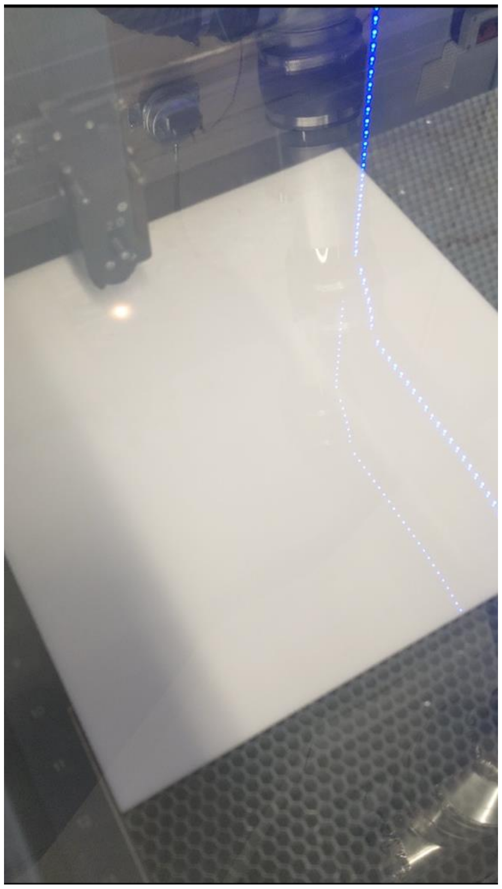


Figure 21: Laser Cutting Acrylic Sheets. The Machine can be found in the M.I.L.L of the Mechanical Engineering Building.

Once all the pieces were cut, acrylic weld-on and super glue was used to assemble the pieces together. Each cutout needed to be glued together on the surface. Figure 22 shows the assembly of the chessboard grid. After all the pieces are glued together, we needed to create a second layer of acrylic for the chess pieces to lay on the surface.

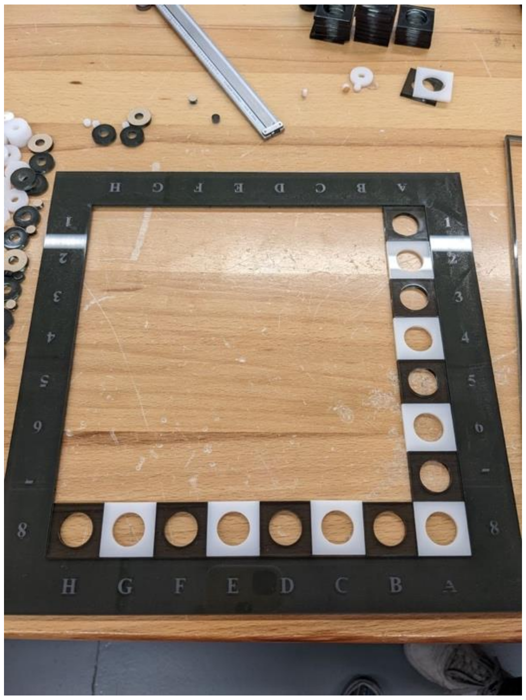


Figure 22: Assembling the Chessboard Grid

Figure 23 shows the finalized surface of the chess board. As mentioned before, there is an extra layer of acrylic underneath the surface that the chess pieces are on top of. To give it a clean look, we decided to make the bottom of the layer of the surface in white, as the LEDs can visibly shine through very clearly. Since the top layer was finished, it was not time to move onto creating the other sides of the chess board.

Figure 23: Finalized Chess Top Surface

Figure 24 allows us to see a much clearer picture of a finalized version of the chess board. Now that the encasing has been assembled, it was very important to test the system to make sure that both the LEDs are shining through clearly, and that the hall effect sensors have been able to sense the magnetic field from the magnets we chose previously.

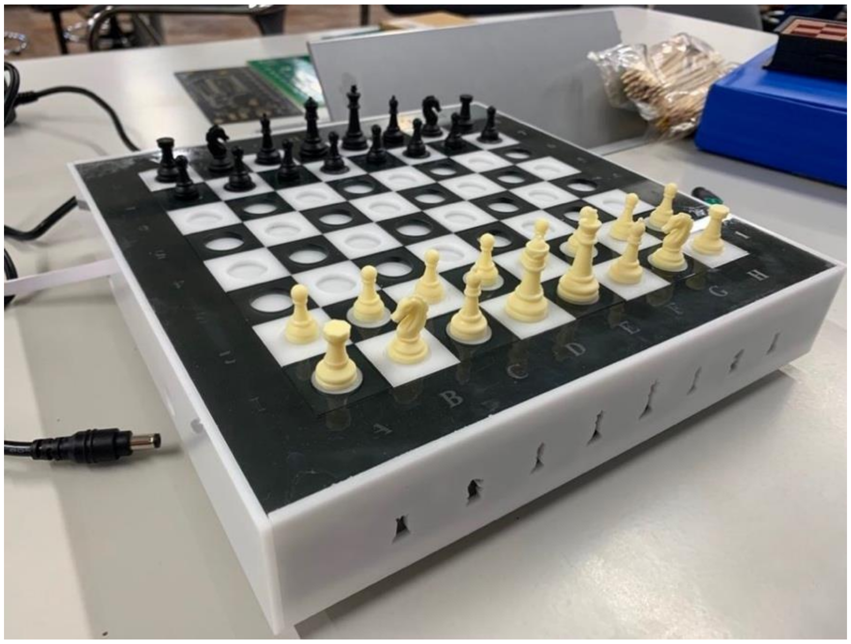


Figure 24: Finalized Chess Top Surface

Once we were ready to start testing, we plugged in all the cables needed to start the boards and input the code into the Raspberry Pi Pico. Figure 25 shows us a clean look of the LED chess board working in the finalized state.

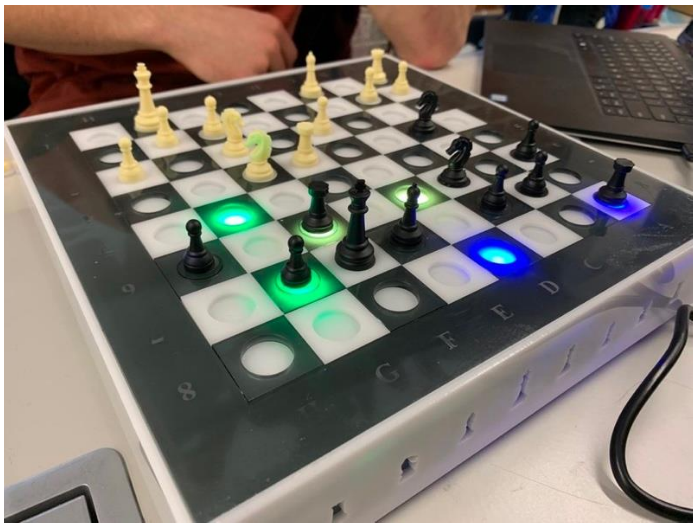


Figure 25: Finalized Chess Board

Hardware Design Modifications

In our initial hardware design, we were planning to have four hall-effect sensors per chess square and use binary encoding to detect check pieces. This would allow us to identify each chess piece using hall effect sensors alone and not rely on the previous position. However, due to budgetary constraints, the amount of money needed to manufacture the PCBs and soldering the parts would cost more than the entirety of the $500 budget. Fortunately, we were able to downscale the size of the project to include one hall effect sensor per square. This not only reduced the number of components needed, but it also reduced the number of PCBs needed for manufacturing.

One other hardware modification we needed to change was the power supply and voltage regulator levels. Initially, we were planning to use a 12V power supply and a 12V-3.3V step down switching node regulator. The 12V power rail was used initially because the LED controllers needed a 12V supply voltage. However, we quickly realized that the 12V LED strip was not suitable for the project since we realized the LED strip was only addressable by every three LEDs. We decided to use the 5V LED light strip since it was individually addressable which worked correctly. This required us to change the voltage regulator to accommodate a 5V power supply rail rather than the 12V power rail. Due to the changes in the amount of current required for the sensor boards, the amount of current supplied by the voltage regulator was reduced by a factor of four. This meant that we needed to change the inductor in the circuit as well as the voltage regulator.

Lastly, one hardware modification we included on the voltage regulator board was an addition of a logic level translator. The Raspberry Pi Pico supplied 3.3V logic and the LED strip takes 5V logic. In our initial testing, the Pico was able to supply the data line of the LED strips without a logic level translator, but we noticed the LEDs flicker at times, especially on the first LED of each strip. We modified the second revision voltage regulator board to include a logic level translator that converted the 3.3V being supplied by the Raspberry Pi Pico into a 5V data signal into the LED controller for the LED light strip. Once the chip was added, the LED flickering issues were mitigated.

Firmware and Software Design Modifications

In our initial plan to implement the firmware and software, we were planning on having the Raspberry Pi Pico run the FSM and keep track of the board state while having the PC run the GUI and make calls to the Stockfish API. In this approach, the bulk of the game logic would run on the embedded device. However, since the GUI would gather the user's recommendation settings and relay that information to the FSM, it would be more difficult to integrate those subsystems if the FSM is running on the embedded device. Considerably more data on the game state would have to be sent via serial between the PC and Raspberry Pi Pico, increasing latency that could otherwise be used to do computation. To work around this, we decided to use the Raspberry Pi Pico to read the sensor network and interface with the LEDs and use the PC to run the FSM and GUI. This way the FSM, implemented in Python, would act as an intermediary between the GUI and embedded device, sending information about recommendation settings and LED display back and forth. This implementation also allowed for easier debugging since the FSM script on the PC was inspected as opposed to the Raspberry Pi Pico.

We were also challenged by integrating the GUI and FSM. We were planning on having the GUI run within the framework of the FSM. However, around late November we were running into problems having the GUI and FSM run and update in real time. To resolve this issue, we decided to implement the output and next state functions as functions that are continuously executed within the GUI. This enabled the two subsystems to run concurrently.

Additionally, we were originally planning on allowing players to configure AI strength and number of recommended moves independently from one another. However, by testing the Stockfish engine by having it play against itself, we discovered that when multiple recommended moves are enabled, the AI always plays at the highest strength. This means that the AI is unable to recommend multiple amateur or beginner level moves at once. Due to this limitation, we added additional logic in the GUI so that when multiple recommendations are enabled - the "grandmaster mode" option - the AI strength is fixed at the highest level. This means that the two settings are dependent on each other, and when grandmaster mode is enabled, the slider for AI strength is fixed in place.

Since we were initially planning on having up to four magnets per piece and using binary encoding to detect the pieces and we redesigned the product to use one magnet per piece, the game algorithm in the software needed to be redesigned to track the chess positions in memory. To make the algorithm work for one magnet per piece, we decided to keep track of the previous state of the chess board and compare it to the new state. If a change is made that reflects a legal chess move being made, the AI updates the position, and the FSM continues executing.

Project Timeline

A free Gantt Chart tool was used to create a timeline of the project. The initial timeline was modified both during the midterm design review and in November. The initial proposed timeline, shown in Figure 26, was not very detailed in the aspects of the project, as we were not very much acquainted with the intricacies of the assistive chess board. We proposed a large timeline for the PCB Design, PCB Testing, and PCB Revision. We also did not specify a timeline for the interactive GUI or interfacing with the LED light strips.

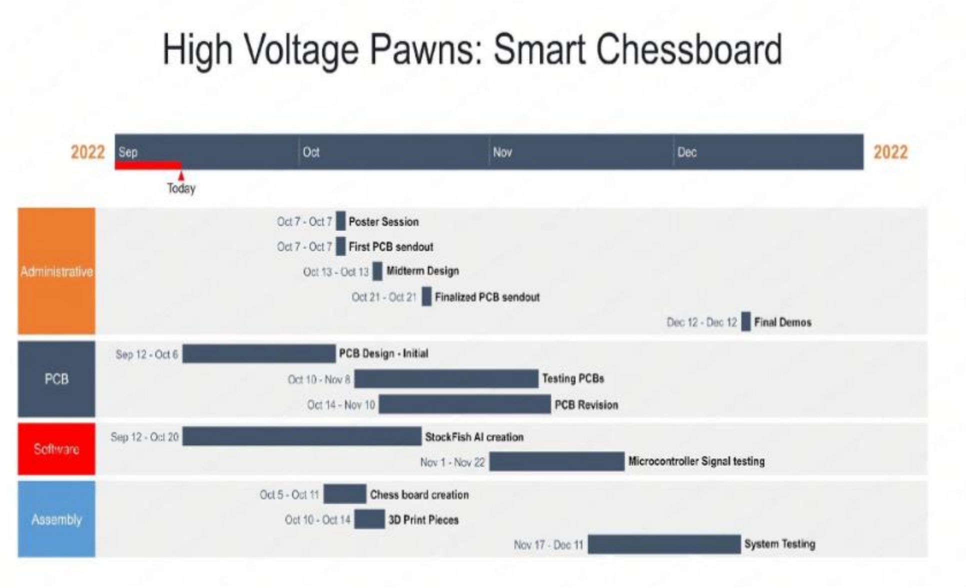
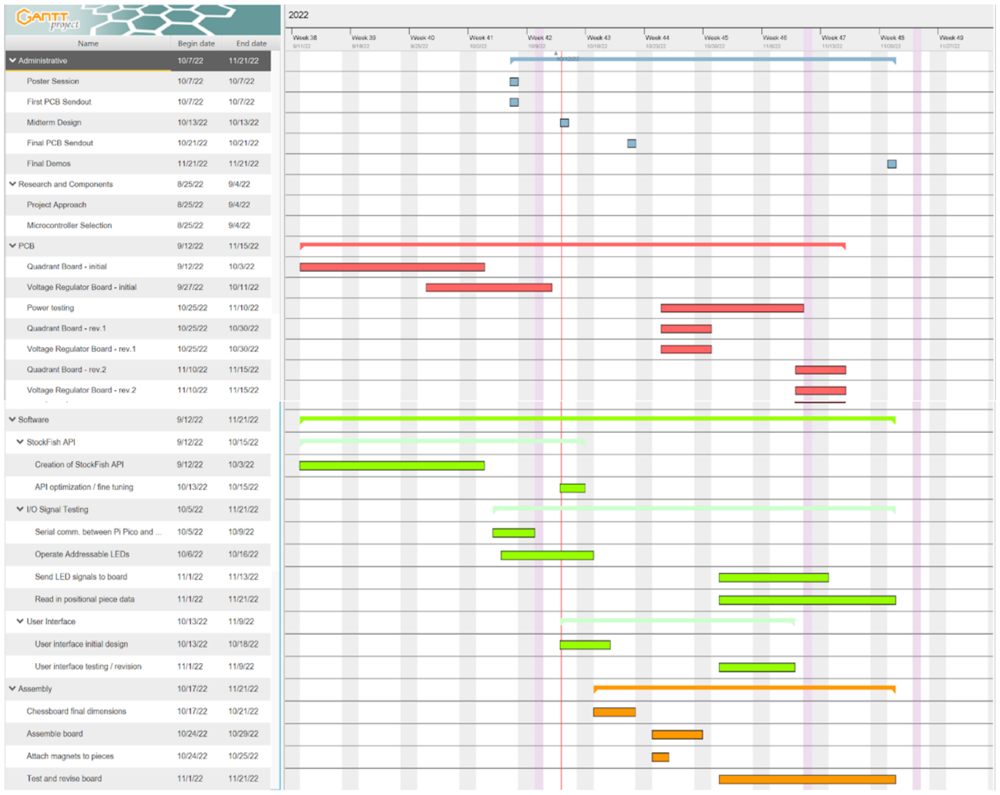


Figure 26: Initial Proposed Timeline

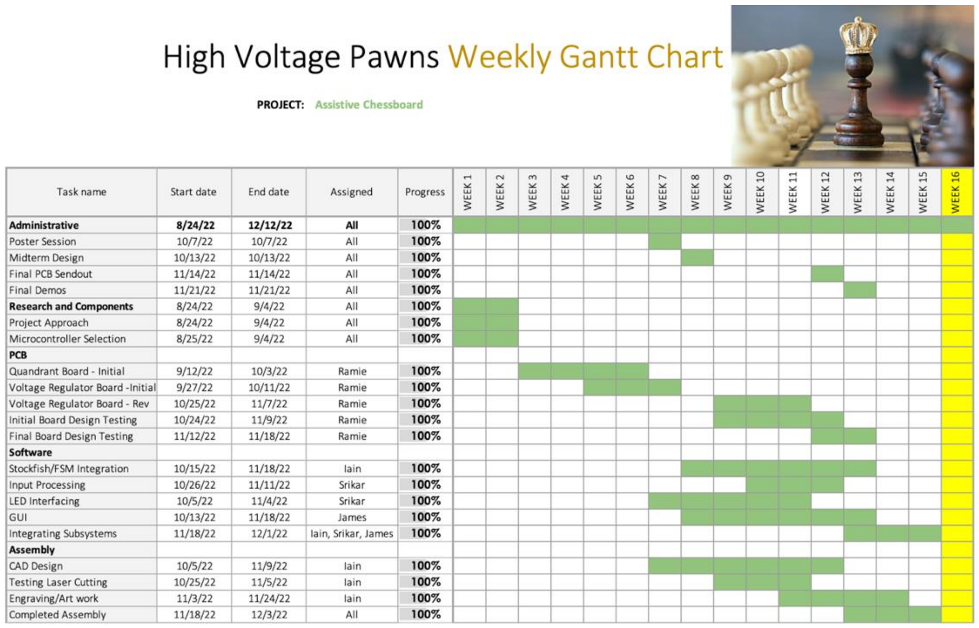
During our midterm design review, we had a much clearer understanding of how the project was going to be implemented. We added a section to interface with the LEDs, revise the boards if necessary, and read data from both the Pico. In Figure 27, we can see a more detailed midterm design review.



Name	Begin date	End date
Administrative	10/7/22	11/21/22
Poster Session	10/7/22	10/7/22
First PCB Sendout	10/7/22	10/7/22
Midterm Design	10/13/22	10/13/22
Final PCB Sendout	10/21/22	10/21/22
Final Demos	11/21/22	11/21/22
Research and Components	8/25/22	9/4/22
Project Approach	8/25/22	9/4/22
Microcontroller Selection	8/25/22	9/4/22
PCB	9/12/22	11/15/22
Quadrant Board - initial	9/12/22	10/3/22
Voltage Regulator Board - initial	9/27/22	10/11/22
Power testing	10/25/22	11/10/22
Quadrant Board - rev.1	10/25/22	10/30/22
Voltage Regulator Board - rev.1	10/25/22	10/30/22
Quadrant Board - rev.2	11/10/22	11/15/22
Voltage Regulator Board - rev.2	11/10/22	11/15/22
Software	9/12/22	11/21/22
StockFish API	9/12/22	10/15/22
Creation of StockFish API	9/12/22	10/3/22
API optimization / fine tuning	10/13/22	10/15/22
I/O Signal Testing	10/5/22	11/21/22
Serial comm. between Pi Pico and ...	10/5/22	10/9/22
Operate Addressable LEDs	10/6/22	10/16/22
Send LED signals to board	11/1/22	11/13/22
Read in positional piece data	11/1/22	11/21/22
User Interface	10/13/22	11/9/22
User interface initial design	10/13/22	10/18/22
User interface testing / revision	11/1/22	11/9/22
Assembly	10/17/22	11/21/22
Chessboard final dimensions	10/17/22	10/21/22
Assemble board	10/24/22	10/29/22
Attach magnets to pieces	10/24/22	10/25/22
Test and revise board	11/1/22	11/21/22

Figure 27: Midterm Design Review Proposed Timeline

In our finalized timeline, as seen in Figure 28, we include all the details needed to complete the project in time. We designated more accurate timing for PCB design and revisions, added more detail into the design of software, and finalized the design process to assemble the capstone project.



High Voltage Pawns Weekly Gantt Chart

PROJECT: Assistive Chessboard

Task name	Start date	End date	Assigned	Progress	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12	WEEK 13	WEEK 14	WEEK 15	WEEK 16
Administrative	**8/24/22**	**12/12/22**	**All**	**100%**	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Poster Session	10/7/22	10/7/22	All	**100%**							🟩									
Midterm Design	10/13/22	10/13/22	All	**100%**								🟩								
Final PCB Sendout	11/14/22	11/14/22	All	**100%**												🟩				
Final Demos	11/21/22	11/21/22	All	**100%**													🟩			
Research and Components	8/24/22	9/4/22	All	**100%**	🟩	🟩														
Project Approach	8/24/22	9/4/22	All	**100%**	🟩	🟩														
Microcontroller Selection	8/25/22	9/4/22	All	**100%**	🟩	🟩														
PCB																				
Quandrant Board - Initial	9/12/22	10/3/22	Ramie	**100%**			🟩	🟩	🟩	🟩										
Voltage Regulator Board -Initial	9/27/22	10/11/22	Ramie	**100%**					🟩	🟩	🟩									
Voltage Regulator Board - Rev	10/25/22	11/7/22	Ramie	**100%**									🟩	🟩	🟩					
Initial Board Design Testing	10/24/22	11/9/22	Ramie	**100%**									🟩	🟩	🟩	🟩				
Final Board Design Testing	11/12/22	11/18/22	Ramie	**100%**												🟩	🟩			
Software																				
Stockfish/FSM Integration	10/15/22	11/18/22	Iain	**100%**								🟩	🟩	🟩	🟩	🟩	🟩			
Input Processing	10/26/22	11/11/22	Srikar	**100%**										🟩	🟩	🟩				
LED Interfacing	10/5/22	11/4/22	Srikar	**100%**							🟩	🟩	🟩	🟩	🟩					
GUI	10/13/22	11/18/22	James	**100%**								🟩	🟩	🟩	🟩	🟩	🟩			
Integrating Subsystems	11/18/22	12/1/22	Iain, Srikar, James	**100%**													🟩	🟩	🟩	
Assembly																				
CAD Design	10/5/22	11/9/22	Iain	**100%**							🟩	🟩	🟩	🟩	🟩	🟩				
Testing Laser Cutting	10/25/22	11/5/22	Iain	**100%**									🟩	🟩	🟩					
Engraving/Art work	11/3/22	11/24/22	Iain	**100%**											🟩	🟩	🟩	🟩		
Completed Assembly	11/18/22	12/3/22	All	**100%**													🟩	🟩	🟩	

Figure 28: Finalized Timeline

The tasks of all the group members have been split to suit our strengths amongst the four group members. Ramie’s primary focus was designing the sensor and voltage regulation board. Srikar’s primary focus was programming the Raspberry Pi Pico to interface with the LEDs and read the sensor network and developing the GUI. James' responsibility was working on the GUI. Lastly, Iain’s primary responsibility was working on the finite state machine, interfacing between the Raspberry Pi Pico and manufacturing the acrylic chess board. Overall, using the Gantt chart, we were able to make sure we were able to make sure we are on track to finish.

Test Plan

Hardware

The hardware section of the project requires multiple steps to follow through to ensure that the devices are working properly. Due to the use of two PCBs, each one needs to be thoroughly tested. For the testing of the sensor board, we needed to make sure that the hall effect sensors are outputting the correct values to the multiplexors. Figure 29 shows the steps we took to verify the correctness of the sensor network board. The sensor board is working on a 3.3V supply rail. This means that the multiplexors and the hall effect sensors both will output either 3.3V or 0V due to the digital nature of the sensor board. The hall effect sensors are active low. This means that when the hall effect sensors are sensing a magnetic field, it will output 0V. When the magnetic is not present, the hall effect sensor will output the supply voltage. This digital nature of the hall effect sensor allows for a very easy way to debug the sensors. Furthermore, the multiplexors will act in a similar way. We will output a signal from the hall effect sensors into the multiplexor. By controlling the select bits on the multiplexor, we can sense the output of a specified hall effect sensor. With this knowledge, we can use a multimeter to measure the signals coming from and towards the multiplexor.

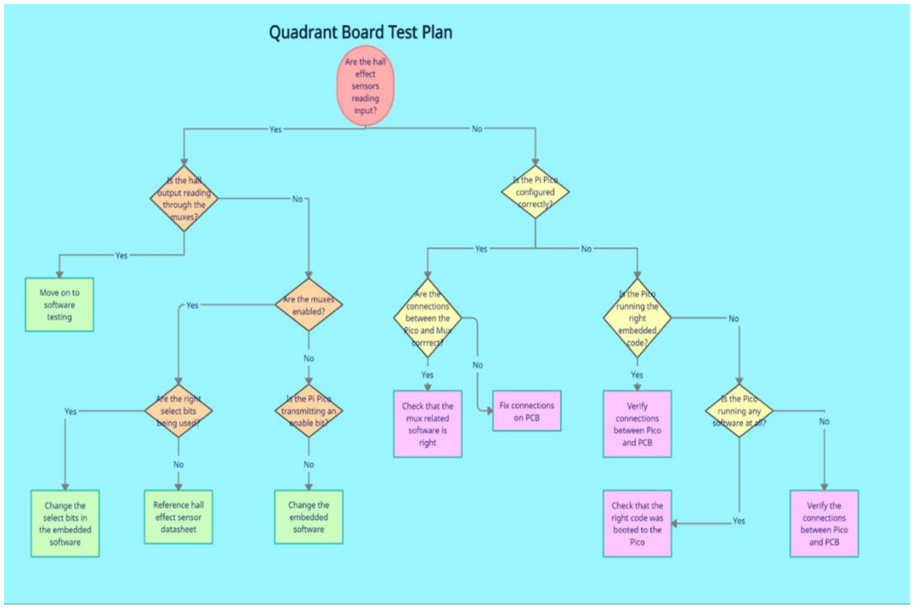

Quadrant Board Test Plan
Are the hall effect sensors reading input?
Yes
No
Is the hall output reading through the muxes?
Yes
No
Move on to software testing
Are the muxes enabled?
Yes
No
Are the right select bits being used?
Yes
No
Change the select bits in the embedded software
Reference hall effect sensor datasheet
Is the Pi Pico transmitting an enable bit?
No
Change the embedded software
Is the Pi Pico configured correctly?
Yes
No
Are the connections between the Pico and Mux corrrect?
Yes
No
Check that the mux related software is right
Fix connections on PCB
Is the Pico running the right embedded code?
Yes
No
Verify connections between Pico and PCB
Is the Pico running any software at all?
Yes
No
Check that the right code was booted to the Pico
Verify the connections between Pico and PCB

Figure 29: Decision Tree for Test Plan

Once the multiplexor has been verified to be working properly, and the enable bits, select bits, and output bits are being sent properly, we can begin to verify if the Raspberry Pi Pico is reading the signals correctly. The Raspberry Pi Pico needs to be able to sense a 3.3V level coming from the output of the central multiplexor and into a specified GPIO pin on the Raspberry Pi Pico.

For the testing of the voltage regulator board, we need to ensure that the power supply rails are 5V for the LEDs and the data signals are being read as 5V for the data line. Using the test points on the voltage regulator board, we can easily check if the voltage regulator is sending power to the LEDs. If it is not sending power to the LEDs, we know that the device is not plugged in as the 5V rail is directly connected to the LEDs. Once that is verified, we can move towards the voltage regulator aspect of the board where we can verify whether the switching node is outputting 3.3V onto the voltage rail. Due to simulations done by the Texas Instruments' WeBench and using the Virtual Bench, we have been able to verify the max load current the regulator circuit is able to provide to the sensor board.

In the beginning, there was flickering on most of the LEDs which was realized because the power adapter we were using for testing was not supplying enough current. We then tested the LEDs using an adapter rated for 15A and noticed most of the flickering go away. However, a few LEDs consistently flickered: the LED at the beginning of each strip. The first LED in a WS2812B strip boosts the data line from 3.3V logic to 5V logic. We realized the WS2812B controller was delayed in boosting the logic signal, causing flickering in particular LEDs. This problem made us pursue a logic-level translator so that the signal didn't need to be boosted by the WS2812B strip itself. This chip solved the problem, making the LEDs work as expected.

Software

The sensor processing in the embedded program was tested using the sensor network board. Magnets were placed over sensors and the raw data readings were observed in the serial monitor. We encountered issues with mismatching the select bits of the multiplexers which produced wrong readings. We debugged this using a multimeter and fixed the connections at which point the sensor processing worked as expected. We also experimented with the distance between the hall effect sensor and the magnet during sensor processing and we found the distance to be within the tolerance of our design. The LED matrix was tested using several start-up sequences to debug color mismatches. When we noticed flickering in the LEDs, we had to identify if it was caused by incorrect PWM signals from the PIO or the LED hardware itself. We used an oscilloscope to debug the issue and found the PWM to be what we expected: it matched the hex code of the LED. We then concluded the flickering in the LED was due to a hardware problem at which point we implemented the logic-level translator to solve the issue.

The FSM was initially tested using hard-code board states to check if the correct states would be satisfied. The debugging tool on VS Code was used to step through the FSM and observe states. Later, the FSM was integrated with the GUI to test the AI settings and recommendations. This is where the bulk of system-wide testing occurred. The chess board display in the GUI allowed for easier debugging on the FSM. At times, the FSM would move to an unexpected state and the chess board on the GUI would show that a chess move was not recognized. The GUI feature to start mid-game was also a valuable debugging tool as it meant we could find a sequence of moves that caused erroneous behavior and try to reproduce the bug.

Final Results

Overall, the capstone project functioned successfully. The hall-effect sensor PCB, voltage regulator PCB, embedded program running on the Raspberry Pi Pico, and GUI all functioned as expected. The sensor network was consistently able to detect the presence of chess pieces on the board. Each chess piece was properly recognized, and movements were also detected regularly. The sensor data was processed on the PC and Stockfish API was incorporated into the game logic to always suggest the engine’s best move(s). The recommended moves were displayed on the LED matrix, and all the LED colors for a given recommendation were accurate and responsive. Lastly, the GUI allowed the user to configure a multitude of settings, including the AI strength, AI depth, and number of recommendations. The GUI also automatically alternated the user’s clock time upon completed moves and had several time options for the clock. The acrylic chess board allows users to move chess pieces and view recommendations easily, making the design intuitive and responsive.

The rubric submitted in our proposal is shown in Table 4. Due to the functionality described previously, all four criteria are satisfied in the 3-point range. According to Table 5, this produces a total score of 12 points, which corresponds to an A grade.

Points	Criteria 1: **Piece Detection**	Criteria 2: **LED Output**	Criteria 3: **Chess Engine Utility**	Criteria 4: **User Interface**
3	Every Piece is detected properly.	All squares on the chessboard light up corresponding to the input from the LED driver.	The chess engine always recommends the (subjectively) best move based on the board position.	The players can configure a multitude of settings for the chess engine, utilize a chess clock, and the board is easy to play on.
2	Some pieces are detected properly.(ie. pawns, queens but not rook)	Some of the squares on the chessboard light up based on the input from the LED driver.	The engine sometimes recommends the best move based on the board position.	The players can configure some settings, use a chess clock, and the board is easy to play on.
1	Very few of the pieces are being detected.	Some of the squares on the chessboard light up.	The engine recommends any move based on the board position.	The players can't configure any settings, they can use a chess clock, and the board is easy to play on.
0	No piece is being detected	None of the LEDs are functional.	The engine does not recommend a move.	The players can't configure any settings, there's no chess clock, and the board is difficult to use.

Table 4: Grading Rubric

Grade	Total Points
A	10-12
B	7-9

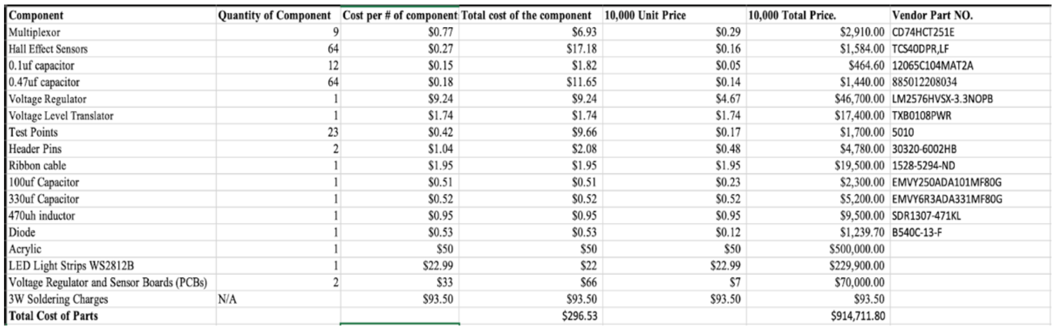
C	4-6
D	0-3

Table 5: Letter Grade based on Rubric

The project also contains several additional features that were not part of the proposed requirements. The LED functionality exceeded the aims described in our proposal. It was not only able to illustrate recommendations, but it was also able to display red on the squares that were involved in an illegal move, showing the user which piece to revert. Similarly, additional features in the GUI exceeded our initial requirements. The GUI had a feature that allowed the user to begin a game from the starting position or from a custom, mid-game position. It also displayed a chess board, which corresponds with the latest moves on the physical board. This visualization lets the user know what position to revert to in case an error arises.

Costs

The cost of one assistive chessboard system was designed with the budget of $500 in mind. A detailed breakdown of the costs is shown in Table 6. The total amount of money one assistive chessboard costs is $296.53, which is within the budget provided to us. However, the detailed list does not include the amount of time spent laser cutting the acrylic used in the assistive chessboard. The most expensive portion of our project is the charge for soldering the boards.



Component	Quantity of Component	Cost per # of component	Total cost of the component	10,000 Unit Price	10,000 Total Price.	Vendor Part NO.
Multiplexor	9	$0.77	$6.93	$0.29	$2,910.00	CD74HCT251E
Hall Effect Sensors	64	$0.27	$17.18	$0.16	$1,584.00	TCS40DPR,LF
0.1uf capacitor	12	$0.15	$1.82	$0.05	$464.60	12065C104MAT2A
0.47uf capacitor	64	$0.18	$11.65	$0.14	$1,440.00	885012208034
Voltage Regulator	1	$9.24	$9.24	$4.67	$46,700.00	LM2576HVSX-3.3NOPB
Voltage Level Translator	1	$1.74	$1.74	$1.74	$17,400.00	TXB0108PWR
Test Points	23	$0.42	$9.66	$0.17	$1,700.00	5010
Header Pins	2	$1.04	$2.08	$0.48	$4,780.00	30320-6002HB
Ribbon cable	1	$1.95	$1.95	$1.95	$19,500.00	1528-5294-ND
100uf Capacitor	1	$0.51	$0.51	$0.23	$2,300.00	EMVY250ADA101MF80G
330uf Capacitor	1	$0.52	$0.52	$0.52	$5,200.00	EMVY6R3ADA331MF80G
470uh inductor	1	$0.95	$0.95	$0.95	$9,500.00	SDR1307-471KL
Diode	1	$0.53	$0.53	$0.12	$1,239.70	B540C-13-F
Acrylic	1	$50	$50	$50	$500,000.00	
LED Light Strips WS2812B	1	$22.99	$22	$22.99	$229,900.00	
Voltage Regulator and Sensor Boards (PCBs)	2	$33	$66	$7	$70,000.00	
3W Soldering Charges	N/A	$93.50	$93.50	$93.50	$93.50	
Total Cost of Parts			$296.53		$914,711.80	

Table 6: Cost of Parts

If 10,000 units were manufactured, there would be massive discounts on the PCB components prices for buying them in large quantities. The price of the PCB's would drop from $66 down to $7, as per Advanced Circuits Custom Quote. One massive expense in our project is the voltage regulator used in the circuit. One voltage regulator would cost around $9.24. However, at the scale of 10,000 units, this price would reduce to $4.67. Lastly, since our sensor board has nearly 100 components, using a machine to solder the components would dramatically reduce the amount of money being spent on soldering. Currently, at the student rate, 3W charges $10 per board + $0.50 per component. Lastly, to reduce costs, we can print out the sensor board and the voltage regulator board in the same board. Furthermore, if that is not an option, we can solder the voltage regulator board directly to the sensor board and avoid an unnecessary ribbon cable, reducing $19,500 of costs. Lastly, if we can reduce the price of acrylic by buying in large quantities, we will be able to reduce the amount of expenditure drastically and increase profits if this product were to go on sale.

Future Work

While this capstone project was successful in detecting chess pieces and recommending moves, there are a few limitations that can be improved on. The Assistive LED chess board can be improved in the following ways.

Gamification

Our game logic has a few limitations that can be addressed for a more comprehensive experience. One such limitation is the current software does not allow the user to under promote their pawn. A feature can be added in the GUI that allows the player to select the piece to which they wish to promote. In addition, in our current design, a player who makes a blunder is forced to either proceed with the game or begin a new game starting from a mid-game position. If they choose to start mid-game, they can copy the ending position of the last game and modify it to revert the blundered move. This would be inconvenient if it happens several times. Therefore, an undo button would be valuable to revert a poor move and create a seamless playing experience.

Recommendations

Although the Stockfish API was comprehensive, there was one limitation we did not anticipate. When recommending multiple chess moves, the strength of the AI engine is automatically set to the maximum. If the user wants recommendations from a lower AI setting, a maximum of one recommendation can be generated. To improve the configurability of chess recommendations, the Stockfish source code can be modified to use the current AI strength to retrieve multiple recommendations.

Piece Detection

The approach of using hardware to solely detect the presence of pieces and relying on software to identify pieces has limitations. In some instances, especially in between moves, two chess pieces can be swapped. In this case, the hardware would detect that the same chess squares are occupied, and since a move hasn’t been made, the software would continue waiting for a move as if nothing happened. Since this implementation doesn’t differentiate between pieces in hardware, this is an area for improvement. Bipolar magnets can be used such that, for example, all white pieces have positive polarity and black pieces have negative polarity. This would account for swapping pieces of different color but would still have a blind spot for swapping pieces of the same color. Computer vision and image processing are promising methods to identify pieces and have the potential to mitigate the limitations found in hardware-based piece recognition.

Wi-Fi Module

The current chess board is aimed to be used in-person by one or two players. A user may want to play another user remotely if they both have the same chess board. A Wi-Fi module can be incorporated into the chess board that allows users to play other online users. A server can be implemented that contains networks with various categories of skill level. Players can request to play other online users of similar skill level with agreed recommendation settings. This feature would expand the possible use cases for the product and build an online community for interactive chess boards.

References

[1] Pretorius, Roline. “Why Buy an Electronic Chessboard.” *House of Chess*, House of Chess, 17 Apr. 2018, https://houseofchess.co.za/blogs/the-house-of-chess-blog/why-buy-an-electronic-chessboard (accessed Dec. 13, 2022).

[2] “ChessUp - Level Up Your Chess Game.” *Bryght Labs*, https://bryghtlabs.com/. (Accessed Sept. 10, 2022).

[3] Kaufman, Robert, et al. *Assistive Chessboard*. 2017, https://courses.engr.illinois.edu/ece445/getfile.asp?id=12121 (accessed Dec. 13, 2022).

[4] *NEMA*, 23 Jan. 2019, https://www.nema.org/ (accessed Dec. 13, 2022).

[5] “NEMA Ratings and IP Equivalency Chart.” *Siemon*, https://Siemon.com/ (accessed 12 Sept. 2022).

[6] “Generic Standard on Printed Board Design.” *IPC*, Feb. 1998 (accessed Dec. 13, 2022).

[7] M. Barr, *Embedded c coding standard.* Germantown, MD: Barr Group, 2018 (accessed Dec. 13, 2022).

[8] Van Rossum, Guido, *PEP 8 - Style Guide for Python Code*. 11 May 2022 (accessed Dec. 13, 2022).

[9] S. Z. M. Muji, M. H. A. Wahab, R. Ambar and W. K. Loo, "Design and implementation of electronic chess set," 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), 2016, pp. 451-456, doi: 10.1109/ICAEES.2016.7888087.

[10] (chess.com), C., 2022. *About Online Chess Cheating*. [online] Chess.com. Available at: <https://www.chess.com/article/view/online-chess-cheating> (Accessed Sep. 27, 2022).

[11] Fab.cba.mit.edu. 2022. *Final Project Tracking*. [online] Available at: <http://fab.cba.mit.edu/classes/863.17/Harvard/people/joshuacoven/week14_final.html> (Accessed Sep. 27, 2022).

[12] “Stockfish 15.” *Stockfish*, https://stockfishchess.org/ (accessed Dec. 13, 2022).

[13] “Forsyth-Edwards Notation.” *Forsyth-Edwards Notation - Chessprogramming Wiki*, https://www.chessprogramming.org/Forsyth-Edwards_Notation (accessed Dec. 13, 2022).

[14] “KiCad Eda.” *Schematic Capture & PCB Design Software*, https://www.kicad.org/ (accessed Dec. 13, 2022).

[15] “Welcome to Python.org.” *Python.org*, https://www.python.org/ (accessed Dec. 13, 2022).

[16] Microsoft. “Visual Studio Code - Code Editing. Redefined.” *RSS*, Microsoft, 3 Nov. 2021, https://code.visualstudio.com/ (accessed Dec. 13, 2022).

[17] “C Language.” *Cppreference.com*, https://en.cppreference.com/w/c/language (accessed Dec. 13, 2022).

[18] “WEBENCH® Power Designer | Overview | Design Resources | TI.com.” https://www.ti.com/design-resources/design-tools-simulation/webench-power-designer.html (accessed Dec. 13, 2022).

[19] *Quality Guidelines - Toshiba-Semicon-Storage.com*. https://toshiba-semicon-storage.com/content/dam/toshiba-ss v3/master/en/semiconductor/knowledge/reliability/quality-guidelines-tdsc-en.pdf (accessed Dec. 13, 2022).

[20] J. Whiteaker and R. Gardham, Opinion: Let's be honest, batteries are bad for the environment, Investment Monitor, 2021 (accessed Dec. 13, 2022).

[21] Mouser, “Surface Mount Aluminum Electrolytic Capacitors,” E1001V, 2021. MVYRA_e-2509074; Mouser: El Cajon, CA, US, 2021 (accessed Dec. 13, 2022).

[22] “Play Chess Online - Free Games.” *Chess.com*, https://www.chess.com/ (accessed Dec. 13, 2022).

[23] “Tecnotion.” 2022. *What is commutation? | Tecnotion*. [online] Available at: <https://www.tecnotion.com/faq/what-is-commutation/#:~:text=Analog%20hall%2Dsensors%20measure%20the,negative%20(South)%20magnet%20pole.> (Accessed Sep. 27, 2022).

[24] “Weetect.” 2022. [online] Available at: <https://www.weetect.com/laser-cutting-acrylic/> (Accessed 27 September 2022).

[25] Academicjournals.org. 2022. [online] Available at: <https://academicjournals.org/journal/JGRP/article-full-text-pdf/915EC0C53587> (Accessed 27 September 2022).

[26] A. B. Pagel, “Game,” DE3309817A1, Sep. 27, 1984 Accessed: Dec. 13, 2022. [Online]. Available: https://patents.google.com/patent/DE3309817A1/en

[27] Wigh, J. B. (2022). *Apparatus, system, and method for an electronically assisted chessboard* (United States Patent No. US11217117B1). https://patents.google.com/patent/US11217117B1/en?oq=11%2c217%2c117 (accessed Dec. 13, 2022).

[28] R. Dudley, “Electronic chess game,” US4391447A, Jul. 05, 1983 Accessed: Dec. 13, 2022. [Online]. Available: https://patents.google.com/patent/US4391447A/en?oq=11%2c217%2c117

[29] “RoHS Compliant Definition | What is RoHS Compliant? | Find RoHS Compliant Companies on Thomasnet.com® Certification Search.” https://certifications.thomasnet.com/certifications/glossary/other-

certification_registration/european-commission/rohs-compliant/ (accessed Dec. 13, 2022).

[30] R. P. Ltd, “Raspberry Pi Pico series,” Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-pico/ (accessed Dec. 13, 2022).

[31] “CMake.” https://cmake.org/ (accessed Dec. 13, 2022).

[32] “Git.” https://git-scm.com/ (accessed Dec. 13, 2022).

[33] “Solidworks.” https://www.solidworks.com/ (accessed Dec. 13, 2022).

[34] “Patentability Requirements,” Justia, Jun. 05, 2019. https://www.justia.com/intellectual-property/patents/patentability-requirements/ (accessed Dec. 13, 2022).

Human vs. Artificial Intelligence (AI) Matchups in Strategy Games have Facilitated the Growth of AI Technology

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Iain Ramsey

Fall 2022

Word Count: 3511

Advisor

Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

Introduction

Human vs. AI Matchups Facilitate the Growth of AI Technology (Argument)

Humans have a tendency to dismiss AI's ability to accomplish humanlike feats and this skepticism has hindered the technology's development in the past. I will argue that human vs. AI matchups in competitive strategy games have increased the validity of AI in the public's perspective and ultimately expedited the growth of the technology. Increased public awareness of the benefits that AI can provide leads to more investment and research in artificial intelligence, which consequently leads to tangible real-world solutions to problems that can benefit everyone. My research will use Actor Network Theory (ANT), which according to Rodger is a research framework that "examines the mechanics of power through the construction and maintenance of networks (both human and non-human)" (Rodger, 2009, p. 647). I chose ANT for my research because a human vs. AI interaction is fundamentally a relationship between human and non-human actors. For my methodology I analyzed three case studies of historically significant human vs. AI events, and for each event I analyzed the implications on various actors. My topic is worth researching because the more recognition that AI is a powerful tool, the more AI technology will grow and serve to benefit society.

Background

Everyone Loves Games (Common Ground)

Games have always been an integral part of culture due to their ability to captivate players, facilitate competitiveness, and often stimulate critical and/or creative thinking. According to Piccione, to first documented board game was called "Senet" and was played in ancient Egypt circa 3500 B.C.E. (Piccione, 2007). While the exact rules of Senet are unknown, Piccione speculates that Senet was a simple racing game where players took turns moving their pawns across a board, with the first player to reach the opposing side winning (Piccione, 2007), From 3500 B.C.E to today, board games have persistently occupied human's time and minds, with humanity's relationship with games expanding and adapting just as culture has changed over time. In 2022, games have become pervasive and saturated in society, with there existing many outlets to get your gaming fix – card games, board games, party games, and video games – just to name a few. Fiedler states that the United States has the biggest market for online poker, having approximately $650.5 million in revenue in 2009 alone (Fiedler, 2011). According to the United Nations, 70% of adults have played chess at some point in their lives, and around 605 million people play regularly (United Nations, 2017). In addition to the sheer adoption of strategy games by the masses, another way that games have evolved has been the introduction of professional leagues and the potential to develop a career playing strategy games at the highest level. Particularly for chess in the United States, playing at the highest level has gained substantial reputability over the past century with some professional chess players gaining the reputation as being geniuses or prodigies.

AI is Important to Cultural and Science Based Actors (Cost and Consequences)

Game-playing AI has existed since as early as 1952 when Arthur Samuel at IBM developed the first program to play checkers against humans (McCarthy, 1990). Since then, the connection between AI and games has persisted over the past century, and games are frequently used as a method to convey the status of state of the art AI technology. I will propose two reasons why the correlation between AI and games is so strong, one considering cultural actors and one considering science and logic-based actors. First, as stated in my common ground, there is no doubt that games have significance to our culture. Humans dedicate countless time and energy towards playing games because of the stimulation it provides, the fulfillment in mastering a game, and the opportunity for competition. In a similar vein, humans dedicate time towards developing game-playing AI because it's simply another way of enjoying the games they already cherish so much. From a scientific perspective, games and AI are correlated because games and computer science are correlated. Beyond their cultural utility, games provide an intuitive way of defining a problem, assigning constraints, and posing a desired outcome. This is the essence of computer science. Games are an arbitrary yet exceedingly convenient tool for applying computer science techniques. In computer science theory, reduction is a technique where a problem with an unknown solution is converted into a problem with a known solution so that the same solution can be used to solve both. An even more pragmatic explanation of why games are studied in computer science is that the technique of reduction can be used to convert game problems into real-world problems of which solutions have obvious tangible utility. Take chess for example: if a computer scientist developed an algorithm or AI that could perfectly solve chess, they could reduce chess (as a problem) to another problem (e.x. allocating food resources at an international scale) and use the known solution to solve both problems. As sensationalized as it sounds, if we could solve the game of chess, we could in theory solve world hunger.

Improving AI Will Help Us Solve P vs. NP (Destabilizing Condition)

P vs. NP is a famous problem in the mathematics and computer science community that has been studied for the past half century but has yet to be solved (Cook, 2003). Stephen Cook, the inventor of the problem, defines it as determining "whether every language accepted by some nondeterministic Turing machine in polynomial time is also accepted by some deterministic Turing machine in polynomial time." (Cook, 2003). In layman's terms, P can be considered the set of relatively easy problems to solve with a computer, NP can be considered the set of seemingly very hard problems to solve with a computer, and the P vs. NP debate is whether these sets are distinct or if they're the same. Should someone develop a solution to P vs. NP the implications would be tremendous: most unproven math conjectures would be provable, lots of real-world problems in industry would be easy to solve, and our entire basis for internet security and cryptology would be broken (Cook, 2003). Needless to say, the stakes riding on the P vs. NP problem are high, and the Clay Mathematics Institute is even offering a million-dollar prize to anyone who is able to solve the problem (Cook, 2003). The P vs. NP debate is relevant to my research on AI because artificial intelligence is essentially just an algorithm that solves a problem. Theoretically, if a perfect AI existed that could solve any problem as efficiently as mathematically possible, we would have an answer to the P vs. NP problem. In this regard it's worth investing resources towards the development of AI technology because it brings us one step closer to resolving the P vs. NP debate.

Using Actor Network Theory and Case Study Analysis (Approach to Resolution)

To approach a resolution to my topic, I chose to analyze three case studies of famous matches between professional game players and AI: Gary Kasparov vs. Deep Blue (1997), Lee Sedol vs. AlphaGo (2016), and professional poker players vs. Libratus (2017). The case studies are in chronological order to convey how attention from the media sources has been a catalyst for the development of AI over time. For each case study, I applied Actor Network Theory (ANT) to evaluate how various actors influenced the outcome of each event. According to Rodger, Actor Network Theory is a research framework that "examines the mechanics of power through the construction and maintenance of networks (both human and non-human)" (Rodger, 2009, p. 647). I chose Actor Network Theory for my research because a human vs. AI matchup is fundamentally a relationship between human and non-human actors. Figure 1 shows a generalized actor network that's applicable to all three of my case studies. My case studies demonstrate that AI has the potential to influence professional players, the computer science community, and various other actors in meaningful ways. A secondary message that is communicated throughout my research paper is that AI is a powerful tool that has the power to metaphorically move mountains, therefore, Actor Network Theory is particularly applicable to my research because it emphasizes that inanimate objects can have significant influence on the network that it's a part of.

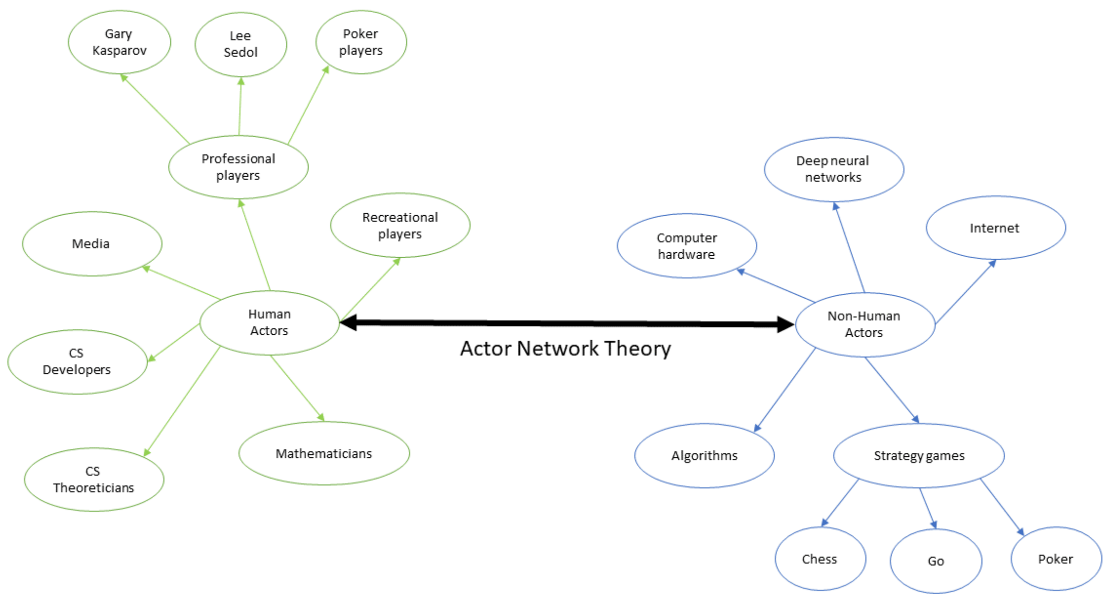


Figure 1. Actor Network of Case Studies

[Created by Author]

Case Study Analysis

Gary Kasparov vs. IBM's Deep Blue (1997)

For many centuries humans have used chess as a metric for intelligence, strategy, and creativity. In 1996, the reigning world chess champion Gary Kasparov played DeepBlue, a chess AI developed by IBM, in a tournament standard chess match (Goodman, 1997). DeepBlue beat Kasparov with a final score of 3.5 – 2.5 (Goodman, 1997). While powerful chess engines had already existed for some time, this match was a pivotal turning point because it conveyed the message that computers had unequivocally surpassed humans in chess playing abilities. Despite DeepBlue's success in chess, many scholars remained adamant that AI would not be capable of other humanlike activities, with Goodman saying, "The computer may excel at chess, but it cannot do many other things, both simple (like recognising faces) and complicated…" (Goodman, 1997, p. 186). Today we know that AI is capable of recognizing faces and much more, but Goodman's skepticism is justifiable when considering the context of the time. In 1997 computers were a fraction of the power they are today, and many engineering tasks like calculating equations, designing schematics, and sending messages were done on paper instead of a computer. The match between Kasparov and DeepBlue was arguably the first time that AI widely reached attention other than the computer science community, effectively shedding new light on AI capabilities, but nonetheless most of the true potential of AI was yet to be discovered.

In response to Gary Kasparov's loss, many professional chess players readily conceded human's inferiority in chess, and instead sought to use AI as a tool to improve their own game. Yasser Seirawan, a top US Grandmaster, chose to interpret the match as an opportunity rather than just a defeat, saying "Now we will also have the computer to analyze and tutor us on how to win or draw key positions" (Seirawan, 1997, p. 22). This sentiment by professional chess players has only grown over time, with some players going as far as to memorize computer moves prior to their matches against their opponents. Figure 2 shows the strength of chess engines over time which further corroborates this behavior because chess engines have continued to improve since Deep Blue in 1997. Sierawan also mentions that the surpassing of computers in chess has little consequence towards his own motivations to play the game because his competitive spirit remains strong, remarking "…I'll always want to clobber my brother." (Seirawan, 1997, p. 22). Considering the cultural actors that pertain to this event, it's clear that the concession of AI being better than humans did little to reduce chess player's motivations to play the game, and in some respects actually reignited top player's passion for the game because of chess engine's novelty as a training tool.

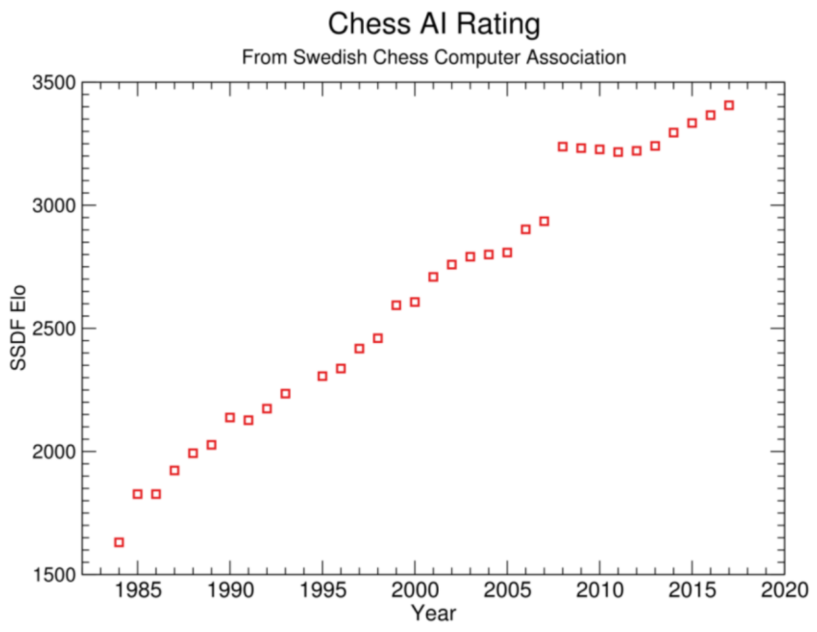


Figure 2.

Improvements in Chess AI Ratings Over Time.

Source: https://aiimpacts.org/historic-trends-in-chess-ai/

Lee Sedol vs. DeepMind's AlphaGo

The game of Go is revered for having simple rules to understand yet being profoundly difficult to master. In 2016, the neural network AI AlphaGo developed by DeepMind played Lee Sedol, the world ranked 2 Go player (Kohs, 2017). In a surprise upset to the Go community, AlphaGo won 4 out of 5 games and claimed a million-dollar prize (Kohs, 2017). This matchup shares many similarities to the Kasparov vs. DeepBlue game played 2 decades previously, but is different because Go is regarded as significantly more complicated as a game than chess. A Go board is a 19 x 19 grid, whereas a chess board is 8 x 8, so from a brute force numbers perspective there are 361! or ~10^{768} possible games of Go, but only 64! or ~10^{89} possible games of chess (Chen, 2016). To put that in perspective, there are approximately 10^{80} atoms in the universe, so it would take roughly 10^{688} *universes* to represent every possible game of Go (Davies, 1978). In addition to the massive numerical size of Go, the game is difficult for an AI to play because there's no easy way to evaluate the state of the game (Chen, 2016). For chess, it's possible to evaluate the game because different chess pieces have different corresponding values: a pawn is worth 1 point, a bishop or knight is worth 3 points, a rook is worth 5 points, and a queen is worth 9 points. It's easy enough to look at a chess game and see that if black has an extra pawn and bishop, they should be roughly 4 points better than white. In contrast, for Go the "the number of stones on the board is a weak indicator of a position's strength, and a territorial advantage is difficult to calculate", so intuition is required for Go players to evaluate the state of the game (Chen, 2016, p. 6). According to Chen, "A professional human player can make relatively easier judgments compared to a machine due to instincts that algorithms can't capture" (Chen, 2016, p. 6). Without a doubt, the game of Go is a tough problem to crack, which makes AlphaGo's success all the more impressive because of the AI's ability to replicate instinctual and "gut feeling" strategy.

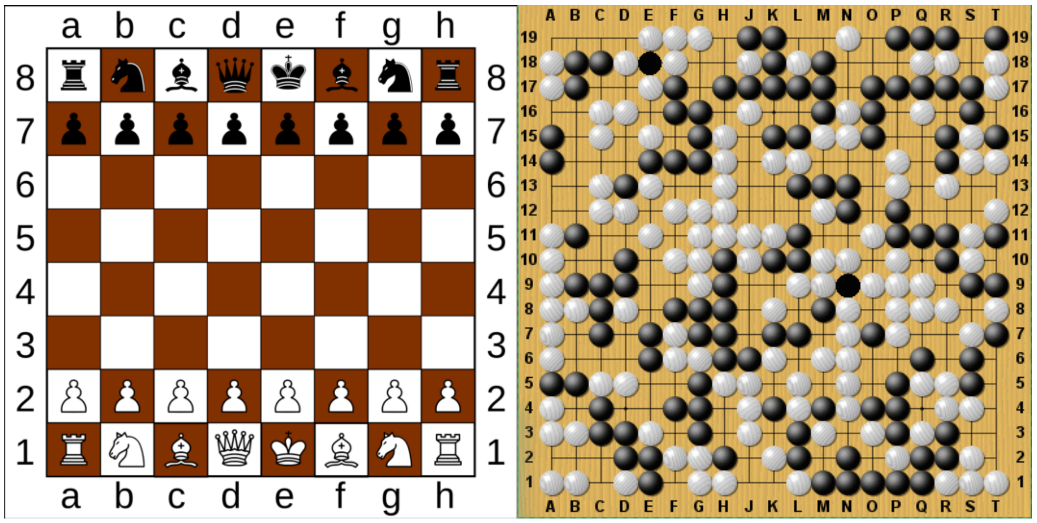


Figure 3.

Comparison of Chess (8 x 8) and Go (19 x 19) boards.

Sources: https://tromp.github.io/go/legal.html

https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg

Go is considered so complicated that some computer science theoreticians posit that the same technology pioneered by AlphaGo can be implemented to solve virtually any difficult computer science problem (Wang, 2016). Fei-Yue Wang, a professor at the University of Arizona and fellow of the IEEE, proposed the "AlphaGo Thesis" which states that "A decision problem for intelligence is tractable by a human being with limited resources if and only if it is tractable by an AlphaGo-like program" (Wang, 2016, p. 116). In other words, the "AlphaGo Thesis" states that any problem that humans are capable of solving is also solvable by an AI that has a similar architecture to AlphaGo. The underlying structure of AlphaGo includes deep neural networks, reinforcement learning, and value functions, all of which can be generalized and used for other situations (Wang, 2016). Should the "AlphaGo Thesis" prove true, the implications for the computer science world would be substantial because programmers would have a reliable tool for solving NP Hard problems, a group notoriously challenging problems that no one has found effective solutions for yet (Wang, 2016). While personally I'm doubtful that AlphaGo is the revolutionary AI that will serve as a cure-all for all computer science problems, I do think there's a strong argument to be made that AI will be the method with which we someday in the future arrive at an answer to these big questions.

Professional Poker Players vs. Carnegie Mellon University's Libratus

Poker players initially thought that AI could never play poker at a high level because it's a game of imperfect information, meaning that players don't have access to all of the variables that influence the game. To elaborate, if you're playing a game of Texas Holdem' against 8 other players, each player has 2 personal cards that only they can see, so there is at least 16 completely unknown variables at large. In contrast, a game of perfect information is a game where all of the relevant information is always known. Chess is an example of a game of perfect information because both players can see the entire chess board and consequently have equal access to data. Games of imperfect information are particularly difficult for AI to play because they rely on extrapolating unknown information and making educated guesses. AI works best when it has access to as much data as possible and can synthesize that data to calculate strategy. In this regard, Poker is much more difficult for a computer to solve than chess. In addition, Poker can be difficult for an AI because skills like bluffing and reading a player's expression are an integral part of the game, but it's much more difficult to represent these factors in a way that is interpretable for a computer.

In 2017 the poker AI "Libratus" engineered by Carnegie Mellon University professors battled against four professional poker players in a "Brains vs. AI" event that spanned 20 days and 120,000 poker hands (Sandholm, 2017). Libratus won by a large margin and 99.98% statistical significance (Sandholm, 2017). Not only did Libratus prove that computers are better than humans at poker, but also that poker itself is “weakly solved” (Bowling, 2015). “Weakly solved” means that there exists a consistently effective strategy that can be employed to give the player a winning advantage (Bowling, 2015). The notion that poker has been solved is particularly relevant because “Many real-world applications can be modeled as imperfect information games, such as negotiations, business strategy, security interactions, and auctions.” (Sandholm, 2017, p. 1). In addition to business related problems, the strategies pioneered by Libratus have applicability for military strategy and war games (Simonite, 2019). One consequence of the “Brains vs. AI” event receiving such widespread attention from the media was that the US Army signed a $10 Million contract with Sandholm, the developer of the Libratus, to reengineer the technology to be used for research at the Pentagon (Simonite, 2019). Libratus demonstrates that AI can be effective even when working with limited information at its disposal, effectively shortening the gap between what we consider human and AI capabilities.

Figure 4.

The “Brains vs. AI” Event Played by Libratus

Source: https://www.wired.com/story/poker-playing-robot-goes-to-pentagon/

Conclusion

For my methodology I chose to research three case studies, one for each type of popular strategy game. For chess, I chose to analyze the Gary Kasparov vs. IBM’s DeepBlue match because it is one of the most publicized chess matches of the 20th century and has had long lasting influence on both the worlds of chess and AI. To this day, this matchup is arguably the most well-known example of human and AI interaction. For Go, I chose to analyze the match of Lee Sedol vs. DeepMind’s AlphaGo because the game of Go is extremely complicated and serves as a good proving ground for AI to demonstrate what it’s capable of accomplishing. The match between Lee Sedol and AlphaGo shares many parallels to that of Kasparov vs. DeepBlue because Lee Sedol was (at the time) the 2nd best human Go player in the world, and they were considered the favorite going into the match. Similarly, Gary Kasparov was the reigning world chess champion, the undisputed best chess player in the world. The fact that both human players had such prestige and reputation reinforces that the stakes were very high for both matches. Their losses demonstrate that computers had unequivocally beaten humans at their respective games. If AI is better than Gary Kasparov and Lee Sedol, and they're better than every other human at their respective games, then by the transitive property AI is better than every human. For Poker, I chose to analyze Libratus because the AI played against a handful of the best poker players and was able to beat all of them with a significant statistical margin. I chose the game of Poker because it differs fundamentally from chess and Go in the sense that it is a game of imperfect information, meaning that the players are unaware of all of the variables that influence the game. Games of imperfect information are interesting when studied through the lens of AI because the AI has to make assumptions and educated guesses about the state of the game, skills that are considered to be human-like in nature.

My case studies demonstrate that victories by AI cause humans to reevaluate their preconceptions and truly consider the possibilities of AI. The recognition of AI as a useful tool consequently leads to more development of the technology. After DeepBlue, chess players started to use AI as a reference tool for finding the best moves, and chess engines have steadily improved since (Seirawan, 1997). Following Libratus, computer scientists realized that they could use AI to solve important and lucrative problems like negotiating (Bowling, 2015). After AlphaGo, theoreticians began to wonder if AI might be our best option for solving essentially any problem (Wang, 2016). When considering how AI influences how humans interact with their favorite games, I speculate that the awareness that AI are the best players mostly serves to inspire humans to get better. AI has been integrated into the professional chess scene for two decades, and I speculate that Poker and Go will follow in chess’ footsteps. As shown throughout my case studies, the improvement of AI has many real-world applications including business negotiations, long term decision making, and even war strategy. These benefits should serve as the underlying motivation for further research in this area and continue to shorten the gap in knowledge between what we speculate AI can accomplish, and the technology’s true limitations.

Works Cited

Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. Science, 347(6218), 145-149.

Brown, N., Sandholm, T., & Machine, S. (2017, January). Libratus: The Superhuman AI for NoLimit Poker. In IJCAI (pp. 5226-5228).

Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. Artificial Intelligence, 134(1-2), 57-83.

Chen, J. X. (2016). The evolution of computing: AlphaGo. Computing in Science & Engineering, 18(4), 4-7.

Davies, P. C. W. (1978). The tailor-made universe. *Sciences*, *18*, 6-10.

Donninger, C., Kure, A., & Lorenz, U. (2004, April). Parallel Brutus: the first distributed, FPGA accelerated chess program. In 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings. (p. 44). IEEE.

Fiedler, I., & Wilcke, A. C. (2011). The Market for Online Poker. SSRN 1747646.

Goodman, D., & Keene, R. (1997). Man versus machine: Kasparov versus deep blue. ICGA Journal, 20(3), 186-187.

Kohs, Greg, director. AlphaGo, DeepMind, 2017, https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind. Accessed 3 Mar. 2022.

McCarthy, J., & Feigenbaum, E. A. (1990). In memoriam: Arthur Samuel: Pioneer in machine learning. AI Magazine, *11*(3), 10-10.

Omran, A. H., & Abid, Y. M. (2018). Design of smart chess board that can predict the next position based on FPGA. Advanced Science Technology Engineering Systems. J, 3(4), 193-200.

Piccione, P. A. (2007). The Egyptian game of senet and the migration of the soul. Ancient Board Games in Perspective, Ancient Board Games in Perspective, 54-63.

Rodger, K., Moore, S. A., & Newsome, D. (2009). Wildlife tourism, science and actor network theory. Annals of Tourism research, 36(4), 645-666.

Seirawan, Y., Simon, H. A., & Munakata, T. (1997). The implications of kasparov vs. deep blue. Communications of the ACM, 40(8), 21-25.

Simonite, Tom. (2019). A Poker-Playing Robot Goes to Work for the Pentagon. Wired, Conde Nast.

United Nations. (2017). World Chess Day. United Nations. Retrieved October 25, 2022.

Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., ... & Yang, L. (2016). Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. IEEE/CAA Journal of Automatica Sinica, 3(2), 113-120.

Prospectus

Development of a Portable Chess Engine in an Embedded System
(Technical Topic)

How Human vs. AI Matchups has Changed Public's Perception Towards Artificial Intelligence
(STS Topic)

By

Iain Ramsey

March 29th, 2022

Technical Project Team Members: NA

Signed: ______*Iain Ramsey*___________________________

Technical Advisor: _______*Harry Powell*________________

STS Advisor: _____*Caitlin Wylie*_______________________

Introduction

Coinciding with the historical origins of society and culture came the introduction of games – an outlet for human interaction, stimulation of the mind, and even social status. The introduction of artificial intelligence (AI) in the past century has changed how we perceive and interact with games, and have caused us to reevaluate the limitations of the human mind and the computer. In my technical topic I will design a portable embedded system that runs a chess engine and recommends moves to the player. In my STS topic, I will look at case studies to evaluate how human vs. AI matchups have changed the public's perspective towards artificial intelligence and aided in the development of the technology. My topics are related because my portable chess engine is an example of human and AI interaction, similar in nature to my case studies. These topics are worth studying because games are culturally and economically relevant to society and because solving seemingly inconsequential games with AI can translate to solving many significant real-world problems.

Technical Topic

While implementations of a chess engine have existed for many decades, most rely on using a large powerful computer (Omran, 2018). For example, the IBM Deep Blue computer that played Gary Kasparov was approximately 6 feet tall and its had a search speed of "50–100 million chess positions per second" (Campbell, 2002, p. 59). For my technical project, I will design a handheld device containing an embedded system that will run a chess engine. My chess engine will deviate from existing products because it will be portable and necessitate operation with a weaker processor with a much smaller search speed. The system will take a board position represented as a bitboard and return one to three recommended moves as an output. As the chess game is played and the position on the board changes, the user will update these changes using buttons as input and the engine will continuously search for the best position.

The target audience for my product are chess players who are interested in improving their play using the aid of artificial intelligence. In the past two decades demand for artificial intelligence as a tool for studying chess has increased significantly (Donninger, 2004). Existing technologies rely on a powerful computer or connection the internet, but my device will operate independently from any aid and could benefit players anywhere. The idea is that you could play a game of chess against your friend in the park and then afterwards reference your portable chess engine and have it evaluate what moves you played well and what moves were blunders. I hope to achieve a simple yet powerful learning tool for chess players that can improve their chess performance anywhere they go.

STS Topic

Humans have a history of dismissing AI's ability to accomplish humanlike feats and this skepticism has hindered the technology's development. In my STS topic I will argue that human vs. AI matchups in competitive strategy games have increased the validity of AI in the public's perspective and ultimately facilitated the growth of the technology. My paper will use Actor Network Theory, a research framework that "examines the mechanics of power through the construction and maintenance of networks (both human and non-human)" (Rodger, 2009, p. 647). I chose ANT for my research because a human vs. AI matchup is fundamentally a relationship between human and non-human actors. For my methodology I will analyze three case studies of historically significant human vs. AI events and analyze the implications for various actors. My topic is worth researching because the more recognition of AI as a powerful tool, the more the technology will grow and serve to benefit society.

For many centuries humans have used chess as a metric for intelligence, strategy, and creativity. In 1996 the reigning world chess champion Gary Kasparov played DeepBlue, a chess AI developed by IBM, in a tournament standard chess match and Kasparov lost to DeepBlue (Goodman, 1997). While powerful chess engines had already existed for some time, this match was a pivotal turning point because it conveyed the message that computers had unequivocally surpassed humans in chess playing abilities. Despite DeepBlue's success in chess, many scholars remained adamant that AI would not be capable of other humanlike activities, saying "The computer may excel at chess, but it cannot do many other things, both simple (like recognising faces) and complicated…" (Goodman, 1997, p. 186). In response to the match, many professional chess players readily conceded human's loss to machine and instead sought to use it as a tool to improve their own game. Yasser Seirawan, a top US Grandmaster, chose to interpret the match as an opportunity rather than just a defeat, saying "Now we will also have the computer to analyze and tutor us on how to win or draw key positions" (Seirawan, 1997, p. 22). Sierawan also mentions that the surpassing of computers in chess has little consequence towards his own motivations to play because his competitive spirit will remain strong, admitting "…I'll always want to clobber my brother." (Seirawan, 1997, p. 22). While Kasparov's loss against DeepBlue caused humans to concede that they might be worse at chess than AI, most remained skeptical of AI's effectiveness in other circumstances.

Poker players initially thought that AI could never play poker at a high level because it's a game of imperfect information, meaning that the players don't know all of the variables that influence the game, and consequently skills like bluffing and reading a player's expression are necessary to win. In 2017 the poker AI Libratus engineered by Carnegie Mellon professors battled against four professional poker players in a "Brains vs. AI" event that spanned 20 days and 120,000 poker hands (Sandholm, 2017). Libratus won by a large margin and 99.98% statistical significance (Sandholm, 2017). Not only did Libratus prove that computers are better than humans at poker, but also that poker itself is "weakly solved" (Bowling, 2015). "Weakly solved" means that there exists a consistently effective strategy that can be employed to give the player a winning advantage (Bowling, 2015). The notion that poker has been solved is particularly relevant because "Many real-world applications can be modeled as imperfectinformation games, such as negotiations, business strategy, security interactions, and auctions." (Sandholm, 2017, p. 1). Libratus demonstrates that AI can be an effective even when working with limited information at its disposal, similar to many human-to-human interactions.

The game of Go is revered for having simple rules to understand yet being profoundly difficult to master. In 2016, the neural network AI AlphaGo developed by DeepMind played Lee Sedol, the world ranked 2 Go player (Kohs, 2017). In a surprise upset to the Go community, AlphaGo won 4 out of 5 of the games and claimed a million-dollar prize (Kohs, 2017). This matchup shares many similarities to the Kasparov vs. DeepBlue game played 2 decades previously, but is different in that Go is regarded as significantly more complicated to solve than chess. Specifically, Go is considered to be a game of intuition with many orders of magnitude more possibilities to consider than chess, so conventional AI implementations fall short (Chen, 2016). In fact, Go is considered so complicated that some computer science theoreticians posit that the same technology pioneered by AlphaGo can be used to solve virtually any exceedingly complicated problem (Wang, 2016). Should these speculations prove true, the implications for the computer science world would be tremendous because programmers would have a tool for solving NP Hard problems, a group notoriously challenging problems that no one has found effective solutions for yet (Wang, 2016). While I'm doubtful that AlphaGo is already the cure-all solution for our problems, I think it's indicative that AI will be the technology that gets us to our final answers.

My case studies demonstrate that victories by AI cause humans to reevaluate their preconceptions and truly consider the possibilities of AI. The recognition of AI as a useful tool consequently leads to more development of the technology. After DeepBlue, chess players started to use AI as a reference tool for finding the best moves, and chess engines have steadily improved since (Seirawan, 1997). Following Libratus, computer scientists realized that they could use AI to solve important and lucrative problems like negotiating (Bowling, 2015). After AlphaGo, theoreticians began to consider that AI might be able to solve any problem (Wang, 2016). When considering how AI influences how humans interact with their favorite games, I speculate that the awareness that AI are the best players mostly serves to inspire humans to get better. AI has been integrated into the professional chess scene for two decades, and I speculate that Poker and Go will follow in chess' footsteps.

Conclusion

For my technical deliverable I will design a handheld device with an embedded system that operates a chess AI and recommends chess moves to the user. For my STS deliverable I will evaluate how significant human vs. AI matchups in strategy games have changed our perspective towards AI and I will argue that they have served as a catalyst for the development of AI. I've learned from my research that it can be humbling to accept that computers will be able to beat humans in our favorite games, but also inspiring to know that AI is capable of such powerful and creative problem solving. Considering that AI continues to have breakthroughs that humans thought would be impossible, I think it's evident that there's much more to learn about the technology and it's crucial that computer scientists continue to push the boundaries of AI until the limits have been reached.

Resources

Bowling, M., Burch, N., Johanson, M., & Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science, 347*(6218), 145-149.

Brown, N., Sandholm, T., & Machine, S. (2017, January). Libratus: The Superhuman AI for NoLimit Poker. In *IJCAI* (pp. 5226-5228).

Campbell, M., Hoane Jr, A. J., & Hsu, F. H. (2002). Deep blue. *Artificial intelligence*, *134*(1-2), 57-83.

Chen, J. X. (2016). The evolution of computing: AlphaGo. *Computing in Science & Engineering*, *18*(4), 4-7.

Donninger, C., Kure, A., & Lorenz, U. (2004, April). Parallel Brutus: the first distributed, FPGA accelerated chess program. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* (p. 44). IEEE.

Goodman, D., & Keene, R. (1997). Man versus machine: Kasparov versus deep blue. *ICGA Journal, 20*(3), 186-187.

Kohs, Greg, director. AlphaGo, DeepMind, 2017, https://www.youtube.com/watch?v=WXuK6gekU1Y&amp;ab_channel=DeepMind. Accessed 3 Mar. 2022.

Omran, A. H., & Abid, Y. M. (2018). Design of smart chess board that can predict the next position based on FPGA. *Adv. Sci. Tech. Engin. Syst. J, 3*(4), 193-200.

Rodger, K., Moore, S. A., & Newsome, D. (2009). Wildlife tourism, science and actor network theory. *Annals of Tourism research, 36*(4), 645-666.

Seirawan, Y., Simon, H. A., & Munakata, T. (1997). The implications of kasparov vs. deep blue. *Communications of the ACM, 40*(8), 21-25.

Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., ... & Yang, L. (2016). Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA Journal of Automatica Sinica, 3*(2), 113-120.