CS/EE 120B Custom Laboratory Project Report

Multiplayer Pong Game

Hugo Centeno Sanz

06/12/2025

1. Introduction

This project will aim to recreate the famous arcade game pong. Pong is a game where two players compete against each other by passing a ball which bounces on a paddle. A player scores a point if its shot is not able to be returned by the opposite player. The first player who reaches 5 points wins.

Known bugs and limitations:

The animation of the ball during the kick-off could be improved. Instead of erasing the last ball and drawing a new one, just the contour of the last ball could be erased. The animation would be more smooth.

Due to structural limitations they joysticks are not very player-friendly and should be treated carefully as the cables are prone to disconnecting.

In order to start again, reset should be applied to the Arduino, or the program should be reloaded.

2. Build-upons
   Successfully implemented build-upons:
   -ST7735 128 * 128 LCD screen to display the game.
   -1602 16 * 2 LCD screen to display the score.
   -Controllers for both players
   -Start button
   Unsuccessfully implemented build-upons:
   -Active buzzer to buzz when ball hits an element (not fully implemented due to lack of unused pins).
   -Using the EEPROM to save past matches, i.e. the who played and what the score was.
   -Adding an ability so that a player can shot multiple balls.

3. User Guide

The players are first shown a wait screen, where they are induced to press the button the begin the game.

Once the button is pressed and released, the game begins. In the main LCD screen the two paddles and the ball are shown. The left joystick controls the left paddle, and the right joystick the right paddle. The players can move up or down.

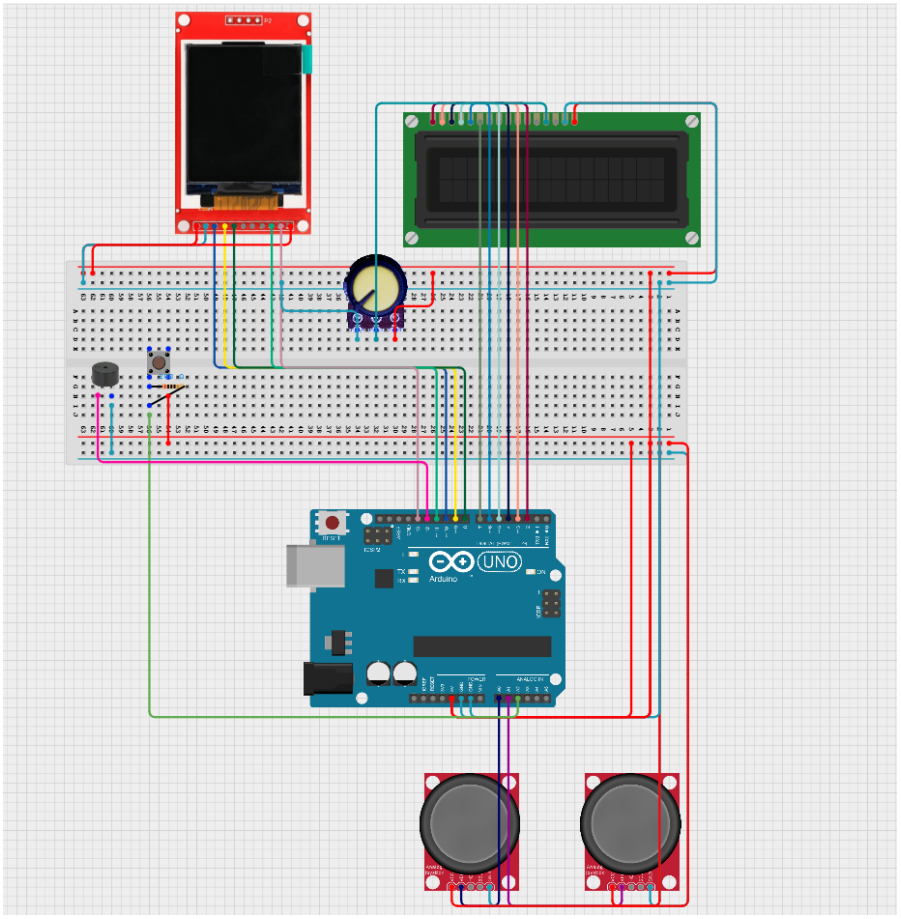
While the game is being played, the current score is displayed on the secondary screen.

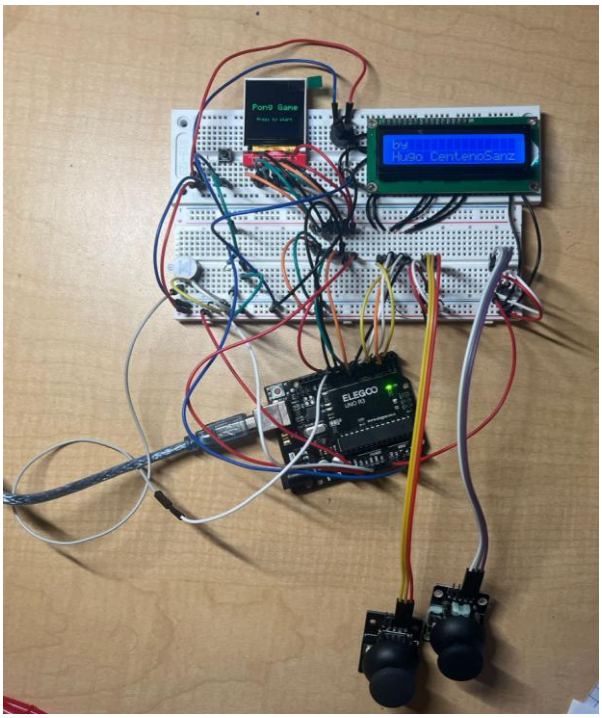
Once one player scores 5 goals the game finished. The winner player is displayed on the primary screen, and a message is displayed on the secondary screen.

4. Hardware Components Used
   - ST7735 128 * 128 LCD screen
   -1602 16 * 2 LCD screen
   -Joysticks
   -Button
   -Resistor
   -Potentiometer
   -Jumping wires
   -Arduino UNO R3 Controller Board

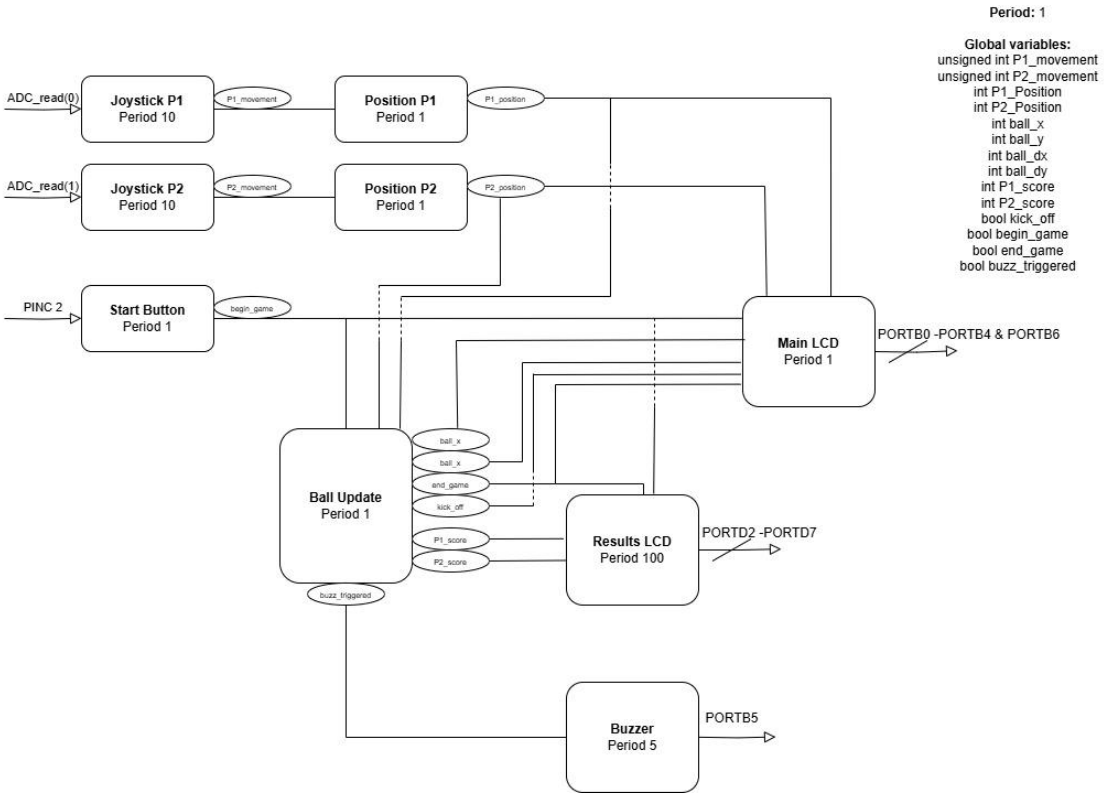
5. Software Libraries Used
   -spiAVR.h helps with the ST7735 LCD communication by defining the initialization, reset, command and data sending functions.
   -periph.h includes abstractions to use the ST7735 LCD to draw strings and shapes. I created an array defining a 5x7 ASCII font for use with the LCD display.
   -LCD.h includes abstractions to use the 1602 16 * 2 LCD screen. Initialize it, draw strings.
   -helper.h includes abstractions to use the joysticks.

6. Wiring Diagram

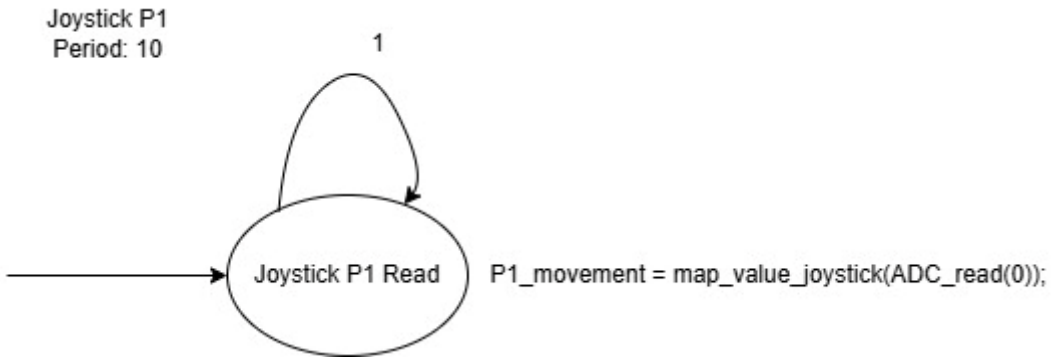


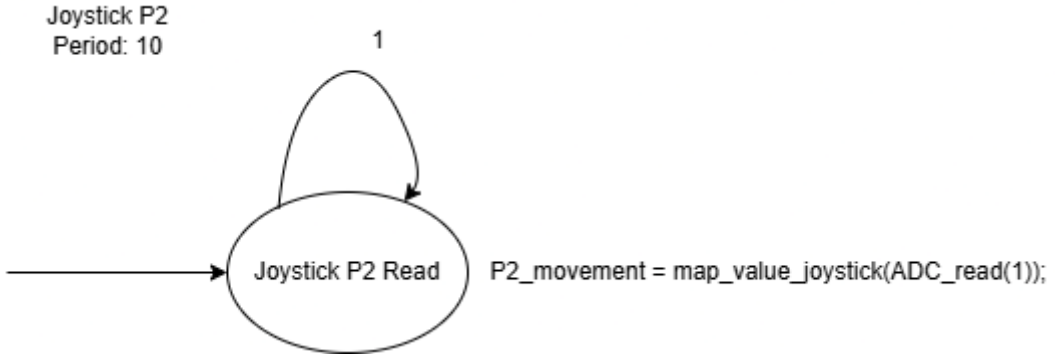


7. Task Diagram



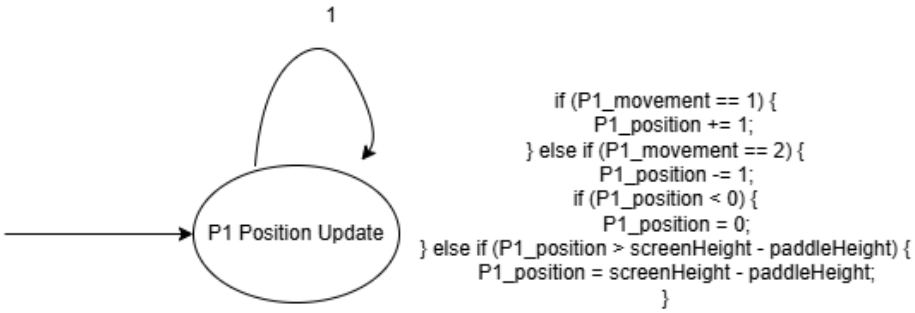
8. SynchSM Diagrams

- 



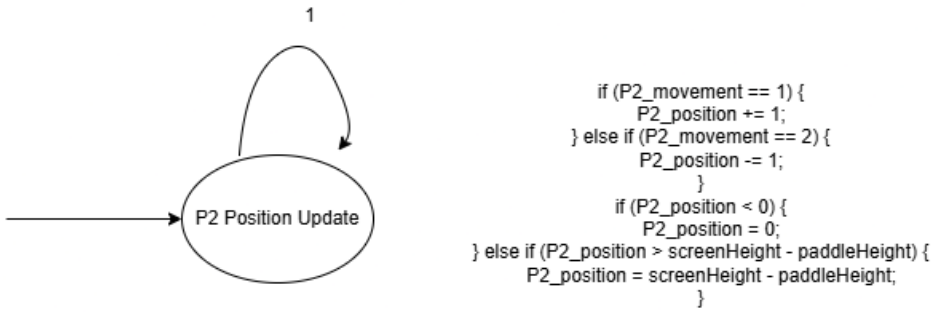
Position P1
Period: 1

screenHeigh and paddleHeight are definitions in the program



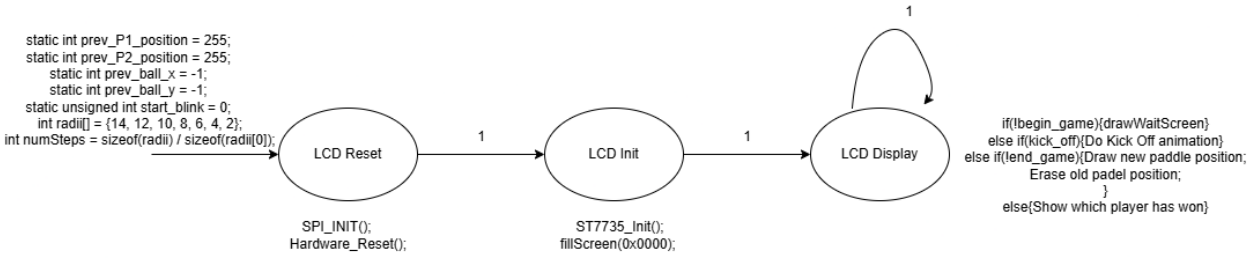
Position P2
Period: 1

screenHeigh and paddleHeight are definitions in the program



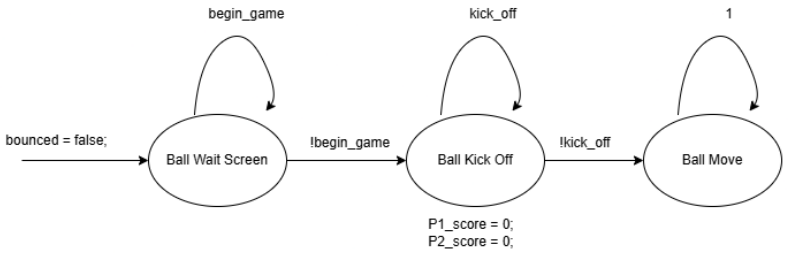
MainLCD
Period: 1

```
static int prev_P1_position;
static int prev_P2_position;
static int prev_ball_x;
static int prev_ball_y;
static unsigned int start_blink ;
int radii[7];
int numSteps;
```



Ball
Period: 1

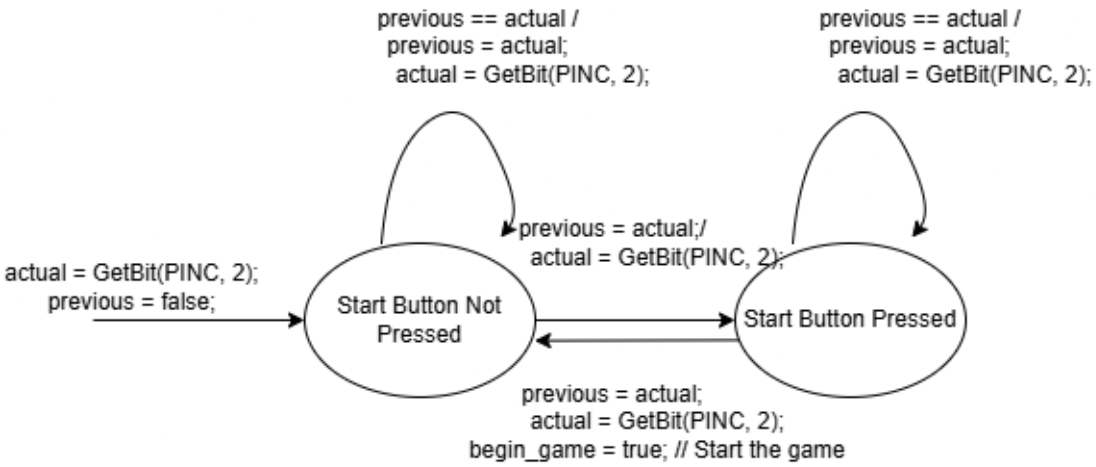
bool bounced;




```
if (!goal_scored) {
        if (ball_x >= screenWidth + X_OFFSET + BALL_RADIUS) {
            P1_score++;
            goal_scored = true;
            ball_x = screenWidth / 2;
            ball_y = screenHeight / 2;
            ball_dx = -ball_dx;
        }
        else if (ball_x <= -BALL_RADIUS) {
            P2_score++;
            goal_scored = true;
            ball_x = screenWidth / 2;
            ball_y = screenHeight / 2;
            ball_dx = -ball_dx;
        }
        if( P1_score >= 5 || P2_score >= 5) {
            end_game = true; // End the game if a player reaches 5 points
        }
    } else {
        if (ball_x > 0 && ball_x < screenWidth) {
            goal_scored = false;
        }
    }
    // Move ball
    ball_x += ball_dx;
    ball_y += ball_dy;
    // Bounce off top/bottom
    if (ball_y <= 0 || ball_y >= screenHeight - BALL_RADIUS) {
        ball_dy = -ball_dy;
        bounced = true; // Set bounced to true when bouncing off top/bottom
    }
    // Paddle collisions
    if (ball_dx < 0) {
        if (ball_x <= X_OFFSET + paddleWidth &&
            ball_y + BALL_RADIUS >= P1_position &&
            ball_y <= P1_position + paddleHeight) {
            ball_dx = -ball_dx;
            bounced = true;
        }
    } else {
        if (ball_x + BALL_RADIUS >= screenWidth - paddleWidth &&
            ball_y + BALL_RADIUS >= P2_position &&
            ball_y <= P2_position + paddleHeight) {
            ball_dx = -ball_dx;
            bounced = true;
        }
    }
    buzz_triggered = bounced;
```

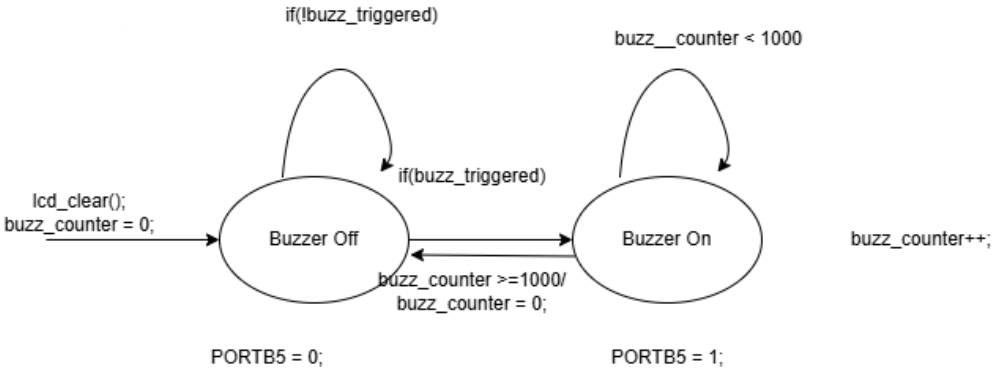
Start Button
Period: 1

bool previous;
bool actual;



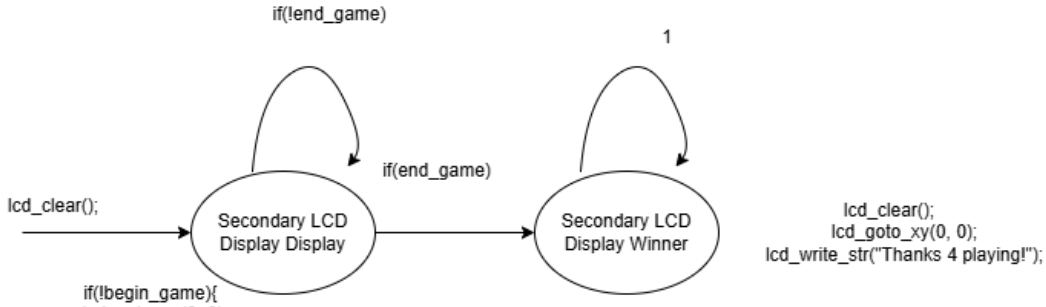
Buzzer
Period: 5

```
static unsigned long buzz_counter;
```



Results LCD
Period: 100

```
char bufferP1[6];
 char bufferP2[6];
```




```
            if(!begin_game){
            lcd_goto_xy(0, 0);
           lcd_write_str("by");
            lcd_goto_xy(1, 0);
lcd_write_str("Hugo CentenoSanz");
                }else{
                lcd_clear();
             // Display scores
            lcd_goto_xy(0, 0);
     lcd_write_str("P1 SCORE:");
           lcd_goto_xy(0, 10);
     itoa(P1_score, bufferP1, 10);
 for (int i = 0; bufferP1[i] != '\0'; i++) {
    lcd_write_character(bufferP1[i]);
                    }

           lcd_goto_xy(1, 0);
     lcd_write_str("P2 SCORE:");
           lcd_goto_xy(1, 10);
       itoa(P2_score, bufferP2, 10);
 for (int i = 0; bufferP2[i] != '\0'; i++) {
    lcd_write_character(bufferP2[i]);
                }
                  }
```