Reproducing the classical pong game

Thorvald Ballestad

March 15, 2020

Abstract

We attemptet reproducing the classical pong game using an Arduino and a 32x32 pixel LED display.

1 Vision

The initial idea was to create a pong game. The pong game is an arcade classic. Released in 1972 it was one of the first arcade video games created. The game is between two players. Each player controls a paddle which can be moved up or down. A ball bounces back and forth, and the players has to hit the ball using the paddle. A player gains a point when the opponent fails to return the ball.

We wished to use a low-pixel display for the project, controlled by an Arduino. The paddles needed some type of control, in the first iteration the idea was to use a simple potentiometer. This input could be made more sophisticated in later iterations.

2 Results

The game is run on an Arduino Uno.

As monitor, a 32x32 RGB LED Matrix Panel from Adafruit, product id 607, was used. Each of the 1024 pixels has a RGB LED pixel, using the Arduino one gets 12-bit color depth.

For user input voltage measurements from simple potentiometers where used.

2.1 Interfacing the display

The first challenge is interfacing the display. On Adafruits homepage there are thorough guides to doing this. The display features a 16 pin connector, the pinout is shown in figure 1.

To reliably connect the display to the Arduino, a cutom shield was created using Arduino's proto shield. This proved to be very advanteagous over simply using jumper cables – it allowed for easily connecting and disconneting the

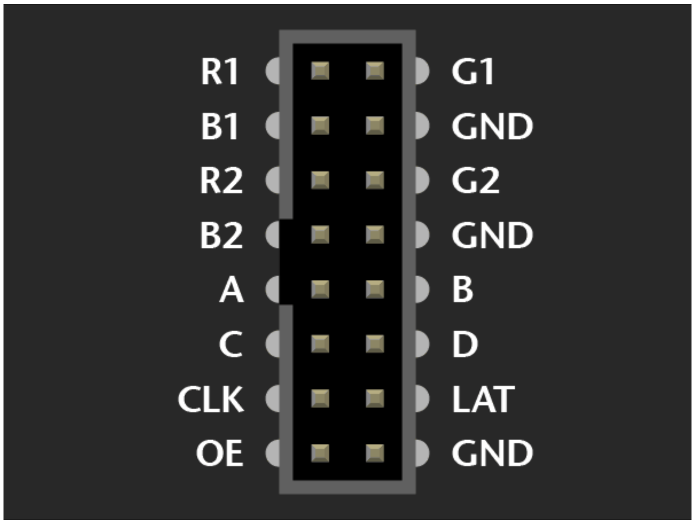

Figure 1: Pinout of connector on the 32x32 display. Note that this is the male connect, mounted on the display itself. Diagram created by Phillip Burges, distributed on Adafruit's homepage under the filename `led_matrix_socket2.png`.

display, and also minimized cables coming loose as a potential source of error. Table 1 shows how the display was connected to the Arduino.

2.2 Software

```
#include <RGBmatrixPanel.h>

#define CLK  8
#define OE   9
#define LAT 10
#define A   A0
#define B   A1
#define C   A2
#define D   A3

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

int rate = 50;
const int WIDTH = 31;
const int HEIGHT = 31;

/*******
* PONG *
*******/
float BALL_SPEED = 1;
int POINTS_TO_WIN = 3;
float BAR_SPEED = 1.8;

```

Display	Arduino
R1	2
G1	3
B1	4
R2	5
G2	6
B2	7
CLK	8
OE	9
LAt	10
A	A0
B	A1
C	A2
D	A3
GND	GND

Table 1: Table explaining the circuit between the 32x32 display and the Arduino. Arduino pins with only a number refers to the digital pins, pins prefixed with “A” refers to analog pins. See figure 1 for the pinout of the display connector.

```
// NB. Initial pos 0 and positive velocity will fuck it up
// (No way that it can reach the state x=0, vx>0, so this is an impossible initial state).
float x = 1;
float y = 1;
float vx = sin(1)*BALL_SPEED;
float vy = cos(1)*BALL_SPEED;

float bar1 = 16;
float bar2 = 16;

int bar1_x = 1;
int bar2_x = 30;

int bar_height = 2;

int points_1 = 0;
int points_2 = 0;

void reset_pong() {
  reset();

  bar1 = 16;
  bar2 = 16;

  points_1 = 0;
  points_2 = 0;
}

void control_bar(float &bar) {
  bar = min(HEIGHT-bar_height, bar);
  bar = max(bar_height, bar);
```

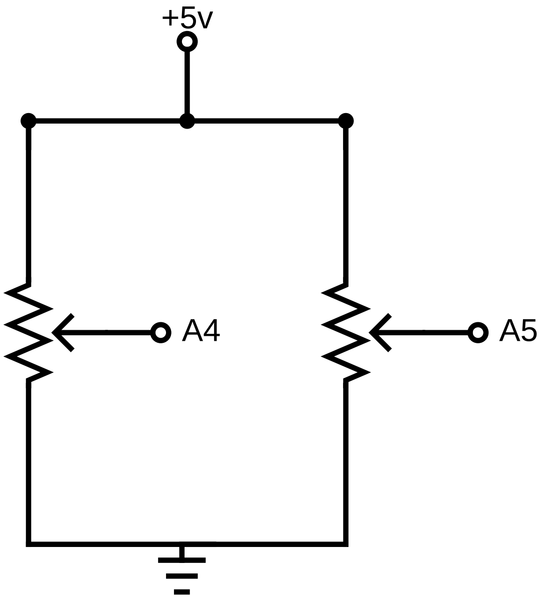

Figure 2: User input circuit. A4 and A5 refers to the analog input pins on the Arduino.

```
}

void game_won() {
  matrix.fillScreen(0);
  for (int i = 0; i < WIDTH/2; i++) {
    matrix.fillCircle(WIDTH/2, HEIGHT/2, i, matrix.Color333(7,0,0))
    ;
    delay(50);
  }

  matrix.setCursor(3, 6);
  matrix.setTextSize(0.5);
  matrix.setTextColor(matrix.Color333(7,7,7));
  matrix.print("Game over!");
  delay(3000);
}

void reset() {
  // Resets game
  // Sets balls velocity in x-direction
  // to between 50 and 100 percent of BALL_SPEED
  // sets y to be such that total speed is BALL_SPEED
  x = int(WIDTH/2);
  y = int(HEIGHT/2);
  vx = random(50, 100)*BALL_SPEED/100.0;
  vy = sqrt(sq(BALL_SPEED)-sq(vx));
  int signx = random(-10, 10) > 0? -1 : 1;
  int signy = random(-10, 10) > 0? -1 : 1;
  vx *= signx;
  vy *= signy;
  draw();
}

void draw() {

  matrix.fillScreen(matrix.Color333(0, 0, 0));
  matrix.drawPixel((int) x, (int) y, matrix.Color333(4, 0, 0));
  matrix.drawLine(bar1_x, int(bar1) - bar_height, bar1_x, int(bar1) + bar_height, matrix.Color333(4, 4, 4));
  matrix.drawLine(bar2_x, int(bar2) - bar_height, bar2_x, int(bar2) + bar_height, matrix.Color333(4, 4, 4));

  matrix.setCursor(10, 0);
  matrix.setTextSize(0.5);
  matrix.setTextColor(matrix.Color333(0,7,0));
  matrix.print(points_1);
  matrix.setCursor(20, 0);   // start at top left, with one pixel of spacing
  matrix.print(points_2);

}

void pong() {
  reset_pong();
  while(true) {
    /***********
     * Measure *
     ***********/
    int sensorValue1 = analogRead(A4);
    int sensorValue2 = analogRead(A5);
    float voltage1 = sensorValue1 * (5.0 / 1023.0);
    float voltage2 = sensorValue2 * (5.0 / 1023.0);

    /********
     * Draw *
     ********/
    draw();

    /**************
     * Game logic *
     **************/

    float delta_bar1 = (2.5 - voltage1)/2.5;
    float delta_bar2 = (2.5 - voltage2)/2.5;
    bar1 += abs(delta_bar1)<0.3 ? 0 : delta_bar1*BAR_SPEED;
    bar2 += abs(delta_bar2)<0.3 ? 0 : delta_bar2*BAR_SPEED;
    control_bar(bar1);
    control_bar(bar2);

    // Check collision with bar
    if (int(x) <= bar1_x and bar1 + bar_height >= y and bar1 - bar_height <= y) {
      vx = abs(vx);
    }

    if (int(x) >= bar2_x and bar2 + bar_height >= y and bar2 - bar_height <= y) {
      vx = -abs(vx);
    }

    // Check collison with top or bottom wall, reflect ball
    if (y <= 0 or y >= HEIGHT) {
      vy = -vy;
    }

    // Check collision with end walls, add points
    if (x <= 0){
      points_2++;
      reset();
      delay(400);
    }
    if(x >= WIDTH){
      points_1++;
      reset();
    }

    x += vx;
    y += vy;

    if(points_1==POINTS_TO_WIN or points_2==POINTS_TO_WIN){
      game_won();
      break;
    }

    delay(rate);
  }
}

void setup() {
  matrix.begin();
}

void loop() {
  while(true){
  matrix.fillScreen(0);
  matrix.setCursor(6, 2);
  matrix.print(”Menu”);
  matrix.fillRect(6, 15, 9, 9, matrix.Color333(4,4,4));
  matrix.fillRect(19, 15, 9, 9, matrix.Color333(4,4,4));
  matrix.fillCircle(10, 19, 2, matrix.Color333(7,0,0));
  matrix.drawLine(20, 22, 25, 16, matrix.Color333(0, 7, 0));
  matrix.drawPixel(26, 16, matrix.Color333(7, 0, 0));
  int x = analogRead(A4)* (5.0 / 1023.0)*6;
  int y = analogRead(A5)* (5.0 / 1023.0)*6;
  matrix.fillCircle(x, y, 1, matrix.Color333(0,0,7));
  delay(5000);
  if(6<x<15){
  pong();}
  delay(40);
}}
```

2.3 Full list of parts

Finally, here is a complete list of parts used.

- 32x32 RGB LED Matrix Panel from Adafruit, product id 607.
- Arduino Uno
- Arduino Proto shield (Uno size)
- Two 10 kΩ potentiometers
- A black metal plate scavenged from an old computer casing

The total cost of the parts was zero NOK, as all the parts where found on the lab.