Project Report

A Driving Game: Move Before the Skyfall

Srećko Ćurković & Yingsi Qin

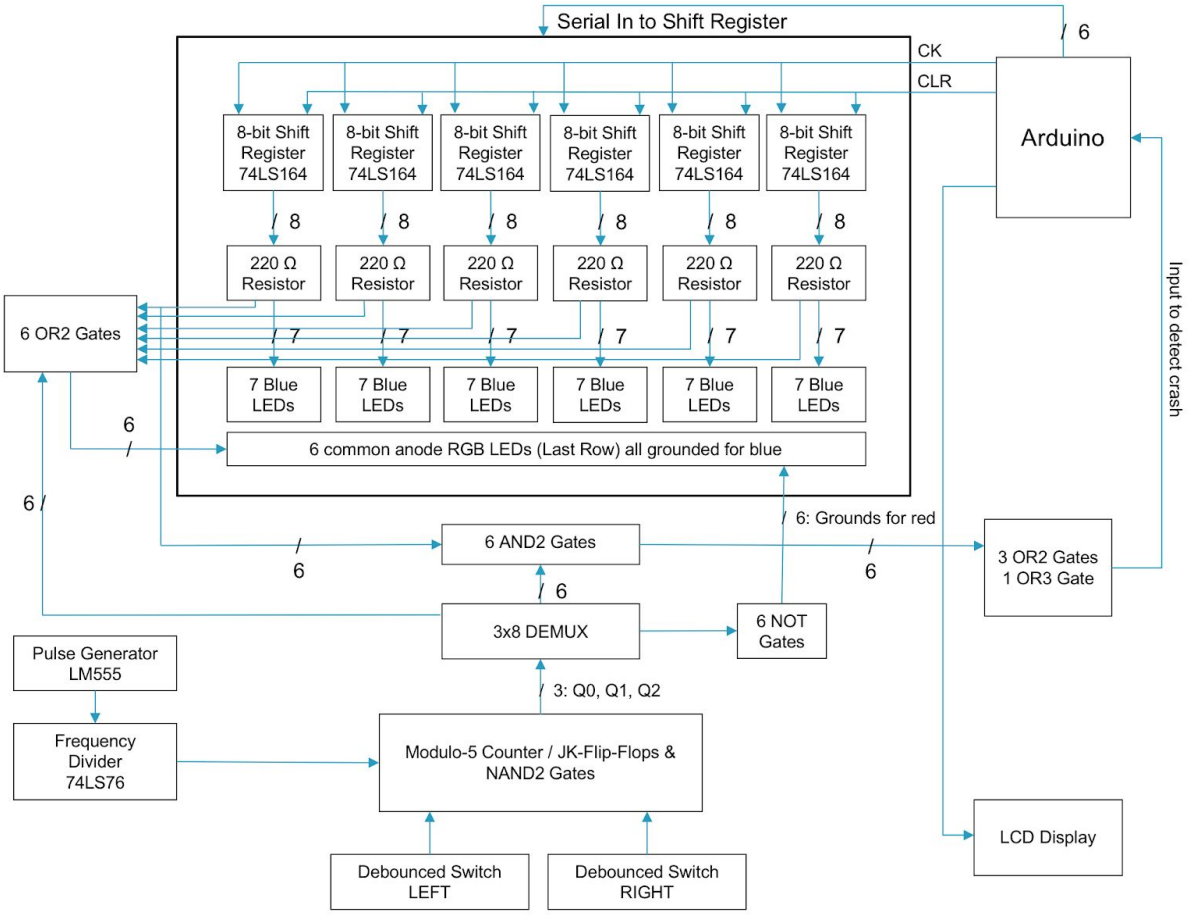
Project Video Demo

https://youtu.be/MYIakPhqPUc

Project Description

Our project is a simulation of the game in which a user controls the motion of the car in order to avoid obstacles that the car is continuously encountering. The car can move left or right by pressing the respective buttons. The car and the obstacles are displayed with a 8x6 matrix of LEDs. The position of the car corresponds to a common anode RGB LED in the last row which is lit up as red at a given time. Other rows of blue LEDs are used to show the obstacles by lighting certain LEDs up. The player is able to view their current score and level on the LCD display. Their score increases for every additional step the player takes. The level increases as the car continues to move. The speed of the car increases as the level increases. The game ends either when the car crashes into an obstacle or when the player reaches a winning level.

There are 4 main parts in our circuit design: the 8x6 matrix of LEDs, the shift registers and resistors, the movement of the car, and Arduino. A circuit diagram is displayed below and the whole circuit picture is displayed after explanation of all parts. We go through each part in detail below. Arduino code is attached at the end.



- Circuit Diagram -

8x6 matrix of LEDs

- Made up of 6 columns, each consists of 7 blue LEDs and 1 common anode RGB LED for the last row (Figure 1). We tested common anode RGB LEDs and found that the red overwrites the blue when both are grounded. The blue grounds of the RGB LEDs are all connected to ground. We NOTed the outputs from the demultiplexer and connected them to the red grounds of the common anode RGB LEDs (Figure 3). So the RGB LEDs appear blue when the shift registers output HIGH and only appear red when the

demultiplexer outputs HIGH for the specific bulb. The players are then able to distinguish the car and the obstacles.

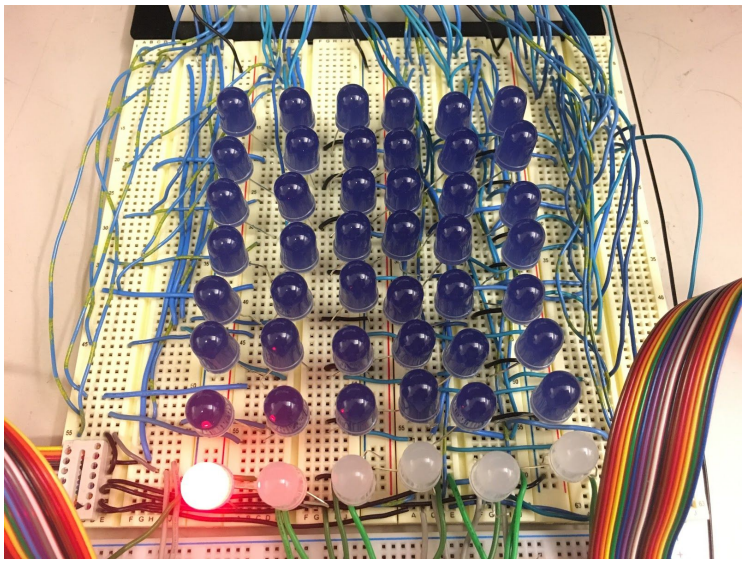


Figure 1. 8x6 matrix of LEDs

Shift registers and resistors

- Each combination of a shift register chip and a resistor chip controls one column of 8 LEDs (Figure 2). For the first six LEDs in each column, the serial data inputs and clock pulses from the arduino are sent to the shift registers and the data outputs from the shift registers are connected to respective 220 Ω resistors and then to the blue LEDs. For the last row of RGB LEDs, the data outputs from the shift registers are connected to the respective 220 Ω resistors then ORed with outputs from the demultiplexer. The outputs from these 6 OR gates are then connected to the RGB LEDs.

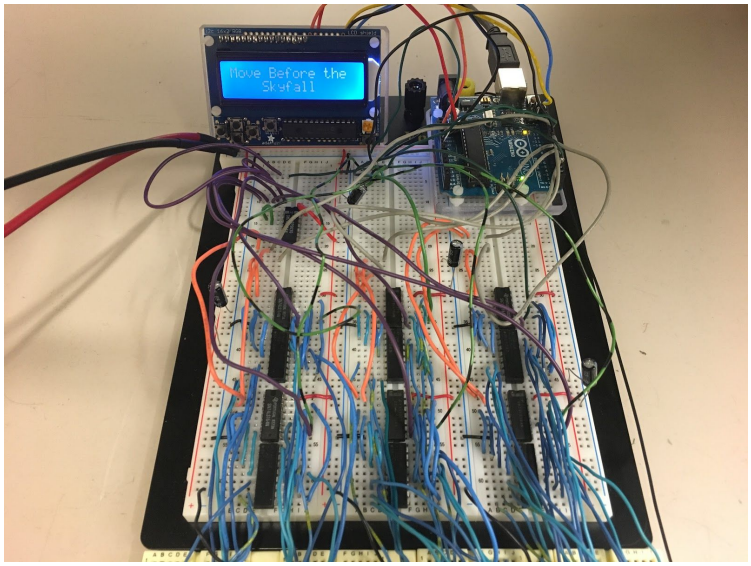


Figure 2. shift registers and 220 Ω resistors

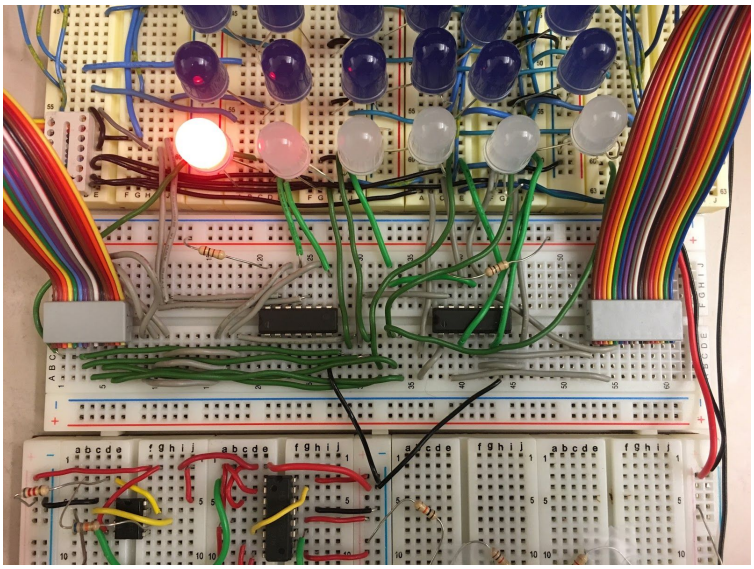


Figure 3. OR2 gates, data outputs and grounds (red) to the last row of RGB LED

Movement of the car (Figure 4)

- Modulo-5 counter
    - The clock pulse is generated from LM555 the pulse generator and then divided in 4 by a frequency divider, 74LS76. We used a decade counter chip, 74LS190 and

made it a modulo-5 counter given that we have 6 LEDs on the last row. Because the counter chip doesn't have CLR, we used the load' (low active) feature to make it a modulo-5 counter.

- Debounced switches (up/down, enable)
    - Switch RIGHT enables the counter and sends 0 to down/up' to the counter
        - When it's pressed, 0000 is sent to the data input of the counter, so that the counter loads from 0000 and counts up when it reaches 0110
    - Switch LEFT enables the counter and sends 1 to down/up' to the counter
        - When it's pressed, 0101 is sent to the data input of the counter, so that the counter loads from 0101 and counts down for every loop
    - Because the switch outputs 1 when it's pressed and the player only presses the switch for a short time, the car moves only by 1 LED each time the player presses the switch.
- 3x8 DEMUX
    - We send data outputs Q0, Q1, and Q2 from the counter to the demultiplexer and take the first six outputs from the DEMUX and OR them respectively with the signals from the shift registers. The six outputs from the OR gates are sent to the last row of common anode RGB LEDs.
    - The first six outputs from the demultiplexers are also NOTed and sent to the grounds for red color of the last row of common anode RGB LEDs.
- Car crash
    - The outputs for the last row from the shift registers are ANDed with the outputs from the 3x8 demultiplexer. The outputs from the 6 AND gates are ORed and

then connected to Arduino. This output will be 1 when the car crashes into an obstacle and remains 0 when there's no crash.

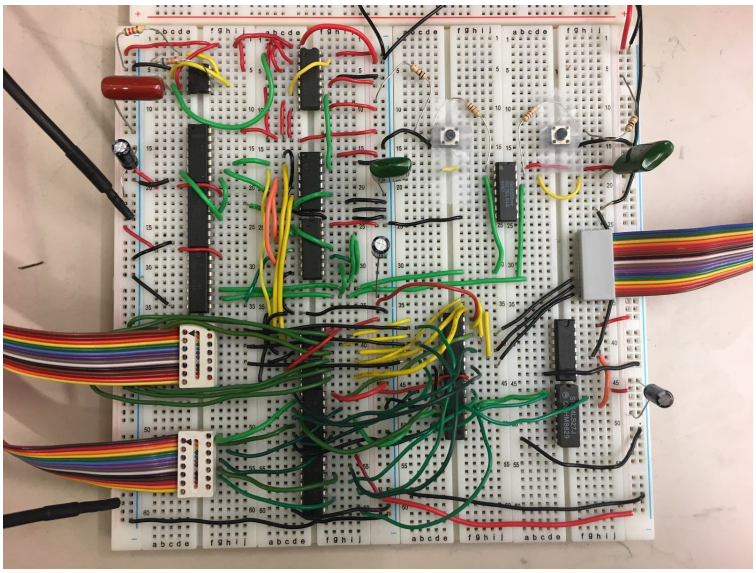
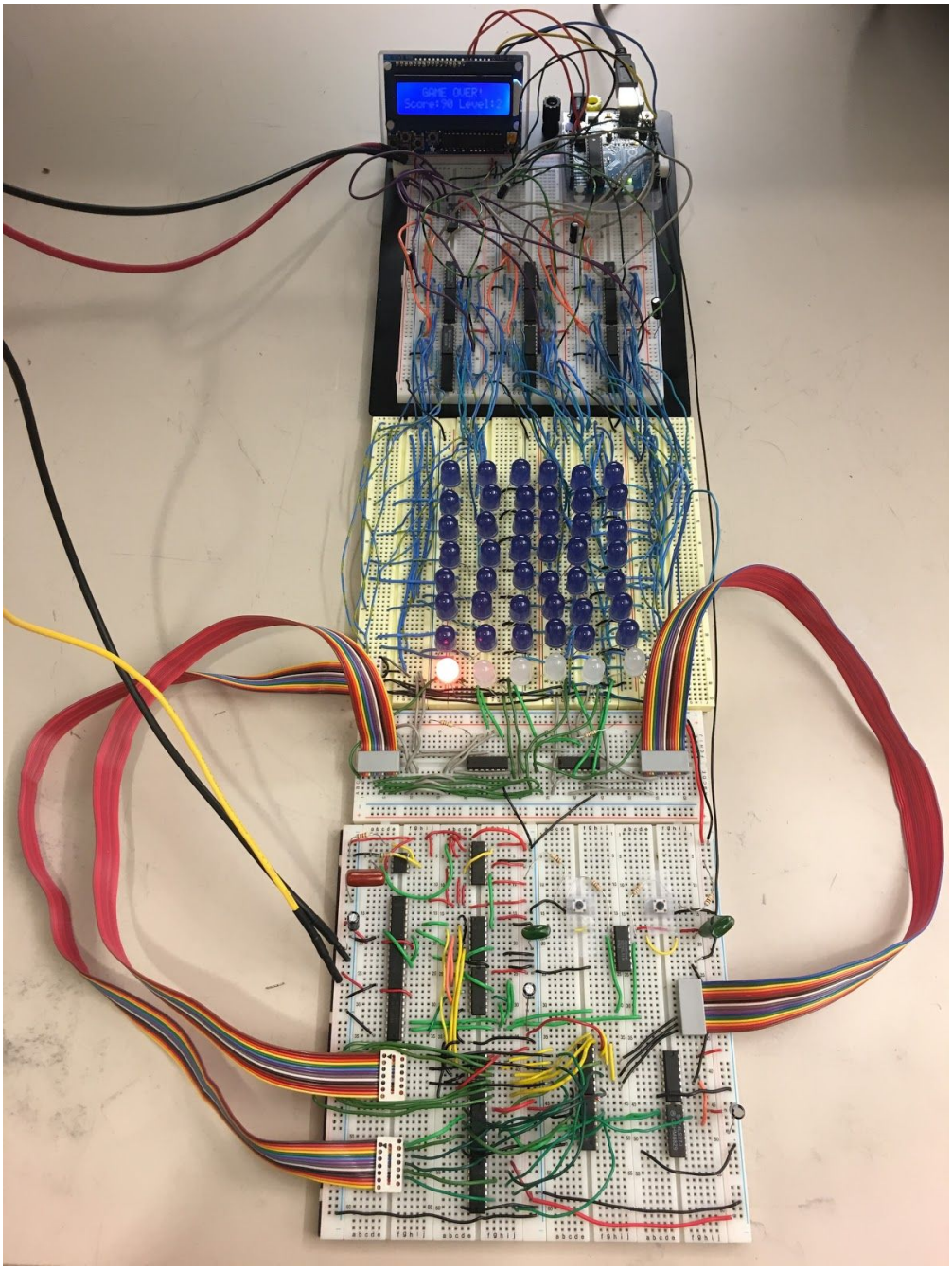


Figure 4. Circuit that controls the movement of the car and detects car crash

Arduino

- The track pattern was programmed in Arduino and high and low voltages are sent to the data inputs and clocks of the 8 shift registers. The current score and level are output from Arduino and shown on the LCD display. Their score increases for every additional step the player takes. The level of the game increases for every loop of serial data that the Arduino finishes sending. The speed of the car increases (delay decreases) as the level increases. The game ends either when the car crashes into an obstacle (when one of the ANDs of the signal from shift registers and debounced switches/DEMUX outputs 1) or when the player reaches the winLevel which can be defined at the beginning of the Arduino code. When the game ends, Arduino outputs 0 to the CLEAR of all shift registers.



- Whole Circuit Photo -

Discussion of challenges

During this project, we encountered various challenges and difficulties. The first main challenge was to determine how to generate the light pattern. Originally we spend two days

building a state machine for each column, which was tremendous amount of work given that we wanted eight LEDs in each column. We soon realized that using shift registers instead is much easier. The next main challenge was finding out why the output from the 6 AND and OR gates that detects the car crash did not output proper logic. We spent a long time reanalyzing the circuit and tried different options to detect the car crash. Originally, we ANDed the current voltages of each LED in the last row (which we thought would be the same as the outputs from the shift registers) with outputs from the demultiplexer and ORed them together. Theoretically, this output will be 1 when there is a car crash. Eventually, we figured out that the problem existed because we had the outputs from the shift registers as well as the outputs from the demultiplexer *directly* connected to the last row. Because only one LED is lit up at a time as the car, other LEDs always have LOW voltage and therefore they did not light up (except for the car) even though there were signals sent from the shift registers. As a result, the current voltages of each LED that we sent to the AND gates were different from outputs from the shift registers. To solve this issue, we ORed the outputs from the shift registers with the outputs from the demultiplexer and send them to the last row of LEDs. Then the LEDs lit up either when the shift registers output HIGH or the demultiplexer output HIGH. We also changed the inputs for the AND gates to detect a car crash: we ANDed the outputs from the shift registers for the last row and outputs from the demultiplexer.

Arduino Code

```
int clockPin = 2;
int dataPin6 = 3;
int dataPin5 = 11;
int dataPin4 = 5;
int dataPin3 = 6;
int dataPin2 = 7;
int dataPin1 = 8;
```

```
int crash = 9;
int CLR = 10;
int speed = 220;
int level = 1;
int score = 0;
int terminate = 0;
int winLevel = 5;


byte bits6[] = {B11111110,B00000000,B00011111,B11111111,
                B11111111,B11110000,B11110000,B00001111,
                B11111111,B11111100,B11100000,B00000001,
                B11111111,B11111111,B11111111,B11111111,B00001111};
byte bits5[] = {B00000000,B00000110,B00000111,B11111111,
                B00110000,B00000000,B00111100,B00001110,
                B01111111,B00100000,B00011111,B11111111,
                B00000000,B11111111,B11000000,B00000000,B11000000};
byte bits4[] = {B00111000,B00011111,B10000001,B11111100,
                B00000000,B00001100,B00001111,B00111000,
                B01111111,B00000011,B10000000,B11111110,
                B00001100,B00011110,B00001100,B11000000,B00110000};
byte bits3[] = {B00000000,B01111111,B11000000,B01110000,
                B11000000,B00111111,B00000011,B11110000,
                B00001100,B00011100,B11100000,B00111000,
                B00110011,B00000000,B11000000,B11110011,B00000000};
byte bits2[] = {B00000001,B11111111,B11111110,B00000000,
                B00001100,B11111111,B11000000,B11000000,
                B00000000,B00110000,B01111000,B00000000,
                B11000000,B11001100,B00000000,B00111100,B00000000};
byte bits1[] = {B11111111,B11111111,B11111110,B00000011,
                B11111111,B11000000,B11111100,B00000111,
                B11110001,B10000000,B00111111,B10000111,
                B00000000,B00111111,B11111111,B00000000,B11111111};

#include <Wire.h>
#include <Adafruit_RGBLCDShield.h>
#include <utility/Adafruit_MCP23017.h>
```

```
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();

void setup() {

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, INPUT);
  pinMode(10, OUTPUT);

  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("Move Before the");
  lcd.setCursor(4,1);
  lcd.print("Skyfall");
  delay(2000);
  lcd.setCursor(0,0);
  lcd.print("       3        ");
  lcd.setCursor(4,1);
  lcd.print("       ");
  delay(1000);
  lcd.setCursor(0,0);
  lcd.print("       2        ");
  lcd.setCursor(4,1);
  lcd.print("       ");
  delay(1000);
  lcd.setCursor(0,0);
  lcd.print("       1        ");
  delay(1000);
  lcd.setCursor(4,1);
  lcd.print("       ");
  lcd.setCursor(0,0);
  lcd.print("      GO        ");
```

```
  lcd.setCursor(4,1);
  lcd.print("       ");
  delay(1000);
  lcd.setCursor(0,0);
  lcd.print("Level: ");
  lcd.setCursor(7,0);
  lcd.print(level);
  lcd.setCursor(0,1);
  lcd.print("Score: ");
  lcd.setCursor(7,1);
  lcd.print(score);

  for (int i = 2; i<9; i++){
    digitalWrite(i,LOW);
  }
  digitalWrite(dataPin5,LOW);
  digitalWrite(CLR,HIGH);

}

void loop() {
    if (terminate==1) {
      digitalWrite(CLR,LOW);
      lcd.setCursor(0,0);
      lcd.print("   GAME OVER!");
      lcd.setCursor(0,1);
      lcd.print("Score:");
      lcd.setCursor(6,1);
      lcd.print(score);
      lcd.setCursor(9,1);
      lcd.print("Level:");
      lcd.setCursor(15,1);
      lcd.print(level);
    }else if (terminate==0){
      for (int j = 0; j < 14; j++){
```

```
shiftOut(clockPin,MSBFIRST,speed,dataPin6,bits6[j],dataPin5,bits5[j],dataPin4,bits4[j],dataPin3,bits3[j],dataPin
2,bits2[j],dataPin1,bits1[j]);
      }
      speed=speed - speed*0.25;

      if (level < winLevel) {
        level++;
        lcd.setCursor(7,0);
        lcd.print(level);
        lcd.setCursor(8,0);
        lcd.print(" ");
      }
      else if (level >= winLevel) {
        digitalWrite(CLR, LOW);
        lcd.setCursor(0,0);
        lcd.print("   YOU WON!");
        lcd.setCursor(0,1);
        lcd.print("Score:");
        lcd.setCursor(6,1);
        lcd.print(score);
        lcd.setCursor(9,1);
        lcd.print("Level:");
        lcd.setCursor(15,1);
        lcd.print(level);
      }
    }

}
void shiftOut(int clockPin,int bitOrder,int speed,int dataPin1,byte val1,int dataPin2,byte val2,int dataPin3,byte
val3,int dataPin4,byte val4,int dataPin5,byte val5,int dataPin6,byte val6)
{
        digitalWrite(clockPin, LOW);
        if (terminate==0){
          for (int i = 0; i < 8; i++)  {
            if (bitOrder == LSBFIRST){
                  digitalWrite(dataPin6, !!(val6 & (1 << i)));
```

```
      digitalWrite(dataPin5, !!(val5 & (1 << i)));
      digitalWrite(dataPin4, !!(val4 & (1 << i)));
      digitalWrite(dataPin3, !!(val3 & (1 << i)));
      digitalWrite(dataPin2, !!(val2 & (1 << i)));
      digitalWrite(dataPin1, !!(val1 & (1 << i)));

 }
 else {
      digitalWrite(dataPin6, !!(val6  & (1 << (7 - i))));
      digitalWrite(dataPin5, !!(val5 & (1 << (7 - i))));
      digitalWrite(dataPin4, !!(val4 & (1 << (7 - i))));
      digitalWrite(dataPin3, !!(val3 & (1 << (7 - i))));
      digitalWrite(dataPin2, !!(val2 & (1 << (7 - i))));
      digitalWrite(dataPin1, !!(val1 & (1 << (7 - i))));
 }
digitalWrite(clockPin, HIGH);
digitalWrite(clockPin, LOW);

if (digitalRead(crash) == HIGH) {
   digitalWrite(CLR,LOW);
   lcd.setCursor(0,0);
   lcd.print("   GAME OVER!");
   lcd.setCursor(0,1);
   lcd.print("Score:");
   lcd.setCursor(6,1);
   lcd.print(score);
   lcd.setCursor(9,1);
   lcd.print("Level:");
   lcd.setCursor(15,1);
   lcd.print(level);
   terminate = 1;
   break;
}
delay(speed);

if (level < winLevel) {
 lcd.setCursor(6,1);
```

```
            lcd.print(score);
            score++;
           }else if (level >= winLevel) {
            digitalWrite(CLR, LOW);
            lcd.setCursor(0,0);
            lcd.print("   YOU WON!");
            lcd.setCursor(0,1);
            lcd.print("Score:");
            lcd.setCursor(6,1);
            lcd.print(score);
            lcd.setCursor(9,1);
            lcd.print("Level:");
            lcd.setCursor(15,1);
            lcd.print(level);
          }
         }
        }
}
```