CS/EE 120B Custom Laboratory Project

Mini - Tetris

Kiyomi Sugita

3/16/2022

Introduction

Mini - Tetris is a one player game, where the player obtains shapes in a random order, and has to place the shapes down to form lines. Any line created will disappear and allow the player to continue. Players will use a joystick to move the dot, and they will be playing on a 8x8 LED display. The game ends when the player stacks too many shapes so that it reaches the top of the LED display. Originally, this Mini tetris was going to have the original tetris shapes, but due to being unable to properly rotate each shape, it was removed and left as one dot.

Complexities

1. Joystick will be used to move the shapes.
2. Buzzer will be used to indicate the start and end of the game
3. Shift registers will be used with the 8x8 led displays to be the game visuals.

Basic Functionality

The 8x8 LED display will be used to display the game. Buttons could be used to represent rotation and movement of the shapes, rather than a joystick.

User Guide

The controls of this game are solely the joystick. Like in normal Tetris, the game will start, and the user must use the joystick to move the object left, or right. The user can also make the object move down faster by pressing down, if they choose to do so. Otherwise the object will move down on its own as time passes. The goal of the game is to never reach the top of the LED display. The game starts with a tune to indicate its start, and ends with a different tune which will keep playing until the game is reset. The game ends when a whole column of the LED display is lit up, indicating that you have reached the top and failed.

Hardware Components

Computing

- Elegoo UNO R3 microcontroller

Inputs

- Joystick

Outputs

- 8x8 LED display

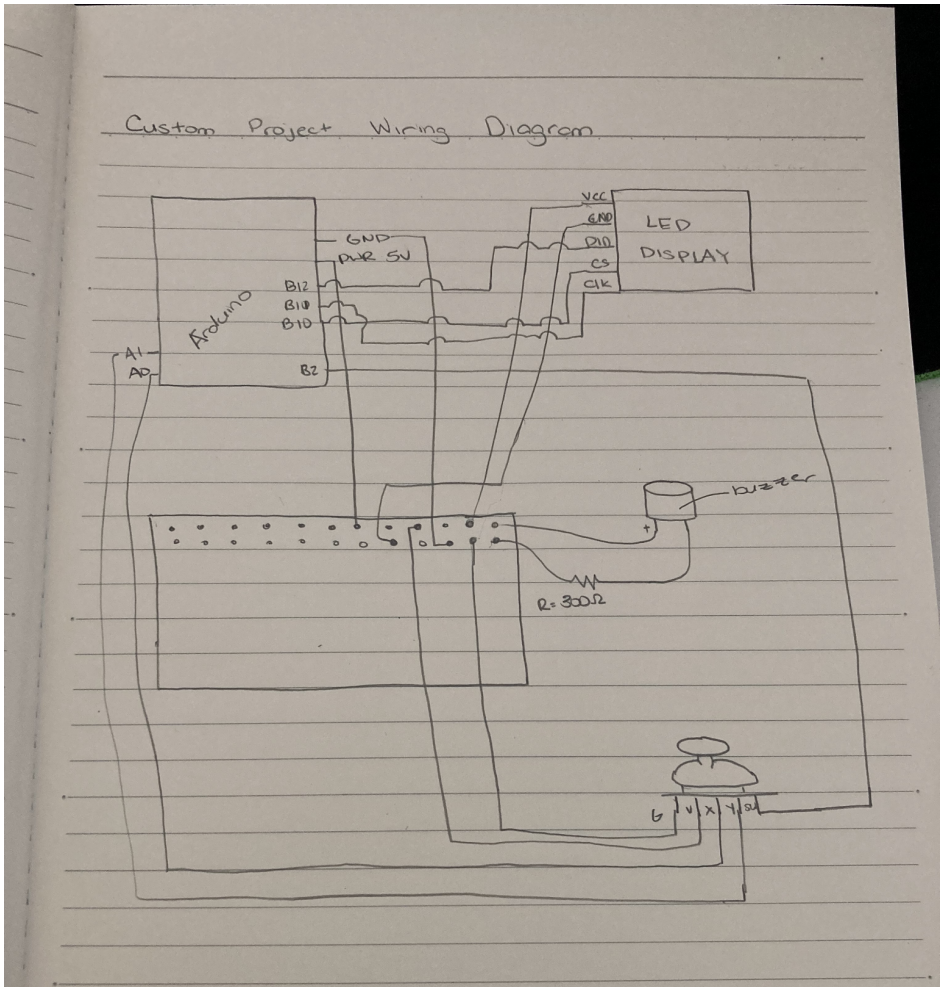
- Buzzer that makes noise at start and end of game

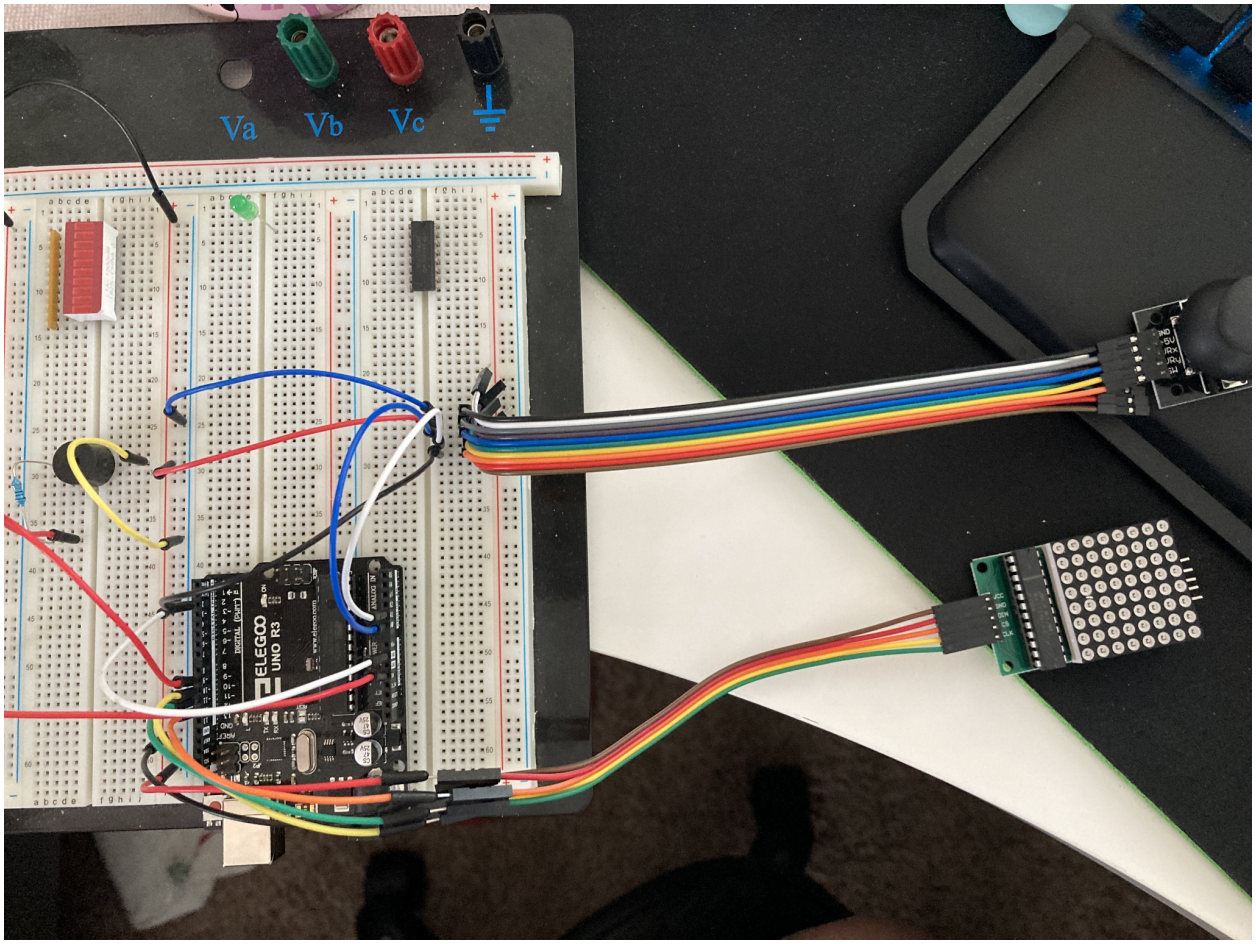
Software Libraries

LedControl.h

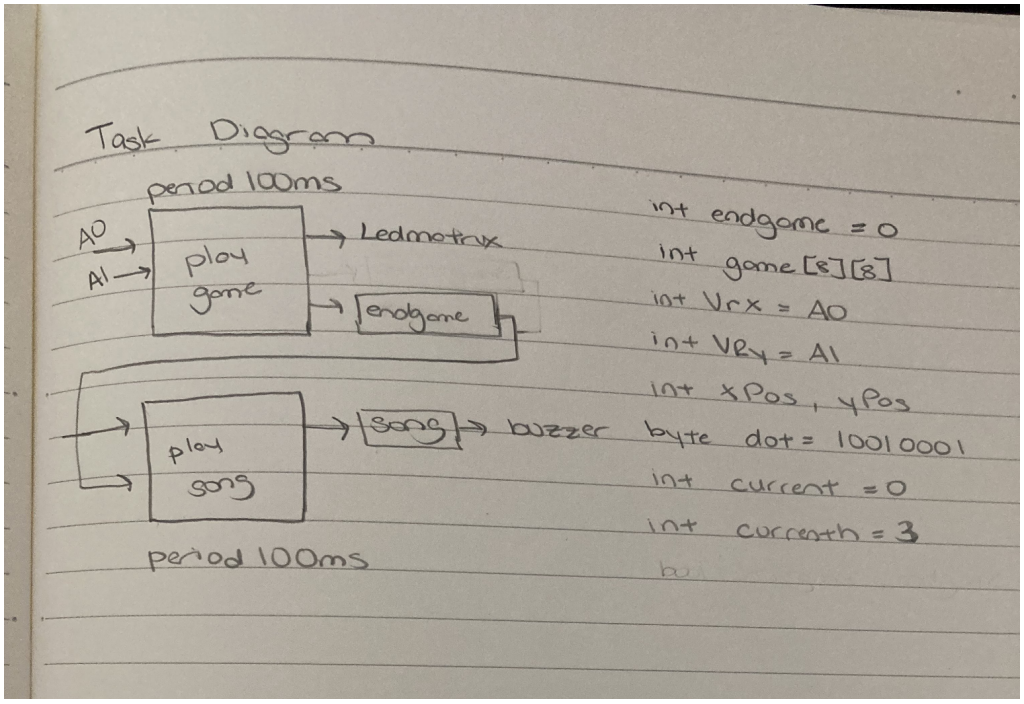
- This library helps control the 8x8 led matrix using the shift register. This allowed for easier access when showing different patterns for the game.

Wiring Diagram

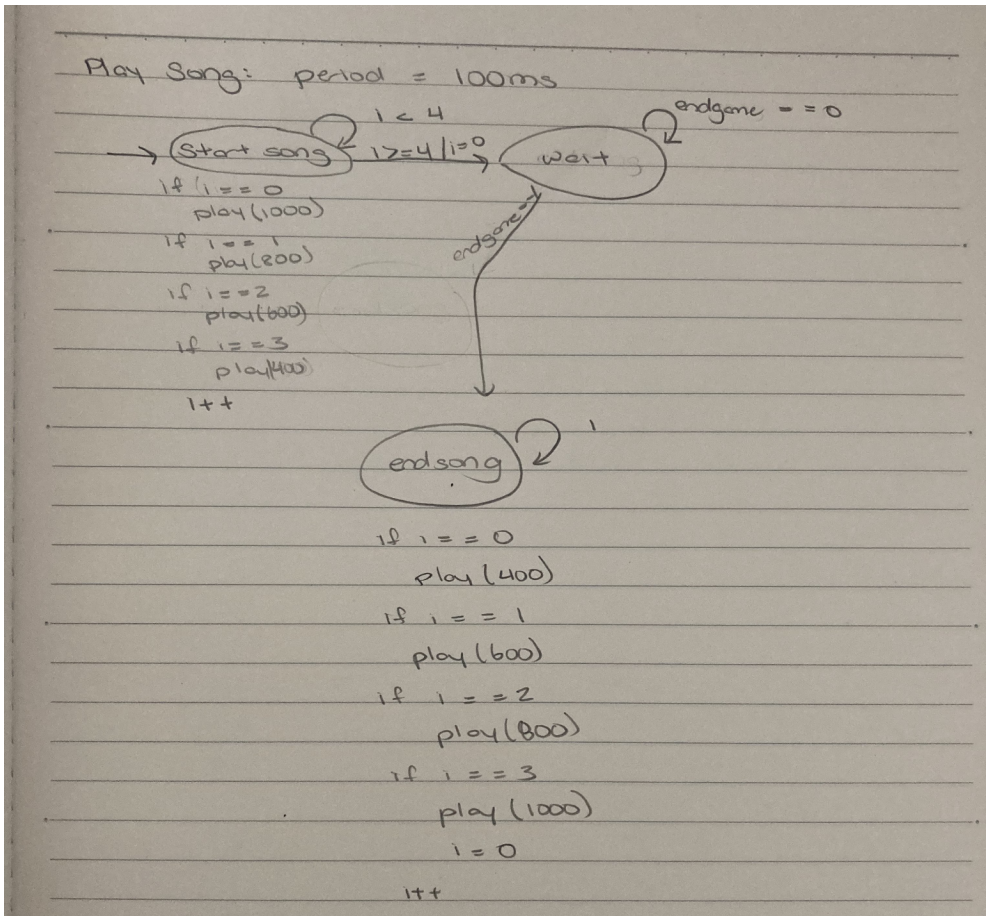


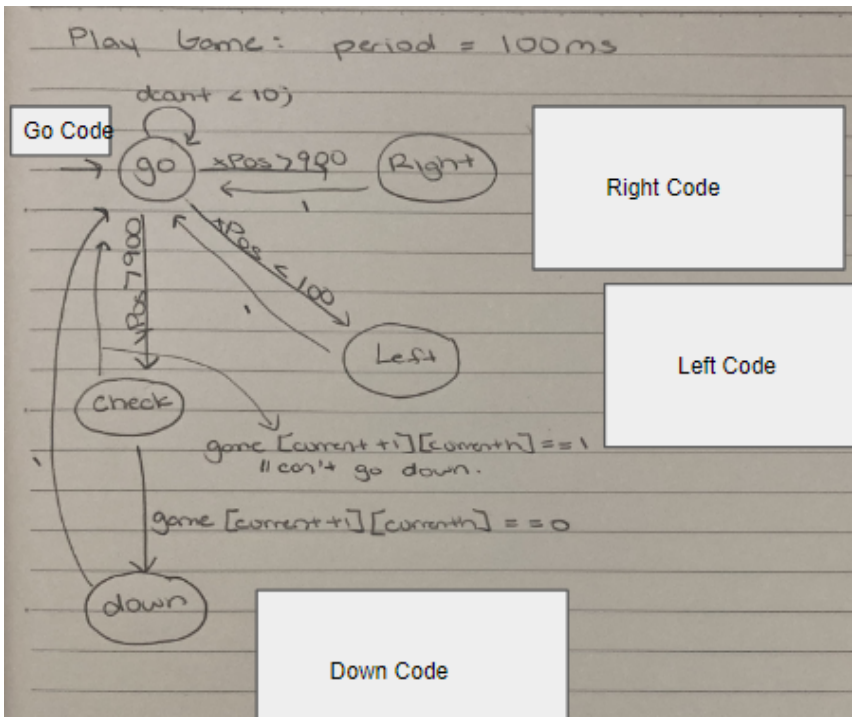


Task Diagram



SynchSM Diagrams





Right Code:

```
if (currenth < 6) {
  currenth++;
  if ((curr == B11000001) && (game[current][currenth] == 0 )) {
    curr = B10100001;
    game[current][currenth] = 1;
    game[current][currenth - 1] = 0;
  } else if ((curr == B10100001) && (game[current][currenth] == 0 )) {
    curr = B10010001;
    game[current][currenth] = 1;
    game[current][currenth - 1] = 0;
  } else if ((curr == B10010001) && (game[current][currenth] == 0 )) {
    curr = B10001001;
    game[current][currenth] = 1;
    game[current][currenth - 1] = 0;
  } else if ((curr == B10001001) && (game[current][currenth] == 0)) {
    curr = B10000101;
    game[current][currenth] = 1;
    game[current][currenth - 1] = 0;
  } else if ((curr == B10000101) && (game[current][currenth] == 0 )) {
    curr = B10000011;
    game[current][currenth] = 1;
    game[current][currenth - 1] = 0;
  }
  a[current] = curr;

}
  lc.setRow(0,0,a[0]);
  lc.setRow(0,1,a[1]);
  lc.setRow(0,2,a[2]);
  lc.setRow(0,3,a[3]);
  lc.setRow(0,4,a[4]);
  lc.setRow(0,5,a[5]);
  lc.setRow(0,6,a[6]);
  lc.setRow(0,7,a[7]);
Serial.print(currenth);
Serial.print('\n');
```

Left Code:

```
  if(currenth > 1) {
    currenth--;
    if ((curr == B10000011) && (game[current][currenth] == 0)) {
      curr = B10000101;
      game[current][currenth] = 1;
      game[current][currenth + 1] = 0;
    } else if ((curr == B10000101) && (game[current][currenth] == 0)) {
      curr = B10001001;
      game[current][currenth] = 1;
      game[current][currenth + 1] = 0;
    } else if ((curr == B10001001) && (game[current][currenth] == 0)) {
      curr = B10010001;
      game[current][currenth] = 1;
      game[current][currenth + 1] = 0;
    } else if ((curr == B10010001) && (game[current][currenth] == 0)) {
      curr = B10100001;
      game[current][currenth] = 1;
      game[current][currenth + 1] = 0;
    } else if ((curr == B10100001) && (game[current][currenth] == 0)) {
      curr = B11000001;
      game[current][currenth] = 1;
      game[current][currenth + 1] = 0;
    }
    a[current] = curr;

  }
    lc.setRow(0,0,a[0]);
    lc.setRow(0,1,a[1]);
    lc.setRow(0,2,a[2]);
    lc.setRow(0,3,a[3]);
    lc.setRow(0,4,a[4]);
    lc.setRow(0,5,a[5]);
    lc.setRow(0,6,a[6]);
    lc.setRow(0,7,a[7]);
  Serial.print(currenth);
  Serial.print('\n');
  printgamestate();
break;
```

Down Code:

```
  current++;
  a[current - 1] = a[current -1] - curr + 129;
  a[current] = curr | a[current];
  game[current][currenth] = 1;
  game[current - 1][currenth] = 0;

  if(game[7][1] == 1 && game[7][2] == 1 && game[7][3] == 1 && game[7][4] == 1 && game[7][5] == 1 && game[7][6] == 1) {
    for(int l = 7; l > 0; l--){
      for(int o = 7; o > 0; o--){
        game[l][o] = game[l-1][o];
      }
      a[l] = a[l-1];
    }
  }
  lc.setRow(0,0,a[0]);
  lc.setRow(0,1,a[1]);
  lc.setRow(0,2,a[2]);
  lc.setRow(0,3,a[3]);
  lc.setRow(0,4,a[4]);
  lc.setRow(0,5,a[5]);
  lc.setRow(0,6,a[6]);
  lc.setRow(0,7,a[7]);
```

Go Code:

```
if((current == botmax) || (current == 0)) {
  curr = dot;
  currr[4] = 1;
  below = a[current + 1];
  a[0] = curr;
  currenth = 3;
  current = 0;
  game[current][3] = 1;
  a[current] = B10010001;
  lc.setRow(0,0,curr);

if((game[0][1] == 1 && game[1][1] == 1 && game[2][1] == 1 && game[3][1] == 1 && game[4][1] == 1 && game[5][1] == 1 && game[6][1] == 1 && game[7][1] == 1) ||
(game[0][2] == 1 && game[2][2] == 1 && game[3][2] == 1 && game[4][2] == 1 && game[5][2] == 1 && game[6][2] == 1 && game[7][2] == 1 && game[1][2] == 1) ||
(game[0][3] == 1 && game[2][3] == 1 && game[3][3] == 1 && game[4][3] == 1 && game[5][3] == 1 && game[6][3] == 1 && game[7][3] == 1 && game[1][3] == 1) ||
(game[0][4] == 1 && game[2][4] == 1 && game[3][4] == 1 && game[4][4] == 1 && game[5][4] == 1 && game[6][4] == 1 && game[7][4] == 1 && game[1][4] == 1) ||
(game[0][5] == 1 && game[2][5] == 1 && game[3][5] == 1 && game[4][5] == 1 && game[5][5] == 1 && game[6][5] == 1 && game[7][5] == 1 && game[1][5] == 1) ||
(game[0][6] == 1 && game[2][6] == 1 && game[3][6] == 1 && game[4][6] == 1 && game[5][6] == 1 && game[6][6] == 1 && game[7][6] == 1 && game[1][6] == 1)){
  endgame = 1;
}
}
```

Additional Help

https://create.arduino.cc/projecthub/Heathen_Hacks-v2/super-basic-max7219-led-matrix-module-project-for-arduinoobs-24e776

- Was used to understand how to use the LED matrix with the arduino.

https://create.arduino.cc/projecthub/MisterBotBreak/how-to-use-a-joystick-with-serial-monitor-1f04f0

- Was used to understand how the joystick works with the arduino.