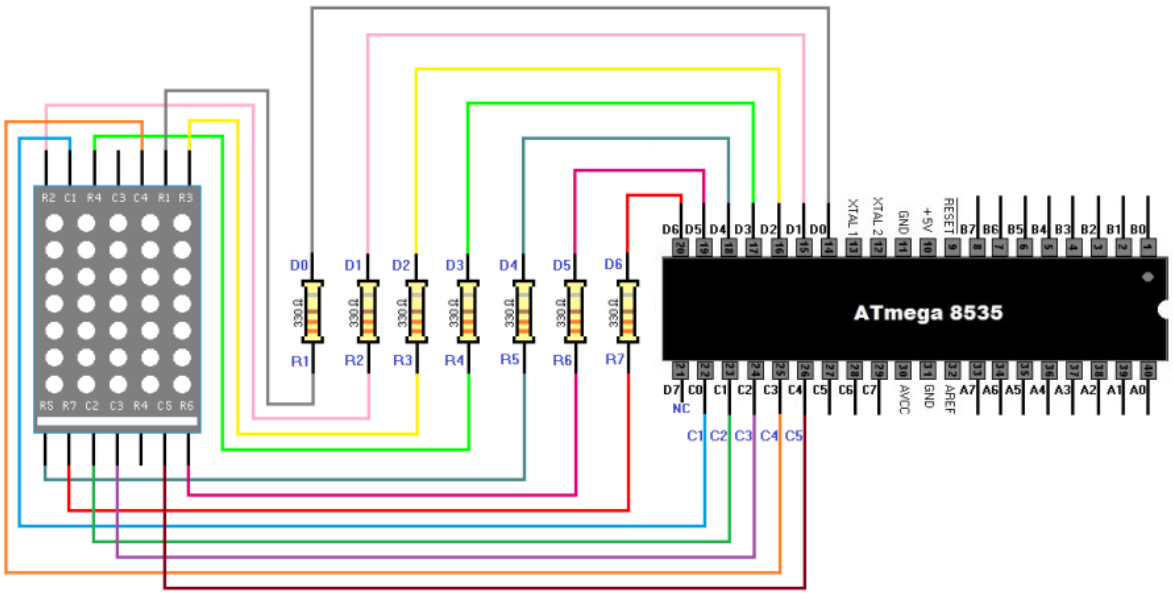
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS EN CHIP

PRACTICA No. 17

"Matriz 7x5"



ALUMNO: RENTERIA ARRIAGA JOSUE.

GRUPO: 6SCM1

PROFESOR: FERNANDO AGILAR SÁNCHEZ.

13/ENERO /2023

I. OBJETIVO GENERAL:

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

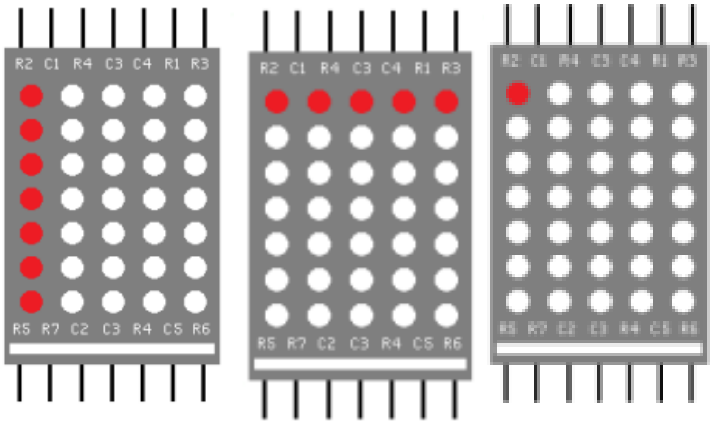
II. MATERIAL Y EQUIPO EMPLEADO

- ✓ CodeVision AVR.
- ✓ AVR Studio 4.
- ✓ Microcontrolador ATmega 8535.
- ✓ 3 Display Cátodo común.
- ✓ 8 Resistores de 330 Ω a ¼ W.
- ✓ 1 Matriz de leds de 7x5.

III. INTRODUCCIÓN TEÓRICA

En esta práctica podremos saber utilizar correctamente y aplicar la implementación del código en el lenguaje de programación C de los conceptos de hacer un matriz de Leds de 7x5 que se pueda observar un conjunto de patrones especificados en la práctica.

En esta práctica se estudiará la implementación de un Programa y Circuito para programar el puerto A y B como salida de nuestra Matriz de Leds, en esta ocasión no hay puertos de entrada ya que todas las secuencias se programarán internamente. La práctica nos mostrara en nuestra pantalla alguno de las siguientes secuencias:



En este caso de la práctica se ocuparán algunos conceptos obtenidos anteriormente en el curso como lo es la implementación de código en C para poder programar nuestro microcontrolador, algunos conceptos de Fundamentos de Diseño Digital (Display de 7 Segmentos), concepto de un convertidor analógico a Digital que se vio en Arquitectura y de Electrónica Analógica para poder realizar correctamente el armado de nuestro circuito y que este funcione correctamente. También se ocupará la siguiente tabla para implementar el mostrado de los números del 0 al 9:

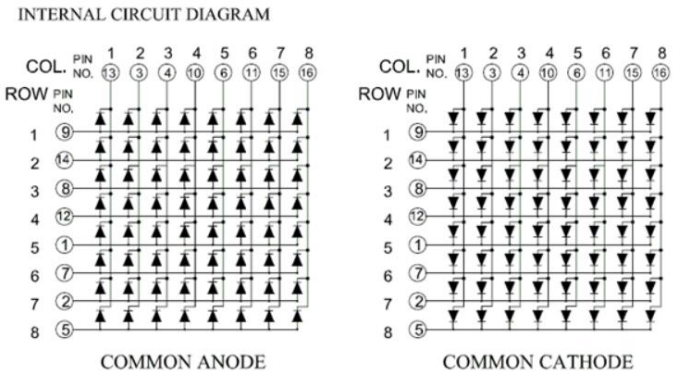
Número	C1	C2	C3	C4	C5
0	0X41	0X2E	0X36	0X3A	0X41
1	0X3F	0X3D	0X00	0X3F	0X3F
2	0X3D	0X1E	0X2E	0X36	0X39
3	0X5D	0X3E	0X36	0X36	0X49
4	0X67	0X6B	0X6D	0X00	0X7F
5	0X58	0X3A	0X3A	0X3A	0X46
6	0X43	0X35	0X36	0X36	0X4F
7	0X7C	0X7E	0X0E	0X76	0X78
8	0X49	0X36	0X36	0X36	0X49
9	0X79	0X66	0X66	0X56	0X69

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

Ocuparemos algunos elementos como lo dice el formato de la practica ya que ocuparemos el simulador Code Vision para poder implementar el código para después programar nuestro ATMega8535, en mi caso se ocupará el programa Khazama AVR con su programador para poder programar nuestro ATMega con el archivo .hex que se generó anteriormente. Para la parte del armado del circuito será algo sencillo ya que solo se implementará una Matriz de Leds.

Algunos conceptos que se necesitan para poder entender algunos conceptos de la practica son los siguientes:

Una matriz de LED es un conjunto de LED cuyos agrupados en filas (o renglones) y columnas, las cuales pueden ser tan grandes como un quiera o requiera. En la siguiente ilustración se muestra el esquema de conexión de una matriz de 8 x 8 de un solo color.



En la ilustración izquierda se muestra que en las filas todos los ánodos comparten un solo nodo mientras que la imagen de la derecha las filas de LED los cátodos están conectados a un solo nodo, de ahí que las matrices tengan la configuración de ánodo común o de cátodo común. Por consiguiente, en el caso de la matriz de ánodo común, los cátodos de los LED de cada columna están conectados a un sólo nodo y, en la matriz de cátodo común, los ánodos de los LED de cada columna están conectados a un nodo.

Para que funcione, además de la matriz de LED, es muy importante que se encuentren perfectamente acoplados las partes que hacen que el sistema funcione como una sola entidad: el hardware y las líneas de código del programa.

Esta práctica nos ayudara a entender cómo se pueden implementar los circuitos con un microcontrolador y todo el proceso que conlleva realizar la programación del microcontrolador.

Con esta introducción se podrá comprender y realizar el desarrollo de la práctica que se desarrolla más adelante.

IV. DESARROLLO EXPERIMENTAL

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

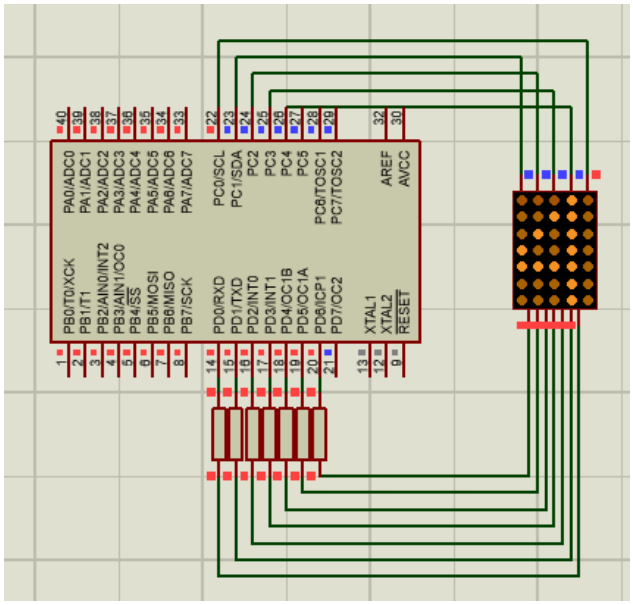


Fig.1_*Circuito Armado y Simulado en Proteus.*

CIRCUITO EN FISICO ARMADO.

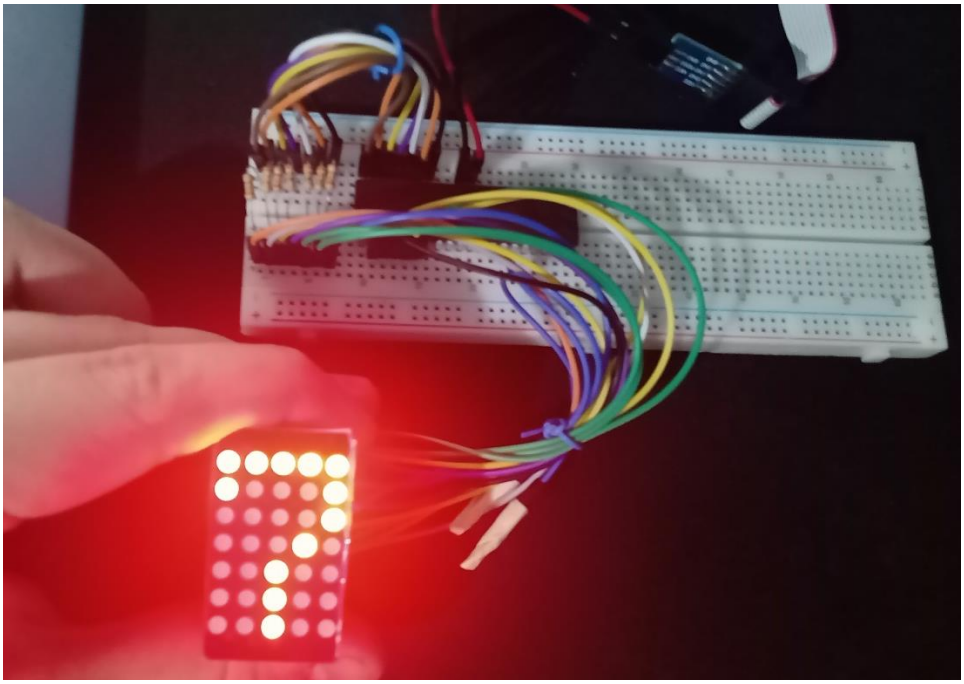


Fig.2_*Circuito Armado en Físico (Matriz de Leds 7x5).*

V. CÓDIGO

```
#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;
char 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}
};

void cambiar_modo(){
    cambio_caso = 0;
    indice = 0;
    renglon = 0;
    numero = 0;
    repetir = 0;
    columnas = 0x01;
    modo++;
    puede_cambiar = 0;
    if (modo == 5) modo = 0;
}
```

```
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;

           default:
            filas = modoCuatro[numero][indice];
```

```
    repetir++;
    if (repetir == 60){
        repetir = 0;
        numero++;
    }

    if (numero == 10) {
        puede_cambiar = 1;
        numero = 0;
    }
}

// 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++;
}

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();
          }

          PORTC = columnas;
          PORTD = filas;
          delay_ms(10);
        }
}
```

VI. OBSERVACIONES Y CONCLUSIONES

Gracias a la realización de la práctica podemos observar cómo se utiliza e implementa un circuito con un microcontrolador (ATMega8535) y como gracias a su programador pudimos logra programar el integrado con nuestro código en lenguaje C. En esta práctica al momento de armar el circuito y simularlo pudimos observar que hay una forma de colocar correctamente la Matriz de Leds, cabe destacar también que en el caso de las Matrices de Leds también hay de Ánodo y Cátodo por lo cual al no saber esto me costo demasiado hacer al inicio mi práctica, cuando descubrí esto supe que tenia una Matriz diferente a lo que estaba programando.

Tras la investigación y explicación de la práctica previa me quedo un poco más claro como poder implementar una Matriz de Leds 7x5 y programar el ATMega8535. También al observar el data set de la Matriz se comprendido mejor como conectarlo en físico este componente.

Para concluir esta práctica podemos decir que fue una experiencia interesante ya que se tuvo que hacer la parte de la Matriz y si me costó mucho hacerlo y electrónica ya había estudiado eso, pero, nunca realizado en físico. Gracias a la anterior practica ya fue más fácil de hacerlo, ya que es casi lo mismo que esta.

VII. BIBLIOGRAFÍA

- ✓ CodeVisionAVR. (2018). CodeVisionAVR. 20/Septiembre/2022, de CodeVisionAVR Sitio web: http://www.hpinfotech.ro/cvavr-download.html

- ✓ Abraham Camarillo. (2017). ¿Cómo funcionan los teclados matriciales y matrices de LEDs?, 13/Enero/2023, de 330 Ohms, Sitio web: https://blog.330ohms.com/2017/09/27/como-funcionan-los-teclados-matriciales-y-matrices-de-leds/