INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

Profesor: Aguilar Sánchez Fernando

Práctica 10 "Teclado matricial"

Integrantes del Equipo:

- **Arizmendi Alvarado Brian**
- **Meza Hernandez Roberto Carlos**
- **Rodríguez López Ricardo**

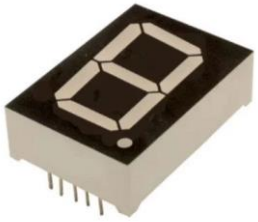
Objetivo

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

Introducción Teórica

En esta práctica, exploramos las capacidades del microcontrolador ATmega8535 al utilizar un teclado matricial y un display de 7 segmentos. El objetivo principal es crear un sistema interactivo en el cual podamos ingresar datos a través del teclado y visualizarlos de manera clara y precisa en el display.

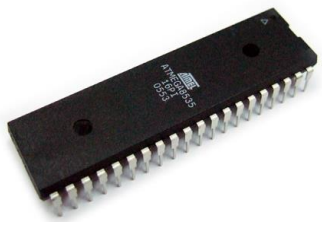
El teclado matricial nos proporciona una interfaz cómoda y versátil para capturar datos de entrada. Con su diseño en forma de matriz, podemos utilizar un número limitado de pines del microcontrolador para controlar una gran cantidad de teclas. Esto nos permite implementar sistemas más complejos sin necesidad de utilizar una gran cantidad de pines GPIO.



Por otro lado, el display de 7 segmentos es ampliamente utilizado para mostrar información numérica de manera legible. Consiste en siete segmentos individuales que pueden ser encendidos o apagados para representar diferentes dígitos numéricos. A través de la programación adecuada del microcontrolador, podremos enviar los datos capturados del teclado al display para su visualización.

Durante el desarrollo de esta práctica, explicaremos cómo configurar los pines del microcontrolador para el teclado matricial y el display de 7 segmentos, así como las técnicas de programación necesarias para leer las teclas presionadas y controlar la visualización en el display.

Al finalizar la práctica, habremos adquirido conocimientos valiosos sobre el manejo de dispositivos de entrada y salida, así como la programación efectiva del microcontrolador ATmega8535. Estos conocimientos nos permitirán diseñar y desarrollar proyectos más complejos y personalizados en el futuro, abriendo las puertas a nuevas aplicaciones y posibilidades en el campo de la electrónica.



Materiales y Equipo empleado

- CodeVision AVR
- AVR Studio 4
- Microcontrolador ATmega 8535
- Display cátodo común
- 7 resistores de 330 Ohms a ¼ W
- 3 resistores de 100 Ohms a ¼ W
- 9 push button

Código

```
// I/O Registers definitions
#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=In Bit1=In Bit0=In
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);
// State: Bit7=T Bit6=T Bit5=T Bit4=T Bit3=T Bit2=T Bit1=T Bit0=T
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=1 Bit6=1 Bit5=1 Bit4=1 Bit3=1 Bit2=1 Bit1=1 Bit0=1
PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

// Port C initialization
// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=Out Bit1=Out Bit0=Out
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (1<<DDC2) | (1<<DDC1) | (1<<DDC0);
// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=1 Bit1=1 Bit0=1
PORTC=(1<<PORTC7) | (1<<PORTC6) | (1<<PORTC5) | (1<<PORTC4) | (1<<PORTC3) | (1<<PORTC2) | (1<<PORTC1) | (1<<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;
```

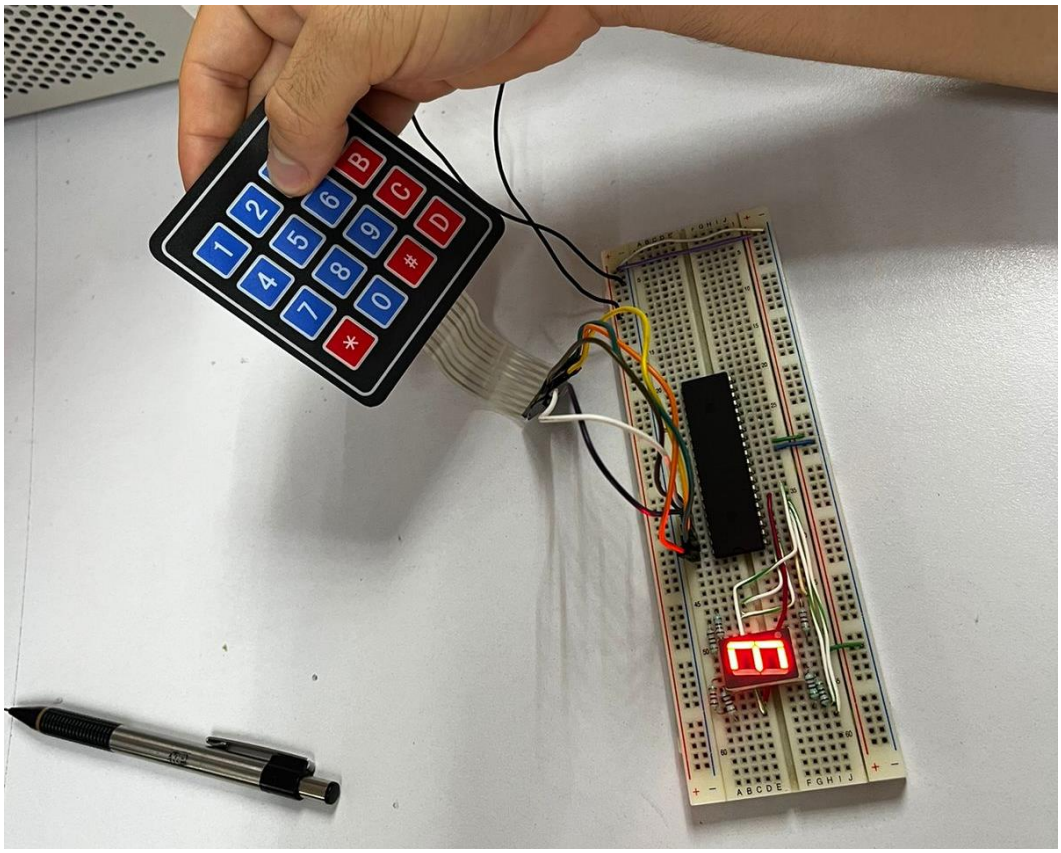
```
// Compare B Match Interrupt: Off
TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);
TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
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);
```

```
// 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);
```

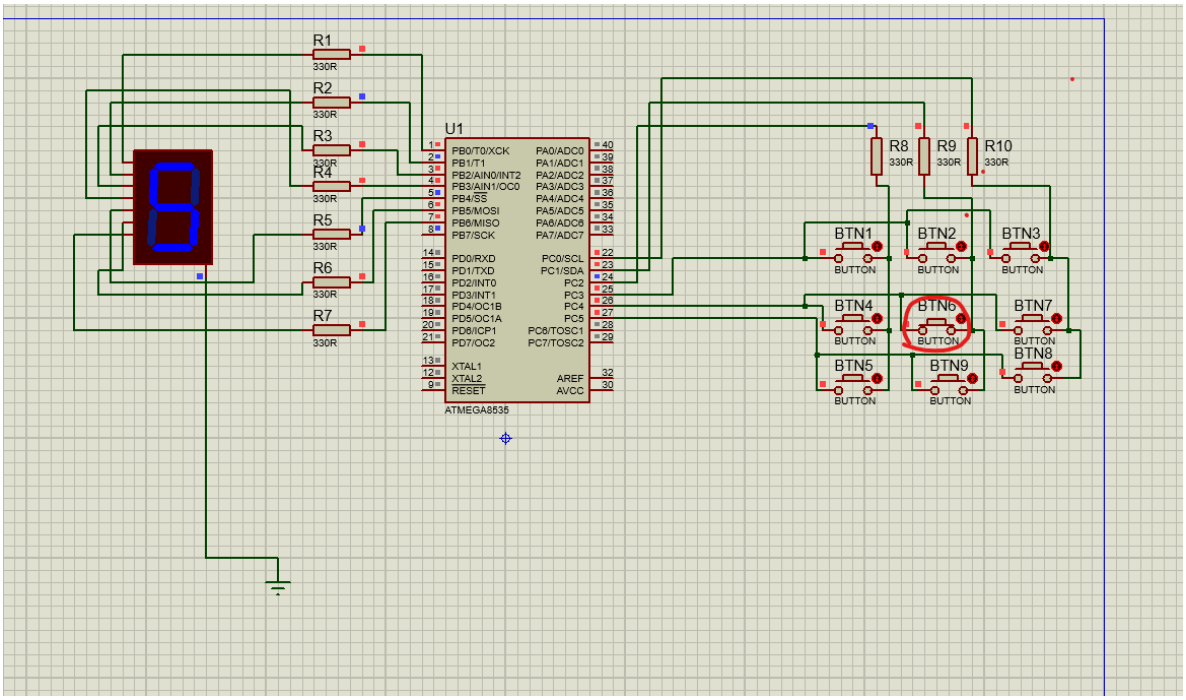
```
while (1)
        {
        // Place your code here
        tecla = 0;
        PORTC = 0b00111110;
        lectura = PINC & 0b00111000;
        if (lectura == 0b00110000)
                tecla = 1;
        if (lectura == 0b00101000)
                tecla = 4;
        if (lectura == 0b00011000)
                tecla = 7;

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

        PORTC = 0b00111011;
        lectura = PINC & 0b00111000;
        if (lectura == 0b00110000)
                tecla = 3;
        if (lectura == 0b00101000)
                tecla = 6;
        if (lectura == 0b00011000)
        tecla = 9;
```

```
        PORTB = tabla7segmentos[tecla];
        delay_ms(100);
        }
}
```

Pruebas



Simulación



Conclusiones

Arizmendi Alvarado Brian:

En conclusión, esta práctica nos ha brindado la oportunidad de explorar las capacidades del microcontrolador ATmega8535 al utilizar un teclado matricial y un display de 7 segmentos. Hemos aprendido cómo configurar los pines del microcontrolador y aplicar técnicas de programación para capturar datos de entrada a través del teclado y visualizarlos de manera clara y precisa en el display. Mediante el uso del teclado matricial, hemos logrado implementar un sistema interactivo eficiente, aprovechando la versatilidad de su diseño en forma de matriz para controlar múltiples teclas con un número limitado de pines.

Meza Hernández Roberto Carlos:

Durante el desarrollo de la práctica, hemos podido comprender cómo configurar y programar el ATmega8535 para establecer la comunicación bidireccional entre el teclado matricial y el display de 7 segmentos. Mediante el uso de técnicas adecuadas, hemos logrado capturar las teclas presionadas en el teclado matricial y reflejarlas en tiempo real en el display, permitiendo una interacción fluida y visualización clara de los datos ingresados.

Rodríguez López Ricardo

En esta práctica pude darme cuenta de la importancia que tiene la conexión con el teclado matricial, ya que dependiendo si se conectaba mal un solo pin en otra conexión, daba como resultado un salto de fila o columna. Ya que esto viene desde el código, dependiendo de donde estemos seleccionando el botón, nos da un código diferente para poder representarlo en el display.

Referencias Bibliográficas:

GSL Industrias. (2022, 12 de enero). Display de 7 segmentos. https://industriasgsl.com/blogs/automatizacion/display-de-7-segmentos

Llamas L. (2016, 2 de Octubre). Usar un teclado matricial con arduino. https://www.luisllamas.es/arduino-teclado-matricial/