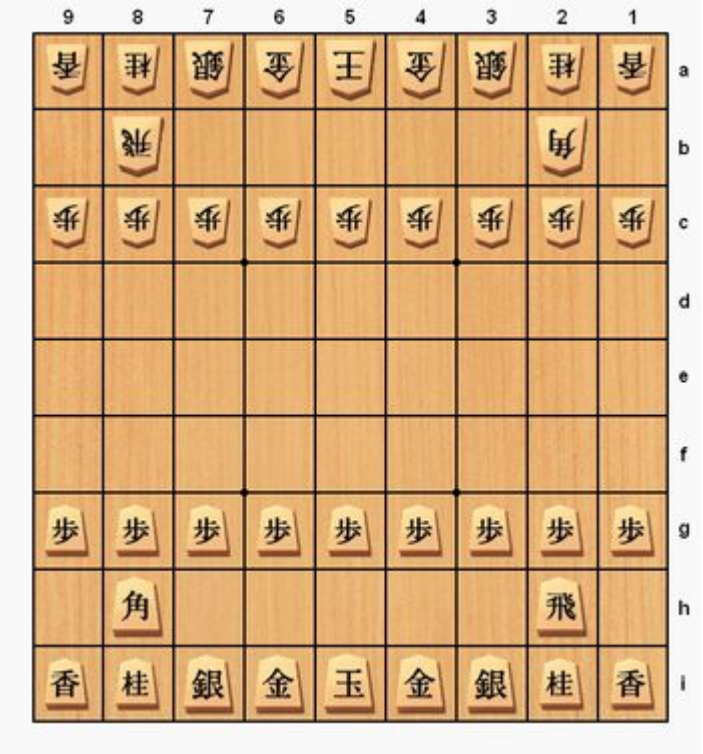
Group 48
Assistive Shogi Board

Rahul Rameshbabu
Maxim Papushin

What is Shogi?

- Commonly referred to as ‘Japanese Chess’
- Similar in game design to Western Chess...
  - Finite length
  - Two-Player
  - Zero-sum
  - Perfect Information




KBS京都
Shogi is no different from tic-tac-toe.

For most people, though...

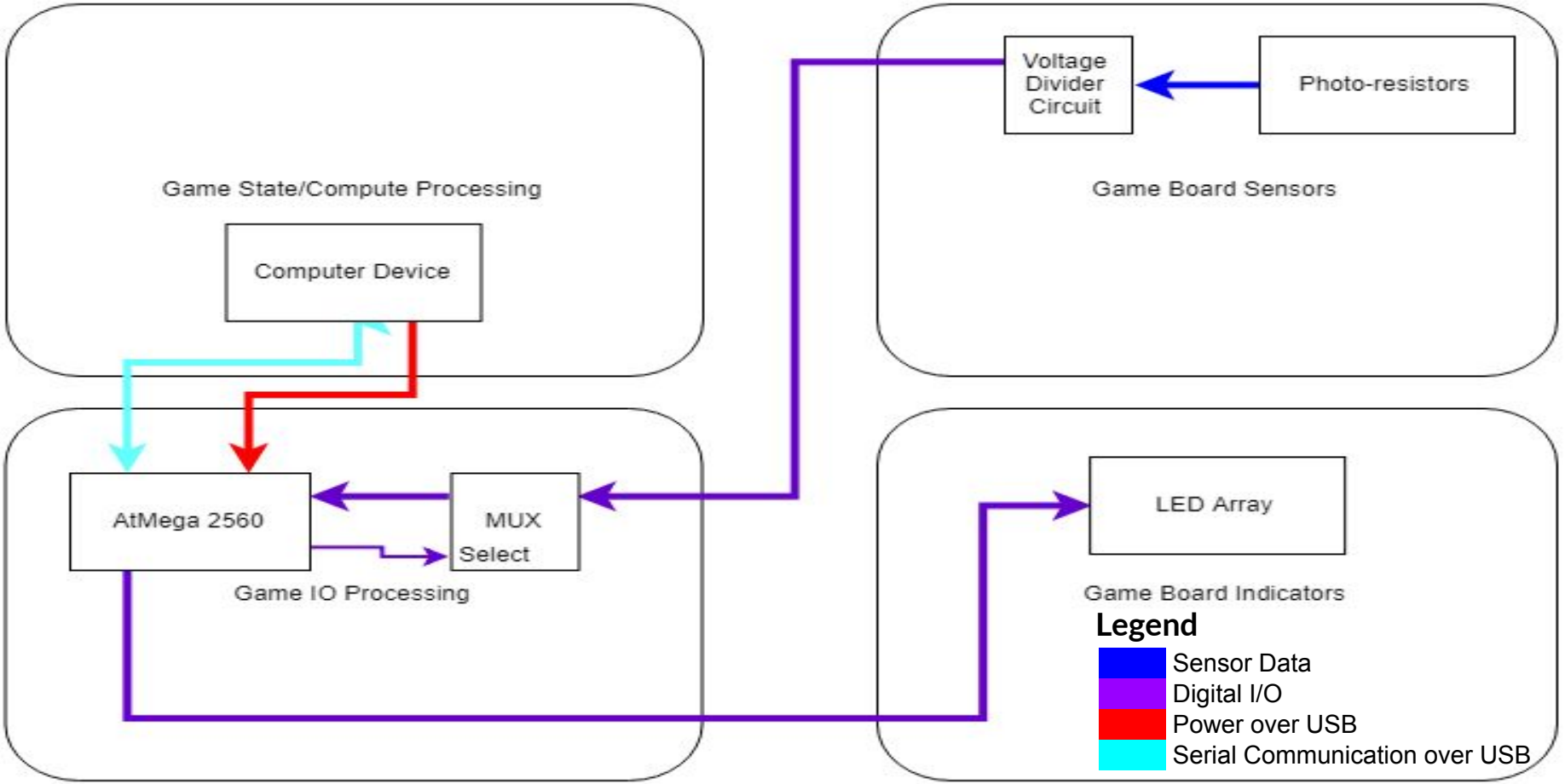
- Much more complex than Western chess
  - More piece types, most pieces can promote
  - Pieces can be captured and replaced
  - Several times more valid moves on average than Western chess
- Very difficult to learn!

(Fortunately not quite this complex)

Assistive Shogi Board

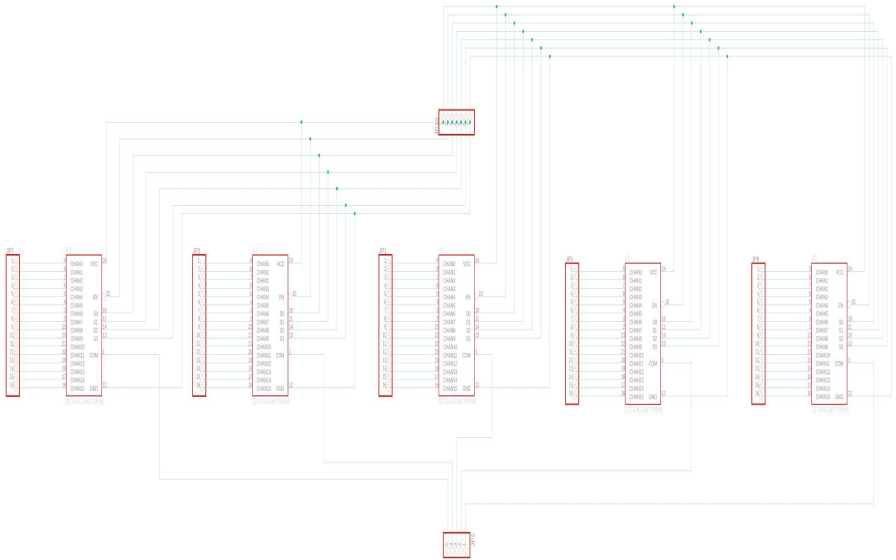
- Pieces are identified by kanji, not familiar to most western players
  - Piece movesets are indicated by the board when pieces are picked up
  - Program can print out localizes abbreviations if needed
- Pieces have individual promotion rules based on the type
  - Board indicates when a promotion is possible, optional, and when it is required
  - Reminds the players that a promotion can be done before continuing
- Some pieces have special drop rules
  - Two Pawns - Cannot drop a pawn into a column with another unpromoted pawn
    - Even professionals can forget, violated in the 2004 NHK Cup on national television
  - Drop Pawn Mate - Cannot drop a pawn to checkmate

Block Diagram



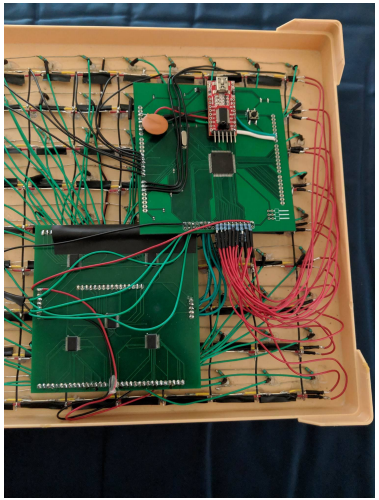
Photoresistor Array

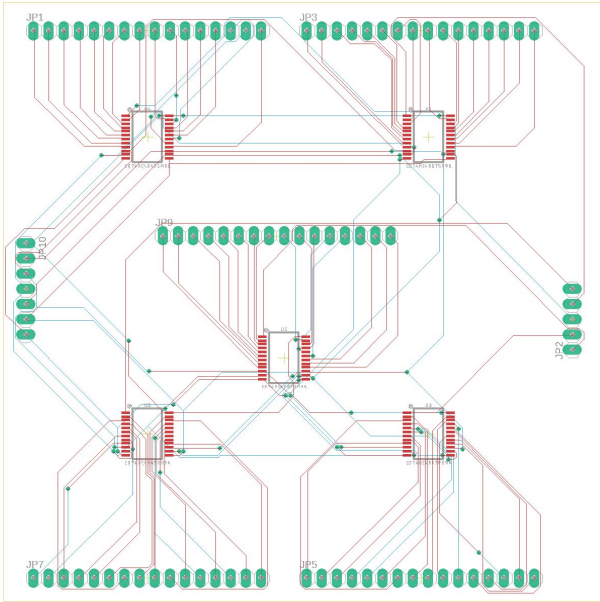
- Should be able to sense each individual piece lifted from the board and provide this information to the AtMega IC.
- Since we need 81 photoresistors for the 9x9 board, used 16:1 Analog MUXes to help make handling the input to the IC much simpler
- Built a MUX PCB breakout that had 5 of these MUXes in parallel, sharing the same select bits.



Photoresistor Array

- Used standard off-the-shelf CdS photocells for the implementation





Photoresistor Testing

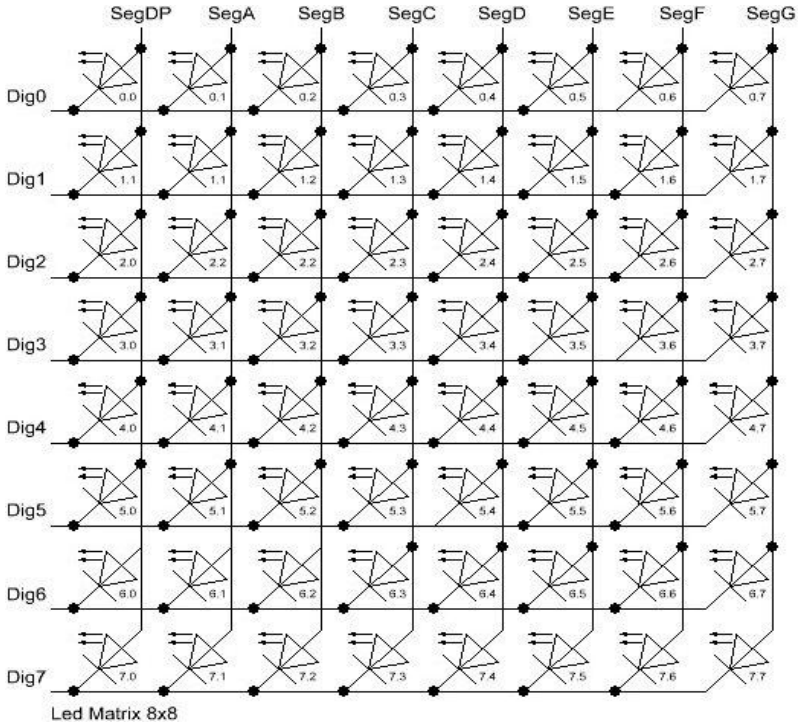
- Original Requirement:
  - Sensor output voltage records >3V when a piece is present, <2V when no piece is present.
- Conclusion: Need direct light for reliable results

Light Level	Resistance (kOhm)	Voltage (V)
Covered	197	4.30
Dark	19.6	1.493
Bright	13.5	1.10
Direct	2.68	0.322

LED Matrix

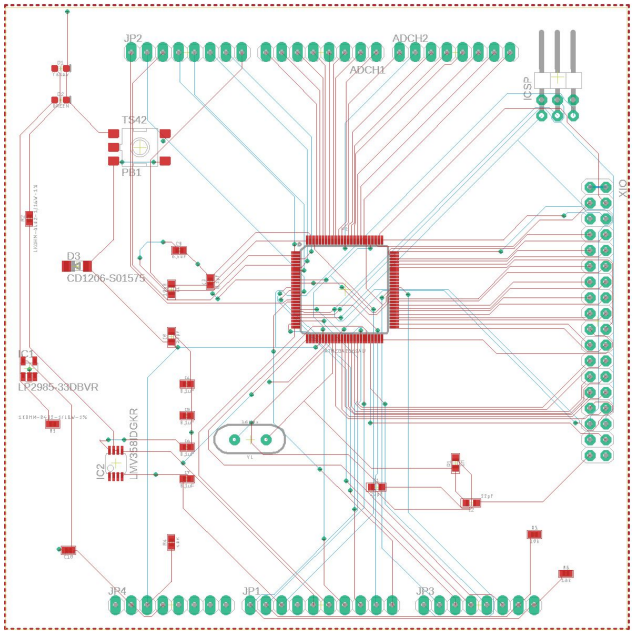
- Need to be able to control all 162 (81 Blue and 81 Green LEDs individually)
- Were originally planning to use the MAXIM7219 LED Matrix driver IC to handle this array but realized that the driver IC wastes space and can be replicated in functionality fairly easily but setting up the LEDs directly as an LED grid.
- Used single RGB LEDs (didn't use the Red) for the matrix

An illustration of a basic matrix lookup configuration for a grid of LEDs

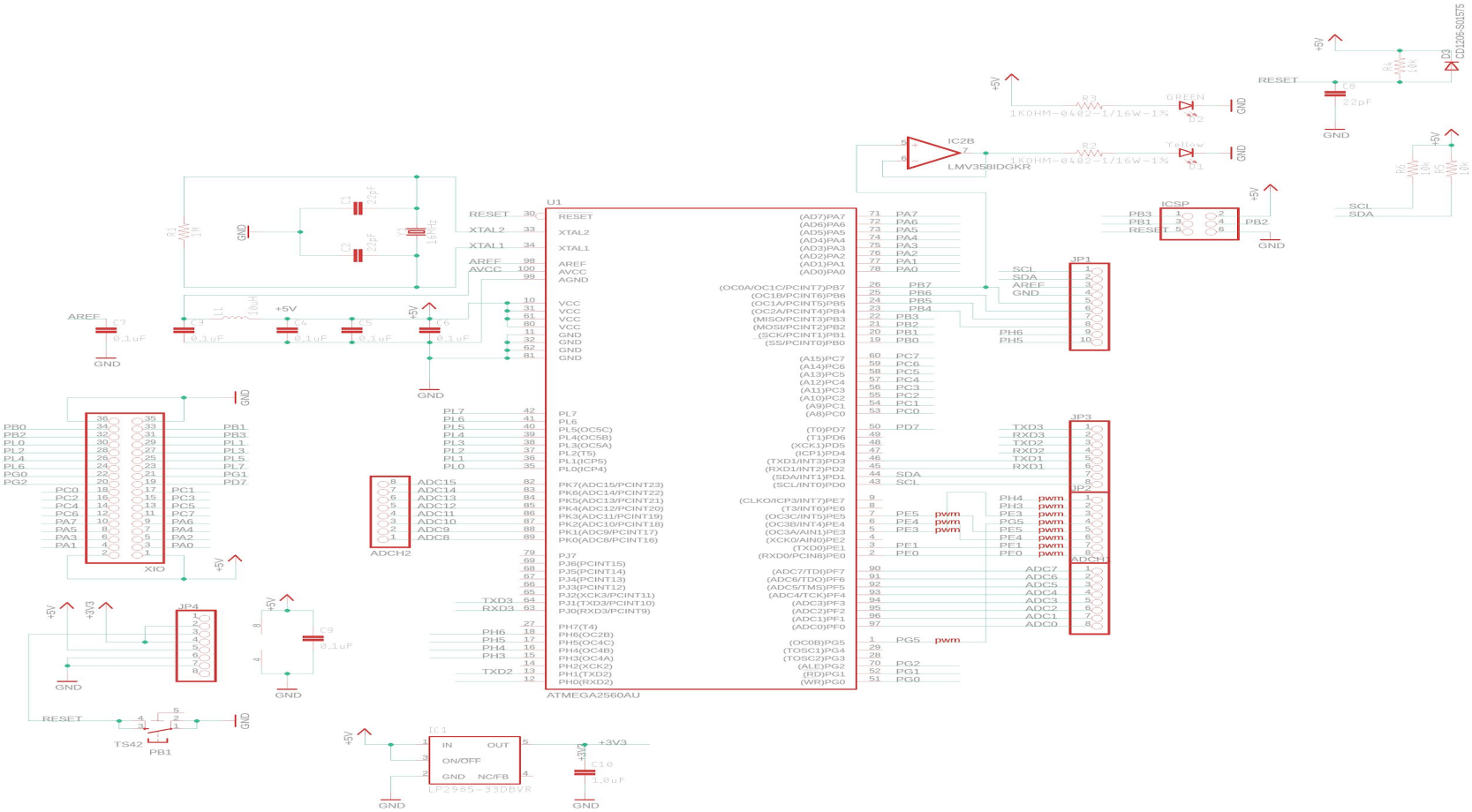


AtMega 2560

- Be able to program this PCB to read the photoresistor input and drive the LED output, while synchronizing with a computer to update the state of the game.
- Build a peripheral PCB to support the AtMega 2560 and allow it to have its bootloader be reprogrammed

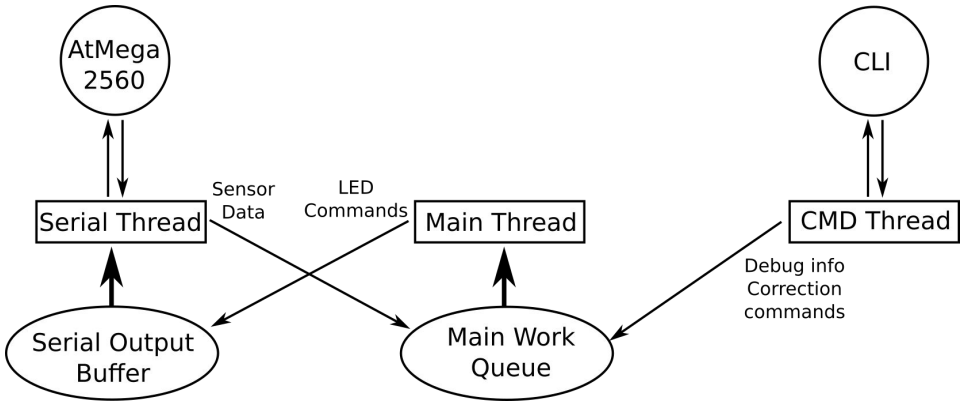


AtMega 2560 Schematic



Computer Game Program

- Computer program stores game state
  - Piece locations on board and in hand
  - Current state of FSM
- Interacts with AtMega to modify state corresponding to sensor data
  - AtMega sends sensor changes over serial
  - Program reads change and updates FSM
- Responds to AtMega with LED state
  - Program updates all LEDs on every FSM state change
  - Sends LED changes to AtMega over serial
- Detects invalid movements
  - Game state monitors piece movements and warns when invalid moves are made



Challenges Faced During Design

- Initial testing of the MUX and LED driver ICs got delayed due to the delay on the PCB orders.
    - Due to this, we realized that even though we ordered the correct MUX ICs the packages for these ICs were of the wrong size. We luckily figured this out early enough to get new MUXes however.
- Soldering the TQFP package for the AtMega 2560 IC was painful given the delicate pins
    - Learned how to use drag soldering to solder this IC.
- Physical construction proved laborious due to the amount of wiring done in such a tight space under the board.

Conclusion

- Managed to integrate all subsystems correctly and meet our requirements set.
- Took approximately 25 hours to design and solder the AtMega PCB. This was mainly due to the complexity of getting the AtMega 2560 to work in comparison to its 328P counterpart plus soldering a TQFP100 package is non-trivial
- Coding and optimizing took approximately 2 hours
- Soldering the LED matrix took approximately 12 hours over 2 sessions
- Soldering the photoresistors took approximately 8 hours over 2 sessions
- To verify, make, and prep the MUX PCB for fabrication through PCBWay took approximately 2-3 hours.
- Connecting all wires to MUX PCB took 10 additional hours in total