INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INTRODUCCIÓN A LOS MICROCONTROLADORES

Práctica 20
Proyecto final: Juego de Ping-Pong

Profesor: Aguilar Sánchez Fernando

Alumno: Domínguez Reyes Jesús Alejandro

Grupo: 3CM16

Fecha de Entrega: 10 de Junio de 2021

Objetivo

Al término de este semestre los alumnos tendrán la capacidad para diseñar y elaborar un proyecto final.

Introducción Teórica

El uso de matrices LED se encuentra principalmente en las calles, con letreros que indican algo en las vialidades o con letreros que informan sobre algún establecimiento, sin embargo, ¿cómo se utilizan estas matrices LED?, en esta práctica se utilizará una para programar un juego de ping pong, particularmente se utilizará una matriz de 8x8, además se utilizará un display de 7 segmentos de 2 dígitos para poder visualizar el puntaje de los jugadores, cada jugador contará con 2 push button que servirán para mover su raqueta respectiva hacia la izquierda o hacia la derecha.

Matriz LED

Una matriz de LED es un display formado por múltiples LED en distribución rectangular, estos LED están cableados de forma conjunta por filas y columnas, por ello aplicar valores de voltaje a varias filas o columnas simultáneamente encenderá los LED de las intersecciones, lo que hace imposible generar gráficos complejos.
Para poder mostrar algo en esta matriz de LED se utiliza un barrido por filas o columnas, si esto se hace a una frecuencia elevada será imposible para el ojo humano notar este barrido, este efecto es denominado “Persistencia de visión” (POV) y es consecuencia de la forma en que los humanos percibimos el movimiento.

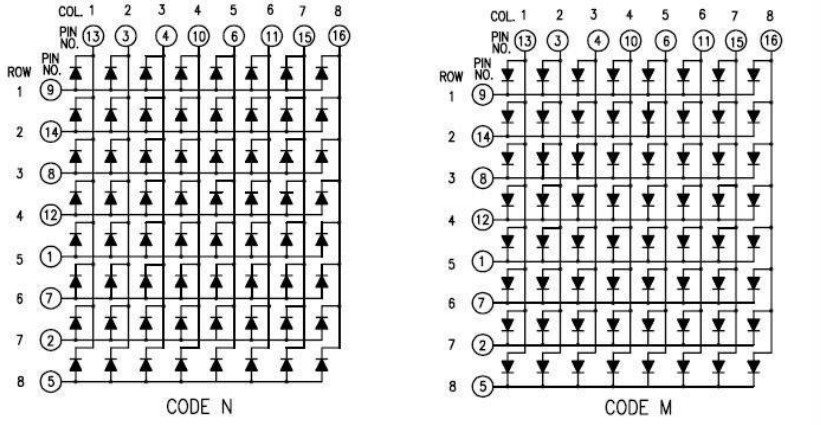

Figura 1. Matriz LED de 8x8.

Las matrices LED pueden ser de tipo ánodo o cátodo comunes:

- Matriz cátodo común: los terminales negativos (cátodos) de todos los LED de cada fila están conectados juntos. Lo mismo ocurre con los pines positivos (ánodos) de cada columna.
- Matriz ánodo común: las conexiones son contrarias, es decir, los ánodos se conectan a las filas y los cátodos a las columnas.

Estos display cuentan con diversos pines, dependiendo de las dimensiones de la matriz, en este proyecto se utilizará una de 8 filas x 8 columnas, por lo que cuenta con 16 pines para su control.

Display de 7 segmentos 2 dígitos

Es un display de 7 segmentos que permite visualizar 2 dígitos, para hacerlo funcionar se hace uso de un barrido que si es lo suficientemente rápido el ojo humano no será capaz de percibir, con este barrido se seleccionará el display en el que se quiere mostrar una salida, la utilidad de este tipo de display es que utiliza menos pines para funcionar que un display de 7 segmentos de un solo digito, evitando duplicar el número de pines que necesitarían como dos displays de este último tipo, existen tanto de cátodo común como de ánodo común.

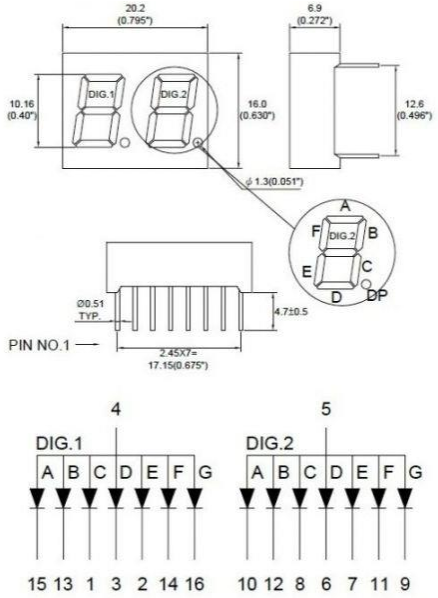

Figura 2. Display de 7 segmentos de 2 dígitos.

Push Button

Un push button es un dispositivo electrónico utilizado para ser pulsado con un dedo y permitir el flujo de corriente mientras son accionados, cuando se deja de presionar vuelve a su posición original.

Figura 3. Imagen de un Push button.

ATmega8535/8535L

El ATmega 8535 y 8535L son los microcontroladores que nos servirán en esta práctica, su diferencia es que el segundo puede trabajar con una señal de reloj de los 0 a los 8 MHz, y el segundo con una señal de los 0 a los 16 MHz.

Algunas de sus características más importantes son las siguientes:

- Microcontrolador AVR de 8 bits de alto rendimiento y bajo consumo.
- Arquitectura RISC avanzada de 130 instrucciones, y cuenta con 32 registros de trabajo de propósito general.
- Memorias de programa y de datos no volátiles, cuenta con una FLASH de 8K bytes autoprogramable en sistema y una EEPROM de 512 bytes.
- Características de los periféricos.
    - 2 Timer/Contadores de 8 bits con prescaler separado y modo comparación.
    - 1 Timer/Contador de 16 bits con prescaler separado, modo comparación y modo de captura.
    - Comparador analógico On-Chip.
    - Timer watchdog programable con oscilador separado On-Chip.
    - Contador en tiempo real con oscilador separado.
    - ADC de 10 bit y 8 canales.
    - 4 canales de PWM.
- El ATmega8535L trabaja de 2.7-5.5V y el ATmega8535 de 4.5-5.5V.

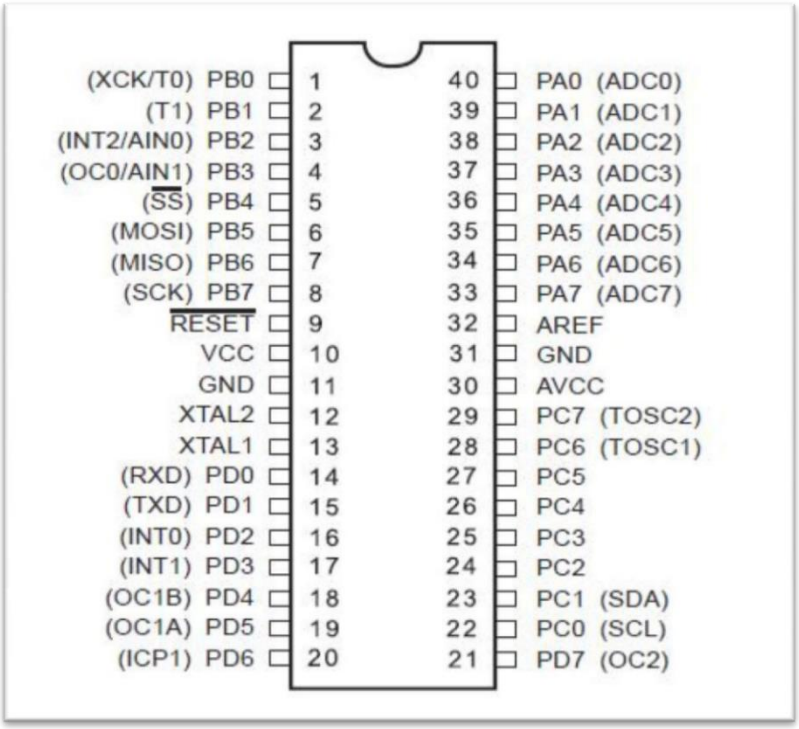

Figura 4. Pines del ATmega8535/8535L.

Materiales y Equipo empleado

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 1 Display de 7 segmentos de 2 dígitos
- ✓ 1 Matriz LED de 8x8
- ✓ 15 Resistores de 330 Ω
- ✓ 4 Push Button

Desarrollo Experimental

1.- Diseñe un Juego de Ping-Pong con las siguientes características armando su circuito final en “PLACA”

a) Use una matríz de leds de 7x5 o de 8x8.
b) En la matriz de leds la pelota rebotará en las orillas y la raqueta estará formada por 2 puntos en la base de la matriz.
c) Se marcará un punto en el display de 7 segmentos por cada pelota que el jugador no alcance con la raqueta.
d) Los push button sirven para mover la raqueta de derecha a izquierda y viceversa.
e) Para la entrega del Proyecto se debe anexar un informe en el que debe incluir su diseño, diagramas eléctricos y código del programa aplicado.

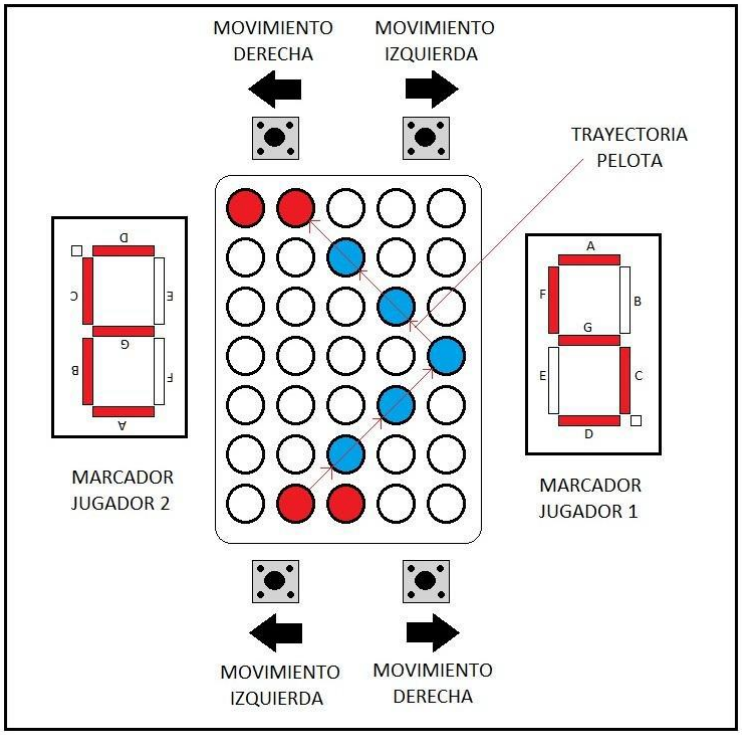

Figura 5. Circuito para el ping-pong.

A continuación, se muestra una captura de pantalla de la simulación realizada, se muestran un par de displays, en los que se tiene a la izquierda el puntaje del jugador 1 y a la derecha el puntaje del jugador 2, en la captura también apreciamos la matriz de 8x8 leds que muestra el tablero de juego, a modo de una comprensión mas sencilla en la parte superior de la matriz se observan los controles del jugador 1, que manipulará raqueta superior de la matriz, mientras que en la parte inferior de la misma se encuentran los controles del jugador 2, que podrán manipular la raqueta mostrada en la parte inferior, en el centro del tablero observamos un led encendido, que corresponde a la pelota del juego, también se logra apreciar en la captura el microcontrolador y los resistores utilizados.

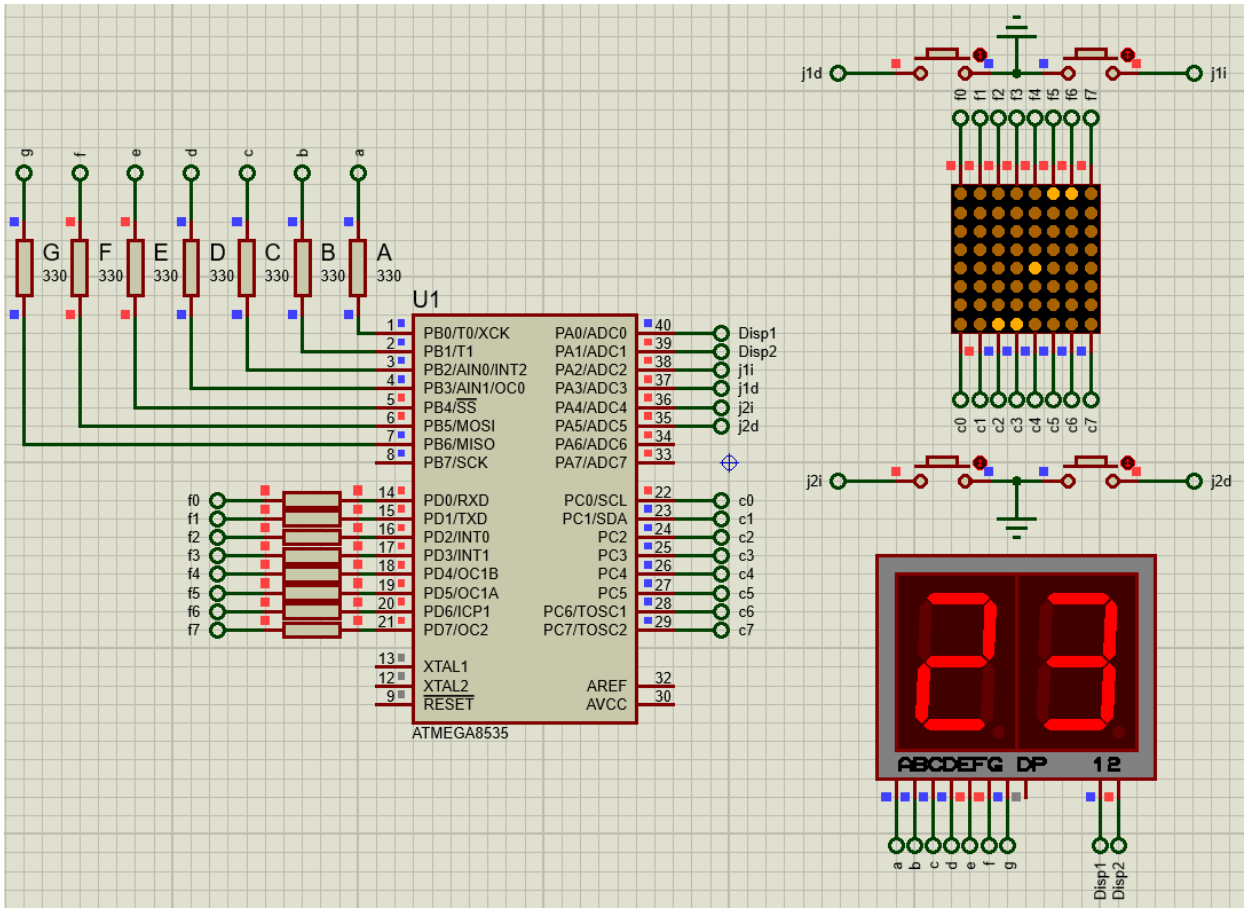

Figura 6. Simulación del circuito.

Código del programa en lenguaje C

En esta sección se muestra el código escrito en CodeVisionAVR.

```
/*****************************************************
This program was created by the CodeWizardAVR V3.43
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 09/06/2021
Author  :
Company :
Comments:


Chip type               : ATmega8535
Program type            : Application
AVR Core Clock frequency: 1.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <mega8535.h>
#include <delay.h>

// Declare your global variables here
unsigned char
j1I,j1D,j2I,j2D,puntos1,puntos2,posRaq1,posRaq2,j1Ia,j1Ip,j1Da,j1Dp,j2Ia,j2Ip,j2Da,j2Dp;
unsigned char tablero[8];
bit iniciar;
signed char trayX,trayY,pelotaX,pelotaY;
const char tabla7segmentos [11]={0xbf,0x86,0xdb,0xcf,0xe6,0xed,0xfc,0x87,0xff,0xbe,0xb8};

void limpiar_tablero(){
    int i;
    for(i=0;i<8;i++){
        tablero[i] = 0xff;
    }
}

void repaint(){
        int t=1,i;
        unsigned char aux=1;
        for(i=0;i<8;i++){
            PORTC=aux;
            PORTD=tablero[i];
            delay_ms(t);
            aux = aux<<1;
        }
}

void apagar(int x, int y){
    unsigned char aux=1;
    aux=aux<<y;
    tablero[x] = tablero[x]|aux;
}

void encender(int x, int y){
    unsigned char aux=1;
    aux=~(aux<<y);
    tablero[x] = tablero[x]&aux;
}

void raqueta_on(int numRaq, int i){
    if(numRaq==2){
        encender(i,7);
```

```
        encender(i+1,7);
    }else{
        encender(7-i,0);
        encender(6-i,0);
    }
}

void raqueta_off(int numRaq, int i){
    if(numRaq==2){
        apagar(i,7);
        apagar(i+1,7);
    }else{
        apagar(7-i,0);
        apagar(6-i,0);
    }
}

void raquetad(int numRaq){
    if(numRaq==2){
        raqueta_off(2,posRaq2);
        if(posRaq2<6)
            posRaq2++;
        raqueta_on(2,posRaq2);
    }else{
        raqueta_off(1,posRaq1);
        if(posRaq1<6)
            posRaq1++;
        raqueta_on(1,posRaq1);
    }
}

void raquetai(int numRaq){
    if(numRaq==2){
        raqueta_off(2,posRaq2);
        if(posRaq2>0)
            posRaq2--;
        raqueta_on(2,posRaq2);
    }else{
        raqueta_off(1,posRaq1);
        if(posRaq1>0)
            posRaq1--;
        raqueta_on(1,posRaq1);
    }
}

void avanzar_pelota(){
    apagar(pelotaX,pelotaY);
    pelotaX+=trayX;
    pelotaY+=trayY;
    encender(pelotaX,pelotaY);
}

unsigned char isOn(int x, int y){
    unsigned char aux=1;
    aux=aux<<y;
    aux = tablero[x]&aux;
    if(aux==0){
        aux = 1;
    }else{
        aux = 0;
    }
    return aux;
}

void ganador(int num){
    limpiar_tablero();
    if(num==1){
        tablero[0] = ~0x81;
        tablero[1] = ~0x81;
        tablero[2] = ~0x81;
        tablero[3] = ~0xff;
```

```
        tablero[4] = ~0xff;
        tablero[5] = ~0x81;
        tablero[6] = ~0x81;
        tablero[7] = ~0x81;
    }else{
        tablero[0] = ~0x81;
        tablero[1] = ~0xff;
        tablero[2] = ~0xff;
        tablero[3] = ~0x81;
        tablero[4] = ~0x81;
        tablero[5] = ~0xff;
        tablero[6] = ~0xff;
        tablero[7] = ~0x81;
    }
}

void reiniciar_tablero(int i){
    int j;
    if(i==2){
        pelotaX=3;
        pelotaY=6;
    }else{
        pelotaX=3;
        pelotaY=1;
    }
    for(j=0;j<8;j++){
        apagar(j,0);
    }
    for(j=0;j<8;j++){
        apagar(j,7);
    }
    posRaq1 = 3;
    posRaq2 = 3;
    raqueta_on(1,posRaq1);
    raqueta_on(2,posRaq2);
    iniciar=0;
    encender(pelotaX,pelotaY);
}

void reiniciar_juego(int i){
    puntos1=0;
    puntos2=0;
    reiniciar_tablero(i);
}

void rebote_pelota(int i){
    if(isOn(pelotaX,i)>0){
        trayY=-1*trayY;
    }else if(trayX>0){
        if(isOn(pelotaX+1,i)>0){
            trayX=-1*trayX;
            trayY=-1*trayY;
        }
    }else{
        if(isOn(pelotaX-1,i)>0){
            trayX=-1*trayX;
            trayY=-1*trayY;
        }
    }
}

void main(void)
{
// Declare your local variables here
int cont=0;
puntos1 = 0;
puntos2 = 0;
posRaq1 = 3;
posRaq2 = 3;
trayX=-1;
trayY=1;
```

```
pelotaX = 3;
pelotaY = 1;
iniciar = 0;

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=Out Bit0=Out
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (1<<DDA1) |
(1<<DDA0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=0 Bit0=0
PORTA=(1<<PORTA7) | (1<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (1<<PORTA3) | (1<<PORTA2) |
(0<<PORTA1) | (0<<PORTA0);

// Port B initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1) |
(1<<DDB0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) |
(0<<PORTB1) | (0<<PORTB0);

// Port C initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<DDC1) |
(1<<DDC0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) |
(0<<PORTC1) | (0<<PORTC0);

// Port D initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1) |
(1<<DDD0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) |
(0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) |
(0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
```

```
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);
TCNT2=0x00;
OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);

// USART initialization
// USART disabled
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8)
| (0<<TXB8);

// Analog Comparator initialization
// Analog Comparator: Off
// The Analog Comparator's positive input is
// connected to the AIN0 pin
// The Analog Comparator's negative input is
// connected to the AIN1 pin
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);

// ADC initialization
// ADC disabled
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1)
| (0<<ADPS0);

// SPI initialization
// SPI disabled
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
(0<<SPR0);

// TWI initialization
// TWI disabled
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

limpiar_tablero();

raqueta_on(1,posRaq1);
raqueta_on(2,posRaq2);
encender(pelotaX,pelotaY);

while (1)
      {
      // Place your code here

//////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Leemos los botones

//////////////////////////////////////////////////////////////////////////////////////////
/////////////

      j1I=PINA&0b00000100;
      j1D=PINA&0b00001000;
      j2I=PINA&0b00010000;
      j2D=PINA&0b00100000;
```

```
////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Eliminamos rebotes y leemos por flancos

////////////////////////////////////////////////////////////////////////////////////////////
/////////////

      if(j1I==0)
        j1Ia=0;
      else
        j1Ia=1;
      if((j1Ip==1)&&(j1Ia==0)) //hubo cambio de flanco de 1 a 0
      {
        raquetai(1);
        if(iniciar==0&&(isOn(3,1)>0)){
            trayX=1;
            trayY=1;
            iniciar=1;
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
      }
      if((j1Ip==0)&&(j1Ia==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes

      if(j1D==0)
        j1Da=0;
      else
        j1Da=1;
      if((j1Dp==1)&&(j1Da==0)) //hubo cambio de flanco de 1 a 0
      {
        raquetad(1);
        if(iniciar==0&&(isOn(3,1)>0)){
            trayX=-1;
            trayY=1;
            iniciar=1;
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
      }
      if((j1Dp==0)&&(j1Da==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes

      if(j2I==0)
        j2Ia=0;
      else
        j2Ia=1;
      if((j2Ip==1)&&(j2Ia==0)) //hubo cambio de flanco de 1 a 0
      {
        raquetai(2);
        if(iniciar==0&&(isOn(3,6)>0)){
            trayX=-1;
            trayY=-1;
            iniciar=1;
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
      }
      if((j2Ip==0)&&(j2Ia==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes

      if(j2D==0)
        j2Da=0;
      else
        j2Da=1;
      if((j2Dp==1)&&(j2Da==0)) //hubo cambio de flanco de 1 a 0
      {
        raquetad(2);
        if(iniciar==0&&(isOn(3,6)>0)){
            trayX=1;
            trayY=-1;
            iniciar=1;
        }
```

```
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes
      }
      if((j2Dp==0)&&(j2Da==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar rebotes


////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Contador que nos sirve para retardar el movimiento de la pelota

////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      cont++;
      if(cont==40 && iniciar==1){
        avanzar_pelota();
        cont=0;
        if(pelotaX==0 || pelotaX==7){
            trayX=-1*trayX;
        }

        if(pelotaY==6&&trayY>0){
            rebote_pelota(7);
        }else if(pelotaY==1&&trayY<0){
            rebote_pelota(0);
        }

        if(pelotaY==7){
            puntos1++;
            reiniciar_tablero(2);
        }else if(pelotaY==0){
            puntos2++;
            reiniciar_tablero(1);
        }
      }else if(cont==125){
        cont=0;
      }

////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Si hay un ganador con 9 puntos lo informamos

////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      if(puntos1==9){
        ganador(1);
        puntos1 = 9;
        puntos2 = 10;
      }else if(puntos2==9){
        ganador(2);
        puntos1 = 10;
        puntos2 = 9;
      }

////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Si el ganador lo considera puede iniciar un nuevo juego

////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      if(puntos1==9&&j1D==0){
        limpiar_tablero();
        reiniciar_juego(1);
      }
      if(puntos2==9&&j2D==0){
        limpiar_tablero();
        reiniciar_juego(2);
      }


////////////////////////////////////////////////////////////////////////////////////////////
/////////////
```

```
      //Dibujamos el tablero

////////////////////////////////////////////////////////////////////////////////////////////
/////////////

      repaint();


////////////////////////////////////////////////////////////////////////////////////////////
/////////////
      //Asignamos valores a los displays

////////////////////////////////////////////////////////////////////////////////////////////
/////////////

      PORTA = 0b11111101;
      PORTB = ~tabla7segmentos[puntos1];
      delay_ms(1);
      PORTA = 0b11111110;
      PORTB = ~tabla7segmentos[puntos2];

      j1Ip=j1Ia;
      j1Dp=j1Da;
      j2Ip=j2Ia;
      j2Dp=j2Da;
      }
}
```

Observaciones y Conclusiones Individuales

Se concluyó que el uso de una matriz LED es bastante similar entre los distintos modelos, se tuvieron algunos problemas para la realización del programa, pero principalmente tuvieron que ver con la cuestión lógica entre la gestión de los distintos LED de la matriz, ya que fue complicado hacer que las raquetas se comportaran de manera adecuada una vez que se tenía la pelota, ya que se buscaban distintas características en su entorno y en ocasiones llegaba a fallar el LED que se debía de apagar, ya que hacia la posición que llegaba la pelota se colocaba la raqueta, sin embargo, se pudo solucionar, una complejidad que se tuvo con Proteus fue que utilizando dos Displays independientes de ánodo común la simulación no lograba mostrar de forma adecuada los puntajes de los jugadores, ya que aunque fuese muy bajo el delay entre cada vez que se mostraban números en uno u otro display resultaba que se lograba apreciar un parpadeo en los mismos, por ello se optó por utilizar un display de 7 segmentos de 2 dígitos, el cual no presentaba esta problemática.

Bibliografía

http://www.sc.ehu.es/sbweb/webcentro/automatica/web_avr/archivos/Otros%20AVRs/ATmega/ATmega8535.htm

https://uelectronics.com/producto/push-boton-2-pines-microswitch-button/

https://www.luisllamas.es/matriz-led-arduino-max7219/

https://programarfacil.com/blog/arduino-blog/matriz-led-arduino-max7219/