A picture containing icon

Description automatically generatedLogo

Description automatically generatedINSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

**PRÁCTICA 15: VÓLMETRO DE 0.0 A 5.0 Y DE 0.0 A 30.0 VOLTS**

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 Vólmetro de 0.0 a 5.0 Volts mostrado en dos displays de siete segmentos multiplexados.

**INTRODUCCIÓN TEÓRICA**

**VOLTÍMETRO**

Se llama voltímetro al dispositivo que permite realizar la medición de la diferencia de potencial o tensión que existe entre dos puntos pertenecientes a un circuito eléctrico. El voltímetro, por lo tanto, revela el voltaje (la cantidad de voltios).1

Los voltímetros tienen que contar con una resistencia eléctrica elevada para que, al ser conectados al circuito para realizar la medición, no generen un consumo que lleve a medir la tensión de manera errónea.1



Figura 1. Voltímetro digital

**DIFERENCIA ENTRE VOLTÍMETRO, VOLTÁMETRO Y AMPERÍMETRO**

Es importante no confundir el voltímetro con el voltámetro. Mientras que el voltímetro mide la tensión (el potencial eléctrico), el voltámetro se encarga de la medición de la carga eléctrica. Los voltímetros reflejan sus resultados en voltios y los voltámetros lo hacen en coulombs.1

Tampoco hay que tener confusión entre el voltímetro y el amperímetro, que es la herramienta empleada para la medición de la intensidad de la corriente. El voltímetro, para funcionar, tiene que conectarse en paralelo: el amperímetro, en cambio, en serie, para que la corriente pase por él. Su resistencia, de este modo, debe ser reducida, y no alta como ocurre en el caso del voltímetro.1

**UTILIDAD DE UN VOLTÍMETRO**

Existen versiones digitales del voltímetro que se usan en la actualidad y presentan una capacidad de aislamiento muy elevada gracias al uso de circuitos específicos de gran complejidad.1

Cuando deseamos medir tensiones que superan los límites de un voltímetro (en el caso de los digitales, el límite de sus circuitos electrónicos; en el de los demás, de sus órganos mecánicos y devanados), debemos recurrir a una resistencia e alto valor y colocarla en serie con éste, de manera que simplemente les llegue una porción de la tensión total.1

Toda persona interesada en la reparación o el mantenimiento de sus aparatos electrónicos debe contar con un voltímetro, ya que es una herramienta fundamental para realizar mediciones preliminares, así como para comprobar que los trabajos hayan resultado como esperaban.1

**DIVISOR DE VOLTAJE**

Un divisor de voltaje es un circuito simple que reparte la tensión de una fuente entre una o más impedancias conectadas. Con sólo dos resistencias en serie y un voltaje de entrada, se puede obtener un voltaje de salida equivalente a una fracción del de entrada. En otras palabras, su utilidad está en poder cambiar un voltaje fijo a voluntad o establecer un voltaje de referencia para otros circuitos. Permite calcular el voltaje de cada resistencia (caída de voltaje).2

Para tener un divisor de voltaje es necesario tener dos o más resistencias en serie, los componentes están conectados en una secuencia, mediante sus terminales (la salida de un componente se conecta a la entrada del siguiente, y así sucesivamente).2

Para obtener el voltaje en la resistencia deseada (VRx) se multiplica el voltaje de la fuente (Vf) por la resistencia deseada (Rx), dividido entre la resistencia total (Rt, que es igual a la suma de todas las resistencias).2

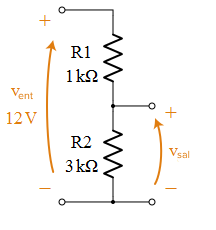


Figura 2. Ejemplo de un circuito divisor de voltaje

**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

**DESARROLLO EXPERIMENTAL**

1. Diseñe un programa para mostrar en dos Displays voltajes entre 0.0V a 5.0 V y otro programa para mostrar en tres Displays voltajes entre 0.0V a 30.0V.

Diagram, schematic

Description automatically generated

Figura 3. Circuito para el Voltímetro de 0.0V a 5.0V

Text

Description automatically generated with medium confidence

Figura 4. Compilación exitosa del código para el vólmetro de 0.0 a 5.0V

Text

Description automatically generated

Figura 5. Compilación exitosa del código para el vólmetro de 0.0 a 30.0V

**CÓDIGO GENERADO POR CODEVISION**

* **VOLTÍMETRO DE 0.0V A 5.0V**

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  This program was created by the CodeWizardAVR V3.48b  Automatic Program Generator  © Copyright 1998-2021 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>  // 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  **const** **char** mem[**11**] = {**0x3F**, **0x06**, **0x5B**, **0x4F**, **0x66**, **0x6D**, **0x7D**, **0x07**, **0x7F**, **0x6F**, **0x6F**};  **unsigned** **char** valor, intensidad;  **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=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);  // 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**);  intensidad = **50** \* valor / **255**;    delay\_ms(**7**);  PORTB = ~mem[intensidad/**10**];  PORTC = **0x02**;    delay\_ms(**7**);  PORTB = ~mem[intensidad%**10**];  PORTC = **0x04**;  }  } |

* **VOLTÍMETRO DE 0.0V A 30.0V**

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  This program was created by the CodeWizardAVR V3.48b  Automatic Program Generator  © Copyright 1998-2021 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>  // 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  **const** **char** mem[**16**] = {**0x3F**, **0x06**, **0x5B**, **0x4F**, **0x66**, **0x6D**, **0x7D**, **0x07**, **0x7F**, **0x6F**, **0x77**, **0x7C**, **0x39**, **0x5E**, **0x79**, **0x71**};  **unsigned** **long** valor, intensidad;  **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=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);  // 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**);  intensidad = **300** \* valor / **255**;    delay\_ms(**5**);  PORTB = ~mem[intensidad/**100**];  PORTC = **0x01**;    delay\_ms(**5**);  PORTB = ~mem[intensidad%**100**/**10**];  PORTC = **0x02**;    delay\_ms(**5**);  PORTB = ~mem[intensidad%**10**];  PORTC = **0x04**;  }  } |

**CIRCUITO ELECTRICO EN PROTEUS**

Diagram

Description automatically generated with low confidence

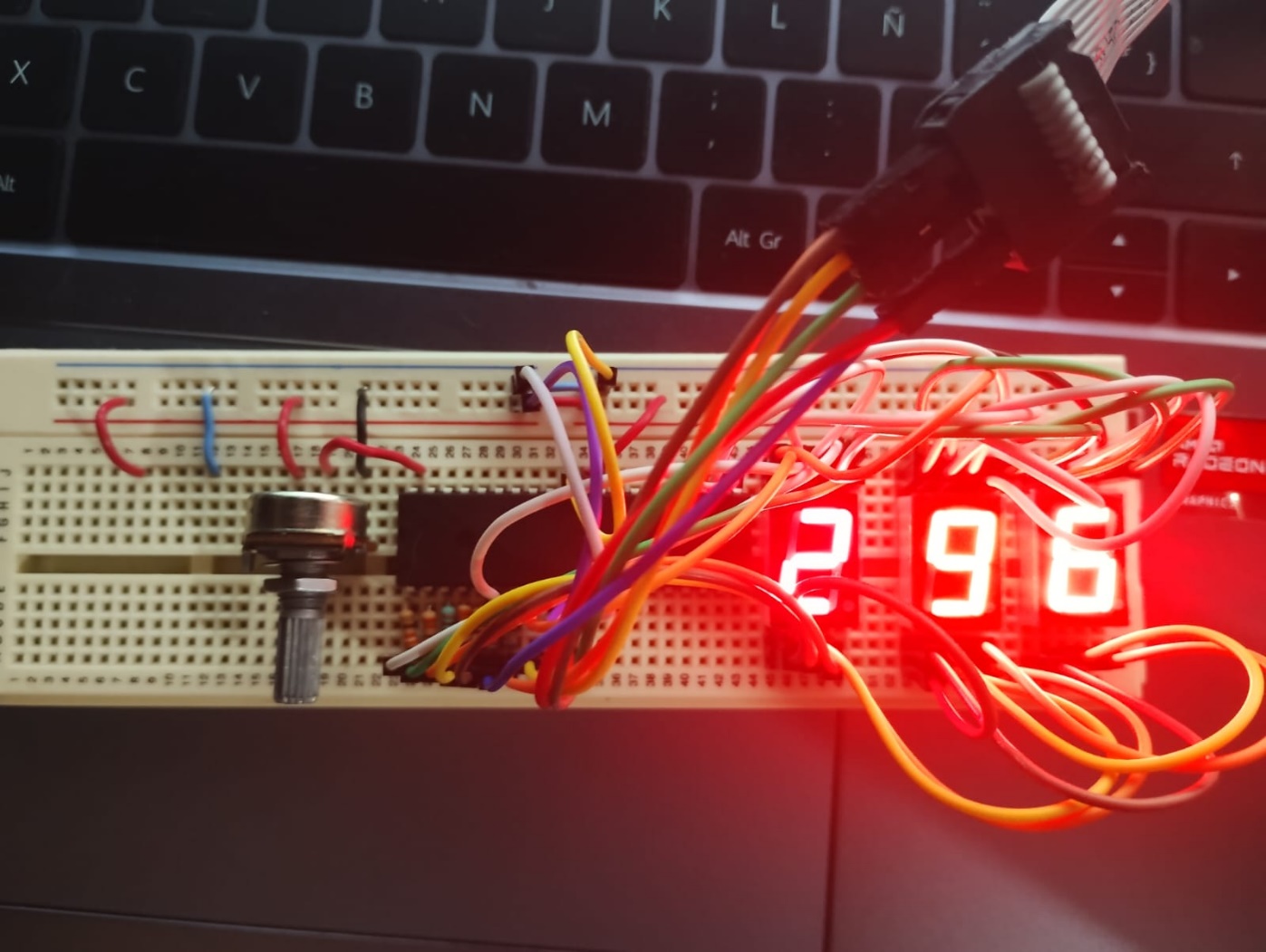
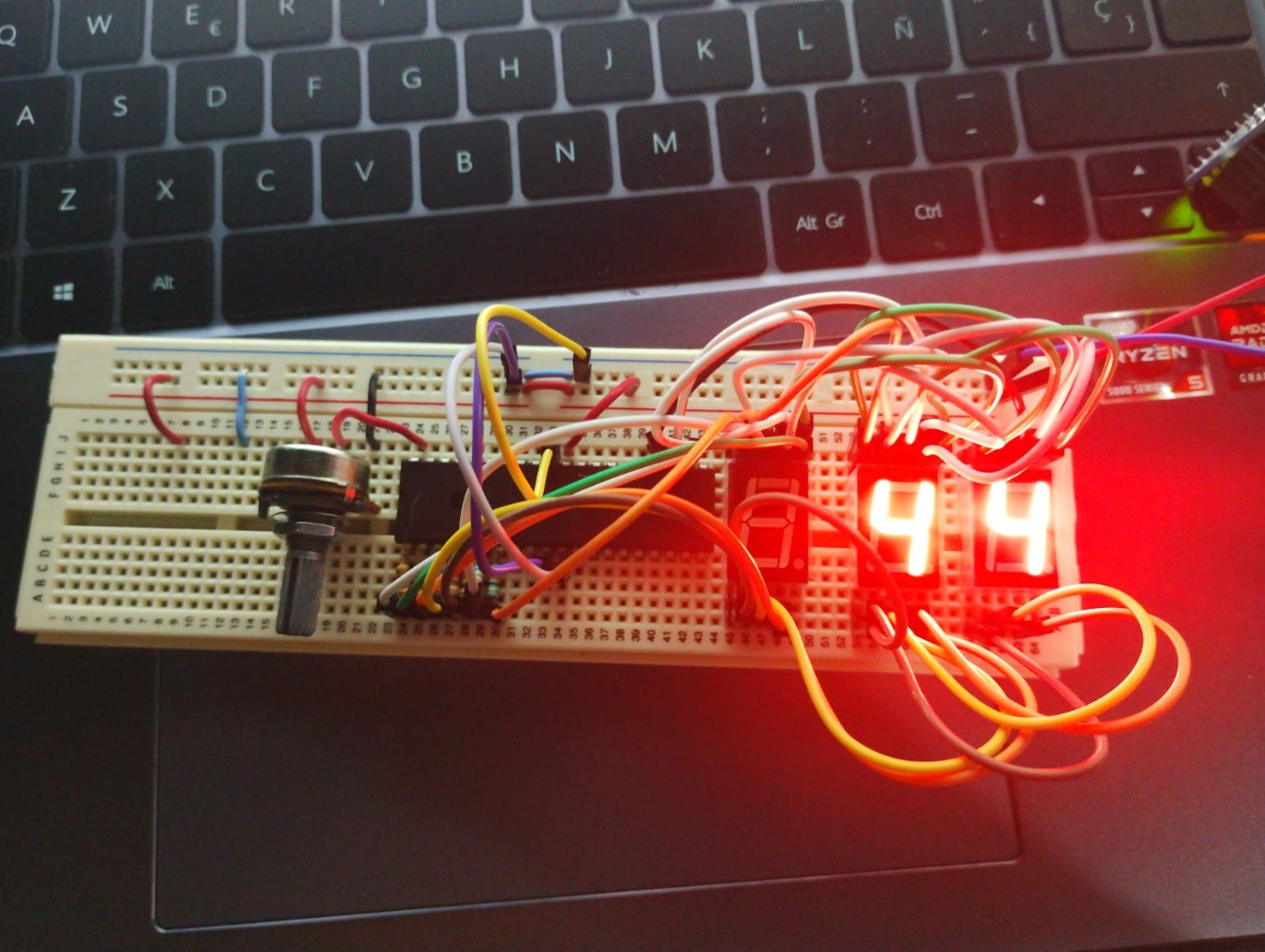
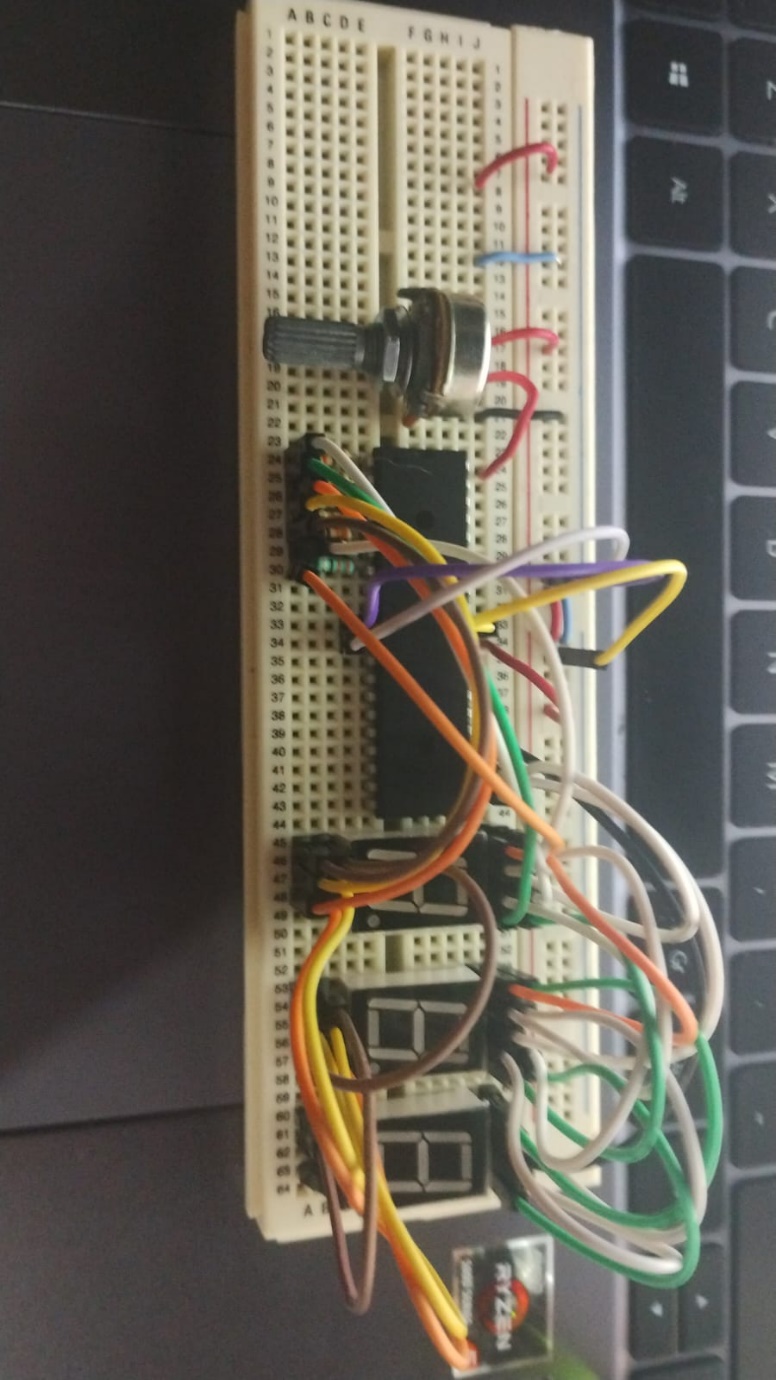
Figura 6. Circuito simulado para el vólmetro de 0.0 a 5.0V

Chart

Description automatically generated

Figura 7. Circuito simulado para el vólmetro de 0.0 a 30.0V

**CIRCUITO EN EL PROTO ARMADO**

****

**OBSERVACIONES Y CONCLUSIONES INDIVIDUALES**

Malagón Baeza Alan Adrian

El vólmetro nos sirve para poder medir la diferencia de voltaje entre dos puntos. Comercialmente lo podemos encontrar como un multímetro debido a que estos dispositivos incluyen, entre otras cosas, la medición del voltaje.

Para poder desarrollar un vólmetro con ayuda de nuestro microcontrolador ATMEGA8535 se utilizó el ADC interno para realizar la lectura del voltaje de entrada y posteriormente mostrarlo con ayuda de displays conectados a las salidas del microcontrolador.

La complejidad para desarrollar esta práctica estuvo en el circuito simulado en Proteus. Se tuvo que implementar un circuito divisor de voltaje para el caso del vólmetro de 0.0 a 30.0V para poder representar ese rango de voltaje en un rango menor para el ADC.

Martínez Chávez Jorge Alexis

Para el desarrollo del código para los vólmetros se utilizaron las fórmulas vistas en clase, de tal manera que la lectura del ADC del microcontrolador fuera interpretada para poder ser mostrada en los displays.

Esta práctica no fue tan complicada como lo esperaba, sin embargo, si requerí de un tiempo para analizar las fórmulas vistas en clase y poder desarrollar el código y el circuito para que funcionaran correctamente.

**BIBLIOGRAFÍA**

[1] Definicion.de, “Definición de voltímetro — Definicion.de,” *Definición.de*, 2022. <https://definicion.de/voltimetro/> (accessed May 11, 2022).

[2] N. de Anda, “Divisor de voltaje • Factor Evolución,” *Factor Evolución*, Oct. 12, 2018. <https://www.factor.mx/portal/base-de-conocimiento/divisor-de-voltaje/> (accessed May 11, 2022).