Instituto Politécnico Nacional

Escuela Superior de Cómputo

Departamento de Ciencias e Ingeniería de la Computación

Sistemas en Chip

Proyecto Final

"Juego de Ping-Pong"

Profesor: Fernando Aguilar Sánchez

Grupo: 6CM1

Equipo 3

Alumnos:

Ocampo Téllez Rodolfo

Patlani Mauricio Adriana

Ruvalcaba Flores Martha Catalina

Sandoval Hernández Eduardo

Fecha de entrega: 15/01/2023

Objetivo

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

Introducción Teórica

El microcontrolador ATMEGA8535 fabricante ATMEL. En la figura 1 se muestra el microcontrolador ATMEGA8535 la cual maneja datos de 8 bits es decir su bus de datos de 8 bits. Aunque este contiene tres registros los cuales son el *x*, *y* y *z*, los cuales manejan datos de 16 bits.

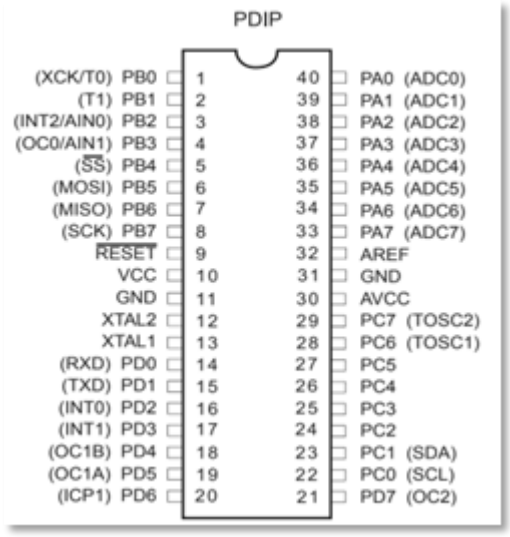

Figura 1. Configuración de pines ATmega8535

El microcontrolador utiliza una arquitectura cerrada, es decir, aquel que es inmodificable por los programadores ajenos a la compañía propietaria del código fuente. Por lo tanto, a este sistema no se le pueden colocar dispositivos periféricos, solo se usa el hardware de la compañía propietaria ya que los dispositivos ajenos a dicha compañía no son compatibles.

El microcontrolador ATMEGA8535 utiliza un encapsulado DIP-40, común en la construcción de circuitos integrados que consiste en un bloque con dos hileras paralelas de pines, observar la figura 2.

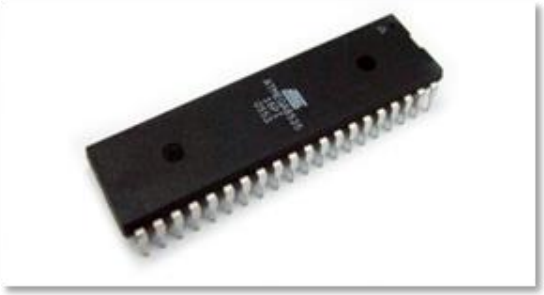

Figura 2. Microcontrolador atmega853.

La comunicación interna del microcontrolador se categoriza en 4 puertos, en la figura 1 se puede analizar la etiquetación de los puertos PA0 al PA7, PB0 al PB7, PC0 al PC7 y PD0 al PD7. Cabe recordar que maneja datos de 8 bits.

El microcontrolador se alimenta de las terminales 10 y 11 como lo muestra la figura 1, los cuales son el VCC (5 V y una tolerancia de ± 0.5V) y GND. Sin embargo, el convertidor se alimenta de forma externa en la terminal 31 y 32, los cuales son GND y AREF.

Posee un oscilador interno de 1MHz, sin embargo, como es un oscilador RC, es susceptible a variar la frecuencia con respecto a las variaciones de temperatura. Por otro lado, puede conectarse un oscilador de 0 a 8 MHz o de 0 a 16 MHz.

Display Cátodo Común

El display de 7 segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9 y o algunas letras. El dispositivo cuenta con 7 leds, uno por cada segmento, a cada uno de estos se le asigna una letra que va de la "a" a la "g" y se encuentran organizados como indica la figura 3.

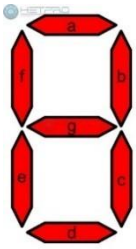

Figura 3. Disposición de los leds en un display de 7 segmentos.

Existen dos tipos de displays de 7 segmentos, de ánodo y cátodo comunes cuya diferencia se encuentra en la forma en que van conectados como lo indica la figura 4.

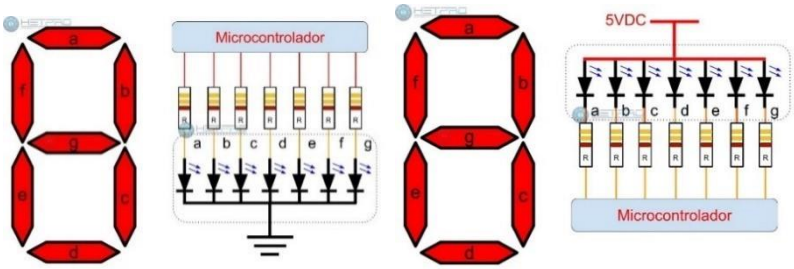

Figura 4. Conexiones en display de 7 segmentos.

Es importante mencionar que los display de 7 segmentos, dado que están construidos con diodos LED, requieren colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED así como de la caída de voltaje. Para calcular la resistencia usamos la Ley de Ohm.

Contador Binario

Un contador electrónico digital es muy útil por ello en la actualidad estamos rodeados de dispositivos que disponen de algún tipo de contador digital, incluso en la mayoría de los electrodomésticos vienen equipados con uno. Nuestro contador digital tiene un campo muy amplio para su aplicación de forma rotacional (cuantas vueltas efectúa un objeto) o de forma secuencial (Ej. en una empresa para el conteo de cajas de producto, etc.)

Este dispositivo permite visualizar la formación numérica de manera ascendiente desde 0 hasta 9 en un display. Consta de pocos componentes electrónicos y es alimentado por una fuente regulable que suministra 5 volts, es ampliamente utilizado a niveles más complejos para fines comerciales en artefactos electrónicos. Contiene circuitos integrados que permiten dicha función.

Matriz de LEDs 8x8

Una matriz LED 8×8 1088AS es un dispositivo que consta de 64 luces led´s agrupados por 8 columnas de 8 led´s cada una, cada led tiene un diámetro de 3mm con dimensiones de 32mm por 32mm. La matriz se divide en columnas y filas, la letra F corresponde para las filas y la letra C para las columnas, tiene 16 pines de los cuales 8 son para las filas y 8 para las columnas.

Esta matriz es muy útil para proyectos electrónicos, cómo lo pueden ser letreros, señalizaciones, indicadores de números y letras. Por su forma de conexión permite controlar cada led de forma individual. Puedes programar esta matriz con la ayuda de las placas de desarrollo Arduino y con un protoboard conectar cada uno de los led de manera independiente para tener control de cada uno, o igual puedes utilizar el circuito integrado MAX7219 para facilitar las conexiones y la programación.

Sus especificaciones y características son:

- Modelo: Matriz LED 8X8 1088AS
- Color: Rojo
- Tipo: Cátodo común
- Voltaje de alimentación: 5V
- Diámetro de cada LED: 3mm
- Tamaño del módulo: 32mm x 32mm x 7mm
- Alta Calidad
- Fácil de instalar
- Rendimiento estable y fiable

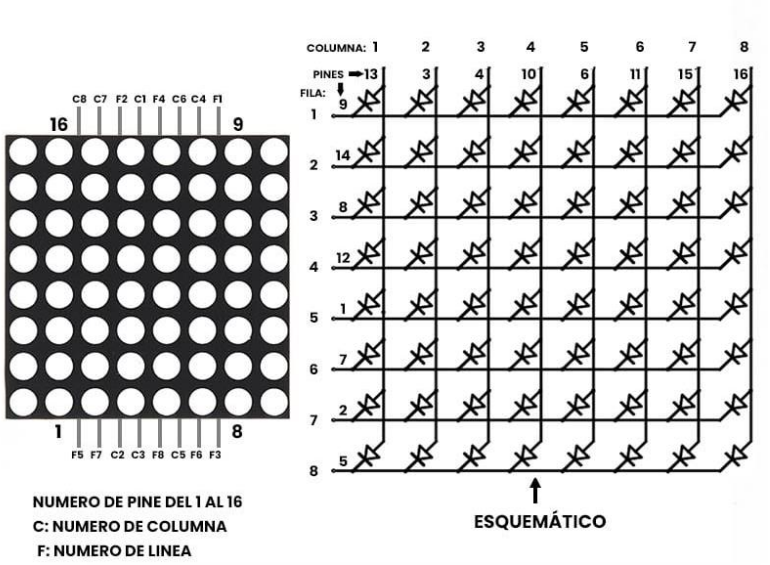

Figura 5. Matriz de LEDs 8x8.

Desarrollo Experimental

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

a) Use una Matriz 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.

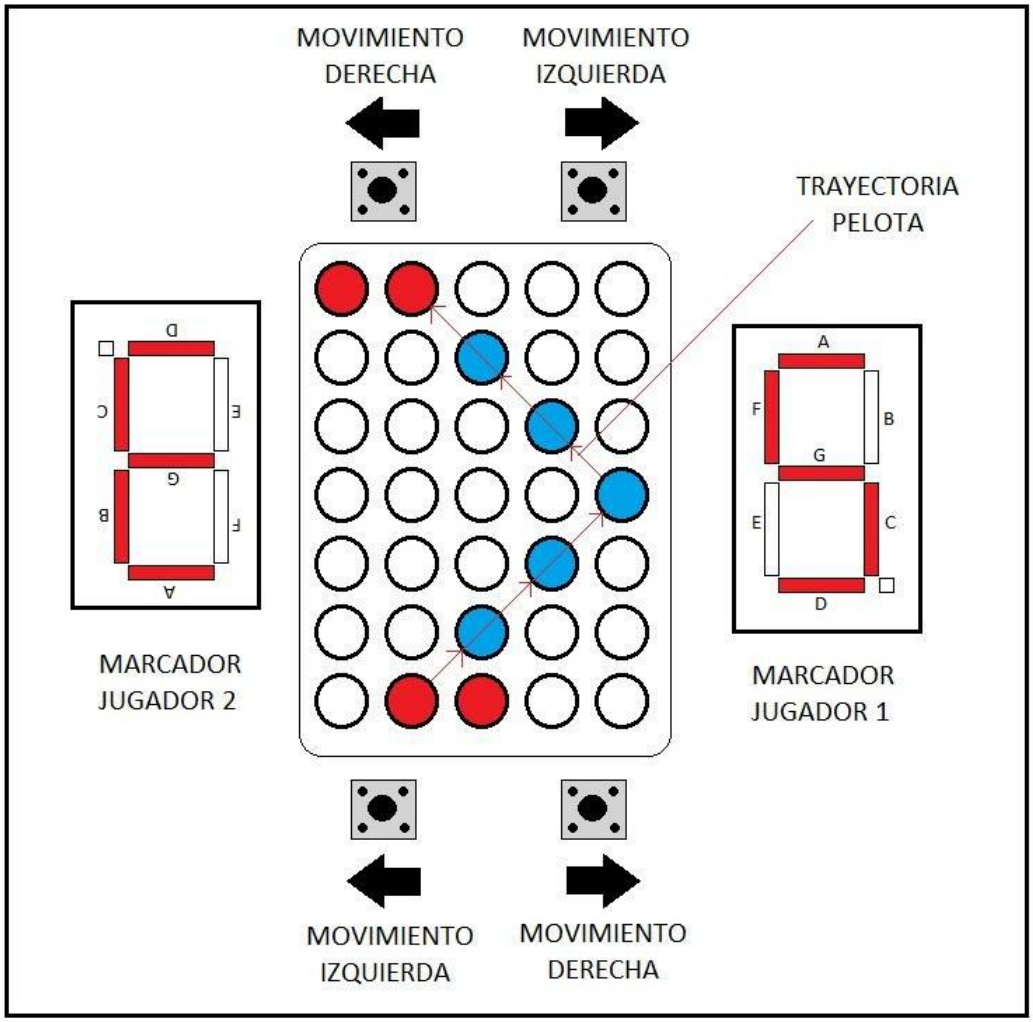

Figura 6. Circuito para el ping-pong.

Diseño del Circuito

A continuación, se muestran los **diagramas del circuito** con los componentes a usar, previo a la realización del proyecto en físico.

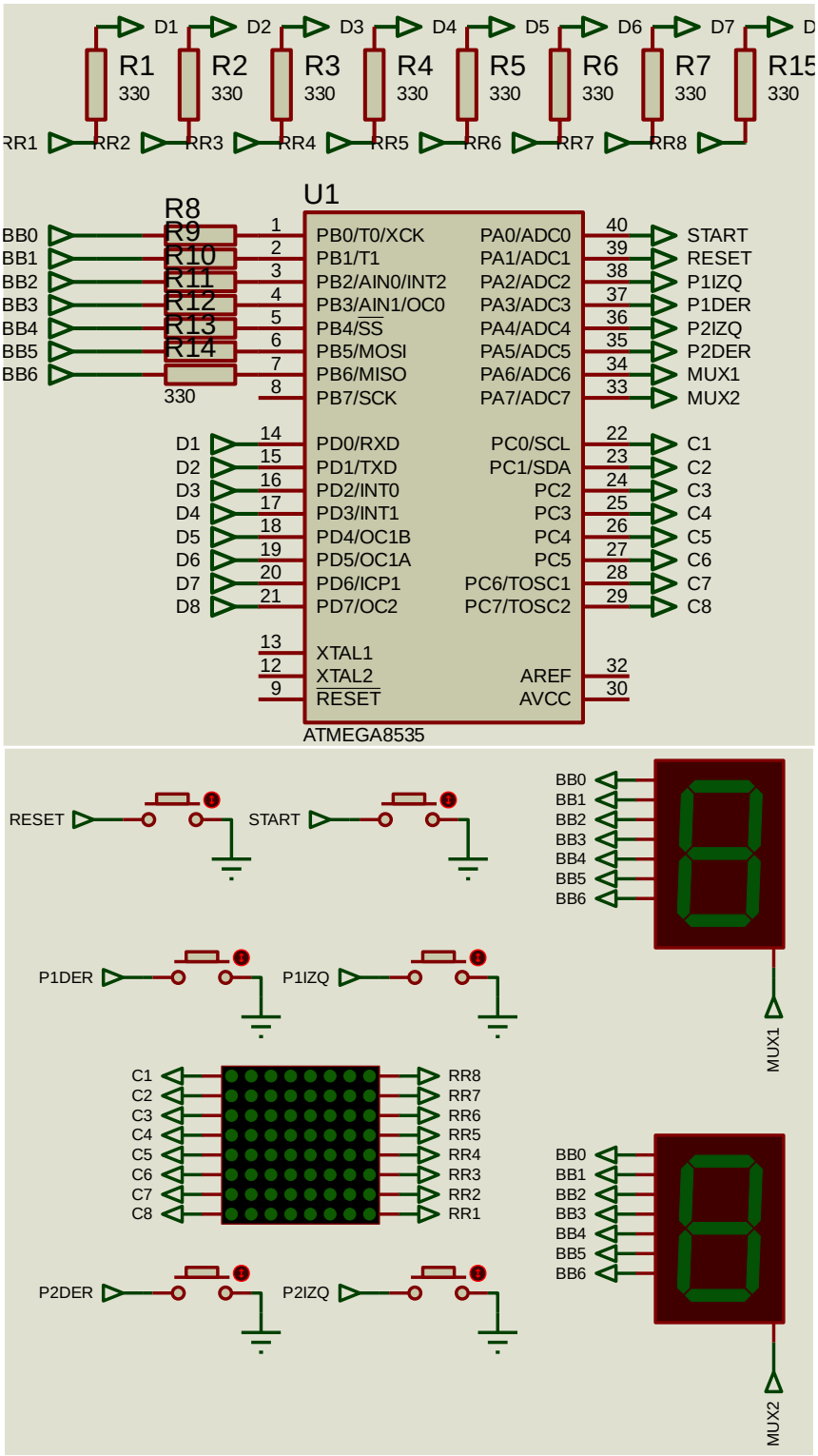

Figura 7. Diagramas del circuito.

Estructura del programa

Código de la configuración de los periféricos utilizados y código del programa principal en C, proporcionados por el IDE de CodeVision.

```
#include <mega8535.h>
#include <delay.h>
#include <stdlib.h>
// Se establecen las dimensiones de la 'cancha': height, weight. Correspondientes a la
matriz
#define H 8
#define W 8

// Cada jugador tiene una posición (column, row)
struct Player {
    unsigned char col;
    unsigned char row;
};

// La pelota tiene posición y desplazamiento
struct Ball {
    int col;
    int row;
    int col_vel;
    int row_vel;
};

// Variables globales
const char display_table[10] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
unsigned char player_1_score = 0;
unsigned char player_2_score = 0;
const int BALL_SPEED = 30;
bit pause = 1;

// Estableciendo los elementos del juego
struct Player player_1;
struct Player player_2;
struct Ball ball;

// Bits para manejar la info de los botones
bit cu_left_1, la_left_1, cu_left_2, la_left_2;
bit cu_right_1, la_right_1, cu_right_2, la_right_2;
bit cu_pause, la_pause;
bit cu_reset, la_reset;

// Funciones
```

```
void set_ball_speed(int);
void set_game();
void paint();
void paint_scoreboard();
void check_changes();
void update_ball();
void check_pause_reset();

int i, j;

void main(void)
{
// Inicialización de puertos de entrada/salida
// Port A initialization
DDRA=(1<<DDA7) | (1<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1)
| (0<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (1<<PORTA3) | (1<<PORTA2)
| (1<<PORTA1) | (1<<PORTA0);
// Port B initialization
DDRB=(1<<DDB7) | (1<<DDB6) | (1<<DDB5) | (1<<DDB4) | (1<<DDB3) | (1<<DDB2) | (1<<DDB1)
| (1<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2)
| (0<<PORTB1) | (0<<PORTB0);
// Port C initialization
DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<DDC1)
| (1<<DDC0);
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2)
| (0<<PORTC1) | (0<<PORTC0);
// Port D initialization
DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1)
| (1<<DDD0);
PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2)
| (0<<PORTD1) | (0<<PORTD0);

// IDE Code
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) |
(0<<CS00);
TCNT0=0x00;
OCR0=0x00;
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;
ASSR=0<<AS2;
TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) |
(0<<CS20);
TCNT2=0x00;
OCR2=0x00;
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) |
(0<<OCIE0) | (0<<TOIE0);
MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) |
(0<<RXB8) | (0<<TXB8);
ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) |
(0<<ACIS0);
SFIOR=(0<<ACME);
ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) |
(0<<ADPS1) | (0<<ADPS0);
SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) |
(0<<SPR0);
TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

// Condiciones inciales
set_game();
set_ball_speed(1);


// Desarrollo del juego, mientras nadie pulso el botón de pausa se continuará hasta que
gane algún jugador.
while (1)
    {
        if (pause == 0) {
            for (j = 0; j < BALL_SPEED; j++) {
                paint();
                check_changes();
            }
            update_ball();
        }
        paint();
        check_pause_reset();
    }
}

// Funcion para modificar el movimiento de la pelota
```

```
void set_ball_speed(int speed){
    ball.col_vel = speed;
    ball.row_vel = speed;
}

// Función para inicializar el juego
void set_game(){
    player_1_score = 0;
    player_2_score = 0;
    player_1.col = 6;
    player_1.row = H - 1;

    pause = 1;

    player_2.col = 0;
    player_2.row = 0;

    ball.col = 3;
    ball.row = 3;
}


// Función que maneja los movimientos permitidos de la pelota
void update_ball() {

    ball.col += ball.col_vel;
    if (ball.col >= W || ball.col < 0) {
        ball.col_vel *= -1;
        ball.col += 2 * ball.col_vel;
    }

    ball.row += ball.row_vel;
    // Si la pelota llega a un límite vertical, se verifica si es un rebote. De no
rebotar, significa que el jugador ha marcado
    if (ball.row == H - 1) {
        if (ball.col == player_1.col || ball.col == player_1.col + 1) {
            ball.row_vel *= -1;
            ball.col_vel *= (1 - 2*(rand() % 2));
            ball.row += ball.row_vel;
        } else {
            player_2_score++;
            ball.col = 3;
            ball.row = 3;
            ball.col_vel = -1;
            ball.row_vel = -1;
        }
```

```
    }
    if (ball.row == 0) {
        if (ball.col == player_2.col || ball.col == player_2.col + 1) {
            ball.row_vel *= -1;
            ball.row += ball.row_vel;
            ball.col_vel *= (1 - 2*(rand() % 2));
        } else {
            player_1_score++;
            ball.col = 4;
            ball.row = 4;
            ball.row_vel = 1;
            ball.col_vel = 1;
        }
    }
    // Se verifica si uno de los jugadores ya ha obtenido más de 9 puntos, de ser así
el juego se reinicia
    if (player_1_score > 9 || player_2_score > 9) {
        set_game();
    }
}

// Funcion para modificar los marcadores
void paint_scoreboard() {
    //MUX
    PORTA.6 = 0;
    PORTA.7 = 1;
    PORTB = display_table[player_2_score];
    delay_ms(1);
    //MUX
    PORTA.6 = 1;
    PORTA.7 = 0;
    PORTB = display_table[player_1_score];
    delay_ms(1);
}

// Funcion para verificar si algún usuario ha presionado un botón de pausa o reinicio
void check_pause_reset() {
    cu_pause = PINA.0;
    cu_reset = PINA.1;

    if (cu_pause == 0 && la_pause == 1) {
        pause = ~pause;
    }
    if (cu_reset == 0 && la_reset == 1) {
        set_game();
    }
```

```
    la_pause = cu_pause;
    la_reset = cu_reset;
    paint_scoreboard();
}

// Función que verifica si los jugadores han presionado botones para desplazarse
void check_changes() {
    cu_left_1 = PINA.2;
    cu_left_2 = PINA.4;
    cu_right_1 = PINA.3;
    cu_right_2 = PINA.5;

    // Verificando el movimiento del jugador 1
    if (cu_left_1 == 0 && la_left_1 == 1) {
        if (player_1.col - 1 >= 0) player_1.col -= 1;
    }
    if (cu_right_1 == 0 && la_right_1 == 1) {
        player_1.col++;
        if (player_1.col > W - 2) player_1.col = W - 2;
    }
    // Verificando el movimiento del jugador 2
    if (cu_left_2 == 0 && la_left_2 == 1) {
        if (player_2.col - 1 >= 0) player_2.col -= 1;
    }
    if (cu_right_2 == 0 && la_right_2 == 1) {
        player_2.col++;
        if (player_2.col > W - 2) player_2.col = W - 2;
    }

    la_left_1 = cu_left_1;
    la_left_2 = cu_left_2;
    la_right_1 = cu_right_1;
    la_right_2 = cu_right_2;
    check_pause_reset();
}

// Función para imprimir el tablero en su estado actual
void paint() {
    const unsigned char columns_on = 0xFF;
    unsigned char curr_row = 0x00;
    for (i = 0; i < W; i++) {
        PORTD = columns_on & ~(1 << i);
        curr_row = 0x00;
        if (i == player_1.col || i == player_1.col + 1) {
            curr_row |= 0x80;
```

```
        }
        if (i == player_2.col || i == player_2.col + 1) {
            curr_row |= 0x01;
        }
        if (i == ball.col) {
            curr_row |= (1 << ball.row);
        }
        PORTC = curr_row;
        delay_ms(1);
    }
}
```

Simulación

Se realizó una simulación en el software Proteus Design Suite, previamente cargado el programa en formato ".hex" obtenido del software CodeVision AVR, el cual es un software para el microcontrolador Microchip AVR y sus contrapartes XMEG, esto antes de proceder a ejecutarlo en el circuito en físico.

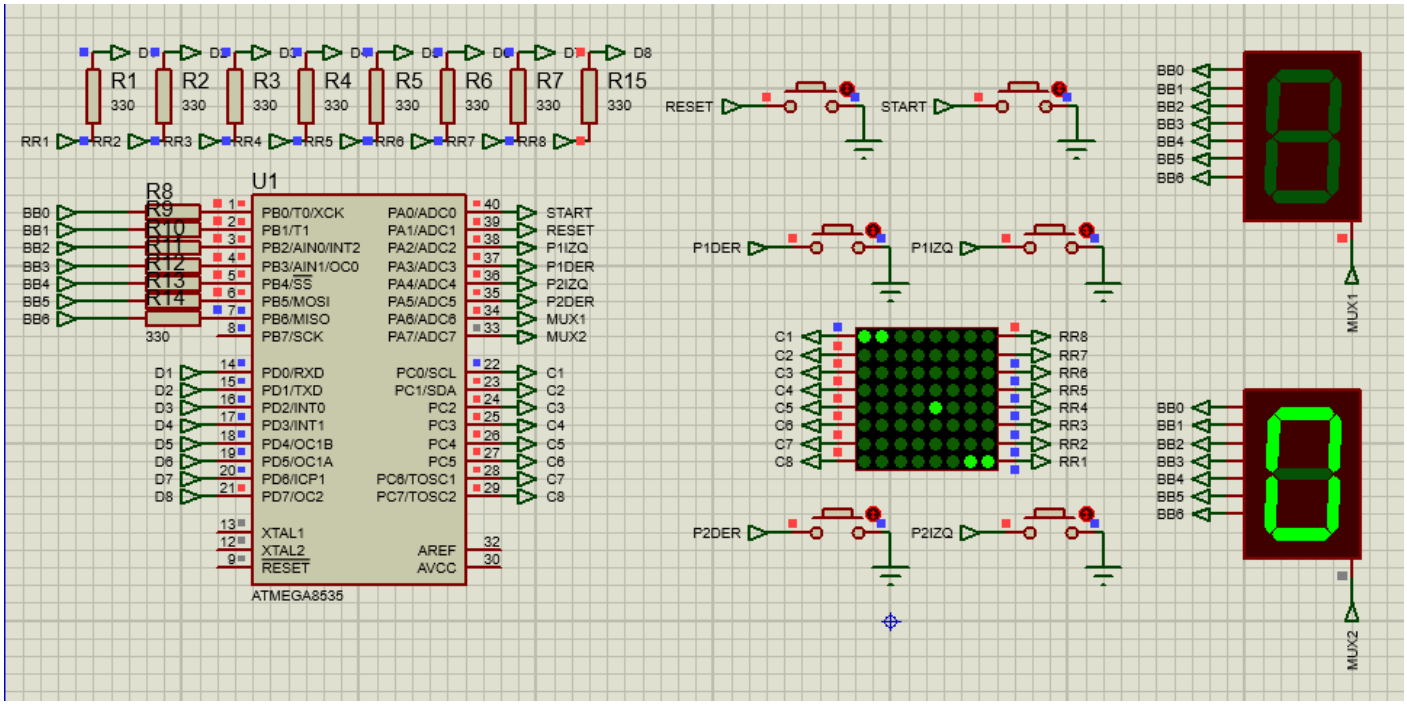

Figura 8. Simulación de la ejecución del programa, estado inicial de un partido de ping-pong, debido a la multiplexación Proteus solo muestra un marcador a la vez.

Fotografía del circuito armado

A continuación, se muestra la figura 9 la evidencia sobre la realización y prueba del proyecto final.

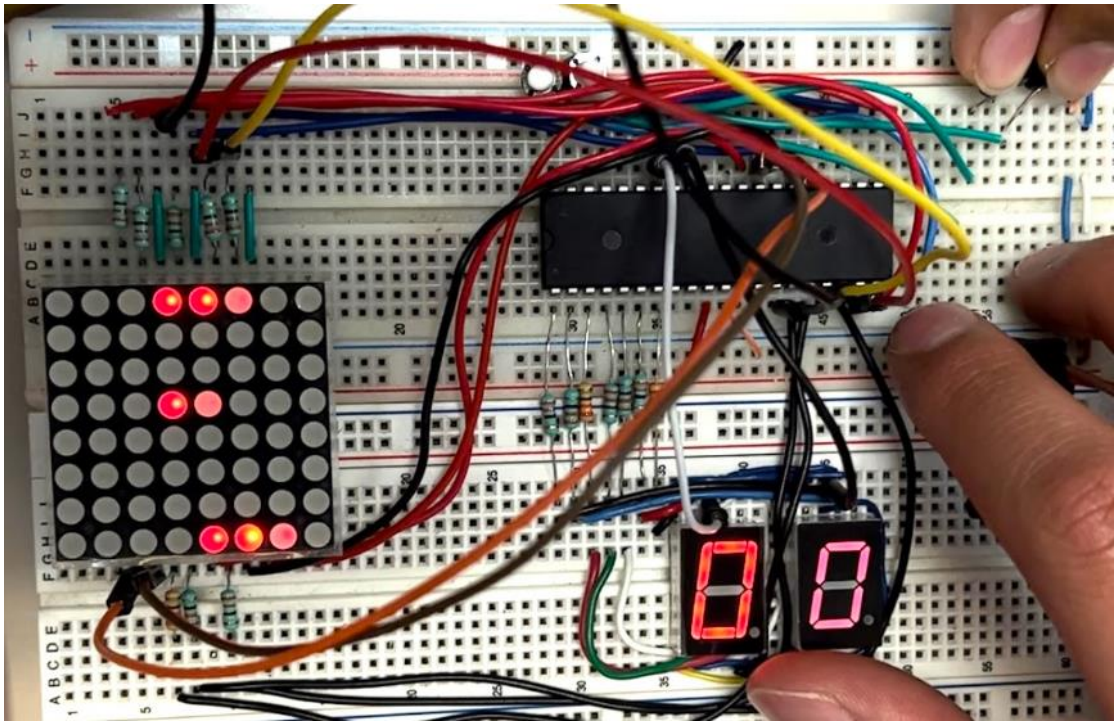

Figura 9. Circuito armado y funcionamiento del Proyecto Final.

Observaciones y conclusiones Individuales

Ocampo Téllez Rodolfo

Al tratarse del proyecto final, se integraron muchas de las cosas realizadas por separado en las anteriores prácticas, como el manejo de displays de 7 segmentos, modificaciones de comportamiento del programa con base a interacciones del usuario, multiplexado para ahorrar uso de puertos y cables, el contador binario sin rebotes, así como una mayor tarea a la hora de realizar toda la programación de funcionalidades que el microcontrolador tenía que ejecutar. Fue un reto interesante desarrollar toda la lógica para obtener un juego de ping-pong funcional, aunado a manejar una matriz de 8x8 por primera vez en un proyecto, y es que este último fue precisamente el punto en que se presentaron los detalles, ya que en el circuito físico la matriz hacía visible un LED de más para los elementos del juego (jugadores y pelota), afortunadamente esto no afectaba realmente la lógica del juego y por tanto se obtuvo un tablero funcional para un partido de ping-pong digital.

Patlani Mauricio Adriana

Como la última práctica, se aplicaron varios conocimientos de prácticas pasadas. En esta práctica se hizo un juego de ping pong usando una matriz de leds como tablero, con un contador de puntos usando displays y de botones para controlar el juego como el inicio. La codificación fue desafiante al igual que el manejo de la matriz de leds de 8x8 ya que mostraba un jugador y una pelota extra; sin embargo, eso no impidió que se mostrara que el proyecto funcionara perfectamente.

Ruvalcaba Flores Martha Catalina

La práctica 20 fue el conjunto de todas las prácticas vistas, puesto que su funcionamiento hace uso de la implementación de una matriz, displays, botones y la programación del microprocesador para que dos usuarios interactúen con la matriz. Sin embargo, debido a la dificultad de la práctica, la implementación de la matriz 8x8 mostraba un LED más en cada jugador y en la pelota, así como se muestra en la fotografía del circuito armado. Finalmente, con esta práctica nos permitió demostrar todos los conocimientos adquiridos durante el curso, demostrando su correcto funcionamiento y armado del circuito.

Sandoval Hernández Eduardo

Esta, al ser la última práctica, fue la más complicada, recurriendo a varias fuentes externas para tener una idea de cómo implementar la práctica, además de que a diferencia de la matriz de 7x5 utilizada en la práctica 17, las conexiones para la matriz de 8x8 fueron más complicadas llegando incluso a cometer varios errores, incluso una vez armado el circuito la matriz no funcionaba de manera adecuada mostrando un punto más por cada jugador y la bola, error que no ocurre en la simulación en proteus pero que en el circuito físico por alguna razón desconocida para nosotros, además de que incluso al mover un poco la matriz esta tendía a fallar. Finalmente, aún con todas las complicaciones se logró el objetivo de implementar un juego de ping pong funcional.

Bibliografía

- Atmel Corporation. (2006). 8-bit Microcontroller with 8K Bytes In-System Programmable Flash. [Online]. Disponible en: https://drive.google.com/file/d/1acpQaDlsyLHr3w3ReSgrlXRbshZ5Z-lb/view

- F. Cereijo. (2004, Diciembre 28). Microcontroladores PIC. [Online]. Disponible en: https://drive.google.com/file/d/1mAydVlkZYqtYXhFQ0Tnr_UjZyjiwIp90/view

- HetPro. (2019). Display 7 Segmentos ánodo y cátodo común. [En línea]. Disponible en https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/

- G. Panton. (2017). Código BCD. [En línea]. Disponible en: https://arquiconsamuel.blogspot.com/2017/08/codigo-bcd.html

- A. Solano. (2014). Contador BCD de 0-9 con temporizador 555 (Automatización): [En línea]. Disponible en: https://www.monografias.com/trabajos102/contador-bcd-0-9-temporizador-555-automatizacion/contador-bcd-0-9-temporizador-555-automatizacion

- UNIT. (s.f.). Matriz LED 8×8 3mm 1088AS. [En línea]. Disponible en: https://uelectronics.com/producto/matriz-8x8-1088as-led/