Pong Ball Dodge Game

Custom Project Final Report

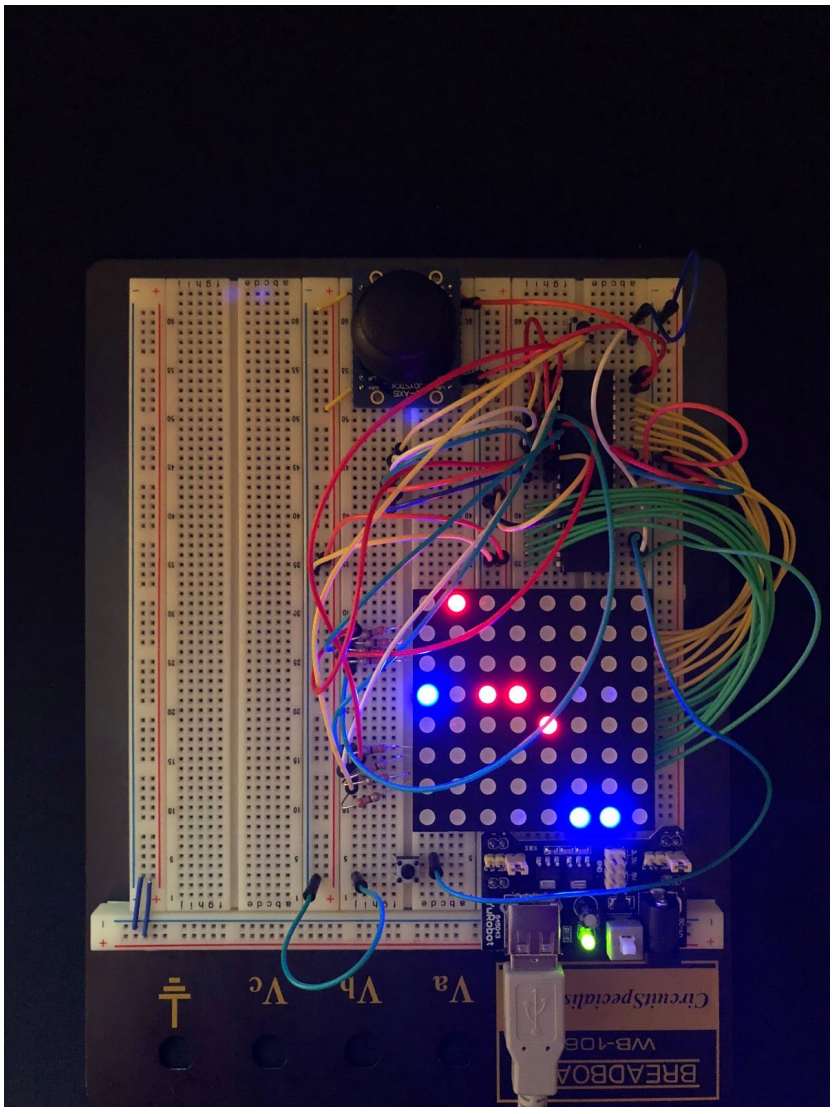
Winter 2018

Andrew Pirelli

Table of Contents

Introduction

The Pong Ball Dodge Game is a game where you maneuver your avatar (a blue dot on the LED matrix) using a joystick in order to dodge 4 bouncing red dots which behave similarly to the ball in Pong. The player's score is displayed on the bottom row of the LED matrix and is increased on every successful player movement cycle. The highest score is then saved to the ATMega1284p's EEPROM and displayed on the bottom row of the game while it is not running. A button above the microcontroller toggles the game to start/end upon being pressed while a button to the bottom left of the matrix resets the high score to a default value of 1 upon being pressed.



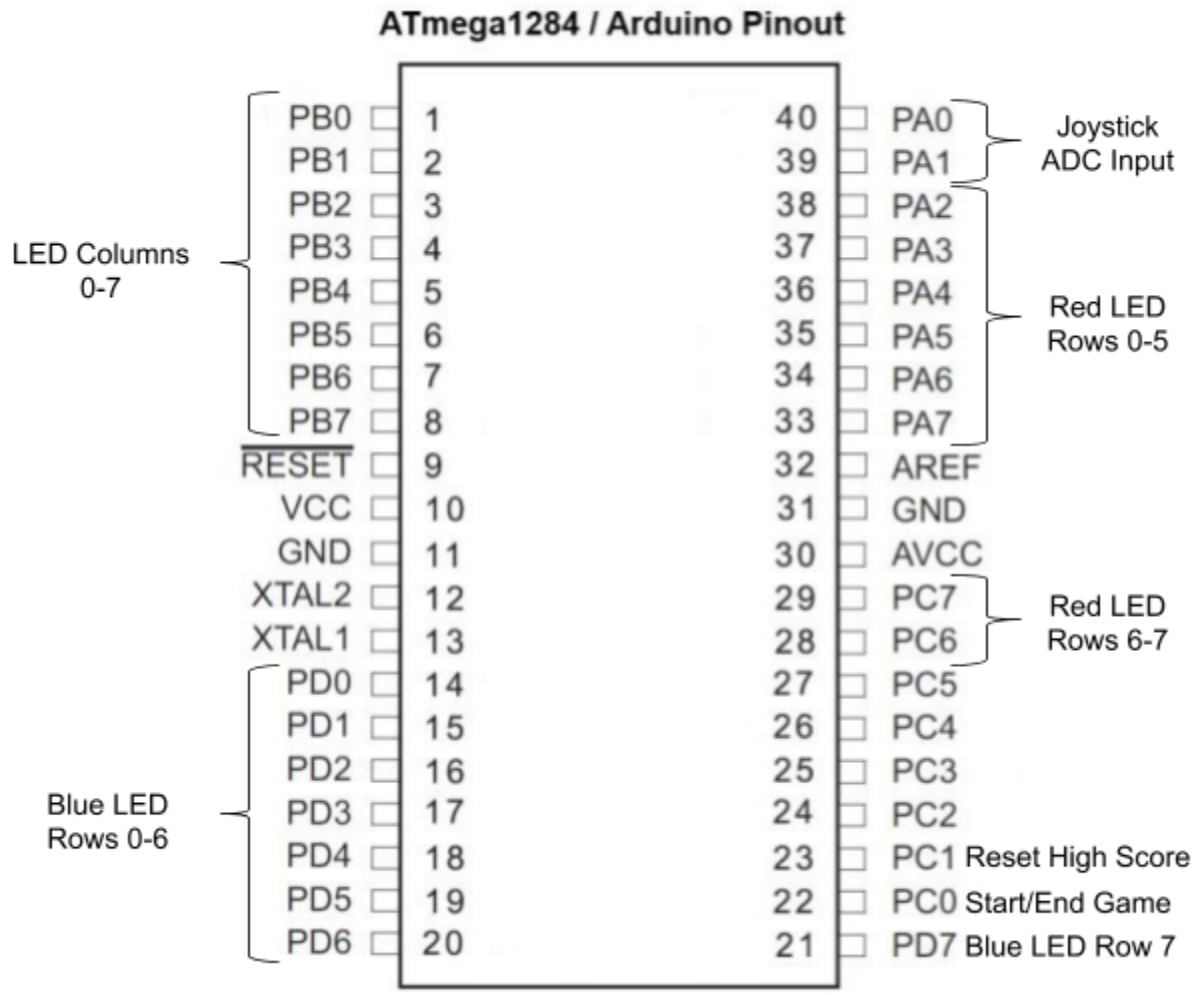
Hardware

Parts List

The hardware that was used in this design is listed below. The equipment that was not taught in this course has been bolded.

- ATMega1284p microcontroller
- **8x8 RGB LED Matrix**
- **ATMega1284p EEPROM**
- 2 Buttons
- **Joystick**

Pinout



Software

```
//NOTE: I refer to the red dots as ghosts in my code
//GameSM
//Period: 200 ms
//Main SM for game and game logic
enum Game_States { START, init, wait, play, end } Game_State;

void Game_Tick() {
    switch(Game_State) { // Transitions
        case START:
        Game_State = init;
        break;

        case init:
        Game_State = wait;
        break;

        case wait:
        if(gameStart()){ //Checks if start button is pressed
            Game_State = play;
        }
        else{
            Game_State = wait;
        }
        break;

        case play:
        if(gameQuit() || gameLose){ //Checks if end button is pressed or if player has lost
            Game_State = end;
        }
        else{
            Game_State = play;
        }
        break;

        case end:
        Game_State = init;
        break;

        default:
        Game_State = init;
        break;
    } // Transitions
```

```
    switch(Game_State) { // State actions
        case init:
        //resets game variables
        gameOn = 0;
        gameLose = 0;
        score = 0;
        //player position variables
        countX = 4;
        countY = 3;
        //ghost position variables
        ghost1 = 0;
        ghost1X = 0;
        ghost1Y = 0;
        ghost2 = 0;
        ghost2X = 0;
        ghost2Y = 6;
        ghost3 = 0;
        ghost3X = 7;
        ghost3Y = 0;
        ghost4 = 0;
        ghost4X = 7;
        ghost4Y = 6;
        break;

        case wait:
        //do nothing
        break;

        case play:
        //Sets gameOn = 1 so that displaySM starts displaying game
        gameOn = 1;
        //Does logic for ghost movement, then updates global ghost position variables
        ghost1Move();
        ghost2Move();
        ghost3Move();
        ghost4Move();
        //Does logic for play movement based on ADC input from joystick
        //Then updates global position variables
        playerMove();
        //Checks if player intersects with any ghosts, sets gameLose = 1 if intersection is found
        intersect();
        break;

        case end:
        if(score > hiScore){ //game ended, check if new hiScore
            hiScore = score;
            //write hiScore into EEPROM
            ByteOfData = (uint8_t)hiScore;
            eeprom_write_byte((uint8_t*)46, ByteOfData);
        }
        //this state also causes the game end screen to display for 300ms since it
        //delays gameLose from being set back to 0 while this happens
        break;

        default:
        //do nothing
        break;
    } // State actions
}
```

```
//DisplaySM
//Period: 1 ms
//Displays the ghosts, player avatar and score using multiplexing
enum Display_States { dSTART, dinit, dwait, dg1, dg2, dg3, dg4, dp, ds } Display_State;

void Display_Tick(){
    switch(Display_State) { // Transitions
        case dSTART:
        Display_State = dinit;
        break;

        case dinit:
        Display_State = dwait;
        break;

        case dwait:
        if(gameOn){ //checks if game has started running
            Display_State = dg1;
        }
        else{
            Display_State = dwait;
        }
        break;

        case dg1:
        if(gameOn){ Display_State = dg2; } else{ Display_State = dwait; }
        break;

        case dg2:
        if(gameOn){ Display_State = dg3; } else{ Display_State = dwait; }
        break;

        case dg3:
        if(gameOn){ Display_State = dg4; } else{ Display_State = dwait; }
        break;

        case dg4:
        if(gameOn){ Display_State = dp; } else{ Display_State = dwait; }
        break;

        case dp:
        if(gameOn){ Display_State = ds; } else{ Display_State = dwait; }
        break;

        case ds:
        if(gameOn){ Display_State = dg1; } else{ Display_State = dwait; }
        break;

        default:
        Display_State = init;
        break;
    } // Transitions
```

```
    switch(Display_State) { // State Actions
        case dwait:
        displayScore(hiScore, 0x80);
        break;

        case dg1:
        displayRed((0x01 << ghost1X), (0x01 << ghost1Y));
        break;

        case dg2:
        displayRed((0x01 << ghost2X), (0x01 << ghost2Y));
        break;

        case dg3:
        displayRed((0x01 << ghost3X), (0x01 << ghost3Y));
        break;

        case dg4:
        displayRed((0x01 << ghost4X), (0x01 << ghost4Y));
        break;

        case dp:
        displayBlue((0x01 << countX), (0x01 << countY));
        break;

        case ds:
        displayScore(score, 0x80);
        break;

        default:
        Display_State = init;
        break;
    } // State Actions
}
```

Complexities

Completed Complexities:

- Integrating and calibrating the joystick
- Using EEPROM to save the high score (minimum time)
- Properly displaying the game on the LED Matrix

Incomplete complexities:

- Using shift registers to display on the LED Matrix

Youtube Link

https://www.youtube.com/watch?v=80mZRALTcxQ

Known Bugs and Shortcomings

- The LED matrix was not able to display the game in as much detail as I would have liked, whereas using an LCD screen would have allowed me to display the player's avatar and the objects to dodge in a more interesting way with custom characters.

Future work

I would implement the LED matrix using shift registers, then used those extra pins along with my free pins to add an LCD screen that would display how to start/quit the game and display both the score and high score. This would allow the full LED matrix to be used for the game and make it more clear when the game is running/waiting and what the scores are on the LCD screen.