

# Practicario de

# microcontroladores

Nombre: \_\_\_\_\_ Grupo: \_\_\_\_\_

Dr. Enrique García Trinidad  
Universidad Tecnológica Fidel Velázquez  
<https://sites.google.com/site/mysillyrobots>  
[phd.enrique.garcia@ieee.org](mailto:phd.enrique.garcia@ieee.org)

# Práctica 7

## Pantalla de cristal líquido

### 7.1. Material

El material enlistado es necesario para la realización de la práctica 7.

Ct	Dispositivo	Descripción	Eti.
1	ATmega328P-PU	Microcontrolador AVR RISC 8-bit 20Mhz	U1
1	Regulador L7805CV	Regulador de voltaje 5V 1 A	IC1
1	Capacitor cerámico de 0.1 $\mu$ F 50V	Código: 104	C1
1	Capacitor electrolítico de 470 $\mu$ F 25V	Tolerancia $\pm$ 20 %	C2
1	Capacitor electrolítico de 220 $\mu$ F 25V	Tolerancia $\pm$ 20 %	C3
2	Resistencia de 330 $\Omega$ 1/4W	Código: Naranja, naranja, café, oro	R1, R3
1	Resistencia de 10k $\Omega$ 1/4W	Código: Café, negro, naranja, oro	R4
1	Led 5mm difuso	Color rojo	LED1
1	Push button (Microswitch)	Tipo push, 4 o 2 terminales	S1
1	Pantalla LCD	16 caracteres, 2 líneas	U2
1	Sensor de temperatura	LM35	LM35
1	Potenciómetro de 10k	Tipo preset horizontal de 10 mm	R2
2	Metro de alambre para protoboard		
1	Protoboard		
1	Grabador Usbasp	Grabador microcontroladores AVR 8-bit	J3
1	Fuente de alimentación de 12V 2A	Eliminador de voltaje	J2
1	Computadora con puerto USB		

## 7.2. Conexión de los componentes

- Conecte el diagrama de la práctica 7 de acuerdo a la Figura 7.1:

## 7.3. Ejercicio 1

- Inicie el software Codevision AVR. Cree un nuevo proyecto dando click en el menú [New>Project](#).
- Cuando el software pregunte si queremos usar el asistente [CodeWizardAVR](#) le indicamos que [Si](#).

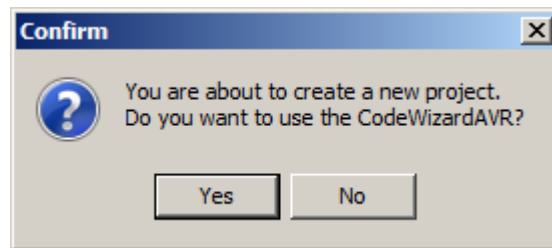


Figura 7.2: Elegir [Yes](#).

- Seleccionar en el tipo de microcontrolador a utilizar [AT90](#), [ATtiny](#), [ATmega](#)

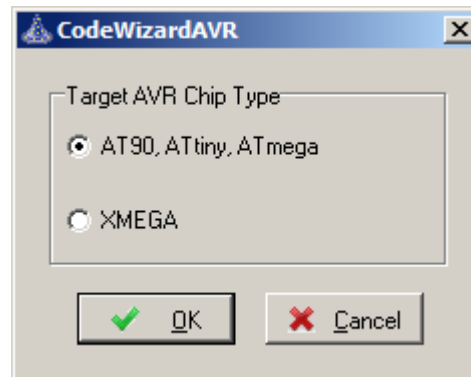


Figura 7.3: Al finalizar seleccionar [OK](#)

- En el asistente, en la ficha [Chip](#) dejar la configuración como sigue:

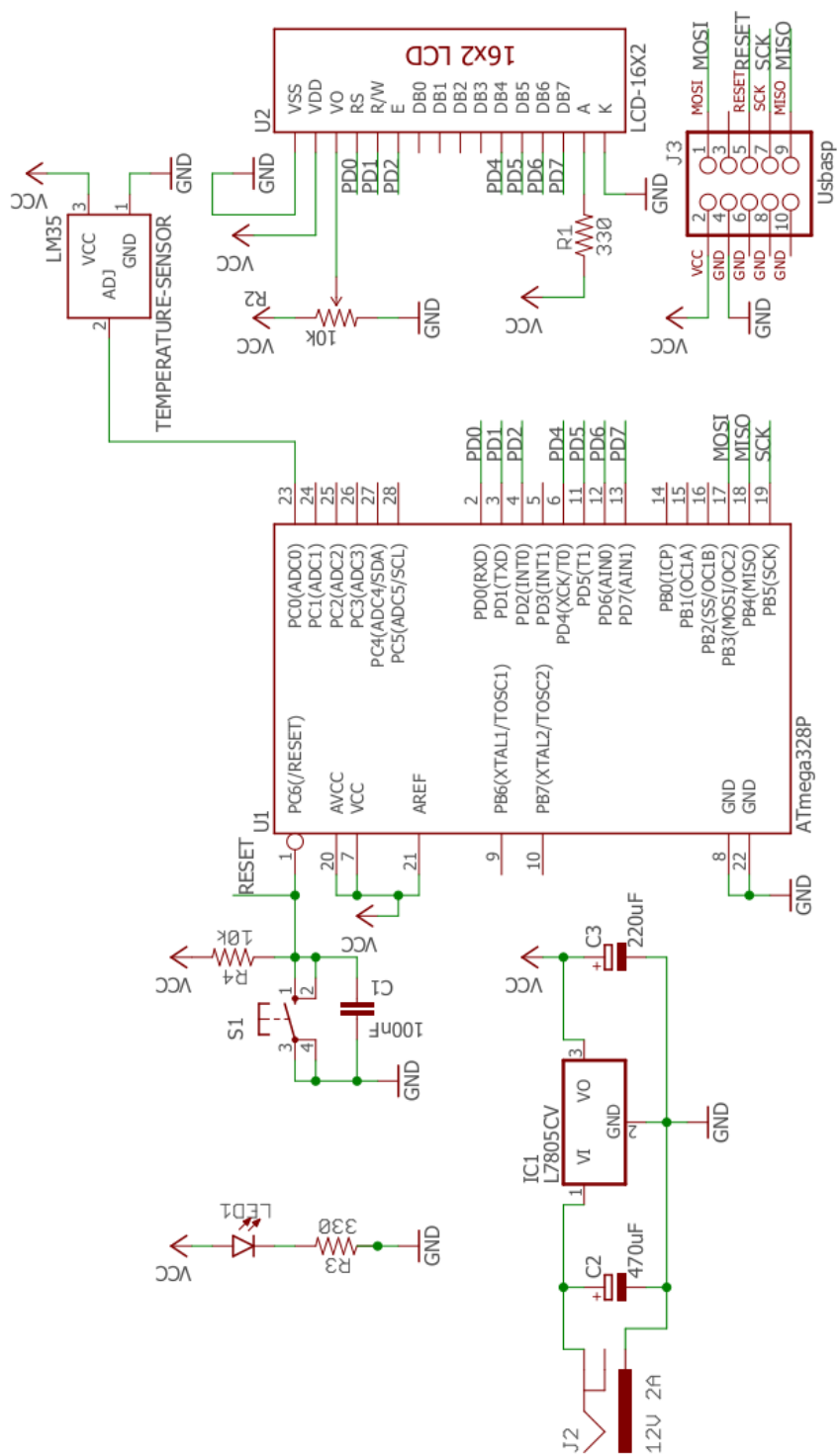


Figura 7.1: Conexión de la práctica 7.

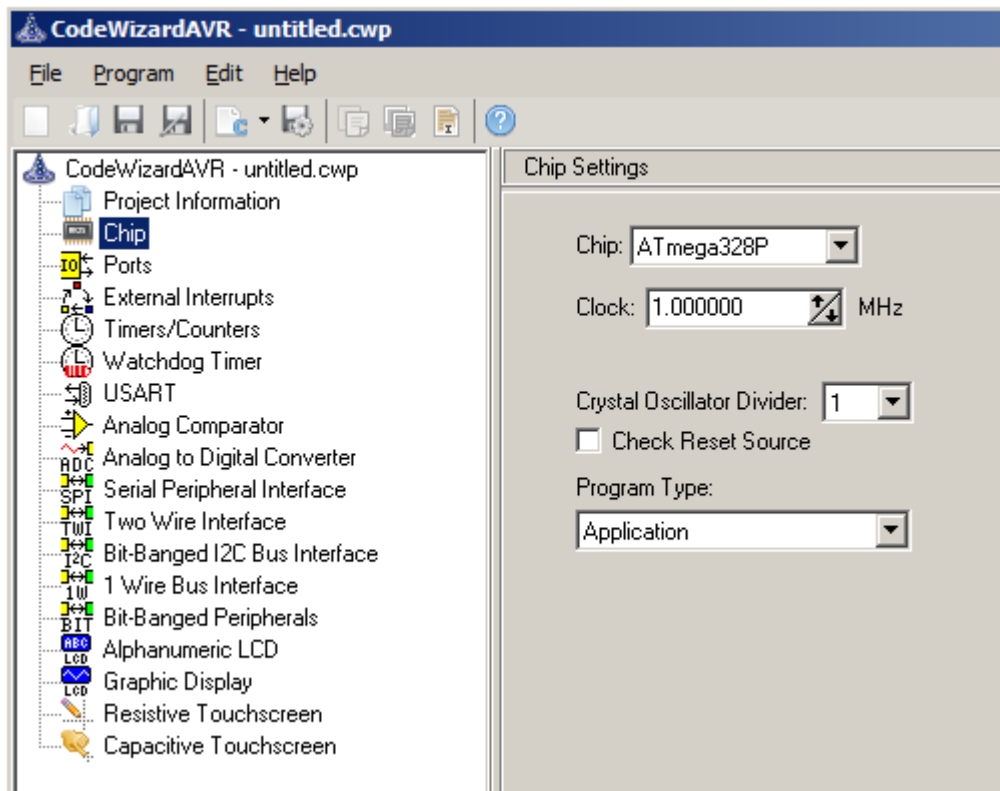


Figura 7.4: Configuración ficha **Chip**

- En el asistente, en la ficha **Ports**, en el Puerto C dejar la configuración como sigue:

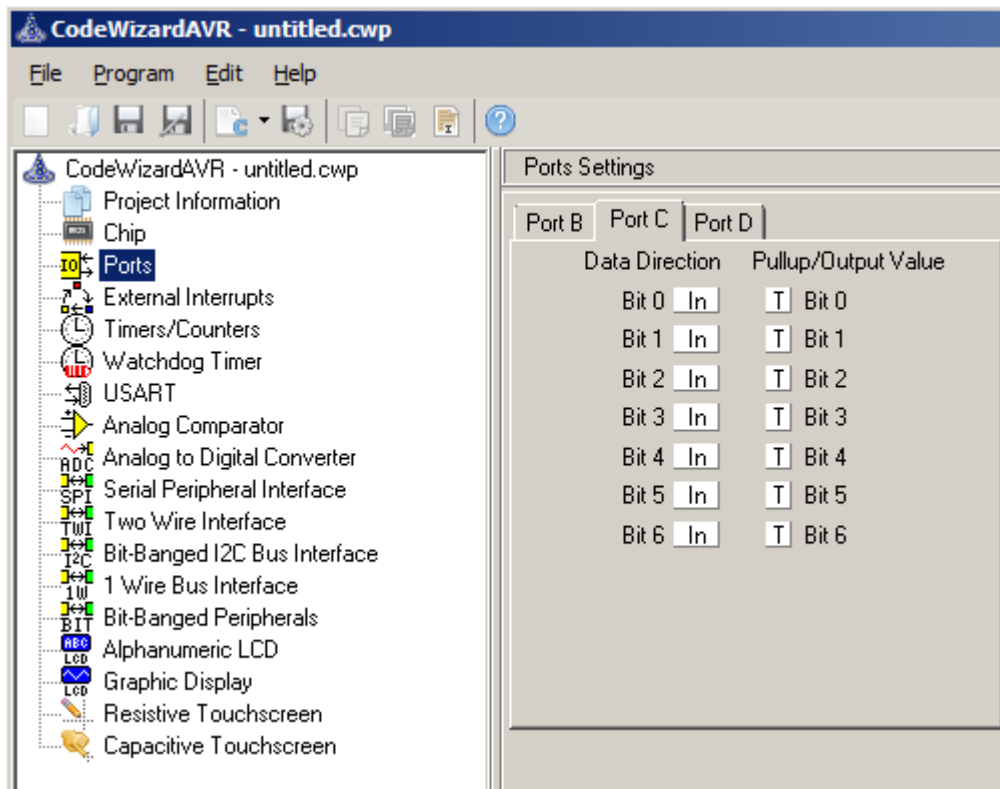


Figura 7.5: Configuración E/S Port C

- En el asistente, en la ficha **Ports**, en el Puerto D dejar la configuración como sigue:

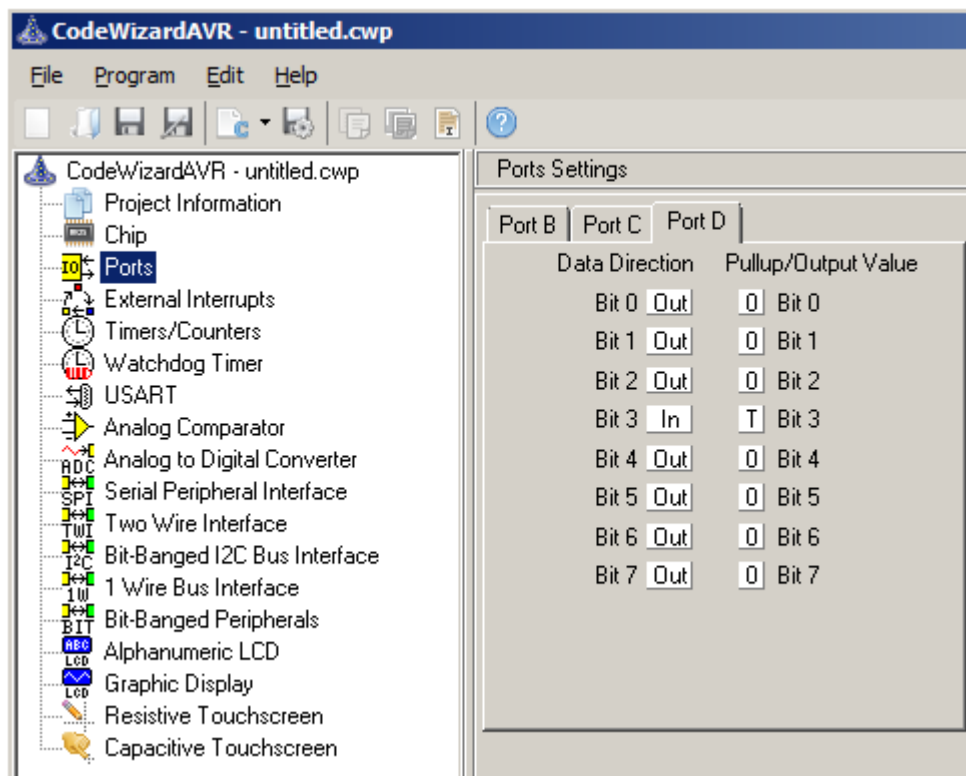


Figura 7.6: Configuración E/S **Port D**

- En el asistente, en la ficha **Analog to Digital Converter**, dejar la configuración como sigue:

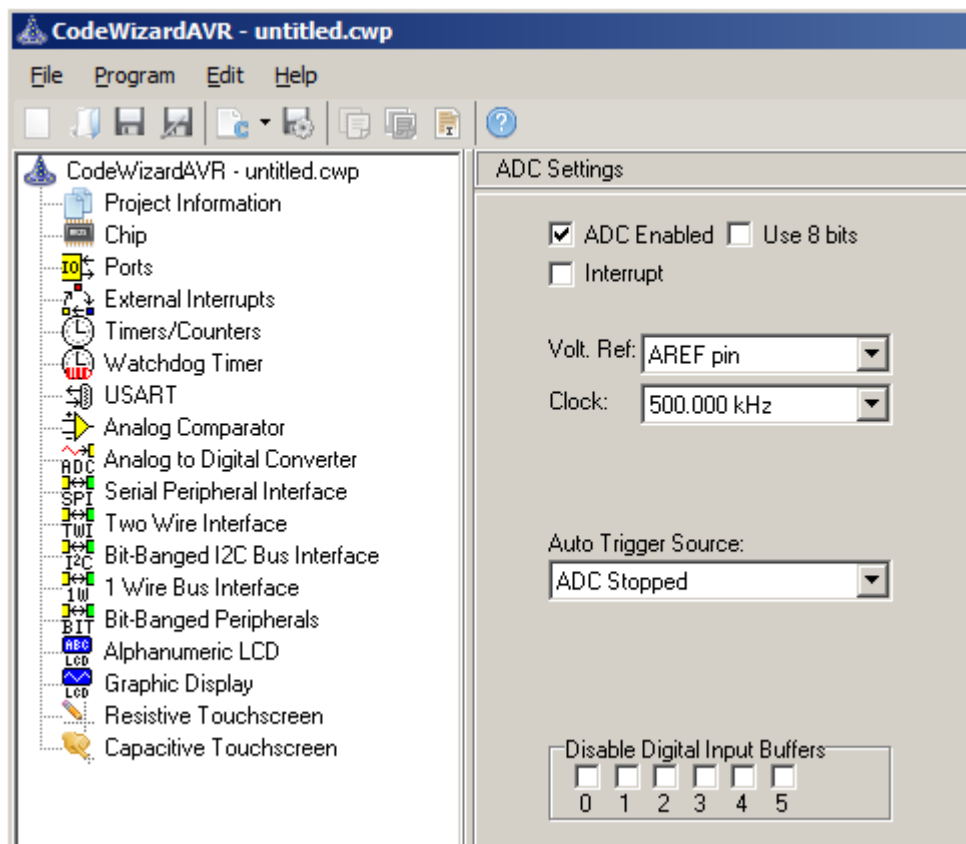


Figura 7.7: Configuración del ADC

- En el asistente, en la ficha **Alphanumeric LCD**, dejar la configuración como sigue:



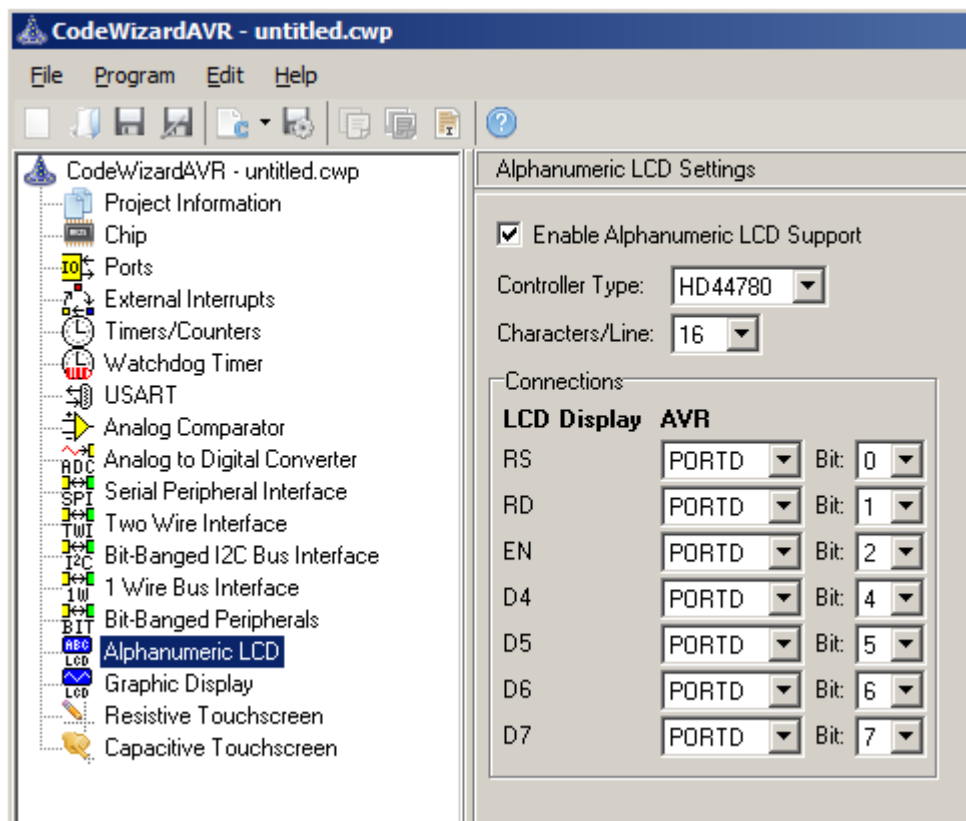


Figura 7.8: Configuración de la pantalla LCD

- Posteriormente debemos de generar el proyecto y el código, eligiendo **Program>Generate, Save and Exit**. Guarde el asistente, código y proyecto con el mismo nombre **prac07**.

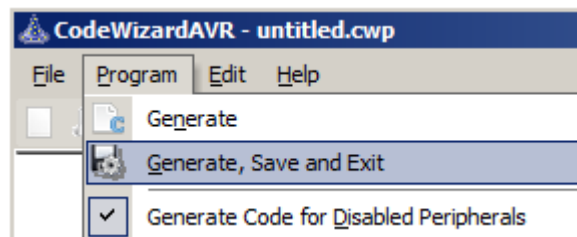


Figura 7.9: Generar el proyecto y el código.

- Mucho del código generado por el asistente no se utiliza en esta práctica. Mo-

difique el código borrando algunas líneas para que sólo quede lo necesario para usar el ADC y la pantalla LCD. El código resultante se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>
#include <alcd.h>

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int 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 ADCW;
}

void main(void)
{
    // 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);
    // Digital input buffer on AIN0: On
    // Digital input buffer on AIN1: On
    DIDR1=(0<<AIN0D) | (0<<AIN1D);

    // ADC initialization
    // ADC Clock frequency: 500.000 kHz
    // ADC Voltage Reference: AREF pin
    // ADC Auto Trigger Source: ADC Stopped
    // Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3
    // : On
    // ADC4: On, ADC5: On
    DIDR0=(0<<ADC5D) | (0<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<
```

```

        ADC1D) | (0<<ADCOD);
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<
        ADIE) | (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

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

while (1)
{

}
}

```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Modifique de nuevo el código como se muestra a continuación:

```

#include <mega328p.h>
#include <delay.h>
#include <alcd.h>

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int 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 ADCW;
}

void main(void)
{
    /* Configuración E/S
    Puerto D:
    PD7=Salida, Estado Inicial= GND
    PD6=Salida, Estado Inicial= GND
    PD5=Salida, Estado Inicial= GND
    PD4=Salida, Estado Inicial= GND
    PD3=Entrada, Pull-down
    PD2=Salida, Estado Inicial= GND
    PD1=Salida, Estado Inicial= GND
    PD0=Salida, Estado Inicial= GND
    Puerto C:
    PC6 a PC0= Entrada, Tri-estado
    */
    DDRC = 0x00;
    PORTC= 0x00;

    DDRD = 0xF7;
    PORTD= 0x00;

    // 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<<AC0) | (0<<ACI) | (0<<ACIE) |
        (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);
    // Digital input buffer on AIN0: On
    // Digital input buffer on AIN1: On
    DIDR1=(0<<AINOD) | (0<<AIN1D);

    // ADC initialization
    // ADC Clock frequency: 500.000 kHz
    // ADC Voltage Reference: AREF pin
    // ADC Auto Trigger Source: ADC Stopped
    // Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3:
    // On
    // ADC4: On, ADC5: On

```

```

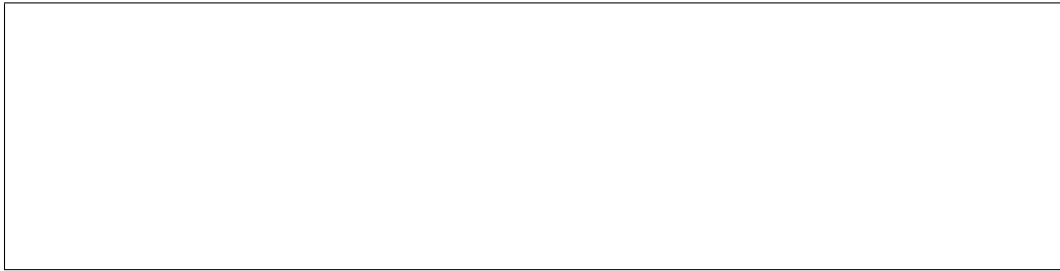
DIDR0=(0<<ADC5D) | (0<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<
    ADC1D) | (0<<ADC0D);
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADAF) | (0<<ADIF) | (0<<
    ADIE) | (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

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

while (1)
{
    lcd_gotoxy(0,0);
    lcd_putsf("UTFV_Mecatronica");
    lcd_gotoxy(6,1);
    lcd_putsf("Hola");
}
}

```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Descargue el archivo que resulto de la compilación **prac07.hex** en la memoria Flash del microcontrolador.
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En la pantalla LCD se debe de mostrar las palabras **UTFV Mecatronica** y **Hola**. De la misma manera muestre la simulación.
- Pruebe cambiar el texto en **lcd\_putsf()** y las coordenadas en **lcd\_gotoxy()** y responda ¿Para qué sirven las dos instrucciones anteriormente citadas?



## 7.4. Ejercicio 2.

- Modifique de nuevo el código como se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>

typedef unsigned char byte;
flash byte char0[8]={
0b00000,
0b00000,
0b01010,
0b10101,
0b10001,
0b01010,
0b00100,
0b00000};

void define_char(byte flash *pd,byte char_code)
{
    byte i,a;
    a=(char_code<<3) | 0x40;
    for (i=0; i<8; i++) lcd_write_byte(a++,*pd++);
}

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int 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 ADCW;
}

void main(void)
{
    unsigned char texto[16]={'H','o','l','a',' ','t','o','d','o',' ','s',' ','U','T','F','V',' '};
    unsigned char temp;
    unsigned char i;
    lcd_init(16);
    define_char(char0,0);

    /* Configuración E/S
    Puerto D:
    PD7=Salida, Estado Inicial= GND
    PD6=Salida, Estado Inicial= GND
    PD5=Salida, Estado Inicial= GND
    PD4=Salida, Estado Inicial= GND
    PD3=Entrada, Pull-down
    PD2=Salida, Estado Inicial= GND
    PD1=Salida, Estado Inicial= GND
    PD0=Salida, Estado Inicial= GND
    Puerto C:
    PC6 a PC0= Entrada, Tri-estado
    */
    DDRC = 0x00;
    PORTC= 0x00;

    DDRD = 0xF7;
    PORTD= 0x00;

    // 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<<ACISO);
    // Digital input buffer on AIN0: On
    // Digital input buffer on AIN1: On

```

```

DIDR1=(0<<AIN0D) | (0<<AIN1D);

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: ADC Stopped
// Digital input buffers on ADC0: 0n, ADC1: 0n, ADC2: 0n, ADC3
: 0n
// ADC4: 0n, ADC5: 0n
DIDR0=(0<<ADC5D) | (0<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<
ADC1D) | (0<<ADC0D);
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<
ADIE) | (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

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

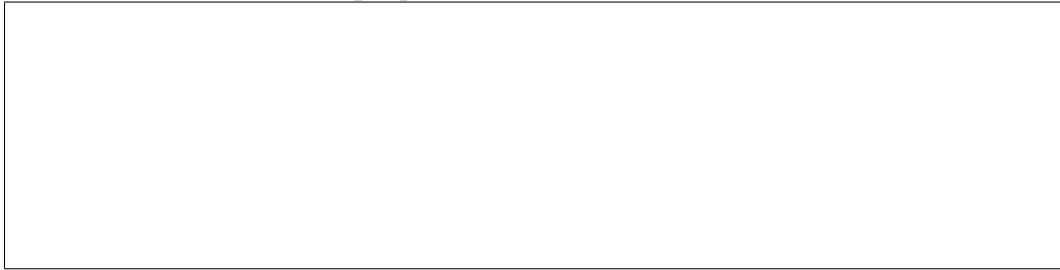
while (1)
{
    temp = texto[0];
    for (i = 0; i < 15; i++)
    {
        texto[i] = texto[i+1];
    }
    texto[15] = temp;
    lcd_gotoxy(0,0);
    lcd_puts(texto);
    for (i=0;i<16;i++)
    {
        lcd_gotoxy(i,1);
        lcd_putchar(0);
    }
    delay_ms(500);
}

```



```
}  
}
```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Descargue el archivo que resulto de la compilación **prac07.hex** en la memoria Flash del microcontrolador.
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En la pantalla LCD se debe de mostrar las palabras **Hola todos UTFV** y una caracter creado por el usuario de forma animada. De la misma manera muestre la simulación.
- Pruebe cambiar los pixeles en **flash byte char0[8]={}** y vea el resultado. Genere su propio caracter pixel por pixel e imprímalo en la pantalla LCD. Escriba la creación de su propio caracter.



- Responda ¿Para que sirve la instrucción **unsigned char texto[16]={'H','o','l','a',' ',' ','t','o','d','o','s',' ',' ','U','T','F','V',' ',' '};**



## 7.5. Ejercicio 3.

- Con el multímetro o con el osciloscopio (de preferencia) sintonice a **1.10V** la señal **AREF** modificando el valor de la resistencia en el potenciómetro **R2**. Esto

para calibrar el valor correcto de la señal del sensor LM35.

- Modifique de nuevo el código como se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int 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 ADCW;
}

void main(void)
{
    unsigned char texto[16];
    unsigned int tempC;

    /* Configuración E/S
       Puerto D:
       PD7=Salida, Estado Inicial= GND
       PD6=Salida, Estado Inicial= GND
       PD5=Salida, Estado Inicial= GND
       PD4=Salida, Estado Inicial= GND
       PD3=Entrada, Pull-down
       PD2=Salida, Estado Inicial= GND
       PD1=Salida, Estado Inicial= GND
       PD0=Salida, Estado Inicial= GND
       Puerto C:
       PC6 a PC0= Entrada, Tri-estado
    */
    DDRC = 0x00;
    PORTC= 0x00;
```

```

DDRD = 0xF7;
PORTD= 0x00;

// 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<<ACISO);
// Digital input buffer on AIN0: On
// Digital input buffer on AIN1: On
DIDR1=(0<<AIN0D) | (0<<AIN1D);

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: ADC Stopped
// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3
// : On
// ADC4: On, ADC5: On
DIDR0=(0<<ADC5D) | (0<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<
      ADC1D) | (0<<ADC0D);
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<
      ADIE) | (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

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

while (1)
{
    //Lee el voltaje en PC0

```

```

tempC = read_adc(0);
//Ajusta la conversión a un rango 0.0 a 1.1V
//Conversión de 10-bit
tempC =(1.1 * tempC * 100.0)/1024.0;
sprintf(texto, "Temp=%d", tempC);
lcd_gotoxy(0,0);
lcd_puts(texto);
delay_ms(500);
}
}

```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Descargue el archivo que resulto de la compilación **prac07.hex** en la memoria Flash del microcontrolador.
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En la pantalla LCD se debe de mostrar la temperatura actual del laboratorio. De la misma manera muestre la simulación.
- ¿Para qué sirven las instrucciones **tempC = read\_adc(0);** y **tempC =(1.1 \* tempC \* 100.0)/1024.0;** respectivamente?



## 7.6. Ejercicio 4.

- Realizar un programa que muestre una barra de progreso al incrementarse la temperatura, como se muestra en la siguiente Figura:

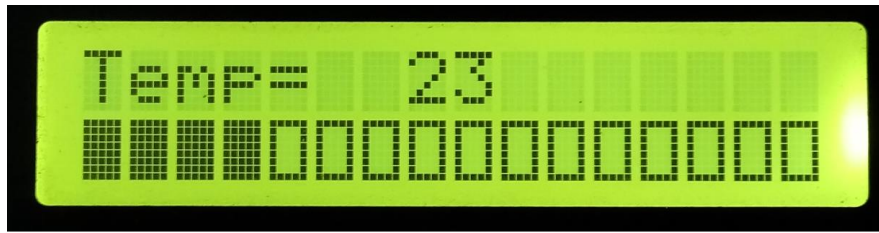


Figura 7.10: Resultado del programa final.

- Modifique de nuevo el código como se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>
#include <alcd.h>
#include <stdio.h>

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))

// Read the AD conversion result
unsigned int 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 ADCW;
}

typedef unsigned char byte;
flash byte char0[8]={
    0b11111,
    0b10001,
    0b10001,
    0b10001,
    0b10001,
    0b10001,
    0b10001,
    0b10001,
    0b11111};

void define_char(byte flash *pd,byte char_code)
```

```

{
    byte i,a;
    a=(char_code<<3) | 0x40;
    for (i=0; i<8; i++) lcd_write_byte(a++,*pd++);
}

void main(void)
{
    unsigned char texto[16];
    unsigned int tempC;
    unsigned char i;
    unsigned char nCuadros;
    lcd_init(16);
    define_char(char0,0);

    /* Configuración E/S
        Puerto D:
        PD7=Salida, Estado Inicial= GND
        PD6=Salida, Estado Inicial= GND
        PD5=Salida, Estado Inicial= GND
        PD4=Salida, Estado Inicial= GND
        PD3=Entrada, Pull-down
        PD2=Salida, Estado Inicial= GND
        PD1=Salida, Estado Inicial= GND
        PD0=Salida, Estado Inicial= GND
        Puerto C:
        PC6 a PC0= Entrada, Tri-estado
    */
    DDRC = 0x00;
    PORTC= 0x00;

    DDRD = 0xF7;
    PORTD= 0x00;

    // 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<<ACISO);
    // Digital input buffer on AIN0: On
    // Digital input buffer on AIN1: On
    DIDR1=(0<<AINOD) | (0<<AIN1D);

```

```

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: ADC Stopped
// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3
: On
// ADC4: On, ADC5: On
DIDR0=(0<<ADC5D) | (0<<ADC4D) | (0<<ADC3D) | (0<<ADC2D) | (0<<
ADC1D) | (0<<ADC0D);
ADMUX=ADC_VREF_TYPE;
ADCSRA=(1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<
ADIE) | (0<<ADPS2) | (0<<ADPS1) | (1<<ADPS0);
ADCSRB=(0<<ADTS2) | (0<<ADTS1) | (0<<ADTS0);

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

while (1)
{
    //Lee el voltaje en PC0
    tempC = read_adc(0);
    //Ajusta la conversión a un rango 0.0 a 1.1V
    //Conversión de 10-bit
    tempC =(1.1 * tempC * 100.0)/1024.0;
    sprintf(texto,"Temp=_%d_", tempC);
    lcd_gotoxy(0,0);
    lcd_puts(texto);
    delay_ms(500);

    //Rutina para pintar la barra de progreso
    nCuadros=tempC/5;
    for(i=0;i<nCuadros;i++)
    {
        lcd_gotoxy(i,1);
        lcd_putchar(0xFF); //Cuadro lleno
    }
}

```

```

    }
    for(i=nCuadros;i<16;i++)
    {
        lcd_gotoxy(i,1);
        lcd_putchar(0);    //Cuadro vacío
    }
}
}

```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Descargue el archivo que resulto de la compilación **prac07.hex** en la memoria Flash del microcontrolador.
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En la pantalla LCD se debe de mostrar una barra de progreso que debe aumentar si la temperatura sube. De la misma manera muestre la simulación.
- ¿Para qué sirve la instrucción **for(i=0;i<nCuadros;i++)** contenida en el programa?