INSTITUTO POLITÉCNICO NACIONAL

INTRODUCCIÓN A LOS MICROCONTROLADORES

PRÁCTICA 11 "Memoria EEPROM"

28 de mayo 2023

3CM16

Equipo:

- Bocanegra Heziquio Yestlanezi
- Domínguez Durán Alan Axel
- Hernández Méndez Oliver Manuel
- Martínez Cruz Jose Antonio

Profesor: Aguilar Sánchez Fernando

Índice

Índice de imágenes

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad de hacer uso de la memoria EEPROM del microcontrolador.

Introducción

Memoria EEPROM

La Electrically Erasable Programmable Read-Only Memory (EEPROM) ha desempeñado un papel crucial en el almacenamiento de datos no volátiles en controladores electrónicos. Los controladores, ya sean utilizados en dispositivos pequeños o sistemas más complejos, suelen hacer uso de la EEPROM para almacenar información vital, configuraciones y datos importantes que deben mantenerse incluso cuando se pierde la energía.

La EEPROM es un tipo de memoria que permite la lectura y escritura de datos de manera eléctrica. A diferencia de la memoria RAM, que pierde su contenido cuando se apaga el sistema, la EEPROM retiene los datos almacenados incluso en ausencia de energía. Esta característica la convierte en una opción ideal para los controladores, ya que les permite almacenar información esencial y mantenerla disponible al reiniciar el dispositivo [1].

En conclusión, la EEPROM desempeña un papel esencial en el almacenamiento de datos críticos y configuraciones importantes en los controladores. Su capacidad para retener información incluso en ausencia de energía ha permitido el desarrollo de sistemas electrónicos más flexibles y personalizables. A través de investigaciones y estudios, se ha demostrado la importancia de la EEPROM en aplicaciones como la automatización del hogar y se ha enfatizado la necesidad de evaluar y seleccionar la implementación adecuada de la EEPROM en sistemas embebidos [2].

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 1 Display cátodo común
- ♥ 7 Resistores de 330 Ω a 1⁄4 W
- ♥ 2 Push Button

Desarrollo experimental

1. Haga un programa en el cual con un botón conectado al pin C0 incrementará el valor en el display conectado en el puerto B y cuando se presione el botón C1 lo guarde en la memoria EEPROM. Después desconecte el microcontrolador de la energía eléctrica y vuélvalo a conectar para que observe que el dato se quedó guardado.

Circuito armado

Con los conocimientos obtenidos en las diferentes materias de electrónica, procedemos a realizar el montado del circuito en la protoboard como se muestra en la imagen 1.

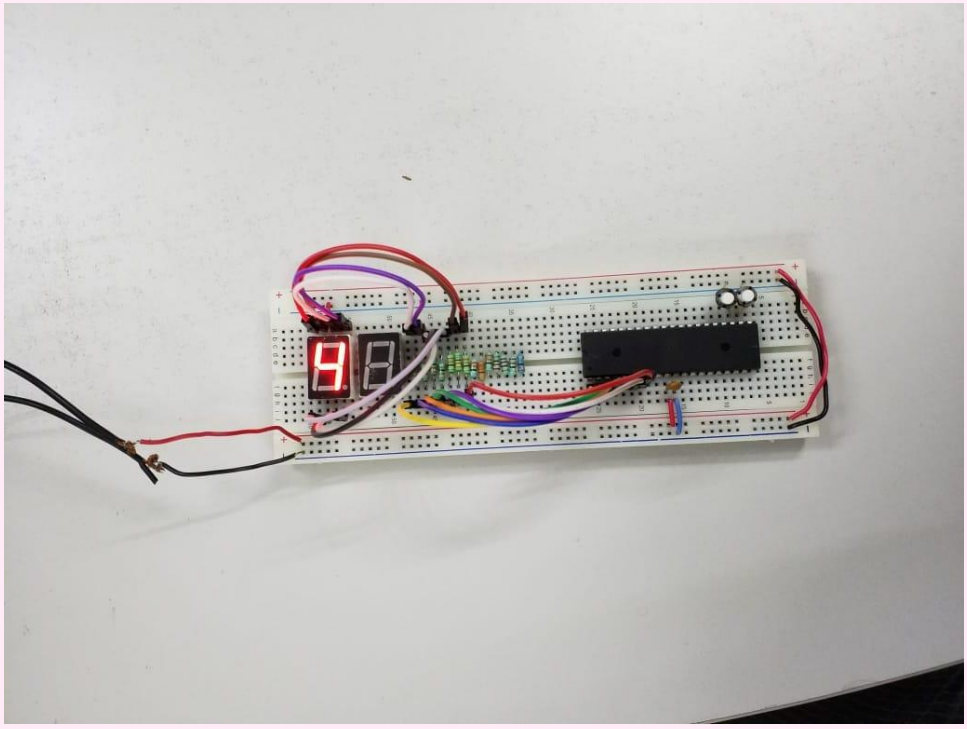

Imagen 1 Circuito memoria EEPROM

Estructura del programa

Código 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    : 09/04/2023
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>
#define btn_incrementa PINC.0
#define btn_guarda PINC.1

bit pasado_i, actual_i, pasado_g, actual_g;
eeprom unsigned char guardable;
unsigned char index;
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=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=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
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;

// 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);

index = guardable;

while (1)
      {
        actual_i = btn_incrementa;
        actual_g = btn_guarda;

        //Boton de incremento
        if (pasado_i == 1 && actual_i == 0) {
            index++;
            if (index > 9) index = 0;
            delay_ms(40);
        }

        if (pasado_i == 0 && actual_i == 1) {
            delay_ms(40);
        }

        //Boton de guardado
        if (pasado_g == 1 && actual_g == 0) {
            guardable = index;
```

```
            delay_ms(40);
        }

        if (pasado_g == 0 && actual_g == 1) {
            delay_ms(40);
        }

        pasado_i = actual_i;
        pasado_g = actual_g;
        PORTB = tabla7segmentos[index];
    }
}
```

Simulación – Proteus

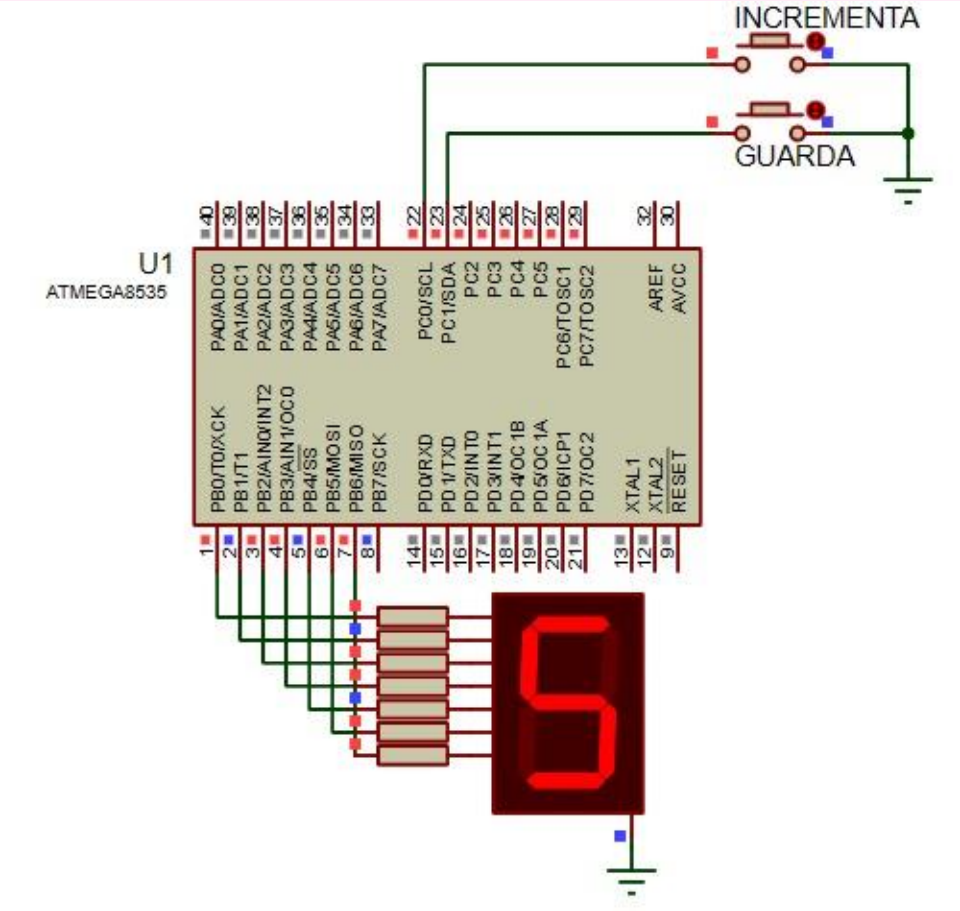

Imagen 2 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

En conclusión, hemos logrado diseñar y construir un teclado matricial de 3x3 con capacidad de desplegar la información en un display de 7 segmentos. A través de la implementación de los pines de salida C0, C1 y C2, podemos enviar los códigos correspondientes a las teclas presionadas mediante un proceso de "sacaneo". Por otro lado, utilizando los pines de entrada C3, C4 y C5, podemos leer los botones presionados por el usuario.

Este proyecto nos ha permitido adquirir la habilidad de crear un teclado matricial funcional, lo cual puede ser aplicado en diversos contextos y proyectos que requieran la interacción con un sistema de entrada de datos. Además, el uso del display de 7 segmentos nos brinda la posibilidad de visualizar la información de manera clara y concisa.

En conclusión, este proyecto ha sido exitoso en alcanzar nuestro objetivo inicial de brindarnos la habilidad de realizar un teclado matricial. Nos hemos enfrentado a desafíos técnicos y hemos adquirido conocimientos valiosos en el proceso. Estamos preparados para aplicar este conocimiento en futuros proyectos y continuar nuestro crecimiento en el ámbito de la electrónica y la programación.

Domínguez Durán Alan Axel

Al termino de esta práctica logramos hacer uso de la memoria interna (EEPROM) del microcontrolador, siendo un preámbulo útil para futuras prácticas donde necesitemos guardar información y estados dentro del micro.

Hernández Méndez Oliver Manuel

Gracias a esta práctica confirmamos que una EEPROM aplicada con un microcontrolador ofrece una solución eficiente y confiable para almacenar datos de forma permanente en un dispositivo electrónico. Permite la escritura y lectura de información de manera programable y eléctrica, lo que brinda flexibilidad y adaptabilidad en diversas aplicaciones. Además, la capacidad de retener datos incluso sin alimentación eléctrica la convierte en una opción ideal para preservar configuraciones y datos importantes en sistemas basados en microcontroladores.

Martínez Cruz José Antonio

Durante la sesión de trabajo, el equipo logró adquirir la habilidad de utilizar la memoria EEPROM del microcontrolador sin enfrentar problemas con el circuito. En la parte experimental, se realizó un programa que incrementaba el valor en el display conectado al puerto B cuando se presionaba el botón se guardaba el valor en la memoria EEPROM cuando se presionaba otro botón. Posteriormente, se desconectó y reconectó el microcontrolador para verificar que los datos se mantuvieron guardados correctamente, a pesar de que se

seguia incrementando el valor, este solo guardaba el valor que era indicado. Aunque se presentaron algunos problemas en el código implementado, el objetivo de utilizar la memoria EEPROM fue alcanzado exitosamente.

Referencias

[1] Liu, Y., Chen, X., Li, Z., Hu, Z., & Zhang, J. (2018). Design of an EEPROM-Based Control System for Smart Home Applications. IEEE Access, 6, 33062-33070.

[2] Rodríguez, C., Peinado, F., Jiménez, A., & Pérez, R. (2019). Analysis and Comparison of Different EEPROM Memory Implementations for Embedded Systems. Electronics, 8(7), 745.