Team 10

MagiChess

Jack DeGuglielmo, Samantha Klein, Weishan Li, Sai Thuta Kyaw

Advisor: Shira Epstein

Meet the team

Shira Epstein
Faculty Team Advisor

Sai Thuta Kyaw
Electrical Engineer

Samantha Klein
Electrical Engineer

Jack Deguglielmo
Computer Engineer

Weishan Li
Computer Engineer

Problem Statement

For centuries, the game of chess has been played by two players sitting across a chessboard. The advent of digital technology in the last decades has brought virtual chess to computers and mobile phones and for the first time, this has allowed players to be anywhere across the world.

Digital chess lacks:

- *A physical aspect/satisfaction of seeing and moving your own pieces*

Physical chess lacks:

- *Ability to play from anywhere and with anyone*

Our Solution

We've decided to close the gap between physical and digital chess. To do this, we plan to create a chess board that allows users to play with an AI or a remote human opponent.

Plan:

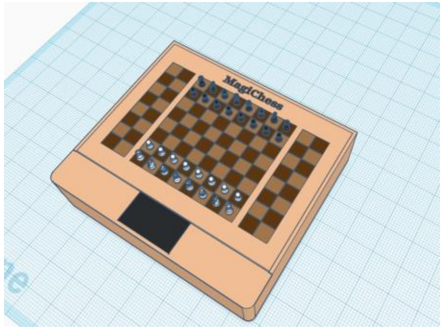
- Sense location of chess pieces on the board
- Interface with LiChess server
- Automate piece moving

Preliminary System Specifications (Design-agnostic)

- Mechanically move a piece to destination cell
- Remove/replace a piece to/from game board
- Provide visual feedback
  - Game setup, tutorial
  - Game announcements
  - Highlights previous move
- Provide audio feedback
  - Notification alerts
- Play versus remote opponent
- Playback previous games
- Includes buffer zone to store captured pieces
- Topple the King after checkmate

Preliminary System Specifications (Quantitative)

- Total system dimensions: no larger than 32.5 in x 30 in x 8in (80 cm x 74 cm x 15 cm)
- Speed of XY plotter: 5 - 8 cm/s
    - Speed increased due to better stepper drivers
    - Absolute maximum time taken for a move 25s
    - Move each pieces under 10s more than half of the time
- Weight: Under 50lbs
    - Upgrading from wood to more robust aluminium frame



Demo for Final System

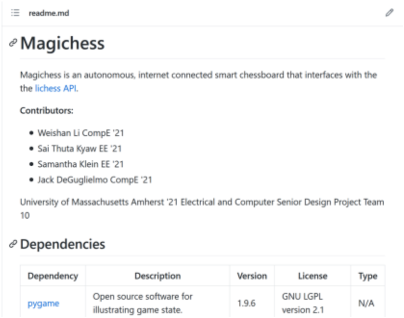
- Tour the final assembly and hardware

- Comprehensive Gameplay Demo
  - GUI Tour
  - Functionalities to be covered:
    - Capture (Movement and Sensor)
    - Physical Gantry Movements
    - Sensor Input Movements

- Game Replay Demo
  - Previous Game Replay
  - Topple King

Documentation Overview

- Communication Protocol Documents
  - Fast Scanning Communication Protocol
  - XY Gantry Communication Protocol
- System Software Documentation
  - x328p_fs_interface.py Document
  - x328p_gantry_interface.py Document
  - readme.md System Software Description
- SDP Final Report



readme.md

Magichess

Magichess is an autonomous, internet connected smart chessboard that interfaces with the the lichess API.

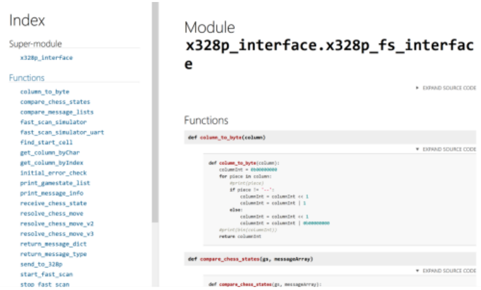
Contributors:

- Weishan Li CompE '21
- Sai Thuta Kyaw EE '21
- Samantha Klein EE '21
- Jack DeGuglielmo CompE '21

University of Massachusetts Amherst '21 Electrical and Computer Senior Design Project Team 10

Dependencies

Dependency	Description	Version	License	Type
pygame	Open source software for illustrating game state.	1.9.6	GNU LGPL version 2.1	N/A



Index

Super-module

x328p_interface

Functions

column_to_byte
compare_chess_states
compare_message_lists
fast_scan_simulator
fast_scan_simulator_uart
find_start_cell
get_column_byChar
get_column_byIndex
initial_error_check
print_gamestate_list
print_message_info
receive_chess_state
resolve_chess_move
resolve_chess_move_v2
resolve_chess_move_v3
return_message_dict
return_message_type
send_to_328p
start_fast_scan
stop fast scan

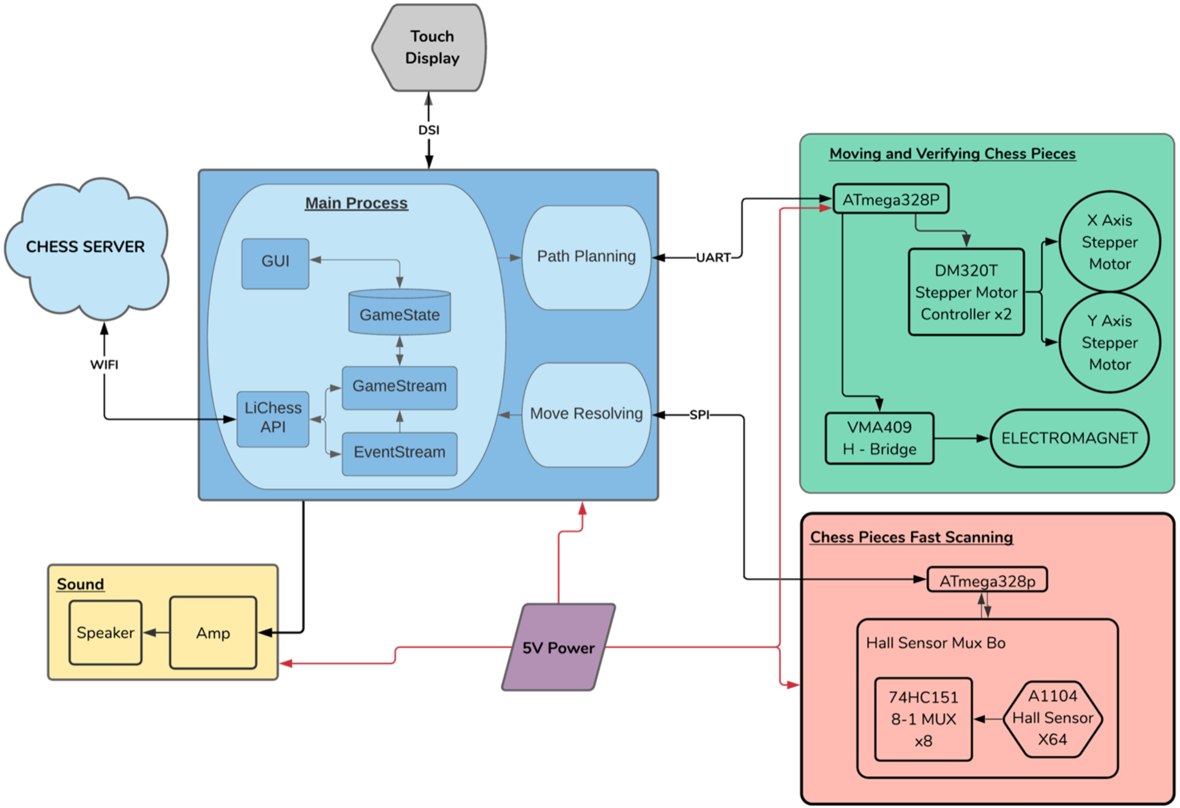
Module
x328p_interface.x328p_fs_interface

Functions

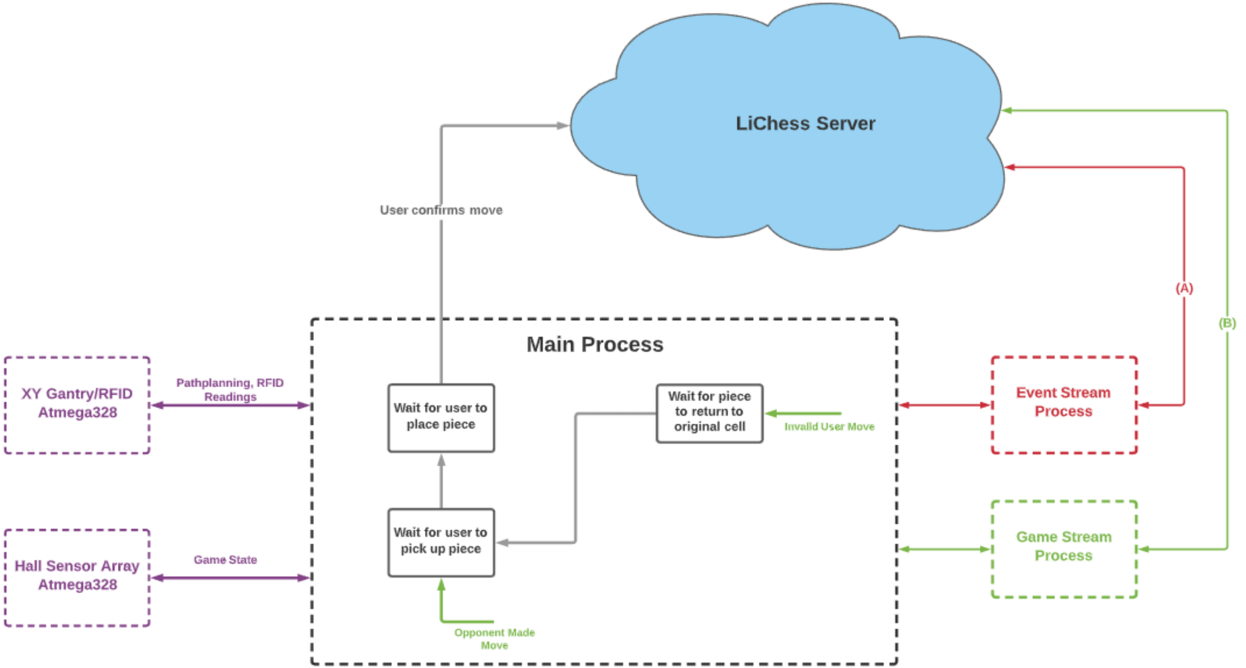
def column_to_byte(column)

def compare_chess_states(gs, messageArray)

Hardware + Software Flowchart



Software Diagram - Game State

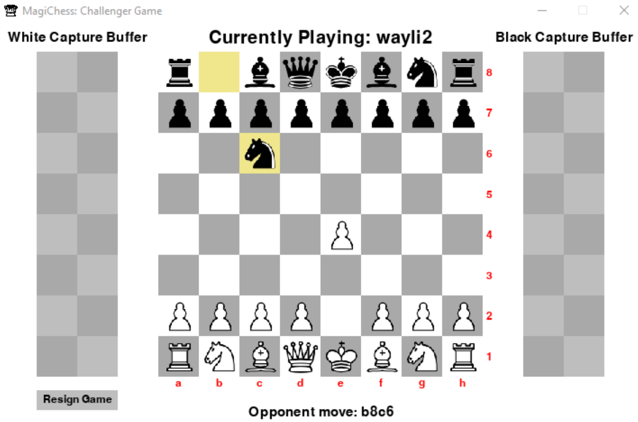


(A) gameStart, challengeDeclined, ...

(A) gameState, chatLine, ...

Final System: GUI, Gamestate, Display

- **Graphical User Interface**
  - Interaction between chessboard, Lichess server, and user
  - Housed on a Raspberry Pi touchscreen display
  - Feedback provided through audio and various visual effects
- **Software Gamestate**
  - All moves (server-side and local-side) compare with and update our software gamestate
  - Keeps track of physical pieces
  - Stays aligned with server side gamestate
  - Always on display to the user



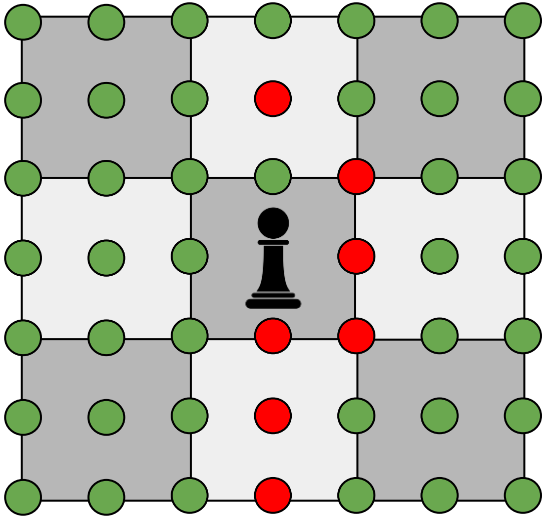
Final System: Lichess Communication

- **Lichess**
  - Open source chess server with very well documented API
  - Request and response communication
- **Gamestream and Eventstream**
  - Separate processes running simultaneously with main application
  - Constantly request data from Lichess
  - Place responses in respective queues and grab from main process

Final System: Path Planning

Path Planning Sequence

1. Translation of gamestate to position map
2. Heuristic calculation for each position state ☆
3. Execute Magichess custom path planning algorithm ☆
4. Transmit solution path to gantry MCU via UART

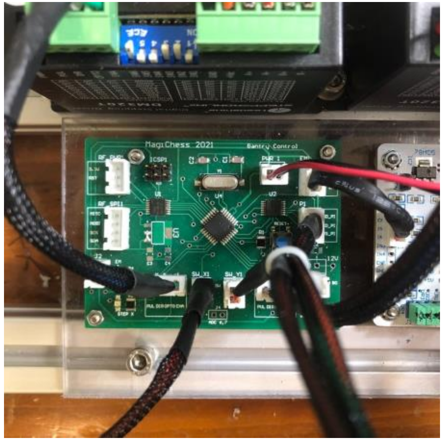


Heuristic: Straight Line Distance

☆ - Optimizations (see slides 32 and 33)

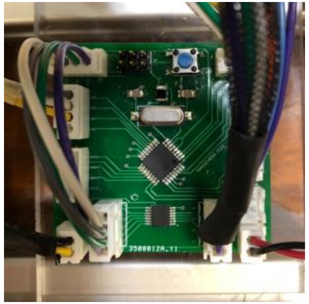
Final System: Gantry

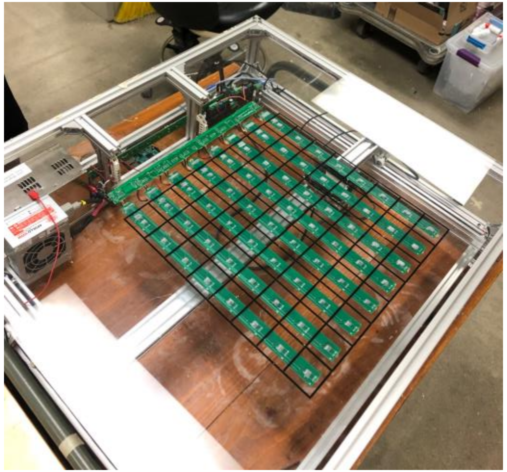
- UART data sent from Path Planning
  - 9600 baud, 8 bits, no parity
  - 3 bit type + 5 bit data
  - See Documentation for more details
- DM320T Stepper Motor Drivers
  - Rising Edge Activation + Direction Control
- LM320T based H-Bridge
  - Electromagnet control
- ATmega328p
  - Processes UART commands
  - Calculates coordinates for the next position point



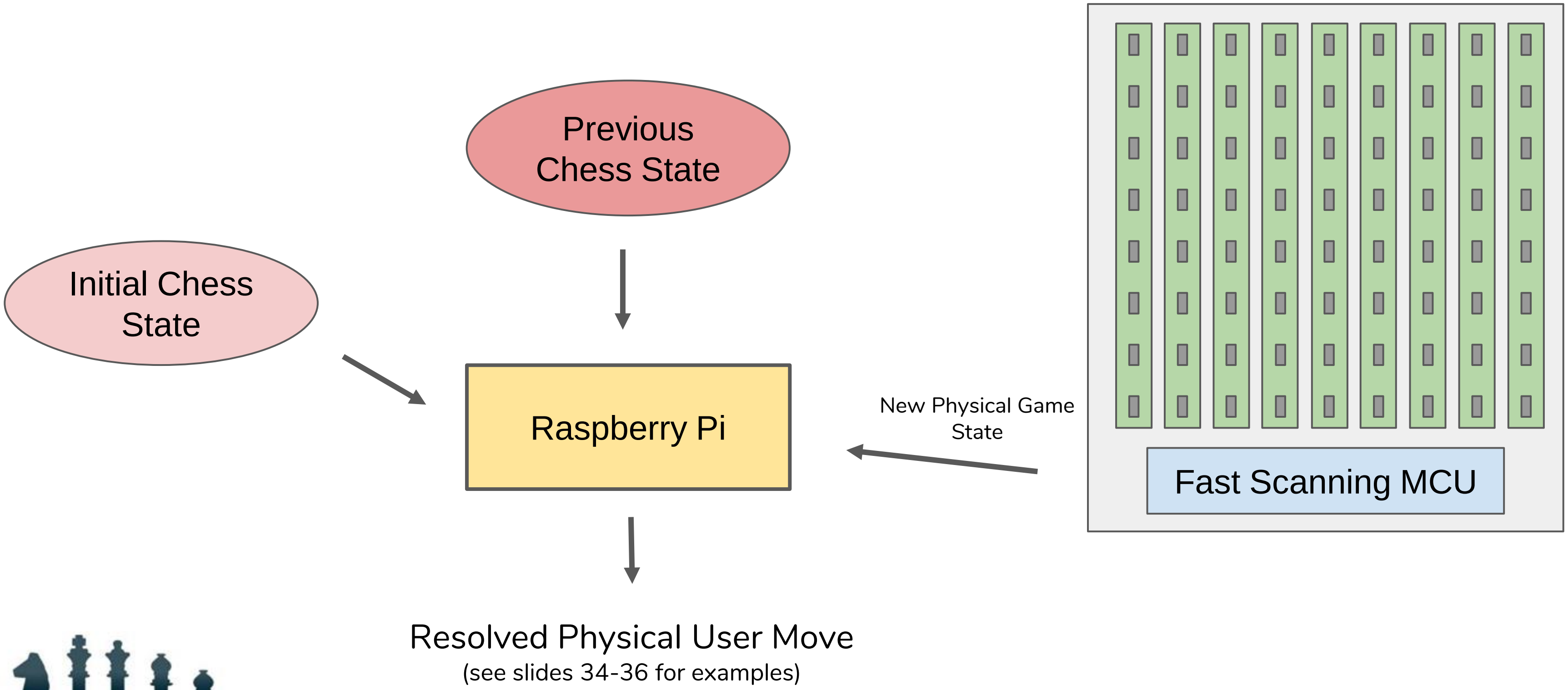
Final System: Fast Scanning

- SPI mode 0
  - ATmega328p configured as Slave
  - See SPI Communication Detail for more information
- 328p waits for prompt from Pi to send data for a specific column (a-h)
  - Pi sends a request for scanning each column
  - 328 responds with data from corresponding column

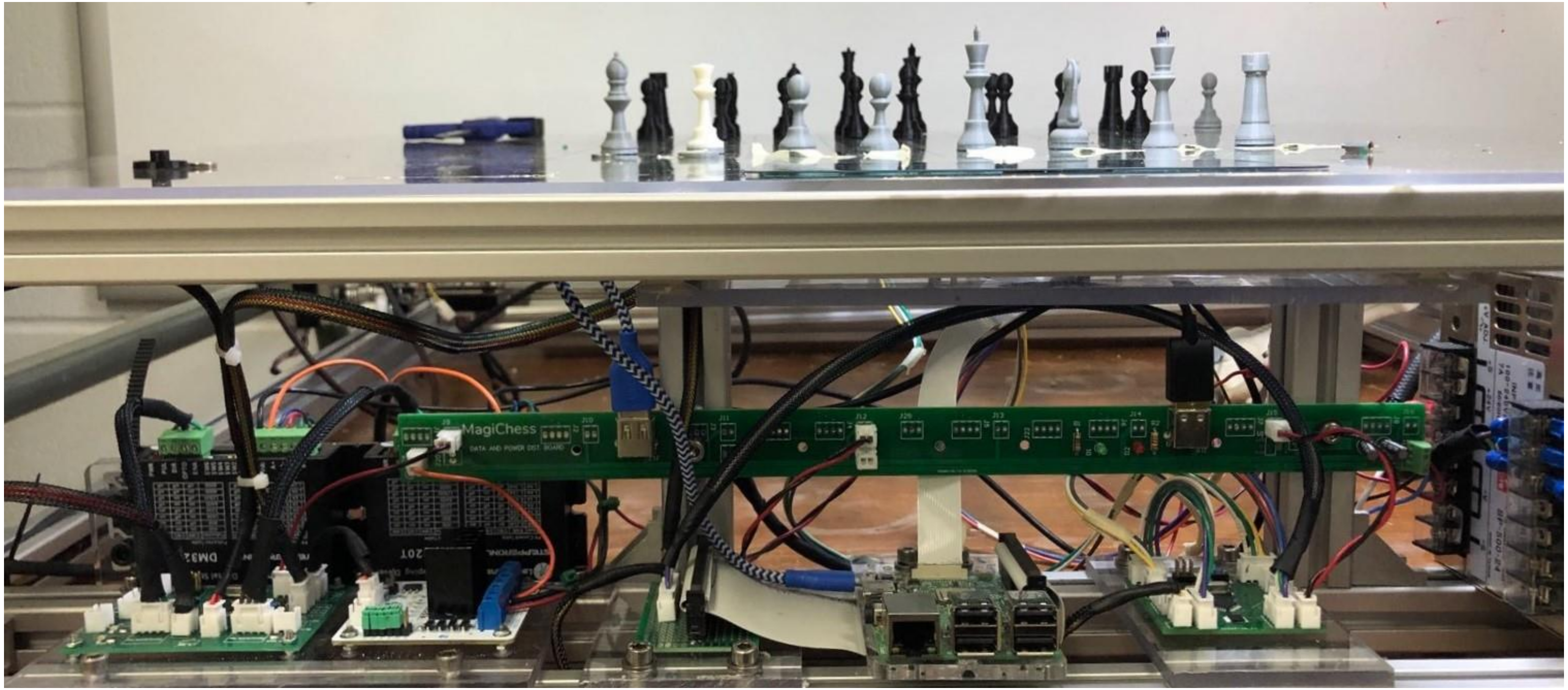




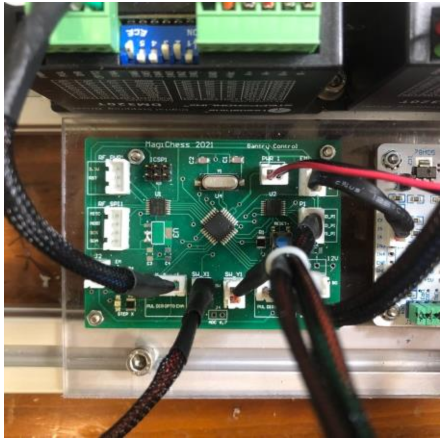
Final System: Move Resolution

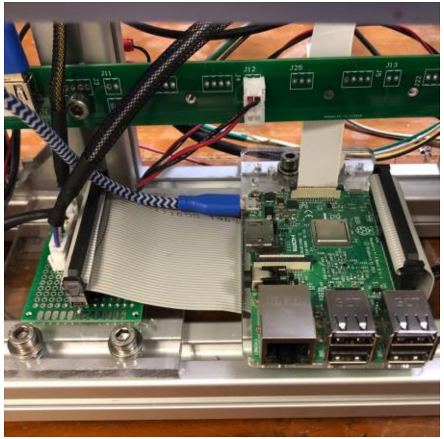


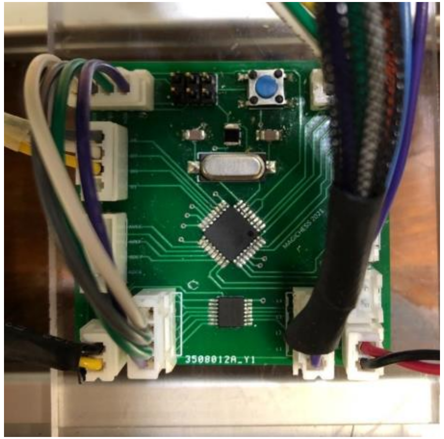
Final system Overview:



Integrated PCB Close Up







FPR Accomplishments

- Fast Scanning
  - Resolve consecutive moves
  - Resolve capturing, castling, etc.
  - Full integration with gantry system
- Movement
  - Optimized for the final system
  - Self-Calibrate and communication with Pi
- GUI
  - Integrated with different subsystems
  - Visual and audio feedback

FPR Accomplishments

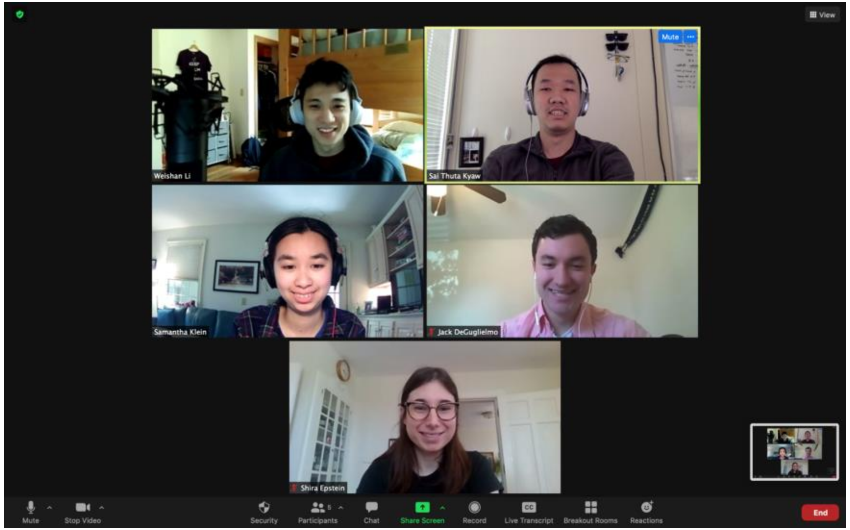
- ✓ Mechanically move a piece to destination cell
- ✓ Remove/replace a piece to/from game board
- ✓ Provide visual feedback
  - ✓ Game announcements
  - ✓ Highlights previous move
  - ✓ Audio feedback
- ✓ Notification alerts
- ✓ Play versus remote opponent
- ✓ Playback previous games
- ✓ Includes buffer zone to store captured pieces
- ✓ Topple the King after checkmate

Total Spending

Total Spending (Fall Semester)	264.83
Stepper Motor Drivers	40.1
Prototype (DigiKey)	29.49
JLC PCB	49.36
PCB Population (DigiKey)	57.19
PCB Population (Newark)	41.84
Budget Extension	+150
Post-CDR	112.19
Total Spent	**595**
Remaining Budget	**55**

Fun Facts!

- 380+ Commits on GitHub
- 3.75 hrs/week of group + advisor meetings



External Links

[Team Website]

[All Demo Videos Playlist]

[Github Repo]

Next Steps...

- Functionalities for SDP Demo Day
  - Sensing Physical User Edge Case Moves (castling, en passant)
  - Autonomous Board Reset
  - Resume Ongoing Games
- Future Direction and Works
  - RFID for piece identification or removal of PCB copper plane (see slide)
  - Sensing Physical User's Promotion
  - Tutorial/Video on how to play and use the Magichess system

Thank You