17-12-2021

Ping Pong

Martínez Coronel Brayan Yosafat

3CM17 Introducción a los microcontroladores

FERNANDO AGUILAR SÁNCHEZ

Objetivo

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

Introducción teórica

Una matriz LED es un display formado por múltiples LED en distribución rectangular. Existen distintos tamaños, siendo el más habitual los cuadrados de 8x8 LED

La matriz está compuesta por una serie de filas y columnas la intersección entre ambas contiene un led, para que este encienda, tiene que recibir simultáneamente un 0 en la fila y un 1 en la columna, cuando se da esta condición la electrónica del circuito se encarga de encender el led correspondiente.

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.

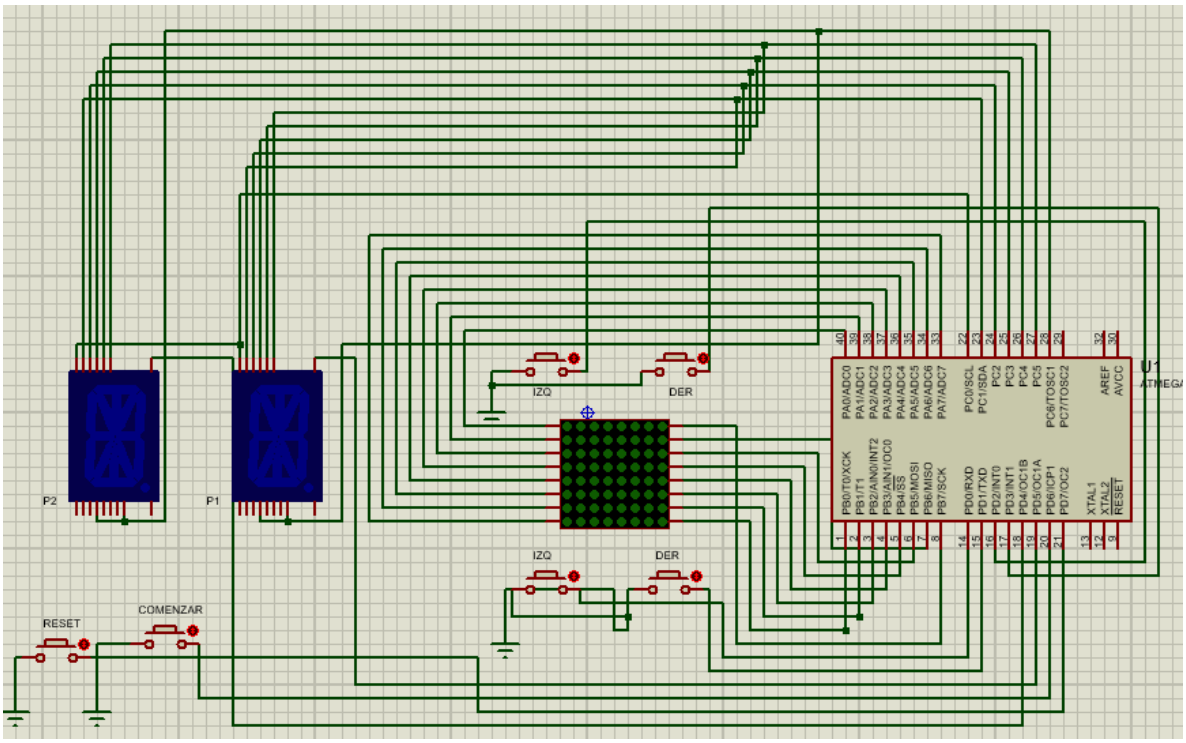
b) En la matriz de leds la pelota rebotará en las orillas y la raqueta estará formada por 2 puntos en la base de la matriz.

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.



Estructura del programa

```
/*****************************************************
This program was created by the CodeWizardAVR V3.45
Automatic Program Generator
© Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    : 16/10/2021
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>
#include <stdlib.h>
#define H 8
#define W 8

struct Player {
    unsigned char col;
    unsigned char row;
};

struct Ball {
    int col;
    int row;
    int col_vel;
    int row_vel;
};

// Declare your global variables here
const char display_table[10] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};
unsigned char player_1_score = 9;
unsigned char player_2_score = 0;
const int BALL_SPEED = 30;
bit pause = 1;

// Players and ball
struct Player player_1;
struct Player player_2;
struct Ball ball;

// For buttons
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;

// Function definitions
void paint();
void paint_scoreboard();
void check_changes();
void update_ball();
void check_pause_reset();

int i, j;


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out
DDRA=(1<<DDA7) | (1<<DDA6) | (1<<DDA5) | (1<<DDA4) | (1<<DDA3) | (1<<DDA2) |
(1<<DDA1) | (1<<DDA0);
// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) |
(0<<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=In Bit6=In Bit5=Out Bit4=Out Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (1<<DDD5) | (1<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=0 Bit4=0 Bit3=P Bit2=P Bit1=P Bit0=P
PORTD=(1<<PORTD7) | (1<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<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);

player_1.col = 6;
player_1.row = H - 1;

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

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

ball.col_vel = 1;
```

```
ball.row_vel = 1;

while (1)
      {
        if (pause == 0) {
            for (j = 0; j < BALL_SPEED; j++) {
                paint();
                check_changes();
                check_pause_reset();
                paint_scoreboard();
            }
            update_ball();
        }
        paint();
        check_pause_reset();
        paint_scoreboard();
      }
}

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;
    // If the ball is in the vertical borders, check if it bounces
    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;
        }
    }

    if (player_1_score > 9 || player_2_score > 9) {
        player_1_score = 0;
        player_2_score = 0;
        player_1.col = 6;
        player_1.row = H - 1;

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

void paint_scoreboard() {
    PORTD = 0xeF;
    PORTC = display_table[player_2_score];
    delay_ms(1);
    PORTD = 0xdF;
    PORTC = display_table[player_1_score];
    delay_ms(1);
```

```
}

void check_pause_reset() {
    cu_pause = PIND.6;
    cu_reset = PIND.7;

    if (cu_pause == 0 && la_pause == 1) {
        pause = ~pause;
    }
    if (cu_reset == 0 && la_reset == 1) {
        pause = 1;
        player_2_score = 0;
        player_1.col = 6;
        player_1.row = H - 1;

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

        player_1_score = 0;
        player_2_score = 0;

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

    la_pause = cu_pause;
    la_reset = cu_reset;
}

void check_changes() {
    cu_left_1 = PIND.0;
    cu_left_2 = PIND.2;
    cu_right_1 = PIND.1;
    cu_right_2 = PIND.3;

    // Checking for player_1 movement
    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;
    }
    // Checking for player 2 movement
    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;
}

void paint() {
    const unsigned char columns_on = 0xFF;
    unsigned char curr_row = 0x00;
    for (i = 0; i < W; i++) {
        PORTB = 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);
        }
        PORTA = curr_row;
```

```
        delay_ms(1);
    }
}
```

Observaciones y Conclusiones

Bueno, fue realmente difícil, como era de esperarse, pero, por fin quedó, no puedo decir que no me halla gustado, es solo que, si tan solo hubiera tenido más tiempo quizá hubiera dedicado mejor la materia, fue muy bueno que no nos presionara, tuve el tiempo para sacar el resto de materias bien, pero, por primera vez, me pregunto si, tal vez, la electrónica es un poco para mí, después de todo este tiempo renegándola, este semestre fue divertido hacer las prácticas.

Bibliografía

Matriz de Leds: https://www.cecyt3.ipn.mx/estudiantes/plan%20continuidad/Archivo%20comprimido13/matriz-leds.pdf