The Maze

Tyler Strach - tstra6 - 660866964
Manav Kohli - mkohli4 - 660950121

Abstract

An arcade-style game with a base premise of a maze, with milestones that occur after the user comes into contact with a game object. These milestones will take advantage of Arduino materials at our disposal. The main output for the game is the 16x32 LED Matrix. This will include the score, main game, and minigames. The two Arduinos will be connected to one another via wiring, with one arduino focusing solely on the LED Matrix, and the other one focusing on all other materials included. This is because the LED Matrix requires a large number of wires to function.

Project Description

1.) Our project is a style of ***arcade game***, with different inputs and the main output being a screen. We want to make our game difficult like an arcade game, along with score-tracking capabilities. The game consists of a circle working through a generated maze. The main objective of the game will be to reach the end of the maze before the time limit runs out. There will be different tiles throughout the game the user can land on which will trigger bonus minigames to provide additional challenges to the player. These challenges will utilize other input devices and test the user's reaction time and skills.

2.) We are a group of 2, so there will be a total of ***2 Arduino UNOs*** used to fully implement the project.

The first UNO will be used to control the main output screen. The main screen will be in charge of showcasing the environment of the game as the generated maze. When the user reaches the mentioned milestones that trigger bonus games, the screen will switch and adjust to display the mini-games.

The second UNO will be used to control all of the user inputs. This will control the logic of the game and how the player and environment react to the user's inputs. Another piece is the unique minigames, which the Arduino will also have to handle the majority of the inputs. This arduino is also in charge of the buzzer and LCD display, which displays the score and messages to the user.

3.) Both UNOs will communicate with each other ***via serial communication*** acting smoothly as 1 game. The second one sending the input information to the first one will be the main path the information will travel. This is because the user inputs will be connected to the second UNO, and the first UNO is dependent on the user inputs to display how the environment changes. The main communication from the first UNO to the second UNO will be when the minigame or endgame is reached and will need to send a signal to the second arduino.

4.) Inputs and Outputs
- Inputs:
  - Pushbuttons x 4 - Used to control the user
- Output:
  - 16 x 32 LED Matrix - Used as the sole display (displays high-score, the actual game, and everything else that outputs information to the player)
  - Buzzer - Used to output sounds when wall is hit
  - LCD - Used to display score throughout the game, outcome of minigame, as well as Final Score.

5.) The original work is centered around the game being played on a large ***16x32 output screen***, as well as the additional challenge of ***minigames*** in special tiles. We are aware that there are a lot of games out there with a similar concept, but a key difference is the game is more dynamic than a normal arcade game. Ours has minigames that are meant to test the player's reaction time and other skills along with their skill to play the game. Our display is not conventionally used to display a dynamic game environment, it is commonly used as a sign to display messages or static content.

6.) Build Guide

a.) Arduino 1 - RGB Matrix (see first hardware link for reference)

i.) Connect pins R1, G1, and B1 to Arduino pins 2, 3, and 4.

ii.) Connect pins R2, G2, and B2 to Arduino pins 5,6, and 7.

iii.) Connect pins A, B, C, and D to Arduino pins A0, A1, A2, and A3.

iv.) The LAT signal connects to Arduino pin 10.

v.) OE connects to Arduino pin 9.

vi.) CLK connects to Arduino pin 8.

b.) Arduino 2 - Handling Input and other Outputs (see second hardware link for reference)

i.) 16x2 LCD Display Wiring
- VSS, RW, K wire to the ground rail - *ground for power from 5V*
- VCC wire to 5V rail - *powers the display*
- VO to Potientiometer VO - *regulates display for clear output*
- Data Pins - *control the data and R/W to use display*
  - RS to Uno D07
  - E to Uno D06
  - D4, D5, D6 to Uno D02, D03, D04 (in that order)
  - D7 to Uno D05
- A to 5V rail (220 Ohm Resistor in between) - *additional power*

ii.) Potentiometer Wiring
- VIN to 5V rail - *powers the potentiometer from 5V*
- VO to 16x2 VO - *Sends value to regulate display for clear output*
- VGND to ground rail - *grounds for the 5V power*

iii.) Buzzer
- The negative side to the ground rail
- The positive side to UNO pin 8

iv.) Buttons (x4)
- Positive to 5V Rail
- Wire to 10k Resistor to UNO (13-10)

7.) User Guide
a.) First, compile the code on each machine without connecting to one another. This is because the code will not compile after a serial connection is established.
b.) After the code is compiled, connect the 0 of one Arduino to the 1 of the other, and vice versa.
c.) Connect the Ground pins of the Arduinos (ensure the ground pins are the same on each board).
d.) Next, use the buttons to navigate through the maze. As soon as the right (->) button is pressed, the score will begin counting down.
e.) You now have two options.
i.) You can navigate your way through the maze and make it to the end, with your final score being <1000
ii.) You can attempt the three minigames scattered around the maze and increase your score by +250 points every time you win the minigame
(1) Once you enter the minigame, the screen will be red. Your objective is to press any one of the buttons as soon as the screen flashes green.
(2) If you do this under two seconds, you will get 250 points and be redirected to your last position in the maze.
(3) If not, you will not gain any points and will be redirected to your last position in the maze.

Supporting Materials

1.) Timeline
a.) Research and Gather Hardware Materials - 10/23
b.) Hardware and Wiring Completed - 10/30
c.) Develop Code for Output Display (Creating Game Elements to Display) - 11/06
d.) Develop Code for Inputs (Handling User Inputs and Subsequent Output) - 11/13
e.) Week of Presentation - 11/20
f.) Project Documentation - 11/27

2.) Materials
a.) Arduino x2
b.) Breadboard x2
c.) Pushbutton x4
d.) Passive Buzzer x1
e.) 16x32 RGB LED Matrix x1
f.) 10k Ohm Resistor x4
g.) Potentiometer x1
h.) 16x2 LCD Screen x1
i.) 1k Ohm Resistor x1
j.) Wires

3.) References

a.) Hardware/Wiring

i.) https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/connecting-with-jumper-wires

ii.) https://www.circuito.io/app?components=97,97,97,97,512,11021,341099,956215

b.) Code Documentation and Inspiration

i.) https://www.instructables.com/I2C-between-Arduinos/

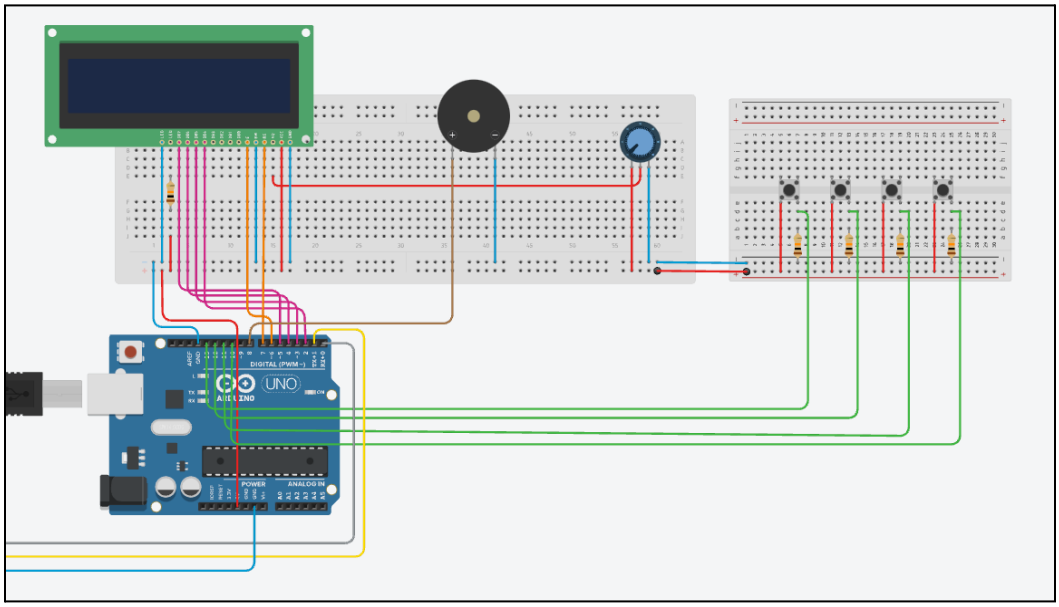
ii.) https://www.instructables.com/Arduino-Snake-Game-16x32-RGB-LED-Matrix/

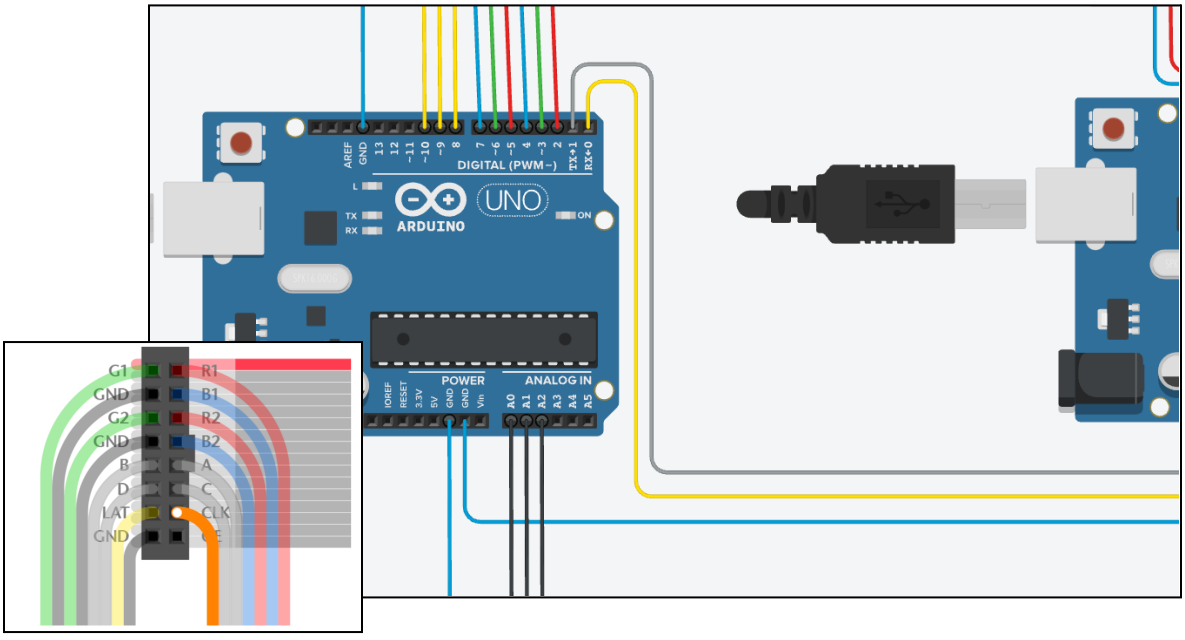
iii.) https://www.arduino.cc/reference/en/libraries/rgb-matrix-panel/

iv.) https://www.arduino.cc/reference/en/libraries/buzzer/

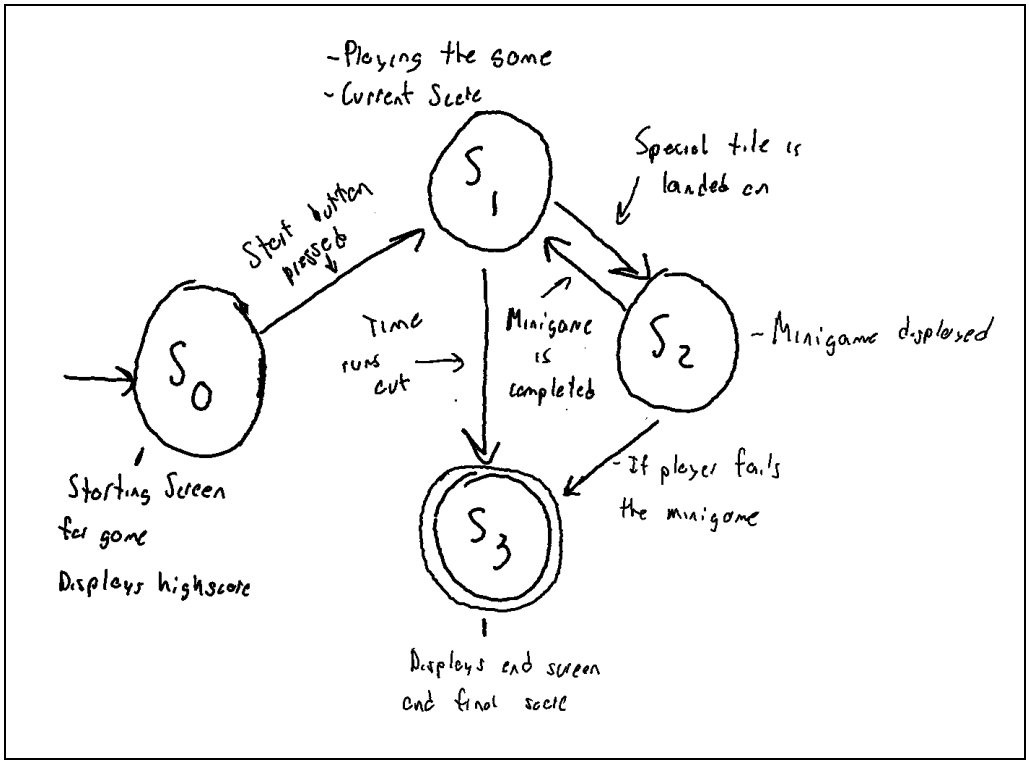
4.) Diagrams

a.) Hardware Diagrams (Arduino 2 and then Arduino 1)





b.) State Diagram



Code

Arduino 1 - Controlling Matrix Display

```
/*
 Tyler Strach - tstra6
 Manav Kolhi - mkohli4

 Group 57
 The Maze

 Arduino 1 - Controlling Matrix Display


 An arcade style game with a base premise of a maze, with milestones that occur after the user comes into contact with a game object.
 These milestones will take advantage of Arduino materials at our disposal. The main output for the game is the 16x32 LED Matrix.
 This will include the score, main game, and minigames. The two Arduinos will be connected to one another via wiring, with one arduino focusing solely on the LED Matrix,
 and the other one focusing on all other materials included. This is because the LED Matrix requires a large number of wires to function.
*/

#include <RGBmatrixPanel.h> // for the output panel
#include <string.h> // for string manipulation
#include <time.h> // for random number


#define CLK  8   // USE THIS ON ARDUINO UNO, ADAFRUIT METRO M0, etc.
//#define CLK A4 // USE THIS ON METRO M4 (not M0)
//#define CLK 11 // USE THIS ON ARDUINO MEGA
#define OE   9
#define LAT 10
#define A   A0
#define B   A1
#define C   A2

#define RED matrix.Color333(7,0,0)
#define GREEN matrix.Color333(0,7,0)
#define BLUE matrix.Color333(0,0,7)
#define BLACK matrix.Color333(0,0,0)


char start[2] = "s";
char win[2] = "w";
```

```
char buzz[2] = "b";

RGBmatrixPanel matrix(A, B, C, CLK, LAT, OE, false);

String userInput = "";
int prevX = 0;
int userX = 0;
int prevY = 7;
int userY = 7;

// represents the maze - 0=blank tile, 1=wall, 2=minigame, 3=end goal
char maze[15][31] = {
   {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
   {1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1},
   {1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1},
   {1, 2, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 2, 1, 0, 1, 0, 1, 0, 1},
   {1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1},
   {1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1},
   {1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1},
   {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 3},
   {1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1},
   {1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
   {1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},
   {1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1},
   {1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1},
   {1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 1},
   {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
};

void drawBoard(){
 //draw border
 matrix.drawLine(0, 0, 0, 14, BLUE);
 matrix.drawLine(30, 0, 30, 14, BLUE);
 matrix.drawLine(0, 0, 30, 0, BLUE);
 matrix.drawLine(0, 14, 30, 14, BLUE);
 delay(250);

 // draw maze (even only)

 // horizontal lines
 matrix.drawLine(0, 4, 2, 4, BLUE);

 matrix.drawLine(2, 8, 2, 14, BLUE);
 matrix.drawLine(2, 4, 2, 2, BLUE);
 delay(50);

 matrix.drawLine(4, 0, 4, 2, BLUE);
```

```
matrix.drawLine(4, 4, 4, 12, BLUE);
delay(50);


matrix.drawLine(6, 0, 6, 4, BLUE);
matrix.drawLine(6, 10, 6, 14, BLUE);
delay(50);

matrix.drawLine(8, 8, 8, 10, BLUE);

matrix.drawLine(10, 2, 10, 4, BLUE);
matrix.drawLine(10, 10, 10, 12, BLUE);
delay(50);

matrix.drawLine(12, 0, 12, 4, BLUE);
matrix.drawLine(12, 8, 12, 10, BLUE);
delay(50);

matrix.drawLine(14, 4, 14, 12, BLUE);
delay(50);

matrix.drawLine(16, 2, 16, 4, BLUE);
matrix.drawLine(16, 6, 16, 10, BLUE);
delay(50);

matrix.drawLine(18, 2, 18, 6, BLUE);
matrix.drawLine(18, 10, 18, 14, BLUE);
delay(50);

matrix.drawLine(20, 0, 20, 2, BLUE);
matrix.drawLine(20, 4, 20, 12, BLUE);
delay(50);

matrix.drawLine(22, 2, 22, 4, BLUE);
matrix.drawLine(22, 6, 22, 8, BLUE);
matrix.drawLine(22, 10, 22, 14, BLUE);
delay(50);

matrix.drawLine(24, 2, 24, 8, BLUE);
delay(50);

matrix.drawLine(26, 2, 26, 6, BLUE);
delay(50);

matrix.drawLine(28, 0, 28, 4, BLUE);
delay(50);
```

```
//vertical lines
matrix.drawLine(8, 2, 10, 2, BLUE);
matrix.drawLine(14, 2, 16, 2, BLUE);
matrix.drawLine(18, 2, 20, 2, BLUE);
matrix.drawLine(22, 2, 24, 2, BLUE);
delay(50);

matrix.drawLine(0, 4, 2, 4, BLUE);
matrix.drawLine(4, 4, 8, 4, BLUE);
matrix.drawLine(10, 4, 12, 4, BLUE);
matrix.drawLine(16, 4, 18, 4, BLUE);
matrix.drawLine(20, 4, 22, 4, BLUE);
delay(50);

matrix.drawLine(2, 6, 4, 6, BLUE);
matrix.drawLine(6, 6, 16, 6, BLUE);
matrix.drawLine(22, 6, 24, 6, BLUE);
matrix.drawLine(26, 6, 30, 6, BLUE);
delay(50);

matrix.drawLine(4, 8, 6, 8, BLUE);
matrix.drawLine(8, 8, 12, 8, BLUE);
matrix.drawLine(16, 8, 20, 8, BLUE);
matrix.drawLine(24, 8, 30, 8, BLUE);
delay(50);

matrix.drawLine(6, 10, 8, 10, BLUE);
matrix.drawLine(22, 10, 28, 10, BLUE);
delay(50);

matrix.drawLine(8, 12, 16, 12, BLUE);
matrix.drawLine(24, 12, 30, 12, BLUE);
delay(50);

//draw minigame tiles
matrix.drawPixel(23, 3, matrix.Color333(7, 0, 0));
matrix.drawPixel(29, 13, matrix.Color333(7, 0, 0));
matrix.drawPixel(1, 3, matrix.Color333(7, 0, 0));

//draw player
matrix.drawPixel(0, 7, BLACK);
matrix.drawPixel(userX, userY, GREEN);

//draw end goal
matrix.drawPixel(30, 7, matrix.Color333(7, 7, 0));

delay(250);
```

```
}



// Receive user input from the other arduino u, d, l, r and adjust player pixel
void getUserInput(){
 if(Serial.available() > 0){
   String move = Serial.readStringUntil('\n');
   if(move == "u"){
     if( ((userY - 1) < 0) || (maze[userY-1][userX] == 1) ){
       Serial.write(buzz, 2);
       return;
     }
     else{
       userY = prevY-1;
     }
   }
   else if(move == "d"){
     if( ((userY + 1) > 14) || (maze[userY+1][userX] == 1) ){
       Serial.write(buzz, 2);
       return;
     }
     else{
       userY = prevY+1;
     }
   }
   else if(move == "l"){
     if( ((userX - 1) < 0) || (maze[userY][userX-1] == 1) ){
       Serial.write(buzz, 2);
       return;
     }
     else{
       userX = prevX-1;
     }
   }
   else if(move == "r"){
     if( ((userX + 1) > 31) || (maze[userY][userX+1] == 1) ){
       Serial.write(buzz, 2);
       return;
     }
     else{
       userX = prevX+1;
     }
   }
 }
}

// where all the code lies, contains a while loop to simulate loop()
```

```
void setup() {
 matrix.begin();
 drawBoard();

 // setting the wire to listen for data from the other arduino
 Serial.begin(9600);

 // replicating the loop() function because matrix cannot be written to inside loop
 while(true){
   // check for the user input
   getUserInput();

   // checking for minigame
   if(maze[userY][userX] == 2){
     maze[userY][userX] = 0;
     // fill the screen with red
     matrix.fillScreen(RED);

     // generate random time from 3-12 seconds
     srand(time(NULL));
     int random_number = (rand() % 10 + 3);

     // delay until 0.5 second before the screen changes, and then send signal
     delay((random_number) * 1000);
     delay(500);
     Serial.write(start, 2); // send 's'
     delay(1000);
     matrix.fillScreen(GREEN); // change the screen for 2 seconds
     delay(500);
     matrix.fillScreen(BLACK); // change the screen for 2 seconds
     delay(5000);
     drawBoard();
     continue;
   }
   else if(maze[userY][userX] == 3){ // handle win state
     Serial.write(win, 2); // send 'w'
     matrix.fillScreen(BLACK);
     matrix.setCursor(6, 5);  // next line
     matrix.setTextColor(GREEN);
     matrix.print("WIN!");
     break;
   }

   //update player location
   matrix.drawPixel(prevX, prevY, matrix.Color333(0, 0, 0));
   matrix.drawPixel(userX, userY, matrix.Color333(0, 7, 0));
   prevX = userX;
```

```
    prevY = userY;
  }
}

void loop() {
  // cannot use loop to update the display
}
```

Arduino 2 - Handling Buttons, LCD, Buzzer

```
/*
 Tyler Strach - tstra6
 Manav Kolhi - mkohli4

 Group 57
 The Maze

 Arduino 2 - Handling User I/O

 An arcade style game with a base premise of a maze, with milestones that occur after the user comes into contact with a game object.
 These milestones will take advantage of Arduino materials at our disposal. The main output for the game is the 16x32 LED Matrix.
 This will include the score, main game, and minigames. The two Arduinos will be connected to one another via wiring, with one arduino focusing solely on the LED Matrix,
 and the other one focusing on all other materials included. This is because the LED Matrix requires a large number of wires to function.
*/
#include <Wire.h>
#include <LiquidCrystal.h>
#include <string.h>

#define leftButtonPin 12
#define upButtonPin 11
#define downButtonPin 10
#define rightButtonPin 9
#define buzzerPin 8

LiquidCrystal lcd(7, 6, 2, 3, 4, 5);

//Used for sending to output arduino
char up[2] = "u";
char down[2] = "d";
char left[2] = "l";
char right[2] = "r";
//states depending on whether the game is in minigame or main game state
int state = 1;
int gameStart = 0;
int score = 1000; //current score
//scoring is dependent on time
unsigned long int time = 0;
unsigned long int currMillis = 0;
unsigned long int prevMillis = 0;

//variables for Button States and Debouncing
```

```
int ButtonState1;
int ButtonState2;
int ButtonState3;
int ButtonState4;

int ButtonState1Confirm;
int ButtonState2Confirm;
int ButtonState3Confirm;
int ButtonState4Confirm;


void setup() {
 // put your setup code here, to run once:
 pinMode(leftButtonPin, INPUT);
 pinMode(upButtonPin, INPUT);
 pinMode(downButtonPin, INPUT);
 pinMode(rightButtonPin, INPUT);
 pinMode(buzzerPin, OUTPUT);
 Serial.begin(9600);
}

void loop() {
 // put your main code here, to run repeatedly:
  if (state == 1){
   ButtonState1 = digitalRead(leftButtonPin);
   ButtonState2 = digitalRead(upButtonPin);
   ButtonState3 = digitalRead(downButtonPin);
   ButtonState4 = digitalRead(rightButtonPin);
   delay(100);
   ButtonState1Confirm = digitalRead(leftButtonPin);
   ButtonState2Confirm = digitalRead(upButtonPin);
   ButtonState3Confirm = digitalRead(downButtonPin);
   ButtonState4Confirm = digitalRead(rightButtonPin);

   // handle user inputs and send to other arduino
   if(digitalRead(leftButtonPin) == HIGH){
     if (ButtonState1 == ButtonState1Confirm){
       Serial.write(left,2);
       gameStart = 1;
       delay(250);
     }
   }
   if(digitalRead(rightButtonPin) == HIGH){
     if (ButtonState2 == ButtonState2Confirm){
       Serial.write(right,2);
       gameStart = 1;
       delay(250);
```

```
    }
  }
  if(digitalRead(upButtonPin) == HIGH){
    if (ButtonState3 == ButtonState3Confirm){
      Serial.write(up,2);
      gameStart = 1;
      delay(250);
    }
  }
  if(digitalRead(downButtonPin) == HIGH){
    if (ButtonState4 == ButtonState4Confirm){
      Serial.write(down,2);
      gameStart = 1;
      delay(250);
    }
  }

  //add score in the form of time
  //update score
  currMillis = millis();
  if(currMillis - prevMillis > 1000 && gameStart == 1){
    prevMillis = currMillis;
    score -= 1;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Score: ");
    lcd.print(score);
  }

  // wait until no button is pressed to continue
  while(ButtonState1 || ButtonState2 || ButtonState3 || ButtonState4){
    ButtonState1 = digitalRead(leftButtonPin);
    ButtonState2 = digitalRead(upButtonPin);
    ButtonState3 = digitalRead(downButtonPin);
    ButtonState4 = digitalRead(rightButtonPin);
  }

  // gather info from other arduino
  if(Serial.available()>0){
    String gameEvent = Serial.readStringUntil('\n');

    if (gameEvent == "s"){
      state = 2;
    }
    else if (gameEvent == "b"){
      //buzzer handling
      tone(buzzerPin, 350);
```

```
      delay(100);
      noTone(buzzerPin);

    }
    else if (gameEvent == "w"){ // print the final score
      currMillis = 0;
      prevMillis = 0;
      int finalScore = score;
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Final Score:");
      lcd.print(finalScore);
      delay(100000);
      return;
    }
  }
}
else if (state == 2){ // minigame state
  //minigame input-handling
  unsigned long int start_timer = 0;
  unsigned long int cur_timer = 0;
  int inTime = 0;
  unsigned long int speed = 0;


  //receive start timer from other arduino
  start_timer = millis();
  cur_timer = millis();
  while(cur_timer - start_timer < 2000){
    ButtonState1 = digitalRead(leftButtonPin);
    ButtonState2 = digitalRead(upButtonPin);
    ButtonState3 = digitalRead(downButtonPin);
    ButtonState4 = digitalRead(rightButtonPin);
    if(ButtonState1 || ButtonState2 || ButtonState3 || ButtonState4){
      inTime = 1;
      speed = cur_timer - start_timer;
    }
    cur_timer = millis();
  }
  //handling for if player wins/loses minigame
  if ( inTime ){
    score += 250;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("WIN! +250  ");
    lcd.print(speed);
    lcd.print("ms");
```

```
    delay(4000);
  }
  else{
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("LOSE! +0  ");
    lcd.print(speed);
    lcd.print("ms");
    delay(4000);
  }

  state = 1;
}
}
```