INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 17: MATRIZ 7x5

ALUMNOS: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 18 DE JUNIO DE 2023

OBJETIVO

Al término de la sesión, los integrantes del equipo contaran con la habilidad para manejar una matriz de leds de 7x5.

INTRODUCCIÓN TEÓRICA

MATRIZ DE LEDS

La matriz de LEDs no es más que un arreglo de LEDs agrupados dentro de un encapsulado, los cuales se encuentran agrupados en forma de matriz. Este acomodo nos ayuda para poder generar cualquier cosa que nosotros queramos siempre y cuando se pueda representar dentro de la matriz.[1]

Figura 1. Matriz de leds de 7x5.

La Matriz de LEDs que se usara en este ejemplo es una como la de la foto superior, esta es de 5 columnas por 7 filas, las columnas son representadas por una C y las filas por una R, en la imagen inferior podemos ver como se encuentran distribuidos los pines de la matriz a usar.[1]

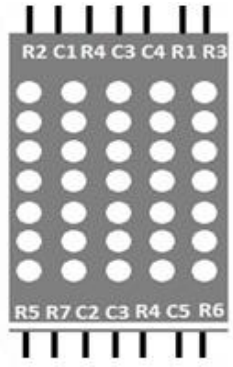


Figura 2. Asignación de pines en matriz de leds de 7x5.

Para poder formar algo en la matriz de led´s, es necesario realizar un barrido en las columnas para controlarlas de manera independiente, cada columna tendrá su código, por lo que debemos formar la figura, numero o letra que necesitemos separando la misma en 5 columnas. A continuación veremos cómo se forma el numero 3 el cual prestando atención al valor de las R's las cuales forman el código deseado mientras que las C's generan un barrido de las columnas. Es importante destacar que en las R's el LED prende con un 0 lógico, lo cual está dado ya que la columna correspondiente está habilitada con un 1 lógico (Vcc), el led en las R's prendera con la diferencia de voltaje, por lo tanto en las R's se usa el 0 como prendido.[1]

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 3 Display Cátodo Común
- ✓ 8 Resistores de 330 Ω a 1⁄4 W
- ✓ 1 Matriz de leds de 7x5

DESARROLLO EXPERIMENTAL

1. Diseñe un programa para visualizar en una matriz de leds de 7x5 los número del 0 al 9 tal y como lo muestra la figura 1.

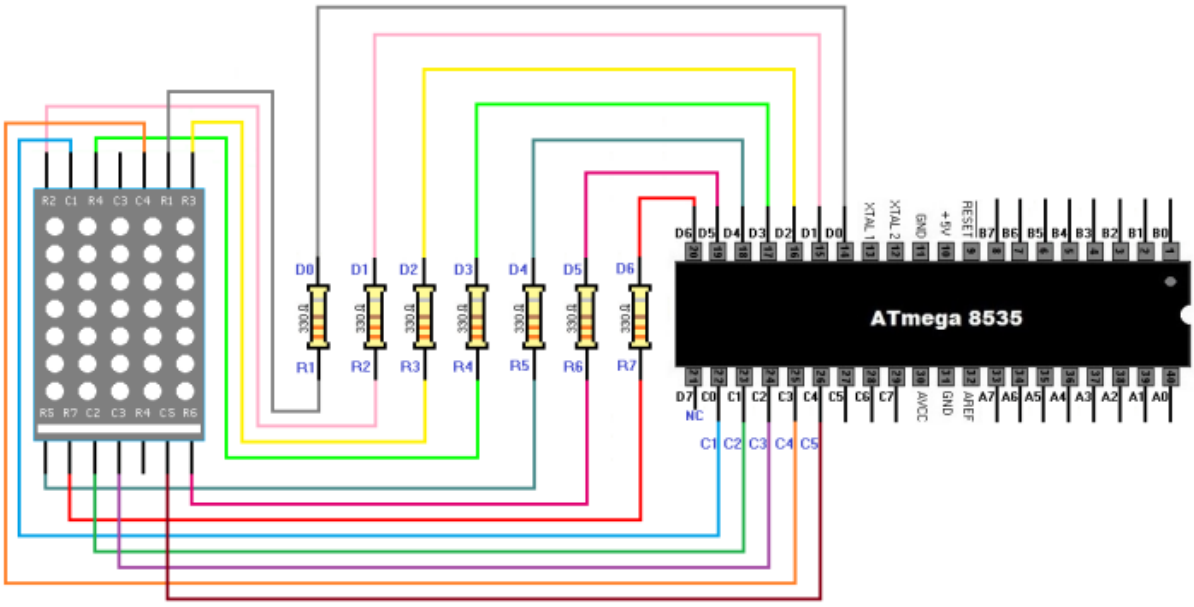


Figura 3. Circuito para conectar las Filas (D0-D7) y las Columnas (C0-C4) de la matriz 7x5.

Matriz de LEDs

La matriz de LEDs no es más que un arreglo de LEDs agrupados dentro de un encapsulado, los cuales se encuentran agrupados en forma de matriz. Este acomodo nos ayuda para poder generar cualquier cosa que nosotros queramos siempre y cuando se pueda representar dentro de la matriz.

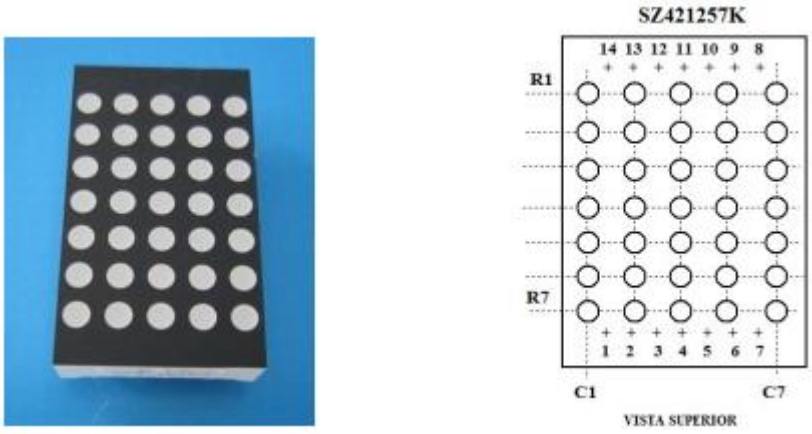


Figura 4. Vista real de la matriz 7x5.

La matriz de LEDs que se usara en este ejemplo es una como la de la foto superior, esta es de 5 columnas por 7 filas, las columnas son representadas por una C y las filas por una R, en la imagen inferior podemos ver como se encuentran distribuidos los pines de la matriz a usar.

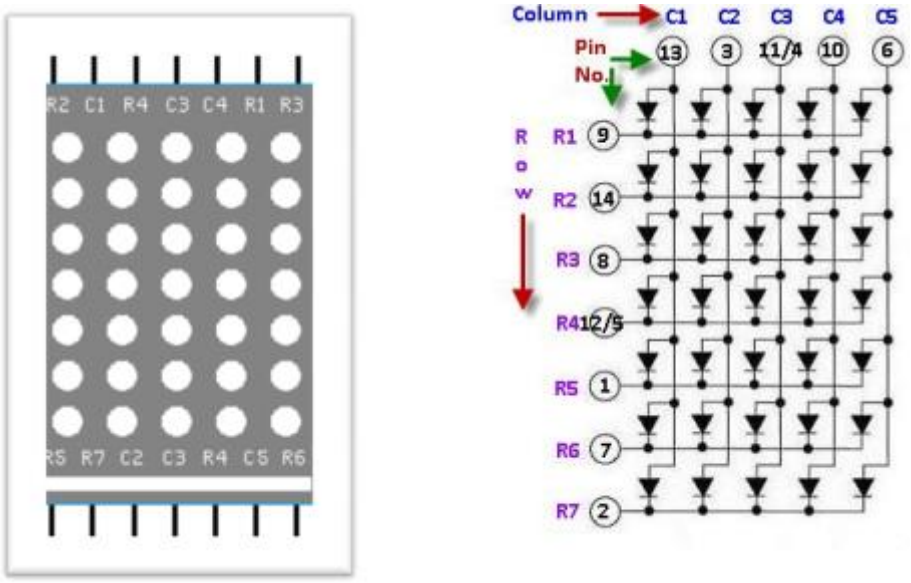


Figura 5. Asignación de pines de la matriz 7x5.

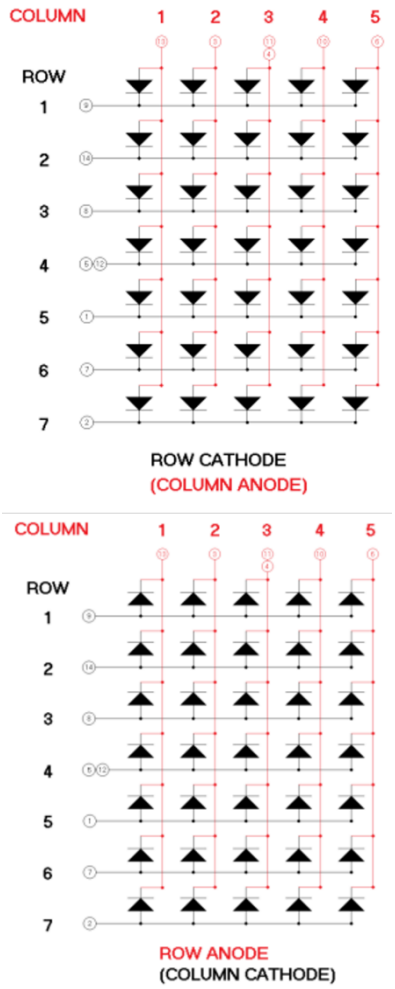


Figura 6. Conexiones para la matriz Fila Cátodo-Columna Ánodo y conexiones para la matriz Fila Ánodo-Columna Cátodo.

Actividad 1

Genere la siguiente secuencia en la matriz para que se observen las columnas como muestra la figura 5.

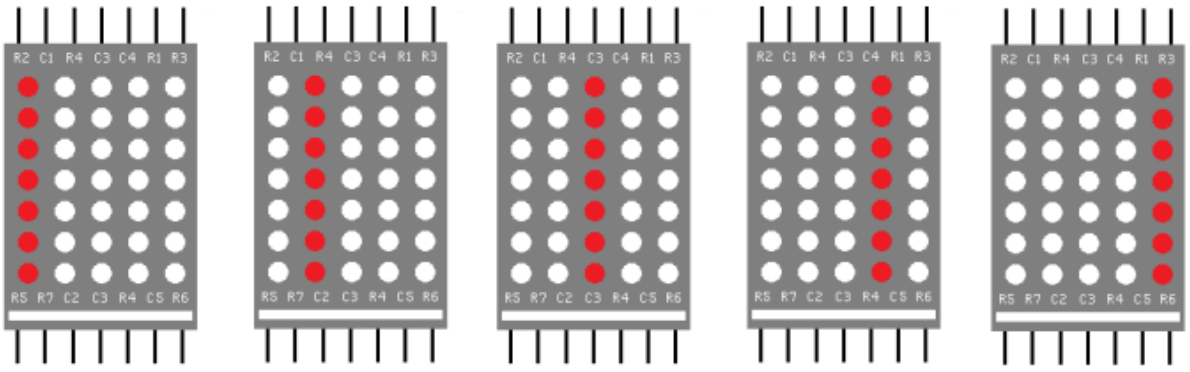


Figura 7. Visualización de las columnas.

Actividad 2

Genere la siguiente secuencia en la matriz para que se observen las filas como muestra la figura 6.

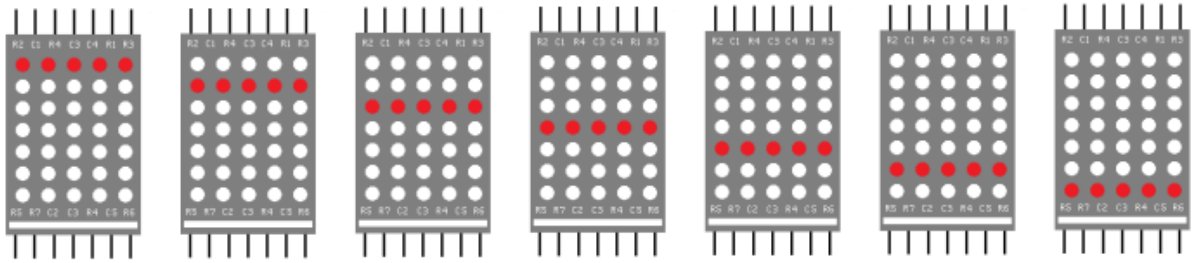


Figura 8. Visualización de las filas.

Actividad 3

Genere la siguiente secuencia en la matriz para que se observen las filas como muestra la figura 7.

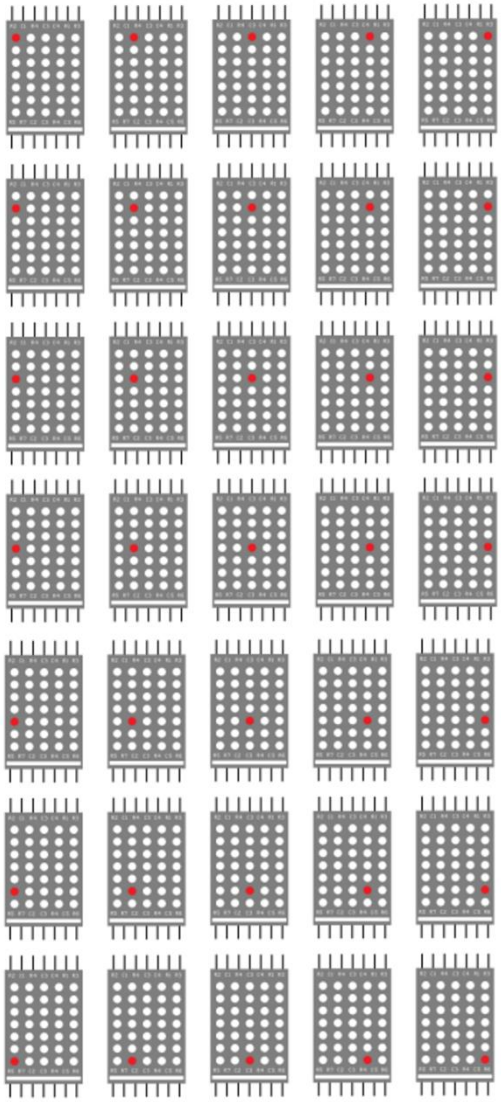
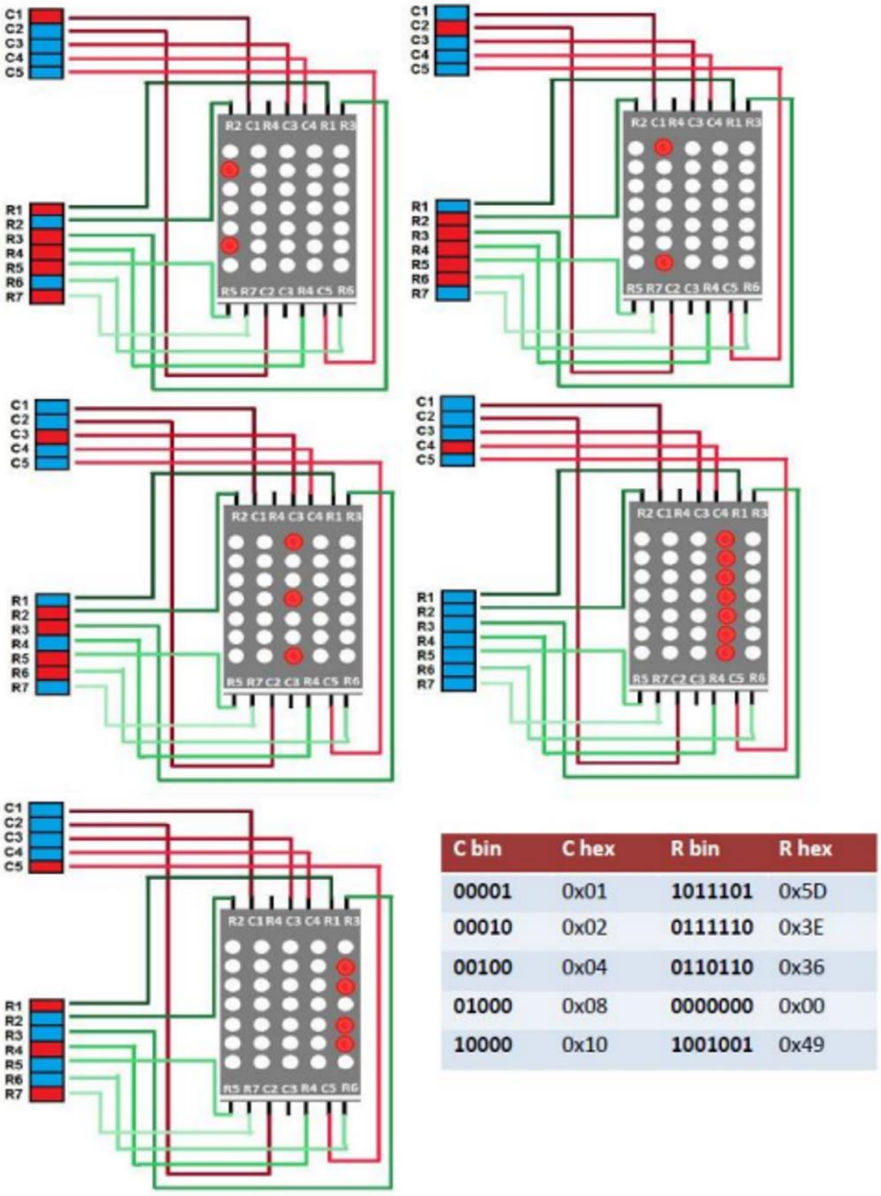


Figura 9. Visualización de todos los LEDS.

Actividad 4

Para poder formar algo en la matriz, es necesario realizar un barrido en las columnas para controlarlas de manera independiente, cada columna tendrá su código, por lo que debemos formar la figura, número o letra que necesitemos separando la misma en 5 columnas. A continuación veremos cómo se forma el numero 3 el cual prestando atención al valor de las R's las cuales forman el código deseado mientras que las C's generan un barrido de las columnas. Es importante destacar que en las R's el LED prende con un 0 lógico, lo cual está dado ya que la columna correspondiente está habilitada con un 1 lógico (Vcc), el led en las R's prendera con la diferencia de voltaje, por lo tanto en las R's se usa el 0 como prendido.

Ejemplo de la formación de un número 3.



C bin	C hex	R bin	R hex
00001	0x01	1011101	0x5D
00010	0x02	0111110	0x3E
00100	0x04	0110110	0x36
01000	0x08	0000000	0x00
10000	0x10	1001001	0x49

Como se puede ver en la imagen anterior, el número 3 se formó en base a la combinación de controlar las C's y las R's, trabajo que le asignaremos al microcontrolador, este proceso se repetirá varias veces a una velocidad lo suficientemente alta, como para no alcanzar a percibir los cambios, y tener la idea de que todos los LEDs deseados se encuentran prendidos a la vez.

Después de hacer un proceso similar, se obtuvo una tabla con los valores de cada número deseados, en este caso los números se crearon de determinada forma, la cual puede cambiar dependiendo las necesidades de cada persona, ya sea que se necesite mostrar letras o caracteres distintos.

Número	C1	C2	C3	C4	C5
0	0X41	0X2E	0X36	0X3A	0X41
1	0X3F	0X3D	0X00	0X3F	0X3F
2	0X3D	0X1E	0X2E	0X36	0X39
3	0X5D	0X3E	0X36	0X36	0X49
4	0X67	0X6B	0X6D	0X00	0X7F
5	0X58	0X3A	0X3A	0X3A	0X46
6	0X43	0X35	0X36	0X36	0X4F
7	0X7C	0X7E	0X0E	0X76	0X78
8	0X49	0X36	0X36	0X36	0X49
9	0X79	0X66	0X66	0X56	0X69

Tabla 1. Tabla para generar los números del 0 al 9 para la matriz 7x5.

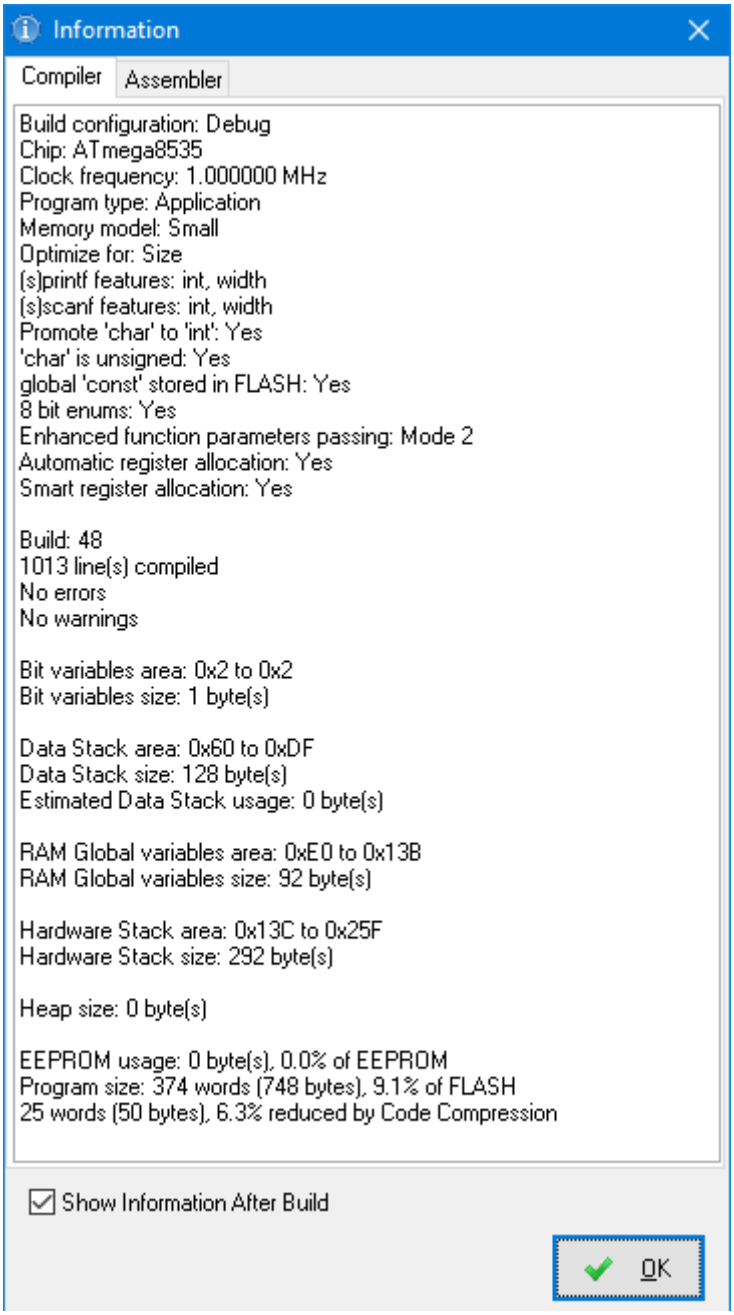


Figura 10. Compilación exitosa en CodeVision.

CÓDIGO GENERADO POR CODEVISION

```
/*****************************************************
This program was created by the CodeWizardAVR V3.48b
Automatic Program Generator
© Copyright 1998-2022 Pavel Haiduc, HP InfoTech S.R.L.
http://www.hpinfotech.ro

Project :
Version :
Date    :
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
char modo = 4;
char columnas = 0x01;
char filas = 0;
char indice = 0, numero = 0;
char repetir = 0, renglon = 0;
int cambio_caso = 0;
bit puede_cambiar = 0;

char modoCero[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

char modoUno[7][5] = {
    {0x01, 0x01, 0x01, 0x01, 0x01},
    {0x02, 0x02, 0x02, 0x02, 0x02},
    {0x04, 0x04, 0x04, 0x04, 0x04},
    {0x08, 0x08, 0x08, 0x08, 0x08},
    {0x10, 0x10, 0x10, 0x10, 0x10},
    {0x20, 0x20, 0x20, 0x20, 0x20},
    {0x40, 0x40, 0x40, 0x40, 0x40}
};

char modoCuatro[10][5] = {
    {0x41, 0x2E, 0x36, 0x3A, 0x41},
    {0x3F, 0x3D, 0x00, 0x3F, 0x3F},
    {0x3D, 0x1E, 0x2E, 0x36, 0x39},
    {0x5D, 0x3E, 0x36, 0x36, 0x49},
    {0x67, 0x6B, 0x6D, 0x00, 0x7F},
    {0x58, 0x3A, 0x3A, 0x3A, 0x46},
    {0x43, 0x35, 0x36, 0x36, 0x4F},
    {0x7C, 0x7E, 0x0E, 0x76, 0x78},
    {0x49, 0x36, 0x36, 0x36, 0x49},
    {0x79, 0x36, 0x36, 0x56, 0x69}
};
```

```
char modoCinco[174] = {
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,   // -
    0x01, 0x76, 0x76, 0x76, 0x01, 0x7f,   // A
    0x00, 0x3f, 0x3f, 0x3f, 0x3f, 0x7f,   // L
    0x01, 0x76, 0x76, 0x76, 0x01, 0x7f,   // A
    0x00, 0x7b, 0x77, 0x6f, 0x00, 0x7f,   // N
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,   // -
    0x00, 0x7d, 0x7b, 0x7d, 0x00, 0x7f,   // M
    0x01, 0x76, 0x76, 0x76, 0x01, 0x7f,   // A
    0x00, 0x3f, 0x3f, 0x3f, 0x3f, 0x7f,   // L
    0x01, 0x76, 0x76, 0x76, 0x01, 0x7f,   // A
    0x41, 0x3e, 0x36, 0x36, 0x45, 0x7f,   // G
    0x41, 0x3e, 0x3e, 0x3e, 0x41, 0x7f,   // O
    0x00, 0x7b, 0x77, 0x6f, 0x00, 0x7f,   // N
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,   // -
    0x5f, 0x3f, 0x3f, 0x3f, 0x40, 0x7f,   // J
    0x41, 0x3e, 0x3e, 0x3e, 0x41, 0x7f,   // O
    0x00, 0x76, 0x66, 0x56, 0x39, 0x7f,   // R
    0x41, 0x3e, 0x36, 0x36, 0x45, 0x7f,   // G
    0x00, 0x36, 0x36, 0x36, 0x3e, 0x7f,   // E
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f,   // -
    0x00, 0x7d, 0x7b, 0x7d, 0x00, 0x7f,   // M
    0x01, 0x76, 0x76, 0x76, 0x01, 0x7f,   // A
    0x00, 0x76, 0x66, 0x56, 0x39, 0x7f,   // R
    0x7e, 0x7e, 0x00, 0x7e, 0x7e, 0x7f,   // T
    0x7f, 0x3e, 0x00, 0x3e, 0x7f, 0x7f,   // I
    0x00, 0x7b, 0x77, 0x6f, 0x00, 0x7f,   // N
    0x00, 0x36, 0x36, 0x36, 0x3e, 0x7f,   // E
    0x1e, 0x2e, 0x36, 0x3a, 0x3c, 0x7f,   // Z
    0x7f, 0x7f, 0x7f, 0x7f, 0x7f, 0x7f    // -
};

void cambiar_modo(){
    cambio_caso = 0;
    indice = 0;
    renglon = 0;
    numero = 0;
    repetir = 0;
    columnas = 0x01;
    modo++;
    puede_cambiar = 0;
    if (modo == 6) modo = 0;
}



char *obtenerArreglo(int pos){
    char arr[5];
    int i, j=0;
```

```
    for(i=pos;i<pos+5;i++){
        arr[j] = modoCinco[i];
        j++;
    }
    return arr;
}

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

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

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

while (1)
      {

        switch (modo){
           case 0:
            filas = ~modoCero[indice];
            delay_ms(100);
           break;

           case 1:
            filas = ~modoUno[renglon][indice];
            repetir++;
            if (repetir == 5){
```

```
            renglon++;
            repetir = 0;
        }

        if (renglon == 7) renglon = 0;
      break;

      case 2:
        filas = ~modoUno[renglon][indice];
        renglon++;
        if (renglon == 7) renglon = 0;
      break;

      case 3:
        filas = ~modoUno[renglon][indice];
        delay_ms(100);
        repetir++;
        if (repetir == 5){
            renglon++;
            repetir = 0;
        }

        if (renglon == 7) renglon = 0;
      break;

      case 4:

        filas = obtenerArreglo(numero)[indice];
        repetir++;
        if (repetir == 60){
            repetir = 0;
            numero++;
        }
        if (numero == 169) {
            puede_cambiar = 1;
            numero = 0;
        }
      break;

      default:

        filas = modoCuatro[numero][indice];
        repetir++;
        if (repetir == 60){
            repetir = 0;
            numero++;
        }

        if (numero == 10) {
```

```
            puede_cambiar = 1;
            numero = 0;
        }
    }


    PORTC = columnas;

    // Contador de anillo
    switch (columnas){
       case 0x01:
        columnas = 0x02;
       break;

       case 0x02:
        columnas = 0x04;
       break;

       case 0x04:
        columnas = 0x08;
       break;

       case 0x08:
        columnas = 0x10;
       break;

       default:
        columnas = 0x01;
    }

    // Indice
    indice++;
    if (indice == 5) {
        indice = 0;
        cambio_caso++;
    }


    //PORTC = columnas;

    switch (modo){
       case 0:
        if (cambio_caso == 4) cambiar_modo();
       break;

       case 1:
        if (cambio_caso == 50) cambiar_modo();
       break;
```

```
            case 2:
             if (cambio_caso == 50) cambiar_modo();
            break;

            case 3:
             if (cambio_caso == 7) cambiar_modo();
            break;

            default:
             if (puede_cambiar) cambiar_modo();
        }


        PORTD = filas;
        if(modo==4) delay_ms(4);
        else delay_ms(10);

      }
}
```

CIRCUITO ELECTRICO EN PROTEUS

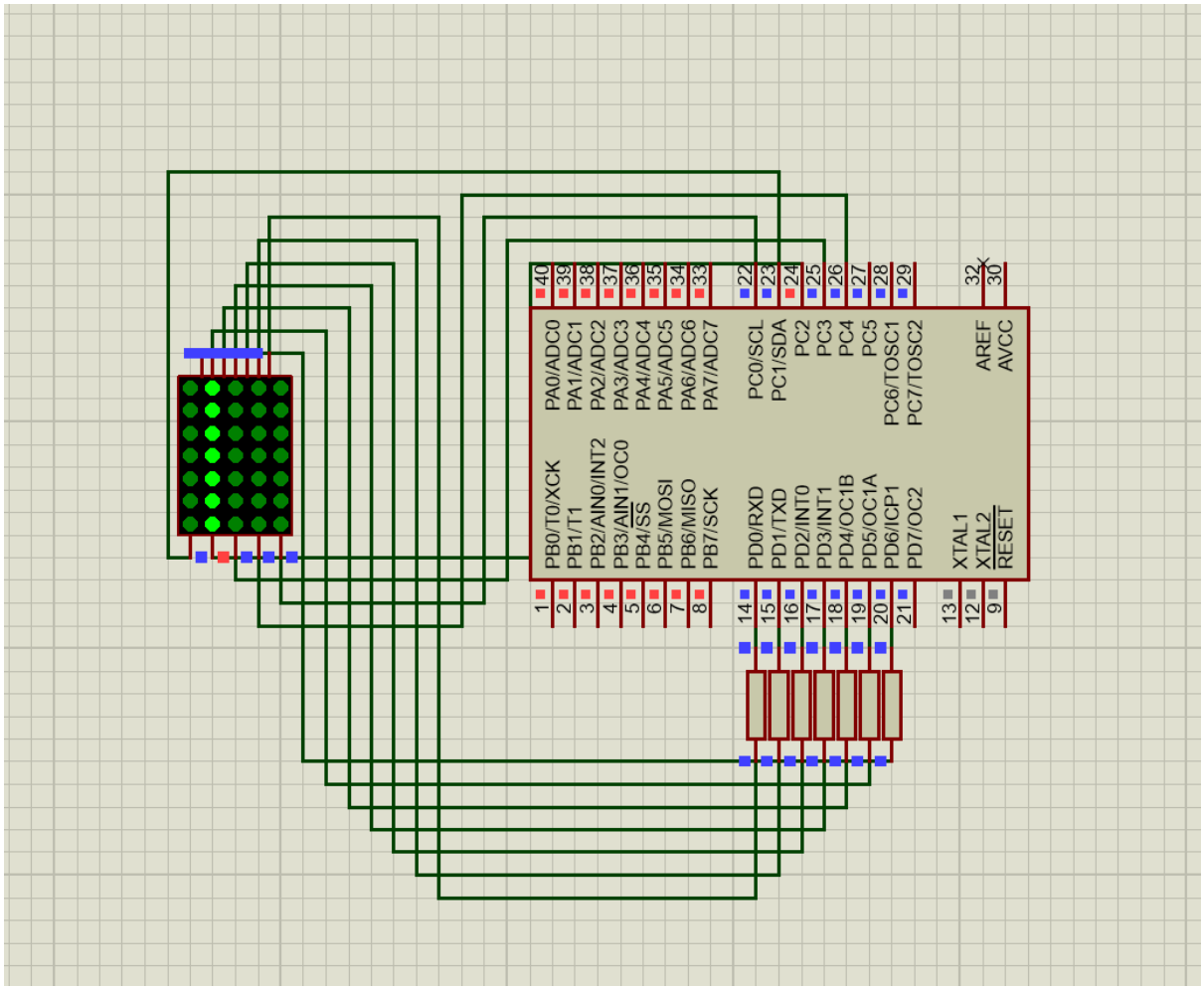
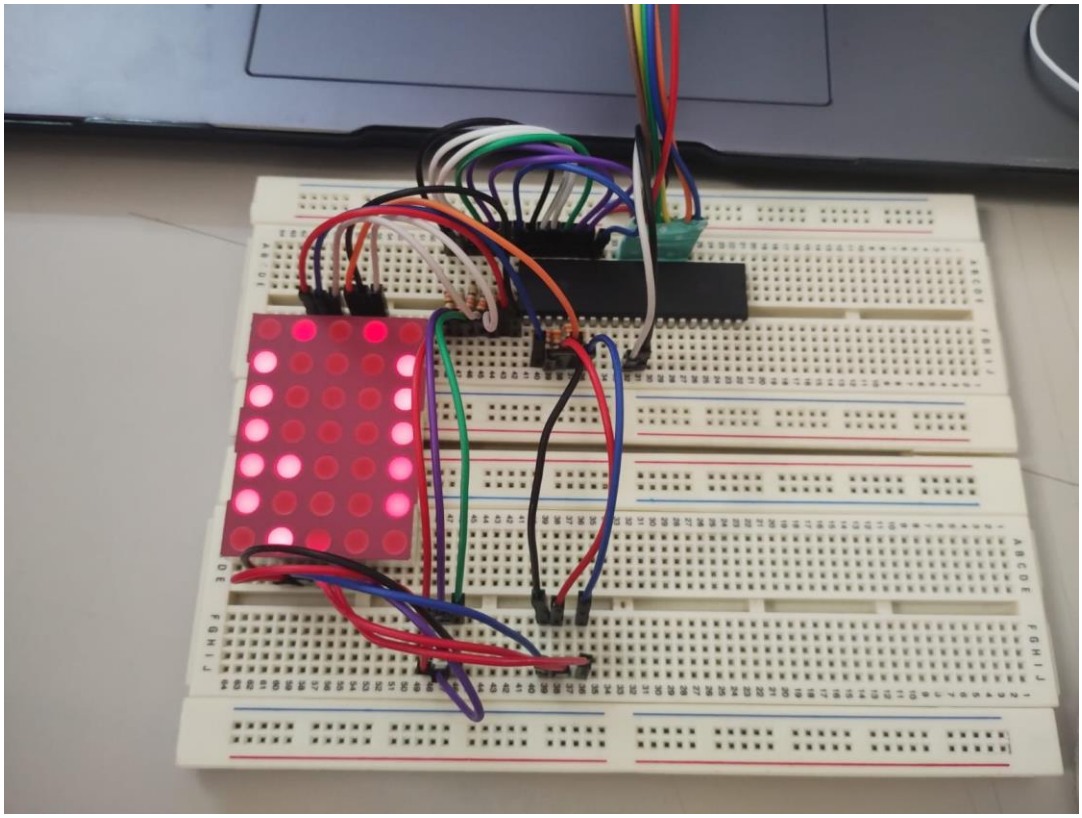
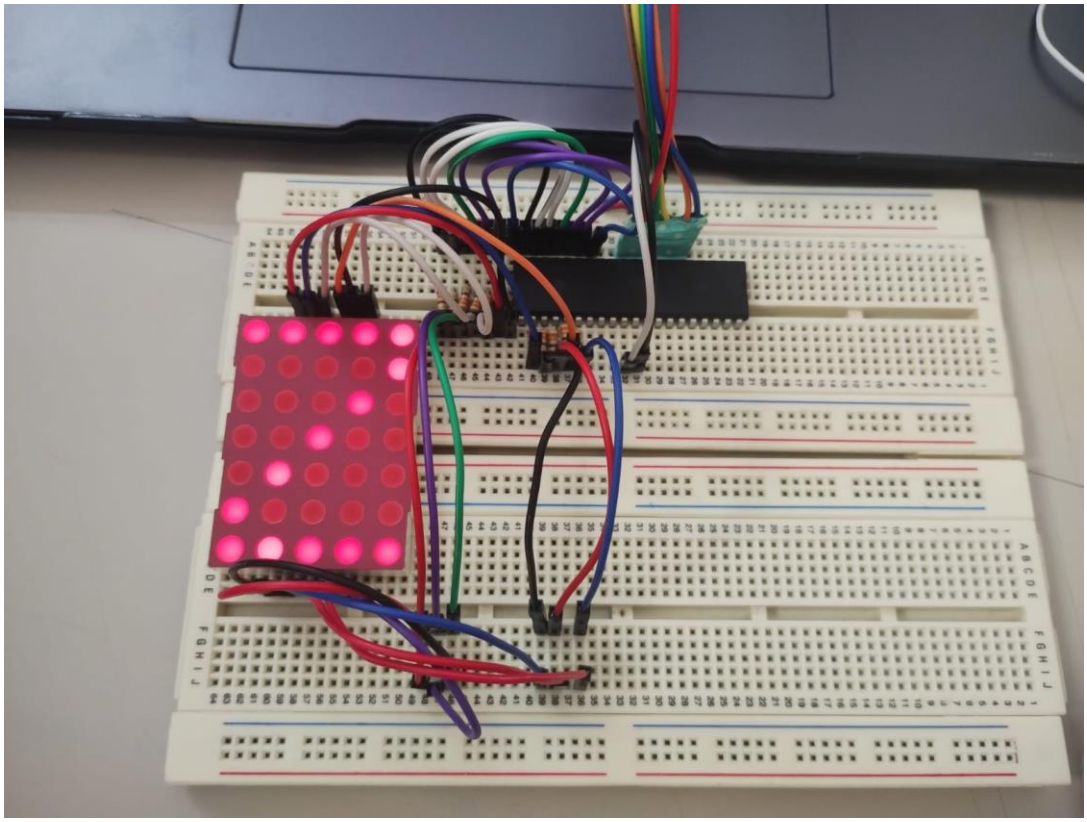


Figura 11. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO





OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrian

La matriz de leds es prácticamente un arreglo de leds de tal forma que podemos representar algunas cosas en ella encendiendo los leds necesarios para interpretar la imagen. Podríamos verlo como en el que podemos hacer tanto números como figuras al estilo de los videojuegos de 8bits.

Al igual que en los displays de 7 segmentos, podemos encontrar matrices de leds en configuración de cátodo o ánodo común, dependiendo el comportamiento que deseemos programar en nuestra matriz.

Martínez Chávez Jorge Alexis

La programación de estas matrices es muy sencilla, siempre y cuando se tengan conocimientos de conversiones de números binarios a hexadecimales o bien el manejo de una herramienta que ayude a realizar esta tarea. Una vez teniendo la combinación hexadecimal correspondiente a la combinación binaria que buscamos para prender los leds en la matriz solo basta asignarla al puerto de salida del microcontrolador para mostrar la información en la matriz.

La complejidad de la práctica estuvo en el barrido de las columnas para poder formar las figuras como los números, puesto que hay que entender bien la lógica que siguen estas matrices para poder manejar cada columna independientemente y así poder encender todos los leds que deseemos para formar una figura.

BIBLIOGRAFÍA

[1] HETPRO/TUTORIALES, “Matriz de LEDs Arduino AVR PIC 5x7 ADC.avr,” *HETPRO/TUTORIALES*, Apr. 15, 2014. https://hetpro-store.com/TUTORIALES/matriz-de-leds-5x7/ (accessed May 27, 2022).