

**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE CÓMPUTO**



# Introducción a los microcontroladores 3CM16

## Practica 10

### Teclado matricial

### Equipo 7

#### **Integrantes:**

- ♥ **Bocanegra Heziquio Yestlanezi**
- ♥ **Dominguez Durán Alan Axel**
- ♥ **Hernandez Mendez Oliver Manuel**
- ♥ **Martinez Cruz José Antonio**

**Profesor: Aguilar Sanchez Fernando**

## índice

Objetivo.....	3
Introducción.....	3
Teclado matricial .....	3
Material y equipo empleado .....	4
Desarrollo experimental.....	5
Circuito en protoboard .....	5
Estructura del programa .....	6
Código CodeVisionAVR.....	6
Simulación – proteus.....	10
Conclusiones .....	11
Bocanegra Heziquio Yestlanezi.....	11
Dominguez Durán Alan Axel.....	11
Hernandez Mendez Oliver Manuel .....	11
Martinez Cruz José Antonio .....	11

## Objetivo

Al término de la sesión, los integrantes del equipo contarán con la habilidad de realizar un teclado matricial.

## Introducción

### Teclado matricial


es un dispositivo de entrada utilizado en la mayoría de los teclados de computadoras y otros dispositivos electrónicos. En lugar de contar con un interruptor mecánico separado para cada tecla, un teclado matricial utiliza una matriz de interruptores que se conectan en filas y columnas.

El diseño básico de un teclado matricial implica la intersección de filas y columnas para formar una matriz. Cada tecla en el teclado está asociada con un interruptor en particular en esa matriz. Los interruptores pueden ser de tipo "abiertos" o "cerrados", dependiendo de si una tecla está siendo presionada o no [1].

Cuando se presiona una tecla en el teclado matricial, el interruptor correspondiente se cierra y se crea una conexión eléctrica entre una fila y una columna en la matriz. El teclado matricial utiliza un proceso conocido como "exploración de matriz" para detectar qué tecla se ha presionado [1].

El proceso de exploración implica escanear secuencialmente cada fila, mientras se detecta el estado de las columnas. Se envía una señal de exploración a cada fila, una por una, y se verifica el estado de las columnas para determinar si alguna tecla ha sido presionada en esa fila en particular. Si se detecta una tecla presionada, se puede identificar su posición exacta en la matriz, lo que permite al dispositivo registrar la tecla que ha sido presionada [1].

Una vez que se ha realizado el escaneo de la matriz y se han detectado todas las teclas presionadas, se envía una señal al dispositivo correspondiente (como una computadora) para que procese la entrada del teclado y realice la acción asociada con esa tecla.



## Material y equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- 1 Display cátodo común
- 7 Resistores de  $330\ \Omega$  a  $1/4\ W$
- 3 Resistor de  $100\ \Omega$  a  $1/4\ W$
- 9 Push Button

## Desarrollo experimental

1.- Diseño de un teclado matricial de 3\*3, con despliegue a display a 7 segmentos. Los pines C0, C1 y C2 serán los pines de salida por donde se enviarán los códigos de "saceneo", los pines C3, C4 y C5 serán los pines de entrada donde se leerán los botones presionados.

### Circuito en protoboard

A continuación, en la imagen 1 se presenta el circuito armado y funcionando en la protoboard

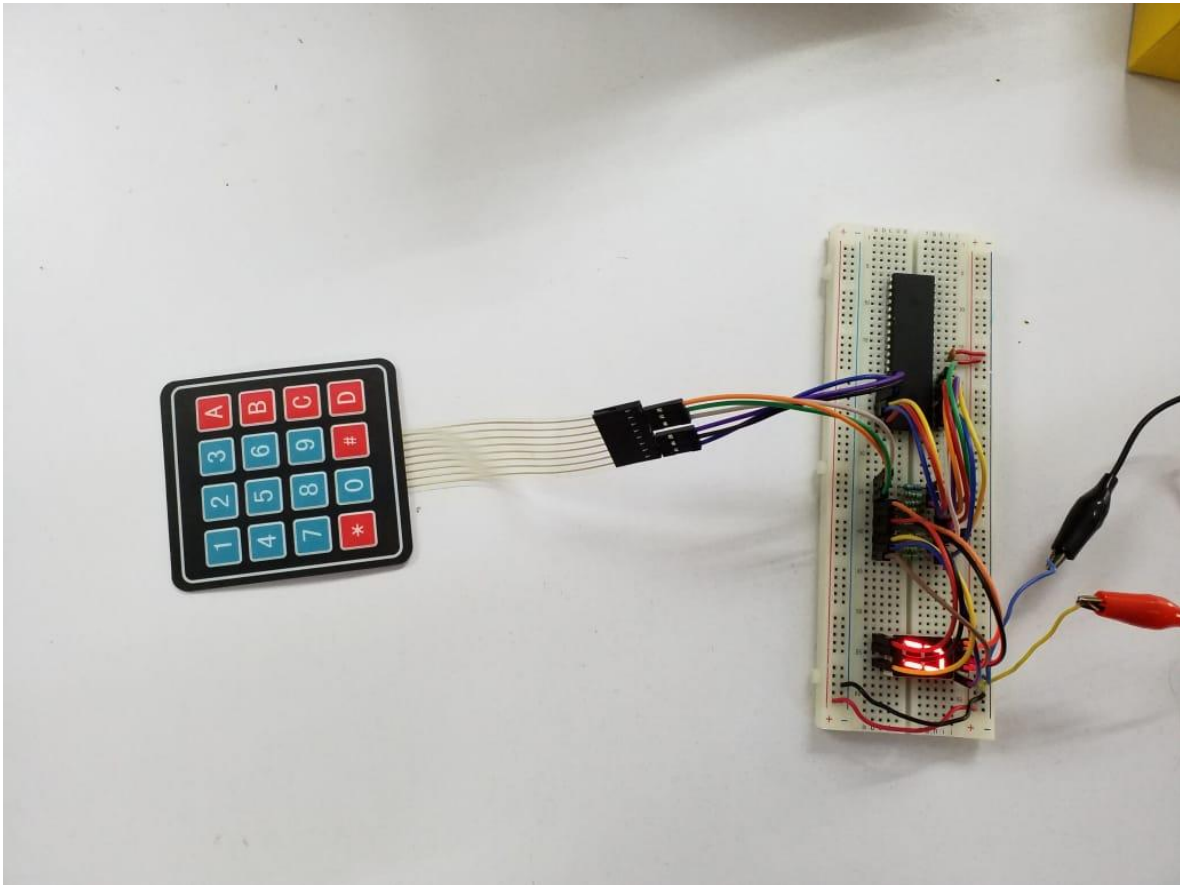


Imagen 1 circuito en protoboard

# Estructura del programa

## Codigo CodeVisionAVR

```
/******  
This program was created by the CodeWizardAVR V3.46a  
Automatic Program Generator  
◆ Copyright 1998-2021 Pavel Haiduc, HP InfoTech S.R.L.  
http://www.hpinfotech.ro  
  
Project :  
Version :  
Date : 20/05/2023  
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 tecla, lectura;  
const char tabla7segmentos [10] =  
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};  
  
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=Out Bit1=Out  
Bit0=Out  
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (1<<DDA2) |  
(1<<DDA1) | (1<<DDA0);  
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=1 Bit1=0 Bit0=0
```

```

PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<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=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) |
(0<<DDC1) | (0<<DDC0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
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=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In
DDRD=(0<<DDD7) | (0<<DDD6) | (0<<DDD5) | (0<<DDD4) | (0<<DDD3) | (0<<DDD2) |
(0<<DDD1) | (0<<DDD0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
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)
{
    tecla = 0;
    PORTA=0b00111110;
    lectura=PINA&0b00111000;
    if (lectura==0b00110000) tecla=1;
    if (lectura==0b00101000) tecla=4;
    if (lectura==0b00011000) tecla=7;

    PORTA=0b00111101;
    lectura=PINA&0b00111000;
    if (lectura==0b00110000) tecla=2;
    if (lectura==0b00101000) tecla=5;
    if (lectura==0b00011000) tecla=8;

    PORTA=0b00111011;
    lectura=PINA&0b00111000;
    if (lectura==0b00110000) tecla=3;
    if (lectura==0b00101000) tecla=6;
    if (lectura==0b00011000) tecla=9;
}

```

```

PORTB=tabla7segmentos [tecla];
delay_ms(50);
}
}

```

## Simulacion - proteus

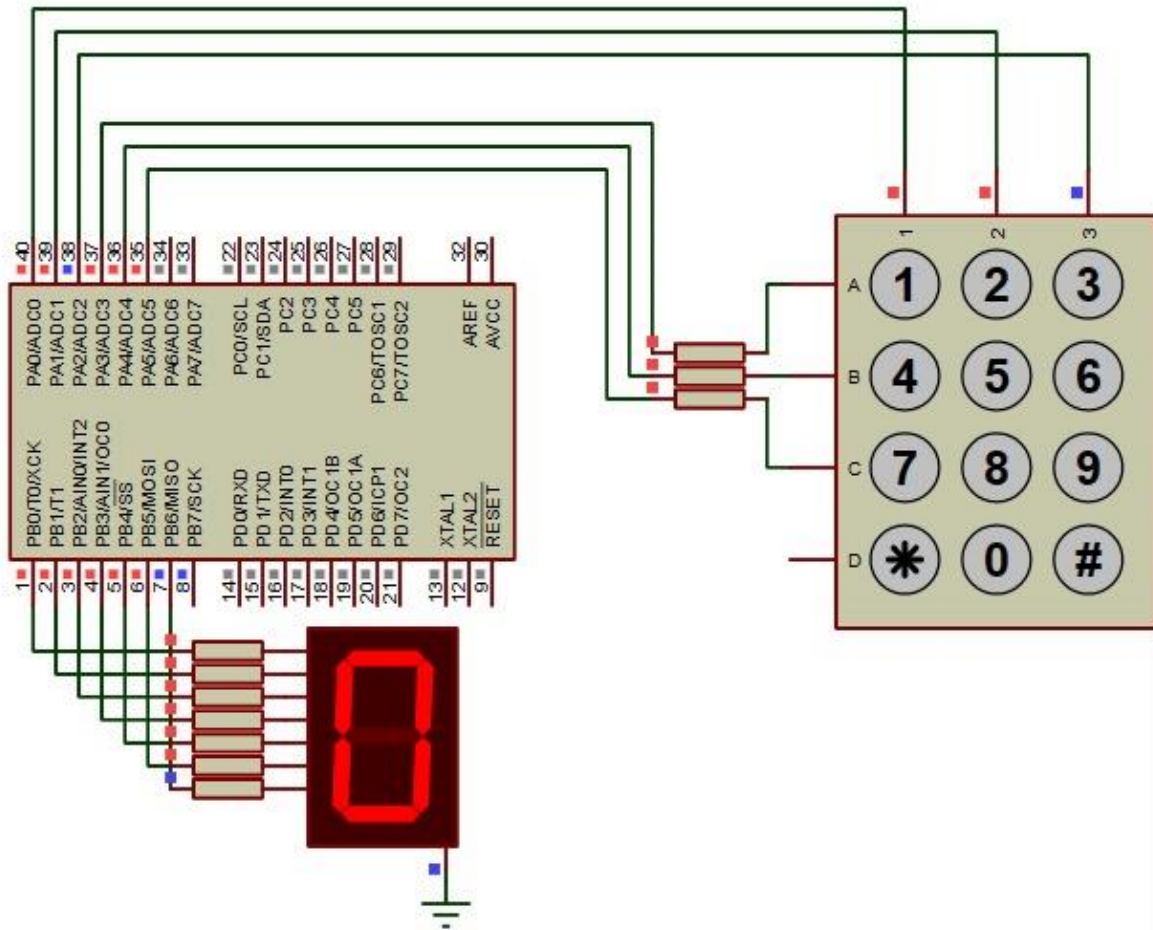


Imagen 2 circuito simulado en proteus

## Conclusiones

Bocanegra Heziquio Yestlanezi

Puedo concluir con base en mis conocimientos adquiridos en la materia que un teclado matricial funciona utilizando una matriz de interruptores organizados en filas y columnas. Cuando se presiona una tecla, se cierra un interruptor específico en la matriz y se detecta mediante la exploración secuencial de las filas y la verificación del estado de las columnas. Esto permite al dispositivo reconocer qué tecla ha sido presionada y procesar la entrada correspondiente.

Dominguez Durán Alan Axel

En esta decima práctica, comenzamos a manipular los dígitos del display con entradas puntuales del micro, de esta forma la parte del código se vuelve un poco más compleja y la parte del hardware comienza a simplificarse con ayuda del propio software. Con este preámbulo de simplificación de armado se da preámbulo a prácticas posteriores y se amplían las posibilidades de construcción para diferentes soluciones de problemas e ideas, dependiendo el caso de uso.

Hernandez Mendez Oliver Manuel

Gracias a esta práctica aprendimos a realizar el diseño de este teclado matricial y su interconexión con un display de 7 segmentos, nos percatamos de que se requiere una cuidadosa asignación de pines de salida y entrada, junto con la implementación adecuada de algoritmos de escaneo y lectura de botones. Este proyecto es una oportunidad para adquirir experiencia en el diseño de interfaces y la programación de microcontroladores, ofreciendo la posibilidad de crear aplicaciones interactivas y funcionales.

Martinez Cruz José Antonio

Para la realización de esta práctica se utilizó un teclado ya utilizado con anterioridad, esto no permitió realizar la práctica de manera más eficaz y sin arriesgarnos a tener problemas utilizando la matriz de botones. Sin embargo, es importante reconocer de manera adecuada los pines del teclado para evitar dañarlo. Una vez ya implementado, solo fue necesario realizar la programación del microprocesador y así poder lograr nuestro objetivo de mostrar el número digitado en el teclado por el usuario en el display de 7 segmentos.