Automated Chess Board

Design Document

Team 13
Client and Advisor: Dr. Zambreno
Luke Dirks - Stockfish interface and GUI design
David Imhoff - Stockfish interface and Raspberry Pi
Isaac Sachse - Sensor/Motor Work and Research
Jeet Nair - Component Design and Programming
Nathan Bellows - Arduino/RPi Logic Programming
Dawson Munday - Floating Help and Website Maintainer
Team Email
https://sdmay22-13.sd.ece.iastate.edu/

Revised: 12/05/2021

Executive Summary

Development Standards & Practices Used

- IEEE 829, Software Test Documentation:
  - Maintain tests for our software and have those tests documented to make a reliable code base
- IEEE 830, Software Requirements Specification:
  - Maintain an ongoing set of requirements defining our software to keep the team on the same page
- IEEE 1016, Software Design Description
  - Have a solid software design to keep our development purposeful
- NFPA 70
  - Adopted in all 50 states, NFPA 70, National Electrical Code (NEC) is the benchmark for safe electrical design, installation, and inspection to protect people and property from electrical hazards.
- IEEE Std 3004.8-2016
  - Industrial and Consumer motor/controller design standards
- FCC PART 15.247
  - Federal bluetooth design standards for low power 2.4 ghz bluetooth communication.

Summary of Requirements

- Board can be played without requiring additional setup per user (easy to walk up and play)
- Board has a display and interactive input (touchscreen) to control start, reset, settings, and difficulty functionality.
- Chess interface is familiar to the player; pieces are recognizable as typical chess pieces.
- Board detects player move inputs without button presses
- Exceptional cases are acceptable (pawn promotion with no possible piece to replace with).
- Development cost is within $1000 (constraint)
- Worst-case capture-and-move time of 5 seconds (constraint)
- Board has reserved spots for captured pieces and extra pieces for pawn promotion, allowing board self-setup/restore and promotion detection.
- Board mechanics are viewable from at least one angle to make it applicable as a EE/CprE demonstration for our client/the department.
- Final board development can be completed within a semester (constraint)
- The playable chess area should be less than 2'x2' to be ergonomic to play (constraint)

Stretch goals:

- Integration with online chess services such as Chess.com or Lichess.
- Simple piece movements move like the chess piece (everything but knight) for better tracking/understanding of the player (a bishop would move along the diagonal).

Applicable Courses from Iowa State University Curriculum

- COM S 309 (Large team based project)
- COM S 227 (Basics of OOP programming)
- COM S 319 (Front end development)
- COM S 252 (Linux)
- CPR E 288 (Embedded systems, sensors)
- CPR E 281 (Multiplexer)
- CPR E 185 (Basics of problem solving)
- EE 230/303 (Transformers, rectifiers)
- EE 201/230 (Circuit design, operational amplifiers)

New Skills/Knowledge acquired that was not taught in courses

- Python
- Raspberry Pi
- Arduino
- CAD/Physical Design
- Market Research

Table of Contents

Table of Contents For Diagrams and Figures

1 Team

1.1 Team Members:

Luke Dirks, Nathan Bellows, Jeet Nair, Isaac Sachse,

Nathan Kelly, Dawson Munday, David Imhoff

1.2 Required Skill Sets for Your Project:

- Electrical / Mechanical Engineering will be required to connect and control the motors.
- Understanding of microcontrollers to integrate the chess AI with the on board microcontroller and physical board components.
- Knowledge of coding will be required to track the current positions of all of the pieces and tell the motor where to move the magnets to control each move.
- Knowledge of Computer Vision may be required if that is the final choice for identifying pieces on the board (as suggested by our client).

1.3 Skill Sets Covered by the Team:

- Luke D: Coding, Computer Vision
- Nathan B: Coding and Microcontrollers
- Nathan Kelly: Motor/power control
- Jeet Nair: Coding and Design (Electrical/Mechanical Aspect)
- Dawson Munday: Coding and Computer Architecture
- Isaac Sachse: Electrical motors, Microcontrollers, minor coding
- David Imhoff: Coding, computer architecture

1.4 Project Management Style Adopted by the Team:

Our team will use an agile method, with one to two-week sprints.

1.5 Initial Project Management Roles: (enumerate which team member plays what role)

- Luke Dirks - Software Development and Testing Lead
- Nathan Kelly - Motor/driver selection and control
- Isaac Sachse - Lead researcher
- David Imhoff - Lead contact and Minute Tracker
- Jeet N - Component Design
- Nathan B - High-level Design
- Dawson Munday - Floating Help

2 Introduction

2.1 Problem Statement

Our group is designing and building a chess board that can play a user in a game of chess by moving its own pieces. In order to do this, we need to track the location of all active pieces on the board, detect when a user moves a piece, and automatically move pieces for the AI controlled player. The board also has a touch screen user-interface that will display the current state of the board, allow the user to update the difficulty of the AI on the fly, get hints on possible moves, and even reset the board. The project is intended to be displayed and used by visitors and prospective students so our goal is to make it as user-friendly as possible.

2.2 Requirements & Constraints

- Board can be played without requiring additional setup per user (easy to walk up and play)
- Board has a display and interactive input (touchscreen) to control start, reset, settings, and difficulty functionality.
- Chess interface is familiar to the player; pieces are recognizable as typical chess pieces.
- Board detects player move inputs without button presses (but might require the player to make exactly one precise move; they cannot try a couple different moves until they make up their mind).
    - Exceptional cases are acceptable (pawn promotion with no possible piece to replace with).
- Development cost is within $1000 **(constraint)**
- Worst-case capture-and-move time of 5 seconds **(constraint)**
- Board has reserved spots for captured pieces and extra pieces for pawn promotion, allowing board self-setup/restore and promotion detection.
- Board mechanics are viewable from at least one angle to make it applicable as a EE/CprE demonstration for our client/the department.
- Final board development can be completed within a semester **(constraint)**
- The playable chess area should be less than 2'x2' to be ergonomic to play **(constraint)**
- Stretch goals:
    - Integration with online chess services such as Chess.com or Lichess.

○ Simple piece movements move like the chess piece (everything but knight) for better tracking/understanding of the player (a bishop would move along the diagonal).

2.3 Engineering Standards

IEEE 829, *Software Test Documentation*: We'll need to write tests for our software and documented those will help the rest of the team understand the tests

IEEE 830, *Software Requirements Specification*: It'd be good to have requirements that we can all follow when building software for this project

IEEE 1016, *Software Design Description*: Having a clear design will help to make sure we write all of the necessary code and keep a solid timeline.

NFPA 70 → Adopted in all 50 states, NFPA 70, National Electrical Code (NEC) is the benchmark for safe electrical design, installation, and inspection to protect people and property from electrical hazards.

IEEE Std 3004.8-2016 → Industrial and Consumer motor/controller design standards

FCC PART 15.247 → Federal bluetooth design standards for low power 2.4 ghz bluetooth communication.

2.4 Intended Users and Uses

Our client intends to use the chess board as a demonstrable/interactive display of EE/CprE design work for visitors and prospective ISU students. It will serve as a source of entertainment and inspiration for those working on future engineering projects. This means that the typical user will use the product by playing and/or watching the chess board be played, and we plan to make the inner workings of the board visible from the side (and potentially through a glass board on the top).

3 Project Plan

3.1 Project Management/Tracking Procedures

Due to the nature of the project, a waterfall approach seems best. Since we are building a physical board and many of the components will rely on other components being installed first it would be hard to break the project into a series of stories.

For our project management tools, we created a discord server that we have been using for communication between group members as well as our client. For the portions of our project that require code we have been using a github repository to store and share the code between group members.

3.2 Task Decomposition

- Hall Effect Sensors needs to be designed and tested for the chess pieces
- A motor framework has to be made within the constraints of the board size
- The board itself has to be designed with movement and size constraints put in mind. (This also includes designing/buying the chess pieces)
- The code for the sensors and board to be fed into the chess engine has to be developed
- The code for whatever display system we decide on (either connected touch screen or app)
- Prototypes have to be designed, deployed, and tested.
- Parts need to be ordered.
- Final product deployment.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- Hall Effect Sensor Prototype
  - Sensors can detect magnets in a 1cm radius with 99% accuracy
- Motor Framework Prototype
  - Motors can move pieces to a destination with a margin of error of 0.5 cm
- UI Prototype
  - UI can display board state, change AI difficulty, display hints, and reset boar
- Entire Board Prototypes
  - Pieces can be moved between squares in under 5 seconds, and with a margin of error of 0.5cm. Piece locations can be detected by sensors and passed to the arduino with 99% accuracy.
- Final Product
  - Pieces can be moved between squares in under 5 seconds, and with a margin of error of 0.5cm. Piece locations can be detected by sensors and passed to the arduino with 99% accuracy. Arduino can communicate with Raspberry Pi to update board state, and to receive AI moves.

3.4 Project Timeline/Schedule

Stage	Category	Specific Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Research	------	Design	🟥	🟥	🟥	🟥								
		Components				🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
Prototyping	Hardware - Hall Effect Sensors	Order Parts								🟦	🟦			
		Verify Sensors Work									🟦	🟦	🟦	
		Verify Sensors Work Through a Base Material												🟦
	Hardware - Motors	Order Parts												
		Build Framework												
	Software - Hall Effect Sensors	Create Program to Read Sensor Inputs												
	Software - Motors	Create Program to Send and Receive Motor Inputs												
	Software - GUI	Design Python GUI for Touchscreen					🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
	Joint Component Prototype	Create Joint Framework for All Components												
		Create Programs to Send and Receive Information Between Components												
		Run Designated Tests on Hall Effect												

Stage	Category	Specific Task	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18	Week 19	Week 20	Week 21	Week 22	Week 23	Week 24	Week 25
Research	------	Design													
		Components													
Prototyping	Hardware - Hall Effect Sensors	Order Parts													
		Verify Sensors Work													
		Verify Sensors Work Through a Base Material	🟦	🟦	🟦										
	Hardware - Motors	Order Parts		🟩	🟩										
		Build Framework				🟩	🟩								
	Software - Hall Effect Sensors	Create Program to Read Sensor Inputs				🟪	🟪								
	Software - Motors	Create Program to Send and Receive Motor Inputs				🟪	🟪								
	Software - GUI	Design Python GUI for Touchscreen	🟨	🟨	🟨										
	Joint Component Prototype	Create Joint Framework for All Components						🟧	🟧						
		Create Programs to Send and Receive Information Between Components						🟪	🟪						
Testing	Software	Run Designated Tests on Hall Effect Sensor I/O								🟫	🟫	🟫			
		Run Designated Tests on Motor I/O								🟫	🟫	🟫			
		Run Designated Tests to Ensure GUI Updates								🟫	🟫	🟫			
		Run Joint Software Tests to Ensure Information is Properly Transferred Between Components								🟫	🟫	🟫			
	Hardware	Test Hall Effect Sensors Ability to Read Signals With Multiple Pieces								🟩	🟩	🟩			
		Test Motor Movement Functionality and Constraints								🟩	🟩	🟩			
		Test Touchscreen Functionality								🟩	🟩	🟩			
Revising	Hardware	Make Any Necessary Changes/Additions											🟦	🟦	
	Software												🟦	🟦	
Final Draft	------	Create and Submit Final Design													🟩

3.5 Risks And Risk Management/Mitigation

- Stockfish engine is incompatible with our design - Risk Factor of 0.01
- Hall Effect Sensors will be too large and will interfere with one another when moving pieces - Risk Factor of 0.4
- Interface isn't working or has a large delay - Risk Factor of 0.5
  - Solution: Built in displays will have a much lower chance of delay but with a touch screen based interface, the chances of it not working are higher. An app based interface faces the opposite problem
- Motor-Magnet System doesn't move a piece properly - Risk Factor of 0.75
  - Solution: Build in a system check to notify a player if a piece is incorrectly moved, not moved, etc.
- Motor System gets jammed/breaks - Risk Factor of 0.2

3.6 Personnel Effort Requirements

Task	Person Hours (total)	Members	Description
Order parts	2	all	Fill out specific order requests
Sensor Design	6	EE/CPRE members	Research and design of hall sensors
Sensor Implementation	20	all members	Add sensors to the board
Motor Design	16	EE/CPRE members	Research and design of motor parts
Motor Implementation	32	EE/CPRE members	Combining parts into working motor with sufficient testing
UI Design	10	all members	Research of good UI practices and frameworks
UI Implementation	30	SE members	Build UI and test standalone and with board
Board Connection Programming	16	all members	Get the board to interface with the motor and UI
Board Prototyping	80	all	Connect all parts and try to get a working prototype
Final Board Design	40	all	Add finishing touches and clean up rough edges

3.7 Other Resource Requirements

- Stockfish - Open source software for the chess engine.
- Various other python libraries - Displaying an appealing board visual, allowing asynchronous processes, and building a GUI for a touchscreen
- Square Off Chess Board - We're tearing into a board to get a baseline on what our prototypes and final design should look like.
- AutoCad or other cad software to help design the board and other physical components
- We will add magnets to chess pieces to allow them to interact with the sensors in the board

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Our project won't have any large consequences on public health or welfare; however, playing chess can be a stress reliever for some people, and it could be useful in that respect.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Chess is a global game, and it is played in almost every country around the world. We need to make sure that the board uses pieces that are recognizable to all cultures and that our interface is understood to all users as well.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	The board will require power to run, and in Ames the power comes from a non-renewable source. Additionally, all of the parts and sensors will also have an environmental cost associated with them as well.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	If we mass-produced and tried to sell the board maybe it would have an economic impact, but as it stands right now, it is a demo board for engineering fairs so it won't have a large economic impact.

4.1.2 User Needs

Users	Needs
Sit down users at engineering fairs or other events	A user needs to play a game in a short amount of time so the goal is to provide a strong technical demonstration of the board through the functionality, storage of game state, and the awe-factor (we assume many people in this use base will not be playing a full game of chess and simply aim to enjoy).
Prospective students	The users are trying to decide if ISU is the right school for them, so the board needs to be an impressive demonstration and display of the work done by ISU engineering students.
People who are just learning to play chess	The difficulty slider allows these users to find an AI level that matches their own skill. Additionally, if they want to play at a higher skill level they can use the hint system to help them.

4.1.3 Prior Work/Solutions

Phantom: Board utilizing hall effect sensors, board made of aluminum and a veneer cover, uses the “bus” system for taken pieces (each piece has a designated location)

Square-Off variants: Grand Kingdom - Released, requires button press. wood; SWAP - Multiple game states on one board, offers coaching; Neo- Magnetic surface for sensing, lichess/chess.com integration

Arduino boards: Dual motor, inspired by Harry Potter, requires personal input for both sides (no chess engine), movement along the lines using a magnet.

Red Square: Has fast sensors that recognize pieces and offers the best move in the position. Useful for sensing information, but all piece movement is human.

Computer Vision/Robot Arm: Board uses computer vision and understands board state (squares and pieces), then moves are completed using a robot arm that utilizes arduino alongside servo motors.

Advantages and disadvantages of previous work

Advantages: Having examples to learn from is helpful so we can see what went well and what went poorly and decide where to make improvements. Our constraints are different than the projects listed above but all aspects are touched on

Disadvantages: Many boards choose EITHER board state or piece movement, not both. Those that do tend to have button presses, computer vision, or a third-party entering system which we are trying to remove.

Pros and cons of our solution

Pros:

- Our board will be able to understand and communicate the board state to a Chess AI (stockfish)
- The Chess AI can then communicate the best move to the board and the piece will move on its own
- Our board will aim to look as human as possible with moves, captures, and time of play
- Our board will not require button presses or computer vision, allowing our board to mimic real chess as closely as possible

Cons:

- Our board, having many traits packed into one device, will be complex.

-Phantom, Square-Off, Arduino board, Red Square, Computer Vision/Robot Arm

4.1.4 Technical Complexity

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles
    a. The physical board
        i. Wooden and glass panels
        ii. Appropriately sized chessboard
        iii. Intuitive and clear pieces
    b. Magnets and Sensors
        i. Hall effect sensors for each square on the board and each piece location in the garage
        ii. Magnets in the bottom of every piece to trigger the hall effect sensors
    c. Stepper Motor and magnet to move the pieces on the board
    d. Arduino to control the stepper motor
        i. Code that interfaces between the raspberry pi and stepper motor so the stepper motor knows where to move the AI-controlled pieces
    e. Stockfish and Raspberry Pi
        i. Code that interfaces between the Arduino and the Stockfish AI so the AI knows which pieces have been moved and then responds with its own move.
2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.
    a. Detecting piece location and movement
        i. At the moment there are a few quick starter projects attempting to solve this issue, however, there is no purchasable product on the market that uses Hall Effect Sensors for this purpose.
    b. Piece tracking and interfacing with the Stockfish AI
        i. At the moment the industry standard is to use either python or C++ to communicate between a microcontroller and Stockfish, and this is what our team plans to do as well.

4.2 Design Exploration

4.2.1 Design Decisions

- What microcontroller do we want to use for communicating with Stockfish?
    - We considered two different microcontrollers for this, the Arduino and the Raspberry Pi. We decided to move forward with the Raspberry Pi, as it has a real-time operating system that is needed to run Stockfish, and the Arduino does not.
- What languages does the group want to use to control the stepper motors and to communicate with Stockfish?
    - To communicate with the steppers motors we are going to use an Arduino, and Arduino's run a modified version of C/C++ so our decision regarding that language was fairly simple.
    - To run and communicate with Stockfish we decided to use a Raspberry Pi. Based on that first decision we decided to use Python as it is the most commonly used language for Raspberry Pis so we would have the most resources if we got stuck and needed help.
- How do we want to track user movement of the chess pieces?
    - We decided to use hall effect sensors with magnets in the pieces. The voltage output on the sensor will change when the magnet is placed within range of the sensor.
    - There exists digital and analog versions of the hall effect sensor, which can be tuned or selected to detect only pieces in the square.

4.2.2 Ideation

One key decision we spent a lot of time ideating on, and eventually testing, was "what is the proper hall effect sensor to use for our project and how effective will it be in testing?". Throughout this ideation phase we had to consider numerous things such as:

- Board material and board thickness (will sensors work with certain materials/thickness)
- Magnet housing (what will we use for pieces)
- Range of sensors (is it effective for our use case)
- Type of sensor (Analog vs digital)
- Number/type of magnets (strength of piece)
- Method of sensor connection (multiplexer, arduino, or combination of)
- Piece to piece interference (Will pieces magnetize to one another, do we need casing)

4.2.3 Decision-Making and Trade-Off

Since there were many different ways to implement the chessboard we needed to agree on what method we thought was best. Below is a weighted decision matrix that we used to help make this decision. In our group's opinion, reliability was the most important factor to consider. We found that the robotic arm was the most costly, and that the computer vision approach would most likely be harder to implement compared to the Hall Effect sensors. Using this chart we decided that the best approach would be to use Hall Effect sensors and a stepper motor.

Factors	*Cost*	*Reliability*	*Ease of Implementation*	*Score*
Weights	*2*	*5*	*3*	*NA*
Hall Effect Sensors and Stepper Motor	*3*2=6*	*4*5=20*	*5*3=15*	*41*
Robotic Arm	**2=2*	*2*5=10*	*2*3=6*	*18*
Computer Vision and Stepper Motor	*2*2=4*	*4*5=20*	*2*3=6*	*30*

4.3 Proposed Design

Hall Effect Sensors - We have ordered a set of hall effect sensors, and we will begin testing them to better understand how they react to different magnets as well as the effective range of the sensors. Once that is done we can begin designing the full board and the pieces.

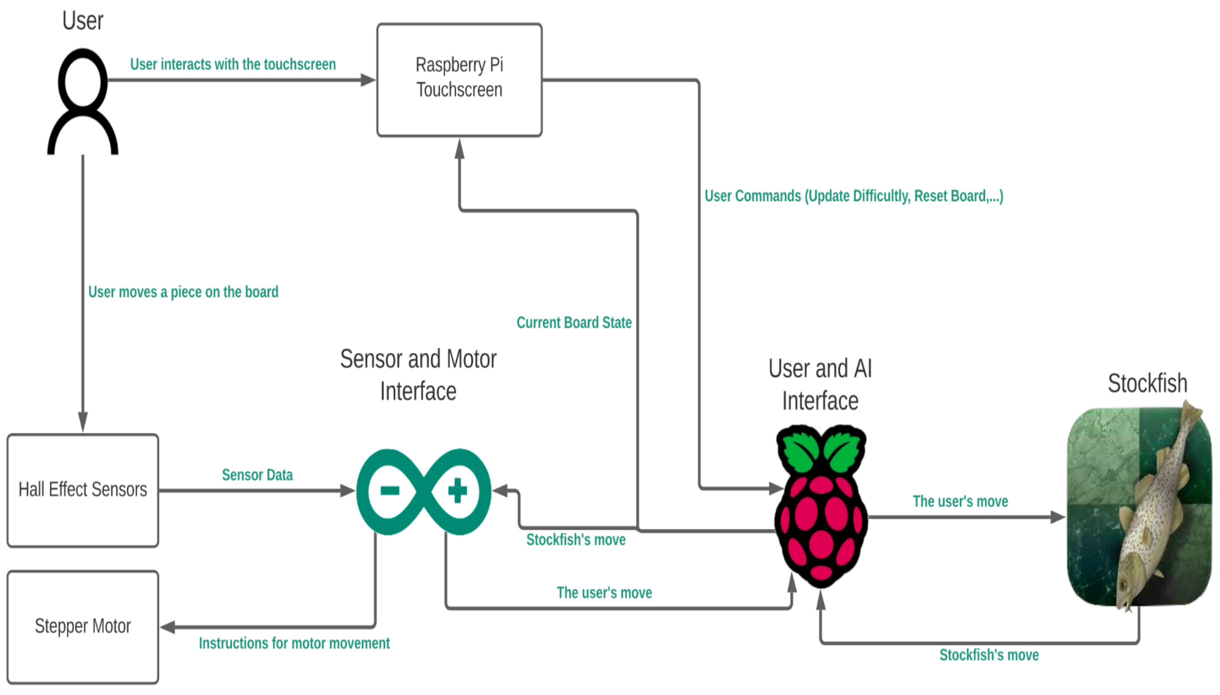
Python Stockfish Interface - On the software side of the projects we have started writing code for an interface that will communicate between the board and the Chess Ai we are using (Stockfish).

Stepper motors- We began to select the motors based on resolution and torque specifications. Motor controllers have been selected and will be tested alongside an Arduino to determine viability.

4.3.1 Design Visual and Description

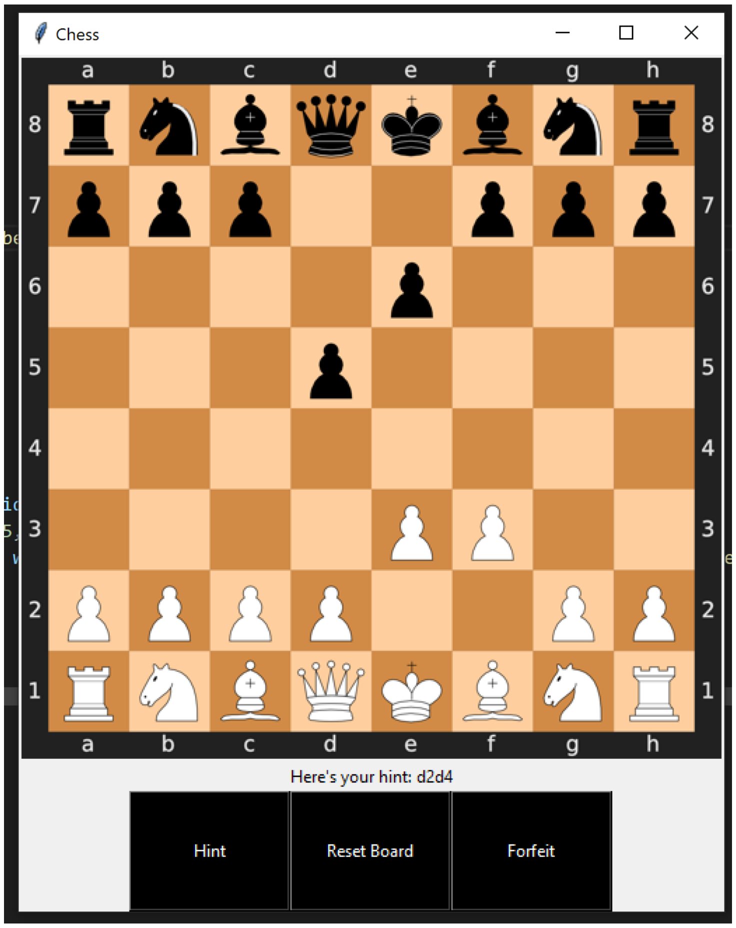
System Flow Chart

Below is a diagram depicting the overall design of our project. The process starts with a user making a move on the physical chess board. The hall effect sensors either activate or deactivate depending on the piece location, and this data is passed to the Arduino. The Arduino processes the data from the Hall Effect sensors, and determines the start and end location of the piece, for instance in chess terminology a move might be A2A4. This data is passed to the Raspberry Pi which communicates with Stockfish. If the user move is valid Stockfish determines a move to play in response. This is passed to Arduino which controls the stepper motor and magnets and moves the piece. Additionally, GUI is updated to reflect the current state of the board.



Gui Prototype

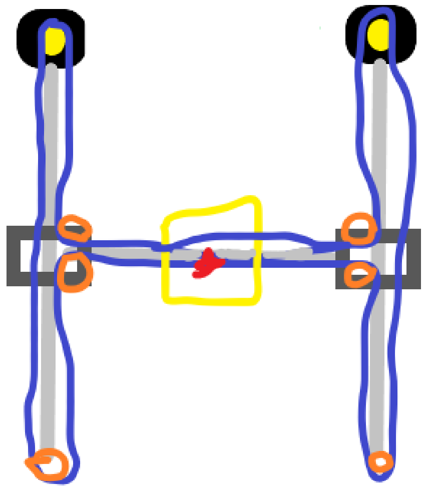
The image below is a prototype of the GUI that will be available to the user via the touchscreen on the chessboard. The current version allows the user to ask for a hint, reset the board, or forfeit the current game. We plan to add a more robust hint system and an AI difficult slider in the future.



Here's your hint: d2d4

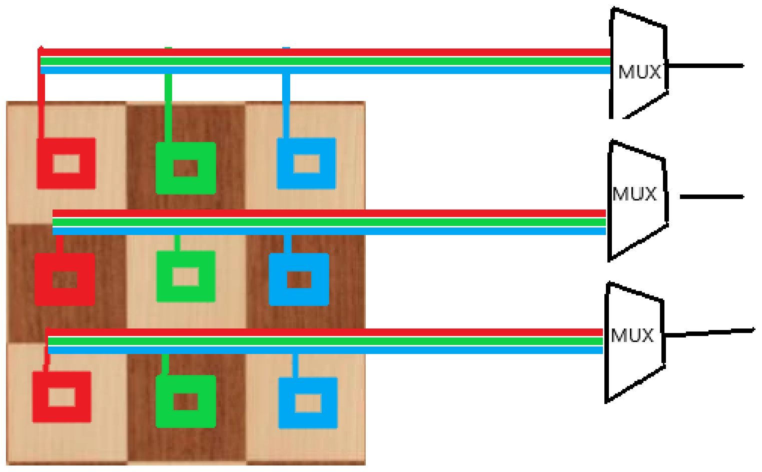
Motor Design

The image below is a prototype of the mechanism used to actually move the pieces on the board. We would use a stepper motor, gantry, and magnet to move specific pieces wherever needed.



Hall Effect Sensor Prototype

The image below is a prototype of how we would get board state information. Each piece would contain a magnet and would activate or deactivate the hall sensors in each square. By reading these values we can tell where pieces are.



4.3.2 Functionality

Our board is intended to be used for presentation purposes with users probably playing a maximum of two games in a row. To best serve this purpose, we're aiming to make a board that is portable, easy to use, and consistent. We have based on design decisions around these main purposes.

- Portability: By using an integrated stepper motor and interface we are able to contain everything necessary to use the board, except the pieces, in a single container. All of our pieces are also lightweight meaning it could easily be moved by a single person
- Ease of Use: By using hall effect sensors, we are able to track the position of the pieces on the board without requiring any extra input from the user. Also we are using an integrated touch screen to allow the user to observe the expected game state and perhaps other information about the game.
- Consistency: To make our board work consistently we've tried to limit the number of things that can go wrong. On the hardwareside, only the stepper motor and wiring could really break, and on the software side the only thing that could break would be a bug in the UI. By limiting what can go wrong, it's easier to test and make sure those things are working properly.

4.3.3 Areas of Concern and Development

One of our main concerns is the uncertainty in size of the pieces, magnets, and board. We need to do some testing to make sure all of these work together nicely. If one of these is off, it could lead to errors while playing or just an awkward playing experience. To solve this, we're already started to order a few parts to do some testing and see if adjustments are necessary. Another concern we have is that the client has requested to see the insides of the board while a user is playing. This could cause some problems since we first need to make sure the magnets work properly and then find a transparent material that doesn't break the board. The ideas we've had to solve this is using transparent panels on the sides instead of the top, and/or inserting a camera inside the board to stream the insides to another display. We've had good contact with our client and I don't believe we have any more questions at this point.

4.4 Technology Considerations

For piece detection we decided to use Hall-Effect sensors. These sensors are very versatile, they can produce analog or digital signals, they can be selected to have different ranges of operation. And this sensor works best with the magnets in the pieces needed for computer controlled movement. Some problems with having a sensor for each square on a chess board is the number of sensors required to track. This raises cost and design complexity since we are dealing with an array of 70+ sensors. Something that will need to be tested is the variation in sensor sensitivity. They may not all behave the same and with 70+ sensors, it may be hard to test each component to verify reliability across the board.

4.5 Design Analysis

Our design up to this point has worked as expected. Although we are waiting on our stepper motors to arrive, our testing with the Chess AI, the user interface, and the piece sensing will hall effect sensors have all worked to our planning and expectation. We are planning to make a stronger prototype that holds a stronger resemblance to our planned final product to improve our functional demonstration. We have observed that there are some inconsistencies with the hall effect sensors through certain objects and we have concerns of pieces interfering with one another; however, we have an idea to modify the piece slightly to prevent flux leakage on the sides of the pieces to prevent this problem. As previously mentioned, to iterate we plan to make a larger scale test surface using a glass board to see if this is a plausible option to allow viewing of the internals during use.

4.6 Design Plan

The final version of our project will have multiple key components that rely on input from each other to function properly. Therefore it is important to design and test prototypes of each component before we begin combining them. The first prototypes we are working on are the Hall Effect sensors, Arduino interface, and Stockfish/user interface. Together these three components make up the core of the system, and we want to make sure that they work before proceeding. The next stage would be to step up and test the stepper motor and magnet system for moving pieces, and to implement the Stockfish/user interface on a Raspberry Pi and touchscreen. Finally, we would move on to connecting the different components one at a time and testing each new combination. After we have finished these tests we would be ready to combine the whole board and do our final set of tests.

5 Testing

5.1 Unit Testing

Software testing

- One unit for our chessboard is the code that will interface with Stockfish. The current plan is to write these tests using PyUnit. We will write a series of test scenarios to try to catch edge cases, for instance, illegal moves, pieces being bumped out of place.

- The other portion of the software will be done in Arduino-compliant C++. This code will be used to control the magnets and the stepper motor. For the C++ code, we will create a series of test cases to confirm that we are able to move the chess pieces across the board and that we can reset the board

Hardware

- The sensors are another unit we need to test, each individual sensor will need to be tested before we add them to the board. This will be done using the oscilloscope and other tools available in the electrical engineering lab rooms. Additionally, once the board is constructed we will need to test every sensor again to make sure it was connected properly.

5.2 Interface Testing

- Arduino and Raspberry PI(GUI)
    - To test this interface we will need to send enough information from the Arduino to the Pi to deduce the user's moves, and then have the Pi send the required actions for the AI's move back. We will use testing software like PyUnit to make sure that the behavior of the Python functions is correct given various (possibly erroneous) Arduino inputs.
    - To test at a higher level than Python unit tests, a simulator of the board hardware is being prepared to be able to develop and run the Raspberry Pi Python code attached to the simulator until a board is actually completed.
- Arduino and Hall Effect
    - To test this interface we need to make sure that the Arduino can correctly detect board state and relay the information over the serial connection. We can test this C++ code with some unit tests using mocking (mocking sensor input and serial output), as well as real-world testing. If Arduino-specifics are mocked out, a popular unit testing library such as Google Test/Google Mock can be used, otherwise we will have to use in-house unit testing or open source Arduino libraries like AUnit or ArduinoUnit. However, the total amount of testable logic code that is not tied to real sensors and Arduino API is likely minimal for this interface, so we can expect minimal (if any) meaningful unit tests specific to this interface. Useful C++ library functions that are not Arduino-specific could be a good candidate for testing.
- Arduino and Motor
    - This interface requires that the arduino can pass a series of steps to the motor, which the motor will then perform. We will use the same testing methodology as the Arduino-to-Hall-Effect interface, using C++ unit tests and mocking when applicable. Again, we expect minimal logic to be tested, since the Ardunio layer serves primarily as a minimal, low-level translation layer (from data serialized in Python to Arduino GPIO inputs/outputs).

5.3 Integration Testing

The main areas of integration will be between the motor and the Arduino and the Arduino and the raspberry pi. For the motor and Arduino testing, we can write tests in C code to run on the Arduino and make sure we are getting the correct results from the motor. For the Arduino to raspberry pi integration, we can write tests in either C or Python and run them manually to make sure our connections are working as expected.

5.4 System Testing

System testing won't come into play until near the end of the project, but it's still important to think about. As part of the system test process, we should test each part of the system separately in quick automated tests just to make sure everything is still working. Then we should try sending a command from the GUI to the motor and making sure the input and output is being passed correctly and the motor is working properly. For this we can probably use Python to initiate the tests and see the results.

5.5 Regression Testing

We will plan on running a base set of tests anytime we add new functionality to the project. This will help us confirm that the old functionality still works. The critical features we need to look out for are the ability to move pieces with the stepper motor and piece location detection with the hall effect sensors. In our case these tests are driven by requirements, as it doesn't matter if the rest of the board works if we can't move or detect pieces.

5.6 Acceptance Testing

When it comes to the functionality of the board a good requirement is to be able to move any piece on the board to any desired location while also accounting for any pieces that are on the board that could possibly interfere. Another goal that we should also be striving for is to make sure that detecting the pieces is going to be reliable for detecting the pieces when they move. For involving the client, it would be good to see if the client is happy with the response time of the movements while also being accurate.

5.7 Results

The results of our testing should be a system that is reliable, modular, and easy to troubleshoot. One of our biggest requirements for this project is that the board needs to be consistent and not randomly break in front of users. By implementing good tests we can make this requirement a reality. Also by implementing good tests we can see if a single part of the system is bottlenecking the rest of the system which would allow us to provide a better user experience.

6 Implementation

- Hall Effect Sensor Prototype
    - The first implementation of the hall effect sensor prototype has already been started. We have determined a general effective range for the sensors, and we have also found that the sensors are still usable through a layer of other material. We plan on creating a prototype piece and a small sample board before our presentation.
- Motor Framework Prototype
    - The motor framework prototype has not been started at this time. We have placed an order for the motor and are waiting for it to arrive. Once it does arrive either at the end of this semester or the beginning of next semester we will make a prototype that allows the arduino to control the stepper motor.
- UI Prototype
    - Basic display of board with buttons for hints, resets and forfeits. Developed with Tkinter which is a python GUI library. Currently works with mouse clicks, but will be edited to work with touch screens.
- Entire Board Prototypes
    - The 3 prior prototypes will be brought together as a single cohesive system. We will have multiple test cases to verify the functionality of each core component as well as any joint functionalities between 2 or more components.
- Final Product
    - Once the complete board prototype has been tested properly, we will create a housing for the whole system to rest in so that none of the major components are exposed.

7 Professionalism

7.1 Areas of Responsibility

Areas	IEEE	NSPE	Personal Description
Work Competence	Be realistic about claims and estimates based on available data. Maintain technical competence and require proper training for tasks.	Be honest about your skillset and don't take on projects that you are not equipped to handle.	Understand your limits and make sure to produce high quality products
Financial Responsibility	Reject bribery. Again, be honest with estimates such as cost.	Create products with values for clients at a reasonable price point.	Don't buy things that aren't very helpful to the end product
Communication Honesty	Disclose factors that could endanger the public/environment. Accept/offer honest criticism.	Honestly report your work, and be truthful in any public statements issued.	Be consistent and clear with updates on your progress
Health, Safety, Well-Being	Make decisions consistent with safety/health of the public. Avoid injuring others.	On all projects work in a way that keeps risk to workers and clients as low as possible.	Take care of yourself and keep the health of other in consideration at all times
Property Ownership	Avoid damaging others' property.	Respect both the equipment and IP of your company and clients.	Make sure to respect the owner of all equipment
Sustainability	Again, disclose factors that might endanger the environment.	During projects keep sustainability in mind and try to reduce climate impact wherever possible.	Make sure all potential negative environmental impacts are disclosed
Social Responsibility	Assist colleagues in their development. Treat all people/identities fairly.	The products you create should benefit people or society in some way.	Treat all co-workers, clients and others with respect

The IEEE ethics has less emphasis on product price point or environment, at least in its direct statements (you can extend the "welfare of the public" clause to relate to these though). Both highly value honesty and expect tasks to be dispatched to those with proper training and experience. Overall the NSPE has more specifics given in their document than the IEEE ethics outline provided.

7.2 Project Specific Professional Responsibility Areas

Areas	Automated Chess Board
Work Competence	(Medium) Team members should be honest about their skill sets so that the team can properly schedule tasks.
Financial Responsibility	(Medium) There is some investment from Iowa State into this project and, as such, we owe it to the school to create a functioning project.
Communication Honesty	(High) Let teammates know when something isn't going well and be honest about your contributions to the project.
Health, Safety, Well-Being	(Low) There are not a lot of health risks involved in the project. If we do end up machining our own parts that may change.
Property Ownership	(Medium) There's no real IP to be concerned about; however, we do need to be careful with the sensors and motors.
Sustainability	(Low) We are only making one board, so there isn't really a lot of environmental impact.
Social Responsibility	(Medium) People might get enjoyment out of playing a working version of our project.

7.3 Most Applicable Professional Responsibility Area

Communication is a very crucial aspect of our project that we have had to exert a lot of focus on. As we have been told by Dr. Zambreno, there were a few groups in the past that had done a similar project (and we learned of their shortcomings) and as such, it is important to have a lot of transparency in our work in order to ensure that any issues that ensue during the project will be addressed effectively and efficiently. We will have to continue to pursue this area to make sure that each of the components is working functionally and we can readjust the workflow accordingly. Currently, we are talking amongst each other with almost every aspect of the project and getting proper feedback from Dr. Zambreno and the rest of our team members to ensure quality control.

8 Closing Material

8.1 Discussion

Our project is on the experimental side of things, yet there are products similar to it on the market so we can discuss both venues. As a product, we are currently meeting requirements on both the software and hardware side of things and remaining within the constraints of our budget. As an experiment, our prototypes and additions have all worked well individually, but we have not had an opportunity to combine everything into one cohesive product/device just yet (as this will come after larger prototypes in our second semester of the project). Overall, I would say that we are meeting the requirements and expectations of this first semester if not exceeding them and we have set ourselves up well for the future with more in-depth prototyping and building a final product.

8.2 Conclusion

As of the work we have done so far: We have designed a functioning communication with the Chess AI, Stockfish, and integrated it into a simple UI. We have also been able to design a recognizable and appealing interface using Raspberry Pi. On the hardware side of things, we have made a functional tester for our hall effect sensors and grabbed an effective range and come up with solutions found during testing. We are still waiting for the arrival of our stepper motors so we have yet to work with our electromagnet and piece movement, and we have yet to integrate the software side of things with the physical side of things.

The project requirements/goals consist of making a board that can set itself up, know the game state, has an interactive UI, interface is recognizable, board is recognizable, realistic piece movement times, and remaining within price and time constraints.

Given the scope of our project, the group has done well adhering to these goals and remaining within constraints and requirements. We have found solutions to make our board appears very similar to chess (allowing it to be recognizable), the UI is interactive and recognizable, and the board can recognize game state and even give hints based on the state of the game; however, it is important to note that, as we do not have a full prototype we cannot speak on the piece movement and self-setup goals we have for ourselves. As for our stretch goals of online integration and simple movements for all pieces, these have yet to be tackled as they are stretch goals. In the future we could have potentially ordered pieces sooner and dedicated more time to integration of software and hardware, but the group worked well on their respective tasks and fell well within the given constraints and expectations given the scope of the project.

8.3 References

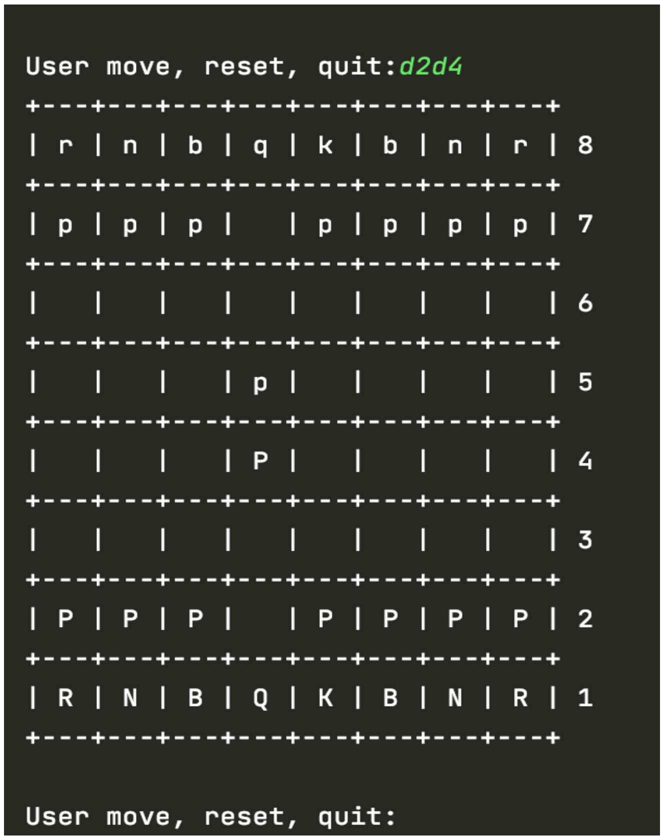
"Stockfish 3.18.0 ," PyPI, 17-Nov-2021. [Online]. Available: https://pypi.org/project/stockfish/. [Accessed: 05-Dec-2021].

"Graphical user interfaces with TK," Graphical User Interfaces with Tk - Python 3.10.0 documentation, Dec-2021. [Online]. Available: https://docs.python.org/3/library/tk.html. [Accessed: 05-Dec-2021].

"Asyncio - asynchronous I/O," asyncio - Asynchronous I/O - Python 3.10.0 documentation, Dec-2021. [Online]. Available: https://docs.python.org/3/library/asyncio.html. [Accessed: 05-Dec-2021].

8.4 Appendices

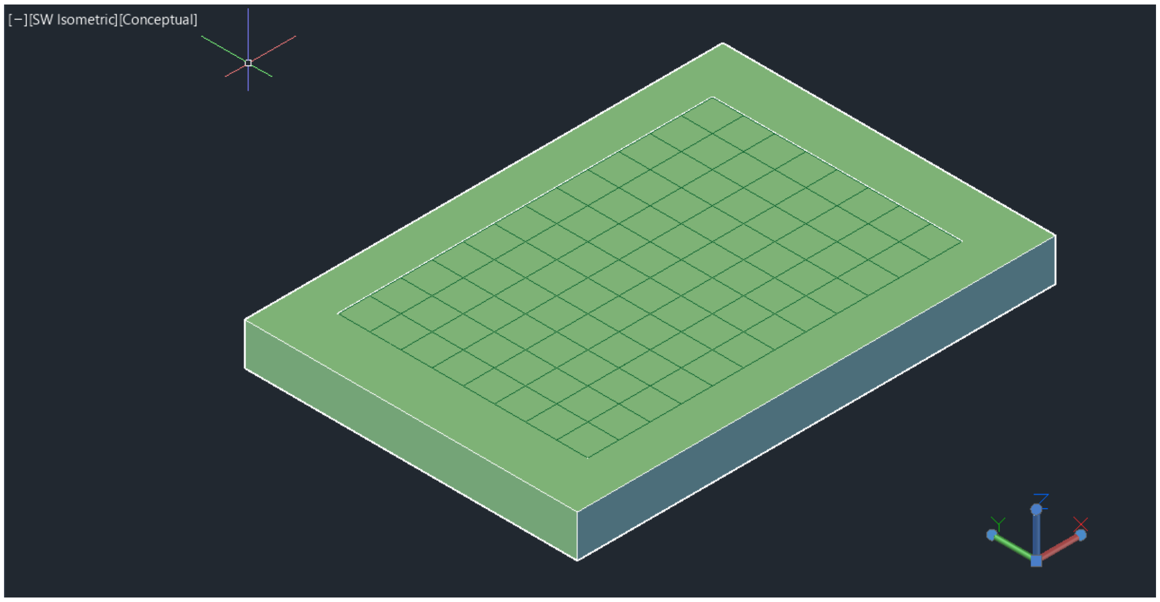
Original Text-Based GUI Prototype



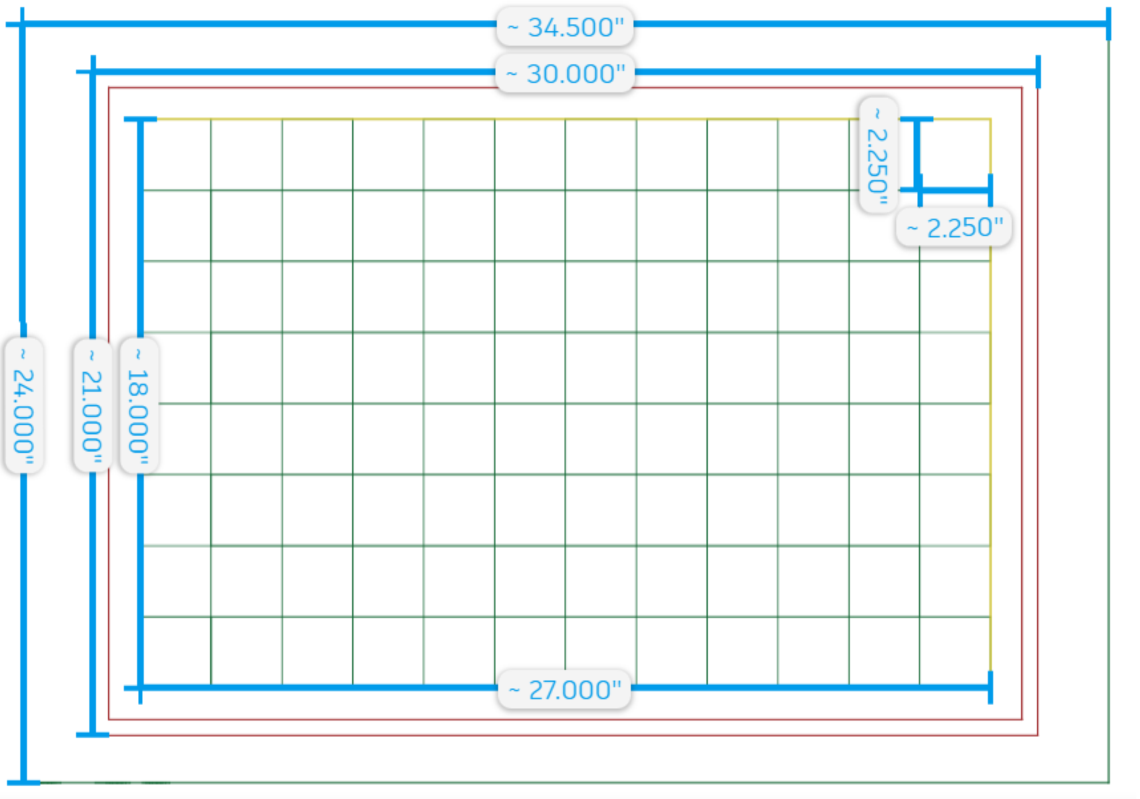
```
User move, reset, quit:d2d4
+---+---+---+---+---+---+---+---+
| r | n | b | q | k | b | n | r | 8
+---+---+---+---+---+---+---+---+
| p | p | p |   | p | p | p | p | 7
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   | 6
+---+---+---+---+---+---+---+---+
|   |   |   | p |   |   |   |   | 5
+---+---+---+---+---+---+---+---+
|   |   |   | P |   |   |   |   | 4
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   | 3
+---+---+---+---+---+---+---+---+
| P | P | P |   | P | P | P | P | 2
+---+---+---+---+---+---+---+---+
| R | N | B | Q | K | B | N | R | 1
+---+---+---+---+---+---+---+---+

User move, reset, quit:
```

CAD 3D Model



CAD Wireframe Model With Measurements



8.4.1 Team Contract

Team Members:

1) Jeet Nair 2) Luke Dirks 3) Nathan Kelly 4) Isaac Sachse

5) Nathan Bellows 6) Dawson Munday 7) David Imhoff

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

Our group meets Wednesday at 3 PM on our Discord server.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

We are using a Discord server as our primary means of communication, and we supplement that with weekly voice calls.

3. Decision-making policy (e.g., consensus, majority vote):

We attempt to find consensus on issues where we can, if that is impossible we can resolve issues with a majority vote.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

David was designated for meeting minutes and keeping all of our major points for each meeting listed in a channel in our Discord Server

Participation Expectations

We have asked members to make an attempt to attend all meetings and arrive on time. If a member has to miss a meeting or they will be late they should inform the group via discord.

Leadership

1. Leadership roles for each team member):

David I: Client Interaction, Minute keeper
Isaac S: Lead Researcher
Jeet N: Component Design
Luke D: Testing, Assignment Tracking
Nathan B: High-level Design
Nathan Kelly: Motor/power control
Dawson Munday: Floating Help

2. Strategies for supporting and guiding the work of all team members:

As a group we try to make sure everyone has work to do. If someone is stuck, the best course of action is to set up a time to work with another team member. We also try to congratulate teammates when they've accomplished something.

3. Strategies for recognizing the contributions of all team members:

When a member of the team provides a contribution their work is recognized in either the Discord or during the team meetings.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the

team.

David: software and software project management
Jeet: CAD and electronics design as well as software development experience
Isaac: Circuit design/analysis (DC/AC/RLC), soldering, E/H fields for Hall Effect Sensors, power
Luke Dirks: software design and agile development
Nathan Bellows: Low and high level C/C++ programming, digital logic design (if required), general software development
Dawson Munday: Software development and some embedded systems
Nathan Kelly: Motor, sensor, and power knowledge

2. Strategies for encouraging and support contributions and ideas from all team members:

Open discord chat for all to comment. Make an effort to ask teammates about the decision before implementing it.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will

a team member inform the team that the team environment is obstructing their

opportunity or ability to contribute?):

If a team member feels like they are having trouble contributing to the project they should reach out via Discord. On the Discord members of the can then help provide ideas of areas or specific tasks that the member could contribute to. This will get the team members involved in the project and give them something to focus on.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

We would like to have finalized the documentation for our project, and also have a working proof of concept chessboard that we can iterate on during the second semester.

2. Strategies for planning and assigning individual and team work:

Our plan is to assign tasks during our weekly meetings. Depending on the size and scope of a task it will be assigned to either an individual or multiple team members. If a task requires two members of the team, those members would need to find a time that works for both of them to work on their tasks.

3. Strategies for keeping on task:

Since our meeting time is limited, discussions should be focused on the project. If a member notices the group getting off task they should speak up and refocus the conversation.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

Remind members of the team contract. Ask if they need any help to keep on track. Encourage them to follow the contract for next time.

2. What will your team do if the infractions continue?

Ask them why they keep breaking the contract. Depending on the reason we could possibly try to help them or ask them to leave the team.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the

consequences as stated in this contract.

Luke Dirks DATE: 09/18/2021
Nathan Kelly DATE: 09/18/2021
Isaac Sachse DATE: 09/18/2021
Nathan Bellows DATE: 09/18/2021
Dawson Munday DATE: 09/18/2021
Jeet Nair DATE: 09/19/2021
David Imhoff DATE: 09/19/2021