Assistive Shogi

Team 48 — Rahul Rameshbabu and Maxim Papushin
ECE 445 Project Proposal — Spring 2019
TA: David Hanley

Introduction

Objective

Shogi is a very difficult game for beginners to learn because of its complex set of rules. One can try playing Shogi at first, but it would be very simple to violate one of its rules or not be able to take advantage of more complex rules. There are a number of guides out there and one can even attempt to play online Shogi; however, the complexity of the game remains overwhelming due to the sheer number of types of pieces, each with their own unique moveset, compounded with added complexity from the ability to replace captured pieces.

To rectify this issue, we propose an augmented Shogi board that helps interactively teach a player all the valid moves that can be made using a piece that has been lifted from the board. This will help players know all possibilities of where to move the pieces and prevent any illegal moves from being made.

Background

While most Americans are at least vaguely familiar with the rules of Chess, if not able to fully play it, far fewer are familiar with Shogi. Whereas Chess has an established community of players and mentors willing to teach the game, Shogi has no such following, with few organized groups, if any. Thus, prospective players are left either playing unbalanced games online with far more experienced players, learning alone through a guide, or to start learning with other beginners, where neither player has enough experience to notice when a rule has been violated.

Chess is already a game where some of its more intricate strategies such as forks (simultaneously threatening two or more pieces with a knight) and pins (a similar strategy using a bishop, rook, or queen) are less utilized by beginner players, and even a decent number of the basic maneuvers are less known to many. Shogi has a 9x9 board over Chess's 8x8, but the more significant difference is Shogi has a heavier emphasis on counter attacking strategies [1]. Also, in Shogi, unlike Chess, pieces are never out of play, and may be replaced on the board by the capturer, creating an immense amount of possible moves compared to Chess. Finally, Shogi has some obscure rules which beginners may easily forget about, such as optional promotion, except under certain conditions where it becomes mandatory. Even though Shogi

and Chess pieces share similar names, they behave quite differently, which makes adapting from Chess to Shogi quite difficult [1].

Chess has quite an astonishing number of people playing the game, 35 million in the US [2]. This growing community helps support the rise in popularity of the game. However, Shogi does not have such a strong community, especially outside East Asia. While it is difficult to get an accurate count of the number of Shogi players in the United States, an estimate can be made by comparing the number of players well-known enough to have articles on Wikipedia, which would suggest roughly 100 times as many Chess players (315) as Shogi players (3) [3][4].

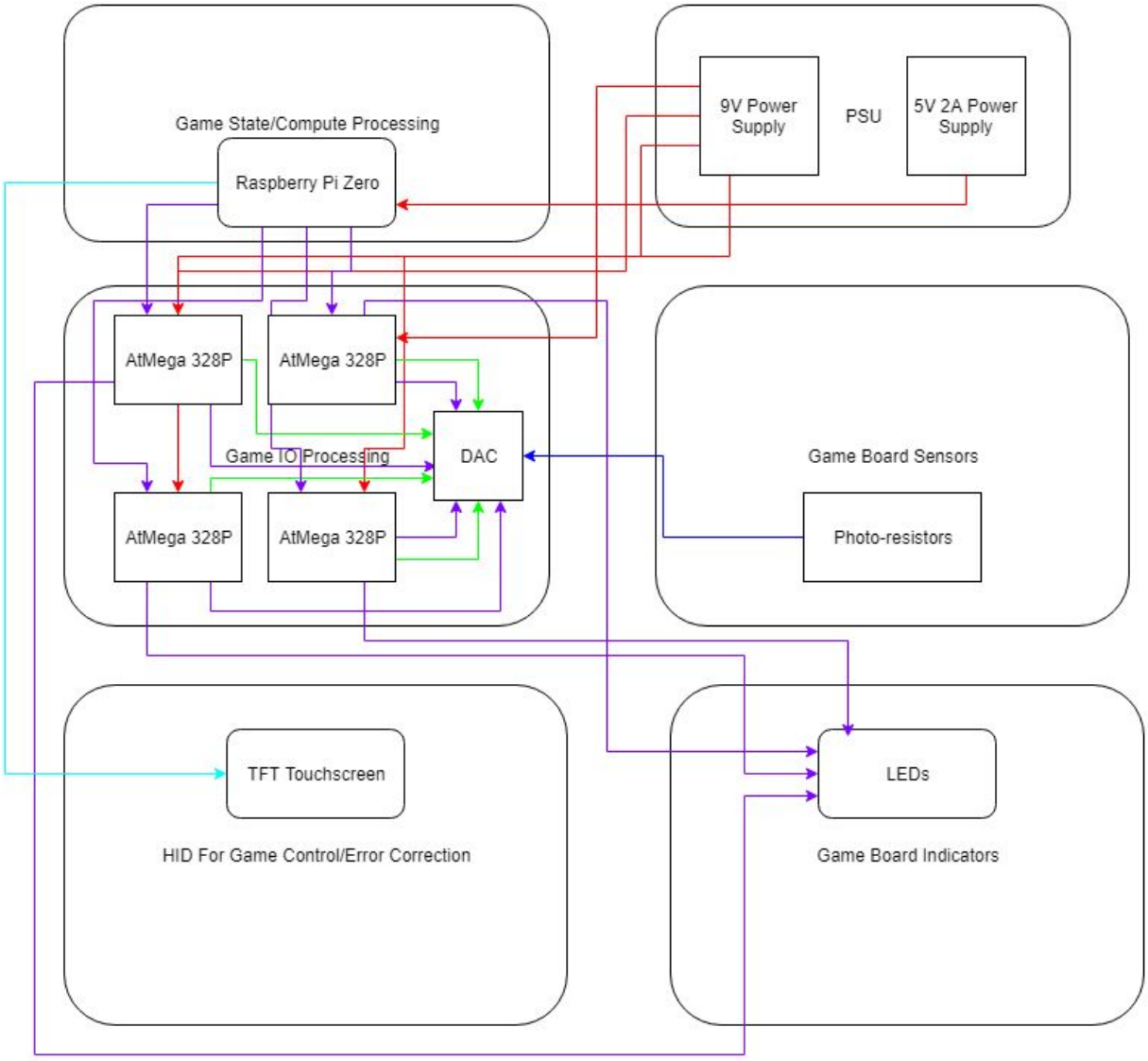
By making learning the game simpler, its popularity could rise significantly.

High-level requirements

- Board must recognize when a piece has been lifted and identify the possible moves for that piece.
- Board must update game state when the piece is placed in a valid location, and give a warning on the display if an invalid move is made.
- Touchscreen must allow for manual error correction, as well as indicating user error, such as misplays.

Design

Block Diagram



The design of the augmented Shogi board has six major components. We have a Game State/Compute Processing unit that handles maintaining the game state for each move made and making the master logical decisions behind which LEDs should be illuminated and processing the sensor I/O. This unit is responsible for back and forth communication with the AtMega 328Ps in the Game I/O block.

The Game I/O block is responsible for handle I/O between the board electronics of the LEDs and photoresistors, as well as for sending data over serial to the Raspberry Pi Zero in the GS/CP unit.

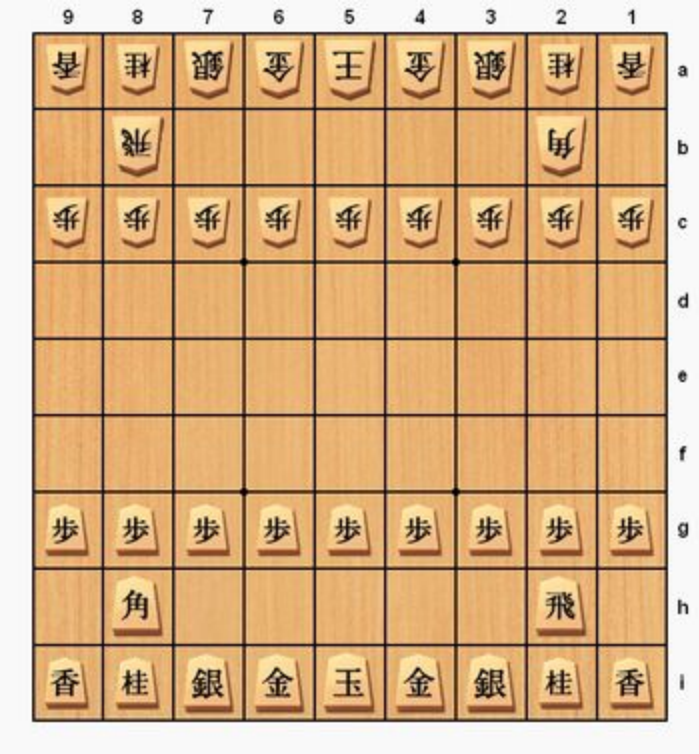
The HID for Game Control/Error Correction is used to allow the user to correct any piece detection errors, as well as to view the game state from the touchscreen display, and to provide additional information to the players, such as warning them when an invalid move has been made.

The Game Board Sensor block provides information about what pieces have been moved with the use of photoresistors. They send data to be processed by the DAC in the Game IO Processing unit.

The Game Board Indicator states what moves are valid to make with the piece selected. They are controlled by signals from the AtMega chips in the Game IO Processing unit.

The Power Supply Block is responsible for handling power distribution to the electronics.

Physical Design Description



Shogi Board Grid Layout [5]

The board will be modified to include photoresistors underneath each of the 81 squares on the board. For each square, a hole will be drilled through the center of the piece, and a photoresistors will be attached into the hole, facing upwards. When a piece is present on that square, the photoresistor is covered; without a piece, the photoresistor is lit by the ambient light of the room. This difference in lighting will be used to detect when a piece is moved. All the electronics will be connected underneath the board, except for the touchscreen, which will be placed off to one side of the board, such that it is accessible to both players.

Functional Overview

Game State/Compute Processing - The goal of the Game State/Compute Processing block is to receive process input from the Game I/O Processing block AtMega 328P microcontrollers and also process any board state correctional data provided through the touchscreen interface from the HID For Game Control/Error Correction block. Once it receives game state updates, the Raspberry Pi Zero then processes the updates to be reflected both on the touchscreen and sent to the AtMega 328Ps to update the respective LEDs for the valid moves that can be made.

Game I/O Processing - The Game I/O Processing blocks goal is to handle the large number of sensory inputs and digital indicator outputs needs for the assistive Shogi board. This block sub-divides the work of piece detection and LED illumination but does not know about the big-picture setup of the overall game board. Therefore, the AtMega 328Ps used in this block must use Digital I/O serial communication with the Raspberry Pi Zero in the Game State/Compute Processing block to update the Pi with sensor information that lets the Pi update the overall game state. Any detection of a lifted piece should be sent over the the Pi Zero, which in turn sends a signal back to the appropriate AtMega chips to tell them, which LEDs to illuminate, thus indicating to the players which spaces are valid moves given the piece which was lifted from the board.

HID For Game Control/Error Correction - This block consists of a touchscreen that interfaces back to the Raspberry Pi Zero and used to as mechanism to both provide information about the current board state perceived by the Raspberry Pi Zero and let the user update it due to any errors, such as pieces being accidentally bumped or moved.

PSU - The PSU block is used to distribute power to the various electronics involved in the system. However, power will be provided to most of the individual components through the AtMega 328P microcontrollers and the Raspberry Pi Zero, which are able to provide power as needed to the rest of the electronic systems. For instance, the Raspberry Pi Zero will use the power it is given from this block and provide power to the touchscreen through its GPIO.

Game Board Indicators - This block consists of LEDs used to indicate when each position is valid to move to on the 9x9 Shogi board. These indicators are physically updated by the AtMega 328Ps, which are in turn signalled to enable or disable certain LEDs to light up by the Raspberry Pi Zero.

Game Board Sensors - This block contains a series of photoresistors placed at each position on the 9x9 Shogi board. These sensors send their input into an analog-digital converter within the Game IO Processing unit subsystem, and the converted sensor data is passed into one of the AtMega 328P microcontrollers.

Block Requirements

Block Name	Technical Requirement	Functional Requirement	Quantitative Requirement
Game State/Compute Processing	Must be able to interface a touchscreen and somehow communicate with multiple microcontroller devices.	Should be able to update the game state of the board from the data provided by multiple AtMegas and from any user error corrections provided.	Needs to utilize a single board microprocessor such as a Raspberry Pi Zero to interface with four AtMegas over serial and use GPIO to interface to a touchscreen.
Game I/O Processing	Must be able to provide sensor input to a microprocessor as well as drive LEDs based on signals provided from a microprocessor over serial.	Should be able to take signals over serial for LED states provided from the microprocessor and turn indicated LEDs respectively on or off to indicate which positions are valid to move to. Also, should be able	Due to massive amount (81 analog input and 81 digital outputs) of I/O involved, we plan to utilize 4 AtMega 328Ps with analog I/O multiplexed with 1 DAC for each AtMega.
HID For Game Control/Error Correction	Must be an interface compatible with the microprocessor that allows for both visualizing how the game is interpreted by the microprocessor and allowing for that interpretation to be corrected by the user through a some form of touch-input capable display	Should allow the user to be able to both view the state of the board as recognized by the microprocessor and allow the user to update that state if any errors have occurred in the game state	1 touchscreen interface for the players to be able to provide correction or to view the game state
PSU	Must be able to power the 4 AtMega 328Ps and the Raspberry Pi Zero and indirectly handle these devices	Simply must be capable of providing the needed power for the board to fully function.	One 9V, 7A (or greater AMP) power supply and one 5V, 2A power supply for the Raspberry Pi

	providing power to the other peripherals such as the touchscreen and LEDs.		Zero
Game Board Indicators	LEDs must be controllable from the AtMega 328P in order to indicate valid moves.	Must visually indicate to the user valid moves by the piece currently selected on the Shogi board	81 LEDs will be needed to indicate valid moves on the Shogi board, one for each space.
Game Board Sensors	Photoresistors whose varying resistance can be read by the AtMega 328Ps in order to determine whether a specific Shogi piece has been lifted or not.	Some sensory input that indicates when a piece on the Shogi board has been lifted to determine which LEDs should be lit for indicating the valid moves with that specific piece.	81 Photoresistors will be needed to sense every piece on the board, one for each space.

Risk Analysis

The most difficult part of the project construction will likely be the circuitry needed to read from the photoresistors. Since there are 81 inputs, the amount of wiring needed to connect each photoresistor to an input on the PCB will be extensive, and will likely cause errors. Furthermore, the amount of inputs needed are more than any single AtMega chip is able to support. As such, it will be necessary to add input multiplexers in order to reduce the number of inputs to a manageable amount, which will in turn add additional complexity to the wiring, as well as require larger amounts of work on each AtMega to poll all inputs consecutively.

Similarly, each square on the board will have to show one or more LEDs, which will also have to be controlled individually by the AtMega chips. In order to avoid having to use so many output lines, it may be necessary to create simple LED controller circuits with latches to save a state, so that they can be programmed with fewer data output lines, instead of creating a 81-wide one-hot bus to control LEDs, which would further complicate wiring.

Ideally, a circuit could be designed in such a way that only one or two AtMega chips are necessary. This will not only reduce the cost of the entire board, but should also simplify the code needed for the controlling Raspberry Pi to interact with each of the AtMega chips in turn. According to the AtMega 328P datasheet, each chip has 23 programmable I/O pins [6], which would have to be shared between the input photoresistors and the output LEDs.

The physical construction of the board is not expected to be difficult, as a relatively large margin of error is acceptable for the photoresistors to fit in the holes while still being entirely covered by the pieces. This is not expected to require any more precision than what could be achieved with a simple drill press. However, depending on what sort of board is used for the design, some effort may be needed to fit electronics into a relatively small space, or otherwise to lift the board up with some supports in order to create more space underneath the board.

Ethics And Safety

Due to our project's lack of data collection from the users, the ethics concerns involved seem to be negligible. Our project also operates under a low voltage environment, minimizing many electric safety concerns, though not removing them completely.

One possible issue is the issue of undervoltage in the case of large loads [7]. We need to be careful when developing the PSU to be sure we accurately accounted for the power consumption needs of the entire design. If not, we might damage our hardware over time due to operating electronics at inconsistent voltages. If we did so, we would be delivering a product that is not capable of sustaining itself, which would be unacceptable as a consumer product.

We should also be very of the case of using incorrect or unregulated power supplies that may result in an overvoltage and thus burn or damage the ICs and other electronic components used in this project. However, aside from power supply concerns, this project operates using mostly low voltage systems and does not take in an special input or sensitive data, reducing the amount of ethical and safety concern significantly. Even though our project itself does not seem to violate any ethical or safety concerns, we as developers should be sure to follow a code such as the IEEE Code of Ethics [8].

References

^[1]https://www.gnu.org/software/gnushogi/manual/Differences-between-shogi-and-chess.html
^[2]https://en.chessbase.com/post/che-redux-how-many-people-play-che-
^[3]https://en.wikipedia.org/wiki/Category:American_chess_players
^[4]https://en.wikipedia.org/wiki/Category:American_shogi_players
^[5]https://upload.wikimedia.org/wikipedia/commons/thumb/9/9f/Shogiban.png/350px-Shogiban.png
^[6]https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf
^[7]https://www.captech.com.au/2016/05/06/common-issues-with-power-supply/^
^[8]https://www.ieee.org/about/corporate/governance/p7-8.html