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:

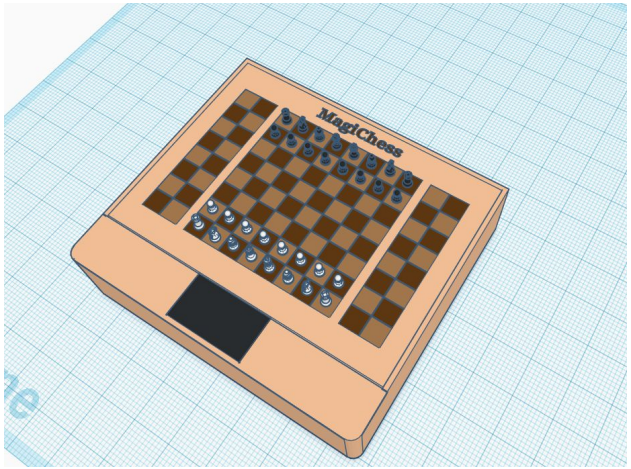
- Sense location of chess pieces on the board
- Interface with LiChess server
- Automate piece moving

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
  - Where pieces can move after picked up
- 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 33.5 in x 28 in x 8in (85cm*68cm*20cm)
- Speed of XY plotter: 4-5 cm/s
  - Max time for a move: 15s, x2 for capture and transport
  - Average time for a piece to move: 7.5s
- Weight: Under 25lbs

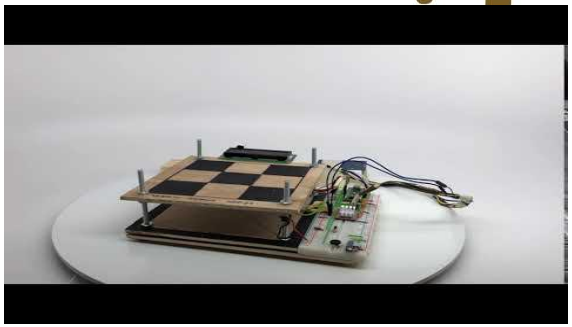


Proposed MDR Deliverables (From PDR)

- LiChess integration with Raspberry Pi (initiate games, execute moves, etc.)
- GUI prototype for Raspberry Pi touch display
- Raspberry Pi outputting digital communication protocols
- Assembly and movement of XY plotter (gantry)
- Results from RFID multiplexing / Hall Effect sensor testing
    - Decision on which sensing technique to use

MDR Deliverables: Sensing Technique

- Initial Testing
  - Magnet Sizes
  - Sensing Methods
  - RFID Test
  - Electromagnet
- Final Product
- Testbench
  - Mimics Gantry (EM moving on a plane)
  - Height Adjustable
  - Controls Electromagnet
  - Sensors and Display



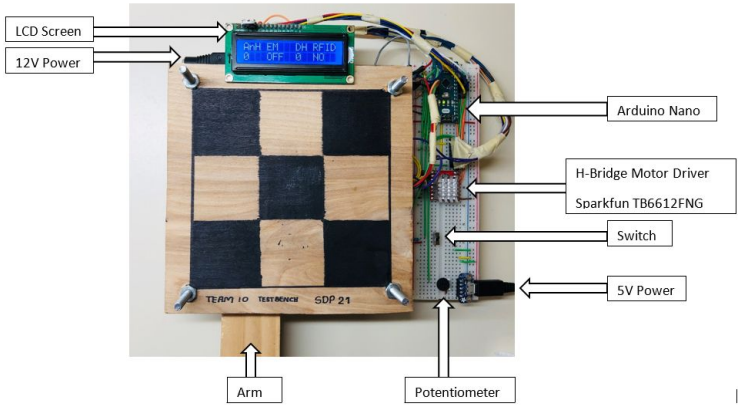
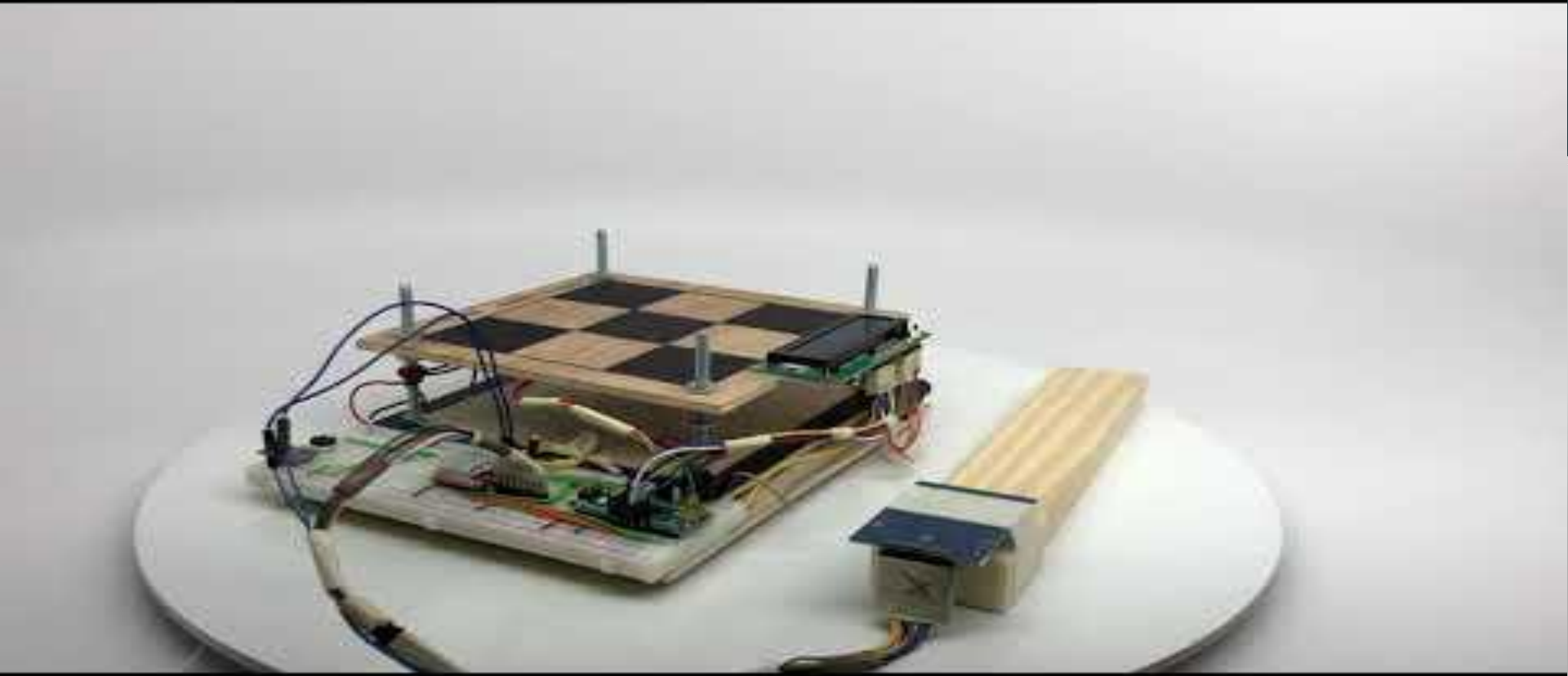


Figure 1: Testbench Setup

Link to code: https://github.com/degugi/IichessTesting/blob/master/arduino/



Chess Pieces Sensing Report

Three different proposed of chess pieces sensing

- Method 1: Use multiplexed RFID antennas across 64 cells.

- Method 2: Use six different strength magnets with different poles to identify pieces.

- Method 3: Use magnets to sense occupied cells and single RFID reader to identify

1. RFID Only Sensing Solution

- Multiplexing RFID Antennas with single RFID Reader Chip
  - Research based on RFID Multiplexer Example from Texas Instrument
  - Ordered the Parts and Breakout Boards
  - Mismatched Breakout Boards for RF Switch
    - Same package (QFN16) different size
    - Breakout board datasheet incomplete

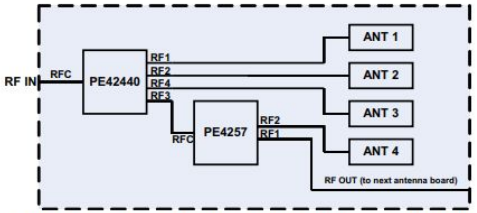


Figure 4. PE42440 and PE4257 Antenna Circuit Arrangement Detail Diagram

- 64 RFID Readers on single TWI interface
  - AT88RF1354 Atmel TWI RFID Reader Chip
  - Two possible addressing (need at least 8)

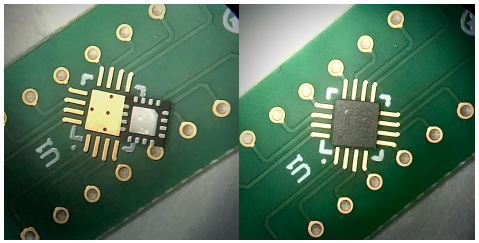
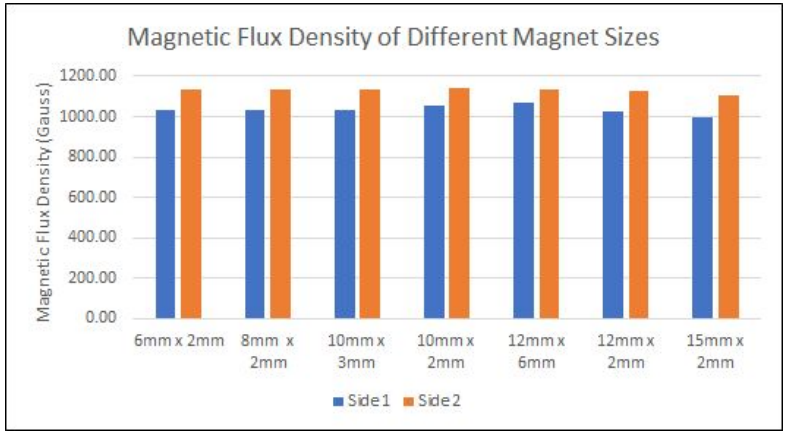


Figure 4: Antenna Circuit Arrangement Adapted from TRF7960A RFID Multiplexer Example, retrieved from https://www.ti.com/lit/an/sloa167/sloa167.pdf

2. Hall Effect Sensor Only

Method 2 : Use six different strength magnets with different poles to identify pieces.

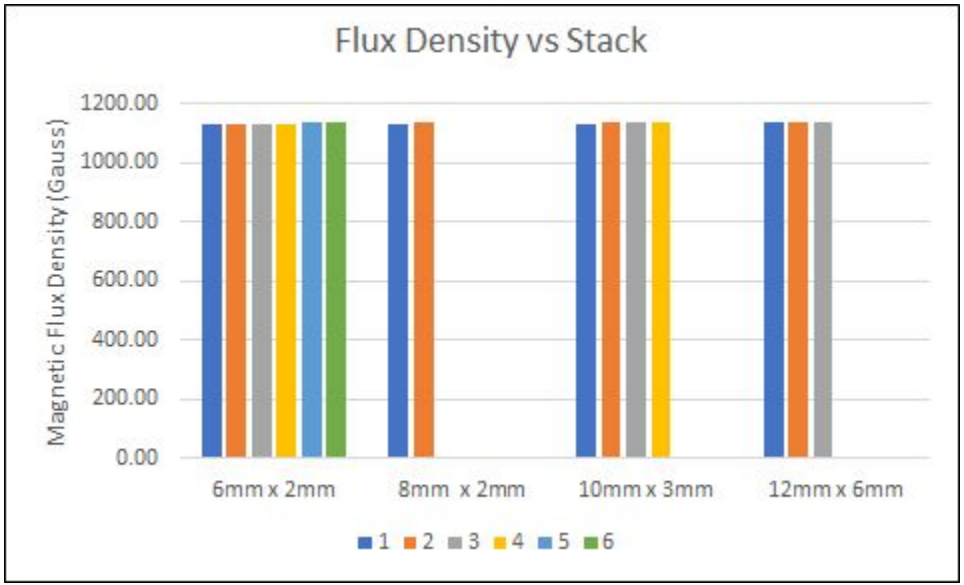
- Know what physical properties of the magnet the sensor can measure



- Analog Hall Sensor SS49E
  - Magnetic Flux Density (Gauss) in terms of Voltage

How to vary reading on Analog Hall Sensor?

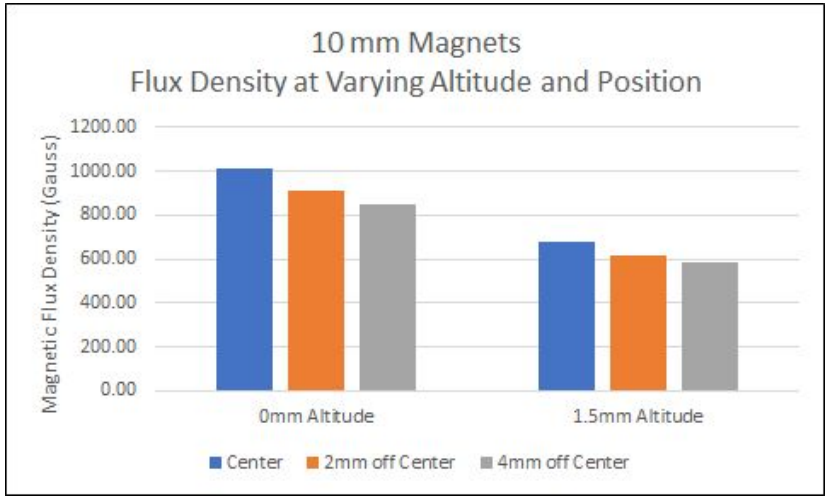
- Will Stacking Magnets Increase Reading on Sensor?

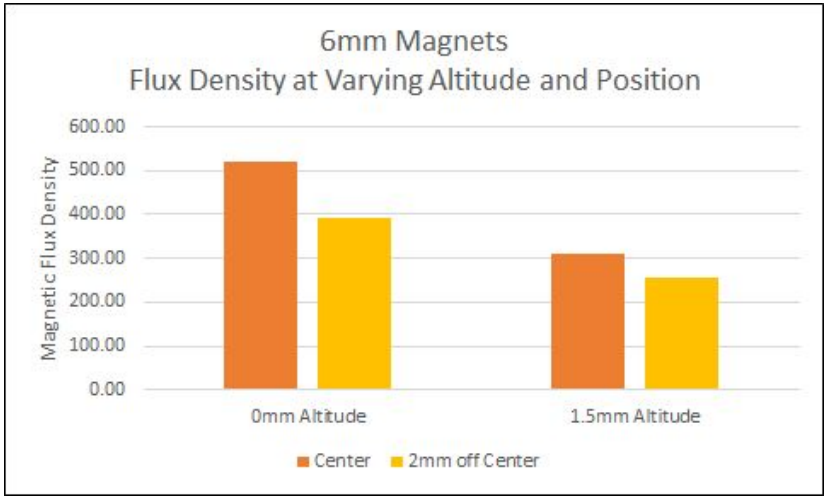


How to vary reading on Analog Hall Sensor?

- Changing the position varies the reading on the Analog Hall Sensor

- Varied the height of the magnet relative to the base





- The Magnetic Flux Density changes with Altitude.
- Tolerance for pieces off-centered is acceptable
- Not the best fit
  - Need 6 different altitudes
  - The higher the altitude, the harder the electromagnet to attract

3. Hall Sensor & RFID Hybrid

- Hall Sensor for fast scanning of ChessBoard
  - Monitor movements of Chess Pieces by user
  - Fast Scanning - Up to 1MHz
  - Software keeps track of chess piece location
- RFID Tags for verification
  - RFID reading works with electromagnet turned on
  - Selectively verify chess pieces
    - Before every game
    - Before and after each move

Sam’s Individual Report: MDR deliverable

MDR Deliverable:

- Results of Multiplexing tests

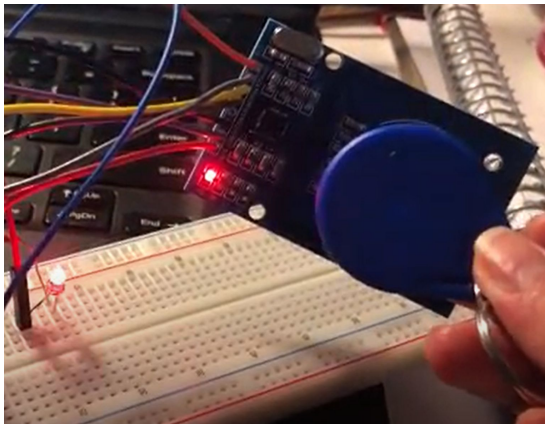
Result of testing (In collaboration with Sai):

- Hybrid method
  - 64 multiplexed hall sensors
  - One RFID reader and antenna for the electromagnet

Sam's Individual Report pt. 2: Post PDR Accomplishments

RC522:

- Got one working (tested w/ LED)

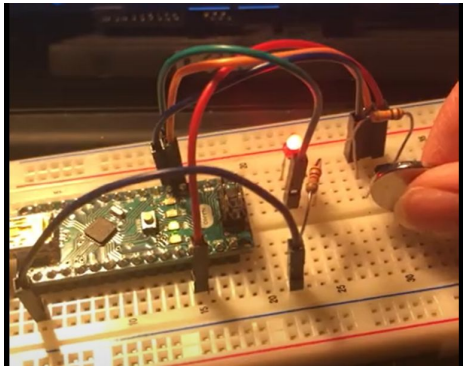


I2C bus:

- Considered creating a bus with RFID readers rather than multiplexing
- Began drafting the bus

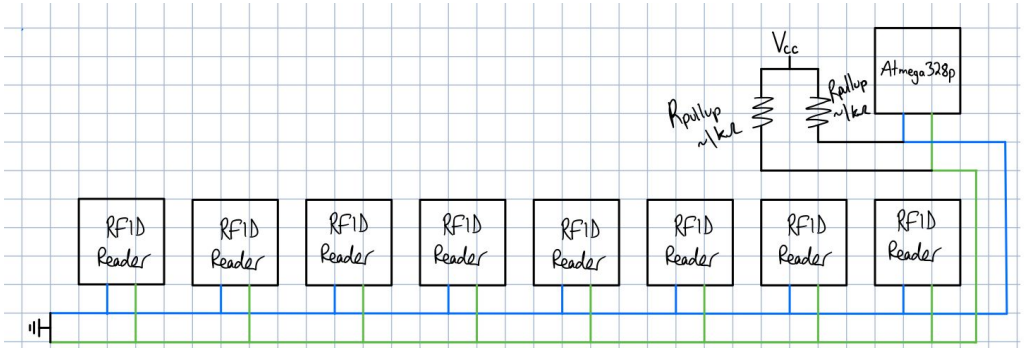
Hall Effect Sensors:

- Got one working (tested w/ LED)
- Multiplexed 4 of them



Sam's Individual Report: I2C bus

- Fast Mode
  - $C_b \leq 400$ pF
  - $f_{CLK} \leq 400$ kHz
  - $t_{rise} \leq 300$ ns
- Pull Up Resistor:
  - $R_{max} = t_{rise} / C_b = 300ns / C_b$
  - $R_{min} = (V_{CC} - V_{0L}(max)) / (I_{0L}) = (3.3\text{-}0.4V) / (.3\ mA) = .966\ k\Omega$
- Bus Capacitance
  - If $C_b \geq 400$ pF, a buffer must be added
  - Atmega328 : 10 pF
  - Reader (AT88RF1354-ZU): ??
    - Only 2 possible address locations

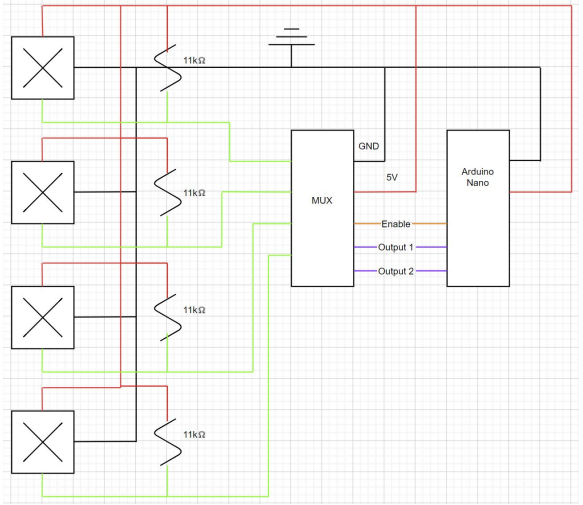


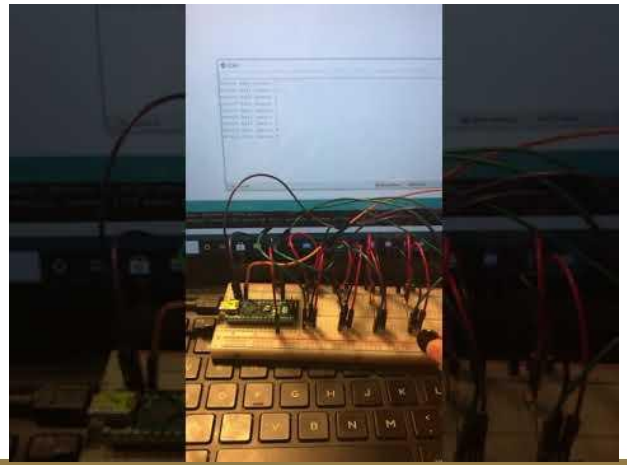
I2C bus with 8 readers and no buffer

Sam’s Individual Report: Demo!

Hall Sensor Multiplexing

- Used the SN54HC157 from Texas Instruments
  - Quadruple 2-1 Mux
- Multiplexed 4 hall effect sensors
  - Used 2 of the 4 2-1 muxes on the IC
  - Verified using the Arduino serial monitor





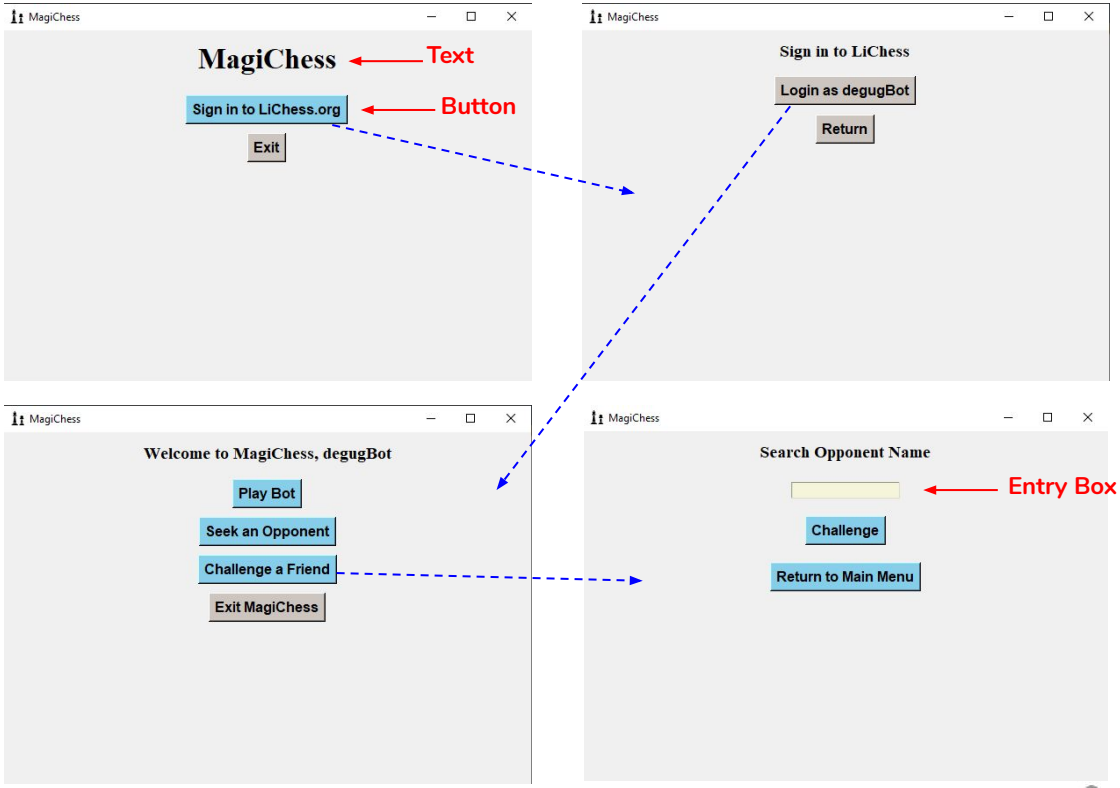
Link to code: https://github.com/degugi/lichessTesting/blob/master/arduino/

Software - Video Demonstration

Graphical User Interface

Python ***tkinter*** library

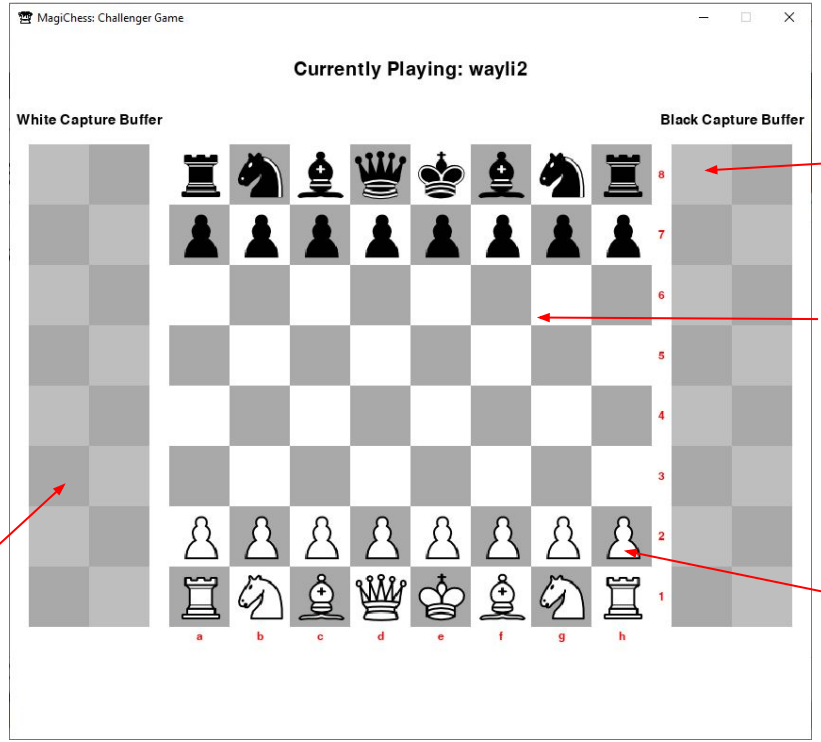
- Frames
- Widgets



Game Window

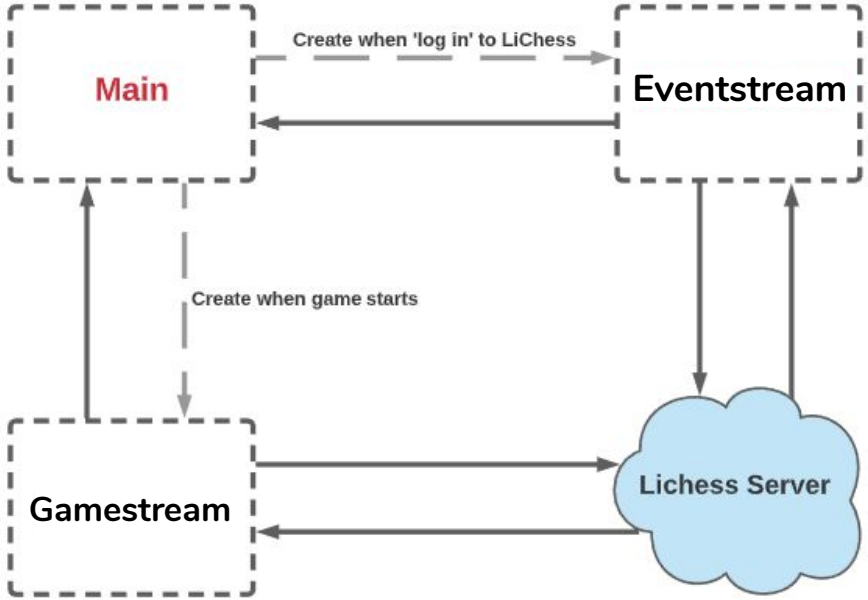
Python **pygame** library

- Draw objects
- Load images
- Update and reflect local gamestate



Event Stream, Game Stream

Python **multiprocessing** library
- Communication
- Event Stream
- Game Stream



LiChess API

Python **requests** library

- LiChess interface file
  - Create streams
  - Make challenges
  - Make moves

Example challenge request:

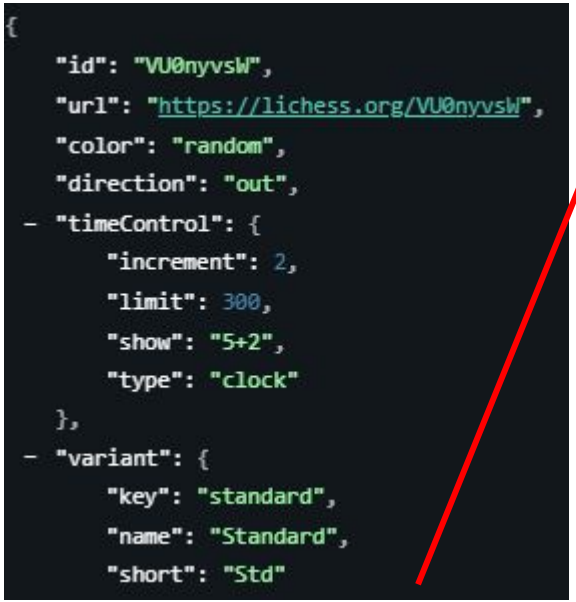
```
r = requests.post('https://lichess.org/api/challenge/' + username, json=configurations, headers={'Authorization': 'Bearer {}'.format(api_key)})
print(r.content)
# check for successful challenge response
if r.status_code == 200:

    # response message from challenge request to LiChess
    json_response = r.json()
    gameid = json_response["challenge"]["id"]
    return gameid

# user was not found
else:
    return 0
```

Example game stream request response:

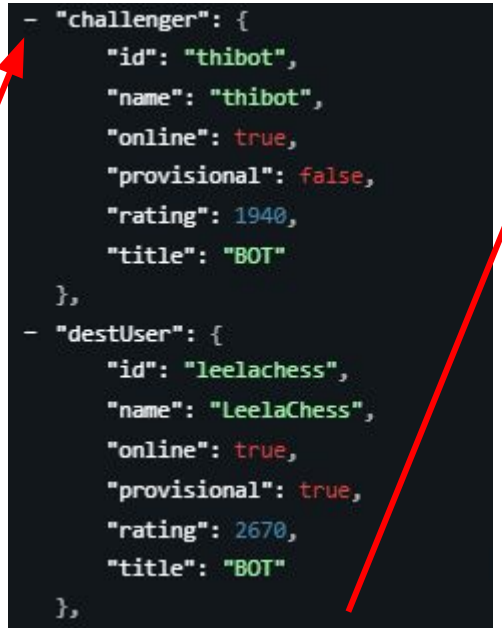
```
{
    "type": "gameState",
    "moves": "e2e4 c7c5 f2f4 d7d6 g1f3 b8c6 f1c4 g8f6 d2d3 g7g6 e1g1 f8g7 b1c3",
    "wtime": 7598040,
    "btime": 8395220,
    "winc": 10000,
    "binc": 10000,
    "status": "started"
},
```

LiChess API

Example challenge request response:



```
{
  "id": "VU0nyvsW",
  "url": "https://lichess.org/VU0nyvsW",
  "color": "random",
  "direction": "out",
- "timeControl": {
      "increment": 2,
      "limit": 300,
      "show": "5+2",
      "type": "clock"
  },
- "variant": {
      "key": "standard",
      "name": "Standard",
      "short": "Std"
```

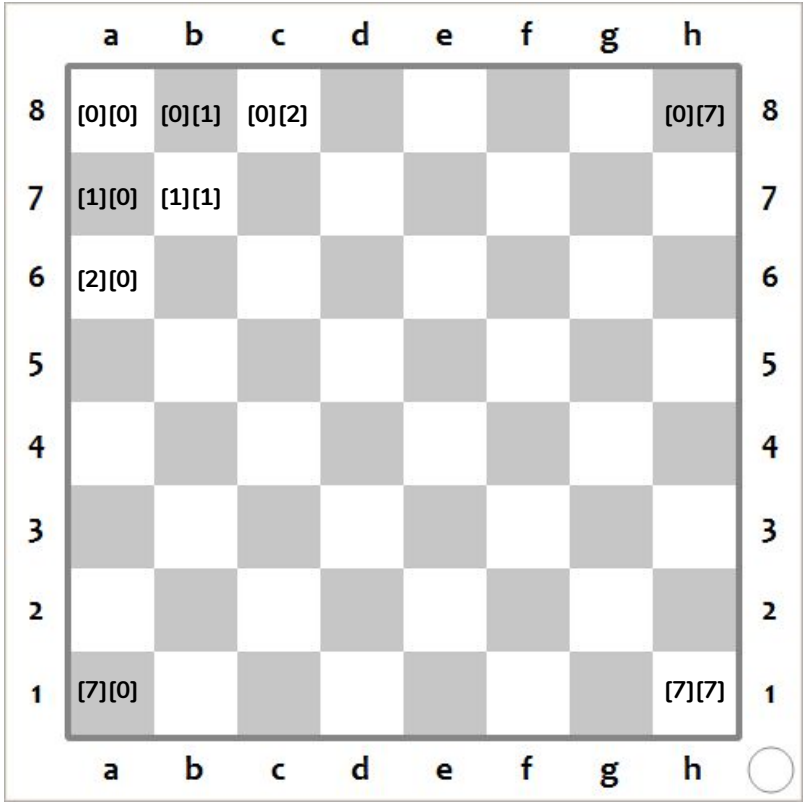


```
- "challenger": {
      "id": "thibot",
      "name": "thibot",
      "online": true,
      "provisional": false,
      "rating": 1940,
      "title": "BOT"
  },
- "destUser": {
      "id": "leelachess",
      "name": "LeelaChess",
      "online": true,
      "provisional": true,
      "rating": 2670,
      "title": "BOT"
  },
```

```
- "perf": {
      "icon": ";",
      "name": "Correspondence"
  },
  "rated": true,
  "speed": "blitz",
  "status": "created"
}
```

Local Gamestate

- GameState object w/ *self.board*
- If user is white side:
  - A >> **[-][0]**, H >> **[-][7]**
  - 1 >> **[7][-]**, 7 >> **[0][-]**
- If user is black side:
  - A >> **[-][7]**, H >> **[7][-]**
  - 1 >> **[0][-]**, 7 >> **[-][0]**



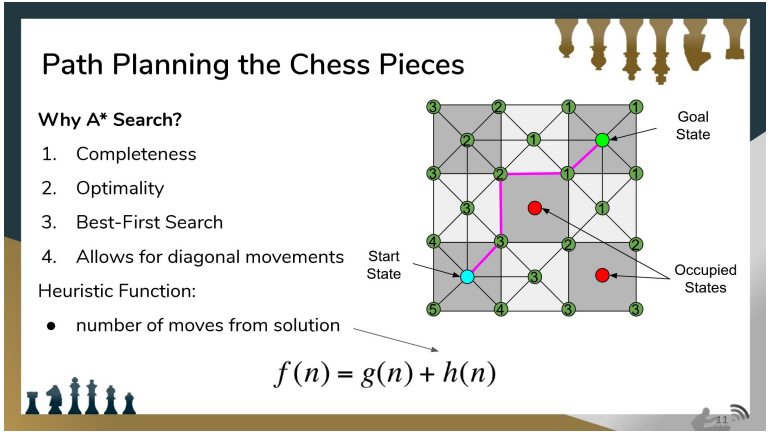
GUI / Game
Demo

Updates on Chess Piece Path Planning

- Progress Updates
  - Developed position map structure
  - Straightline heuristic implementation
  - Transitioned to greedy best first search

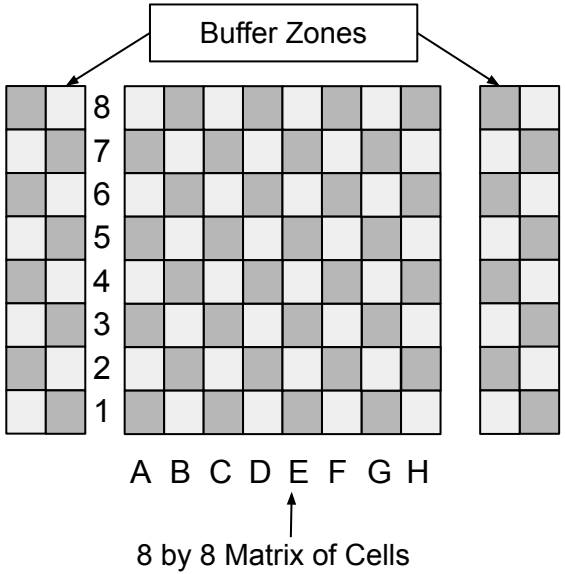
Path Planning Sequence

1. Translation of gamestate to position map
2. Heuristic calculation for each position state
3. Greedy Best First Search
4. UART Transmission of path to 328P

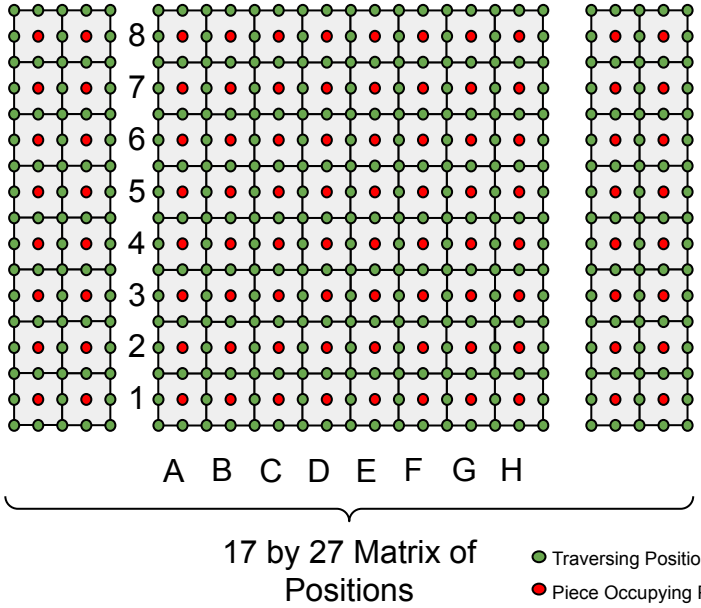


Translation of Gamestate

8x8 Gamestate with Buffers



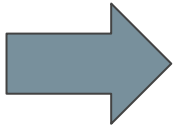
Position Map



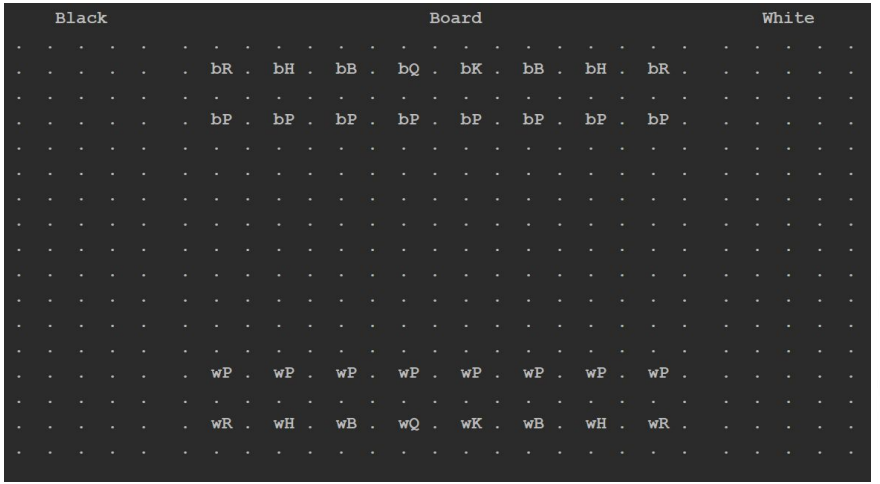
Translation of Gamestate (python)

8x8 Gamestate

```
['bR', 'bH', 'bB', 'bQ', 'bK', 'bB', 'bH', 'bR']
['bP', 'bP', 'bP', 'bP', 'bP', 'bP', 'bP', 'bP']
['--', '--', '--', '--', '--', '--', '--', '--']
['--', '--', '--', '--', '--', '--', '--', '--']
['--', '--', '--', '--', '--', '--', '--', '--']
['--', '--', '--', '--', '--', '--', '--', '--']
['wP', 'wP', 'wP', 'wP', 'wP', 'wP', 'wP', 'wP']
['wR', 'wH', 'wB', 'wQ', 'wK', 'wB', 'wH', 'wR']
```



Position Map



Heuristic Calculation

Straight Line Distance

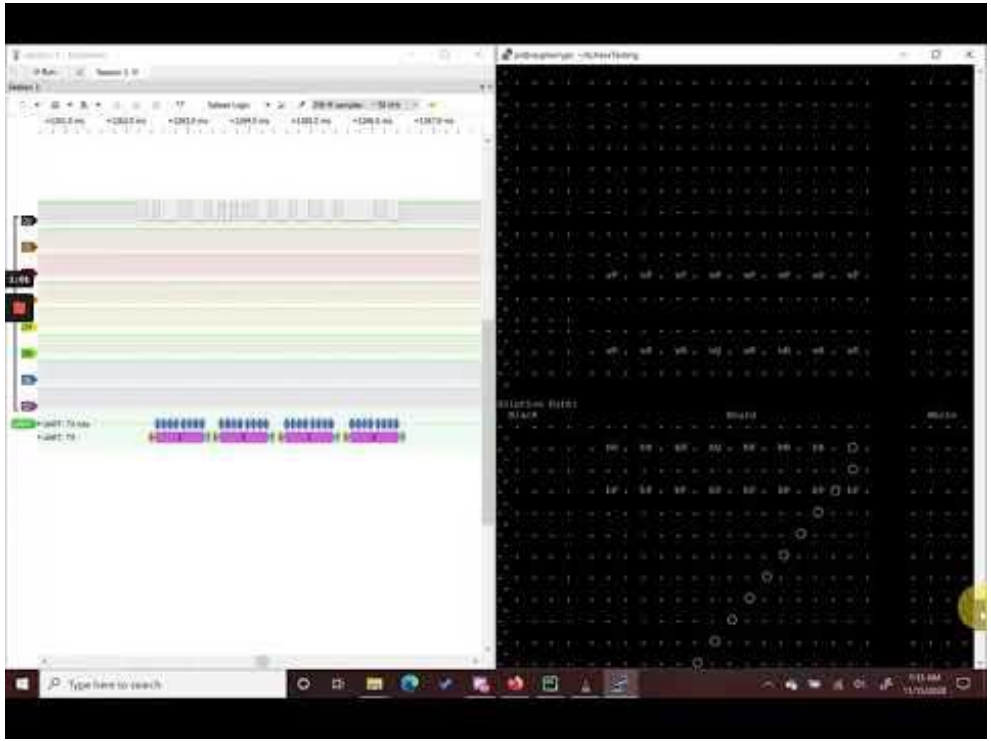
Each position state's heuristic is calculated manually using the distance formula relative to the goal state

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
straightLineDist = math.sqrt(math.pow(endPos[0]-i,2) + math.pow(endPos[1]-j, 2))
posMap[i][j].heuristic = straightLineDist
```

Greedy Best First Implementation

- Why Greedy Best First over A*?
  - Better computational and space complexity
  - Yields similar results to A* (near optimal)
    - Paths are mostly straight lines with small variances
  - Overlaying our own constraints will be more manageable
    - Diagonal move constraints for magnets

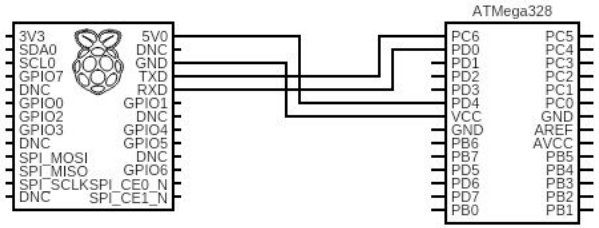


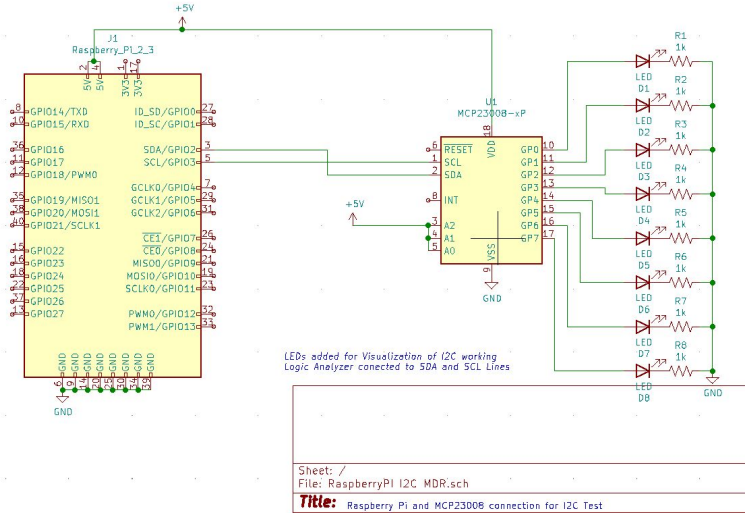
Path Planning with UART Communication Demo

Raspberry Pi Digital Communication

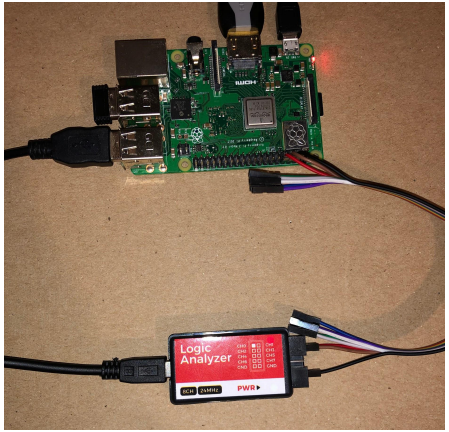
MDR Deliverable: Raspberry Pi outputting digital communication protocols

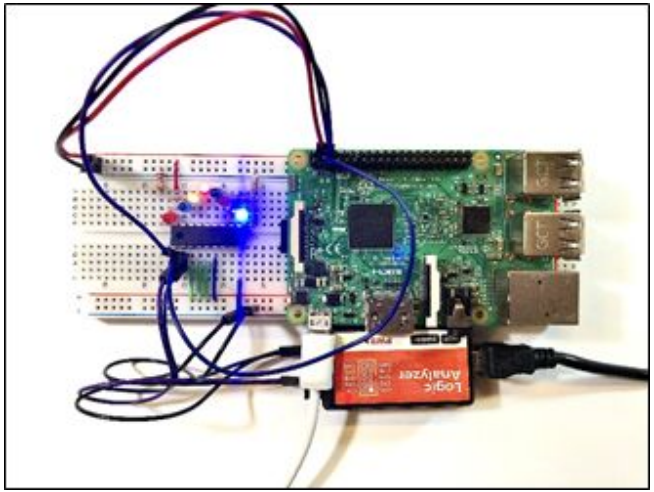
- Accomplishments:
  - UART Communication with Atmega328P
  - I2C Communication with MCH23008
  - Live Demo of Path Transmission via UART

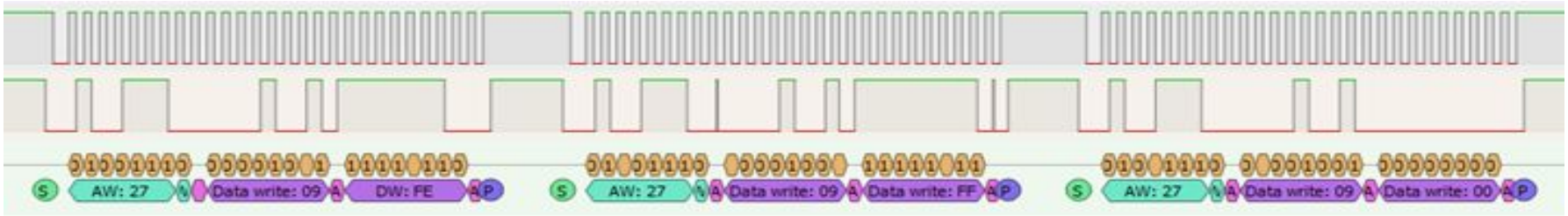




Evaluation Setup



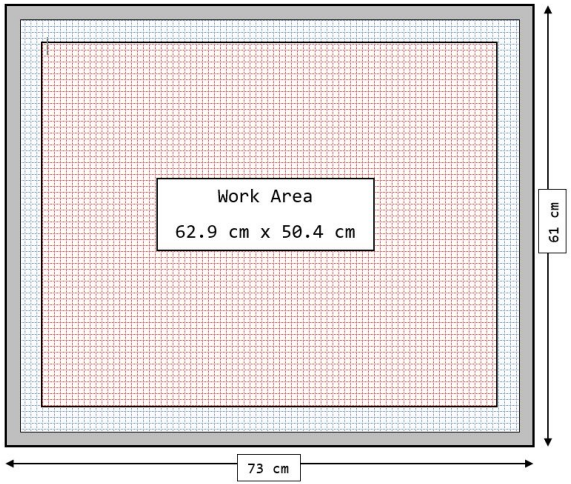




MDR Deliverables: Gantry

Assembly and movement of XY Plotter Gantry

- Specifications
  - Dimensions 73 cm x 61cm
  - Working Area 62.9cm x 50.4cm
  - Gantry Speed
    - 4cm/s @ 60rpm
    - 5cm/s @ 75rpm
  - Stepper Motor
    - 200 steps/rev
    - 50 steps/cm



Link to code: https://github.com/degugi/lichessTesting/blob/master/arduino/

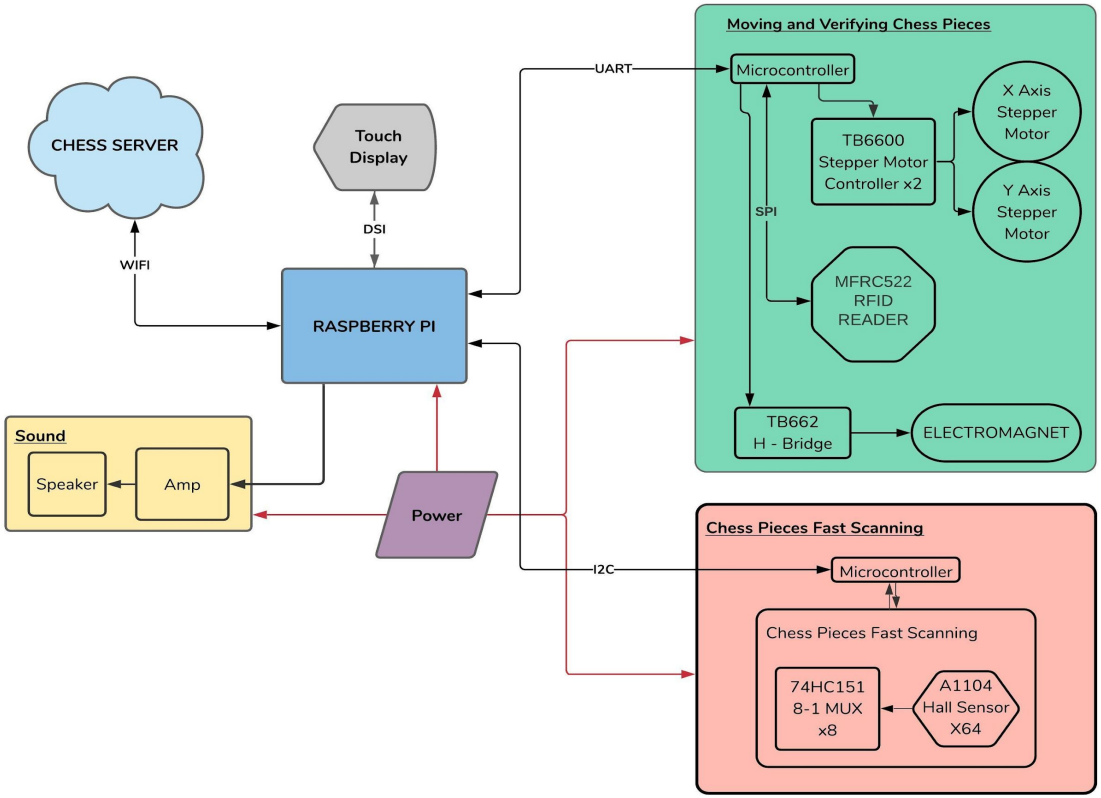
TEAM 10

GANTRY DEMO

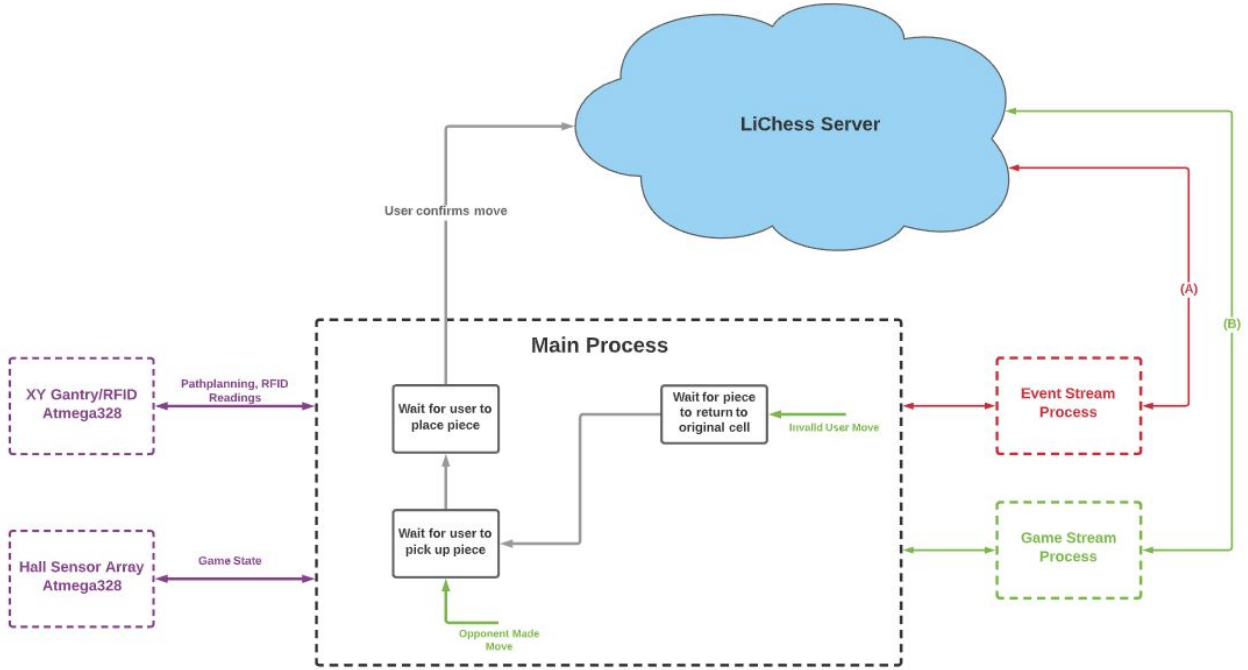
MDR Deliverables

- ✓ Results from RFID multiplexing / Hall Effect sensor testing
  - ✓ Decision on which sensing technique to use
  - ✓ Working Subsystems
- ✓ LiChess integration with Raspberry Pi (initiate games, execute moves, etc.)
- ✓ GUI prototype for Raspberry Pi touch display
- ✓ Path Planning with Greedy BFS
- ✓ Raspberry Pi outputting digital communication protocols
- ✓ Assembly and movement of XY plotter (gantry)

System Block Diagram



Software Diagram - Game State



(A) gameStart, challengeDeclined, ...

(B) gameState, chatLine, ...

Hardware Plan for FPR

- Custom PCB to mount Hall Sensors underneath the ChessBoard
- 2 PCBs
  - Fast Scanning with Hall Sensor
  - Moving and Verification of Chess Pieces.
- PCBs should have
  - Microcontroller, Power Input, ICSP Headers, Connectors, Protection Circuit, Relevant Components
- Raspberry Pi, Stepper Motor Drivers, RFID Reader, H-Bridge

Total Spendings

Items Purchased (Digikey)	Cost ($)
16QFN breakout board	12
P25/20 Electromagnet	9.95
13MHz NFC antenna (10)	12.32
Tariff for NFC antenna	1.23
32QFN Breakout board	5.95
AH3564 Hall Sensors (5)	3.58
Tariff for Hall Sensors	0.36
PE4244 RF switches (4)	4.8
TI7960 RFID reader 32QFN (3)	14.46
At88RF RFID tag (10)	8.6
AT88RF RFID reader (5)	9.85
USPS Shipping	4.99

Items Purchased (AliExpress)	Cost ($)
Gantry	162.34

Items Purchased (eBay)	Cost ($)
36QFN breakout board (5)	9.25
USPS Shipping	5.15

Total Spent	264.83
Remainding	235.17

Estimated Costs for Next Semester

Item	Cost total ($)
Hall Sensor PCB (x10)	15.5
MCU PCB	4
Gantry Control PCB	4
Hall Sensors (x60)	109.53
Mux (x8)	5.6
Motor Controllers (x2)	26
Magnets	11
Total	175.63

List of Hardware

Hardware	MDR	FPR
Single Board Computer	Raspberry Pi	Raspberry Pi
Hall Sensors	SS49E/ A1104LUA-T	A1104LUA-T
Multiplexor	SN74HC157N	SN74HC151DR
Electromagnet	Adafruit P25/20	Adafruit P25/20
Stepper Motor	NEMA14	NEMA14
Stepper Motor Controller	Velleman VMA409	TB6600
H-Bridge	SparkFun TB6612FNG	SparkFun TB6612FNG
RFID Reader	MFRC522	MFRC522
MCU	Arduino NANO	ATmega328p

List of Software

- ATmega328p code
  - Gantry Control, Look up tables, reading rfid and digital communication
  - Fast Sensing, Signal Multiplexing, Mapping and digital communication
- Python modules:
  - GUI
  - Game / Event Stream
  - Pygame Game Window
  - LiChess Interface
  - GameState / Position Map
- Supporting libraries:
  - *tkinter*
  - *multiprocessing*
  - *pygame*
  - *requests*

Technical Responsibilities

Jack

- Raspberry Pi interface with 328Ps
- Board scanning and piece verification
- Autonomous game playback
- Altium Lead

Weishan

- Improving and refining GUI
- Interfacing between Raspberry Pi and 328Ps
- Autonomous game playback

Sam

- Hall Effect Sensor Multiplexing
- Budget Manager

Sai

- Movement of Chess Pieces
- RFID Verification
- Team Coordinator

Gantt Chart Until CDR

Task	Team Member	Dec 6 - Dec 19	Dec 27 - Jan 16	Jan 17 - Jan 30	Jan 31 - Feb 6	Feb 7 - Feb 13	Feb 14 - Feb 20	Feb 21 - Feb 27	Feb 28 - Mar 6
Path Planning	Jack								
Gantry Integration	Jack								
Display Integration	Wei								
Visual/ Audio Notification	Wei								
Fast Scanning Integration	Wei								
Fast Scanning w/ Hall Sensors	Sam	Learn Altium	Design PCB	Design PCB (Order by 1/30)	Start Code	Code	Test PCB	Integration	
Gantry	Sai	Calibration	MCU / SMC Migration	LUT/UART	Power		Test PCB	CDR Prep.	
RF/MCU	Sai	Wiring / I2C test	AT88RF	Design PCB	Communication test	Integration Test		PCB integration	
Misc.	Sai	Test Bottom Materials	3D Print Pieces	Wooden Frame	Chess Board Cut and Etched	Finalize Chess Pieces			

Gantt Chart After CDR

Task	Team Member	Mar 7 - Mar 13	Mar 14 - Mar 20	Mar 21 - Mar 27	Mar 28 - Apr 3	Apr 4 - Apr 10	Apr 11 - Apr 18	Apr 19
Bug Fix	Jack							
Training/Replay	Jack							
Bug Fix	Wei							
Training/Replay	Wei							
System Integration	Sam							
Bug Fix	Sam							
Iron Out Bugs and Inconsistency	Sai							
System Integration	Sai							

External Links

All Demo Videos Playlist

- https://www.youtube.com/playlist?list=PLN2cGbKly2ZLFxfQogWYSfkWECBJ7iLU9

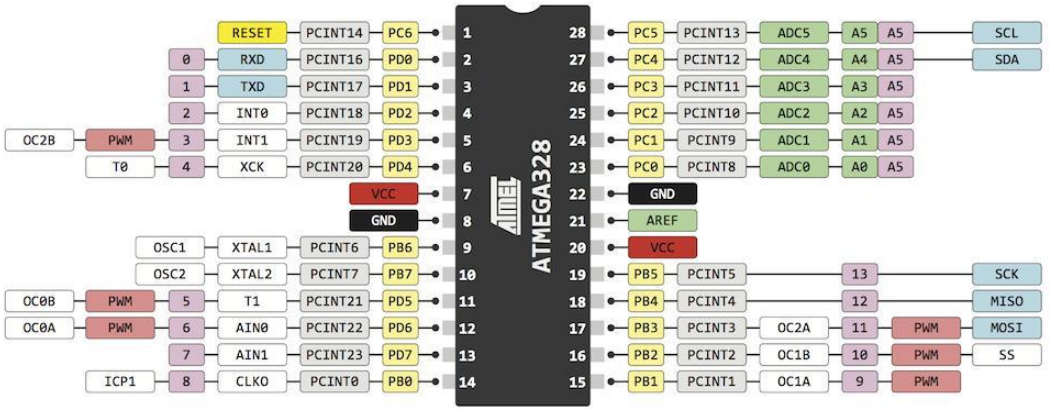
Github Repo

- https://github.com/degugj/lichessTesting

Thank You

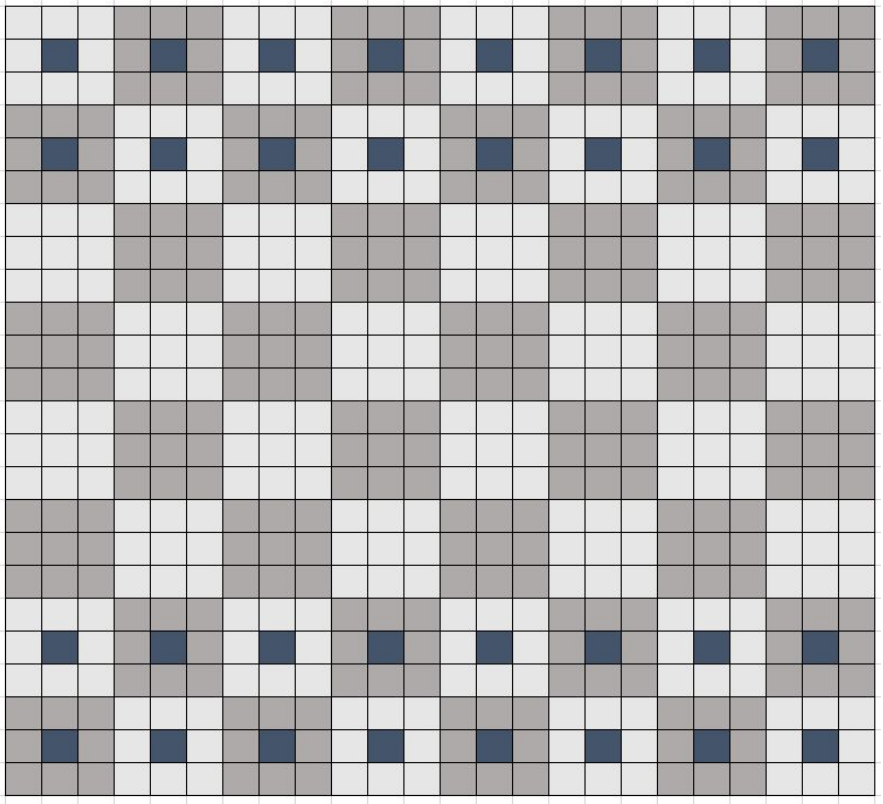
Justification of Atmega328P

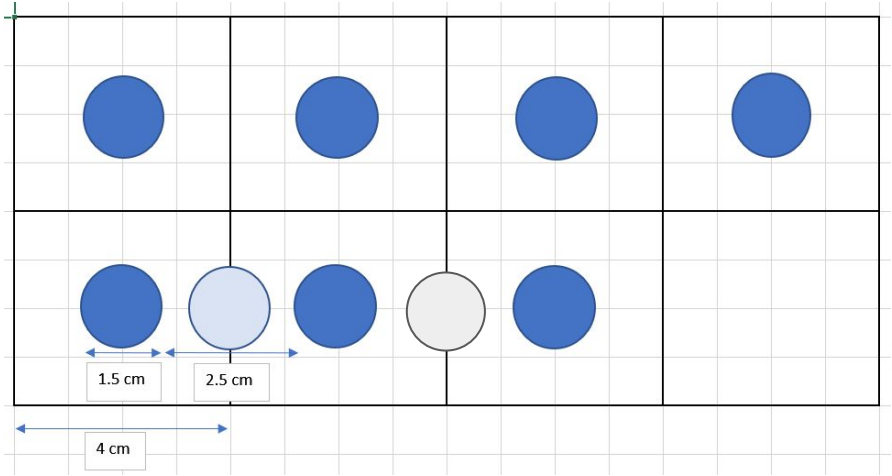
- Why?
  - I/O Considerations
    - UART Pins
    - Stepper Motor Drivers
  - Familiarity



Path Planning Metrics

- What is a heuristic?
  - An estimated cost from node *n* to a goal node
- What is Greedy BFS?
  - Greedy Best First Search is a search algorithm that expands nodes closest to the goal state, on the grounds that the closest nodes will return a solution quickly ($f(n) = h(n)$)
  - Time and space complexity?
    - Worst case: O(b^m) where m is the max depth of the tree
    - However, with our straight line distance and finite chess board, this will be reduced substantially




1.5 cm
2.5 cm
4 cm