INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 18: PANTALLA LCD 16x2

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 para manejar una pantalla LCD.

INTRODUCCIÓN TEÓRICA

PANTALLA LCD

Una pantalla de cristal líquido o LCD (sigla del inglés liquid-crystal display) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.[1]

Cada píxel de un LCD típicamente consiste en una capa de moléculas alineadas entre dos electrodos transparentes, y dos filtros de polarización, los ejes de transmisión de cada uno que están (en la mayoría de los casos) perpendiculares entre sí. Sin cristal líquido entre el filtro polarizante, la luz que pasa por el primer filtro sería bloqueada por el segundo (cruzando) polarizador.[1]

Figura 1. Pantalla LCD de 4x20 comercial

La superficie de los electrodos que están en contacto con los materiales de cristal líquido es tratada a fin de ajustar las moléculas de cristal líquido en una dirección en particular. Este tratamiento suele ser normalmente aplicable en una fina capa de polímero que es unidireccionalmente frotada utilizando, por ejemplo, un paño. La dirección de la alineación de cristal líquido se define por la dirección de frotación.[1]

Antes de la aplicación de un campo eléctrico, la orientación de las moléculas de cristal líquido está determinada por la adaptación a las superficies. En un dispositivo twisted nematic, TN (uno de los dispositivos más comunes entre los de cristal líquido), las direcciones de alineación de la superficie de los dos electrodos son perpendiculares entre sí, y así se organizan las moléculas en una estructura helicoidal, o retorcida. Debido a que el material es de cristal líquido birrefringente, la luz que pasa a través de un filtro polarizante se gira por la hélice de cristal líquido que pasa a través de la capa de cristal líquido, lo que le permite pasar por el segundo filtro polarizado. La mitad de la luz incidente es absorbida por el primer filtro polarizante, pero por lo demás todo el montaje es transparente.[1]

Cuando se aplica un voltaje a través de los electrodos, una fuerza de giro orienta las moléculas de cristal líquido paralelas al campo eléctrico, que distorsiona la estructura helicoidal (esto se puede resistir gracias a las fuerzas elásticas desde que las moléculas están limitadas a las

superficies). Esto reduce la rotación de la polarización de la luz incidente, y el dispositivo aparece gris. Si la tensión aplicada es lo suficientemente grande, las moléculas de cristal líquido en el centro de la capa son casi completamente desenrolladas y la polarización de la luz incidente no es rotada ya que pasa a través de la capa de cristal líquido. Esta luz será principalmente polarizada perpendicular al segundo filtro, y por eso será bloqueada y el pixel aparecerá negro. Por el control de la tensión aplicada a través de la capa de cristal líquido en cada píxel, la luz se puede permitir pasar a través de distintas cantidades, constituyéndose los diferentes tonos de gris.[1]

Figura 2. Pantalla LCD en un despertador.

Cuando un dispositivo requiere un gran número de píxeles, no es viable conducir cada dispositivo directamente, así cada píxel requiere un número de electrodos independiente. En cambio, la pantalla es multiplexada. En una pantalla multiplexada, los electrodos de la parte lateral de la pantalla se agrupan junto con los cables (normalmente en columnas), y cada grupo tiene su propia fuente de voltaje. Por otro lado, los electrodos también se agrupan (normalmente en filas), en donde cada grupo obtiene una tensión de sumidero. Los grupos se han diseñado de manera que cada píxel tiene una combinación única y dedicada de fuentes y sumideros. Los circuitos electrónicos o el software que los controla, activa los sumideros en secuencia y controla las fuentes de los píxeles de cada sumidero.[1]

Figura 3. Televisor con pantalla LCD.

MATERIALES Y EQUIPO EMPLEADO

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

DESARROLLO EXPERIMENTAL

1. 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.

Figura 4. Asignación de pines de la pantalla LCD 16x2.

LCD PIN #	DESCRIPCIÓN	CONEXIÓN
1	VSS	GND
2	VDD	+5V
3	VO	Terminal de ajuste de contraste
4	Register Select (RS)	Conecte al bit del Puerto como indica CodeVision
5	Read/Write (R/W)	Conecte al bit del Puerto como indica CodeVision
6	Clock Enable (E)	Conecte al bit del Puerto como indica CodeVision
7	Data Bit 0	No Conectar
8	Data Bit 1	No Conectar
9	Data Bit 2	No Conectar
10	Data Bit 3	No Conectar
11	Data Bit 4	Conecte al bit del Puerto como indica CodeVision
12	Data Bit 5	Conecte al bit del Puerto como indica CodeVision
13	Data Bit 6	Conecte al bit del Puerto como indica CodeVision
14	Data Bit 7	Conecte al bit del Puerto como indica CodeVision
15	Backlight Anode (+)	+5V
16	Backlight Cathode (-)	GND

Tabla 1. Asignación y conexión de pines de la pantalla LCD 16x2.

En el asistente de CodeVision habilite la pestaña de la LCD dándo clic como se muestra en la siguiente figura, si desea colocar la LCD en otro puerto diferente, sólo indíquele donde quiere conectarlo y el compilador le dirá donde conectar los pines y él se encargará de inicializar la LCD.

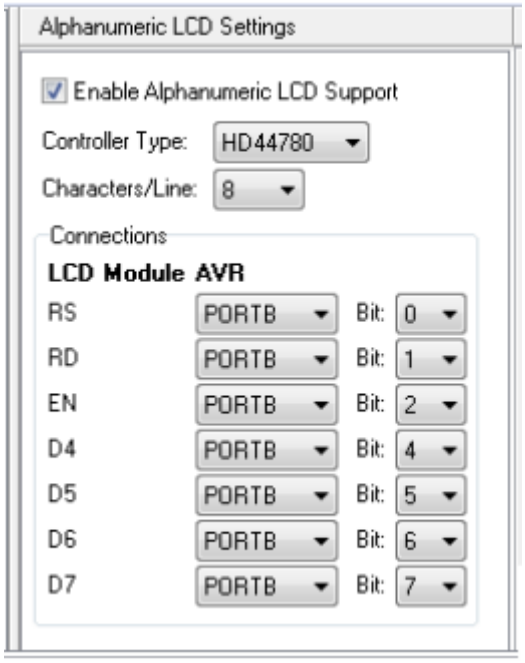


Figura 5. Asignación de pines del Puerto a la pantalla LCD 16x2.

Tome en cuenta que las siguientes funciones sólo se pueden usar si se inicializó la LCD en el CodeVision:

FUNCIÓN	DESCRIPCIÓN
lcd_gotoxy(x,y);	Esta función permite indicarle donde colocar el cursor, que es donde empezará a escribir el mensaje. x es la columna, y es la fila.
lcd_putsf("mensaje");	Con la función anterior le indicamos el mensaje a escribir en la LCD.
lcd_clear();	Esta función borra el mensaje en el display.
lcd_putchar('A');	Esta función escribe en pantalla una sóla letra o un carácter.
lcd_putchar(0x40)	Esta función escribe en pantalla una sóla letra o un carácter en código ASCII.
_lcd_ready(); _lcd_write_data(0xFF);	Los bits para controlar el cursor y la pantalla son: Dato=0xFF=00001DCB D=1 Enciende pantalla, D=0 coloca en stand by la pantalla (bajo consumo) C=1 Cursor on, C=0 Cursor Off B=1 Cursor parpadea, B=0 Cursor fijo
_lcd_ready(); _lcd_write_data(0xFF);	Los bits para controlar el desplazamiento del cursor y la pantalla son: Dato=0001 S/C R/L 00 Y los bits S/C y R/L tienen la siguiente descripción
	S/C R/L Descripción 0 0 Desplaza el cursor a la izquierda 0 1 Desplaza el cursor a la derecha 1 0 Desplaza la pantalla y cursor a la izquierda 1 1 Desplaza la pantalla y cursor a la derecha

Tabla 2. Funciones para la LCD reconocidas en CodeVision.

POSICIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
FILA 01	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)	(10,0)	(11,0)	(12,0)	(13,0)	(14,0)	(15,0)	(16,1)	(17,1)	(18,1)	(19,1)
FILA 02	(4,1)	(5,1)	(6,1)	(7,1)	(8,1)	(9,1)	(10,1)	(11,1)	(12,1)	(13,1)	(14,1)	(15,1)	(16,2)	(17,2)	(18,2)	(19,2)

Tabla 3. Posiciones de la LCD al usar la función lcd_gotoxy(x,y); .

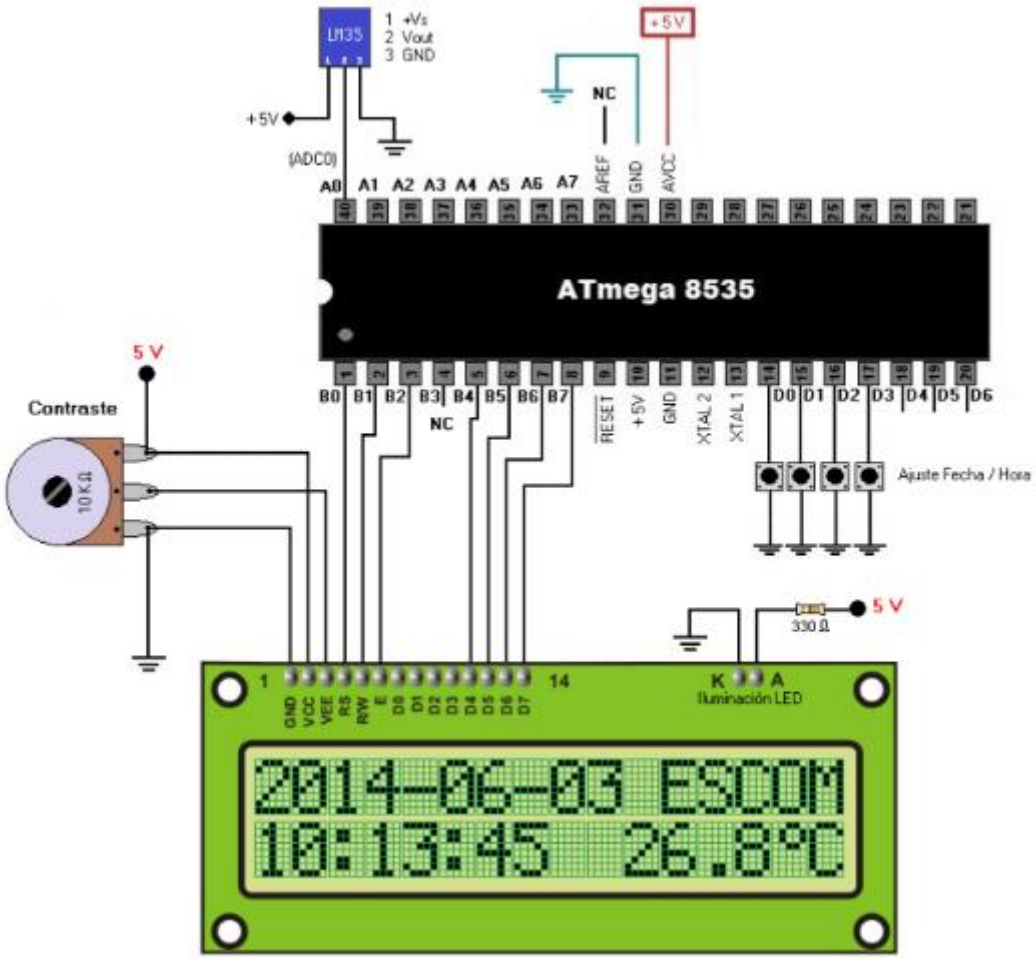


Figura 6. Circuito conectando la LCD en el Puerto B.

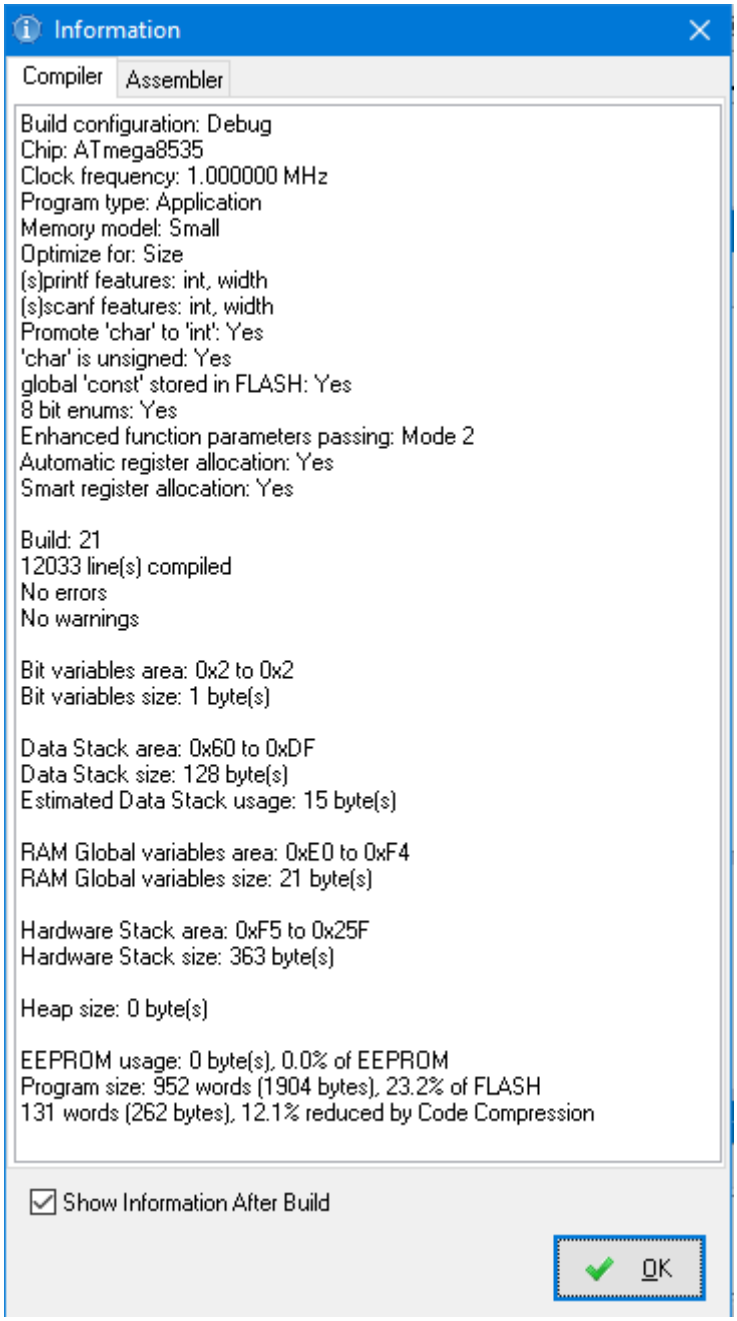


Figura 7. Compilación exitosa en CodeVision.

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 botonC PIND.0
#define botonPr PIND.1
#define botonSe PIND.2
#define botonTe PIND.3

// Alphanumeric LCD functions
#include <alcd.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
unsigned char Vin;
unsigned char decimal, unidades, decenas;
int centi,temp;
unsigned char millarA=2, centenaA=0, decenaA=2, unidadA=3;
unsigned char decenaM=0, unidadM=4, decenaD=2, unidadD=6;
unsigned char decenaH=0, unidadH=7, decenaMin=0, unidadMin=0;
```

```
unsigned char decenaSec=0, unidadSec=0;
unsigned char segundos=0;
const char tabla [10]={48,49,50,51,52,53,54,55,56,57};
unsigned char fecHor=0;
bit botona,botonp;

void checaBoton(){
    if(botonC==0)
        botona = 0;
    else
        botona = 1;
    if((botonp==1)&&(botona==0)){ //hubo cambio de flanco de 1 a 0
        if(fecHor==0){
            fecHor=1;
            lcd_gotoxy(10,0); lcd_putchar('.');
            lcd_gotoxy(9,1); lcd_putchar(0x20);
        }
        else{
            fecHor=0;
            lcd_gotoxy(9,1); lcd_putchar('.');
            lcd_gotoxy(10,0); lcd_putchar(0x20);
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    }
    if((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    botonp=botona;
}

void checaPrim(){
    if(botonPr==0)
        botona=0;
    else
        botona=1;
    if((botonp==1)&&(botona==0)){ //hubo cambio de flanco de 1 a 0
        if(fecHor==1){ //fecha
            unidadA+=1;
            if(unidadA==10){
                unidadA=0;
                decenaA+=1;
                if(decenaA==10){
                    decenaA=1;
                    unidadA=9;
                }
            }
        }
        else{ //hora
```

```
            unidadH+=1;
            if(unidadH==10){
                unidadH=0;
                decenaH+=1;
            }
            if(decenaH==2 && unidadH==4){
                decenaH=0;
                unidadH=0;
                //aumentar dia
                unidadD+=1;
            }
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    }
    if((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    botonp=botona;
}

void checaSeg(){
    if(botonSe==0)
        botona=0;
    else
        botona=1;
    if((botonp==1) && (botona==0)){ //hubo cambio de flanco de 1 a
0
        if(fecHor==1){ //fecha
            unidadM+=1;
            if(unidadM==10){
                unidadM=0;
                decenaM+=1;
            }
            if(decenaM==1 && unidadM==2){
                unidadM=0;
                decenaM=0;
                //aumenta año
                unidadA+=1;
            }
        }
        else{ //hora
            unidadMin+=1;
            if(unidadMin==10){
                unidadMin=0;
                decenaMin+=1;
                if(decenaMin==6){
                    decenaMin=0;
                    unidadMin=0;
```

```
                    unidadH+=1;
                }
            }
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    }
    if((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    botonp=botona;
}

void checaTer(){
    if(botonTe==0)
        botona=0;
    else
        botona=1;
    if((botonp==1)&&(botona==0)){ //hubo cambio de flanco de 1 a 0
        if(fecHor==1){ //fecha
            unidadD+=1;
            if(unidadD==9){
                unidadD=0;
                decenaD+=1;
            }
            if(decenaD==3 && unidadD==1){
                decenaD=0;
                unidadD=0;
                //aumenta mes
                unidadM+=1;
            }
        }
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    }
    if((botonp==0)&&(botona==1)) //hubo cambio de flanco de 0 a 1
        delay_ms(40); //Se coloca retardo de 40mS para eliminar
rebotes
    botonp=botona;
}

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)
      {
      //temperatura
      Vin=read_adc(0);
      centi=(Vin*50)/255;
      if(centi>99) centi=99;

      temp=centi*10;
      decenas=temp/100;
```

```
        temp%=100;
        decimal=temp%10;
        unidades=temp/10;

        checaBoton();
        checaPrim();
        checaSeg();
        checaTer();

        //segundos
        segundos+=1;
        delay_ms(750);
        if(segundos==60){
          segundos=0;
          unidadMin+=1;
        }
        decenaSec=segundos/10;
        unidadSec=segundos%10;

        if(unidadH==10){
          unidadH=0;
          decenaH+=1;
        }

        if(unidadA==10){
          unidadA=0;
          decenaA+=1;
        }

        lcd_gotoxy(0,0); lcd_putchar(tabla[millarA]);
        lcd_gotoxy(1,0); lcd_putchar(tabla[centenaA]);
        lcd_gotoxy(2,0); lcd_putchar(tabla[decenaA]);
        lcd_gotoxy(3,0); lcd_putchar(tabla[unidadA]);
        lcd_gotoxy(4,0); lcd_putchar('-');
        lcd_gotoxy(5,0); lcd_putchar(tabla[decenaM]);
        lcd_gotoxy(6,0); lcd_putchar(tabla[unidadM]);
        lcd_gotoxy(7,0); lcd_putchar('-');
        lcd_gotoxy(8,0); lcd_putchar(tabla[decenaD]);
        lcd_gotoxy(9,0); lcd_putchar(tabla[unidadD]);
        lcd_gotoxy(11,0);
        lcd_putsf("ESCOM");
        lcd_gotoxy(0,1); lcd_putchar(tabla[decenaH]);
        lcd_gotoxy(1,1); lcd_putchar(tabla[unidadH]);
        lcd_gotoxy(2,1); lcd_putchar(':');
        lcd_gotoxy(3,1); lcd_putchar(tabla[decenaMin]);
        lcd_gotoxy(4,1); lcd_putchar(tabla[unidadMin]);
        lcd_gotoxy(5,1); lcd_putchar(':');
        lcd_gotoxy(6,1); lcd_putchar(tabla[decenaSec]);
        lcd_gotoxy(7,1); lcd_putchar(tabla[unidadSec]);
```

```
        lcd_gotoxy(10,1); lcd_putchar(tabla[decenas]);
        lcd_gotoxy(11,1); lcd_putchar(tabla[unidades]);
        lcd_gotoxy(12,1); lcd_putchar('.');
        lcd_gotoxy(13,1); lcd_putchar(tabla[decimal]);
        lcd_gotoxy(14,1); lcd_putchar(0xDF);
        lcd_gotoxy(15,1); lcd_putchar('C');
        }
}
```

CIRCUITO ELECTRICO EN PROTEUS

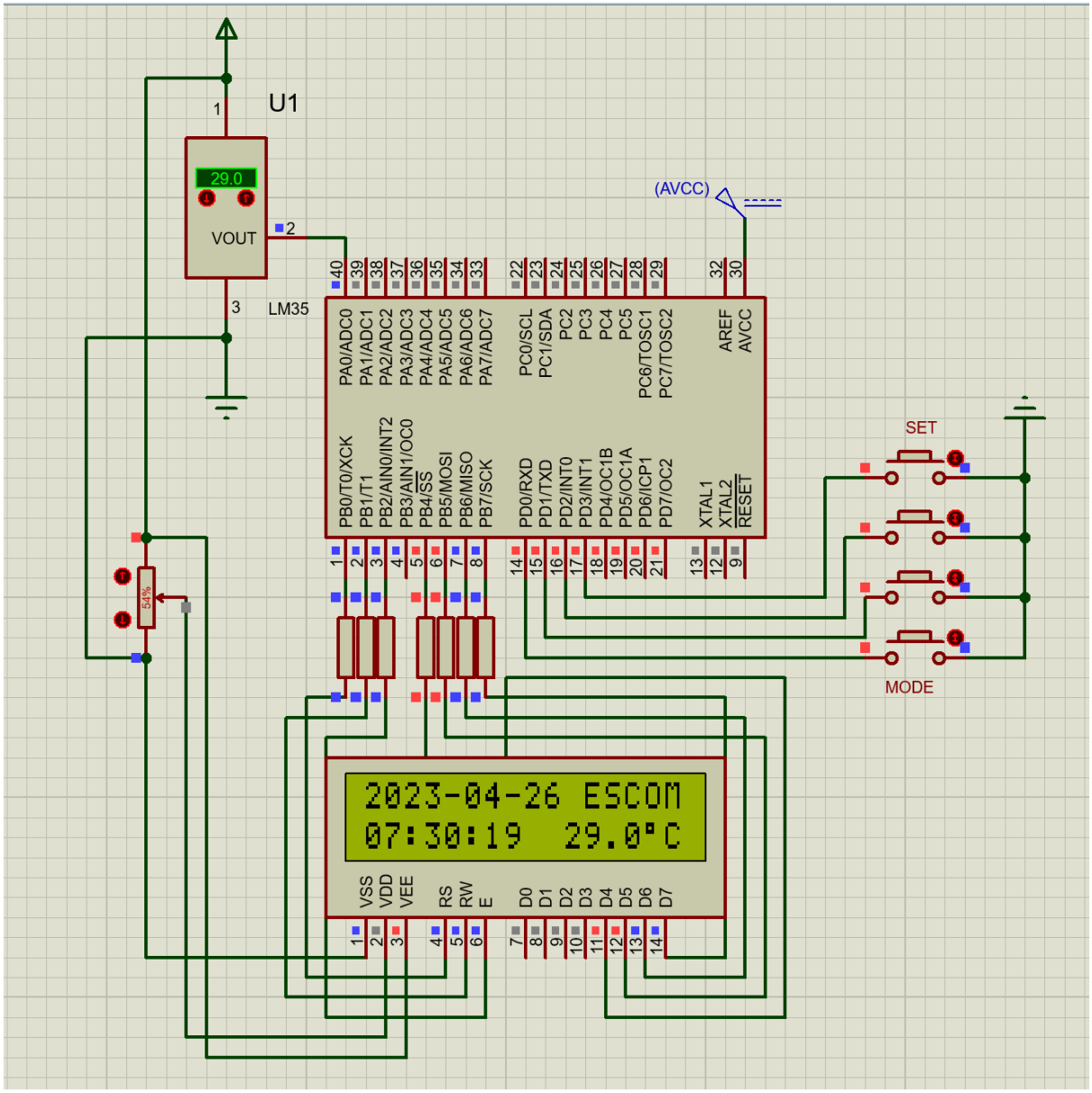
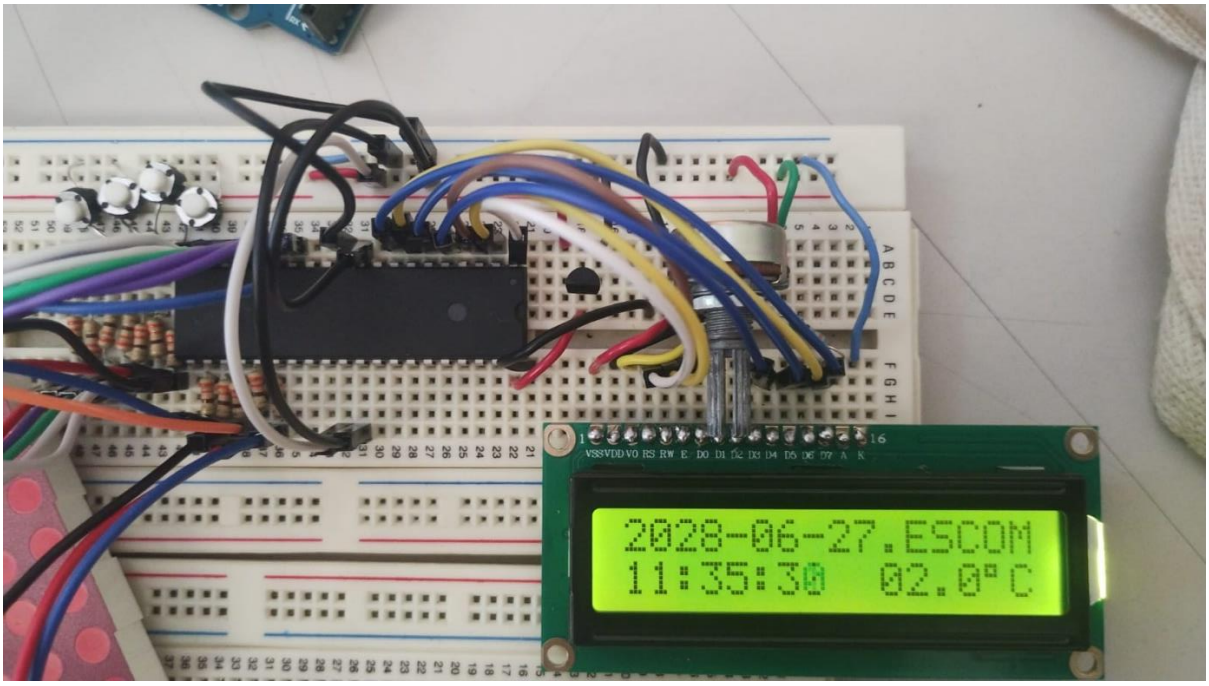


Figura 8. Circuito simulado en Proteus.

CIRCUITO EN EL PROTO ARMADO



OBSERVACIONES Y CONCLUSIONES INDIVIDUALES

Malagón Baeza Alan Adrian

Las pantallas LCD son dispositivos que nos sirven para poder visualizar información producida en nuestros circuitos electrónicos. Anteriormente hemos manejado los displays de 7 segmentos y la matriz de leds, que si bien son dispositivos que nos sirven para el mismo fin, la pantalla LCD tiene la ventaja de tener una mayor definición. Podemos programar la pantalla LCD de tal manera que se muestre la información por secciones, de tal manera que podemos mostrar más de un dato a la vez.

La programación de la pantalla LCD en CodeVision es muy sencilla. Primeramente, debemos de configurar el microcontrolador para poder utilizar la LCD en uno de los puertos de salida. Con ayuda del asistente de CodeVision podemos determinar en qué puerto queremos programar la LCD, así como la asignación de los pines que se conectaran a la LCD.

Martínez Chávez Jorge Alexis

El código para mostrar la información en la LCD es muy sencillo; basta con utilizar la función "lcd_gotoxy()" para posicionarnos en un segmento de la LCD, para posteriormente utilizar la función "lcd_putchar()" para colocar el carácter que deseamos visualizar en el segmento seleccionado. La ventaja de las pantallas LCD es que ya tienen programados los caracteres a mostrar en la pantalla, sólo debemos de indicarle que carácter queremos que se muestre y la pantalla lo mostrará sin necesidad de indicarle cómo debe de encender.

La complejidad de esta práctica estuvo en el manejo de los datos como lo son la fecha y la hora, puesto que, al tener unos botones que los ajustan, se deben de colocar algunas condiciones para poder manipular el comportamiento del circuito.

BIBLIOGRAFÍA

[1] Colaboradores de los proyectos Wikimedia, "Pantalla delgada y plana formada por un número de píxeles de color o monocromo," *Wikipedia.org*, Sep. 26, 2004. https://es.wikipedia.org/wiki/Pantalla_de_cristal_l%C3%ADquido (accessed May 28, 2022).