INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 16: TERMÓMETRO 0 A 50 °C (32 A 122 °F)

ALUMNOS: MALAGON BAEZA ALAN ADRIAN
MARTINEZ CHAVEZ JORGE ALEXIS

GRUPO: 6CM3

U.A: SISTEMAS EN CHIP

PROFESOR: FERNANDO AGUILAR SÁNCHEZ

FECHA DE ENTREGA: 18 DE JUNIO DE 2023

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 TEÓRICA

TERMISTOR

Un termistor es un elemento de detección de temperatura compuesto por un material semiconductor sinterizado que exhibe un gran cambio en la resistencia en respuesta a un pequeño cambio en la temperatura. Los termistores normalmente tienen coeficientes de temperatura negativos lo que significa que la resistencia del termistor disminuye a medida que la temperatura aumenta.[1]

Los termistores son uno de los tipos de sensores de temperatura más precisos. Los termistores tienen una precisión de ±0.1 °C o ±0.2 °C dependiendo del modelo de termistor en particular. Sin embargo, los termistores son bastante limitados en su rango de temperatura, y trabajan sólo en un rango nominal de 0 a 100 °C.[1]

Los termistores terminados son químicamente estables y no resultan afectados significativamente por el envejecimiento.[1]

Figura 1. Termistor comercial NTC 1.5D

TERMISTOR LM35

El LM35 es un circuito electrónico sensor que puede medir temperatura. Su salida es analógica, es decir, te proporciona un voltaje proporcional a la temperatura. El sensor tiene un rango desde −55°C a 150°C. Su popularidad se debe a la facilidad con la que se puede medir la temperatura. Incluso no es necesario de un microprocesador o microcontrolador para medir la temperatura. Dado que el sensor LM35 es analógico, basta con medir con un multímetro, el voltaje a salida del sensor.[2]

Para convertir el voltaje a la temperatura, el LM35 proporciona 10mV por cada grado centígrado. También cabe señalar que ese sensor se puede usar sin offset, es decir que, si medimos 20mV a la salida, estaremos midiendo 2°C.[2]

CARACTERÍSTICAS PRINCIPALES DEL LM35

- **Resolución:** 10mV por cada grado centígrado.

- **Voltaje de alimentación**. Por ejemplo, esté sensor se puede alimentar desde 4Vdc hasta 20Vdc.
- **Tipo de medición**. Salida analógica.
- **Numero de pines:** 3 pines, GND, VCC y VSalida.
- **No requiere calibración.**
    - Tiene una precisión de ±¼°C.
    - Esta calibrado para medir °C.
- **Consumo de corriente:** 60 µA
- **Empaquetados comunes:**
    - TO-CAN.
    - TO-220.
    - TO-92.
    - SOIC8.
- **El pinout del sensor de temperatura son tres**: GND, VCC y VSalida. Entonces dependiendo del empaquetado será el orden de conexión de los pines.

CIRCUITO RECOMENDABLE PARA SU USO

La simplicidad del circuito hace que sea muy fácil ser utilizado en alguna aplicación embebida. Entonces sólo basta alimentarlo con digamos 5VDC, conectar la tierra GND a la tierra del circuito digital y la salida de voltaje a la entrada del ADC.[2]

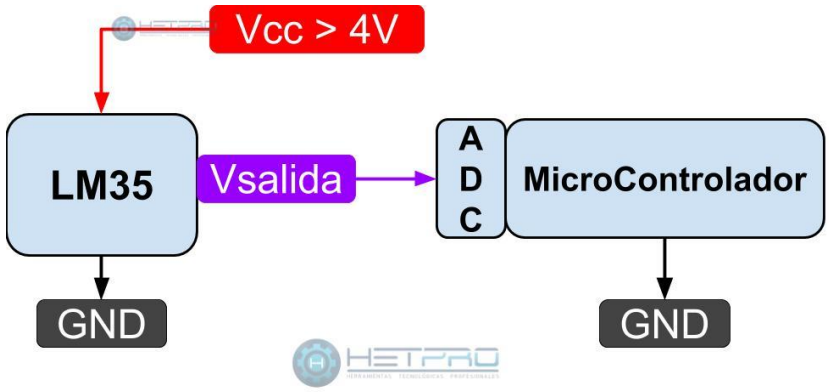


Figura 2. Circuito recomendable para el uso del LM35

MATERIALES Y EQUIPO EMPLEADO

- ✓ CodeVision AVR
- ✓ AVR Studio 4
- ✓ Microcontrolador ATmega 8535
- ✓ 3 Display Ánodo Común
- ✓ 8 Resistores de 330 Ω a 1⁄4 W
- ✓ 1 Push Botón
- ✓ 1 Termistor LM35

DESARROLLO EXPERIMENTAL

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

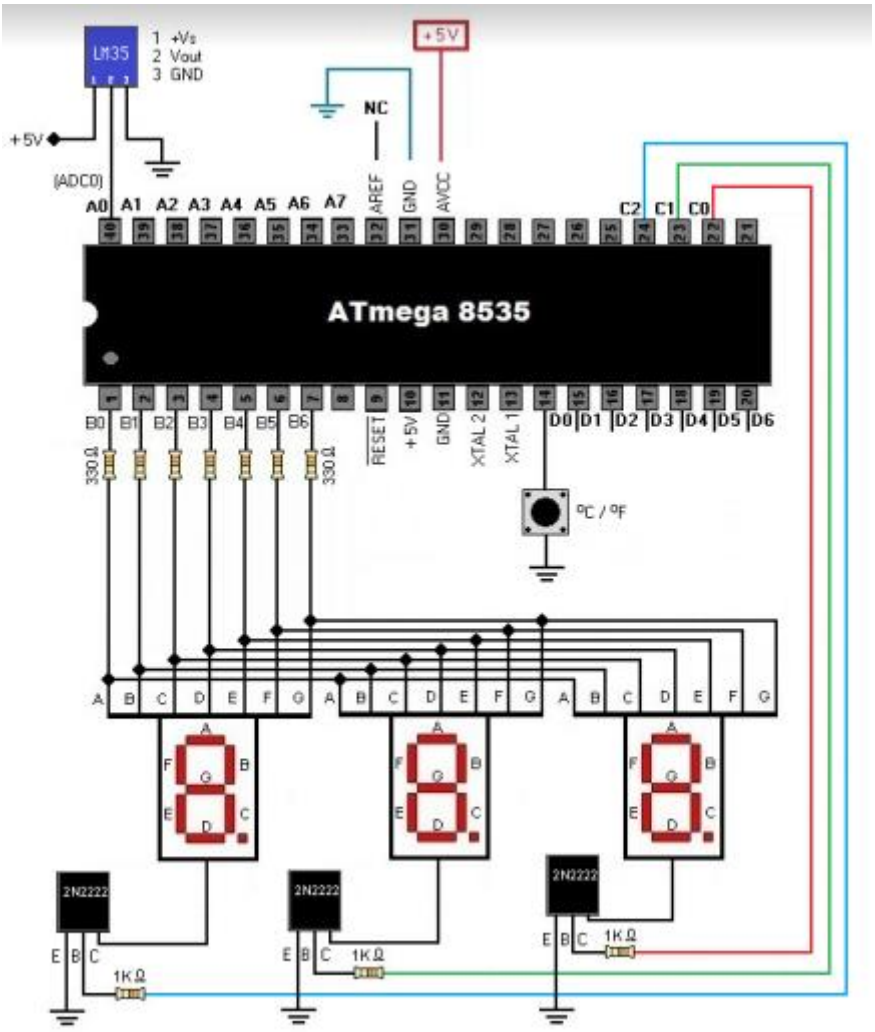


Figura 3. Circuito para el Termómetro

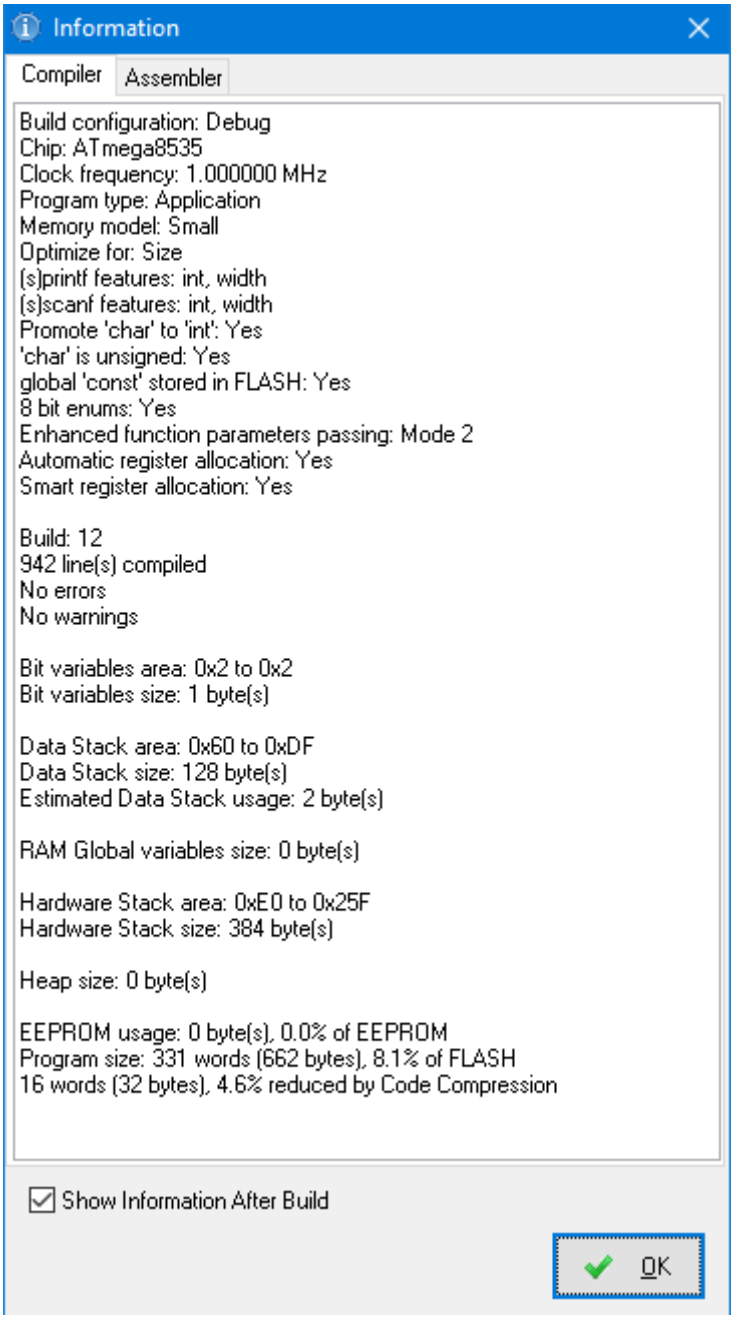


Figura 4. Compilación exitosa del código en CodeVision

CÓDIGO GENERADO POR CODEVISION

```
/*****************************************************
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    :
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 PIND.0

// Voltage Reference: AVCC pin
#define ADC_VREF_TYPE ((0<<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
unsigned char intensidad, valor;
bit isFahrenheit = 1;
bit pasado, actual;
const char mem[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D,
0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};

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: 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);

while (1)
      {
        valor = read_adc(0);
        actual = btn;

        //Boton de modo
        if (pasado == 1 && actual == 0) {
            isFahrenheit = ~isFahrenheit;
            delay_ms(20);
        }

        if (pasado == 0 && actual == 1) {
            delay_ms(20);
        }


        if (isFahrenheit) {
            intensidad = 9*(50 * valor / 255)/5 + 32;
            delay_ms(7);
            PORTB = ~mem[intensidad/100];
```

```
            PORTC = 0x01;

            delay_ms(7);
            PORTB = ~mem[(intensidad%100 - intensidad%10)/10];
            PORTC = 0x02;

            delay_ms(7);
            PORTB = ~mem[intensidad%10];
            PORTC = 0x04;

        } else {
            intensidad = 50 * valor / 255;
            delay_ms(7);
            PORTB = ~mem[0];
            PORTC = 0x01;

            delay_ms(7);
            PORTB = ~mem[intensidad/10];
            PORTC = 0x02;

            delay_ms(7);
            PORTB = ~mem[intensidad%10];
            PORTC = 0x04;
        }

        pasado = actual;
      }
}
```

CIRCUITO ELECTRICO EN PROTEUS

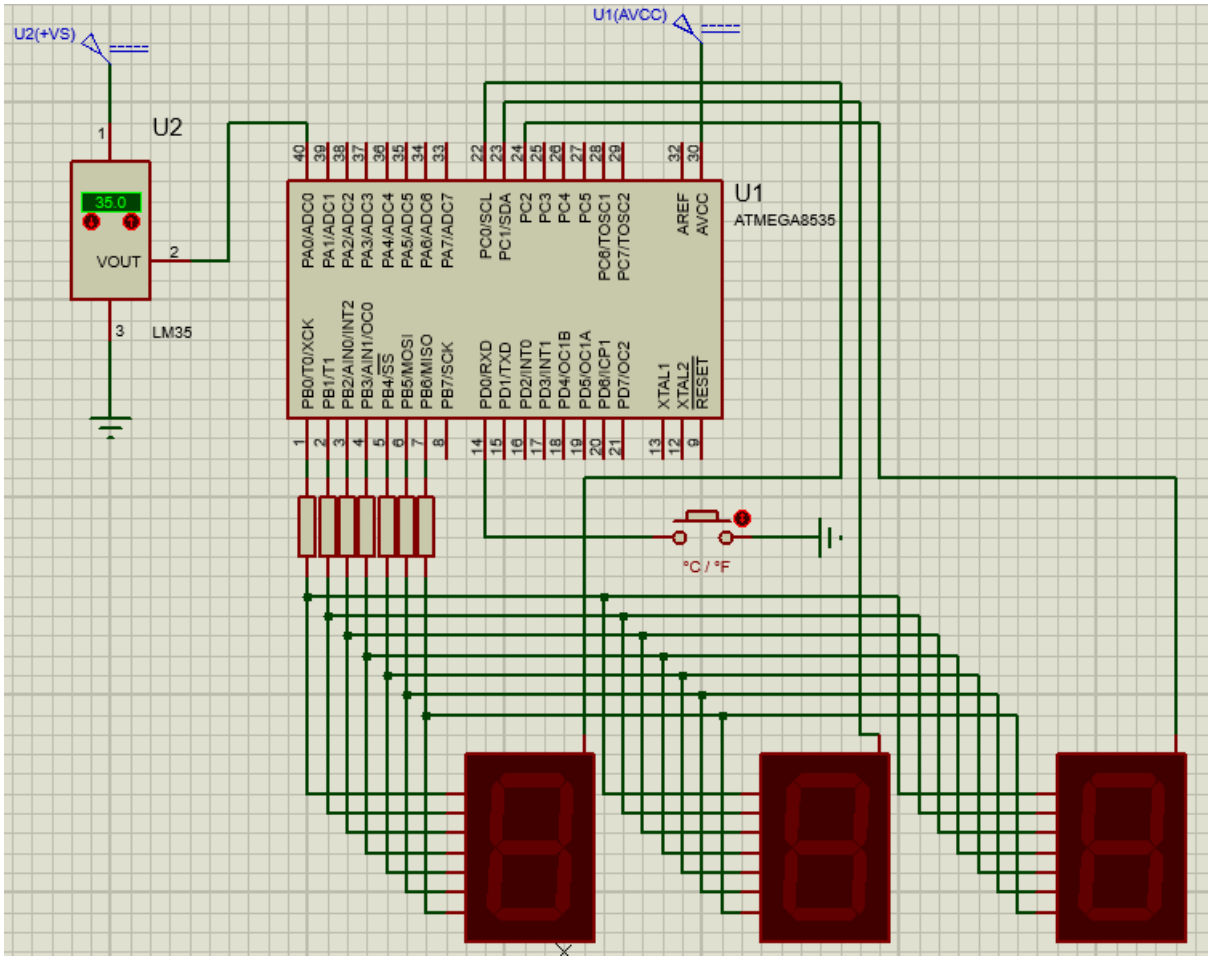
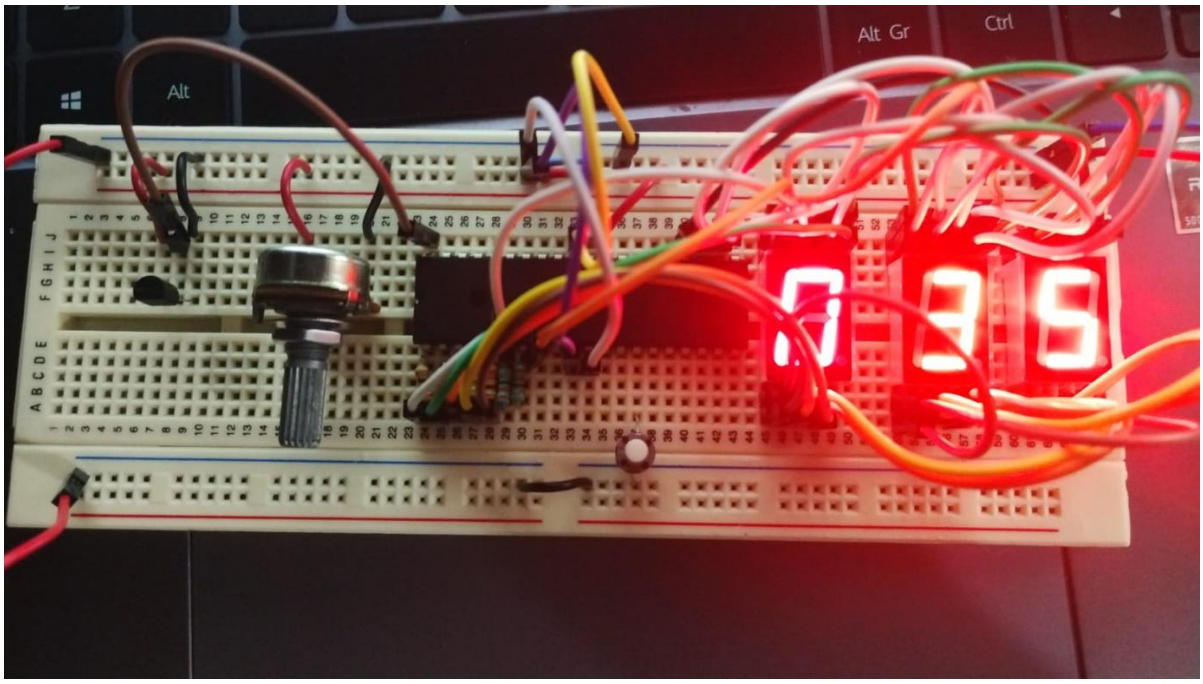
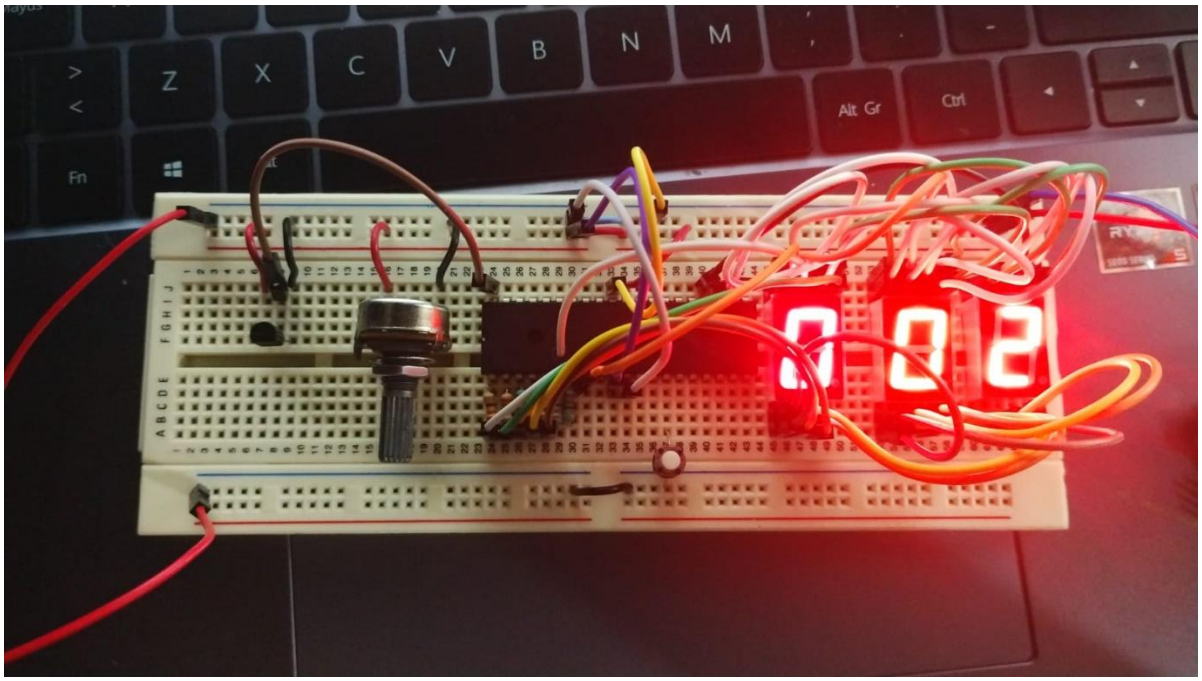


Figura 5. Circuito simulado en Proteus

CIRCUITO EN EL PROTO ARMADO





OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrian

Para desarrollar la práctica se reutilizó el código y el circuito armado para la práctica del vólmetro de 0.0 a 30.0V, realizando los cambios necesarios para los nuevos objetivos.

En la práctica se utilizó un termistor muy utilizado en las prácticas de electrónica desarrolladas previamente en la trayectoria escolar. El termistor es el LM35, y este termistor es ideal debido a que está perfectamente caracterizado para poder entregar la medida de la temperatura (en °C) en voltaje analógico, de tal manera que sólo hay que hacer la conversión con el ADC para poder interpretar la información y mostrarla en los displays.

Martínez Chávez Jorge Alexis

Lo que se tuvo que editar en el circuito fue que se agregó el LM35 en la entrada del ADC de nuestro microcontrolador para medir y convertir el voltaje de entrada. Además de que se agregó un push botón para poder cambiar entre el modo °C y °F.

Para el desarrollo del código se tuvo que poner la lógica para identificar la pulsación del push botón y cambiar una variable de tipo bit para cambiar entre los modos. Una vez identificado el modo se procede a mostrar la información en los displays, teniendo en cuenta que cuando se seleccione el modo °F hay que sumar 32 unidades al valor obtenido por el LM35 (recordando que este termistor mide la temperatura en °C).

BIBLIOGRAFÍA

[1] OMEGA Engineering Inc., “Termistor,” *mx.omega.com*, 2022. https://mx.omega.com/prodinfo/termistor.html (accessed May 11, 2022).
[2] HETPRO/TUTORIALES, “LM35 - El sensor de temperatura más popular,” *HETPRO/TUTORIALES*, Nov. 25, 2017. https://hetpro-store.com/TUTORIALES/lm35/ (accessed May 11, 2022).