Smart Chessboard with Dual Hall Sensors

Abstract:

The goal of the project is to create a physical chess board with much of the functionality seen in digital boards online. Electronic boards have existed for a while, with super simple player vs. AI capabilities being the main inspiration for my project. The project uses dual-output hall chips to sense the location of pieces, and recognizes the polarity of the magnet placed on each square. This information was shown to be the minimum required to handle an entire game of classic chess.

Sam Rosenberg
Dr. Dann
Applied Science Research

March 25th, 2019

Introduction

The primary goal was to create a “smart chessboard.” Online chess has added a huge amount of functionality and versatility to the game of chess, including automatic rankings, timing, game analysis and person vs. AI play, however physical chess boards remain optimal in one specific way: feel. Online chess, while increasing functionality, loses a lot of what makes chess such a fun game to play simply in that it obviously loses the physical feel of the pieces. This project was to create a chess board that has these advantages on online play while still having the feel of a physical set.

The origins of chess are hard to track down, with several different locations being the potential birthplace for the famous game. The games appears to have been created in India, then modernized in Persia before spreading across the globe.[1] This version of the game however was a closer reflection of checkers than chess: pieces moved differently, but were exceedingly limited in the options they had. It wasn’t until the 15th century in Italy that chess finally became the fully complete game we see today.

Chess has become known for being the pinnacle of 1 vs. 1, perfect information strategy games. A chess board never changes: each game will behave under the exact same ruleset as the last. But despite that, each game morphs into its own form, into a different style. Professionals have popularized sequences of moves, and the literature of the strategy of the game has rapidly expanded. It’s this depth, coupled with the simplicity of the ruleset, that has kept chess king for so long. Other games might have more strategy, but no other game is so reliant on the mind and through process on the players playing as chess.

People have been interested in the idea of a computerized chess board for a long time. In 1770, an inventor claimed to have created the first mechanical chess board, one called the “Mechanical Turk” when mechanized man moved the pieces of the board and could beat another player. In the end, it turned out that there was simply a person hiding inside the body of the machine controlling the mechanized person. It wasn’t until 1953 that Turing was able to create a (bad) chess computer that was able to successfully lose a match to another human. Several more innovations were made in the field of chess AI, including programs that could win in the last few moves of a game, and ones that took a very long time to think through each move they made.

In 1957 Alex Bernstein was able to make the first full chess AI, one that could play a complete game. He was working with IBM at the time, the company that would soon lead in the field of chess AI. In 1977, the first computerized chess set created by Fidelity Electronics was finally created. To determine moves, players were required to manually enter each move they made into a keypad built into the side of the board they played on. Basic computing would take into account the current board position then determine a move for the computer based on the

[1] https://www.chesshere.com/resources/chess_history.php, Website Accessed March 2nd, 2019

level at which it was set. A small readout told which moves the computer made, which the player would then adjust the pieces on the board. Later, fully automated robotics would even take care of removing the pieces from the board.[2]

Electronic chess boards have existed for a while, but with current high quality wooden sets going for more than 600 dollars, they are hardly accessible to the average consumer. Most of these boards include pressure sensors on each square and likely function as a result of differing weight of the pieces. While no further details were available on most boards, some did have simple tags confirming at least the use of pressure as a primary way to detect the location of pieces.[3] In Figure 1, you can see one such board using the pressure sensors addressed above.

Because the pieces appear to be custom made for the board itself, exact weights were available to the manufacturer and the pressure sensors were likely able to track where each of the individual pieces are moved to. Other differences from this board include the built in display and the arrow keys built into the machine, as well as the lack of a chess clock. This board is able to use highly specific sensors in a machine created shell, two things that take massive amounts of R&D to be able to create and test repeatedly.

[2] https://www.pcworld.com/article/2036854/a-brief-history-of-computer-chess.html, Accessed March 23rd, 2019

[3] https://www.chesshouse.com/collections/electronic-chess/products/millennium-chess-computer-chess-genius-pro Website Accessed February 13th, 2019

The goal of the board is to provide as much functionality as possible from a more limited number of hardware components relying on software for a major portion of the heavy lifting necessary. While the PCB boards were expensive in the creation of the project, it was mainly concentrated in the startup cost. While the first 64 boards were 148 dollars all together, the next 140 boards were only 2 dollars total. The cost would be dramatically reduced in the creation of large quantities of boards. The major intent of the project is to show that these boards (and the chip placed on them, are close to the only kind of sensing and equipment necessary to provide a fully functioning electronic chess set.

Design

To keep track of pieces, the board takes a snapshot of the location of pieces before and after a move. In Figure 2, you can see a subsection of a chess board with a pawn of each color and a black knight. At a given moment, this could the the board state of the game of chess. Then, in Figure 3 you can see the knight has moved to a new square. The board takes another snapshot of the pieces, and can tell that a piece has appeared and another one has disappeared. Thus, it can come to the conclusion that the knight has moved from its initial position. Further, imagine the second image had only Xs, instead of the pieces themselves. It is still possible, due to knowing the prior board, which pieces are where. This works practically recursively all the way back until the initial board setup. So what this means is that the sensors on a given piece do not need to be distinguishable, as between each move, only 1 piece is different (except in very specific circumstances).

Even in Figure 4 and 5, where a white pawn captures a black pawn, if you think back to the Xs there will be one missing (the white pawn), and one legal space for it to have disappeared (taking the black pawn). Unfortunately, this does introduce a problem: what if the pawn had two legal capture points? Figure 6 and 7 display the scenario, in which the white pawn can take either black pawns. Once the pawn is taken, the method with the Xs no longer works. There is a piece in all three locations, before one of them disappears with two legal targets. Thus, it is necessary that black an white pieces have different methods of sensing, as if you imagine white pieces as Xs and black as Os, the above problem is never encountered. While the method will require more complicated coding for things like castling (a swap of the king and rook) and en passant (taking a double-pushed pawn by moving a pawn behind), it likely will simply come down to a bit of extra work and nothing requiring a change in the design of the product.

The sensors for the project are unipolar hall chips with two outputs, one for north and one for south, called EM-1791.[4] This allows for magnets to be placed at the bottom of each chess piece and simply reorient the magnet for different colored pieces. Each square of the board will

---

4
https://www.akm.com/akm/en/file/datasheet/EM-1791.pdf, Website Accessed February 13th, 2019

have one such sensor, with four vias, two for power and ground, two for outputs. Figure 8 is an image of the PCB board that will be used in the design. A multiplexor will be used to held consolidate the section of squares into a manageable number of pins for an arduino handling the information being transmitted. Once the data is collected, the data will be sent into a computer using the “universal chess interface” to request moves in one-player games, or analyze the board state during a two player game.[5] A button is used to send a signal to the arduino that a move has been made and a “snapshot” should be taken of the current state. This will likely be placed on a closed loop with one of the arduino pins, and will send a signal to the board on click. It will also cause the clocks to switch (so the stopped clock starts and running clock stops). Figure 9 details the flow of code in relation to what inputs are received and how the computer chooses to react. The board is processed in one of two ways depending on which mode it’s set to.

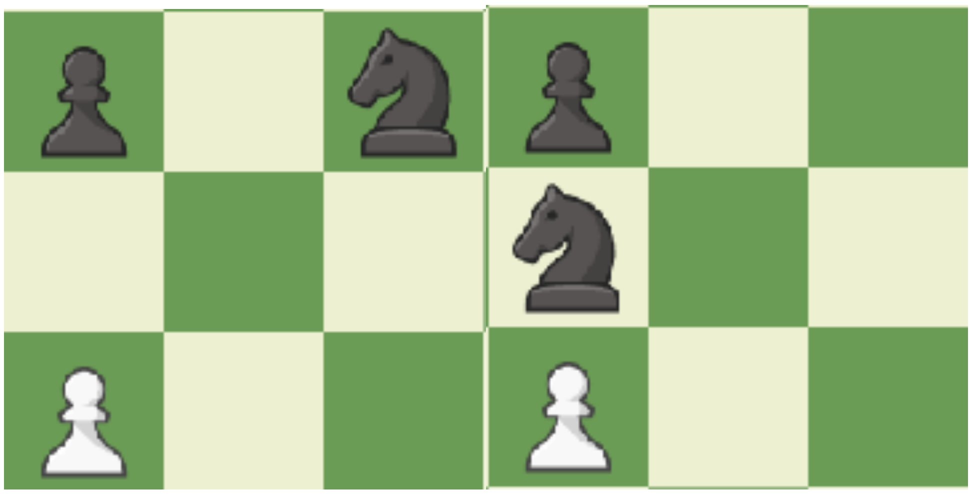
The board is a hollow box 2 inches tall with tournament sized 2.25 x 2.25 inch squares. Inside the box there is a plate with the hall chip PCBs placed underneath so that each square triggers a given hall chip perfectly. The plate inside the board is not yet set in height, as there is a worry that a magnet placed on a square could trigger multiple different squares if the sensitivity of the hall chips too high and they are placed too close. The PCBs have through holes, allowing output and power to be provided from the top or bottom, which will allow for the most flexibility and least interference with the sensors.

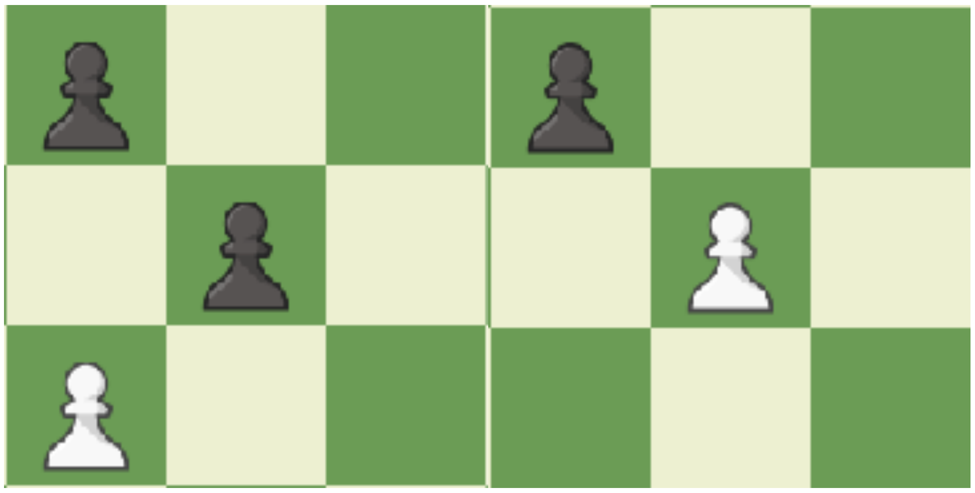
The arduino is placed on the PCB tray, and cables run directly to the arduino. The trouble is keeping the internal components properly managed so that nothing is either getting in the way of other things or is knocked out of position messing with the sensing ability of the set. 1/4th inch wood is being used as is not only sturdy, but it allows clean magnetic fields to pass onto the hall chips without the entire array being triggered by a single magnet, and still is strong enough to trigger the sensors. Figure 11 is a CAD drawing of the board and the scaled image of a single chip. Some images don’t have the top and bottom of the board, exposing the PCB and arduino. The PCB board (labeled in the image) is repeated 64 times. That PCB can be seen with a hall chip soldered onto it in Figure 12. The chips are surface mount-solder with a reflow oven in order to secure the hall chips to the board. Each board is used for one square and has two inputs and two outputs. The entire board is around 1 cm^2, with the chips being roughly .2mm by .3mm. The through-holes need 30-gauge wire in order to fit inside the small slot. Figure 12 shows the way the hall chips connect to the multiplexer at X0-X7. The three pins go the the arduino and are the same for every multiplexor, where one output pin also goes to a unique spot on the board. There are 16 multiplexers on the board, and each one has 4 hall chips connected.

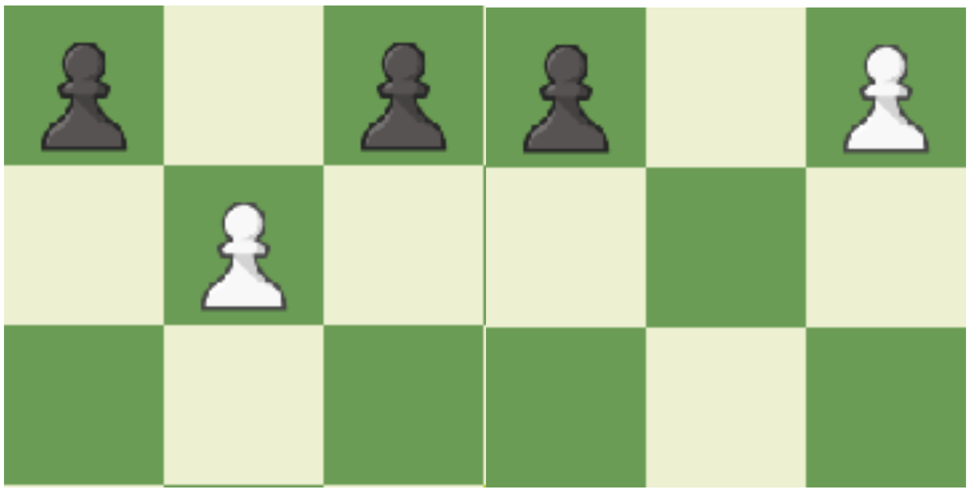
On the board there are seven power rails, made of solid copper wiring. There are two power lines, 2 ground lines, then pins 0-2 for each of the multiplexers are all on the same rails for signals being sent from the arduino. Those 7 rails (4 of which are connected in pairs of two,

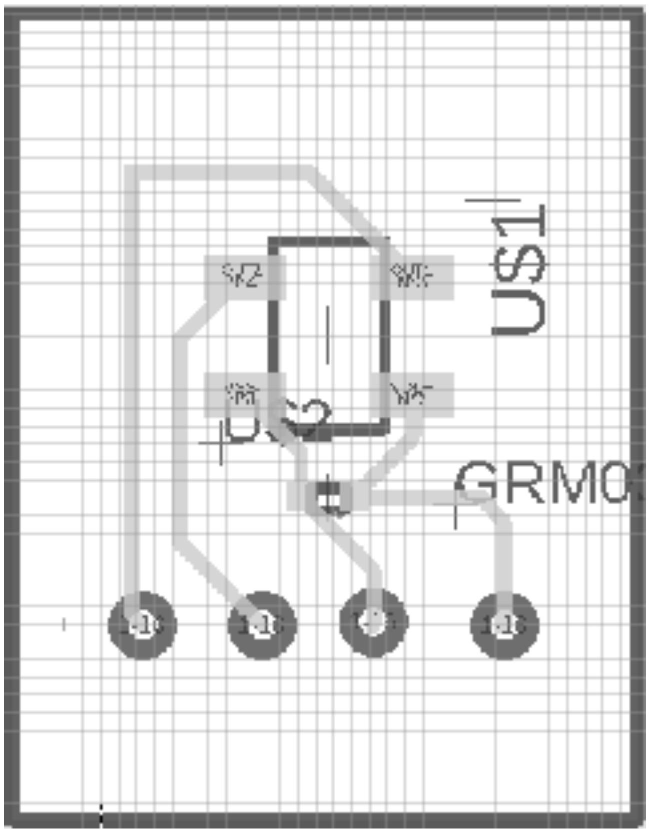
[5] https://www.shredderchess.com/chess-info/features/uci-universal-chess-interface.html, Website Accessed March 2nd, 2019

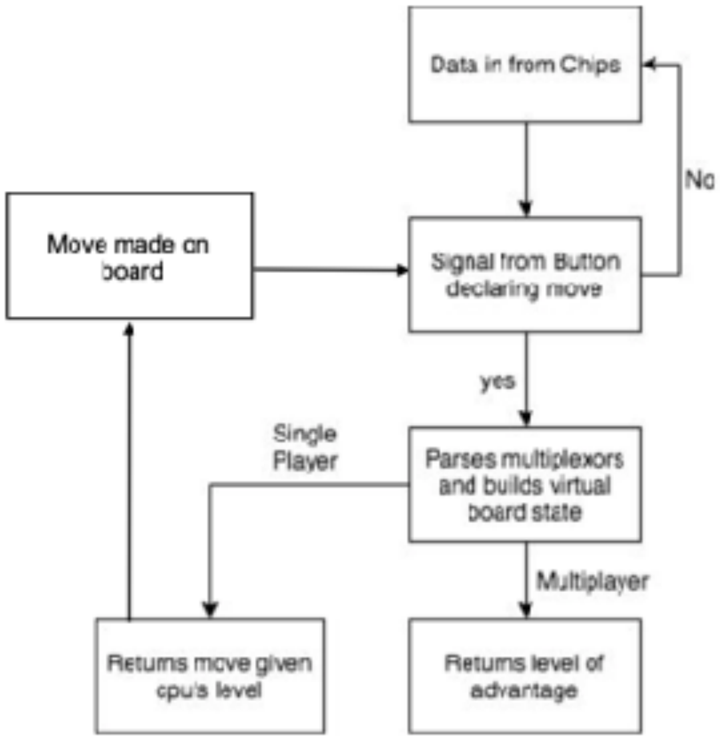
both power and ground) handle all transfer of power and signals both to the multiplexers and the hall chips from the 5 outputs of the arduino.

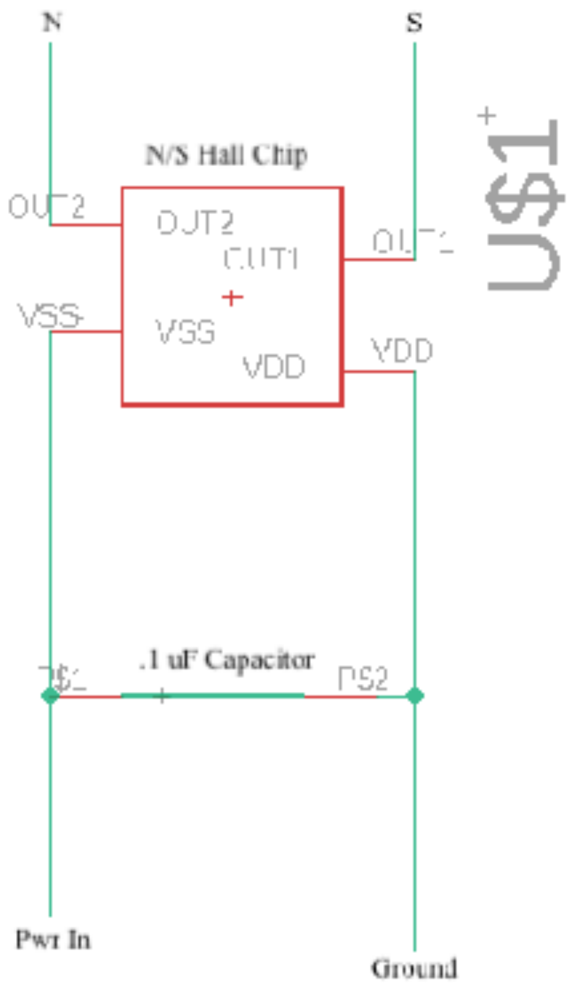


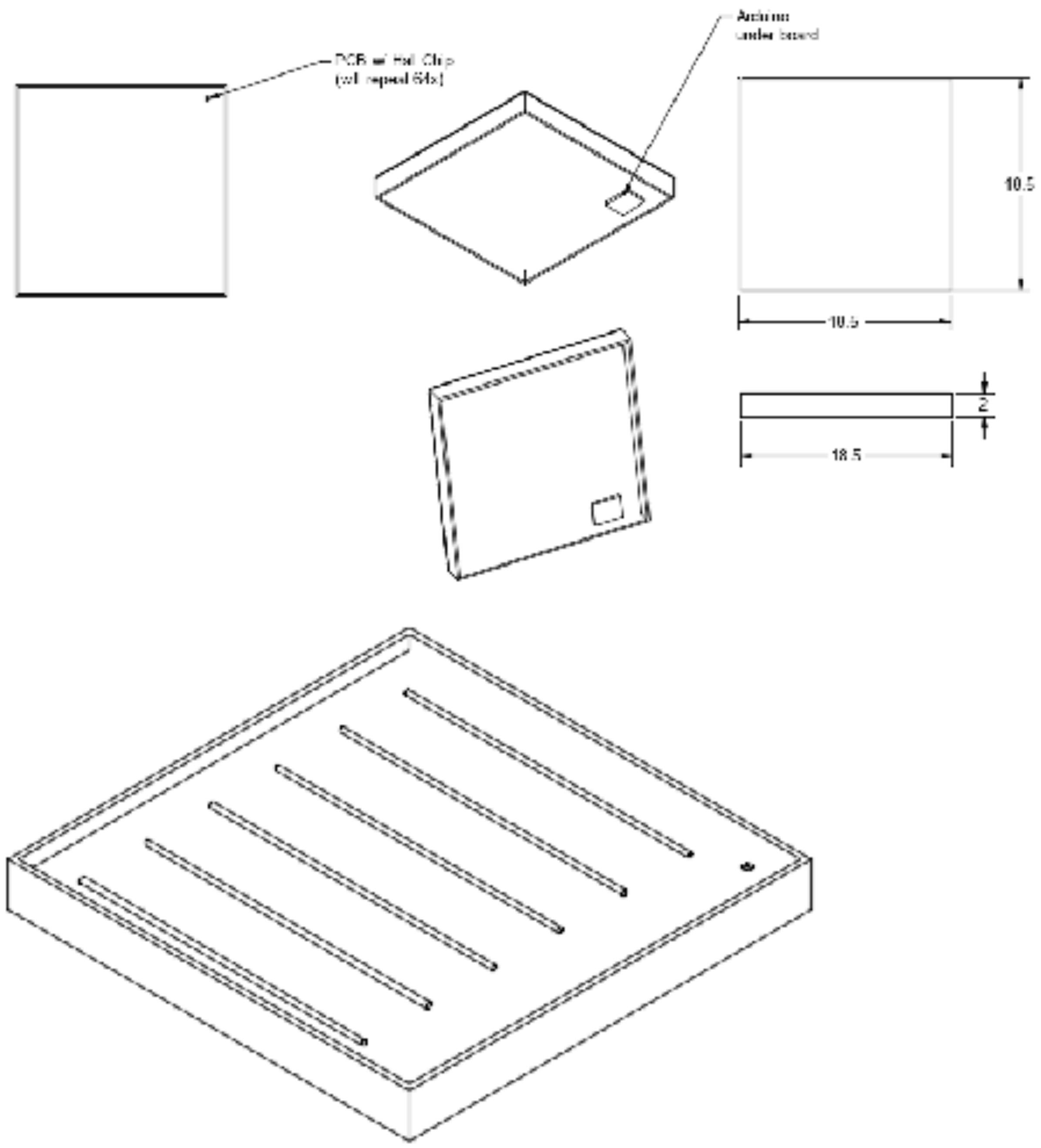


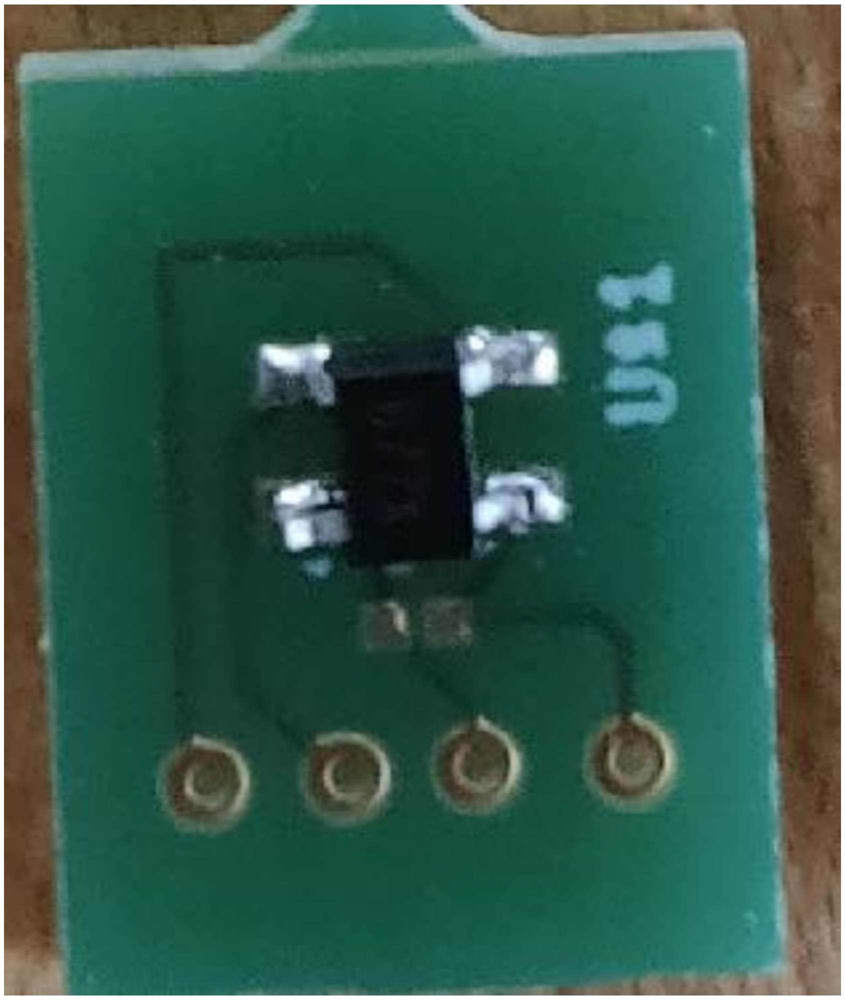


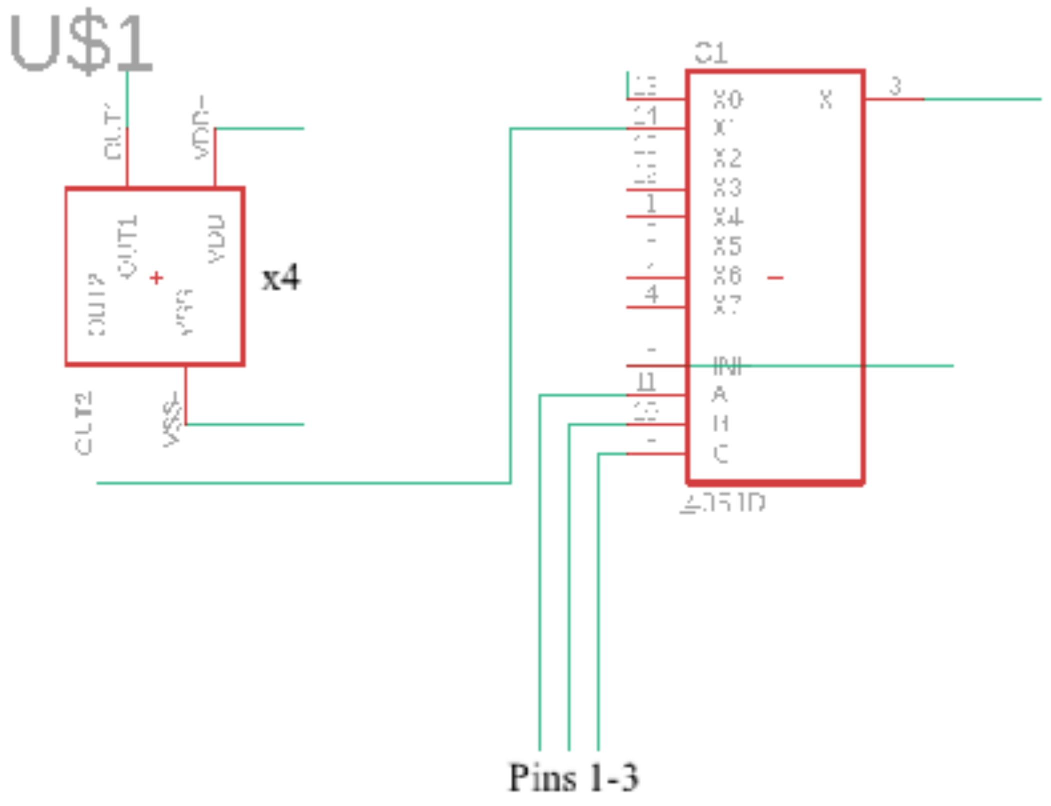

US1
US2
GRM0


Data in from Chips
No
Move made on board
Signal from Button declaring move
yes
Single Player
Parses multiplexors and builds virtual board state
Multiplayer
Returns move given cpu's level
Returns level of advantage


N
S
N/S Hall Chip
U$1
OUT2
OUT2
OUT1
OUT1
VSS
VSS
VDD
VDD
.1 uF Capacitor
PS2
PS2
Pwr In
Ground


PCB w/ Hall Chip
(will repeat 64x)
Arduino
under board
10.5
10.5
2
10.5

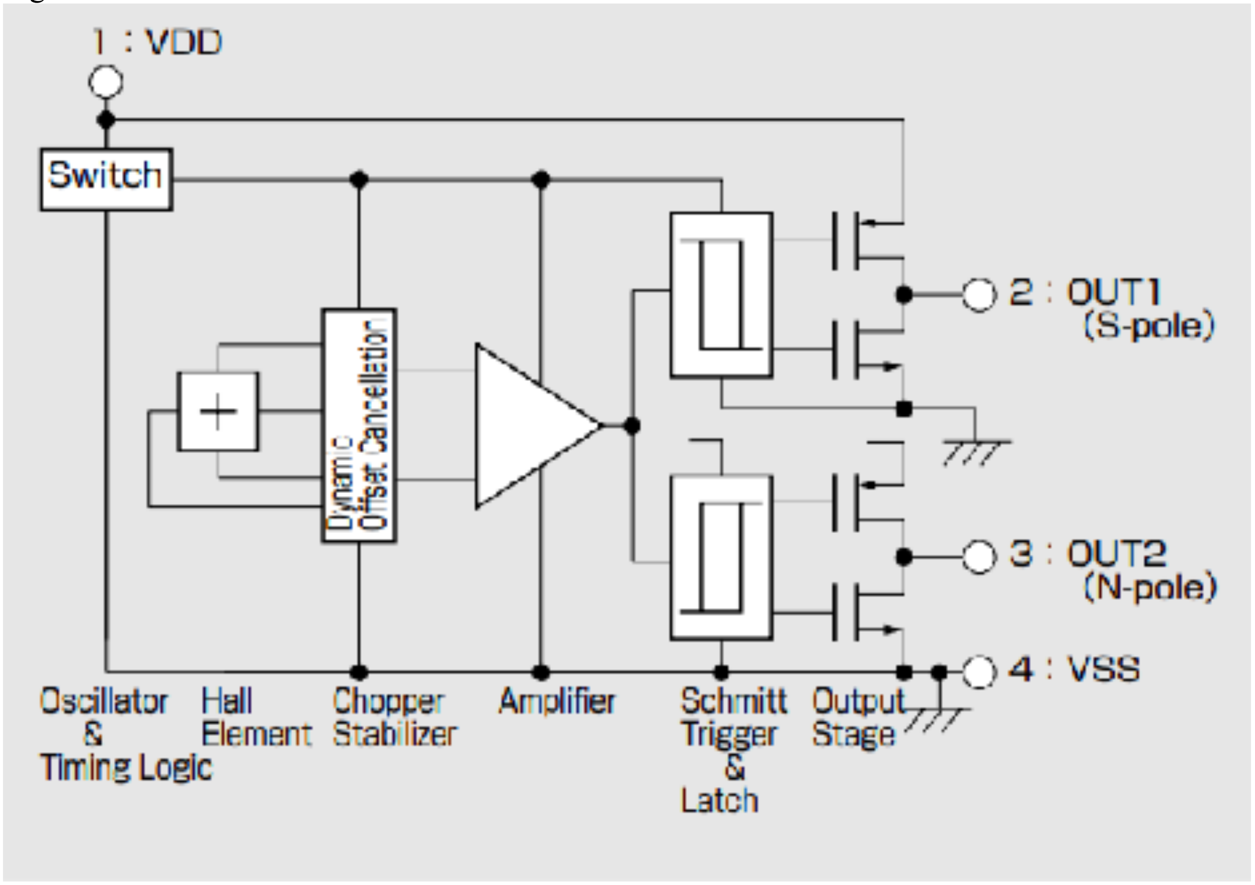

U11



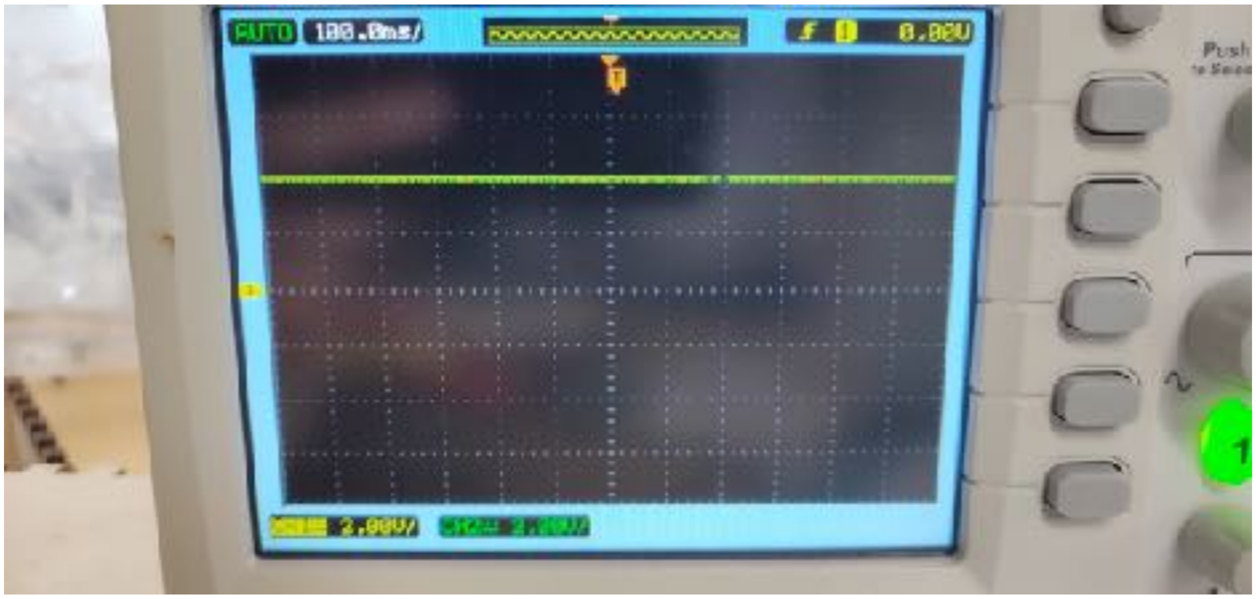
Theory

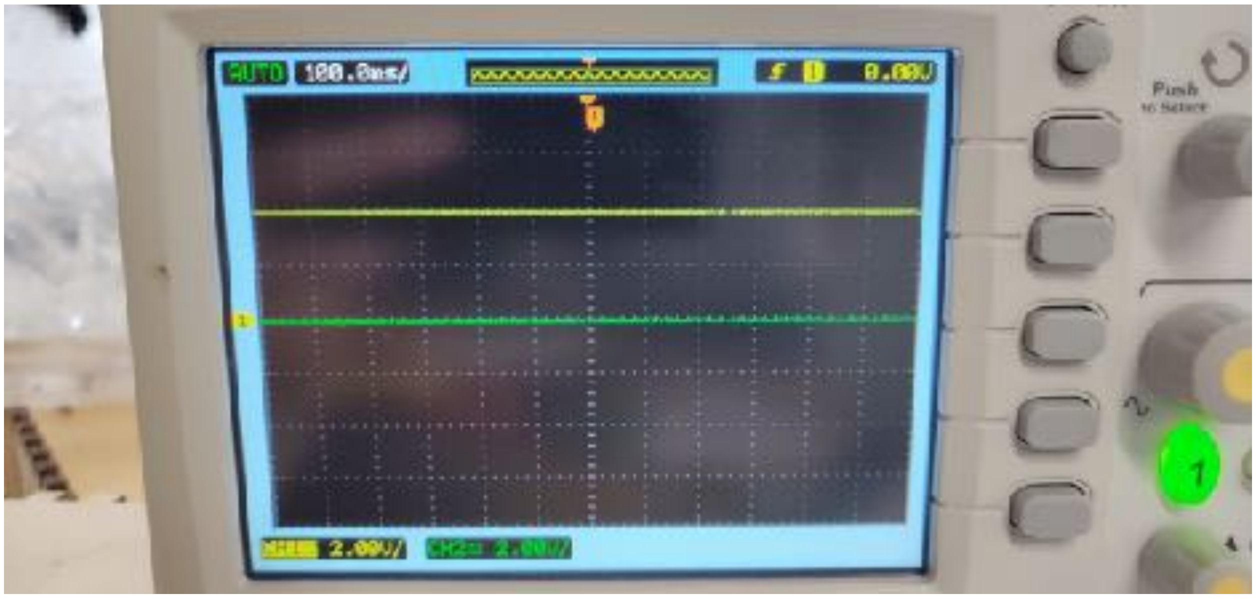
To record the location of the pieces the board uses dual-output hall chip, the internal circuits for which can be seen in Figure 14 below. The chip has two inputs (VDD for power and VSS for ground). There are also two outputs, one that activates on the North, and the other on the South pole of the magnet. Both the outputs are connected to an Output Stage, which stops the flow of current to the output of the hall chip when a current is applied to the side facing the hall element. The Schmitt triggers turn from off to on when interacting with a magnet, and cause the output of the hall element. This means that when a magnet is placed against the chip, the hall element outputs a current, which then is passed through whichever trigger is open, the S-latch if it's South, or the N-latch if it's North. The hall current is then passed through a chopper stabilizer and an amplifier to allow for the outputs to successfully read the signal sent by the element. It's this mechanism that allows us to see if a white piece (N-side of a magnet) or black piece (S-side) is currently over any given square of the board. The default output of the hall chip is detailed in

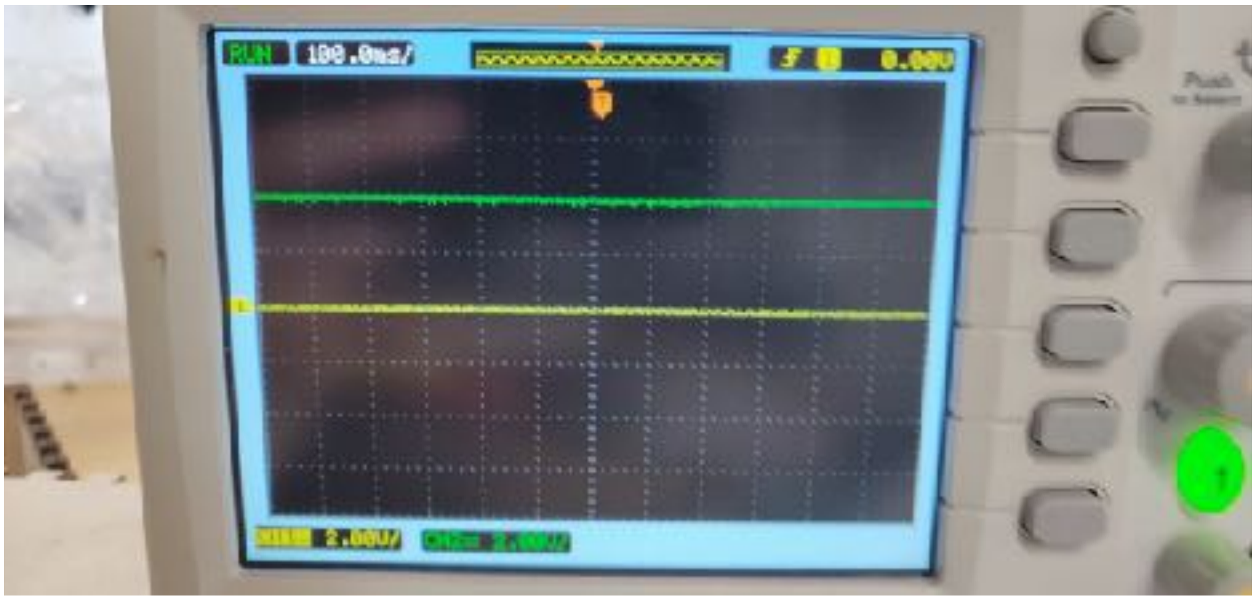
Figure 15.



The green line denotes the output from the north output of the chip, while the yellow denotes the south output, each triggered by their respective side of a magnet. Once the north side of a magnet is placed on the chip, the hall element creates a current, the switches direct it the the correct output, and we see the result visible in Figure 16. In Figure 17, it is show the result of a south side of a madnet being placed near the hall chip. Once again, the voltage drops once the magnet is placed a distance close enough to trigger the chip.

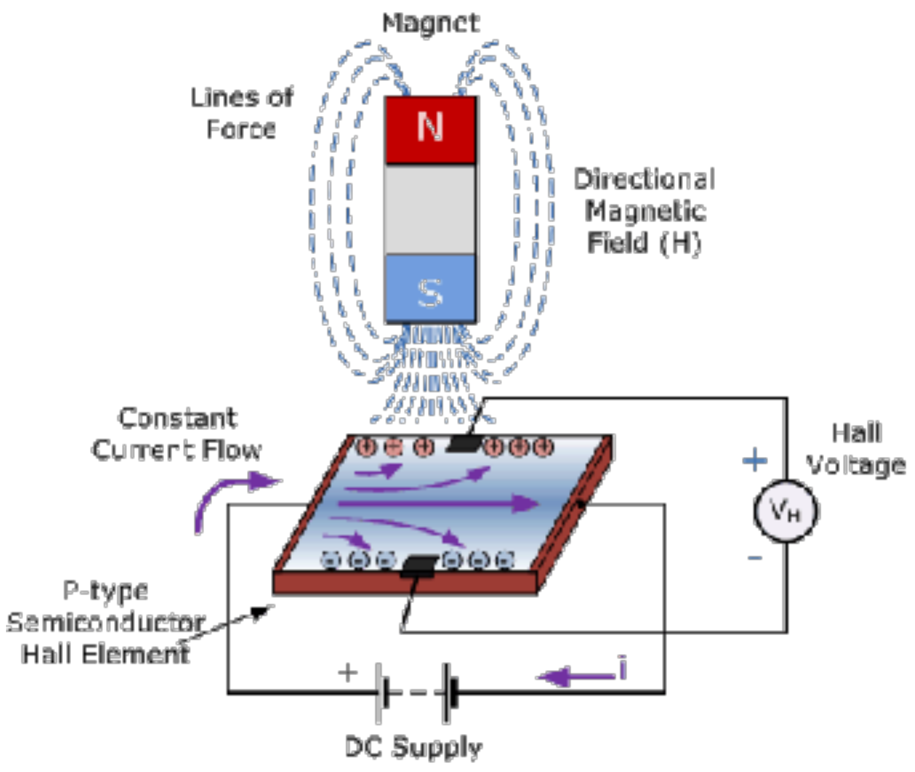

AUTO
100.0ms/
0.00V
2.00V/
CH2= 2.00V/
Push
1


AUTO
100.0ms/
0.00V
2.00V/
CH2= 2.00V/
Push to Select
1



The hall element works based on the premise on the Hall Effect. Figure 18[6] detail a basic form of the hall effect. A current through a semiconductor essentially splits into positive and negative charges on either edge of the plane when a magnet is placed near the plane of the element. This causes a change in voltage to occur between the different sides of the plate and a voltage drop and current flow across the entirety of a hall chip, which has an output pin (of in the case of the EM-1791, two) through which the current is able to flow. The chip needs the input and output in order to pass the current through the plate in order to observe the effect. The chip I am using also requires the latches discussed earlier to have multiple different outputs where a north magnet activates one while a south magnet activates the other.

[6] https://blog.digilentinc.com/what-is-the-hall-effect/, Accessed March 24th, 2019



Results

The hall chips generally triggered at a distance of up to 2 inches, with nearly 100% frequency and nearly never triggered on the opposite pole when the magnet was brought into range. When the area above the sensor is wood (rather than air) the sensors trigger at around 1.5 inches. There was still not problem with both double triggers as well triggering both outputs of a single hall chip. With all this said, the magnets need to be nearly exactly vertical of the hall chips, as even a half inch shifted off the top of the chip no longer triggers the sensor. While the chips trigger easily though the above materials, a gap between the wood and the chip (so .5 inches of air, then an inch of wood) fails to consistently trigger the chips.

Conclusion

While the project so far isn't a complete functioning chess board, it shows several things: the first, is that the information given by the used sensor is enough to track an entire game of chess. The second, is that the specific hall chip used: The EM-1791 from Asahi Kasei Microdevices is a perfect option for the necessary requirements for a functioning board, with a good range, high accuracy, few false positives and almost no observable noise. Despite the undeniable functionality of the hallchip, and the theoretical success of the tracking, there are a handful of reasons the project as created is far from the optimal way to create a chess board. The first, is the size of the through holes on the PCB: the wiring needed was far too delicate. And the second is that despite it being possible, the code to track the game is very complicated, and special moves like en passant and castling are both very difficult to account for. In the end, the project is a cool proof of concept for the idea of minimalistic sensors to accomplish a goal, and a cool test of the hall chip technology, but likely isn't a viable option towards a easy to create model.

Next Steps

The next step of this revolves around taking the successful sensing of the board and the processing of the inputs that has already been done, and combining it with the theory of the chess board shown above. Both theoretically show that this kind of chess board is possible, however, it remains to be show that this board is able to connect those two components. There are still other challenges, such as trying to keep track of an entire board state, but the problems addressed in the conclusion make it very difficult that such a final product being created out of this project are highly unlikely and potentially even impossible. With a more streamlined pcb board setup, or a less fragile wire connecting the board to the arduino, the progress would be smoother.

Acknowledgements

To start, a huge thank you to Dr. Dann for supporting both me and this project from start to finish: whenever something needed to happen for the project to be completed, Dr. Dann always helped make it happen, from supplies to technology needed to put those supplies together. Another thank you to Mr. Ward, for helping in a variety of small ways over the course of the project. Without his help on the random things there isn't any way this project could make it as far as it did. Also, thank you to Ian Walker at GMW Associates for providing me with the dual-output hall chips, they truly are the part that made this project possible.

Appendix A: Table of Parts

Item	Reason	Price
64 EM-1791	Sensing pieces on squares	Free
16 PCBs	Mounting hall chips	$150
2 TM1637	Chess clocks	~$12
Arduino	Taking in signals from the chips and button	$20
Push Button	Mark when a move is complete	~$1
LED Strip / LEDs	To display the advantage of a given player on the board.	~$2
32 Small Magnets	To place in the pieces to trigger the hall chips	~$10
Tournament Size Chess Set	To play chess.	~$15 (more if the set is actually nice)
16 .1uF Capacitors	To remove high frequency voltages from the circuit involving the hall chips.	~$3

Appendix B: Arduino Code for Live-Updating Checkerboard

```
/
*****************************************************************
****************
Mux_Analog_Input
SparkFun Multiplexer Analog Input Example
Jim Lindblom @ SparkFun Electronics
August 15, 2016
https://github.com/sparkfun/74HC4051_8-Channel_Mux_Breakout

This sketch demonstrates how to use the SparkFun Multiplexer
Breakout - 8 Channel (74HC4051) to read eight, separate
analog inputs, using just a single ADC channel.

Hardware Hookup:
Mux Breakout ----------- Arduino
     S0 ------------------- 2
     S1 ------------------- 3
     S2 ------------------- 4
     Z -------------------- A0
    VCC ------------------- 5V
    GND ------------------- GND
    (VEE should be connected to GND)

The multiplexers independent I/O (Y0-Y7) can each be wired
up to a potentiometer or any other analog signal-producing
component.

Development environment specifics:
Arduino 1.6.9
SparkFun Multiplexer Breakout - 8-Channel(74HC4051) v10
(https://www.sparkfun.com/products/13906)
*****************************************************************
****************/
/////////////////////
// Pin Definitions //
/////////////////////
const int selectPins[3] = {2  , 3, 4}; // S0~2, S1~3, S2~4
const int multi[6] = {A0,A1, A2, A3, A4, A5};
const int zOutput = 5;
const int zInput = A0; // Connect common (Z) to A0 (analog
input)


void setup()
{
  Serial.begin(9600); // Initialize the serial port
  // Set up the select pins as outputs:
  for (int i=0; i<3; i++)
  {
    pinMode(selectPins[i], OUTPUT);
```

Appendix C: Processing Code Displaying Checkerboard

```
import processing.serial.*;

Serial myPort;  // Create object from Serial class
String val;     // Data received from the serial port
String topleft4[] = new String[48];

int squareSize = 120; // Dimensions ask for 40x40
int cols, rows;



void setup()
{
  // I know that the first port in the serial list on my mac
  // is Serial.list()[0].
  // On Windows machines, this generally opens COM1.
  // Open whatever port is the one you're using.
  String portName = Serial.list()[0]; //change the 0 to a 1 or
2 etc. to match your port
  myPort = new Serial(this, portName, 9600);

   size(960, 960); // Specified by assignment

  // Initialize columns and rows
  cols = 4; // Specified by assignment | size/squareSize
  rows = 8; // Specified by assignment | size/squareSize
}

void draw()
{
  delay(2000);
  if ( myPort.available() > 0)
  {  // If data is available,
   //println(myPort.readStringUntil('\n'));
   try
   {
    // print("START");
   //  print(myPort.readString());
//print("STOP");
     String input = myPort.readString();
     topleft4 = split(input, "\n");
   }
   catch (NullPointerException l)
   {

     print("sad");
   }
  }// read it and store it in val
   for (int slot=0; slot<=47; slot++)
```