INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 18

Pantalla

LCD 16x2

Entrega 17-06-2023

Equipo 7

Integrantes:

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

Profesor: Aguilar Sanchez Fernando

Índice

Índice de imágenes

Objetivo

Al término de la sesión, los integrantes del equipo contaran con la habilidad para manejar una pantalla LCD.

Introducción

Termómetro 0 a 50 °C (32 a 122 °F)

Una pantalla LCD 16x2 es un tipo común de pantalla de cristal líquido utilizada en proyectos electrónicos para mostrar información de manera visual. La pantalla consta de 16 caracteres por 2 líneas, lo que proporciona suficiente espacio para mostrar mensajes, números y otros datos relevantes. En esta práctica, se utiliza un microcontrolador para controlar y enviar los datos a la pantalla LCD, permitiendo una interfaz de usuario visual y legible.

El microcontrolador desempeña un papel clave en la implementación de la pantalla LCD 16x2. Este dispositivo programable se encarga de generar los comandos y enviar los datos necesarios para mostrar información en la pantalla. A través de los pines de salida del microcontrolador, se establece la comunicación con la pantalla LCD utilizando un protocolo específico, como el protocolo de control de visualización de caracteres (HD44780).

El diseño de esta práctica implica la conexión adecuada de los pines del microcontrolador a los pines de la pantalla LCD, siguiendo un esquema de conexión específico. Además, se requiere programación para enviar los datos de visualización y comandos al controlador de la pantalla LCD desde el microcontrolador. Esto permite la visualización de mensajes, caracteres y gráficos en la pantalla LCD 16x2, brindando una interfaz intuitiva y legible para el usuario.

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 1 Pantalla LCD 16x2
- ♥ 1 Potenciómetro 10KΩ
- ♥ 1 Resistor de 330Ω

Desarrollo experimental

Con la información que a continuación se menciona, diseñe un programa para visualizar en una pantalla LCD 16x2 la fecha, la hora y la temperatura actual, tal y como se muestra en la figura 3. Los botones son para el ajuste de fecha y hora.

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.

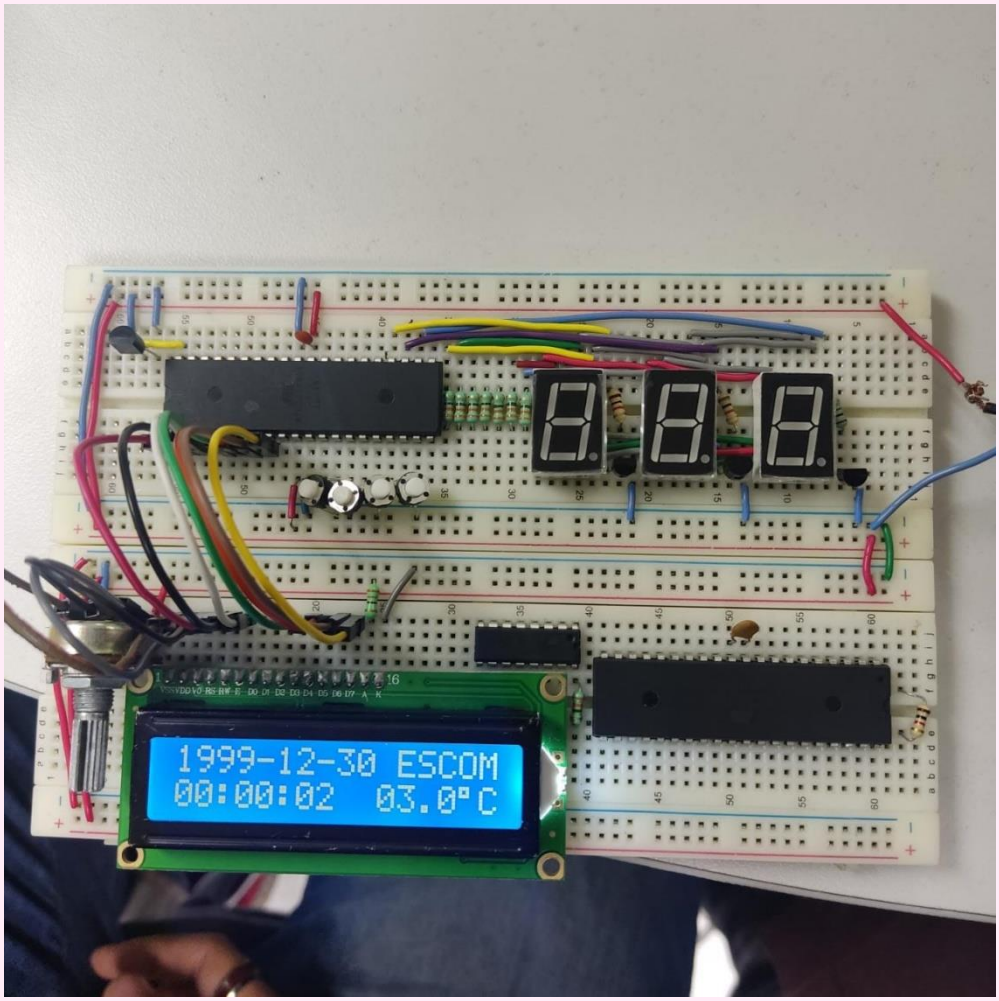

Imagen 1 Termómetro 0 a 50 °C (32 a 122 °F)

Estructura del programa

Código CodeVisionAVR

```
/*****************************************************
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    : 16/12/2022
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
*****************************************************/

// I/O Registers definitions
#include <mega8535.h>

#include <delay.h>
bit botona,botonp, botona1,botonp1, botona2,botonp2, botona3,botonp3;
int var,opcion=0;
unsigned char x,x1,a,b,c;
unsigned char an=29,ap=99,an1,an2,an3,an4,m=12,m1,m2,d=30,d1,d2;
unsigned char h=23,h1,h2,mi=59,mi1,mi2,s=50, s1,s2;
// Alphanumeric LCD functions
#include <alcd.h>

// Voltage Reference: AVCC pin
#define ADC_VREF_TYPE ((0<<REFS1) | (1<<REFS0) | (1<<ADLAR))
#define boton PIND.0
#define primero PIND.1
#define segundo PIND.2
#define tercero PIND.3

//an=2022, m=12,
```

```
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
return ADCH;
}

// Declare your global variables here

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=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=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P
PORTD=(1<<PORTD7) | (1<<PORTD6) | (1<<PORTD5) | (1<<PORTD4) | (1<<PORTD3) |
(1<<PORTD2) | (1<<PORTD1) | (1<<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);

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: ADC Stopped
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE;
```

```
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) |
(0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
SFIOR=(1<<ADHSM) | (0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

// 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);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS: PORTB Bit 0
// RD: PORTB Bit 1
// EN: PORTB Bit 2
// D4: PORTB Bit 4
// D5: PORTB Bit 5
// D6: PORTB Bit 6
// D7: PORTB Bit 7
// Characters/line: 16
lcd_init(16);

while (1)
      {
      // Place your code here

        if (boton==0)
            botona=0;
        else
            botona=1;

        if ((botona==0)&&(botonp==1)){ //hubo cambio de flanco de 1 a 0
            var++; //Se incrementa la variable
            delay_ms(40);
        }

        if ((botonp==0)&&(botona==1))
            delay_ms(40);
        botonp=botona;

        if (var==1){
```

```
            opcion=1;
        }
        if (var==2){
            var=0;
            opcion=0;
        }

        x = read_adc(0)/2.55;

        a = (x%10)^0x30;
        x = x / 10;
        b = (x%10)^0x30;
        x = x / 10;
        c = (x%10)^0x30;

        lcd_gotoxy(11,1);
        lcd_putchar(c);
        lcd_gotoxy(12,1);
        lcd_putchar(b);
        lcd_gotoxy(13,1);
        lcd_putchar(a);
        lcd_gotoxy(14,1);
        lcd_putchar(0xdf);
        lcd_gotoxy(15,1);
        lcd_putchar('C');

        an1=(an%10)^0x30;
        x1 = an / 10;
        an2=(x1%10)^0x30;
        x1 = x1 / 10;

        an3=(ap%10)^0x30;
        x1 = ap / 10;
        an4=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(0,0);
        lcd_putchar(an2);
        lcd_gotoxy(1,0);
        lcd_putchar(an1);
        lcd_gotoxy(2,0);
        lcd_putchar(an4);
        lcd_gotoxy(3,0);
        lcd_putchar(an3);
```

```
        lcd_gotoxy(4,0);
        lcd_putchar(0xb0);

        m1=(m%10)^0x30;
        x1 = m / 10;
        m2=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(5,0);
        lcd_putchar(m2);
        lcd_gotoxy(6,0);
        lcd_putchar(m1);

        lcd_gotoxy(7,0);
        lcd_putchar(0xb0);

        d1=(d%10)^0x30;
        x1 = d / 10;
        d2=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(8,0);
        lcd_putchar(d2);
        lcd_gotoxy(9,0);
        lcd_putchar(d1);

        h1=(h%10)^0x30;
        x1 = h / 10;
        h2=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(0,1);
        lcd_putchar(h2);
        lcd_gotoxy(1,1);
        lcd_putchar(h1);

        lcd_gotoxy(2,1);
        lcd_putchar(0x3a);

        mi1=(mi%10)^0x30;
        x1 = mi / 10;
        mi2=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(3,1);
```

```
        lcd_putchar(mi2);
        lcd_gotoxy(4,1);
        lcd_putchar(mi1);

        lcd_gotoxy(5,1);
        lcd_putchar(0x3a);

        s1=(s%10)^0x30;
        x1 = s / 10;
        s2=(x1%10)^0x30;
        x1 = x1 / 10;

        lcd_gotoxy(6,1);
        lcd_putchar(s2);
        lcd_gotoxy(7,1);
        lcd_putchar(s1);


        lcd_gotoxy(11,0);
        lcd_putsf("ESCOM");

        if (opcion==0){
            if (primero==0)
                botona1=0;
            else
                botona1=1;

            if ((botona1==0)&&(botonp1==1)){ //hubo cambio de flanco de 1 a
0
                ap++; //Se incrementa la variable
                delay_ms(40);
            }

            if ((botonp1==0)&&(botona1==1))
                delay_ms(40);
            botonp1=botona1;

            if (segundo==0)
                botona2=0;
            else
                botona2=1;

            if ((botona2==0)&&(botonp2==1)){ //hubo cambio de flanco de 1 a
0
                m++; //Se incrementa la variable
```

```
                delay_ms(40);
            }

            if ((botonp2==0)&&(botona2==1))
                delay_ms(40);
            botonp2=botona2;

            if (tercero==0)
                botona3=0;
            else
                botona3=1;

            if ((botona3==0)&&(botonp3==1)){ //hubo cambio de flanco de 1 a
0
                d++; //Se incrementa la variable
                delay_ms(40);
            }

            if ((botonp3==0)&&(botona3==1))
                delay_ms(40);
            botonp3=botona3;
        }

        if (opcion==1){
            if (primero==0)
                botona1=0;
            else
                botona1=1;

            if ((botona1==0)&&(botonp1==1)){ //hubo cambio de flanco de 1 a
0
                h++; //Se incrementa la variable
                delay_ms(40);
            }

            if ((botonp1==0)&&(botona1==1))
                delay_ms(40);
            botonp1=botona1;

            if (segundo==0)
                botona2=0;
            else
                botona2=1;
```

```
            if ((botona2==0)&&(botonp2==1)){ //hubo cambio de flanco de 1 a
0
                mi++; //Se incrementa la variable
                delay_ms(40);
            }

            if ((botonp2==0)&&(botona2==1))
                delay_ms(40);
            botonp2=botona2;

            if (tercero==0)
                botona3=0;
            else
                botona3=1;

            if ((botona3==0)&&(botonp3==1)){ //hubo cambio de flanco de 1 a
0
                s++; //Se incrementa la variable
                delay_ms(40);
            }

            if ((botonp3==0)&&(botona3==1))
                delay_ms(40);
            botonp3=botona3;
        }

        delay_ms(1000);

        s++;
        if(s==60){
            s=0;
            mi++;
        }
        if(mi==60){
            mi=0;
            h++;
        }
        if(h==24){
            h=0;
            d++;
        }
        if(d==31){
            d=1;
            m++;
        }
```

```
        if(m==13){
            m=1;
            ap++;
        }
        if(ap==100){
            ap=0;
            an++;
        }
        if(an==100){
            an=0;
        }


      }
}
```

Simulación – Proteus

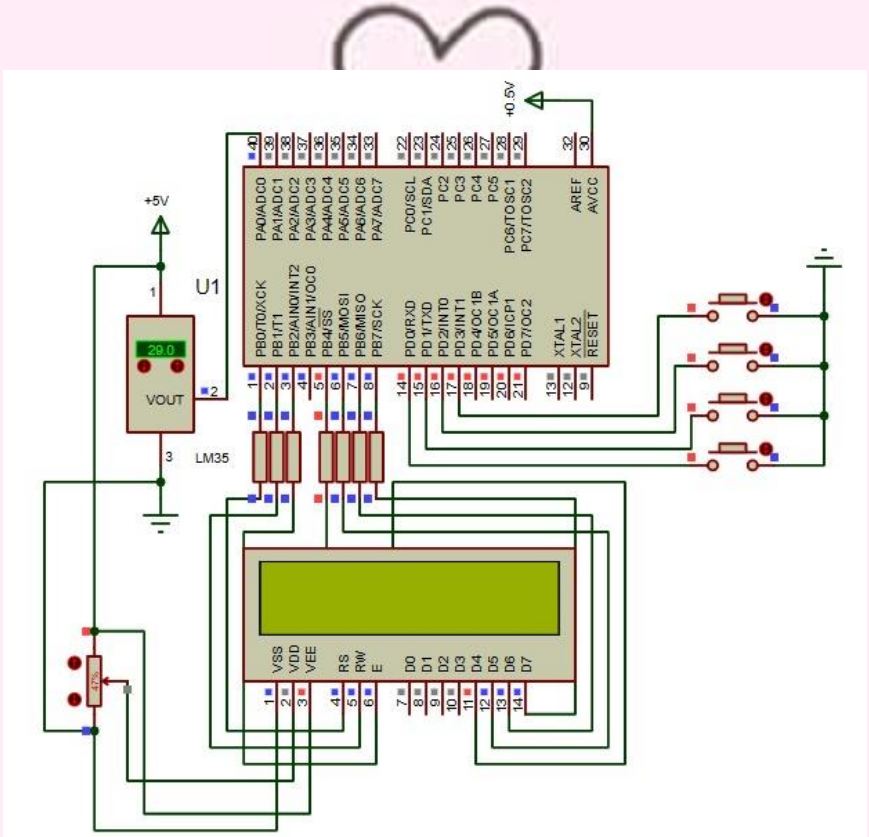

Imagen 2 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

Puedo concluir que se diseñó un programa utilizando CodeVision y el asistente correspondiente para visualizar la fecha, hora y temperatura actual en una pantalla LCD 16x2, tal como se muestra en la figura 3. Además, se utilizaron botones para realizar ajustes en la fecha y hora.

El programa se creó habilitando la pestaña de la LCD en el asistente de CodeVision y configurando los pines de conexión de acuerdo a las necesidades del usuario. El compilador se encargó de inicializar la LCD y proporcionar las funciones necesarias para mostrar la información requerida.

Este programa permite obtener y mostrar de manera eficiente la fecha, hora y temperatura actual en una pantalla LCD, brindando una interfaz amigable para el usuario y facilitando el seguimiento de estos datos importantes. El diseño del programa se basa en la configuración de los pines de conexión y el uso de las funciones provistas por el compilador, lo que simplifica el desarrollo del proyecto.

Domínguez Durán Alan Axel

Similar a la matriz de leds, en esta práctica se implementó una pantalla lcd de de 16x2 segmentos, al conectarla al microcontrolador logramos enviar información a la pantalla e imprimirla para informar al usuario de algún dato relevante que sea necesario comunicar. En este caso imprimimos la fecha y hora actual, una palabra y la temperatura. Para poder leer la temperatura, utilizamos un sensor lm35 que conectamos a una de las terminales del micro; lo cual fue un reto ya que por medio del software, tuvimos que modificar y ajustar la lectura de voltaje de entrada para que las lecturas del sensor tuvieran sentido, ya que el sensor da valores de voltaje en milivolts dependiendo del estimulo de temperatura.

Hernández Méndez Oliver Manuel

En conclusión, la práctica de construir una pantalla LCD 16x2 utilizando microcontroladores proporciona una interfaz visual y legible para mostrar información en proyectos electrónicos. La combinación de un microcontrolador programable y una pantalla LCD permite la visualización de mensajes, números y otros datos relevantes en un formato claro y conciso. Esta práctica ofrece flexibilidad en la presentación de información y puede ser utilizada en una amplia gama de aplicaciones, desde sistemas de monitoreo y control hasta dispositivos de visualización de datos en tiempo real. La pantalla LCD 16x2, junto con el microcontrolador, permite una interacción intuitiva con el usuario y mejora la experiencia de los proyectos electrónicos.

Martínez Cruz José Antonio

Retomando lo realizado en la practica 16, la practica 18 nos permite mostrar de otra manera la temperatura que se obtiene en el sensor. Aunque surgió el mismo problema con la conexión del sensor, este trabajo de manera apropiada. La parte mas extensa en esta ocasión fue la implementación del código, aunque se explicaba de manera detallada cada una de las funciones, al incio resultaba ser demasiada la información que tuvimos que analizar, pero a pesar de lo anterior logramos el objetivo solicitado.

Referencias

[1] "Interfacing 16x2 LCD with 8051 Microcontroller" por Ajay Bhargav. Este tutorial se encuentra en el sitio web CircuitDigest y proporciona una guía paso a paso sobre cómo conectar y programar una pantalla LCD 16x2 con un microcontrolador 8051. Incluye diagramas de conexión, código fuente y explicaciones detalladas.

[2] "Using an LCD with an Arduino" en el sitio web oficial de Arduino. Este recurso es un tutorial oficial proporcionado por Arduino y muestra cómo utilizar una pantalla LCD 16x2 con una placa Arduino. Ofrece instrucciones detalladas sobre la conexión física, el código de programación y la visualización de datos en la pantalla LCD.