Billy's Checkers

School of Engineering and Applied Sciences
Harvard University

Cambridge MA.

An electronic LED and push button board that allows you to play checkers.

By: Victor Yang, Ryan Wood, Billy Koech, Sean Park

Class: Engineering Sciences 50
Instructors: Marko Loncar & Christopher Lombardo

December 2016

ABSTRACT

Billy's Checkers is an electronic checkerboard controlled using an Arduino microcontroller and powered using a single 9V battery. It allows two users to play the game of checkers, much like you would on a normal cardboard board. Each of 32 squares on the board contains an RGB LED and a push button to dictate to the board which piece to move and where. These two elements are wired through different circuits. Due to limits on the number of pins on the Arduino, creativity was used to control all 32 LEDs and push buttons. For the LEDs, two sets of four daisy-chained SN74HC595 shift registers write a 32-bit binary number to the board, where each bit controls the LED for one square, and the two sets of shift registers control red and blue colors for each player. For the push buttons, 8 of them are connected in series, with one side of the buttons shorted (connected to one Arduino analog input pin) and the other side containing a 2.2k Ohm resistor between each button (connected to power)[2]. In this way, when power is on and a button is pressed, each button opens a circuit with a different amount of resistance, causing a different voltage to be read by the Arduino. The problem with this idea is that two buttons cannot be pressed at the same time, but that should never happen in a checkers game.

INTRODUCTION

The goal of the project was to make checkers more exciting from its typical bland board and pieces. Additionally, we wanted to make to make the setup process easier and avoid the confusion of missing pieces. To that end, we decided to create an electronic checkerboard where RBG LEDs take the place of pieces. Blue and red represent each of two players, and moving a piece simply involves clicking a push button next to the piece you want to move and then clicking the square to move it to. The board would disallow any invalid moves and we recognize when a player won. Setting up the board would only take a push of a button, and all the LEDs are soldered in place so there is no possibility of losing them.

DESIGN

The hardware design aspect of this project focused primarily around two problems: controlling 64 LED inputs with a limited number of Arduino output pins, and taking input from 32 tactile push switches from a limited number of Arduino input pins. The software design was a separate issue.

We decided to use 32 RGB LEDs, one for each of the 32 game board squares that are used during gameplay. This was because the alternatives involves two single-color LEDs, which looked bulky and would require wiring another set of LEDs to ground. Red was to represent a piece from one player, and blue was to represent a piece from the other player, adding up to a sum of 64 total LED outputs to control. As the Arduino only has 13 GPIO pins, we decided to use shift registers to control the LEDs. This was largely because it was the only method that any of us four knew well, and other solutions seemed to be much more complicated anyways. We used two daisy-chains of shift registers, each chain controlling one color of LED (kindly see **figure 3** for a diagram of the LED circuit with the shift registers). In this way, we could still output two colors from the same Arduino. Thus, each daisy chain was in control of 32 LEDs, meaning each was 4 shift registers long. Each daisy chain was controlled by a different clock and different data pins, but shared the latch input. We discovered that this was necessary during the implementation process written about below. The shift registers are rated to optimally source no more than 6 milliamps; current limiting resistors were carefully selected based on what was used in lab 6[1] and attached to each LED to ensure this current limit was not exceeded.

On each square that had an LED, there was also a tactile push switch to sense user input (kindly see **figure 2** for a diagram of the push button circuit). Having 32 inputs was a challenge. Our first idea was to use shift registers in an input configuration. However, we abandoned this idea for another simpler idea that would allow for less wiring in the circuit. Each row of buttons (four total rows of eight buttons each) was attached on one side to a voltage divider, such that the potential on one side of each button was different. The other side of the buttons were shorted together and connected to an analog input pin on the Arduino. In our code, we could determine which button was pushed by reading the analog value on the input pin. To prevent the voltage on the input pins from floating while not in use, they were all attached to ground through a high value resistor. This was deemed necessary through an instructional video found through online research[2].

In addition, we added the capability to run the system off a 9V battery source rather than USB power from a computer. The benefit of this was portability, as a laptop was no longer needed to power the apparatus. Although the Arduino has an on-board regulator, allowing it to accept up to 12V power, that regulator is low powered and inadequately cooled as known through prior experience. We added a 5v regulator in a TO-220 package to increase the power capacity and heat dissipation, knowing that controlling up to 32 LEDs may add up to hundreds of milliamps of current draw.

The software part of the project involved reading the analog inputs, analyzing the situation, and writing to the shift registers in sequence. Writing to the shift registers involved sending two 32-bit binary numbers to the SHIFT_OUT function, each bit representing an LED, and each 32-bit number writing to a different shift register. We decided to display LED output in this manner because it only involved one line of code and, by adding up the decimal values of all the pieces that we wanted to light up at one time, we could store the state of the board in one simple number, making the code much easier to write. We wrote several functions that allowed the movement of certain pieces to certain positions, each position having a unique set of possible moves. Two arrays kept track of the status of each square (on or off); this made the most sense because it became much more easy to access the current state of any one LED and reference that state with our overall number representing the unique state of the board.

PARTS LIST

Part Name	Source	URL	Unit Price	Quantity	Total Price
RGB LED	Lab	https://www.amazon.com/Chanzon-100pcs-Emitting-Multicolor-Tri-color/dp/B01C19ENFK/ref=sr_1_3?ie=UTF8&qid=1478130026&sr=8-3&keywords=5mm+rgb+led	$8.96/100 pack	32	$8.96
Tactile Push Switch	Lab	https://www.adafruit.com/product/367	$2.50/20 pack	32	$5.00
Foam Board	Lab	https://www.walmart.com/ip/Elmer-s-Guide-Line-Paper-Laminated-Polystyrene-Foam-Display-Board-20-x-30-Black/14940238	$8.79	20” by 20”	$8.79
Breadboard	Lab	https://www.adafruit.com/product/239	$5.95	2	$11.90
Ribbon Cable	Lab	https://www.sparkfun.com/products/10647	$4.46/18’	9’	$4.46
Hookup Wire	Lab	https://www.sparkfun.com/pro	$2.25	1 roll	$2.25

		ducts/8022			
Jumper Wire	Lab	https://www.sparkfun.com/products/11026	$1.76	1 pack	$1.76
Arduino Uno	Lab	https://www.amazon.com/Arduino-Uno-R3-Microcontroller-A000066/dp/B008GRTSV6/ref=sr_1_1?ie=UTF8&qid=1478129794&sr=8-1&keywords=arduino	$16.06	1	$16.06
Shift Register	Lab	http://www.digikey.com/product-detail/en/texas-instruments/SN74HC595N/296-1600-5-ND/277246	$0.58	8	$4.64
Acrylic Sheet	Lab	http://www.homedepot.com/p/LEXAN-10-in-x-8-in-Polycarbonate-Sheet-31-GE-XL-1/202090134	$4.26	9" by 18"	$4.26
5V Voltage Regulator	Lab	http://www.digikey.com/product-detail/en/stmicroelectronics/L7805CV/497-1443-5-ND/585964	$0.44	1	$0.44

Grand Total: $68.48

PROJECT IMPLEMENTATION

Note: The links in parenthesis are to video segments displaying the implementation process in action.

A. First we started by building the LED circuit with shift registers on a breadboard. We chose to have two daisy chains of shift registers[1], each with four shift registers, to control 32 RGB LEDs. We planned to have one daisy chain control the red components of the LEDs and the other daisy chain control the blue components.
B. Getting the shift registers to work was tough partly because we had faulty breadboards. Once we got new breadboards we realized that we had another challenge still: the shift registers were not responding to our code as expected. With the help of Professor Loncar we solved this issue by implementing two separate clock pins, one for each daisy chain of shift registers. Previously, we would write out the number written into the first daisy-chain once we started on the second because we had the same clock that would still open the first set for writing when we incorrectly thought it would be left untouched. Here is a video segment showing the daisy chains working after innovating with two different clocks: https://youtu.be/GNGKmRxZbts?t=1m15s

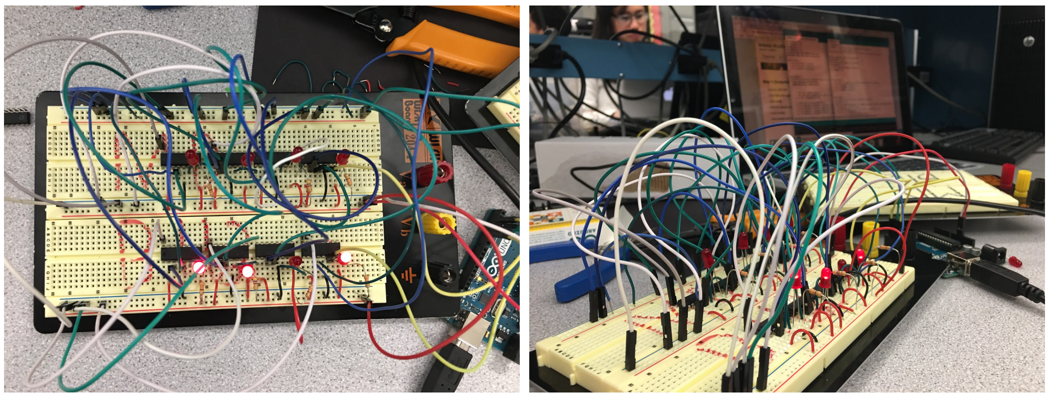
Figure 1: The two sets of four daisy-chained shift registers

C. The daisy chains together with the LEDs would provide the output for our board, but we still needed to take input from users. For this part, we put together a circuit of resistors in series, and we connected buttons at one terminal to each node of the chain of resistors. The other terminal of the buttons was then connected to an analog input pin of the Arduino (kindly see design section for details). Here is a video segment explaining the button circuit: https://youtu.be/GNGKmRxZbts?t=1m26s

D. With the buttons, LEDs and shift registers in place, we now had all that we need to take input and give output.

E. For the checker board, we used foam board and made holes on it for both the buttons and LEDs. We then placed the buttons and LEDs in their right places (https://youtu.be/GNGKmRxZbts?t=2m5s) and wired the circuit underneath the foam board in the following order. First, we soldered each ground pin on the LEDs to a 330 ohm resistor. Then, we wired them together in series and connected them to the ground pin on the Arduino. We then connected the red and blue pins on the LED separately to the two different daisy chains of shift registers. After that, we proceeded to make the push button circuit by first shorting the terminals that would go to the analog input pin and then connecting the other terminals in series with a 2.2k ohm resistor between each push button (see **figure 4** for a diagram of one square of our physical finished product).

F. We used hot glue to keep pieces on the board from falling off.

G. Once the board, button and LED setup was complete we proceeded to tackle the coding part of this project. Since we hoped to control each piece on the board with its respective buttons we used thresholds to distinguish inputs from different then had these inputs trigger a sequence of variables and functions that would either turn on or off LEDs. In **figure 5** is the code used for the implementation of the board. Kindly see the comments above each line of code to understand the purpose of the code. The working board can be seen in action here: https://youtu.be/GNGKmRxZbts?t=3m30s

TEAM MANAGEMENT

Every time our team went to the lab, we went as a collective unit. We did not divide up the work into large segments, but rather stuck together as a team for every large task, and then split up those large tasks into smaller chunks doable for one person as necessary. For example, one person would be stripping wires while another person soldered, and the other two team members would do the same exact thing on another area of the board. During the earlier stages of the project, each of the four team members would be working on researching different aspects of the project, such as how to send input from push buttons to the Arduino and how to write Arduino code for two sets of daisy chained shift registers. We did not have a team leader, but some team members' strengths lead them to take charge of certain aspects of the project. Ryan knew a lot about different circuits that could be created using the Arduino microcontroller, so he spearheaded the effort to design and build the circuit for the push buttons. Billy was especially passionate about computer science and he lead the effort to program the checkers game itself. With that said, it is important to reiterate that everyone worked a great deal on all aspects of the project.

OUTLOOK AND POSSIBLE IMPROVEMENTS

This project could be improved further aesthetically and in terms of the code for the checkers game if there were more resources and more time. First, in terms of appearances, wiring up the green portions of the LEDs in addition to the red and blue would allow for greater creativity in displaying flashes when a player wins, for showing which piece is a king, or for informing players when they have made an error in gameplay (illegal move, playing out of turn, etc.). Second, moving onto the code for the game, adding a feature that keeps track of which player's turn it is would allow for smoother gameplay, since the board itself would do much more of the administrative work in the game. Third, a smaller board of LEDs could be created that keeps track of how many pieces each player has and how many games each player has won. These improvements are all quite feasible to achieve, and the only barrier that stopped them from being implemented in the original project was time.

ACKNOWLEDGEMENTS:

We would like to acknowledge our project teaching fellow Gregory Hewett. He was a great support for our group throughout the process and especially assisted us in thinking about the appearance of the checkers board. We would also like to give special thanks to Professor Loncar for helping us out with the idea of using two clocks for the two sets of daisy chained shift registers.

DISCLAIMER

Use of our project is for educational purposes only.

REFERENCES:

Note: This is an original project coined by ourselves, and that is why we do not have as many references here.

(1) ES 50 Lab 6 for instructions on how to daisy-chain shift registers:

https://canvas.harvard.edu/courses/16709/assignments/98793

(2) Inspiration (although ours is quite different) for the push button circuit: https://www.youtube.com/watch?v=nXl4fb_LbcI

(appendix on next page)

APPENDIX

Figure 2: Push Button Circuit (Our project contained four such circuits in total)
Note: Each button corresponds to one checkers piece. Pressing it signals to the Arduino what piece is being called.

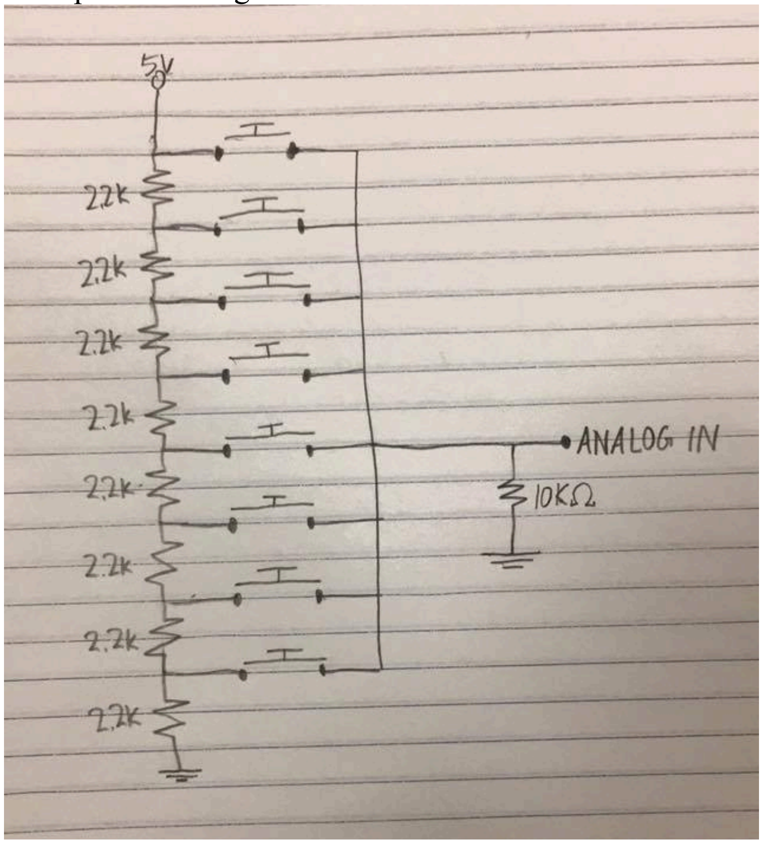

Figure 3: RGB LED Circuit

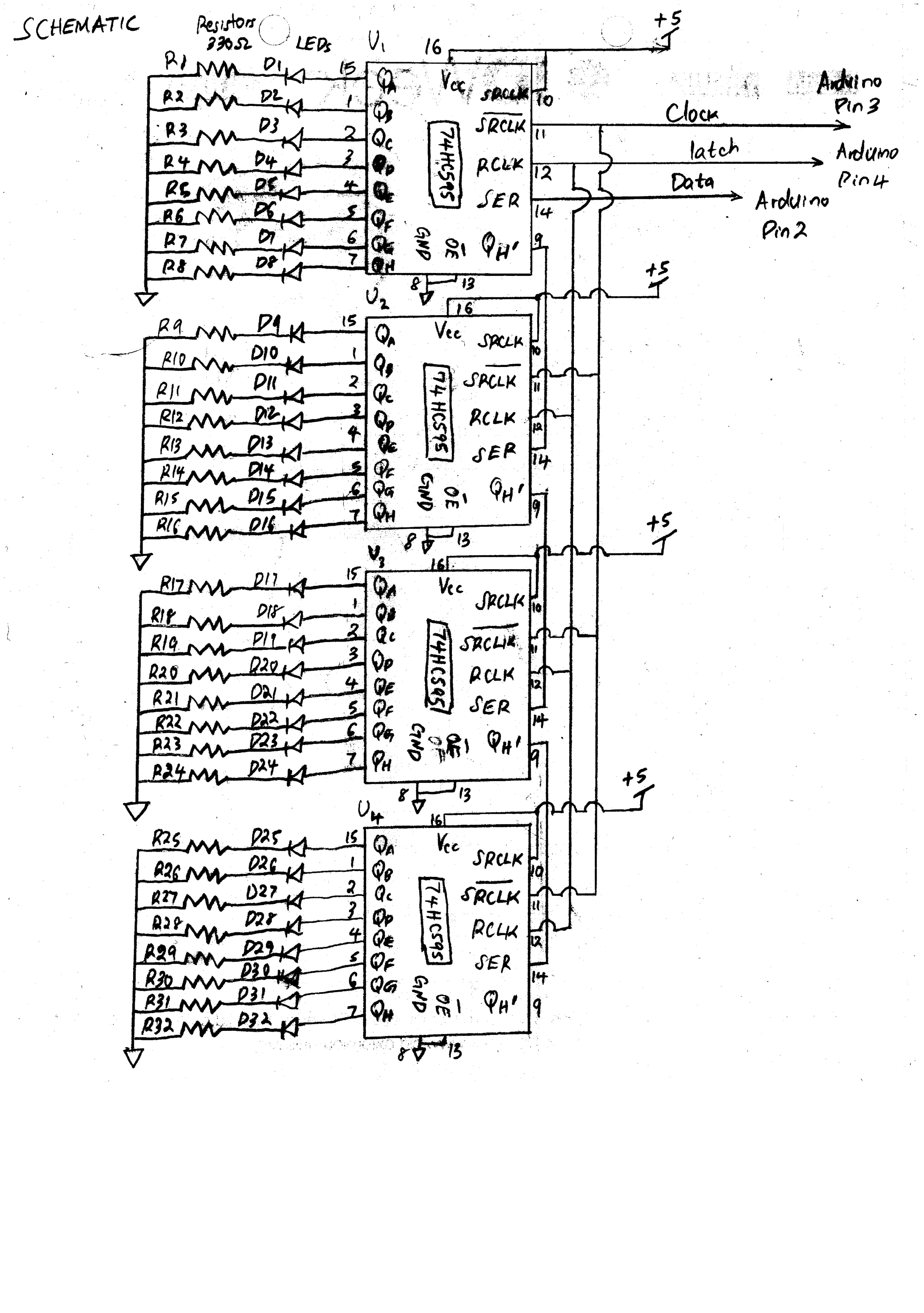

Figure 4: Diagram of physical wiring for one square above and below finished checkerboard product

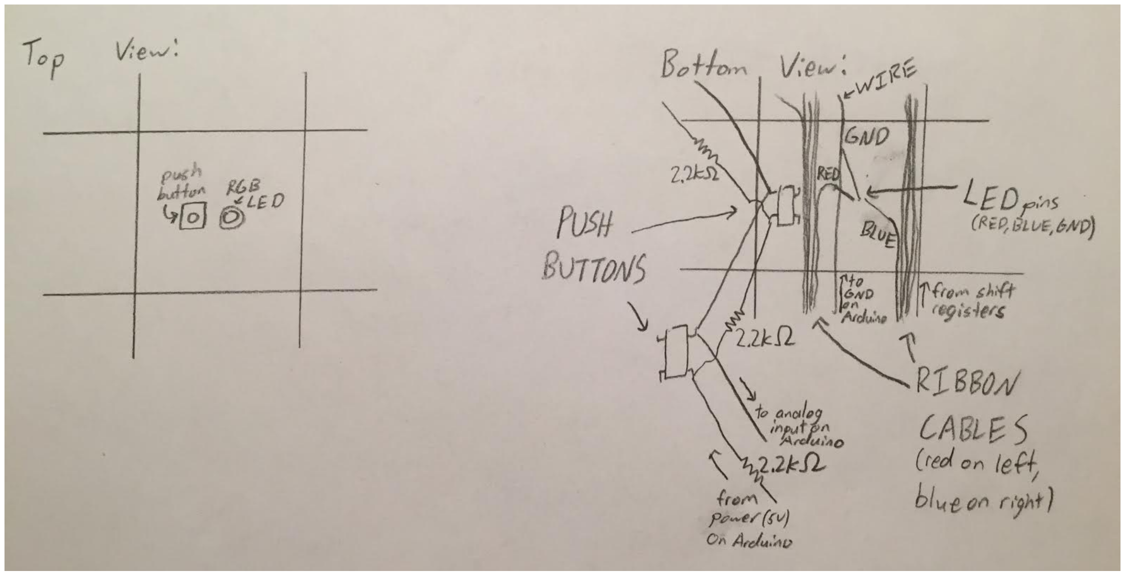
Top View:
push button
RGB LED
Bottom View:
WIRE
GND
RED
BLUE
2.2kΩ
LED pins (RED, BLUE, GND)
PUSH BUTTONS
to GND on Arduino
from shift registers
2.2kΩ
RIBBON CABLES (red on left, blue on right)
to analog input on Arduino
2.2kΩ
from power (5V) on Arduino

Figure 5: Code for the checker's game

```
//Thresholds to distinguish input from buttons
int threshold1 = 850;
int threshold2 = 600;
int threshold3 = 520;
int threshold4 = 400;
int threshold5 = 310;
int threshold6 = 250;
int threshold7 = 150;
int threshold8 = 70;

//Setup for pins from shift registers
int latchPin = 4;
int clockPin = 3;
int clockPin2 = 6;
int dataPin = 2;
int dataPin2 = 5;
//Variables for sending bits to shift registers
unsigned long numberToDisplay=0;
unsigned long numberToDisplay2=0;

//Variables to distinguish bits sent to red and blue shift registers
unsigned long blueSum;
unsigned long redSum;

//Decimal values that correspond with the LED that lights when the value is shifted out.
unsigned long led1 = 1;
unsigned long led2 = 2;
unsigned long led3 = 4;
unsigned long led4 = 8;
unsigned long led5 = 16;
unsigned long led6 = 32;
unsigned long led7 = 64;
unsigned long led8 = 128;
unsigned long led9 = 256;
unsigned long led10 = 512;
unsigned long led11 = 1024;
unsigned long led12 = 2048;
unsigned long led13 = 4096;
unsigned long led14 = 8192;
unsigned long led15 = 16384;
unsigned long led16 = 32768;
unsigned long led17 = 65536;
unsigned long led18 = 131072;
unsigned long led19 = 262144;
unsigned long led20 = 524288;
unsigned long led21 = 1048576;
unsigned long led22 = 2097152;
unsigned long led23 = 4194304;
unsigned long led24 = 8388608;
unsigned long led25 = 16777216;
unsigned long led26 = 33554432;
unsigned long led27 = 67108864;
unsigned long led28 = 134217728;
unsigned long led29 = 268435456;
unsigned long led30 = 536870912;
unsigned long led31 = 1073741824;
unsigned long led32 = 2147483648;



//Array of LED's decimal values
unsigned long led[32] = {led1,
led2,
led3,
led4,
led5,
led6,
led7,
led8,
led9,
led10,
led11,
led12,
led13,
led14,
led15,
led16,
led17,
led18,
led19,
led20,
led21,
led22,
led23,
led24,
led25,
led26,
led27,
led28,
led29,
led30,
led31,
led32};

bool red[32];              //boolean array for red
bool blue[32];             //boolean array for blue
int clicked=0;             //integer that keeps track of clicked button
int prev_clicked=0;        //integer that keeps track of previously clicked button
bool button_state[32];     //boolean array that keeps track of whether a button's LED is on(true) or
off(false)
int button_color[32];      //integer array that keeps track of a button's LED color: 0 red, 1 blue and 2
none.
int order=0;               //integer that keeps track of the order of buttons clicked through concatnation
int difference=0;          //integer that keeps track of the difference of values of clicked buttions

//Function that listens for moves from RED LEDs
void mover(int from_to){
if (order== from_to && button_state[clicked-1]== false && button_color[prev_clicked-1] == 0){
    turn_on(button_color[prev_clicked-1], clicked);
    turn_off(button_color[prev_clicked-1], prev_clicked);
    order=0;
  }
}
//Function that listens for moves from BLUE LEDs(reverse direction)
void mover_reverse(int from_to){
if (order== from_to && button_state[clicked-1]== false && button_color[prev_clicked-1] == 1){
    turn_on(button_color[prev_clicked-1], clicked);
    turn_off(button_color[prev_clicked-1], prev_clicked);
    order=0;
  }
}

//Function that listens for jumping moves from RED LEDs
void jumper(int from_to, int jumpee){

 if (order== from_to && button_color[prev_clicked-1]!= button_color[jumpee-1] && button_state[clicked-
1]==false && button_color[prev_clicked-1] == 0){
    turn_on(button_color[prev_clicked-1], clicked);
    turn_off(button_color[prev_clicked-1], prev_clicked);
    turn_off(button_color[jumpee-1], jumpee);
    order=0;
  }

}

//Function that listens or jumping moves from BLUE LEDs
void jumper_reverse(int from_to, int jumpee){

 if (order== from_to && button_color[prev_clicked-1]!= button_color[jumpee-1] && button_state[clicked-
1]==false && button_color[prev_clicked-1] == 1){
    turn_on(button_color[prev_clicked-1], clicked);
    turn_off(button_color[prev_clicked-1], prev_clicked);
    turn_off(button_color[jumpee-1], jumpee);
    order=0;
  }

}
```

```
//Function that turns on an LED when called
void turn_on(int color, int button_number){

unsigned long turn_on_sum=0;
//if button_color blue
if (color== 1){

   for (int i=0; i<32; i++)
   {
   if (button_number== i+1){
    turn_on_sum+=led[i];
    button_state[i]=true;
    button_color[i]=1;
   }
}
blueSum+=turn_on_sum;


}
//if button_color red
else if(color== 0){
 Serial.print("working");
 for (int i=0; i<32; i++)
 {
 if (button_number== i+1){
 turn_on_sum+=led[i];
 button_state[i]=true;
 button_color[i]=0;
 }
 }
 redSum+=turn_on_sum;



}
}

//Function that turns off an LED when called.
void turn_off(int color, int button_number){

unsigned long turn_off_sum=0;
//if button_color blue
if (color== 1){

   for (int i=0; i<32; i++)
   {
   if (button_number== i+1){
    turn_off_sum+=led[i];
    button_state[i]=false;
   }
}
blueSum-=turn_off_sum;


}
//if button_color red
else if(color== 0){
 Serial.print("working");
 for (int i=0; i<32; i++)
 {
 if (button_number== i+1){
 turn_off_sum+=led[i];
 button_state[i]=false;
 }
 }
 redSum-=turn_off_sum;



}
}

//Function that concatenates integers
int concat(int x, int y) {
int temp = y;
while (y != 0) {
   x *= 10;
   y /= 10;
}
return x + temp;
}


void setup() {
 Serial.begin(9600);
 pinMode(latchPin, OUTPUT);
 pinMode(clockPin, OUTPUT);
 pinMode(clockPin2, OUTPUT);
 pinMode(dataPin, OUTPUT);
 pinMode(dataPin2, OUTPUT);

//Board initialization

 for(int i = 0; i < 32; i++) {
//turn on first 12 RED LEDs
   if(i < 12){
     red[i] = true;
     redSum += led[i];
     blue[i] = false;
     button_state[i]=true;
     button_color[i]= 0;
   }
//turn on last 12 BLUE LEDs
   else if(i > 19){
     blue[i] = true;
     blueSum += led[i];
     red[i] = false;
     button_state[i]=true;
     button_color[i]= 1;
   }
//keep the remaining LEDs off
   else {
     red[i] = false;
     blue[i] = false;
     button_state[i]= false;
```

```
    button_color[i]= 2;

  }
 }



}

void loop() {
Serial.print(clicked);
Serial.print("\n");
Serial.print("order is:");
Serial.print(order);




 //Store input from analog pins in integer variables
 int a = analogRead(0);
 int b = analogRead(1);
 int c = analogRead(2);
 int d = analogRead(3);

 //Use thresholds to distinguish input from different buttons
    if(a > threshold1){
      Serial.print("25");
      if (clicked != 25){
     order= concat(clicked,25);
     difference=abs(clicked-25);
     prev_clicked=clicked;
     clicked=25;
     }

    }
    if(a < threshold1 && a > threshold2){
      Serial.print("32");
      if (clicked != 32){
     order= concat(clicked,32);
     difference=abs(clicked-32);
     prev_clicked=clicked;
     clicked=32;
     }

    }
    if(a < threshold2 && a > threshold3){
      Serial.print("26");
      if (clicked != 26){
     order= concat(clicked,26);
     difference=abs(clicked-26);
     prev_clicked=clicked;
     clicked=26;
     }

    }
    if(a < threshold3 && a > threshold4){
      Serial.print("31");
      if (clicked != 31){
```

```
 order= concat(clicked,31);
 difference=abs(clicked-31);
 prev_clicked=clicked;
 clicked=31;
 }

}
if(a < threshold4 && a > threshold5){
  Serial.print("27");
  if (clicked != 27){
 order= concat(clicked,27);
 difference=abs(clicked-27);
 prev_clicked=clicked;
 clicked=27;
 }

}
if(a < threshold5 && a > threshold6){
  Serial.print("30");
  if (clicked != 30){
 order= concat(clicked,30);
 difference=abs(clicked-30);
 prev_clicked=clicked;
 clicked=30;
 }

}
if(a < threshold6 && a > threshold7){
  Serial.print("28");
  if (clicked != 28){
 order= concat(clicked,28);
 difference=abs(clicked-28);
 prev_clicked=clicked;
 clicked=28;
 }

}
if(a < threshold7 && a > threshold8){
  Serial.print("29");
  if (clicked != 29){
 order= concat(clicked,29);
 difference=abs(clicked-29);
 prev_clicked=clicked;
 clicked=29;
 }
}


if(b > threshold1){
  Serial.print("17");
  if (clicked != 17){
 order= concat(clicked,17);
 difference=abs(clicked-17);
 prev_clicked=clicked;
 clicked=17;
 }

}
```

```
if(b < threshold1 && b > threshold2){
  Serial.print("24");
  if (clicked != 24){
 order= concat(clicked,24);
 difference=abs(clicked-24);
 prev_clicked=clicked;
 clicked=24;
 }

}
if(b < threshold2 && b > threshold3){
  Serial.print("18");
  if (clicked != 18){
 order= concat(clicked,18);
 difference=abs(clicked-18);
 prev_clicked=clicked;
 clicked=18;
 }

}
if(b < threshold3 && b > threshold4){
  Serial.print("23");
  if (clicked != 23){
 order= concat(clicked,23);
 difference=abs(clicked-23);
 prev_clicked=clicked;
 clicked=23;
 }

}
if(b < threshold4 && b > threshold5){
  Serial.print("19");
  if (clicked != 19){
 order= concat(clicked,19);
 difference=abs(clicked-19);
 prev_clicked=clicked;
 clicked=19;
 }

}
if(b < threshold5 && b > threshold6){
  Serial.print("22");
  if (clicked != 22){
 order= concat(clicked,22);
 difference=abs(clicked-22);
 prev_clicked=clicked;
 clicked=22;
 }

}
if(b < threshold6 && b > threshold7){
  Serial.print("20");
  if (clicked != 20){
 order= concat(clicked,20);
 difference=abs(clicked-20);
 prev_clicked=clicked;
 clicked=20;
 }
```

```
}
if(b < threshold7 && b > threshold8){
  Serial.print("21");
  if (clicked != 21){
 order= concat(clicked,21);
 difference=abs(clicked-21);
 prev_clicked=clicked;
 clicked=21;
 }

}

if(c > threshold1){
  Serial.print("9");
  if (clicked != 9){
 order= concat(clicked,9);
 difference=abs(clicked-9);
 prev_clicked=clicked;
 clicked=9;
 }

}
if(c < threshold1 && c > threshold2){
  Serial.print("16");
  if (clicked != 16){
 order= concat(clicked,16);
 difference=abs(clicked-16);
 prev_clicked=clicked;
 clicked=16;
 }

}
if(c < threshold2 && c > threshold3){
  Serial.print("10");
  if (clicked != 10){
 order= concat(clicked,10);
 difference=abs(clicked-10);
 prev_clicked=clicked;
 clicked=10;
 }

}
if(c < threshold3 && c > threshold4){
  Serial.print("15");
  if (clicked != 15){
 order= concat(clicked,15);
 difference=abs(clicked-15);
 prev_clicked=clicked;
 clicked=15;
 }

}
if(c < threshold4 && c > threshold5){
  Serial.print("11");
  if (clicked != 11){
 order= concat(clicked,11);
 difference=abs(clicked-11);
 prev_clicked=clicked;
 clicked=11;
 }

}
if(c < threshold5 && c > threshold6){
  Serial.print("14");
  if (clicked != 14){
 order= concat(clicked,14);
 difference=abs(clicked-14);
 prev_clicked=clicked;
 clicked=14;
 }

}
if(c < threshold6 && c > threshold7){
  Serial.print("12");
  if (clicked != 12){
 order= concat(clicked,12);
 difference=abs(clicked-12);
 prev_clicked=clicked;
 clicked=12;
 }

}
if(c < threshold7 && c > threshold8){
  Serial.print("13");
  if (clicked != 13){
 order= concat(clicked,13);
 difference=abs(clicked-13);
 prev_clicked=clicked;
 clicked=13;
 }

}

if(d > threshold1){
 Serial.print("1");

 if (clicked != 1){
 order= concat(clicked,1);
 difference=abs(clicked-1);
 prev_clicked=clicked;
 clicked=1;
 }
  }

if(d < threshold1 && d > threshold2){
  Serial.print("8");
 if (clicked != 8){
 order= concat(clicked,8);
 difference=abs(clicked-8);
 prev_clicked=clicked;
 clicked=8;
 }

}
if(d < threshold2 && d > threshold3){
  Serial.print("2");
  if (clicked != 2){
 order= concat(clicked,2);
 difference=abs(clicked-2);
 prev_clicked=clicked;
 clicked=2;
 }

}
if(d < threshold3 && d > threshold4){
  Serial.print("7");
  if (clicked != 7){
 order= concat(clicked,7);
 difference=abs(clicked-7);
 prev_clicked=clicked;
 clicked=7;
 }

}
if(d < threshold4 && d > threshold5){
  Serial.print("3");
  if (clicked != 3){
 order= concat(clicked,3);
 difference=abs(clicked-3);
 prev_clicked=clicked;
 clicked=3;
 }

}
if(d < threshold5 && d > threshold6){
  Serial.print("6");
  if (clicked != 6){
 order= concat(clicked,6);
 difference=abs(clicked-6);
 prev_clicked=clicked;
 clicked=6;

 }
}
if(d < threshold6 && d > threshold7){
  Serial.print("4");
  if (clicked != 4){
 order= concat(clicked,4);
 difference=abs(clicked-4);
 prev_clicked=clicked;
 clicked=4;
 }

}
if(d < threshold7 && d > threshold8){
  Serial.print("5");
  if (clicked != 5){
 order= concat(clicked,5);
 difference=abs(clicked-5);
 prev_clicked=clicked;
 clicked=5;
 }
```

```
   }

  numberToDisplay = redSum ;// Shift out RED sum
  numberToDisplay2 = blueSum;//Shift ot BLUE sum

  digitalWrite(latchPin, LOW);

  shiftOut(dataPin2, clockPin2, MSBFIRST, (numberToDisplay2>>24));  // most significant bit (MSB) goes
out first;
  shiftOut(dataPin2, clockPin2, MSBFIRST, (numberToDisplay2>>16));
  shiftOut(dataPin2, clockPin2, MSBFIRST, (numberToDisplay2>>8));
  shiftOut(dataPin2, clockPin2, MSBFIRST, numberToDisplay2);  // most significant bit (MSB) goes out
first;

  shiftOut(dataPin, clockPin, MSBFIRST, (numberToDisplay>>24));  // most significant bit (MSB) goes out
first;
  shiftOut(dataPin, clockPin, MSBFIRST, (numberToDisplay>>16));
  shiftOut(dataPin, clockPin, MSBFIRST, (numberToDisplay>>8));
  shiftOut(dataPin, clockPin, MSBFIRST, numberToDisplay);
  //take the latch pin high so the LEDs will light up:
  digitalWrite(latchPin, HIGH);

//Listedn for moves and call the function whose condition(s) is satisfied
  mover(18);
  jumper(110,8);
  mover(28);
  jumper(29,8);
  mover(27);
  jumper(211,7);
  mover(37);
  jumper(310,7);
  mover(36);
  jumper(312,6);
  mover(46);
  jumper(411,6);
  mover(45);
  mover(512);
  jumper(514,12);
  mover(612);
  jumper(613,12);
  mover(611);
  jumper(615,11);
  mover(711);
  jumper(714,11);
  mover(710);
  jumper(716,10);
  mover(810);
  jumper(815, 10);
  mover(89);
  mover(916);
  jumper(918,16);
  mover(1016);
  jumper(1017,16);
  mover(1015);
  jumper(1019,15);
  mover(1115);
  jumper(1118,15);
  mover(1114);
  jumper(1120,14);
  mover(1214);
  jumper(1219,14);
  mover(1213);
  mover(1320);
  jumper(1322,20);
  mover(1420);
  jumper(1421,20);
  mover(1419);
  jumper(1423,19);
  mover(1519);
  jumper(1522,19);
  mover(1518);
  jumper(1524,18);
  mover(1618);
  jumper(1623,18);
  mover(1617);
  mover(1724);
  jumper(1726,24);
  mover(1824);
  jumper(1825,24);
  mover(1823);
  jumper(1827,23);
  mover(1923);
  jumper(1926,23);
  mover(1922);
  jumper(1928,22);
  mover(2022);
  jumper(2027,22);
  mover(2021);
  mover(2128);
  jumper(2130,28);
  mover(2228);
  jumper(2229,28);
  mover(2227);
  jumper(2231,27);
  mover(2327);
  jumper(2330,27);
  mover(2326);
  jumper(2332,26);
  mover(2426);
  jumper(2431,26);
  mover(2425);
  mover(2532);
  mover(2632);
  mover(2631);
  mover(2731);
  mover(2730);
  mover(2830);
  mover(2829);
  mover_reverse(3225);
  mover_reverse(3226);
  jumper_reverse(3223,26);
  mover_reverse(3126);
  jumper_reverse(3124,26);
  mover_reverse(3127);
  jumper_reverse(3122,27);
  mover_reverse(3027);
  jumper_reverse(3023,27);
  mover_reverse(3028);
  jumper_reverse(3021,28);
  mover_reverse(2928);
  jumper_reverse(2922,28);
  mover_reverse(2922);
  jumper_reverse(2819,22);
  mover_reverse(2821);
  mover_reverse(2722);
  jumper_reverse(2720,22);
  mover_reverse(2723);
  jumper_reverse(2718,23);
  mover_reverse(2623);
  jumper_reverse(2619,23);
  mover_reverse(2624);
  jumper_reverse(2617,24);
  mover_reverse(2524);
  jumper_reverse(2518,24);
  mover_reverse(2417);
  mover_reverse(2418);
  jumper_reverse(2415,18);
  mover_reverse(2318);
  jumper_reverse(2316,18);
  mover_reverse(2319);
  jumper_reverse(2314,19);
  mover_reverse(2219);
  jumper_reverse(2215,19);
  mover_reverse(2220);
  jumper_reverse(2213,20);
  mover_reverse(2120);
  jumper_reverse(2114,20);
  mover_reverse(2013);
  mover_reverse(2014);
  jumper_reverse(2011,14);
  mover_reverse(1914);
  jumper_reverse(1912,14);
  mover_reverse(1915);
  jumper_reverse(1910,15);
  mover_reverse(1815);
  jumper_reverse(1811,15);
  mover_reverse(1816);
  jumper_reverse(189,16);
  mover_reverse(1716);
  jumper_reverse(1710,16);
  mover_reverse(169);
  mover_reverse(1610);
  jumper_reverse(167,10);
  mover_reverse(1510);
  jumper_reverse(158,10);
  mover_reverse(1511);
  jumper_reverse(156,11);
  mover_reverse(1411);
  jumper_reverse(147,11);
  mover_reverse(1412);
  jumper_reverse(145,12);
  mover_reverse(1312);
  jumper_reverse(136,12);
  mover_reverse(125);
  mover_reverse(126);
  jumper_reverse(123,6);
  mover_reverse(116);
  jumper_reverse(114,6);
  mover_reverse(117);
  jumper_reverse(112,7);
  mover_reverse(107);
  jumper_reverse(103,7);
  mover_reverse(108);
  jumper_reverse(101,8);
  mover_reverse(98);
  jumper_reverse(92,8);
  mover_reverse(81);
  mover_reverse(82);
  mover_reverse(72);
  mover_reverse(73);
  mover_reverse(63);
  mover_reverse(64);
  mover_reverse(54);





  // pause before next value:
  delay(20);

 Serial.print("\n");
}
```