INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Introducción a los microcontroladores

3CM16

Practica 16

Termómetro 0 a 50 °C (32 a 122 °F)

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 de hacer uso del convertidor analógico digital del microcontrolador implementando un termómetro de 0 a 50 °C (32 a 122°F).

Introducción

Termómetro 0 a 50 °C (32 a 122 °F)

Un termómetro es un dispositivo utilizado para medir la temperatura. En este caso, se trata de un termómetro que opera en un rango de temperatura de 0 a 50 °C (32 a 122 °F) y está diseñado utilizando microcontroladores.

Los microcontroladores son circuitos integrados programables que contienen una unidad central de procesamiento (CPU), memoria y periféricos de entrada/salida en un solo chip[1]. Estos componentes son utilizados para controlar y monitorear sistemas electrónicos de manera eficiente[2].

En el caso del termómetro, el microcontrolador se encarga de procesar las señales provenientes del sensor de temperatura, que podría ser un termistor o un sensor de temperatura digital, para convertirlas en valores de temperatura legibles. También puede tener la capacidad de mostrar la temperatura en una pantalla LCD o enviarla a otros dispositivos a través de interfaces de comunicación como UART o USB.

El rango de temperatura de 0 a 50 °C (32 a 122 °F) es comúnmente utilizado en aplicaciones domésticas, industriales y científicas [3]. El diseño con microcontroladores permite una mayor precisión en la medición de la temperatura y ofrece flexibilidad en la visualización y procesamiento de los datos obtenidos.

En resumen, un termómetro 0 a 50 °C (32 a 122 °F) hecho con microcontroladores es un dispositivo electrónico que utiliza un microcontrolador para medir y mostrar la temperatura en un rango específico. Esto brinda mayor precisión y funcionalidad en comparación con los termómetros convencionales, lo que lo hace adecuado para una variedad de aplicaciones.

Material y equipo empleado

- ♥ CodeVision AVR
- ♥ AVR Studio 4
- ♥ Microcontrolador ATmega 8535
- ♥ 2 Display Cátodo común
- ♥ 8 Resistores de 330 Ω a 1⁄4 W
- ♥ 3 Resistores de 1K Ω a 1⁄4 W
- ♥ 3 Transistores BC547 o 2N222.

Desarrollo experimental

Diseñe junto con el siguiente circuito, un termómetro que trabaje en el rango de 0 a 50 °C (32 a 122°F).

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.

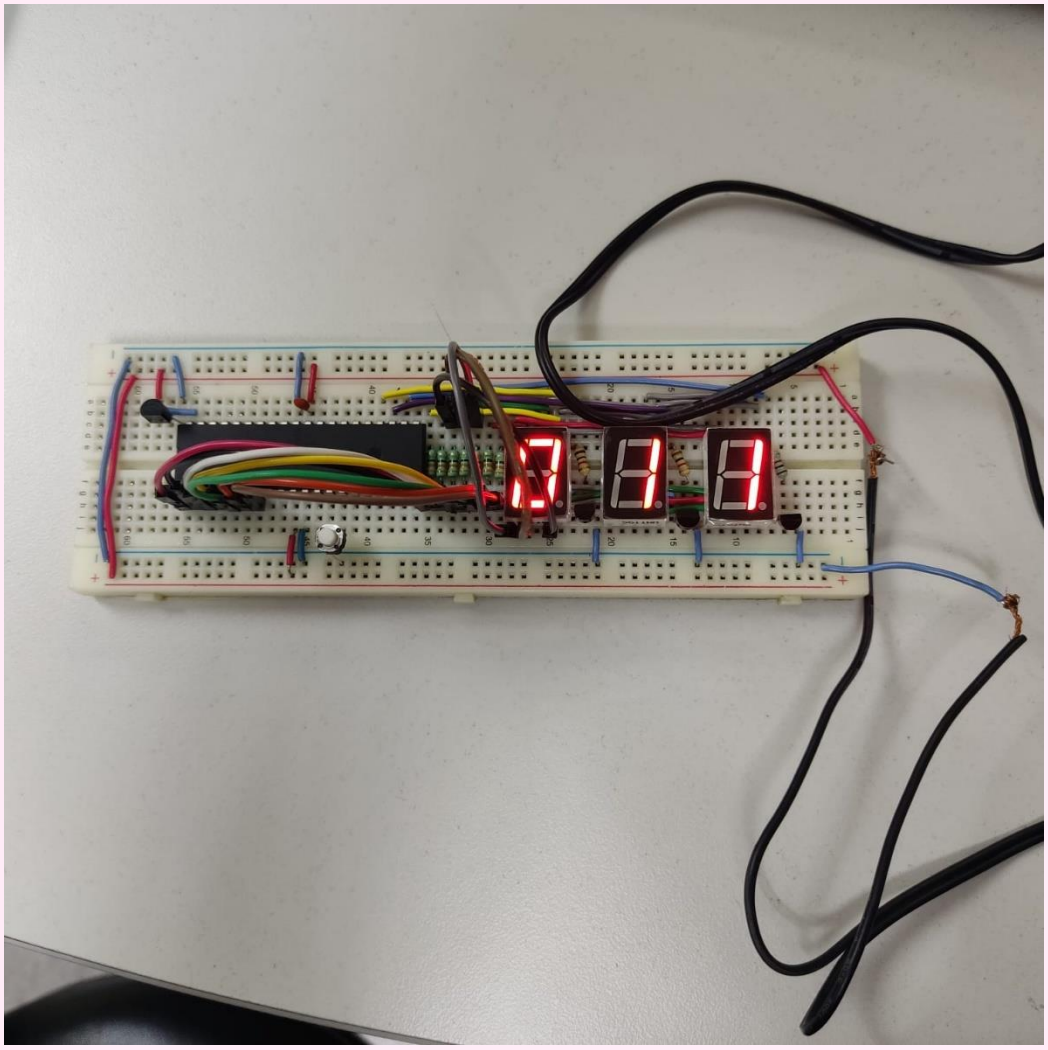

Imagen 1 Termómetro 0 a 50 °C (32 a 122 °F)

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    : 07/11/2022
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>
#define boton PIND.0
bit botona, botonp=1;
unsigned char x;
int cn, f, var, opcion=0;
const char tabla7segmentos
[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7c,0x07,0x7f,0x6f};

// Voltage Reference: Int., cap. on AREF
#define ADC_VREF_TYPE ((1<<REFS1) | (1<<REFS0) | (1<<ADLAR))

// 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=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=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: Int., cap. on AREF
// 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);

while (1)
      {
      // Place your code here
        x = read_adc(0);
        cn = (x*110)/255;
        f = (1.8*cn)+32;

        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;
        }


        if (opcion==0){
            PORTC = 0x00;
            PORTB = tabla7segmentos[cn/100];
            PORTC = 0x04;
            delay_ms(1);

            PORTC = 0x00;
            PORTB = tabla7segmentos[cn/10];
```

```
            PORTC = 0x02;
            delay_ms(1);

            PORTC = 0x00;
            PORTB = tabla7segmentos[cn%10];
            PORTC = 0x01;
            delay_ms(1);
        }

        if (opcion==1){
            PORTC = 0x00;
            PORTB = tabla7segmentos[f/100];
            PORTC = 0x04;
            delay_ms(1);

            PORTC = 0x00;
            PORTB = tabla7segmentos[f/10];
            PORTC = 0x02;
            delay_ms(1);

            PORTC = 0x00;
            PORTB = tabla7segmentos[f%10];
            PORTC = 0x01;
            delay_ms(1);
        }


      }
}
```

Simulación – Proteus

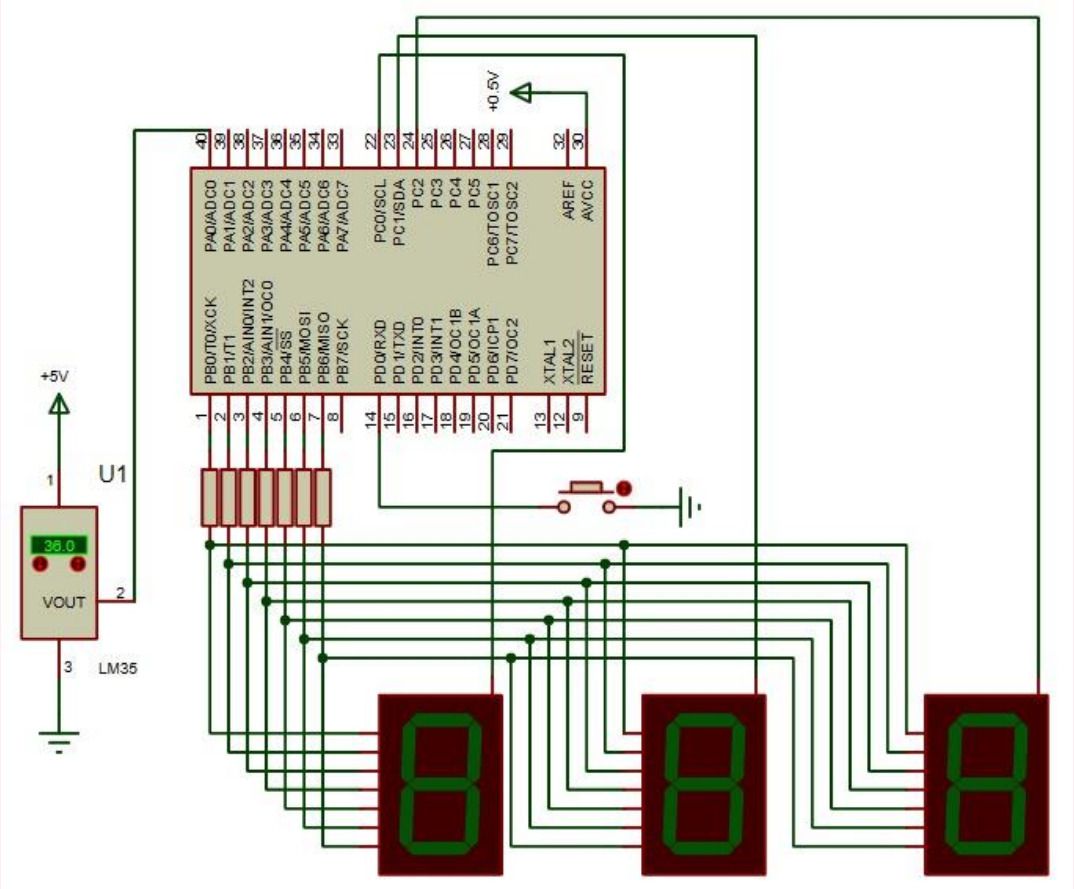

Imagen 2 Circuito simulado en proteus

Conclusiones

Bocanegra Heziquio Yestlanezi

En conclusión, durante la sesión, adquirimos las habilidades necesarias para utilizar el convertidor analógico-digital del microcontrolador ATmega 8535 y desarrollaron un termómetro que puede medir temperaturas en el rango de 0 a 50 °C (32 a 122°F). Para lograrlo, se utilizaron herramientas como CodeVision AVR y AVR Studio 4. Además, se emplearon los siguientes materiales y equipo: 3 display de cátodo común, 8 resistores de 330 Ω a 1/4 W, 3 resistores de 1K Ω a 1/4 W y 3 transistores BC547 o 2N2222.

Pudimos aprovechar eficientemente estas herramientas y componentes para implementar un sistema funcional de medición de temperatura. Al finalizar la sesión, nos encontramos capacitados para aplicar estas habilidades en futuros proyectos que requieran el uso del convertidor analógico-digital y la medición precisa de temperaturas. Este logro representa un avance significativo en el conocimiento y las capacidades técnicas del equipo en el campo de la electrónica y el control de microcontroladores.

Domínguez Durán Alan Axel

Esta práctica es exactamente igual a la práctica 15 (volmetro), sin embargo, en esta ocasión la entrada del microcontrolador es la de un sensor de temperatura. Desafortunadamente no pudimos acondicionar el sensor, pero por medio del software acoplamos la entrada para poder medir correctamente la temperatura del termómetro.

Hernández Méndez Oliver Manuel

En mi opinión, la construcción de un termómetro utilizando microcontroladores permite obtener mediciones precisas y flexibilidad en el procesamiento de datos, lo que resulta beneficioso en una amplia gama de aplicaciones donde se requiere un monitoreo y control de temperatura confiable.

Martínez Cruz José Antonio

Existieron varios problemas con la implementación del sensor de temperatura dentro de esta practica ya que este se sobrecalentaba demasiado o no enviaba nada en su pin de salida. Después de volver armar el circuito se soluciono este detalle y realizaba de manera correcta la toma de temperatura mostrando de manera ascendente y sin realizar brincos. Solo quedo por verificar la condición al momento en que el microprocesador alcanzaba el tope de temperatura, el cual fue solucionado posteriormente.

Referencias

[1] "Design and Implementation of Digital Thermometer using Microcontroller" por R. P. Joshi, D. D. Gorad and S. D. Ghodake. Puedes encontrar este artículo en la revista International Journal of Computer Applications (IJCA). Proporciona detalles sobre el diseño y la implementación de un termómetro digital utilizando un microcontrolador.

[2] "Temperature Monitoring and Control System using Microcontroller" por A. H. M. Razibul Islam, S. M. Towfiqul Islam y A. A. M. Shafiullah. Este artículo, publicado en la revista International Journal of Scientific and Research Publications (IJSRP), describe un sistema de monitoreo y control de temperatura utilizando un microcontrolador.

[3] "Digital Temperature Sensor Design using Microcontroller" por T. M. Praveen, G. Geetha y P. Rajalakshmi. Este artículo se encuentra en la revista International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering (IJIREEICE). Presenta el diseño de un sensor de temperatura digital utilizando un microcontrolador.