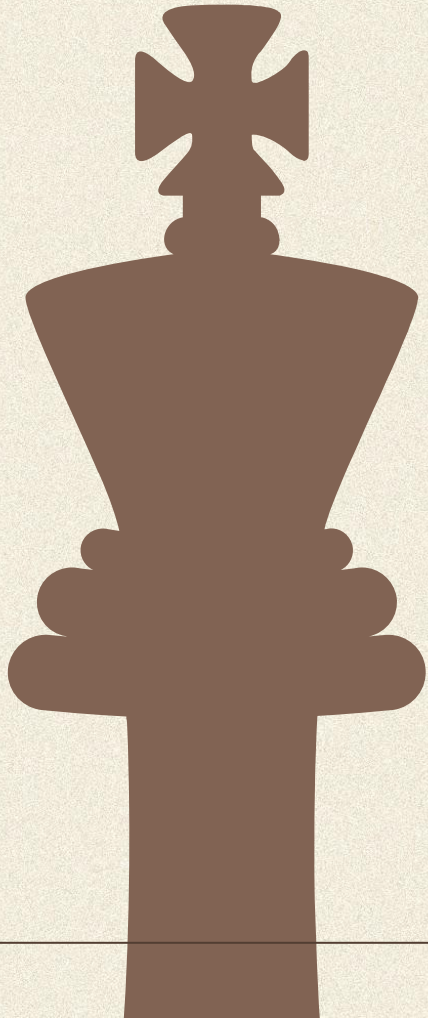
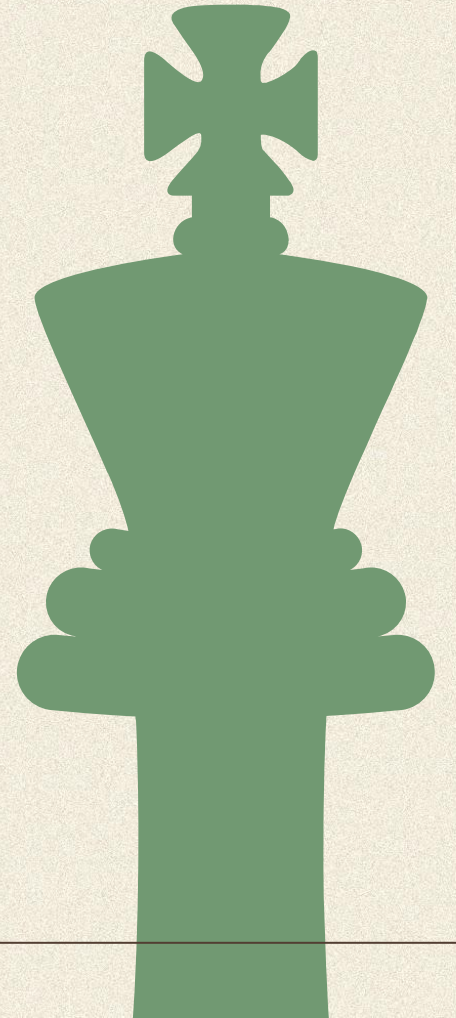


Automated Chess Board





Luke Dirks - Python GUI and Stockfish Interface
David Imhoff - Python GUI and Raspberry Pi
Isaac Sachse - Sensor/Motor work/research VA
Nathan Kelly-Sensors/E-magnet/movement
Dawson Munday - Floating Help
Nathan Bellows - Arduino/RPi programming
Jeet Nair - Component Design and Programming
Client, Advisor - Dr. Zambreno







Project Vision

- Our group is developing an automated chessboard that can detect a user's move and make a move of its own on a physical chessboard.
 - The board will use stepper motors, magnets, Hall Effect sensors, and two microcontrollers to detect and process user moves to receive an AI determined move from the chess AI, then will automatically move the AI's piece on the physical board.
- 
- 



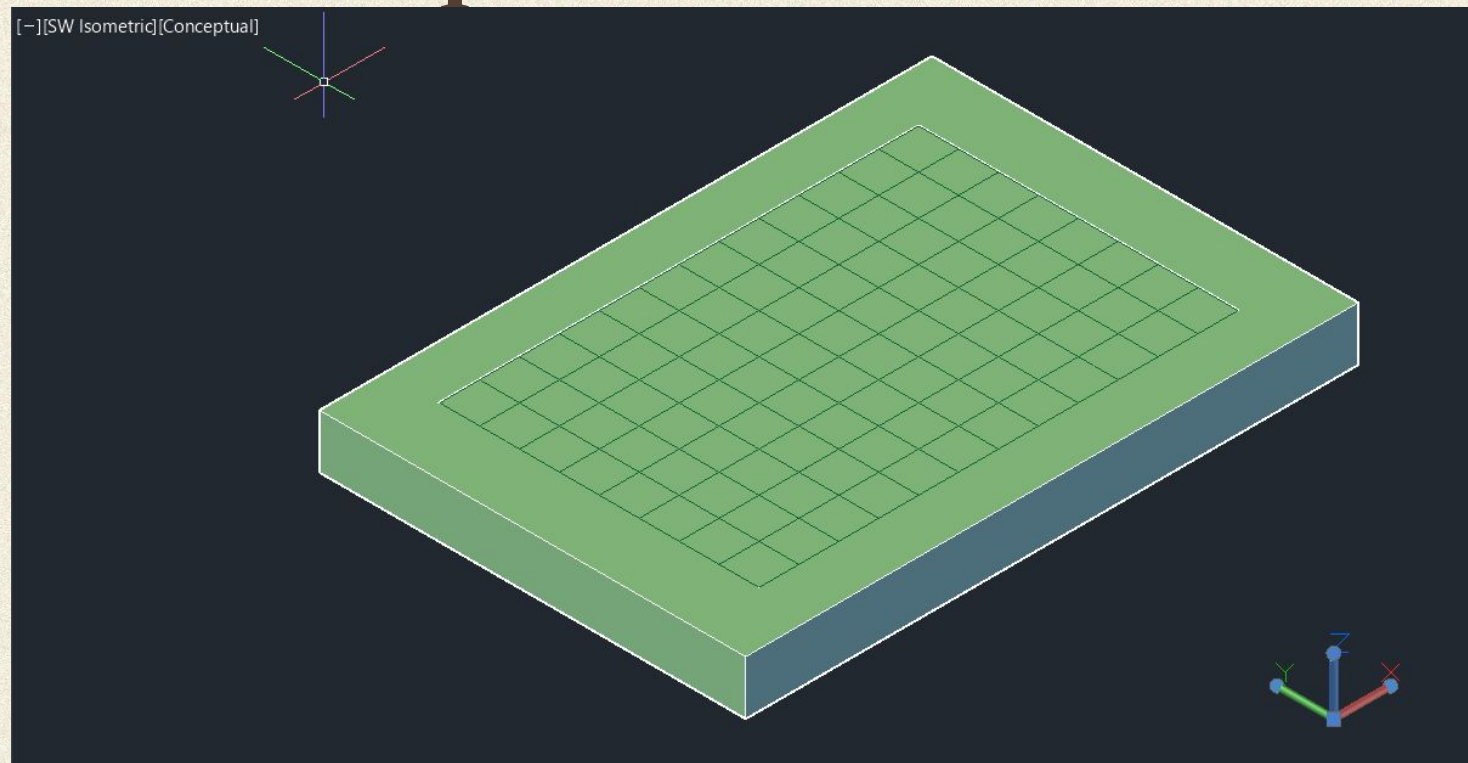
Use Cases

- Our client has asked for this board to be used for engineering fairs or other university functions.
 - It could also be used as a demonstration of the work engineering students are doing on campus for tours, with alumni or perspective students.
 - The difficulty scaling of the AI and hint system will make the board accessible to players of all skill levels.
 - We also have considered adding puzzles or replays of famous chess games as quick demonstrations of the boards potential.
- 
- 

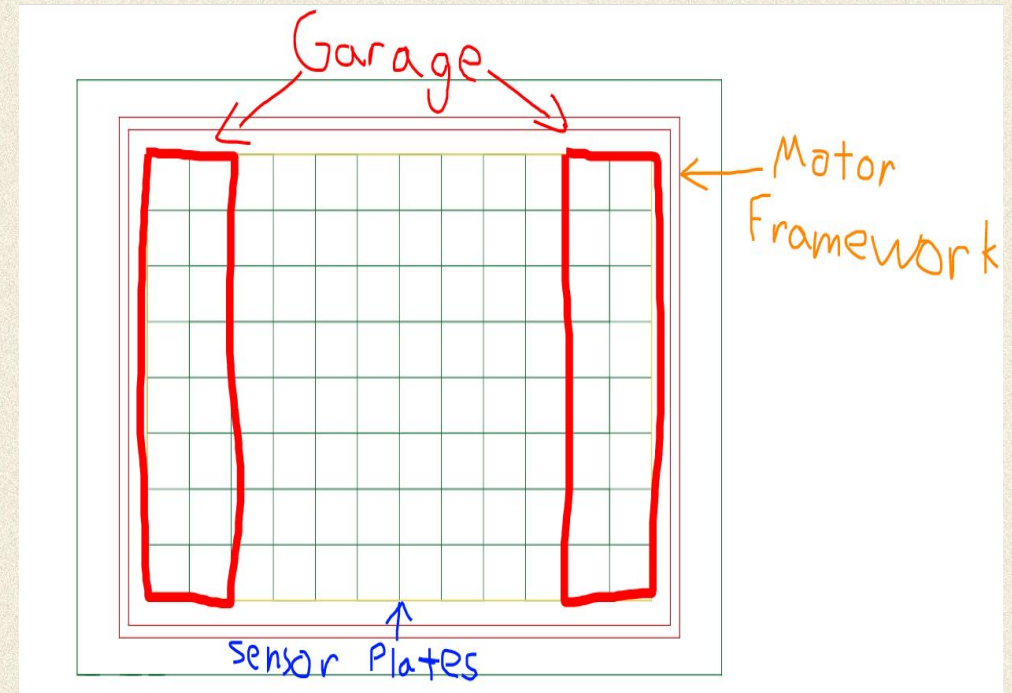
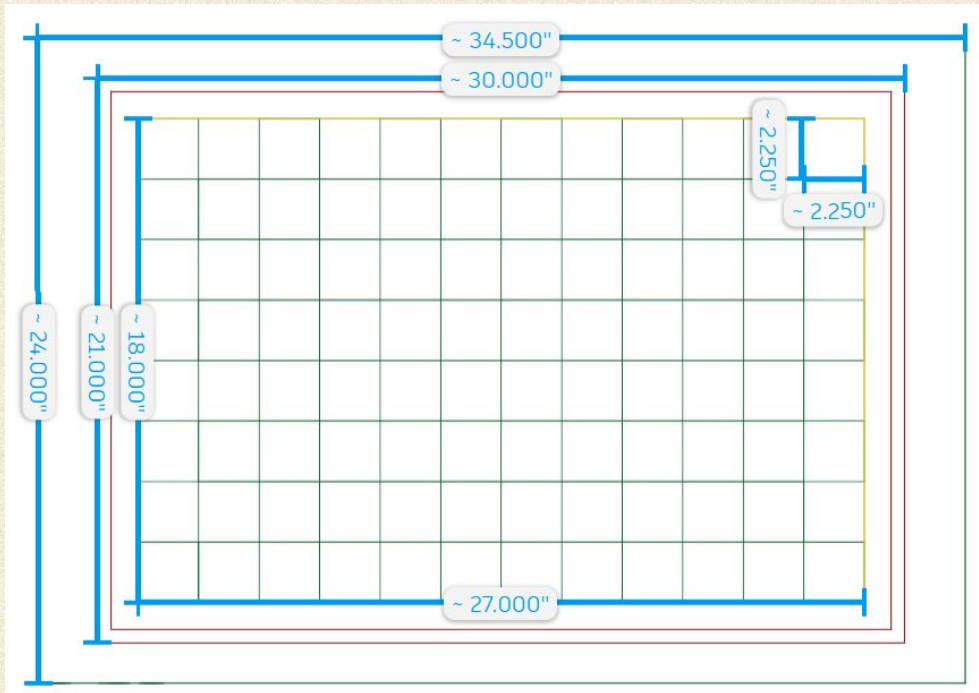
Conceptual/Visual Sketch



Conceptual/Visual Sketch

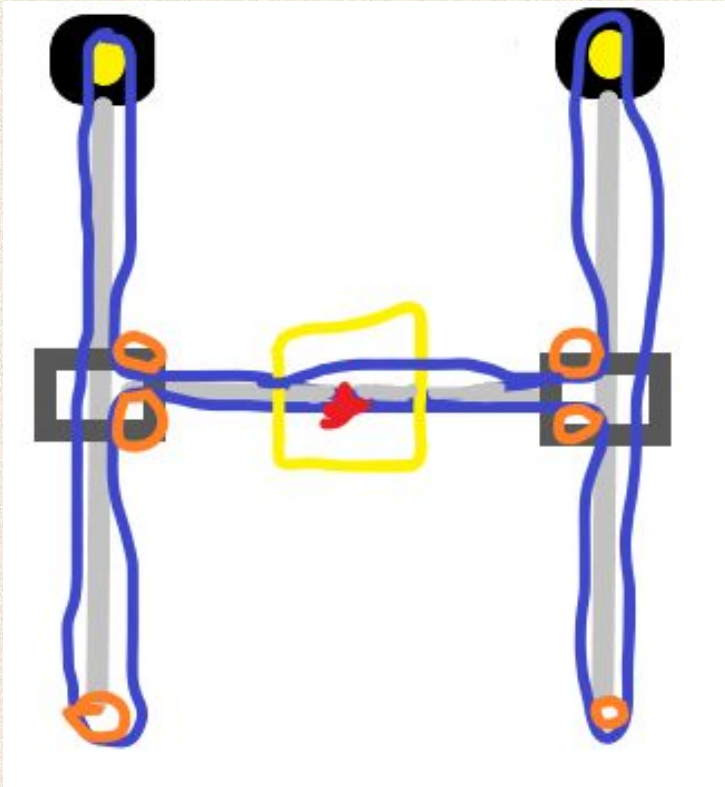


Conceptual/Visual Sketch

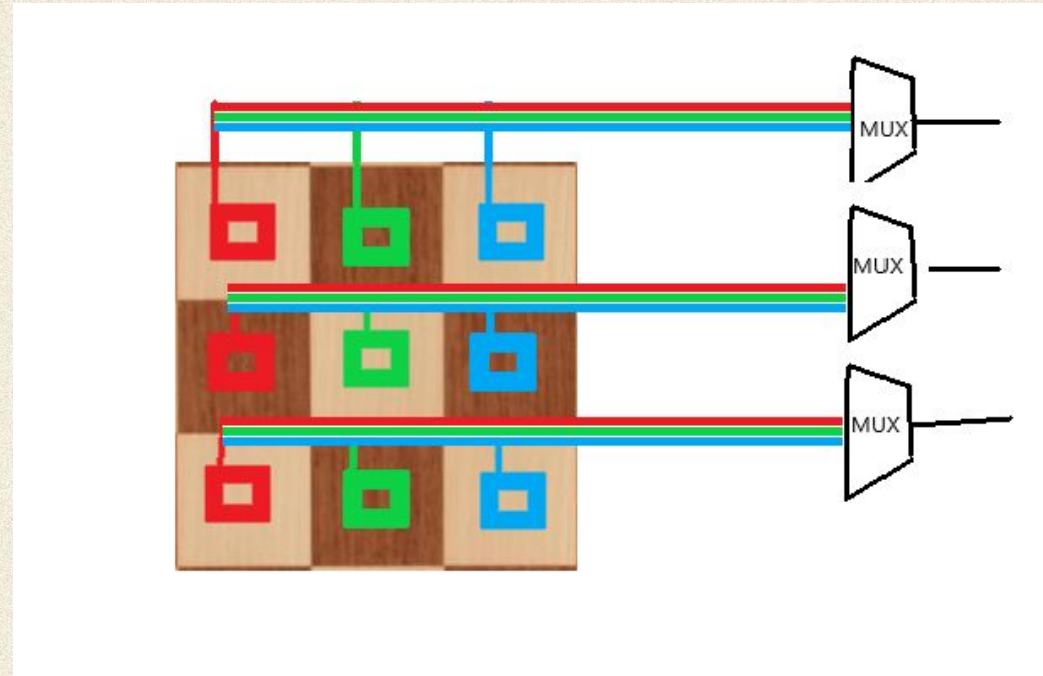


Conceptual Board Internals

Gantry Plotter Design





Hall Effect Sensor Design




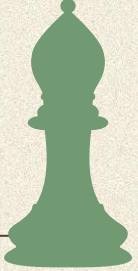


Requirements

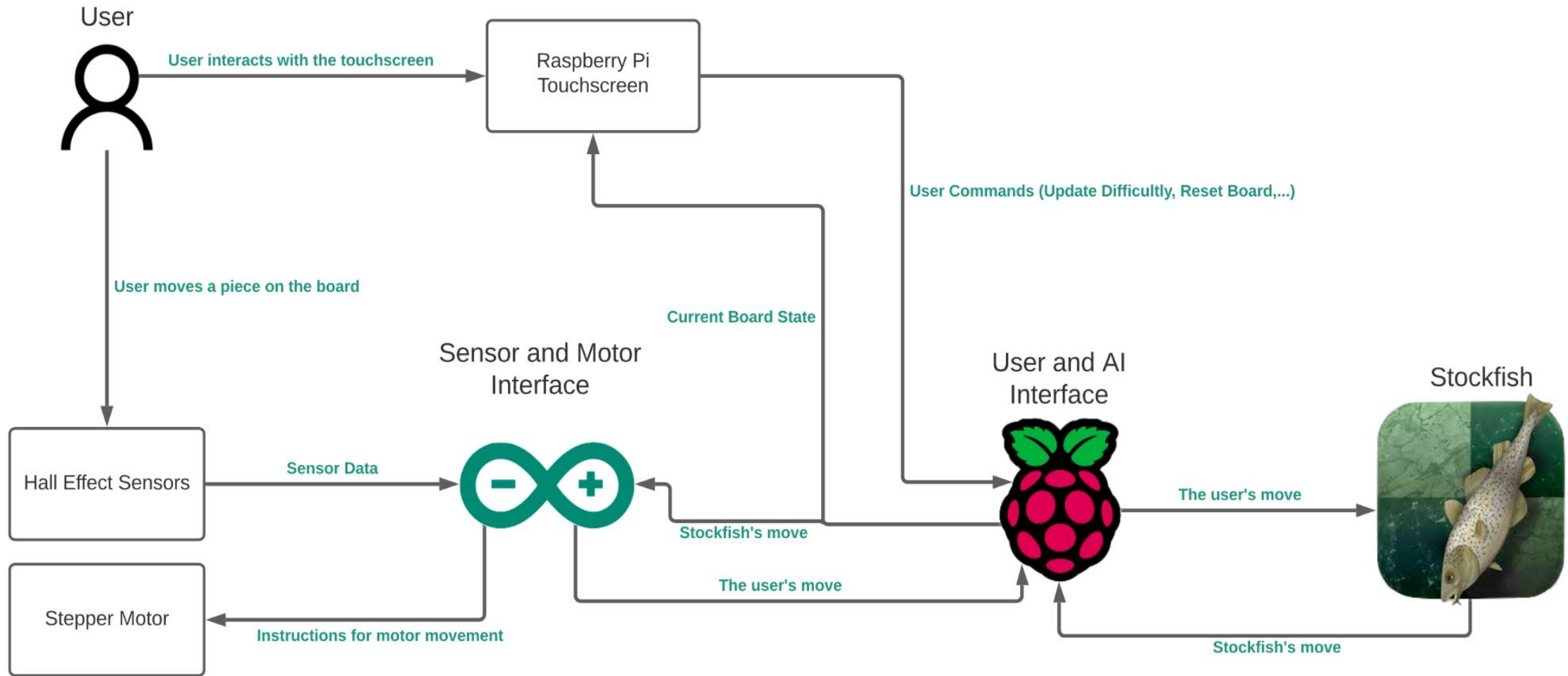
- Board can be played without requiring additional setup per user (easy to walk up and play).
 - This includes the board setting itself up prior to playing
 - Board has a display and interactive input (touchscreen) to control start, reset, settings, and difficulty functionalities.
 - Chess interface is familiar to the player; pieces are recognizable as typical chess pieces.
 - Board detects player move inputs without button presses.
 - 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).
- 
- 



Requirements Continued

- Board has reserved spots for captured pieces (“garage”) 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.
 - Stretch goals:
 - Integration with online chess services such as Chess.com or Lichess.org.
 - Simple, human-like piece movements so that the player can easily track and understand the AI move.
 - Pieces other than the knight move along the typical path (bishops along diagonals, rooks along rank/file, etc.). The knight/castling will need to route between pieces.
 - Human-like capture movements.
- 
- 

Conceptual Design Diagram





System Design Software

- On the Raspberry Pi we are using Python for both the:
 - Chess AI Interface
 - Stockfish
 - GUI
 - Tkinter
 - Chess.SVG
 - asyncio
- C++ and Python
 - Arduino
 - Sensor Polling
 - Motor Control
 - Communication with Pi over USB



Prototype Implementation GUI

Command Line Text Based Interface

```
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:
```

Raspberry PI Tkinter GUI



Prototype Implementation Hardware

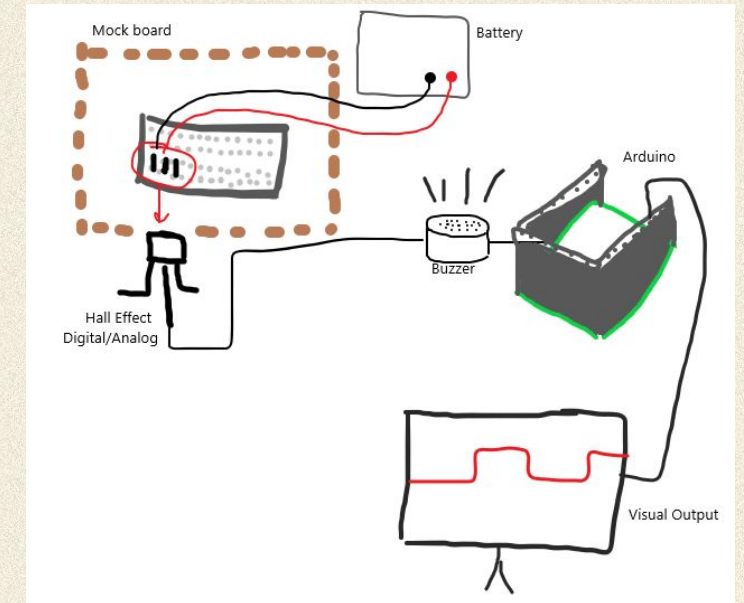
The attached video demonstrates our tests of the hall effect sensors. A prototype was built that allowed us to test each sensor and make decision as to which sensor would work best for our design. We added a buzzer to accompany



the visual hysteresis curve we see on the arduino output. As demonstrated in the videos below, hall effect sensors (both the digital/analog) worked well and to our expectations.

(Second Test Digital: <https://youtu.be/73jjqdxrCDs>)

(Second Test Analog: <https://youtu.be/4zn8xMs0mjM>)





Design Complexity

- Tracking the location of pieces on the board
 - More complex moves like queening and castling
- Resetting the board to its original state or to a save position
- Stepper motor and magnet precision
- Integrating UI, Stockfish, and physical hardware
 - Communicating via a multiplexor.





Project Plan – Task Decomposition



Hardware

- Order and test magnets and Hall Effect Sensors (Complete)
- Determine board and piece sizes
- Design and test a motor framework
- Create and test a single prototype with both sensors and the motor

Software


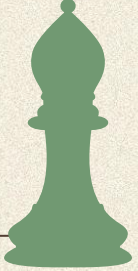
- Create a Python based GUI that will be used on the touchscreen (Complete)
- Order a Raspberry PI Touchscreen and test the GUI on the touchscreen
- Create a Python interface that will communicate with both the sensors and GUI
- Write Arduino software to control the board hardware

Final

- Combine and test the Python interface and board
 - Combine and test the Python interface and GUI
 - Combine and test the board, interface, and GUI
- 
- 



Project Plan – Risk Mitigation


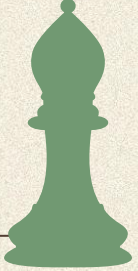
- Hall Effect Sensors will be too large and will interfere with one another when moving pieces
 - Risk Factor of 0.4
 - Interface isn't working or has a large delay
 - Risk Factor of 0.5
 - Solution: Built in displays will have a much lower chance of delay
 - Motor-Magnet System doesn't move a piece proper
 - Risk Factor of 0.75
 - Solution: Stockfish provides a function that allows us to check if a move is illegal
 - Motor System gets jammed/breaks
 - Risk Factor of 0.2
- 
- 

Project Plan - Schedule/Milestones 1st Half

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												
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													



Test Plan

- Sensors
 - To test the effectiveness of the sensors, we will be placing them under a thin glass chessboard and placing the pieces on top of the board to determine a) the maximum thickness we can have for the board, which will also be necessary for the e magnet. b) the size of each square
 - After a working gantry has been made we can tune the sensors for optimal sensing range, so players don't place pieces where the e-magnet can't retrieve them.
 - Motors
 - Torque testing: reliable movement precision on gantry with dummy weight
 - Electromagnet integration, being able to move a piece without interference
 - Sending pieces to correct locations based on manual input, and eventually interface input.
- 
- 





Test Plan cont.

- GUI
 - Interface testing involves simple testing of the chess engine to ensure that all moves are properly detected and their legality is checked.
 - Check if the touch interface on the board receives proper inputs and sends the right signals to the rest of the components.
- Full Board
 - Full board mockup tests to ensure all system components interact with each other correctly.
 - Check for board loading/resetting functionality




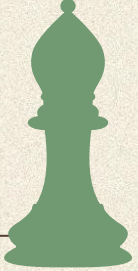


Conclusion

- Achievements
 - On the hardware side of the project we have run several successful tests on the Hall Effect Sensors, and we have ordered the stepper motors.
 - The frontend GUI for the board has been created, and we have ordered an Raspberry Pi so that we can begin setting up the touchscreen.
 - We did some basic testing on the Arduino interface, and have decided to shift that code to Python as well.
 - Next Steps
 - Next semester we will begin testing the stepper motor
 - The frontend team will test their GUI on an actual touchscreen
 - The full board integration phase will likely take the most time
- 
- 



Conclusions

- Luke Dirks
 - General frontend and touchscreen research, designed and wrote code for the GUI, documentation work and assignment tracking.
 - Jeet Nair
 - Made CAD models for the project, working on motor framework design and programming, and created project schedule.
 - Isaac Sachse
 - Worked to find and test hall effect sensors/stepper motors, documentation work, device power, general research, and professional voice acting
 - David Imhoff
 - Frontend research and implementation, project analysis and documentation, raspberry pi implementation
 - Dawson Munday
 - Researched how Stockfish is interfaced with and documentation
 - Nathan Kelly
 - Research and purchasing, testing and verification, sensors, motor drive and gantry design.
 - Nathan Bellows
 - Worked on high-level design research and initial sensor testing, and writing hardware simulator to expedite RPi software progress before a physical board is completed.
- 
- 

Thank You For Listening

