Automated Chess

Team:

Name	Email	NetId
Julian Gonzales	jgonz96@uic.edu	jgonz96
Dan Hrubec	dhrube2@uic.edu	dhrube2

Abstract:

Automated Chess using Arduinos. The Arduino will take in input from the user on an input square and output square. Then using Serial Communication, it will communicate the user input to a second Arduino. This Arduino will then read in the input, being the starting square and the target square. Using stepper motors, it will move the piece from the starting square to the target square.

Materials Needed:

- **3 metal rods**
- **Chess pieces**
- **Set of buttons**
- **Two timing belt**
- **Epoxy**
- **Double Sided Tape**
- **2 stepper motors**
- **Sheet of glass**
- **Electro magnet**
- **Metal discs for chess pieces**
- **Sheet Metal**

Overall Description:

The project will consist of two arduinos controlling a game of chess. At first between two players and if there's extra time a simple AI. it would use the first to control the communication between the two players and the second to control the movement of the pieces

How two arduinos will be used:

The first arduino will control the communication between the players and possibly if there's time control a simple AI that plays against the player.The arduino will serve as the user input portion. It will use multiple push buttons to allow the user to make a selection on what square the piece is currently at, and then make a selection on where to move the piece. The input will then be validated before it will be passed to the second arduino. The arduinos will communicate via the Serial Monitor, as there is enough distance between the two to be wired together.

The second arduino will control all the hardware used to move the piece mechanically. The arduino would wait on an input to be sent from the previous arduino. It would then start off by calibrating itself so it always starts off in the same position. Based on the coordinates selected from the user, it would use stepper motors to move an electromagnet to the starting square, over to the target square, moving the piece with it.

Communication:

The communication will be done via Serial Communication through the arduinos, hard wiring the two arduinos together. There is not a large amount of physical distance between the two arduinos, so to make the transfer of information as consistent and reliable as possible, we could hard wire them together. It is possible to use other forms of communication like bluetooth, but Serial Communication would be the most reliable, giving us the best possible result in this situation. The first arduino would gather the input from the player, and convert the data into a string. Then the second arduino would read in the string, parse the information, and make the movements accordingly.

Inputs and Outputs:

The input would be given by the players, and would be tiles based on the chess board. The player would pick a starting tile, and a target tile. The output would be using stepper motors and electromagnets to perform this movement for them. Based on the inputs, the stepper motors would move the electromagnets and move the piece to the target tile. This movement would be similar to how a 3d printer would function.

Original Work:

Our original work would be the mechanical movement of the pieces. Online, there have been other variants of chess, and having the pieces being represented by LEDs, but no other project would physically move an existing chess set according to the players inputs. Our implementation of the project would be unique to include the usage of stepper motors, rails, and electromagnets.

How to Build Project

The project build can be separated into different sections. Given the required materials, the project can easily be replicated. Starting off with the input arduino. Wiring for a 16x2 LCD screen and 5 push buttons are required. Four of the push buttons will be aligned in a directional pad, and the last being offset to the side. The four aligned buttons control the movement, which is reflected on the 16x2 LCD screen so the player knows what tile is currently being selected. The last buttons is to proceed to the next set of coordinates, then pressed again to validate the coordinate set before being sent to the other arduino.

First create four holes centered and in the middle of each pvc pipe and allow room at the top and bottom for the rail system to travel through. Make sure to create the holes first since we made the mistake of creating the holes after sticking it to the board. Then take the sheet of glass preferably something around the size of a typical chess board and stick the 4 pvc pipe connectors on each corner acting as feet for the board to be suspended since there has to be room for the mechanical components under the board. Each pair of holes facing each other to place the rods. Use epoxy to stick the pvc pipe. After this cut the 2 metal rods with either a grinder or some type of cutting wheel to the length of at most the distance from the center of one pipe to the other. This gives us room to slide it into the holes created and also the ability to remove them. Making them longer will make it impossible to put into place.

Then after the rails are put into place create a case for the 3rd rail that spans the two rails at each pair of pvc footing. This would be your y axis. After the case is built use bearings preferably ones that are small and the width of the rail you are using to encase the rail at each end of the case so that it smoothly moves along the rail. This is important since slight differences in resistance between the 1st and 2nd rail will cause it to become stuck or crooked. I also recommend using a 3d printer to print this casing as anything that is slightly off will cause some slop and inaccuracies. After the case for the 3rd rail is attached at both ends use a timing belt and attach it to the middle of the case spanning the entire board.

This will pull the case containing 3rd rail along the x axis and cause it to move back and forth. Use a bracket to attach the stepper motor to the board at one end and another bracket to attach the gear it will ride on at the other end. The stepper motor will be attached to a stepper motor controller which will then be connected to the arduino. After the first timing belt is attached create the yaxis rail and manufacture a cart for the electromagnet to ride on and attach it to the yaxis rail. There should be some type of bushing for the cart to ride on that provides low friction. We decided to use a skateboard wheel spacer and it gave it just enough space to move without causing alot of slop. Once again create a small bracket on this rail or attach the stepper motor to the case of the y rail and then a gear on the other end for the timing belt to ride on. Attach the timing belt to the cart and that should result in 2d movement.

After the 2d movement is done construct an electromagnet or use the purchased on. Place it on the cart on the y rail and this should be flush with the sheet of glass in order to provide the most power. Use a power supply to power the electromagnet and a relay to turn it on with the arduino. Finally construct the input for the user and that should consist of some buttons on a board and display. One arduino will control the input and another will control the movement and the electromagnet

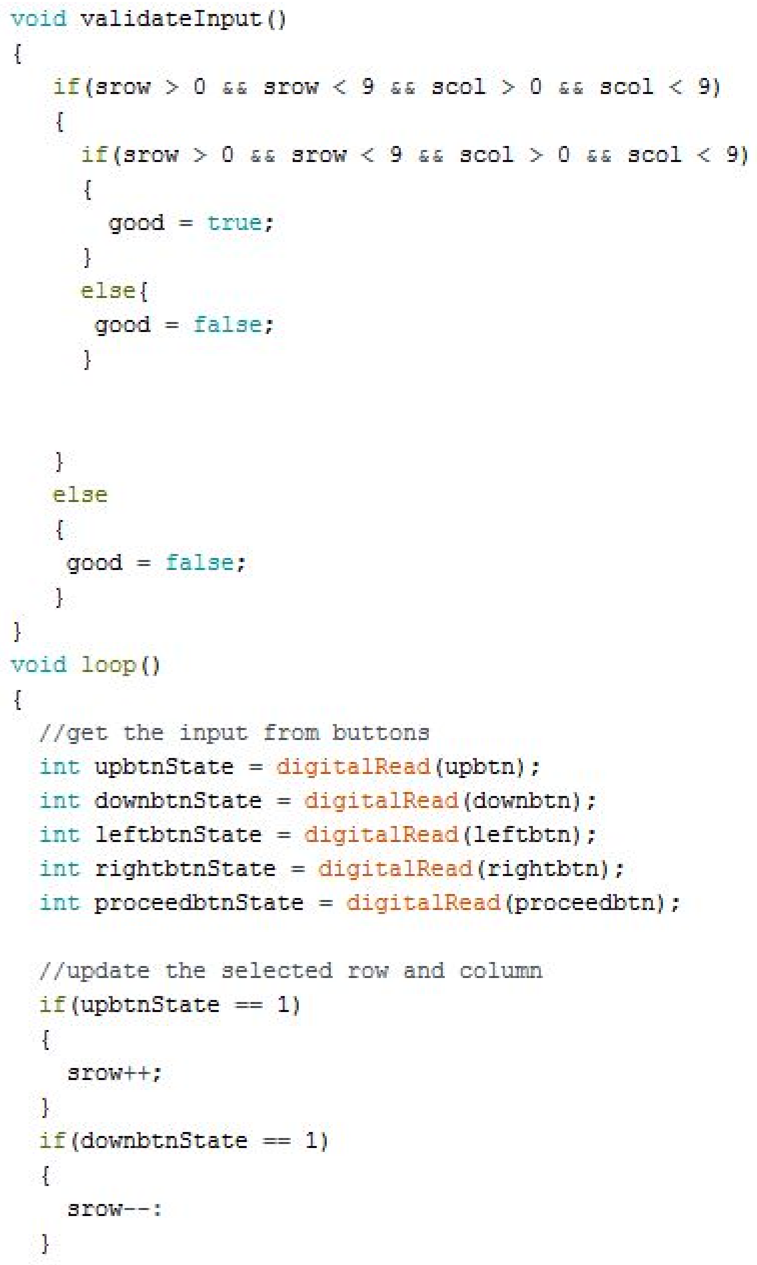
How to Use Project

Assuming the project is fully assembled, the usage of the build is straightforward. A directional pad, a proceed button, and a 16x2 LCD screen should be the only part the player interacts with. The player would use the directional pad to select a starting and target tile, which is displayed on the LCD screen. After making a selection, the player uses the proceed button, then the second arduino will move the piece based on the player's selection. Players then take turns moving their pieces to play chess.

Code:

Arduino Code for Player Input

```
//Dan Hrubec
//Julian Gonzales
//CS 362 Arduino Project
//Player Input Code

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);


//pin numbers for each of the buttons
int upbtn = 9;
int downbtn = 8;
int leftbtn = 7;
int rightbth = 6;
//button to move onto the next set of input
int proceedbtn = 10;

//keeping track of the selected row and column
int srow = 0;
int scol = 0;

//trow/tcol for target row/col
int trow = 0;
int tcol = 0;

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  pinMode(upbtn,INPUT);
  pinMode(downbtn,INPUT);
  pinMode(leftbtn,INPUT);
  pinMode(rightbtn,INPUT);

  pinMode(proceedbtn,INPUT);
  Serial.begin(9600);

}
```

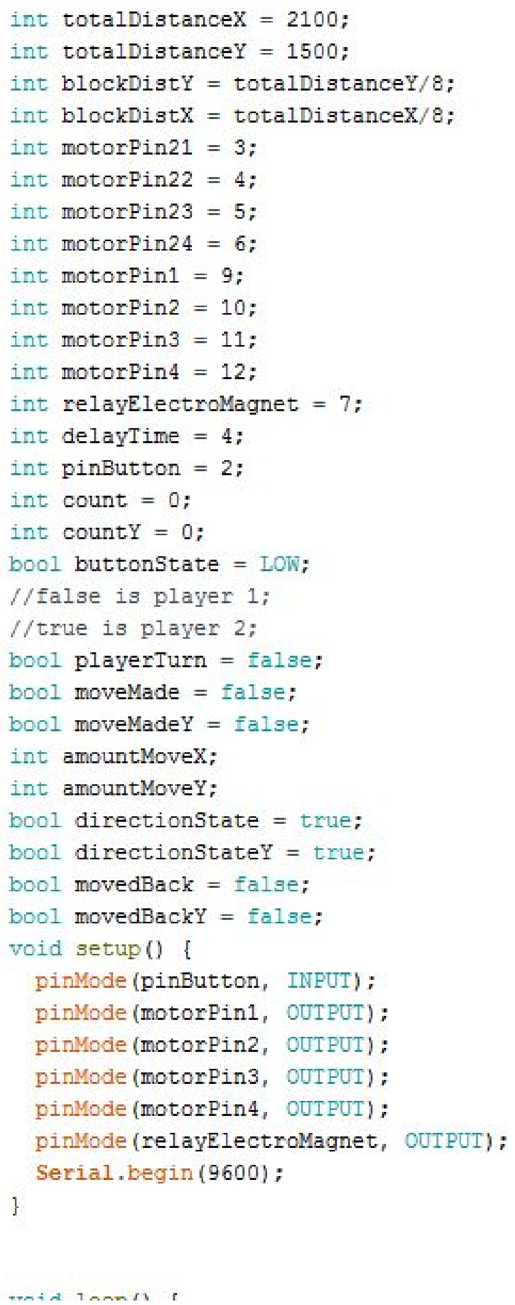


```
void validateInput()
{
   if(srow > 0 && srow < 9 && scol > 0 && scol < 9)
   {
     if(srow > 0 && srow < 9 && scol > 0 && scol < 9)
     {
       good = true;
     }
     else{
      good = false;
     }


   }
   else
   {
    good = false;
   }
}
void loop()
{
  //get the input from buttons
  int upbtnState = digitalRead(upbtn);
  int downbtnState = digitalRead(downbtn);
  int leftbtnState = digitalRead(leftbtn);
  int rightbtnState = digitalRead(rightbtn);
  int proceedbtnState = digitalRead(proceedbtn);

  //update the selected row and column
  if(upbtnState == 1)
  {
    srow++;
  }
  if(downbtnState == 1)
  {
    srow--:
  }
```

```
if(leftbtnState == 1)
{
  scol--;
}

if(rightbtnState == 1)
{
  scol++;
}
//print it on the lcd screen for the user to keep track of where to move
lcd.setCursor(0,0);
lcd.print(srow);
lcd.setCursor(1,0);
lcd.print(scol);

//now if the button to proceed is pressed, we want to do the same thing for the target location
if(proceedbtnState == 1)
{
    if(upbtnState == 1)
      {
    trow++;
     }
  if(downbtnState == 1)
    {
    trow--:
  }

  if(leftbtnState == 1)
  {
    tcol--;
  }
```

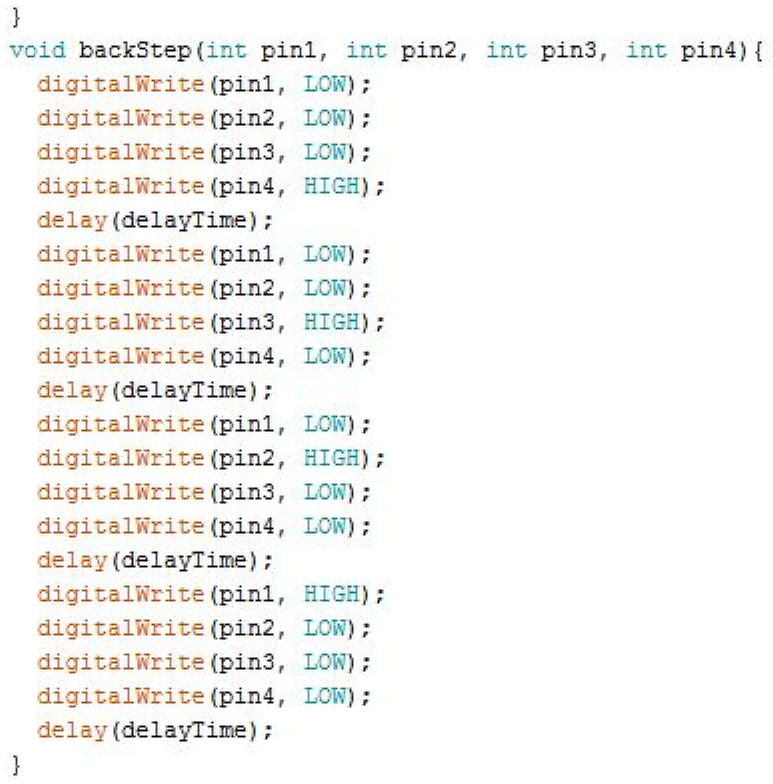
```
      if(rightbtnState == 1)
      {
        tcol++;
      }
      lcd.setCursor(0,1);
      lcd.print(trow);
      lcd.setCursor(1,1);
      lcd.print(tcol);

    }
    //validate our input to make sure we are going to send out valid coordinates to the second arduino
    validateInput();
    if(good == true)
    {
        Serial.write(srow);
        Serial.write(scol);
        Serial.write(trow);
        Serial.write(tcol);

    }



    delay(1000);


  }
```

Arduino Code for Output



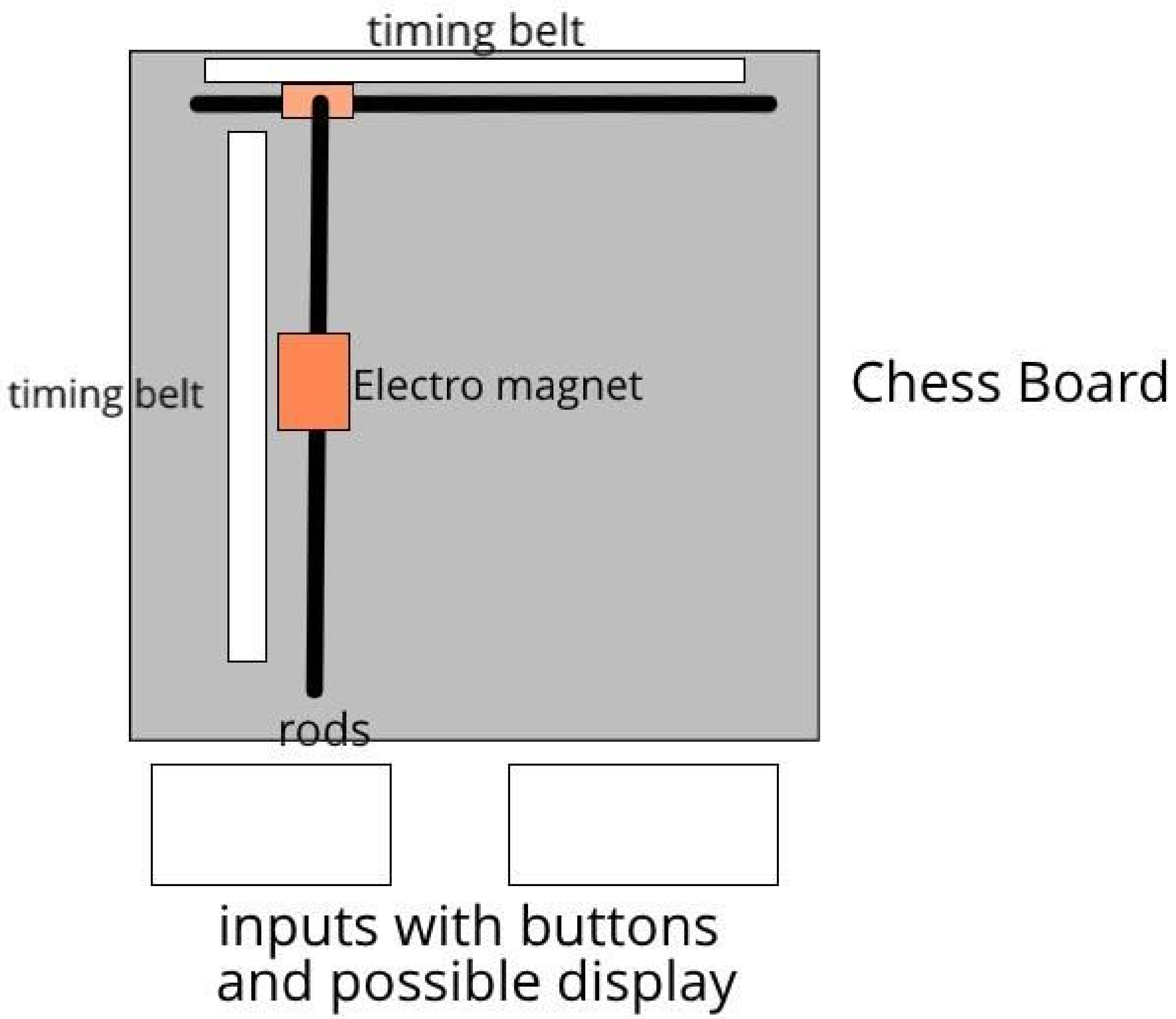
```
int totalDistanceX = 2100;
int totalDistanceY = 1500;
int blockDistY = totalDistanceY/8;
int blockDistX = totalDistanceX/8;
int motorPin21 = 3;
int motorPin22 = 4;
int motorPin23 = 5;
int motorPin24 = 6;
int motorPin1 = 9;
int motorPin2 = 10;
int motorPin3 = 11;
int motorPin4 = 12;
int relayElectroMagnet = 7;
int delayTime = 4;
int pinButton = 2;
int count = 0;
int countY = 0;
bool buttonState = LOW;
//false is player 1;
//true is player 2;
bool playerTurn = false;
bool moveMade = false;
bool moveMadeY = false;
int amountMoveX;
int amountMoveY;
bool directionState = true;
bool directionStateY = true;
bool movedBack = false;
bool movedBackY = false;
void setup() {
  pinMode(pinButton, INPUT);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
  pinMode(relayElectroMagnet, OUTPUT);
  Serial.begin(9600);
}


void loop() {
```

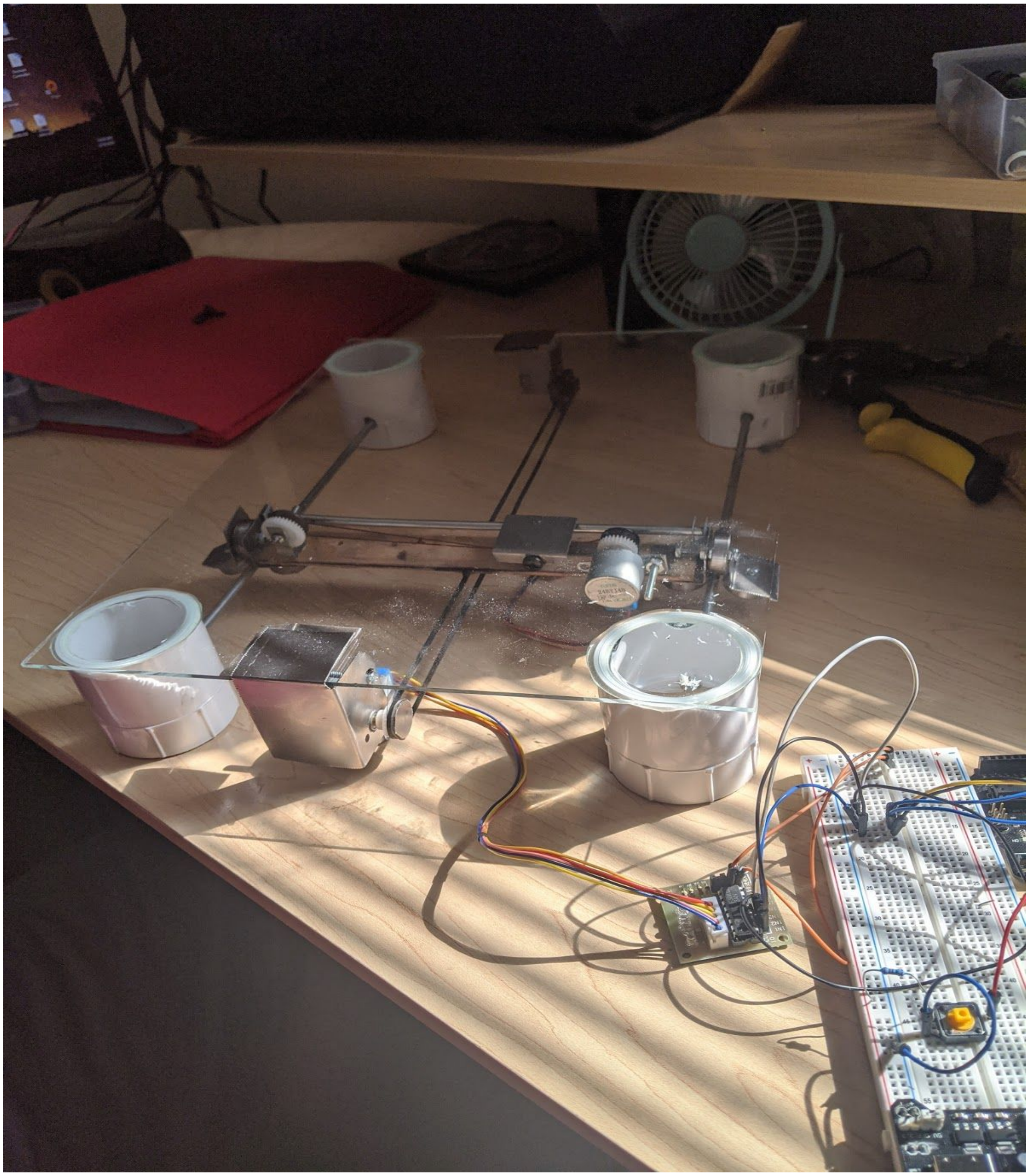
```
void loop() {
  if(!moveMade && !moveMadeY){
    //would receive input from second arduino instead of Serial input
    int playerMove = Serial.parseInt();
    int playerMoveY = Serial.parseInt();
    if(playerMove > 0 && playerMoveY > 0){
      amountMoveY = playerMoveY * blockDistY;
      amountMoveX = playerMove * blockDistX;
      moveMade = true;
      moveMadeY = true;
      Serial.print(playerMove);
    }
  }

  //once both moved to spot then turn on electro magnet
  if(countY == amountMoveY && count == amountMoveX){
    //turn on electro magnet relay
    digitalWrite(relayElectroMagnet, HIGH);
    //wait for it to turn on
    delay(4000);
  }


  //move for x
  if(moveMade){
    if(movedBack && directionState ){
      moveMade = false;
      movedBack = false;

    }
    //wait for both to finish moving
    if(countY == amountMoveY && count == amountMoveX){
      directionState = !directionState;
      count = 0;
      movedBack = true;
    }
    count = count + 1;
    //switch to other direction
    if(directionState){
      forwardStep(motorPin1, motorPin2, motorPin3, motorPin4);
    }else{
      backStep(motorPin1, motorPin2, motorPin3, motorPin4);
    }
  }
  //move for y
  if(moveMadeY){
    if(movedBackY && directionStateY){
      moveMadeY = false;
      movedBackY = false;
    }
    //wait for both to finish moving
    if(countY == amountMoveY && count == amountMoveX){
      //turn on relay for electro magnet
      directionStateY = !directionStateY;
      countY = 0;
      movedBackY = true;
    }
    countY = countY + 1;
    //Switch to other direction
    if(directionStateY){
      forwardStep(motorPin21, motorPin22, motorPin23, motorPin24);
    }else{
      backStep(motorPin21, motorPin22, motorPin23, motorPin24);
    }
  }
}


void forwardStep(int pin1, int pin2, int pin3, int pin4){
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(delayTime);
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, HIGH);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(delayTime);
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, HIGH);
  digitalWrite(pin4, LOW);
  delay(delayTime);
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, HIGH);
  delay(delayTime);
```

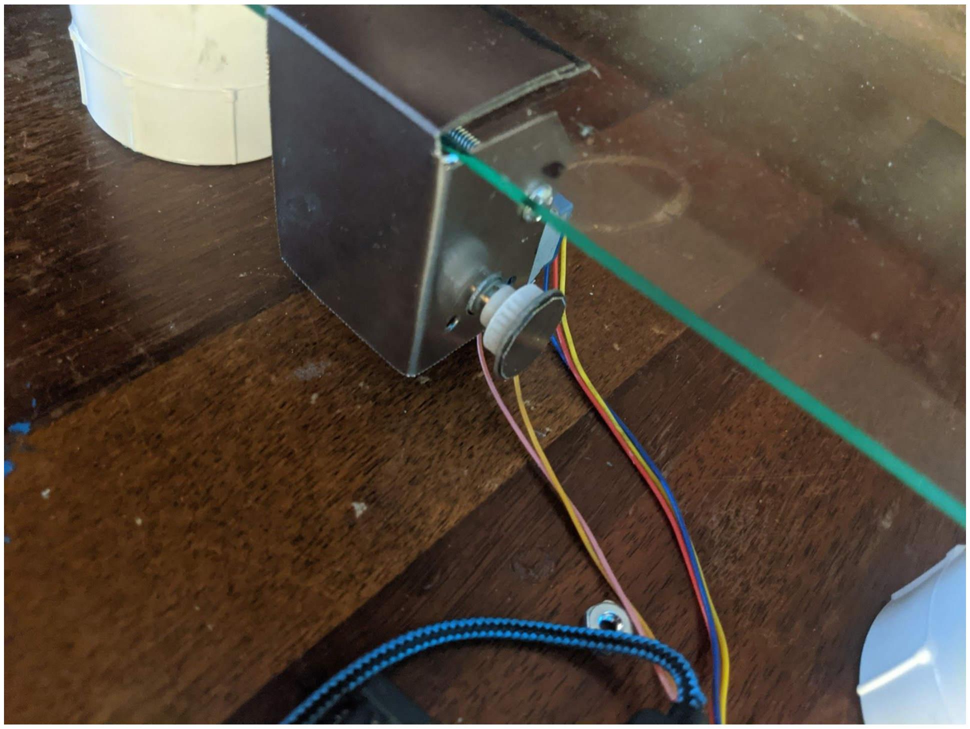


```
}
void backStep(int pin1, int pin2, int pin3, int pin4){
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, HIGH);
  delay(delayTime);
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, HIGH);
  digitalWrite(pin4, LOW);
  delay(delayTime);
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, HIGH);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(delayTime);
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(delayTime);
}
```

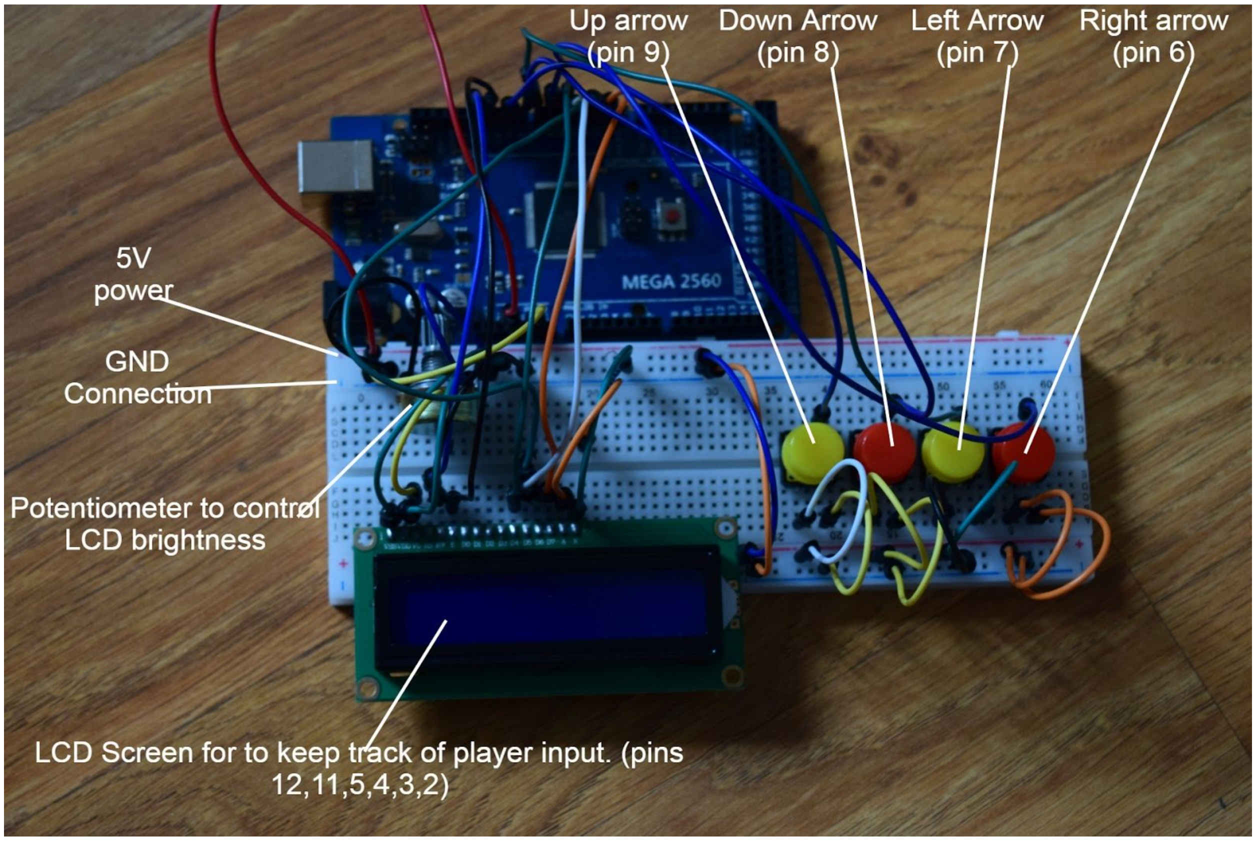
Hardware Diagrams







YOFON
24BYJ48
12V DC


Up arrow
(pin 9)
Down Arrow
(pin 8)
Left Arrow
(pin 7)
Right arrow
(pin 6)
5V
power
GND
Connection
Potentiometer to control
LCD brightness
LCD Screen for to keep track of player input. (pins
12,11,5,4,3,2)
MEGA 2560

Build Schedule

Date	Activity
Week 1	Gathered all required materials for the build.
Week 1	Using the materials, worked out a rough prototype mechanical movement.
Week 2	Created a working arduino to get input from the player.
Week 3	Using the prototype, made adjustments to get the pieces to move, not necessarily with precision.
Week 4	Worked out communication between the two arduinos.
Week 4	Validated the user input before sending information to second arduino.
Week 5	Test and work out any bugs
Week 6	Re-adjusted design and implementation due to COVID-19.
Week 7	Testing and fixing bugs.
Week 8	Testing and fixing bugs.
Week 9	Refined design and presentation, recorded video on project.
Week 10	Demonstration

Problems Encountered & Resolutions:

Highly recommend the use of a 3d printer to obtain the tolerances needed for it to run smoothly. Do not permanently attach the rails as removing them for trouble shooting is very useful. Also make sure the rails are as parallel as possible as it will cause problems if slightly off. Also using a power supply that is more powerful is also recommended as ours was not strong enough to grab pieces through glass.

What Worked/Didn't Work:

The board did move to certain positions but was not able to accurately move to a spot because of tolerances this is why a 3d printer is recommended for some parts. Also the electromagnet was not strong enough to grab pieces and hold on to them. Input was done through serial communications but could easily be implemented with the other arduino that communicates with the user

Potential improvements:

More resources like the maker space. Better tools to cut sheet metal accurately. Better materials suited for the job like appropriate bearing size and bushings. Appropriate stepper motor sizes and experimenting with different stepper motors of different torque.

List of References

- https://www.arduino.cc/en/Tutorial/Button
- https://www.arduino.cc/en/Tutorial/LiquidCrystalScroll
- http://arduino.cc/en/Reference/Serial#.UwYyzfldV8E
- http://arduino.cc/en/Serial/Available#.UwYy2PldV8E
- http://arduino.cc/en/Serial/ReadBytesUntil#.UwYy6_ldV8E
- https://www.arduino.cc/en/Reference/Stepper
- https://www.arduino.cc/en/tutorial/stepperSpeedControl
- https://www.arduino.cc/en/Tutorial/StepperOneStepAtATime