



OBJETIVO:

Diseñar e implementar sistemas microcontrolados.

INDICE

- 1. Introducción.**
- 2. Manejo de Puertos. (Prácticas A)**
- 3. LCD (Prácticas B).**
- 4. Convertidor ADC (Prácticas C).**
- 5. PWM (Prácticas D).**
- 6. Convertidor DAC (Prácticas E).**
- 7. Comunicación Serial (Prácticas F).**
- 8. Comunicación Inalámbrica. (Prácticas G)**
- 9. Aplicaciones (Prácticas H).**
- 10. Guía de Programas Demo.**
- 11. Guía de Actividades.**
- 12. Apéndice.**





CONTENIDO TEMÁTICO

1. Introducción.
2. Manejo de Puertos. (Prácticas A)
3. LCD (Prácticas B).
 - 3.1 Introducción.
 - 3.2 Set de Instrucciones.
 - 3.3 Conexión de LCD a través de comunicación a 4 bits.
 - 3.4 Generación de mensajes por medio de Arreglos.
 - 3.5 Uso de un teclado Matricial.
4. Convertidor ADC (Prácticas C).
5. PWM (Prácticas D).
6. Convertidor DAC (Prácticas E).
7. Comunicación Serial (Prácticas F).
8. Comunicación Inalámbrica. (Prácticas G)
9. Aplicaciones (Prácticas H).
10. Guía de Programas Demo.
11. Guía de Actividades.





CONTENIDO TEMÁTICO

- 1. Introducción.**
- 2. Manejo de Puertos. (Prácticas A)**
- 3. LCD (Prácticas B).**
- 4. Convertidor ADC (Prácticas C).**
 - 4.1 Introducción.**
 - 4.2 Configuración de Entradas Analógicas.**
 - 4.3 Adquisición de señal por medio del ADC.**
 - 4.4 Especificación de un Rango de Medición.**
 - 4.5 El controlador ON OFF.**
- 5. PWM (Prácticas D).**
- 6. Convertidor DAC (Prácticas E).**
- 7. Comunicación Serial (Prácticas F).**
- 8. Comunicación Inalámbrica. (Prácticas G)**
- 9. Aplicaciones (Prácticas H).**
- 10. Guía de Programas Demo.**
- 11. Guía de Actividades.**





GUÍA DE PROGRAMAS DEMO

- 1. Uso de #define.**
- 2. Generación de Métodos Básicos.**
- 4. Entrada y Salida de un solo Bit.**
- 5. Espera a Pulsar haciendo una Tarea.**
- 6. Espera a Pulsar sin realizar ninguna Acción.**
- 7. Esperar a Pulsar, Esperar a Soltar, Realizar Acción.**
- 8. Salida de Datos – Asignación Directa.**
- 9. Salida de Datos – Asignación Indirecta.**
- 10. Convertidor de Decimal a Binario.**
- 11. Generando un Corrimiento de Datos.**
- 12. Generando Corrimientos con Bit de Inicio por Pulsador.**
- 13. Agregando Bits a un Corrimiento.**
- 14. Eliminando Bits a un Corrimiento.**
- 15. Generando un Corrimiento de Datos por medio de Arreglos.**
- 16. Manejo Básico de Display.**
- 17. Manejo de Display por medio de Arreglos.**
- 18. Contador Continuo de 0 a 20.**
- 19. Intercambiando un Display de Ánodo Común por un Cátodo Común.**
- 20. Letrero Fijo en un LCD.**
- 21. Contando en un LCD.**
- 22. Recorriendo el LCD.**
- 23. Generando mensajes por medio de Arreglos.**
- 24. Monitoreando un Puerto.**





GUÍA DE PROGRAMAS DEMO

- 25. Usando un Teclado Matricial.**
- 26. Uso de un Teclado Matricial por medio de una Librería.**
- 27. Menú Básico de Opciones.**
- 28. Lectura básica del ADC.**
- 29. Voltímetro a Leds.**
- 30. Medición de dos o más Canales ADC.**
- 31. Especificando un Rango de Medición.**
- 32. Eliminando el Parpadeo en el LCD.**
- 33. El Termómetro básico.**
- 34. Controlador ON-OFF de Temperatura.**





GUÍA DE ACTIVIDADES

- 1. Actividad 01 –Realiza una Función Nula.**
- 2. Actividad 02 –Realiza una Función con Respuesta.**
- 3. Actividad 03 – Puerto Reflejado.**
- 4. Actividad 04 – Esperando Pulsar para Detener un Corrimiento.**
- 5. Actividad 05 – Esperando Pulsar para Generar Intermitente.**
- 6. Actividad 06 – Espera a Pulsar, Espera a Soltar, Encendido Gradual y Apagado Gradual.**
- 7. Actividad 07 – Puerto Espejo.**
- 8. Actividad 08 - Convertidor de Binario a Decimal.**
- 9. Actividad 09 – Va y viene.**
- 10. Actividad 10 – Aplausos!!!**
- 11. Actividad 11 – Brincando la cerca.**
- 12. Actividad 12 – Rebotando la pelota.**
- 13. Actividad 13 – Indicando el camino.**
- 14. Actividad 14 – Contando Pasajeros.**
- 15. Actividad 15 – Contador continuo de 0 a 99.**
- 16. Actividad 16 – Convertidor de Binario a Decimal Extendido.**
- 17. Actividad 17 – Eliminando el Efecto Rebote.**
- 18. Actividad 18 – Contador continuo de 0 a 999 up/down.**





GUÍA DE ACTIVIDADES

19. [Actividad 19 – Imprimiendo la fecha.](#)
20. [Actividad 20 – Contador Industrial en LCD.](#)
21. [Actividad 21 – Cronometro.](#)
22. [Actividad 22 – Efecto Desvanecer.](#)
23. [Actividad 23 – Dejando huellas.](#)
24. [Actividad 24 – Monitor de Puertos Digitales.](#)
25. [Actividad 25 – Brincando la cerca II.](#)
26. [Actividad 26 – Rebotando la pelota II.](#)
27. [Actividad 27 – Calculadora básica.](#)
28. [Actividad 28 – Segundas funciones.](#)
29. [Actividad 29 – Tecleando en el Celular.](#)
30. [Actividad 30 – Sensor de Luminosidad.](#)
31. [Actividad 31 – Sensor de Corriente Eléctrica.](#)
32. [Actividad 32 – El canal mayor.](#)
33. [Actividad 33 – Voltímetro a Leds.](#)
34. [Actividad 34 – Voltímetro en Displays.](#)
35. [Actividad 35 – El termómetro gráfico.](#)
36. [Actividad 36 – Llenando el tinaco.](#)





DISPLAY LCD

Introducción

Una pantalla LCD es un conjunto de matrices de caracteres distribuidos normalmente en dos líneas con capacidad de hasta 32 caracteres, donde el control de lo que es mostrado en pantalla lo lleva un microcontrolador incorporado en él.

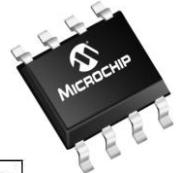


Figura 6. Pines de un LCD.

Las terminales principales para su operación son alimentación de 5v (Vcc) y tierra (GND), además de los pines de control de (RS, EN, R/W) y sus pines de datos (DB0 DB7).

Existen dos formas de conexión de los LCD, uno por medio de comunicación a 8 bits y otro a 4 bits, la principal diferencia entre los modos de comunicación son el uso y/o generación de caracteres, ya que en el modo de 8 bits podemos generar nuevos caracteres a partir de la realización de una matriz, caracteres que se almacenan en la memoria RAM del LCD para posteriormente usarlos, mientras que en el modo de 4 bits solo podemos usar los caracteres estándar del fabricante ([figura 7](#)). Cada uno de los caracteres programados pueden ser llamados referenciándolos con su código ASCII o simplemente usar una librería del set de instrucciones del LCD para escritura de texto.





Lower 4 bit	Upper 4 bit	0000 (\$0x)	0010 (\$2x)	0011 (\$3x)	0100 (\$4x)	0101 (\$5x)	0110 (\$6x)	0111 (\$7x)	1010 (\$Ax)	1011 (\$Bx)	1100 (\$Cx)	1101 (\$Dx)	1110 (\$Ex)	1111 (\$Fx)
		CG RAM (0)	0	ä	P	~	P		-	ä	ë	æ	p	
xxxx0000 (\$x0)	(1)		!	1	A	Q	a	ä	»	ä	ç	ä	q	
xxxx0001 (\$x1)	(2)		"	2	B	R	b	r	「	イ	ツ	メ	ß	ß
xxxx0010 (\$x2)	(3)		#	3	C	S	c	s	」	ウ	テ	モ	»	»
xxxx0011 (\$x3)	(4)		\$	4	D	T	d	t	、	エ	ト	タ	μ	μ
xxxx0100 (\$x4)	(5)		%	5	E	U	e	u	・	オ	ナ	ユ	ö	ö
xxxx0101 (\$x5)	(6)		&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0110 (\$x6)	(7)		*	7	G	W	g	w	ア	キ	ヌ	ラ	g	π
xxxx0111 (\$x7)			<	8	H	X	h	x	イ	ク	ネ	リ	σ	σ
xxxx1000 (\$x8)	CG RAM (0))	9	I	Y	i	y	タ	ケ	ノ	ル	·	·
xxxx1001 (\$x9)	(1)		*	:	J	Z	j	z	エ	コ	ハ	レ	j	≠
xxxx1010 (\$xA)	(2)		+	;	K	C	k	{	オ	サ	ヒ	ロ	*	¤
xxxx1011 (\$xB)	(3)		,	<	L	¥	l		ヤ	シ	フ	ワ	◊	¤
xxxx1100 (\$xC)	(4)		-	=	M	J	m	}	ュ	ズ	ヘ	ン	‰	÷
xxxx1101 (\$xD)	(5)		.	>	N	^	n	→	ヨ	セ	ホ	”	ñ	
xxxx1110 (\$xE)	(6)		/	?	O	_	o	←	ョ	ソ	マ	”	ö	█
xxxx1111 (\$xF)	(7)													

Figura 7. Caracteres pre-programados en un LCD.

Estos caracteres pueden ser escritos en cualquiera de las posiciones disponibles en el LCD, donde para el caso de un display 16x2 se tienen 16 columnas "x" y solo 2 renglones "y"

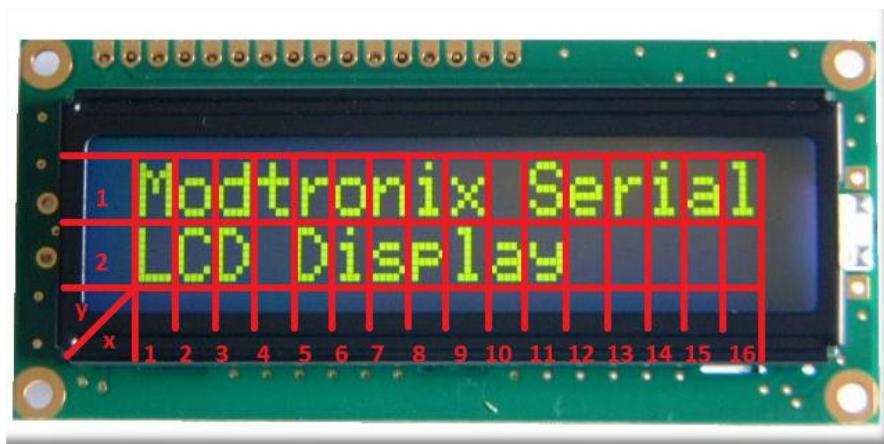


Figura 8. Sistema de posición de un Display LCD.





Set de Instrucciones

Dado que en un microcontrolador no se tienen demasiados pines disponibles, la mayoría de fabricantes proporcionan librerías para la comunicación con el LCD a 4 bits, de manera que los pines necesarios para esta comunicación son R/W, EN, RS y D4, D5, D6, D7. En el caso del microcontrolador PIC la librería necesaria es LCD.c, donde su set de instrucciones es el siguiente.

Instrucción	Función
lcd_init()	Rutina de inicialización del LCD.
lcd_putc(c)	Manda a imprimir un carácter o un conjunto de caracteres en la posición donde se encuentre el LCD. lcd_putc('f'); Limpia pantalla. lcd_putc('\n'); Posiciona al LCD en el renglón 2. lcd_putc('b'); Posiciona al LCD una posición atrás. lcd_putc('A'); Imprime el carácter A. lcd_putc("Prueba LCD"); Imprime el texto "Prueba LCD".
lcd_gotoxy(x,y)	Posiciona al LCD en la columna "x" y el renglón "y".
lcd_getc(x,y)	Obtiene el carácter almacenado en la columna "x" y el renglón "y".
printf(lcd_putc,"c")	Imprime un texto con formato. printf(lcd_putc,"Esta es una prueba"); Imprime el texto "Esta es una prueba". printf(lcd_putc,"%d",num); Imprime un valor numérico decimal que proviene de la variable "num". printf(lcd_putc,"%3d",num); Imprime un valor numérico decimal de 3 dígitos que proviene de la variable "num". printf(lcd_putc,"%f",num); Imprime un valor numérico flotante que proviene de la variable "num". printf(lcd_putc,"%2.3f",num); Imprime un valor numérico flotante de 2 dígitos enteros y 3 dígitos decimales que proviene de la variable "num". printf(lcd_putc,"%X",num); Imprime un valor numérico decimal en formato hexadecimal que proviene de la variable "num".





Conexión de un LCD a través de comunicación a 4 bits

Para poder usar un LCD por medio de un microcontrolador PIC es necesario el incluir la librería de este en el programa en C, esta declaración se realiza como sigue:

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>
```

donde la conexión de pines por default es como se muestra en la [figura 9](#).

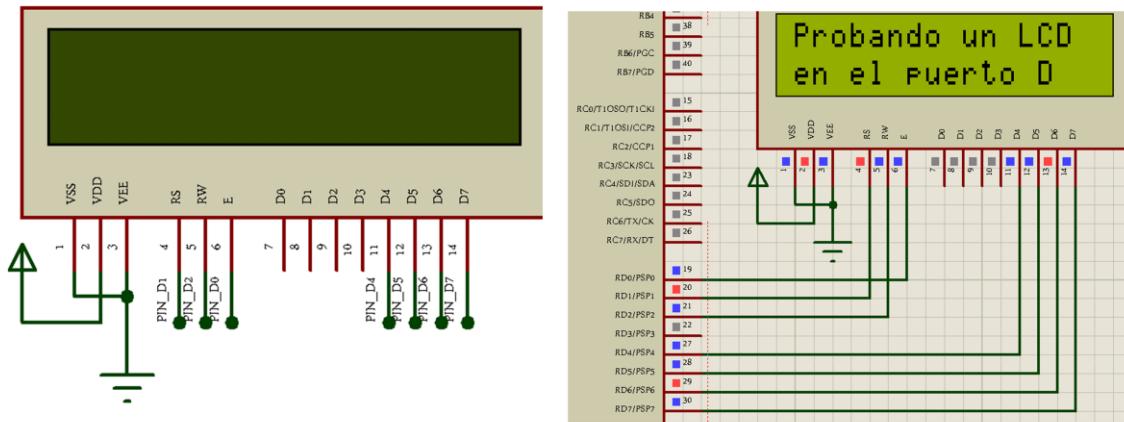
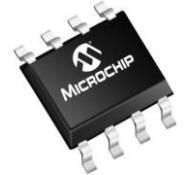


Figura 9. Conexión típica a un LCD.

Como se puede ver la conexión por default se realizaría en el puerto D del microcontrolador, sin embargo esto no significa que sea obligatorio usar ese puerto, ya que la librería del lcd nos permite hacer un cambio para que el puerto B sea el controlador de este y solo con la inclusión de la línea “#define use_portb_lcd TRUE” antes de la declaración de la librería.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#define use_portb_lcd TRUE
#include <lcd.c>
```

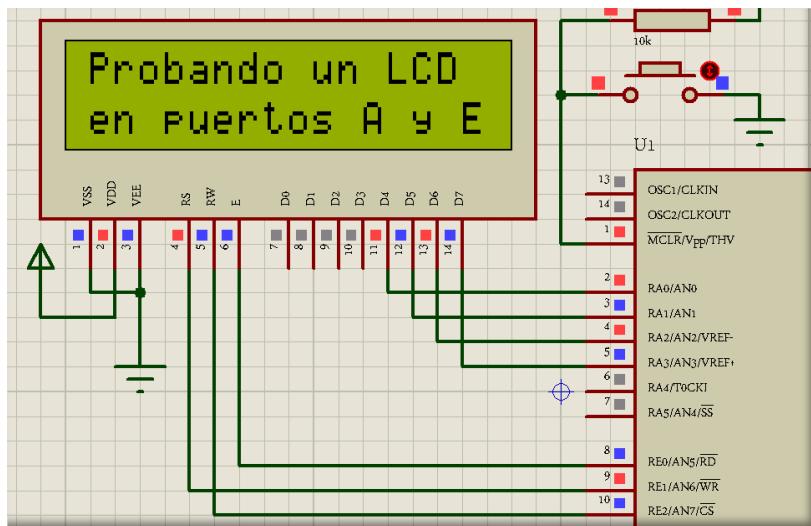




Esto nos permitirá una distribución de pines igual a la de la [figura 9](#) pero esta vez con los pines del puerto B.

En ocasiones no será posible tener un puerto completo para el uso del LCD, por lo que también es posible asignar pines de diferentes puertos para manipularlo, lo único que tendremos que agregar al programa es la especificación de que pines usaremos para cada pin del LCD. Esto es posible si definimos cada pin como sigue:

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#define LCD_ENABLE_PIN    PIN_E0
#define LCD_RS_PIN        PIN_E1
#define LCD_RW_PIN        PIN_E2
#define LCD_DATA4          PIN_A0
#define LCD_DATA5          PIN_A1
#define LCD_DATA6          PIN_A2
#define LCD_DATA7          PIN_A3
#include <lcd.c>
```



usando `#define` podremos asignar pines según nuestras necesidades.





Demo 20 - Letrero Fijo.

El siguiente programa hace uso de un LCD para imprimir un texto tanto en el primer renglón como el segundo.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Definiciones de Variables Globales

// Declaración de Subrutinas

// Programa Principal
void main() {

// Definiciones de Variables Locales

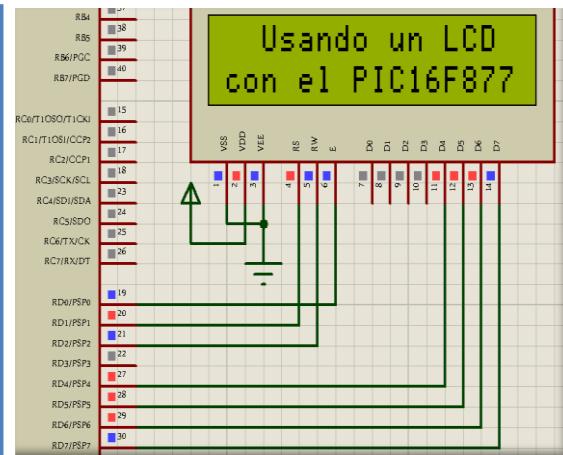
// Configuración de Puertos

// Inicialización del LCD
    lcd_init();           1

// Bucle Principal
    lcd_gotoxy(3,1);     2
    lcd_putc("Usando un LCD"); 3
    lcd_gotoxy(1,2);     4
    lcd_putc("con el PIC16F877"); 5

    while (1){
        // Instrucciones del programa

    } //end while
}
```



Como se puede observar en el programa, lo primero que realizamos es inicializar el LCD (1), posteriormente le indicamos que se ubique en la columna 3 y el renglón 1 (2) para imprimir el primer texto (3), enseguida lo ubicamos en la columna 1 y el renglón 2 (4) para finalmente imprimir un segundo texto (5).





Demo 21 - Contando en un LCD.

El siguiente programa muestra la cuenta de una variable a través del LCD, donde usamos la instrucción printf para imprimir tanto texto como el valor de la variable. Es importante en este tipo de eventos que se tome en cuenta el número de dígitos que queremos mostrar de una variable, para nuestro caso, contador solo requiere unidades y decenas, por lo que será suficiente 2 dígitos (flecha naranja), si no se toma en cuenta la información se traslape (Realice este demo colocando %2d y %d en flecha naranja para notar el efecto).

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Definiciones de Variables Globales

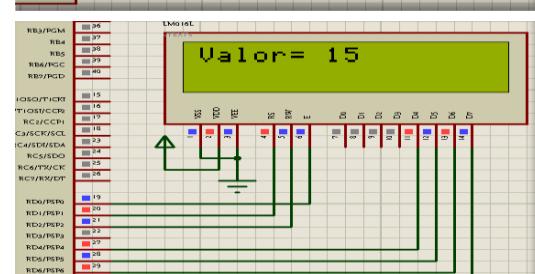
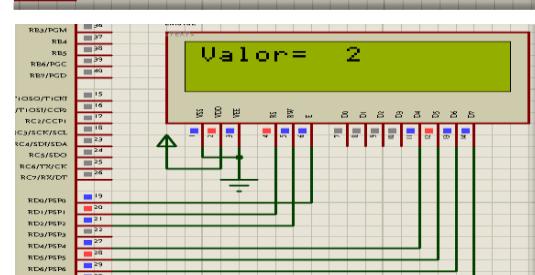
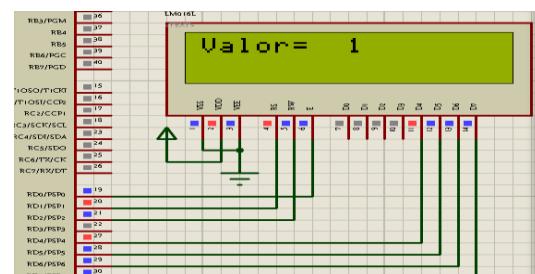
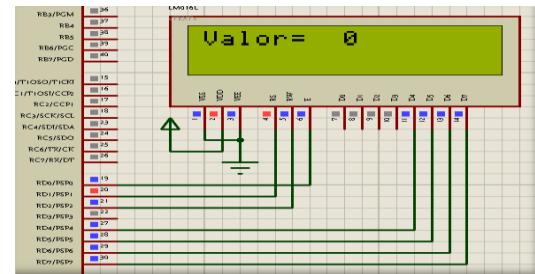
// Declaración de Subrutinas

// Programa Principal
void main() {

// Definiciones de Variables Locales
    int cont;
// Configuración de Puertos

// Inicialización del LCD
    lcd_init();

// Bucle Principal
    while (1){
        // Instrucciones del programa
        for(cont=0;cont<=15;cont++)
        {
            lcd_gotoxy(1,1);
            printf(lcd_putc,"Valor= %2d",cont);
            delay_ms(500);
        }
    } //end while
}
```





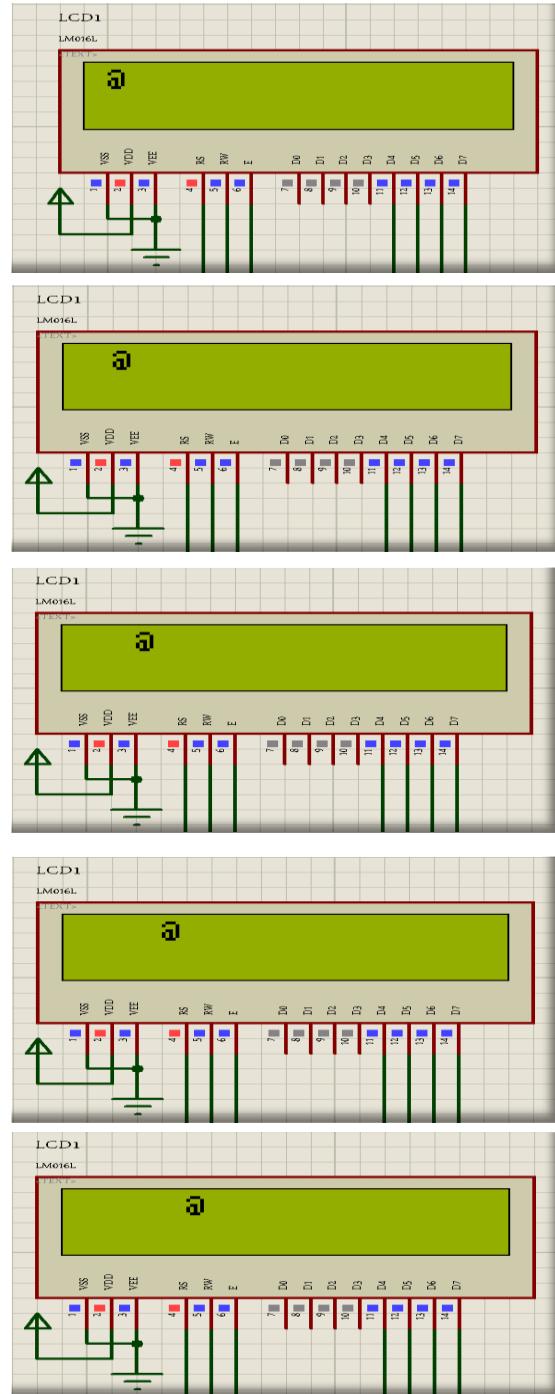
Demo 22 – Recorriendo el LCD

El siguiente programa hace uso de ciclos FOR para imprimir un carácter en cada una de las posiciones posibles del LCD, mientras va borrando la posición anterior, de forma que parece que el carácter va caminando.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {
// Definiciones de Variables Locales
    int x;
// Inicialización del LCD
    lcd_init();

// Bucle Principal
    while (1){
        // Instrucciones del programa
        for(x=1;x<=16;x++) {
            // Borrar el anterior
            lcd_gotoxy(x-1,1);
            lcd_putc(" ");
            // Ubicar el nuevo
            lcd_gotoxy(x,1);
            lcd_putc('@');
            delay_ms(500);
        }
        // Borrar esquina derecha
        lcd_gotoxy(16,1);
        lcd_putc(" ");
        for(x=16;x>=1;x--){
            // Borrar el anterior
            lcd_gotoxy(x+1,2);
            lcd_putc(" ");
            // Ubicar el nuevo
            lcd_gotoxy(x,2);
            lcd_putc('@');
            delay_ms(500);
        }
        // Borrar esquina izquierda
        lcd_gotoxy(1,2);
        lcd_putc(" ");
    }//end while
}
```





Generación de Mensajes por medio de Arreglos

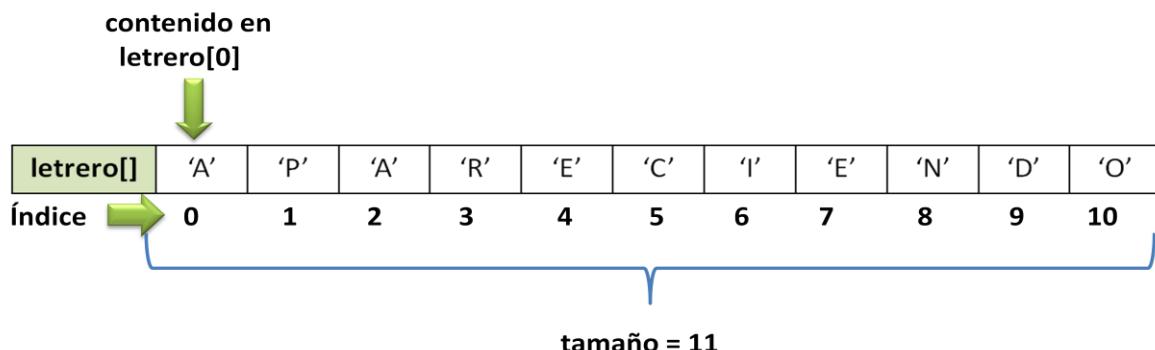
En ocasiones se tiene la necesidad de crear efectos cuando transmitimos mensajes, tal como los letreros desplazables, o aplicaciones donde se tiene un mensaje donde solo cambian unos cuantos caracteres, no todo el mensaje en sí. Para estos casos la instrucción “**printf**” no es tan útil como se quisiera, sin embargo tenemos la posibilidad de generar un arreglo para el almacenamiento de cada carácter, de tal manera de manipular cada carácter por su posición dentro del arreglo.

Esta vez la declaración del arreglo cambia a la forma:

```
char letrero[]={“APARECIENDO”};  
int tamaño=sizeof(letrero);
```

donde el arreglo letrero almacena cada uno de los caracteres que nosotros definimos dentro de las comillas. Además de esto, tendríamos que contar cuantos elementos tiene nuestro arreglo para leer su contenido por su posición, sin embargo podemos usar la instrucción “**sizeof**” que nos proporcionará la cantidad de elementos dentro de nuestro arreglo.

Realizando la declaración de la manera anterior le estaremos indicando que nuestros caracteres estarán almacenados de la siguiente manera:



de tal manera que podremos acceder a cada uno de ellos por medio de su posición en el arreglo letrero.





Demo 23 - Generando un Mensaje por Arreglos

El siguiente programa muestra un mensaje en el LCD creando el efecto como si este fuera apareciendo poco a poco, auxiliandonos de un arreglo para la lectura de cada carácter.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

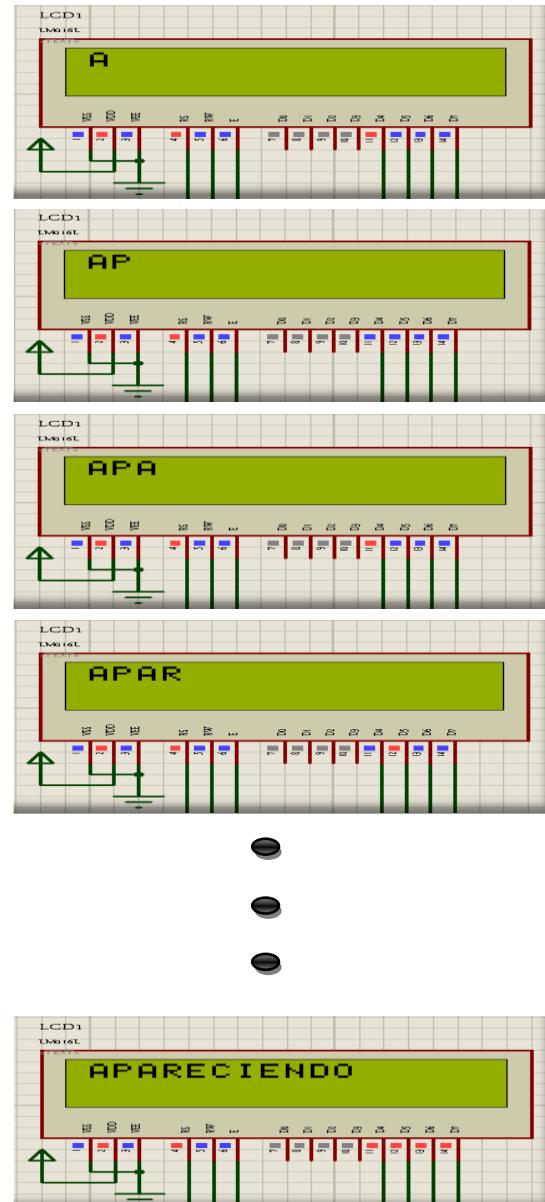
// Programa Principal
void main() {

    // Definiciones de Variables Locales
    char letrero[]{"APARECIENDO"};
    int tamano=sizeof(letrero);
    int cont;
    // Configuración de Puertos

    // Inicialización del LCD
    lcd_init();

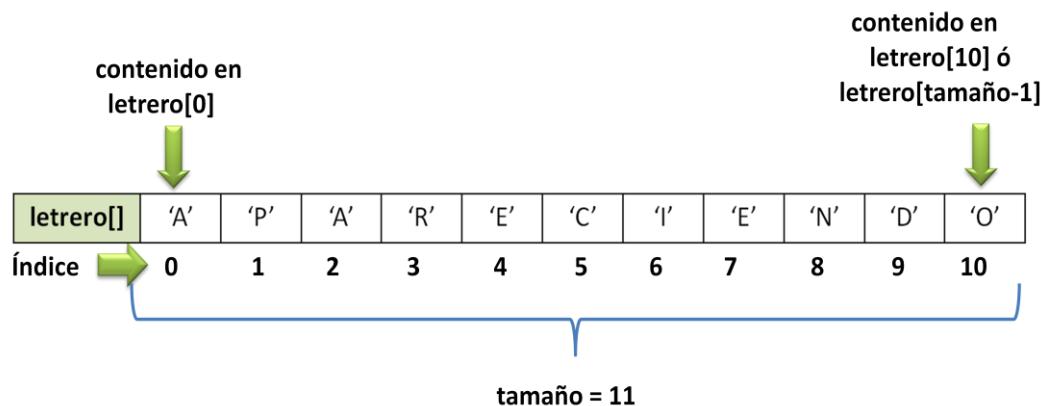
    // Bucle Principal
    while (1){
        // Instrucciones del programa
        // Limpiar el LCD
        lcd_putc('\f');
        delay_ms(300);
        for(cont=0;cont<tamano-1;cont++)
        {
            lcd_gotoxy(cont+1,1);
            lcd_putc(letrero[cont]);
            delay_ms(300);
        }

        } //end while
    }
}
```

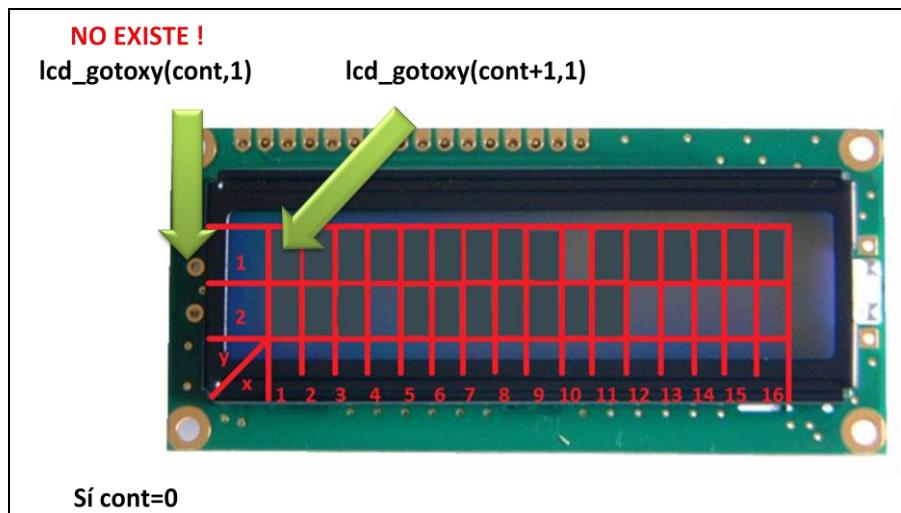




Es de vital importancia el tener cuidado en este tipo de aplicaciones el inicio y termino de los ciclos, ya que nos estamos refiriendo a diferentes posiciones de memoria para cada dispositivo que estamos controlando. Para este caso hay que recordar que en los arreglos la posición inicial es 0 y el final del ciclo está determinado por “tamaño-1” dado que tamaño es “11” y las posiciones posibles dentro del arreglo van hasta el número 10.



Por otra parte si el contador está iniciando en 0, recordemos que las coordenadas en x del LCD comienzan en 1, por lo que en vez de `lcd_gotoxy(cont,1)` usamos `lcd_gotoxy(cont+1,1)`.





Demo 24 - Monitoreando un Puerto

El siguiente programa hace uso del LCD para mostrar el contenido que existe en el puerto B, mostrando tanto el valor de cada bit por medio de los códigos 0xFF (cuadro con fondo negro) y 0xDB (cuadro con fondo blanco), así como el código hexadecimal que se forma.

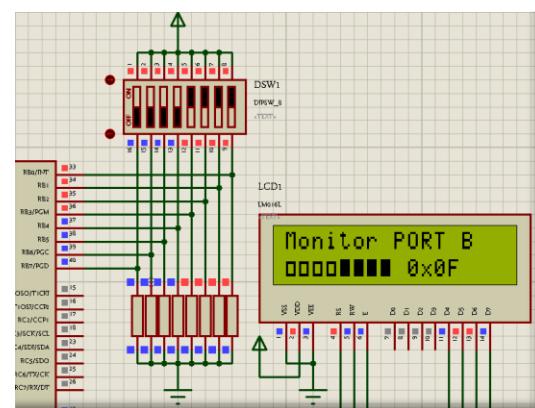
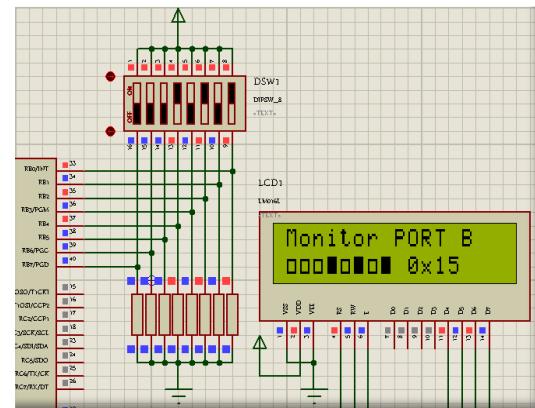
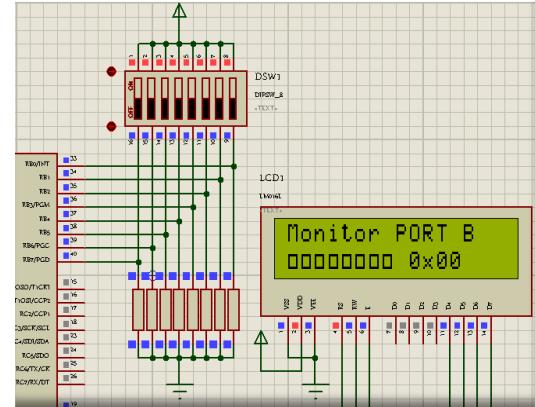
```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>
#define portB=0x06

// Programa Principal
void main() {

// Definiciones de Variables Locales
    int cont;
    int1 pin;
    int valor;
// Configuración de Puertos
    set_tris_b(0xFF);

// Inicialización del LCD
    lcd_init();
// Impresión del Mensaje título
    lcd_gotoxy(1,1);
    lcd_putc("Monitor PORT B");
// Bucle Principal
    while (1){
// Instrucciones del programa
        // Lectura del puerto B
        valor=portB;
        // Escaneo de cada pin del puerto
        lcd_gotoxy(1,2);
        cont=0;
        while(cont<8)
        {
            pin=bit_test(valor,cont);
            // Impresión del estado de cada puerto
            lcd_gotoxy(8-cont,2);

```





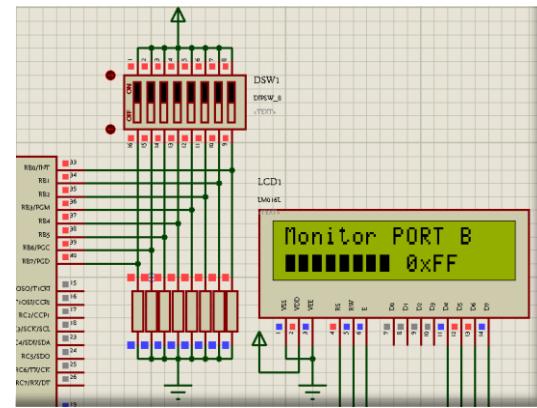
```

if(pin==1)
{
    lcd_putc(0xFF); // cuadro fondo negro
}
else
{
    lcd_putc(0xDB); //cuadro fondo blanco
}
cont++;
}

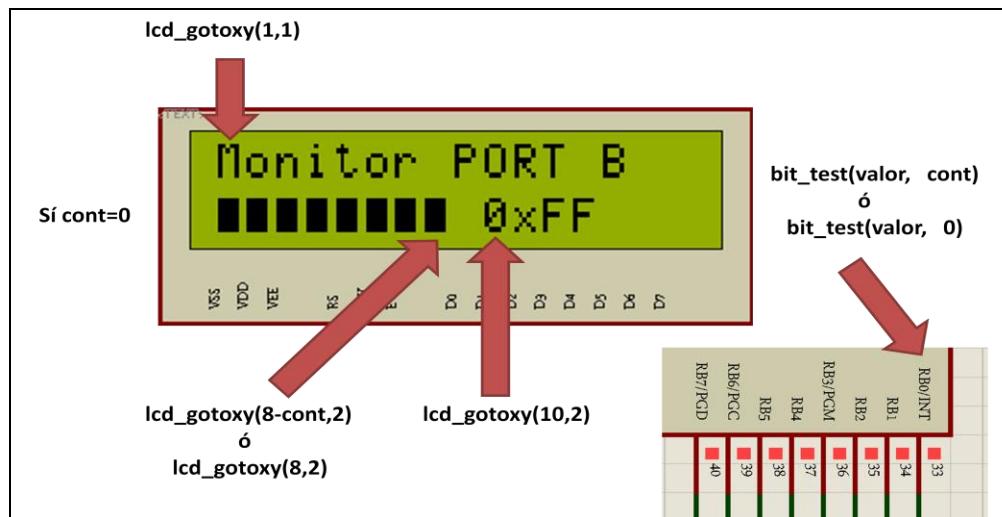
// Impresión del contenido del puerto en
hexadecimal
lcd_gotoxy(10,2);
printf(lcd_putc,"%0x%2X",valor);
delay_ms(200);
} //end while

}

```



Al igual que en el ejemplo anterior es de especial cuidado el uso de un contador para referirnos a los diferentes dispositivos que estamos usando, ya que mientras va el contador incrementando, estamos checando el estado de cada bit del puerto b desde el 0 hasta el 7 y para imprimirlas en el LCD con el bit más pequeño del lado derecho tendremos que imprimirlas en orden contrario desde la posición 8 hasta la posición 1.





Uso de un teclado Matricial

Un teclado matricial es un arreglo de pulsadores conformado de renglones y columnas interconectados, tal que al enviar información por medio de uno de sus renglones podemos recibirla por medio de una columna al presionar un pulsador y viceversa.

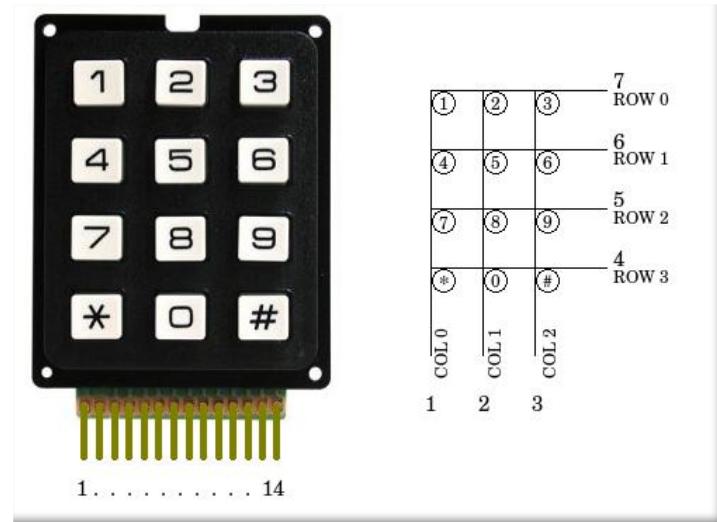
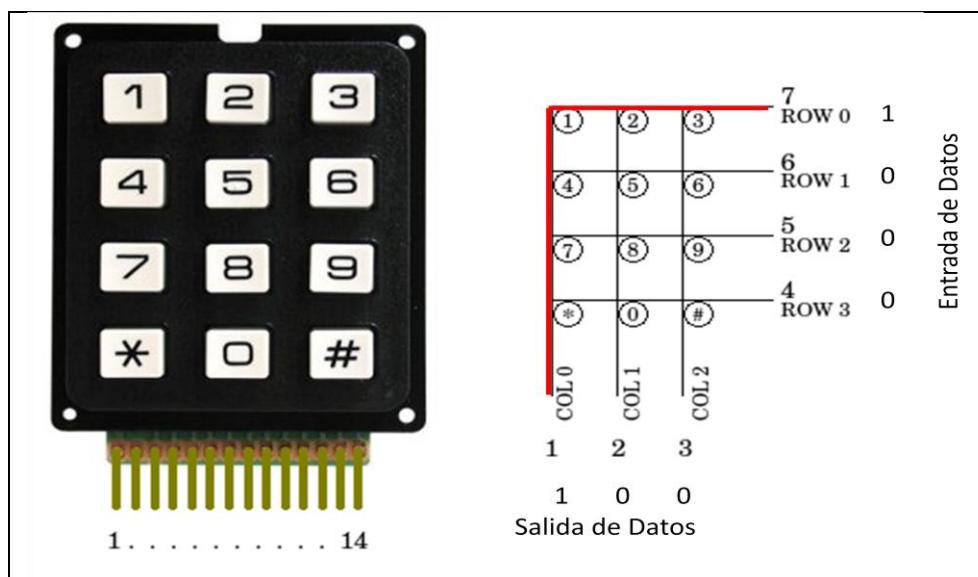


Figura 10. Teclado matricial estándar.

Por ejemplo si mandamos un “1” por medio de “COL 0” podremos recibirla en “ROW 0” al presionar la tecla “1”.

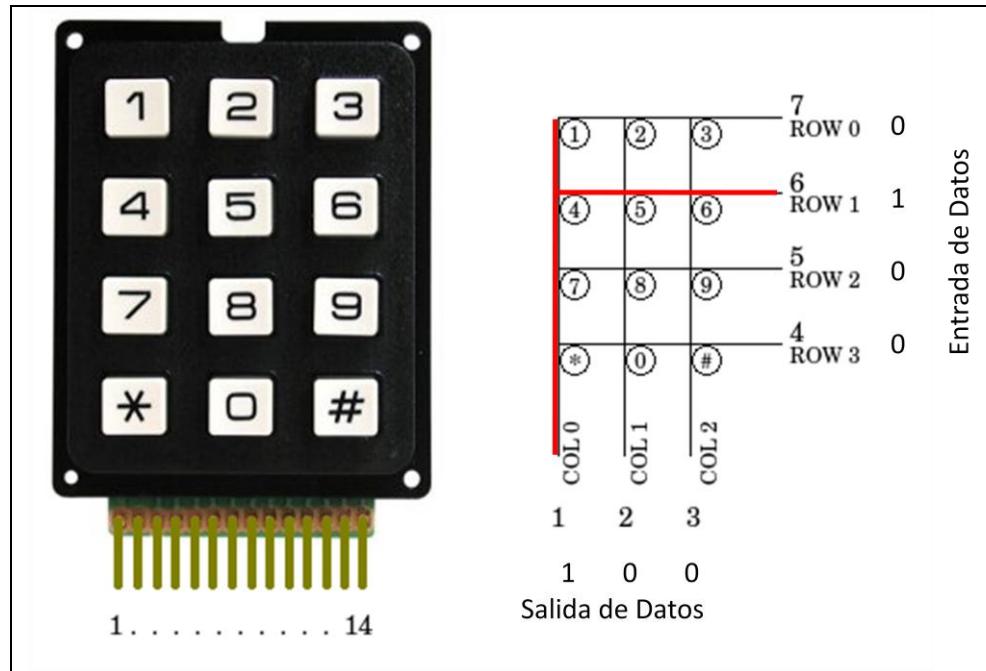


Manual – Aplicaciones Prácticas con Microcontroladores PIC

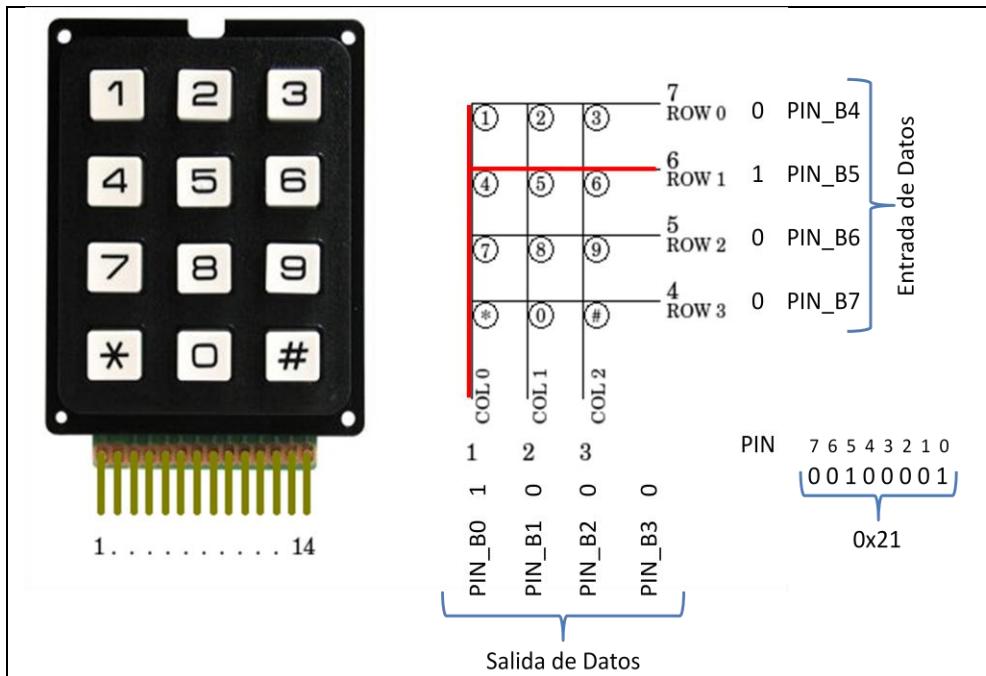
Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



O de igual manera si presionamos la tecla “4” estaremos recibiendo el dato por “ROW 1”



Como se puede observar al transmitir un dato por una columna y recibirlo por un renglón se generan códigos diferentes, por lo que si conectamos estas señales a un microcontrolador, podemos asignarle una función a cada código que se forme por la presión de alguna tecla.





Partiendo de este orden para el acomodo del teclado sobre un puerto, los códigos generados son los siguientes:

Enviando por	Tecla Presionada	Código generado
COL 0	1	0x11
	4	0x21
	7	0x41
	*	0x81
COL 1	Ninguna	0x01
	2	0x12
	5	0x22
	8	0x42
	0	0x82
COL 2	Ninguna	0x02
	3	0x14
	6	0x24
	9	0x44
	#	0x84
Ninguna	Ninguna	0x04

Tabla. Códigos generados por un teclado matricial.

Tomando en cuenta la forma en que se producen estos valores, el diagrama de conexión básico para un teclado matricial puede generarse como se muestra en la figura.

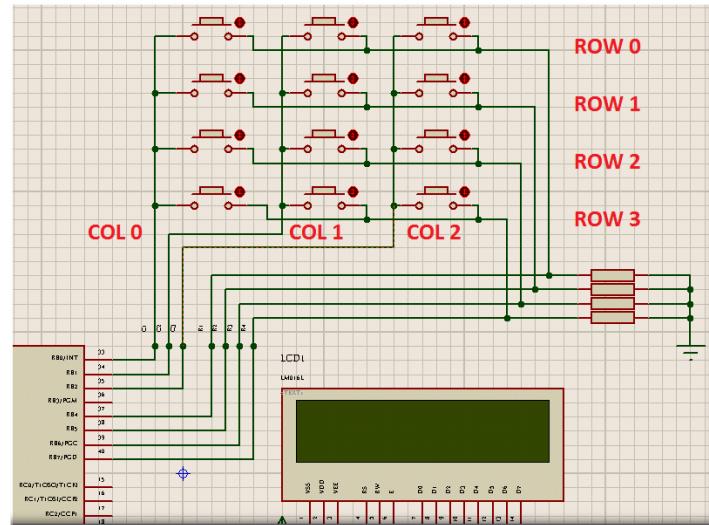


Figura 11. Conexión básica de un Teclado Matricial





Demo 25 - Usando un Teclado matricial

El siguiente programa muestra cómo se pueden usar los códigos generados en un teclado matricial, asignándole una función a cada tecla. El programa se centra en la idea de usar un puerto completo para el uso del teclado, donde la mitad de este es configurado como salida, de manera de ir activando cada uno de los pines, uno por uno, para posteriormente leer el puerto completo, verificando si se generó alguno de los códigos ante descritos tras la presión de alguna tecla.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>
#byte portB=0x06

// Definiciones de Variables Globales
char cont;
char dato;

// Declaración de Subrutinas

// Programa Principal
void main() {

// Definiciones de Variables Locales

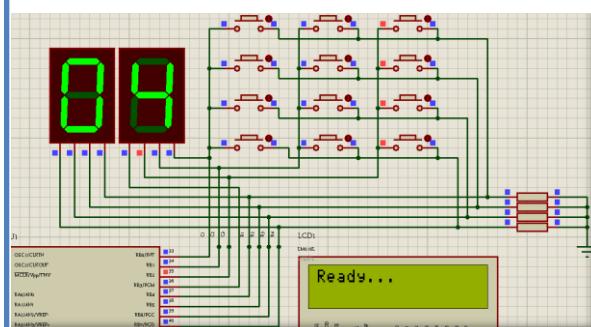
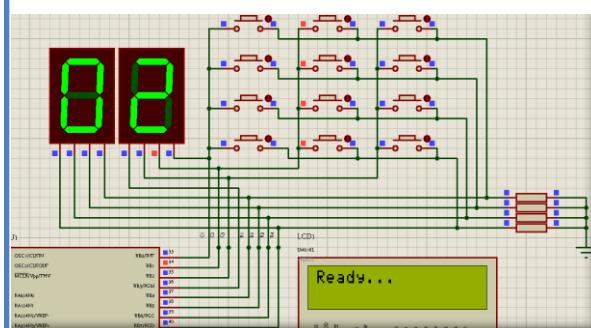
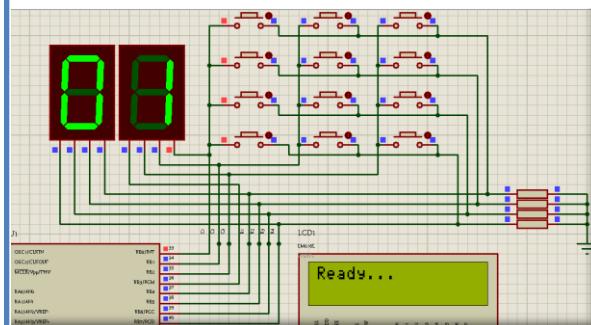
// Configuración de Puertos
lcd_init();
set_tris_b(0xF0);

// Bucle Principal
lcd_putc("Ready...\n");

while (1){
// Instrucciones del programa
cont=0;
portB=0x01;
while (cont<4)
{
delay_ms(10);
// Capturar el contenido del puerto B
dato=portB;

// Si ha recibido alguna pulsación
if((dato&0xF0)!=0){
```

SIN PRESIÓN DE TECLAS





```
//Seleccion de la accion para cada tecla (dato)
switch (dato) {
    case 0x81: lcd_putc("*"); break;
    case 0x41: lcd_putc("7"); break;
    case 0x21: lcd_putc("4"); break;
    case 0x11: lcd_putc("1"); break;

    case 0x82: lcd_putc("0"); break;
    case 0x42: lcd_putc("8"); break;
    case 0x22: lcd_putc("5"); break;
    case 0x12: lcd_putc("2"); break;

    case 0x84: lcd_putc("#"); break;
    case 0x44: lcd_putc("9"); break;
    case 0x24: lcd_putc("6"); break;
    case 0x14: lcd_putc("3"); break;

    case 0x88: lcd_putc("D"); break;
    case 0x48: lcd_putc("C"); break;
    case 0x28: lcd_putc("B"); break;
    case 0x18: lcd_putc("A"); break;
    default: break;
}

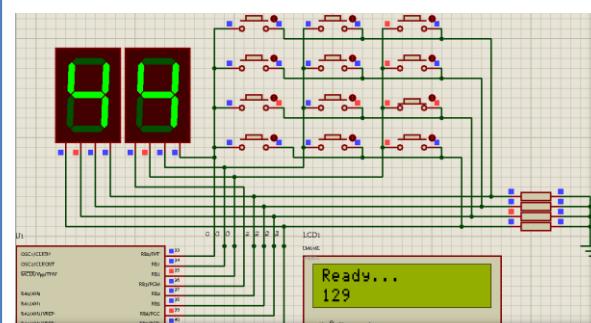
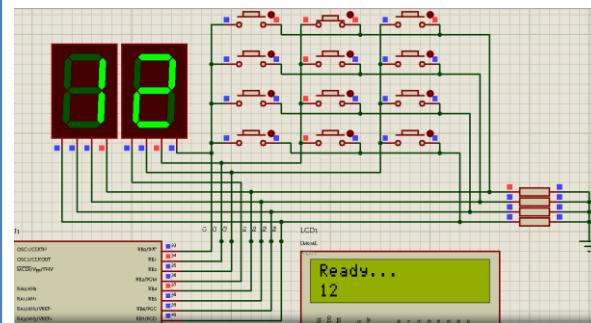
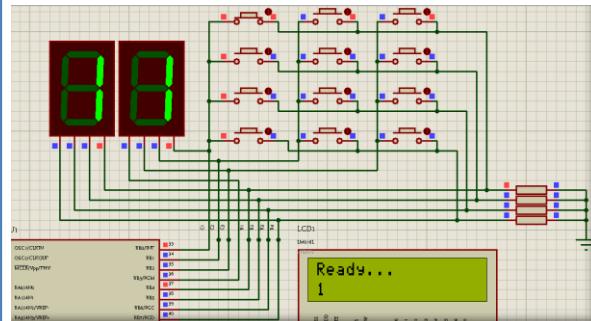
// Dar un tiempo para soltar la tecla
delay_ms(300);
}

portB<=1;
cont++;
} //end while

} //end while

}
```

PRESIONANDO TECLAS



Como se puede observar, en este caso el dato del puerto B puede tomar muchos valores, por lo que el uso de SWITCH-CASE es una mejor opción que el uso de varios IF, asignando una función cuando se genera cada código por medio del teclado matricial, que para este ejemplo, para cada código su única función es imprimir el valor de la tecla correspondiente en el LCD. Otra observación importante es el uso de la condición “**(dato&0xF0)!=0**” en un IF, donde como se ha de recordar la operación AND la podemos usar para borrar registros que no estamos ocupando y recordando lo visto en la [tabla de códigos](#), cuando no se presiona ninguna tecla existe un código en común, el primer dígito en cero del código hexadecimal, por lo que con el uso “**dato&0xF0**” estamos borrando el segundo dígito hexadecimal para solo tomar en cuenta el primer dígito, de tal



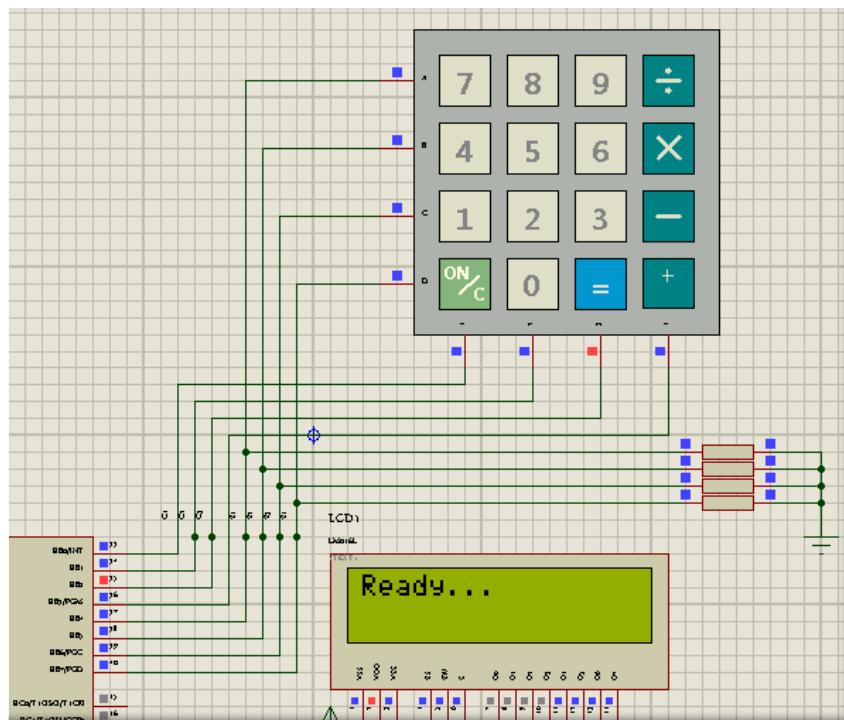
Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



forma que si al leer este registro resulta 0, significa que no se ha presionado ninguna tecla y si es diferente de 0 significa que se presionó alguna tecla.

De los conceptos anteriores de la generación y lectura de un teclado matricial y pensando en un teclado comercial solo tendremos que tener cuidado en identificar correctamente los pines de cada renglón y columna, conectando adecuadamente cada uno de ellos al puerto que deseemos usar del microcontrolador.



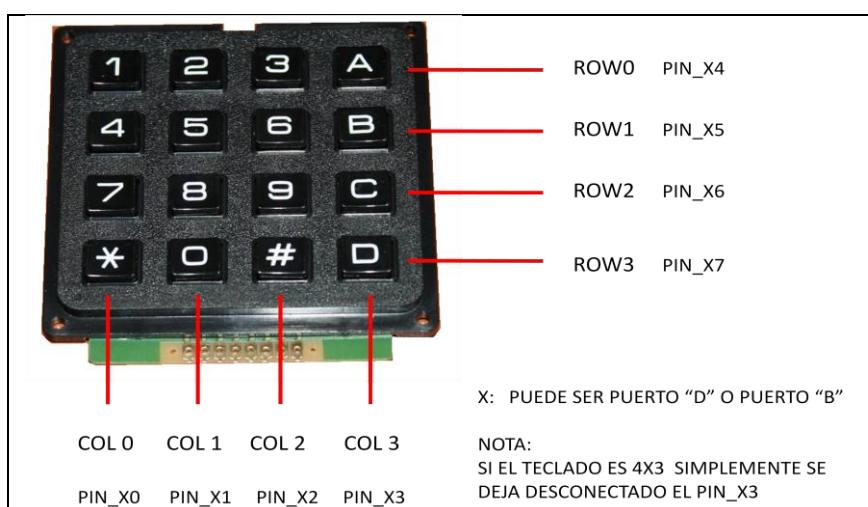


Demo 26 - Uso de un teclado Matricial por medio de una Librería

Como se pudo ver en el ejemplo anterior, la generación de los códigos en un teclado matricial es un procedimiento secuencial, donde pones un bit en “1”, checamos enseguida si se presionó alguna tecla, se le asigna una función si se presionó para finalmente mandar un “0” a ese bit, y así para cada bit de cada columna.

Existe otra forma de manipular un teclado matricial usando una librería específica para él (kbd.h), de forma similar al LCD. Sin embargo, existen diferentes librerías en Internet con modificaciones, ya que algunos programadores la modifican a su gusto, esto puede dificultar nuestro proyecto ya que se tendría que descifrar cual es el funcionamiento de cada una de ellas y como se conecta al teclado.

En nuestro caso manejaremos la librería “kbd_hm.h” también con algunas modificaciones que he hecho para usar el teclado matricial ya sea en el puerto B o puerto D según la siguiente forma de conexión.



Para definir qué puerto usaremos para el teclado matricial bastará con declarar al inicio del programa “#define use_portb_keypad TRUE” y antes de declarar la librería. Si lo hacemos así los pines de uso se asignarán al Puerto B, de lo contrario se asignarán al Puerto D.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#define use_portb_keypad TRUE
#include <kbd_hm.h>
#include <lcd.c>
```





A través de la librería podremos realizar la misma función que el demo anterior pero solo usando dos instrucciones.

Instrucción	Función
<code>kbd_init()</code>	Inicializa la librería del teclado matricial.
<code>kbd_getc()</code>	Espera a que se presione una tecla, una vez presionada y soltada regresa el código ASCII de la tecla pulsada. <code>tecla=kbd_getc();</code>

El siguiente programa muestra en la pantalla LCD, la tecla que se presiona en el teclado matricial.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#define use_portb_keypad TRUE
#include <kbd_hm.h>
#include <lcd.c>

void main() {

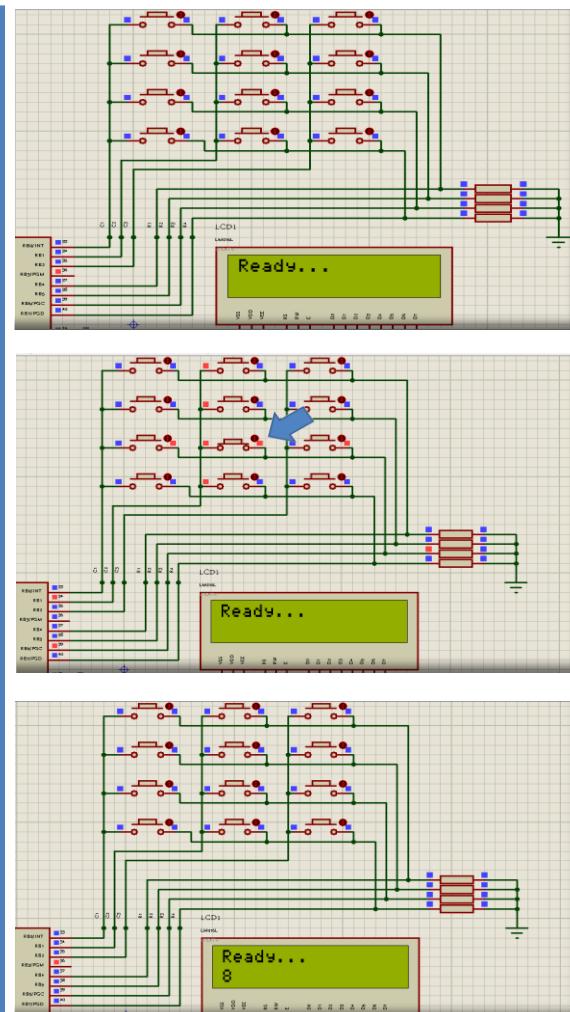
    // Definiciones de Variables Locales
    char tecla;

    // Inicialización de LCD y teclado matricial
    lcd_init();
    kbd_init();

    // Bucle Principal
    lcd_putc("Ready...\n");

    while (1){
        // Instrucciones del programa
        tecla=kbd_getc();
        lcd_putc(tecla);

        //end while
    }
}
```





Demo 27 - Menú Básico de Opciones

El ejemplo anterior es bastante práctico, ya que podemos destinar una función para cada tecla, por lo que para generar un menú de opciones solo tendremos que modificar el contenido de cada caso en el SWITCH. El siguiente programa usa la variable tecla como memoria para la última tecla presionada, de tal manera que realice una función diferente según sea el valor de esta ultima tecla.

```
#include <16f877.h>
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#byte PORT_C=0x07
#define use_portb_keypad TRUE
#include <kbd_hm.h>
#include <lcd.c>

void main() {

    // Definiciones de Variables Locales
    char tecla;

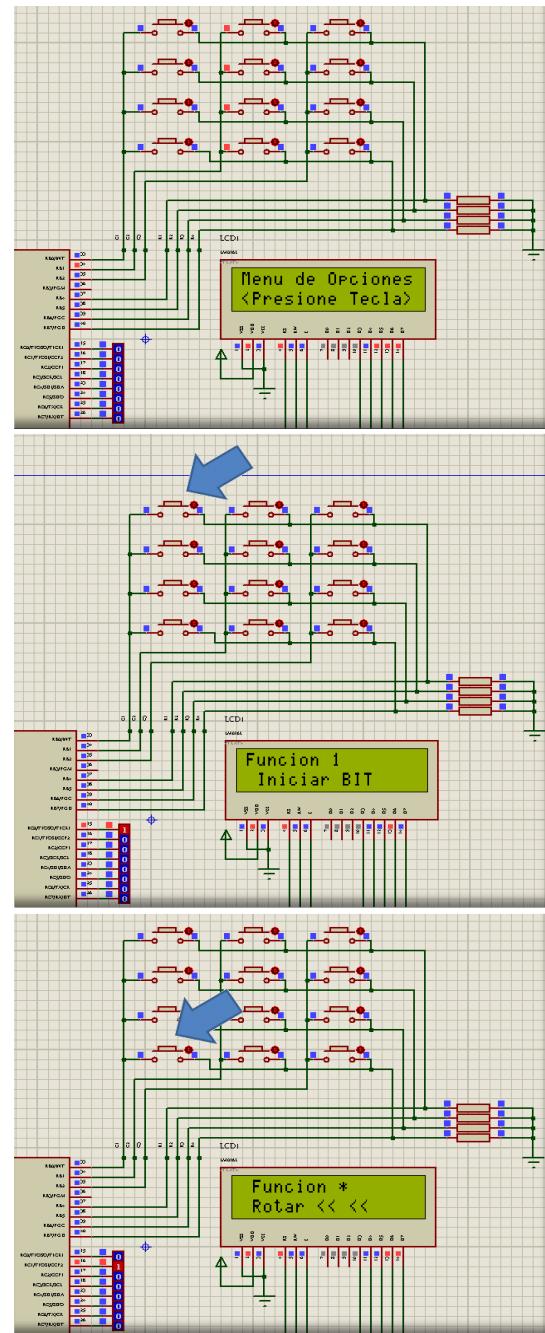
    // Inicialización de LCD y teclado matricial
    lcd_init();
    kbd_init();
    // Configuración de Puertos
    set_tris_c(0x00);

    // Valores iniciales de los puertos
    PORT_C=0x00;

    // Bucle Principal
    lcd_putc("Menu de Opciones\n");
    lcd_putc("<Presione Tecla>");

    while (1){
        // Instrucciones del programa
        tecla=kbd_getc();
        // Funciones para cada tecla
        switch(tecla)
        {

```



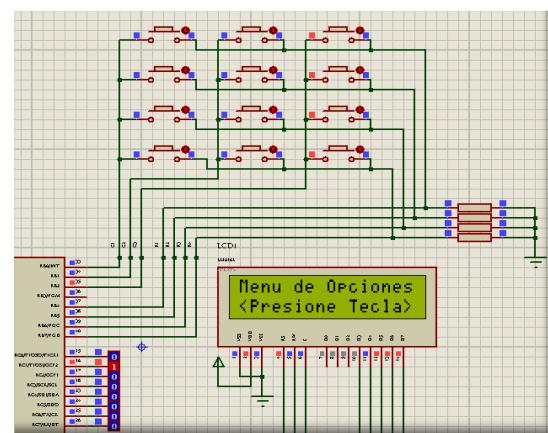
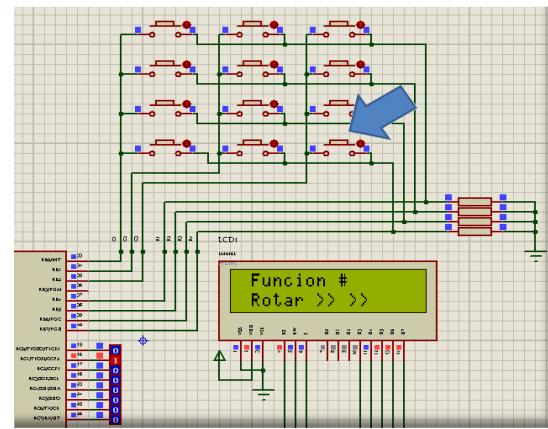
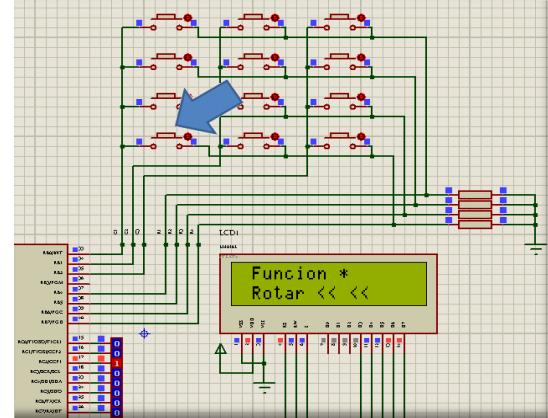


```

case '1':
    lcd_putc('\f');
    lcd_gotoxy(1,1);
    lcd_putc("Funcion 1\n");
    lcd_putc(" Iniciar BIT");
    PORT_C=0x01;
    break;
case '*':
    lcd_putc('\f');
    lcd_gotoxy(1,1);
    lcd_putc(" Funcion *\n");
    lcd_putc(" Rotar << <<");
    PORT_C<<=1;
    break;
case '#':
    lcd_putc('\f');
    lcd_gotoxy(1,1);
    lcd_putc(" Funcion #\n");
    lcd_putc(" Rotar >> >>");
    PORT_C>>=1;
    break;
default:
    break;
}
// Retardo para visualizar cada Función
delay_ms(500);
// Inicializar el Menu
lcd_putc('\f');
lcd_gotoxy(1,1);
lcd_putc("Menu de Opciones\n");
lcd_putc("<Presione Tecla>");

} //end while
}

```





CONVERTIDOR ANALÓGICO A DIGITAL

Introducción

Un Convertidor Analógico a Digital es un dispositivo que convierte una señal eléctrica continua a una representación digital, de tal manera que un rango de voltaje podrá ser representado en 2^n combinaciones. Este número de combinaciones dependerá de la capacidad del convertidor en n bits.

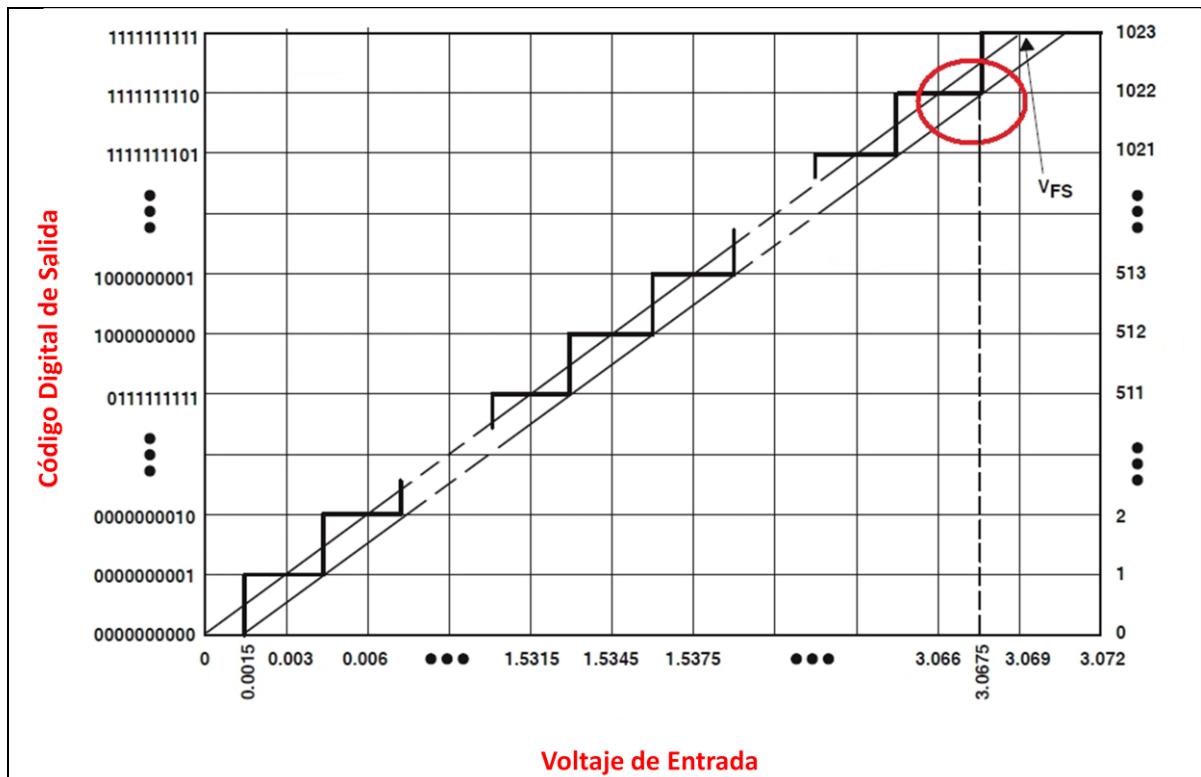


Figura 12. Conversión Analógica a Digital.

Dado que el convertidor no puede advertir todos los cambios de una señal continua, nuestra medición tiende a ser escalonada, donde la cuenta de cada bit incrementa cada vez que pasamos el nivel de voltaje más pequeño que puede advertir nuestro convertidor (**Resolución**).





La **resolución** de nuestro convertidor será entonces:

$$\text{Res} = \frac{V_{ref}}{2^n - 1}$$

donde para un voltaje de referencia de 5v y un convertidor de 10 bits, la cantidad más pequeña medible para nuestro convertidor es 4.8875mV y será nuestro primer escalón de la escala digital.

Entonces la relación entre el voltaje medido y la representación digital que nos ofrece el convertidor es:

$$V_{medido} = res(D)$$

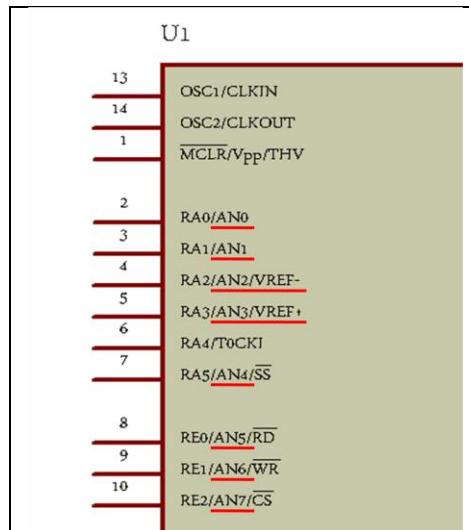
donde D es la cantidad digital ofrecida por nuestro convertidor.

Otro parámetro importante a considerar es el tiempo que tardamos en adquirir una muestra de nuestra señal continua, este tiempo está dado por la estructura del ADC del microcontrolador, frecuentemente de $T_{ACQ}=19.72\mu\text{seg}$.

Por lo que si se requiere hacer una aplicación donde decidamos cada cuando toma una muestra el microcontrolador, se deberá tomar en cuenta esta cifra.

Configuración de Entradas Analógicas

Los pines dedicados en un microcontrolador PIC como canales analógicos están etiquetados como ANx, cuidando de configurarlos adecuadamente para poder usarlos.





Esta configuración se realiza por medio de la instrucción “`setup_adc_ports()`”, donde dentro del paréntesis tendremos que especificar que configuración deseamos según las combinaciones de la siguiente tabla.

OPCIÓN	CONFIGURACIÓN
<code>NO_ANALOGS</code>	Ningún Canal Analógico
<code>ALL_ANALOG</code>	Todos como Canales Analógicos
<code>AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF</code>	Todos como Canales Analógicos con Vref+ en AN3
<code>AN0_AN1_AN2_AN3_AN4</code>	AN0 a AN4 como Canales Analógicos
<code>AN0_AN1_AN2_AN4_VSS_VREF</code>	AN0, AN1, AN2, AN4 como Canales Analógicos con Vref+ en AN3
<code>AN0_AN1_AN3</code>	AN0, AN1 y AN3 como Canales Analógicos
<code>AN0_AN1_VSS_VREF</code>	AN0 y AN1 como Canales Analógicos con Vref+ en AN3
<code>AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF</code>	Todos como Canales Analógicos con Vref+ en AN3 y Vref- en AN2
<code>AN0_AN1_AN2_AN3_AN4_AN5</code>	AN0 a AN5 como Canales Analógicos
<code>AN0_AN1_AN2_AN4_AN5_VSS_VREF</code>	AN0 a AN5 como Canales Analógicos con Vref+ en AN3
<code>AN0_AN1_AN4_AN5_VREF_VREF</code>	AN0, AN1, AN4 y AN5 como Canales Analógicos con Vref+ en AN3 y Vref- en AN2
<code>AN0_AN1_AN4_VREF_VREF</code>	AN0, AN1 y AN4 como Canales Analógicos con Vref+ en AN3 y Vref- en AN2
<code>AN0_AN1_VREF_VREF</code>	AN0 y AN1 como Canales Analógicos con Vref+ en AN3 y Vref- en AN2
<code>AN0</code>	AN0 como Canal Analógico
<code>AN0_VREF_VREF</code>	AN0 como Canal Analógico con Vref+ en AN3 y Vref- en AN2

Muchas de las configuraciones usan el voltaje que se le suministra al microcontrolador de alimentación, sin embargo también es posible la medición de un rango más cerrado usando, los voltajes de referencia de los canales AN2 y AN3, permitiéndonos ser más precisos en la medición de rangos entre 0 y 5 volts.





Por otra parte, el ADC al igual que el microcontrolador ocupa de una señal de reloj para generar las conversiones, estas fuentes de reloj también son programables por medio de la instrucción “`setup_adc()`”, pudiendo configurarlo usando las siguientes opciones:

OPCIÓN	CONFIGURACIÓN
ADC_CLOCK_INTERNAL	Funcionamiento del ADC por medio del oscilador interno.
ADC_CLOCK_DIV_32	Recomendado para máxima frecuencia en el micro de 1.25MHz.
ADC_CLOCK_DIV_8	Recomendado para máxima frecuencia en el micro de 5MHz.
ADC_CLOCK_DIV_2	Recomendado para máxima frecuencia en el micro de 20MHz.
ADC_OFF	Desactivación de la señal de reloj para el ADC.

Adquisición de señal por medio del ADC

A pesar de que en un microcontrolador podemos tener varios canales de adquisición, no es posible usarlos al mismo tiempo, ya que son canales multiplexados, es decir solo podremos acceder a uno a la vez, tal que si usáramos los 8 canales para medir 8 mediciones distintas de una a una, el tiempo de adquisición por canal aumentaría 8 veces.

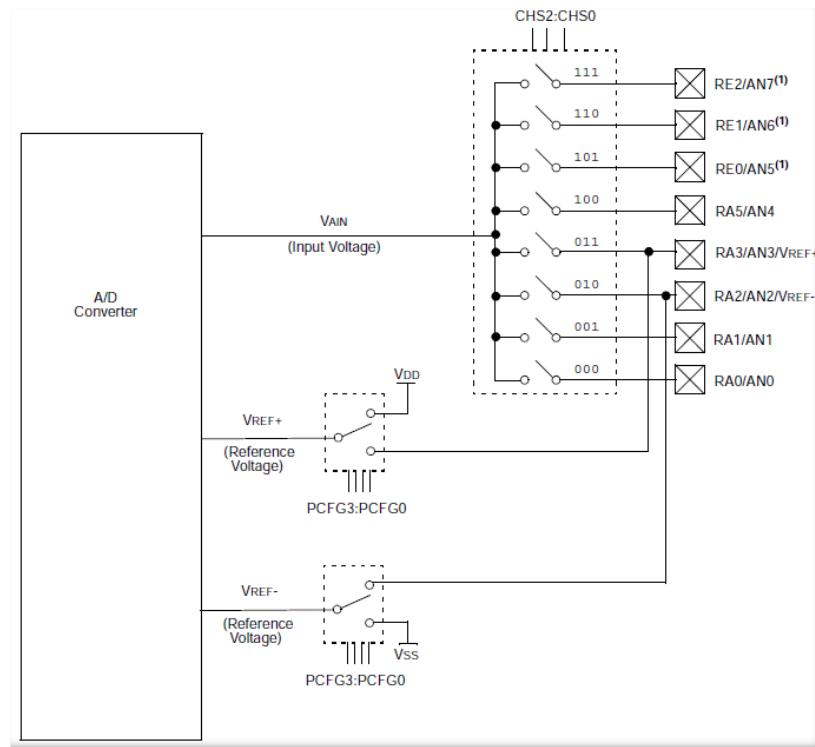


Figura 13. Estructura de los Canales de un ADC.





De igual forma, si usamos los canales AN3 y AN2 como referencias de voltaje para nuestra conversión, estos no podrán ser usados como canales de adquisición.

Para poder hacer la medición de alguno de los canales del ADC usaremos dos instrucciones principalmente:

```
set_adc_channel(0);
datoADC=read_adc();
```

La primera instrucción nos permitirá establecer por cual canal estaremos realizando la adquisición (0-7), mientras en la segunda una variable de 16 bits se encargará de almacenar la conversión realizada por la instrucción “**read_adc()**”.

Demo 28 - Lectura básica del ADC

El siguiente programa hace la lectura del canal AN0, mostrando en un LCD, el valor digital que produce la conversión A/D y el valor que le corresponde analógico.

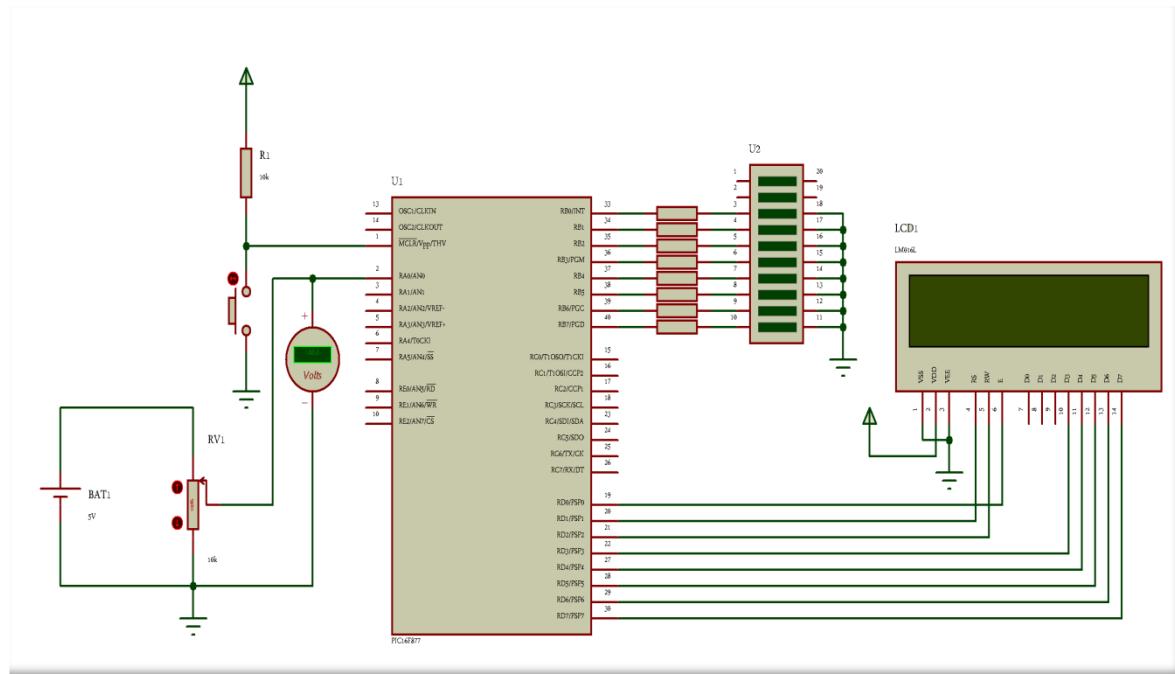


Figura 14. Diagrama de conexión para ADC y LCD.



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {

    // Definiciones de Variables Locales
    int16 datoADC;
    float voltaje;

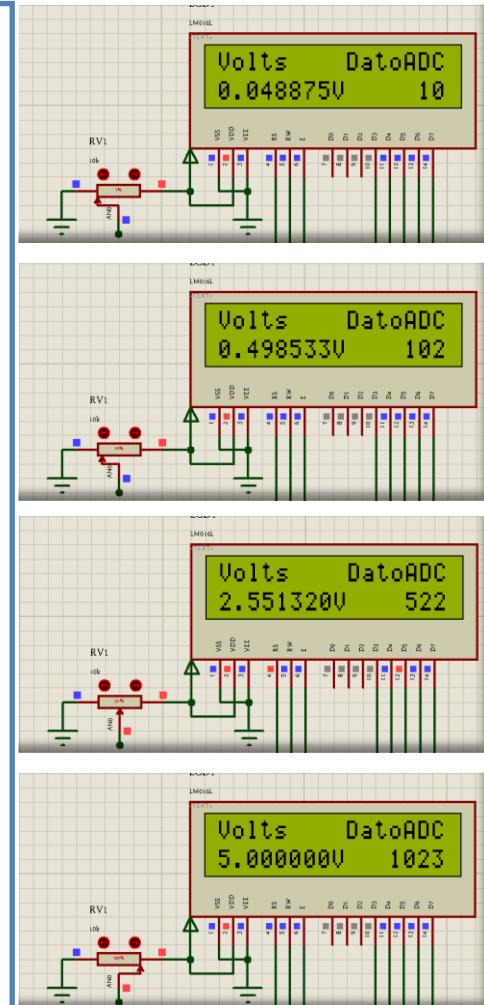
    // Configuración de Puertos
    lcd_init();
    lcd_gotoxy(1,1);
    lcd_putc("Volts DatoADC");

    // Configuración del ADC
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(AN0);
    set_adc_channel(0);
    delay_us(20);

    // Bucle Principal
    while (1){
        // Instrucciones del programa
        datoADC=read_adc();
        voltaje=0.004887585533*datoADC;
        lcd_gotoxy(1,2);
        printf(lcd_putc,"%1.6fV %4ld",voltaje,datoADC);

    } //end while
}
```

Resolución



Como se puede apreciar, ya una vez hecha la configuración del ADC antes del bucle principal no es necesaria repetirla en el programa, solo basta con la instrucción de la lectura del ADC para empezar a manipular los datos que provienen de él y las conversiones necesarias.





Demo 29 – Voltímetro a Leds.

El siguiente programa hace uso de las conversiones del ADC para ir encendiendo o apagando leds según el nivel de voltaje existente en la entrada AN0.

```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

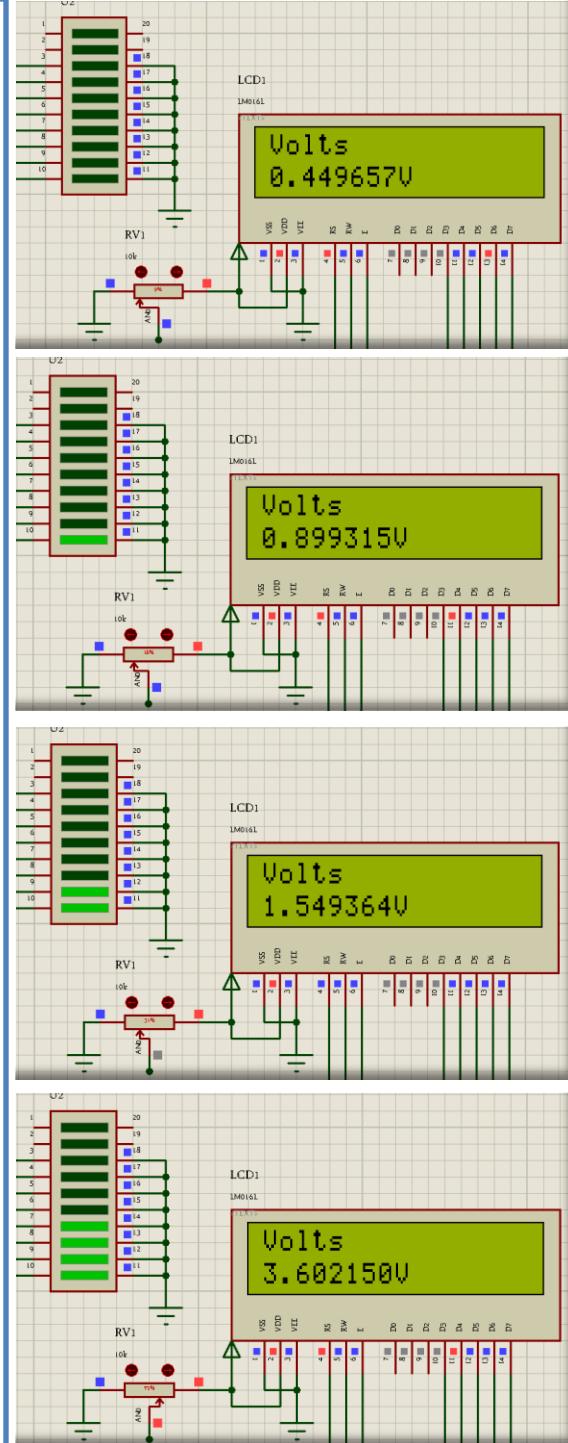
// Programa Principal
void main() {

// Definiciones de Variables Locales
int16 datoADC;
float voltaje;

// Configuración de Puertos
lcd_init();
lcd_gotoxy(1,1); lcd_putc("Volts");

// Configuración del ADC
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(AN0);
set_adc_channel(0);
delay_us(20);

// Bucle Principal
while (1){
    // Instrucciones del programa
    datoADC=read_adc();
    voltaje=0.004887585533*datoADC;
    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.6fV",voltaje);
    // Comparación por Niveles
    if(voltaje>=0.5)    output_high(PIN_B7);
    else                 output_low (PIN_B7);
    if(voltaje>=1.5)    output_high(PIN_B6);
    else                 output_low (PIN_B6);
    if(voltaje>=2.5)    output_high(PIN_B5);
    else                 output_low (PIN_B5);
    if(voltaje>=3.5)    output_high(PIN_B4);
    else                 output_low (PIN_B4);
    if(voltaje>=4.5)    output_high(PIN_B3);
    else                 output_low (PIN_B3);
} //end while
}
```





Demo 30 – Medición de dos o más Canales.

El siguiente programa muestra en pantalla la medición de dos canales ADC, mostrando en el LCD cual de las dos mediciones es mayor.

```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {

// Definiciones de Variables Locales
int16 datoADC;
float voltajeAN0, voltajeAN1;

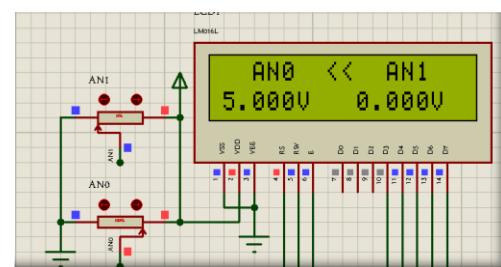
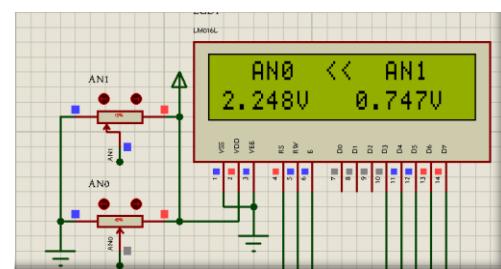
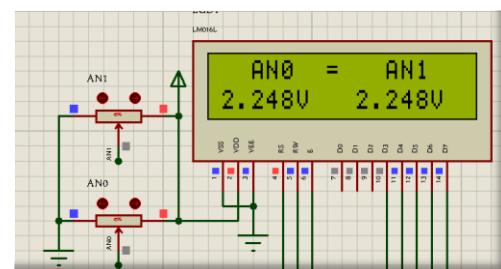
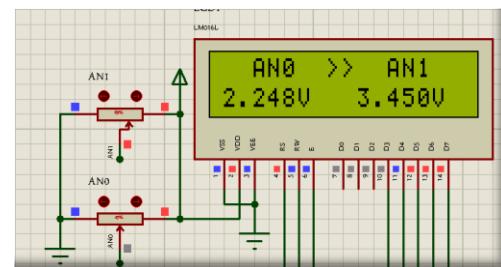
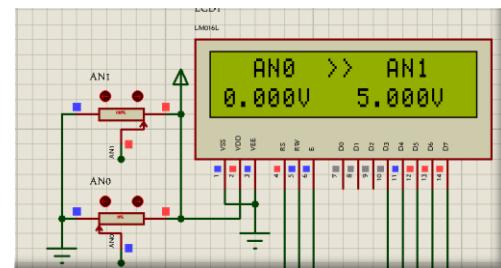
// Configuración de Puertos
lcd_init();
lcd_gotoxy(1,1); lcd_putc(" AN0      AN1");

// Configuración del ADC
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(ALL_ANALOG);

// Bucle Principal
while (1){
    // Instrucciones del programa
    // Medición del Canal AN0
    set_adc_channel(0);
    delay_us(20);
    datoADC=read_adc();
    voltajeAN0=0.004887585533*datoADC;

    // Medición del Canal AN1
    set_adc_channel(1);
    delay_us(20);
    datoADC=read_adc();
    voltajeAN1=0.004887585533*datoADC;

    // Mostrar Valores en el LCD
    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.3fV  %1.3fV",voltajeAN0,
    voltajeAN1);
}
}
```





```
// Comparación entre valores
lcd_gotoxy(8,1);
if(voltajeAN0>voltajeAN1)    lcd_putc("<<");
else
    if(voltajeAN0<voltajeAN1)  lcd_putc(">>");
    else                      lcd_putc("= ");

// Retardo entre cada Medición
delay_ms(100);

} //end while

}
```

Como se puede observar, para leer dos canales primero debemos seleccionar el primer canal, realizar la lectura del ADC y posteriormente cambiar el canal para hacer la nueva lectura. Es importante que después de hacer el cambio de canal, se realice un pequeño retardo, con el fin de darle tiempo al microcontrolador para que realice el cambio adecuadamente.





Especificación de un Rango de Medición.

En ocasiones es necesario el medir un rango más pequeño de voltaje, en lugar de mediciones entre 0 y 5v, el ADC de un microcontrolador PIC posee pines adicionales por medio de los cuales especificar este rango de voltaje, de tal manera que la medición sea más precisa. Si por medio de los pines AN2 y AN3 introducimos referencias de voltaje, la medición del ADC se centrará sobre ese rango, tal que, si el voltaje de entrada está por debajo del Voltaje de Referencia Negativo (Vref-) el convertidor arrojará el mínimo valor digital posible, y de forma similar, si el voltaje de entrada esta por arriba del Voltaje de Referencia Positivo (Vref+) el convertidor arrojará el máximo valor digital posible. De esta forma solo los valores de entrada que estén dentro del rango especificado por Vref+ y Vref- serán devueltos por el convertidor como escala digital ([figura 15](#)).

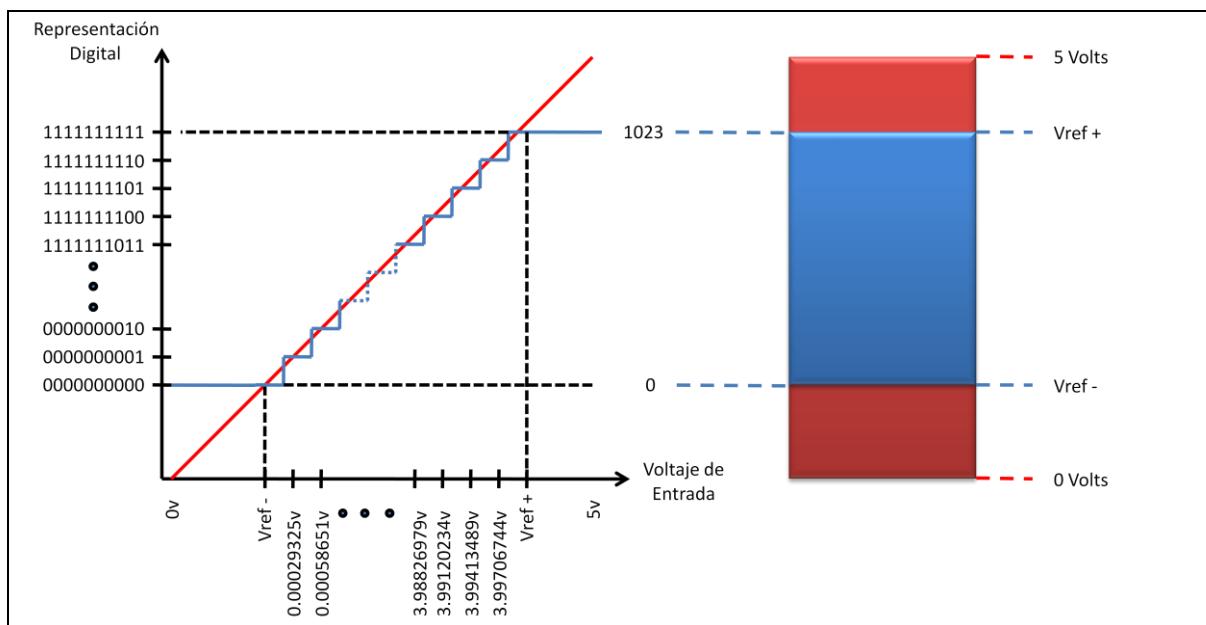


Figura 15. Usando voltajes de Referencia para la conversión A/D.

Tomando en cuenta lo anterior la Resolución de nuestro convertidor, así como la relación entre la representación digital y el valor medido cambian a:

$$\text{res} = \frac{V_{\text{ref}}^+ - V_{\text{ref}}^-}{2^n - 1} \quad \text{y} \quad V_{\text{medido}} = \text{res}(D) + V_{\text{ref}}^-$$





Por ejemplo, supóngase que se desea realizar mediciones que se encuentran en un rango entre 1v y 4v, entonces se deberá suministrar al microcontrolador esas referencias de voltaje por medio de los pines AN2 (Vref-) y AN3 (Vref+), además de especificar en la programación del micro que se usarán los pines de esta manera.

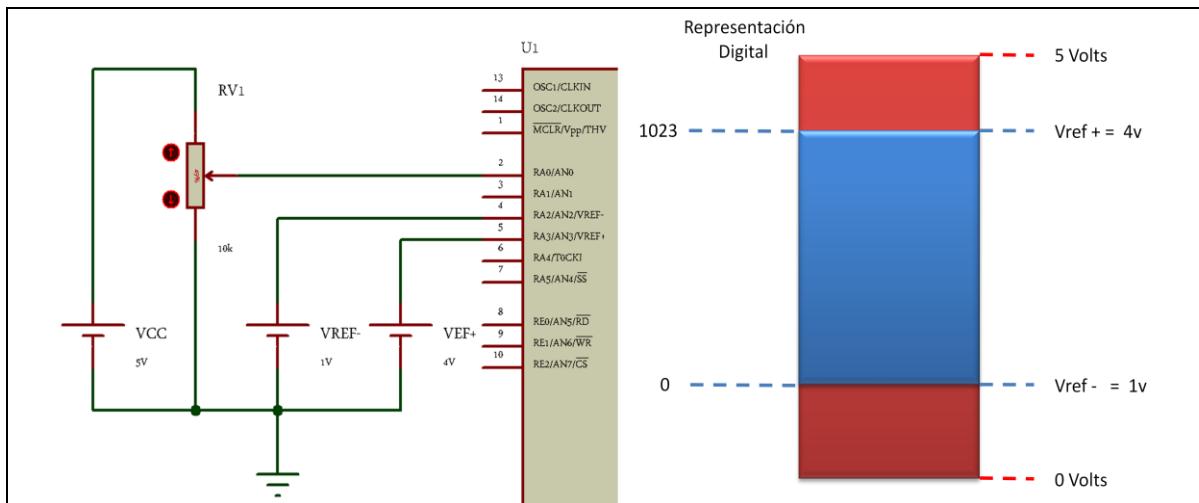


Figura 16. Conexión para ADC usando referencias de voltaje.

De esta manera entonces nuestra resolución será:

$$\text{res} = \frac{Vref^+ - Vref^-}{2^n - 1} = \frac{4-1}{1023}$$

$\text{res} = 2.932551332 \text{mV}$

Nótese que la resolución a comparación de los ejemplos anteriores ha aumentado, ya que el valor más pequeño medible ahora es de 2.93mV. Y dado el cambio ahora la relación voltaje medido contra representación digital queda:

$$V_{medido} = res(D) + Vref^-$$





Demo 31 - Especificando un Rango de Medición.

El siguiente ejemplo muestra el efecto de elegir un rango de medición sobre la lectura del ADC, mostrando su comportamiento por debajo y por arriba de los voltajes de referencia 1v y 4v.

```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {

    // Definiciones de Variables Locales
    int16 datoADC;
    float voltaje;

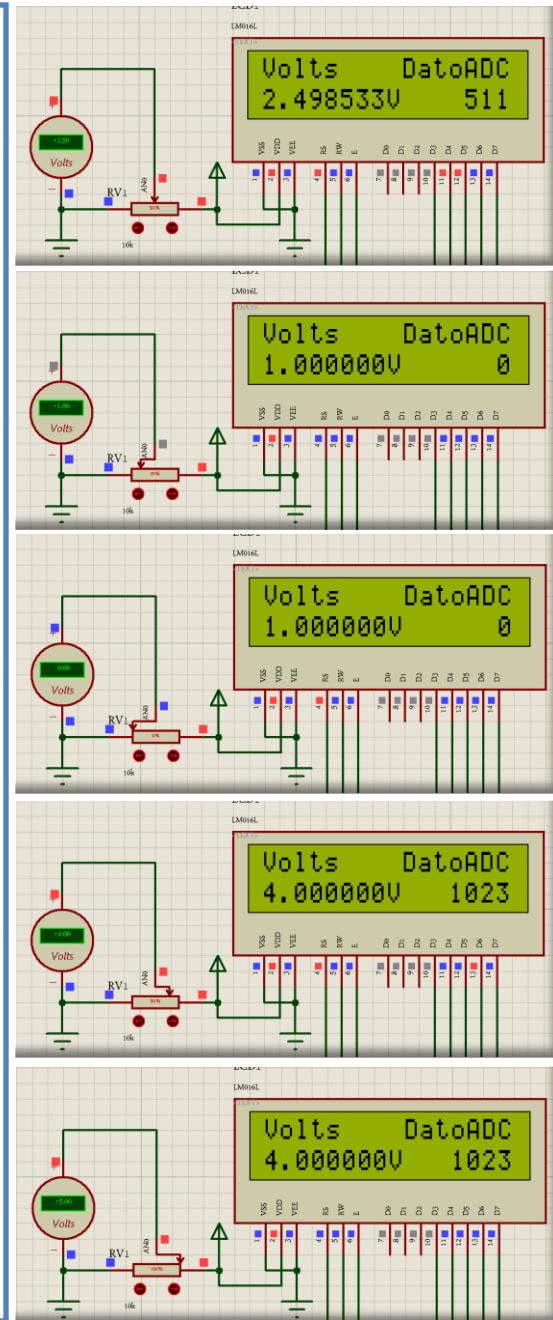
    // Configuración de Puertos
    lcd_init();
    lcd_gotoxy(1,1); lcd_putc("Volts DatoADC");

    // Configuración del ADC
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(AN0_VREF_VREF );
    set_adc_channel(0);
    delay_us(20);

    // Bucle Principal
    while (1){
        // Instrucciones del programa
        datoADC=read_adc();
        voltaje=0.00293255132*datoADC+1;
        lcd_gotoxy(1,2);
        printf(lcd_putc,"%1.6fV
        %4ld",voltaje,datoADC);

    }//end while

}
```





Demo 32 - Eliminando el Parpadeo en el LCD.

Muchas ocasiones cuando hacemos mediciones y visualizamos por medio de un LCD, nos encontramos que la pantalla parpadea, este efecto sucede por la impresión en el LCD tan rápida, si las mediciones que estamos imprimiendo vienen del ADC o de algún otro sensor cuya velocidad es alta, el LCD puede estar imprimiendo hasta 200 veces por segundo cuando la vista humana solo requiere de 30 imágenes por segundo para detectar el movimiento, por lo que estaremos observando cómo parpadea sin parar nuestro LCD.

Una forma de evitar este parpadeo, es por medio de un retardo considerable usando la instrucción “delay_ms” después de mandar a imprimir en el LCD, sin embargo si el retardo es muy amplio, estaremos evitando que el microcontrolador realice mediciones rápidas.

La segunda forma de evitar el parpadeo es por medio de una memoria adicional de las mediciones, cuya función será detectar si en las mediciones ocurrió un cambio, de forma que solo se mande a imprimir en el LCD para actualizar la información si es necesario. El siguiente programa es un ejemplo de este caso.

```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {

    // Definiciones de Variables Locales
    int16 datoADC;
    int16 ultimo_datoADC=-1;
    float voltaje;

    // Configuración de Puertos
    lcd_init();
    lcd_gotoxy(1,1); lcd_putc("Volts  DatoADC");

    // Configuración del ADC
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(AN0);
    set_adc_channel(0);
    delay_us(20);

    // Bucle Principal
    while (1){
        // Instrucciones del programa
        datoADC=read_adc();
```





```
// Verificar si hubo un cambio
if(datoADC!=ultimo_datoADC)
{
    voltaje=0.004887585533*datoADC;
    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.6fV
        %4ld",voltaje,datoADC);
}
// Recordar último dato leído
ultimo_datoADC=datoADC;
}//end while

}
```

Cabe mencionar que el parpadeo es casi indistinguible en simulación, sin embargo en práctica ocurre, para la realización de este ejercicio se recomienda comparar en práctica el programa [demo28](#) y [demo32](#).

Otra recomendación sugerida para la eliminación del parpadeo es contemplar en el programa, el solo imprimir en ubicaciones del LCD que se van a actualizar datos, evitando el imprimir nuevamente letreros que siempre permanecerán fijos.





Demo 33 - El Termómetro básico.

Algunas de las aplicaciones comunes de un microcontrolador es el censar variables tales como voltaje, corriente y temperatura. En el caso de la temperatura, uno de los sensores más usados es el sensor LM35, el cuál proporciona como salida 10mV por cada grado Celsius en el ambiente sensado, esto significaría que a una temperatura de 25°C el sensor proporciona de salida 0.25v, un valor muy pequeño cuando se desea un sensor para temperaturas entre los 0° y 100°C, por lo que muchas de las ocasiones es necesario un acondicionador de señal para que nos proporcione valores entre 0 y 5v.

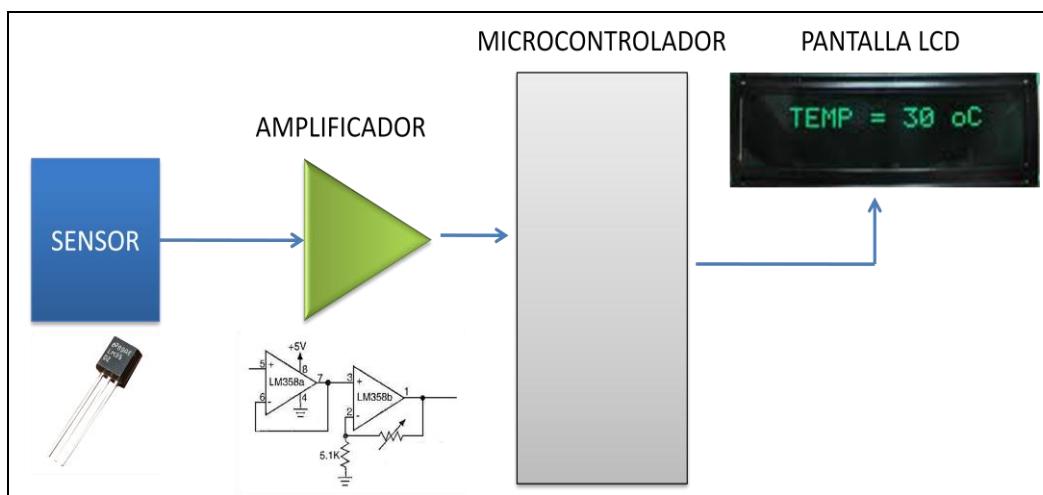


Figura 17. Sensor de Temperatura típico.

Una forma rápida de hacer mediciones precisas a través del microcontrolador es hacer la conexión directa del sensor LM35 a un pin analógico, y configurar los pines para aceptar voltajes de referencia. Por ejemplo sí se desea realizar un termómetro que mida entre los 0°C y 120°C, el sensor nos otorgará voltajes de salida entre 0 volts y 1.2 volts, sabiendo esto podemos usar los voltajes de referencia del microcontrolador estableciendo Vref- en 0 volts y Vref + en 1.2 volts, esto permitirá una máxima precisión en la conversión del ADC sin necesidad de la etapa de amplificación, siendo así nuestra resolución para el convertidor será de 1.173020528mV que están por debajo de los 10mV que entrega en sensor LM35.

El siguiente ejemplo muestra la medición de temperatura del LM35, usando los voltajes de referencia antes mencionados y mostrando su valor en el LCD.



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

// Programa Principal
void main() {

// Definiciones de Variables Locales
int16 datoADC;
unsigned int temp;
float voltaje;

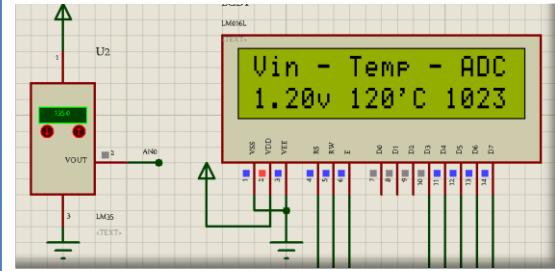
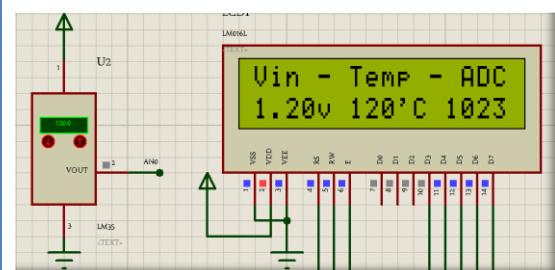
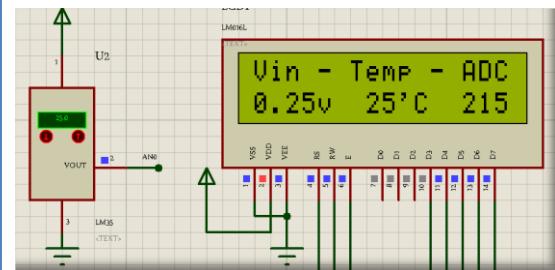
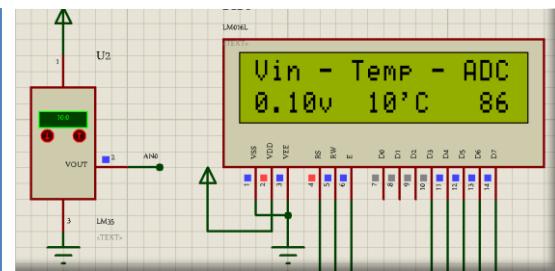
// Configuración de Puertos
lcd_init();
lcd_gotoxy(1,1); lcd_putc("Vin - Temp - ADC");

// Configuración del ADC
setup_adc(ADC_CLOCK_INTERNAL);
setup_adc_ports(AN0_VREF_VREF );
set_adc_channel(0);
delay_us(20);

// Bucle Principal
while (1){
    // Instrucciones del programa
    datoADC=read_adc();
    voltaje=0.001173020528*datoADC;
    temp=100*voltaje;
    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.2fv %3u'C
        %4ld",voltaje,temp,datoADC);

} //end while

}
```



Nota: Recuérdese que al usar los voltajes de referencia del convertidor ADC, todo aquello que este fuera del rango especificado, no causara ningún efecto en la medición.





Demo 34 - Controlador ON-OFF de Temperatura.

Un ejercicio básico de control, se tiene cuando se desea automatizar dispositivos que deben mantener una temperatura en un rango aceptable, dispositivos que generalmente trabajan en base al calentamiento que produce una resistencia calefactora en un ambiente cerrado. Este tipo de sistemas generalmente están diseñados por medio de un sensor, un acondicionador de señal, un elemento que genere el control y un relevador que lleve la acción de elevar o disminuir la temperatura.

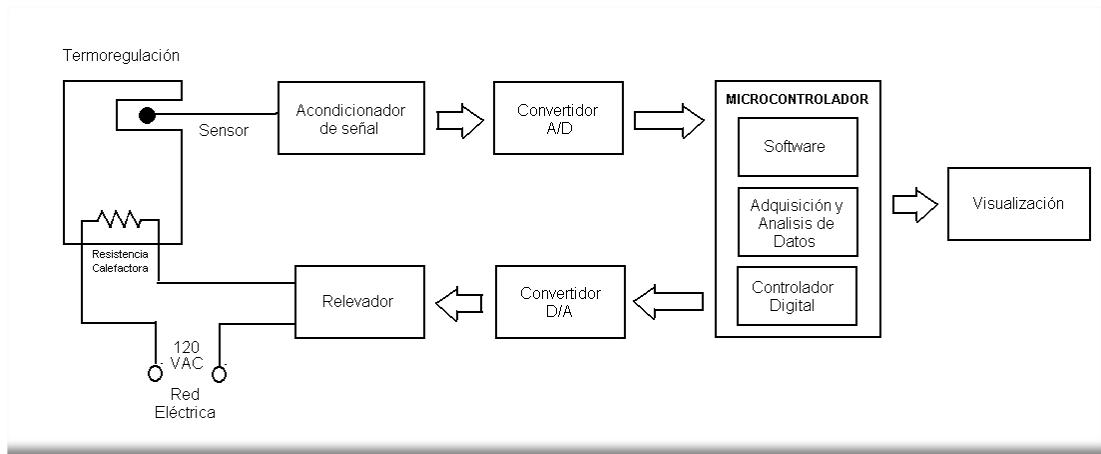


Figura 18. Control de Temperatura basado en un microcontrolador.

Basados en nuestro microcontrolador PIC, este se encargará de realizar la medición a partir del convertidor A/D, procesar la información, visualizarla en un LCD y enviar la señal de apagado o encendido de la resistencia calefactora, de tal manera que la temperatura este en un rango que nosotros determinaremos.

Para la simulación de nuestro sistema usaremos un emulador de horno, el cuál consta de una entrada digital para apagar o encender la resistencia calefactora y también de un sensor que provee la temperatura en voltaje, tal que otorga 25mV por cada °C presente en el horno.

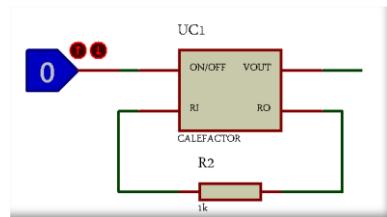


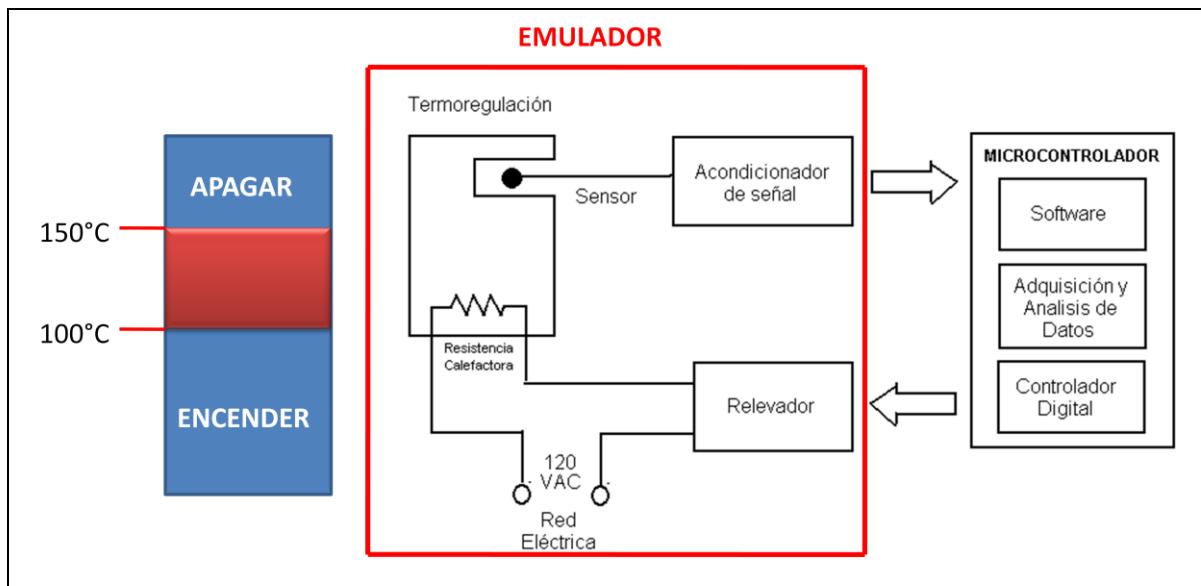
Figura 19. Emulador de horno.





Así mismo, el emulador posee dos terminales para conexión de una resistencia de referencia, la cual configura el tiempo en que el horno llega a la máxima temperatura de 200°C, y proporcionando 1 seg por cada kOhm.

Por medio de lo anterior, el microcontrolador solo se encargará de medir el voltaje de este emulador y mandar encender o apagar el horno por medio de un PIN.



El objetivo del controlador ON-OFF será el mantener la temperatura en un rango entre los 100°C y los 150°C, tal que si la temperatura está por debajo de los 100°C se debe encender el horno y si esta por arriba de los 150°C se deberá apagar.

Ahora bien recordando que el emulador entrega 25mV por cada °C, la relación entre el voltaje medido y la temperatura equivalente es:

$$\text{Temp} = 40V_{IN}$$

Siendo así, el programa del microcontrolador se encargará de medir el voltaje del sensor, convertirlo a una escala de temperatura y revisar si está esta dentro de los rangos especificados, para mostrar la temperatura actual del horno y dar la orden que mando para mantener la temperatura dentro de este rango por medio del PIN_E1, que se encargará de enviar un '1' o un '0' para prender o apagar la resistencia calefactora del horno.



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



El diagrama esquemático de nuestro circuito se muestra a continuación.

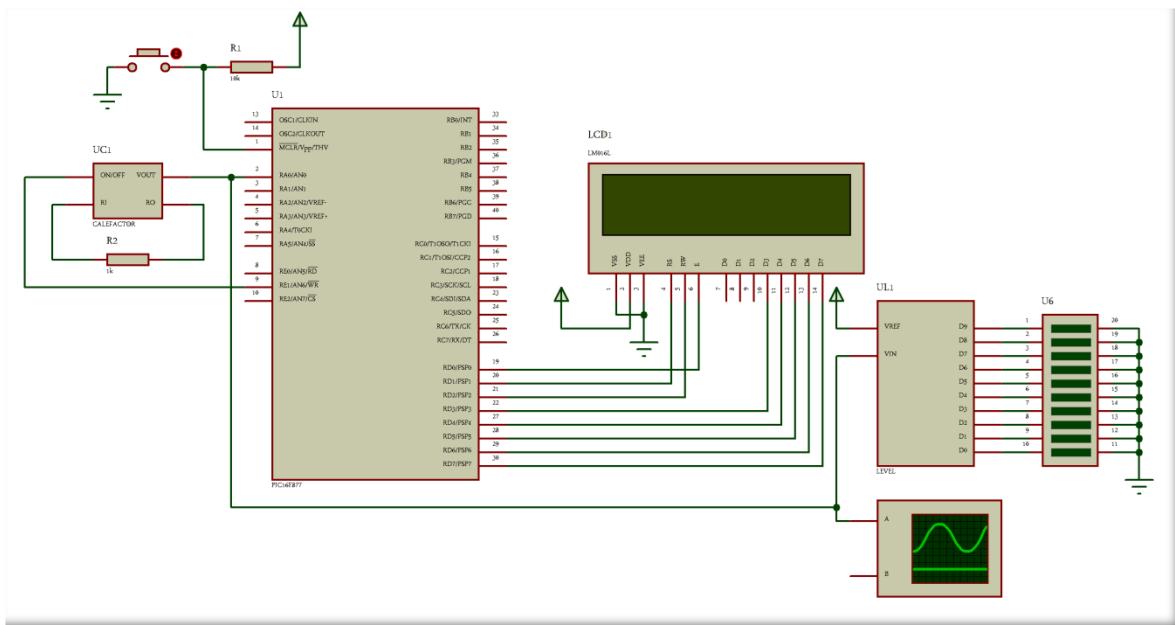


Figura 20. Diagrama esquemático del control ON-OFF de temperatura.

```
#include <16f877.h>
#DEVICE ADC=10
#FUSES XT,NOWDT,NOPROTECT,PUT
#USE DELAY( CLOCK=4000000 )
#include <lcd.c>

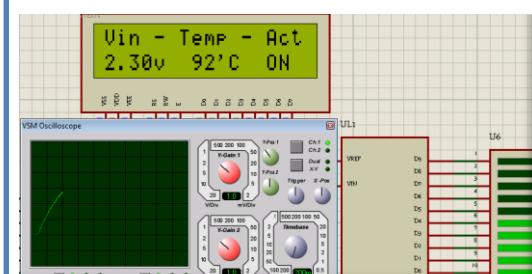
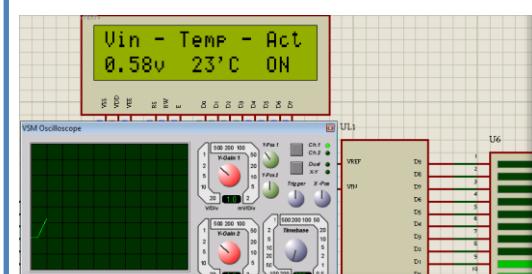
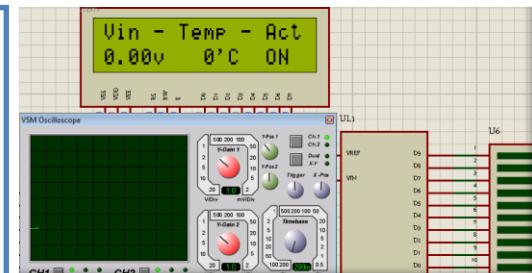
// Programa Principal
void main() {

    // Definiciones de Variables Locales
    int16 datoADC;
    float temp;
    float voltaje;

    // Configuración de Puertos
    lcd_init();
    lcd_gotoxy(1,1); lcd_putc("Vin - Temp - Act");

    // Configuración del ADC
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_adc_ports(AN0);
    set_adc_channel(0);
    delay_us(20);

    while(1) {
        datoADC = adc_read();
        temp = (datoADC * 0.02) + 27;
        voltaje = (datoADC * 0.02) + 2.0;
        lcd_gotoxy(1,1); lcd_putc("Vin - Temp - Act");
        lcd_gotoxy(1,2); lcd_putf("%d.%d", (datoADC / 100));
        lcd_gotoxy(2,2); lcd_putf("%d.%d", temp);
        lcd_gotoxy(3,2); lcd_putc("ON");
        delay_ms(1000);
    }
}
```



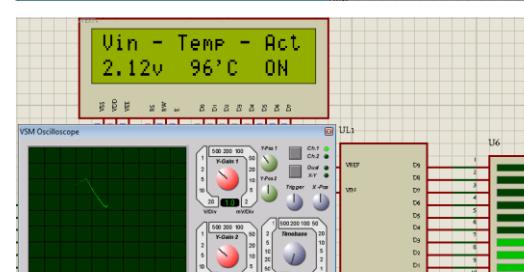
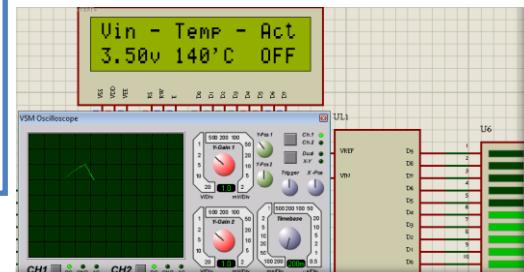
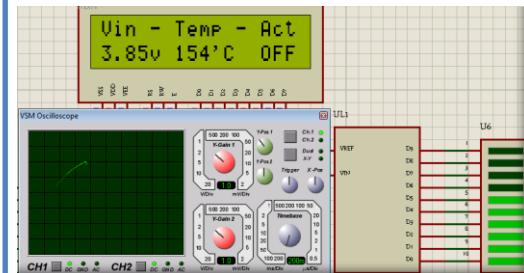
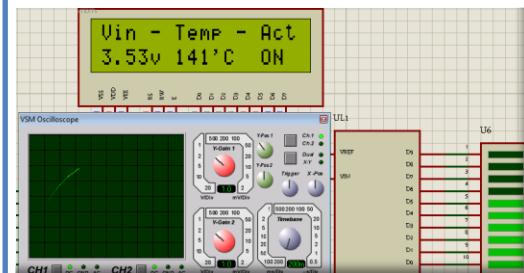
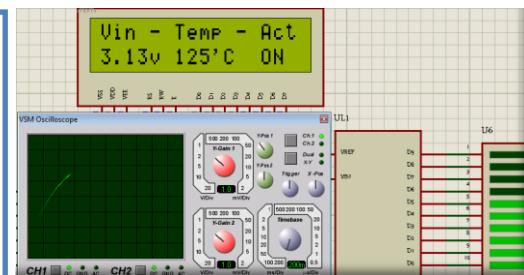
Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



```
// Bucle Principal
while (1){
    // Instrucciones del programa
    datoADC=read_adc();
    voltaje=0.004887585533*datoADC;
    temp=40*voltaje;
    // Verificación de la temperatura
    lcd_gotoxy(1,2);
    printf(lcd_putc,"%1.2fv
        %3.0f'C",voltaje,temp);
    if(temp>150)
    {
        output_low(PIN_E1);
        lcd_gotoxy(14,2);
        lcd_putc("OFF");
    }
    if(temp<100)
    {
        output_high(PIN_E1);
        lcd_gotoxy(14,2);
        lcd_putc("ON ");
    }

} //end while
}
```



Como se puede observar en el programa, cada vez se hace una lectura del ADC, se manda a imprimir cuál es el estado de la temperatura y solo cuando se superan los límites del rango de temperatura, se manda a imprimir un letrero de la acción que realizo el microcontrolador. Dado el programa, si se requiere cambiar los límites de operación del controlador ON-OFF, solo se tendrá que cambiar los datos en la comparación.





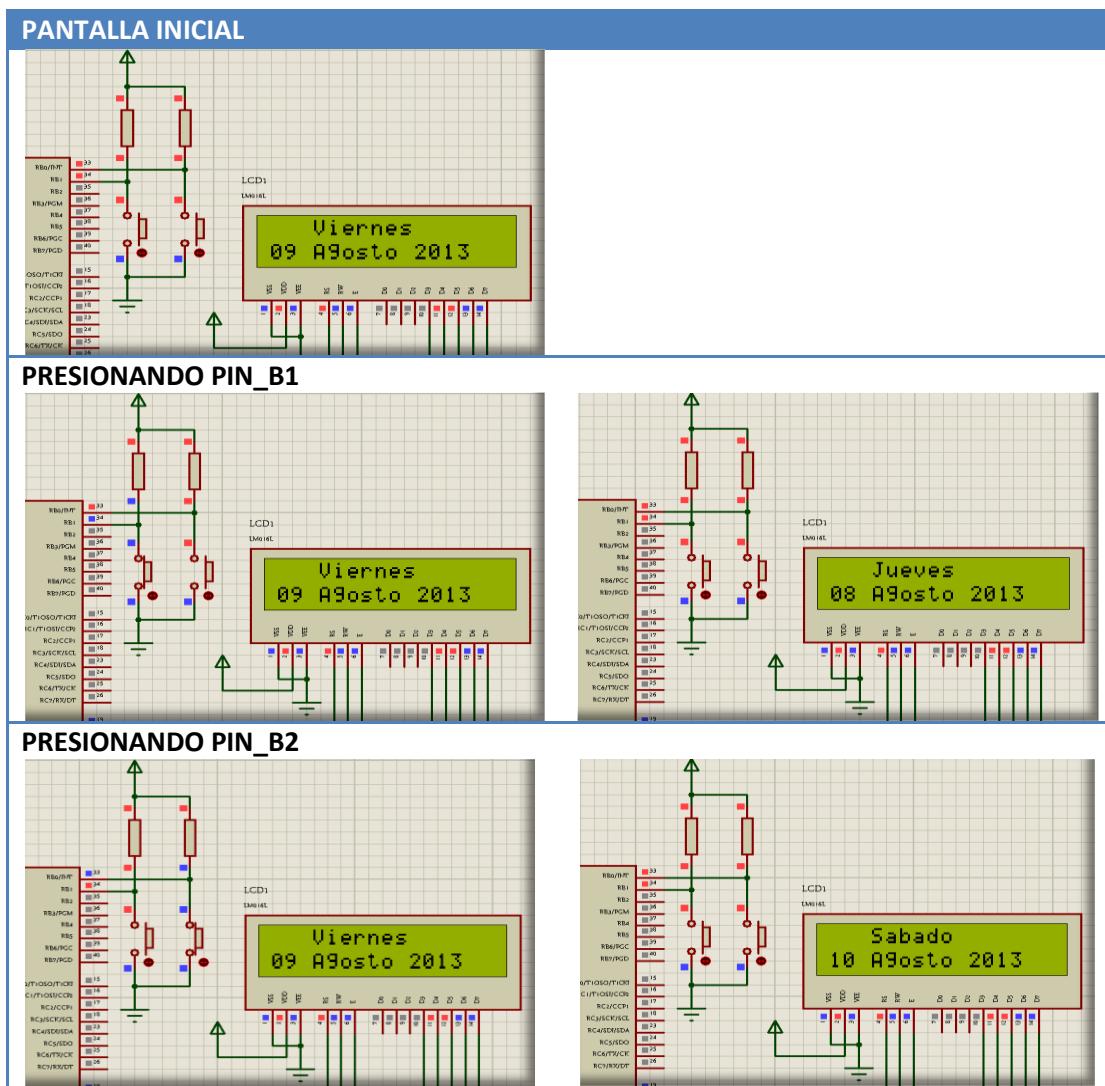
RELACIÓN DE ACTIVIDADES

ACTIVIDAD 19 - Imprimiendo la fecha.

Descripción:

Genere un programa que muestre en un LCD la fecha actual, un día anterior y un día después de la fecha de realización de la aplicación. Al comenzar el programa deberá mostrar la fecha actual, si es presionado un pulsador colocado en el PIN_B1 deberá mostrar la fecha de un día antes, de igual manera si es presionado un pulsador en el PIN_B0 deberá mostrar la fecha de un día posterior.

Resultado Esperado:



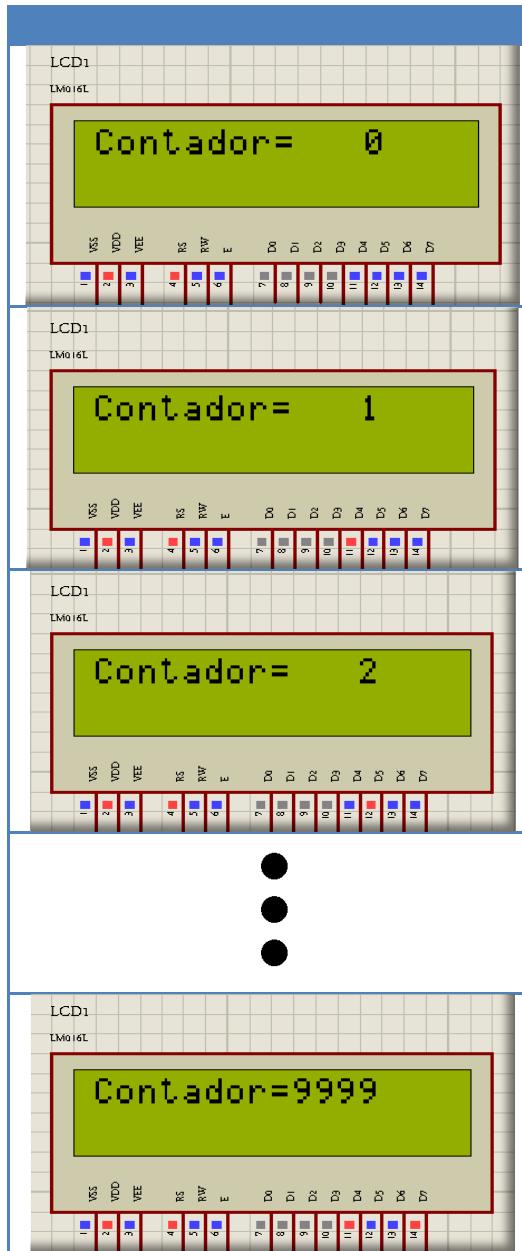


ACTIVIDAD 20 - Contador Industrial en LCD.

Descripción:

Genere un programa que muestre una cuenta de 0 a 9999 en un LCD. Cada cuenta deberá ser mostrada con un retardo entre una y otra de a lo más 100 mseg.

Resultado Esperado:

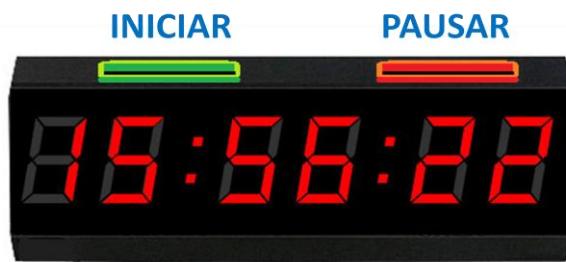




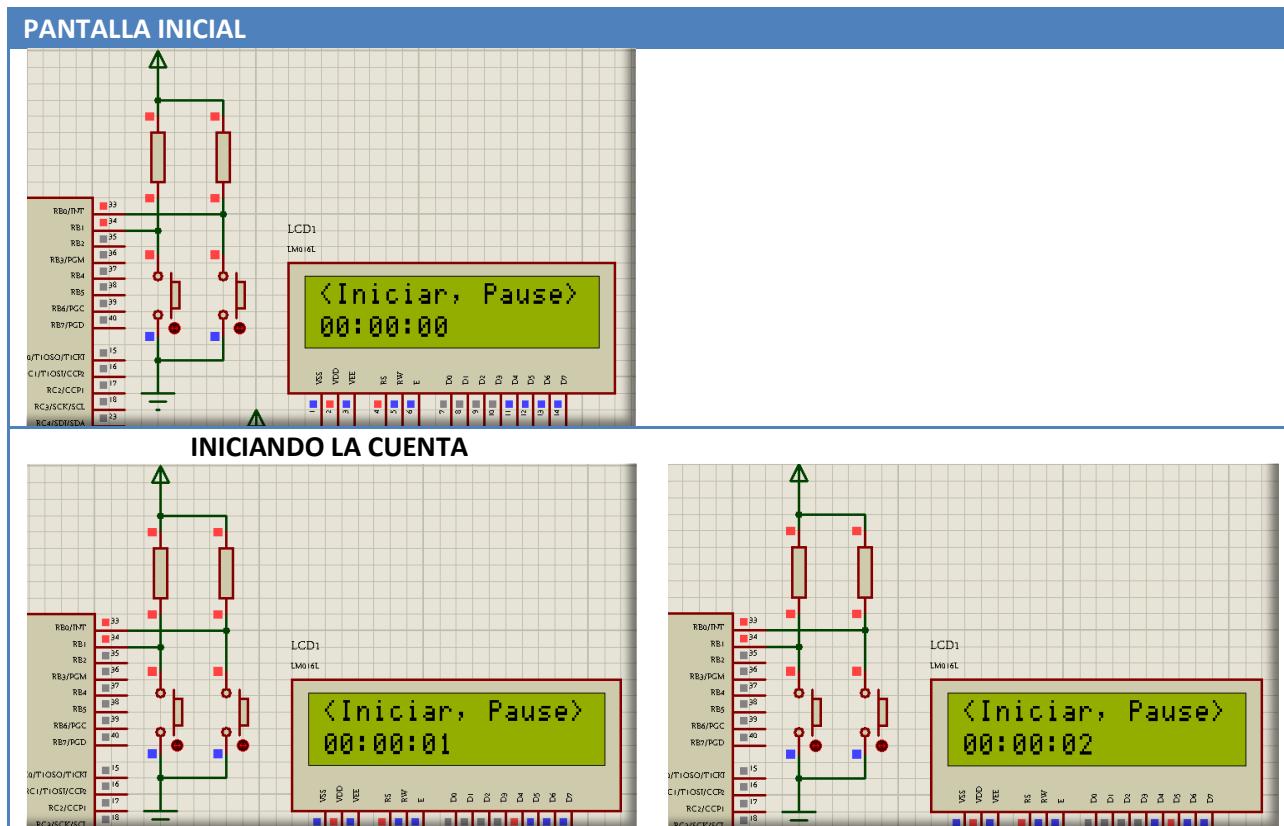
ACTIVIDAD 21 - Cronometro.

Descripción:

Genere un programa que muestre en un LCD la cuenta de un cronómetro. El cronómetro deberá mostrar la cuenta de los segundos, minutos y horas. Además contará con dos botones, uno que permitirá el iniciar la cuenta y otro que permitirá pausar o continuar la cuenta.



Resultado Esperado:



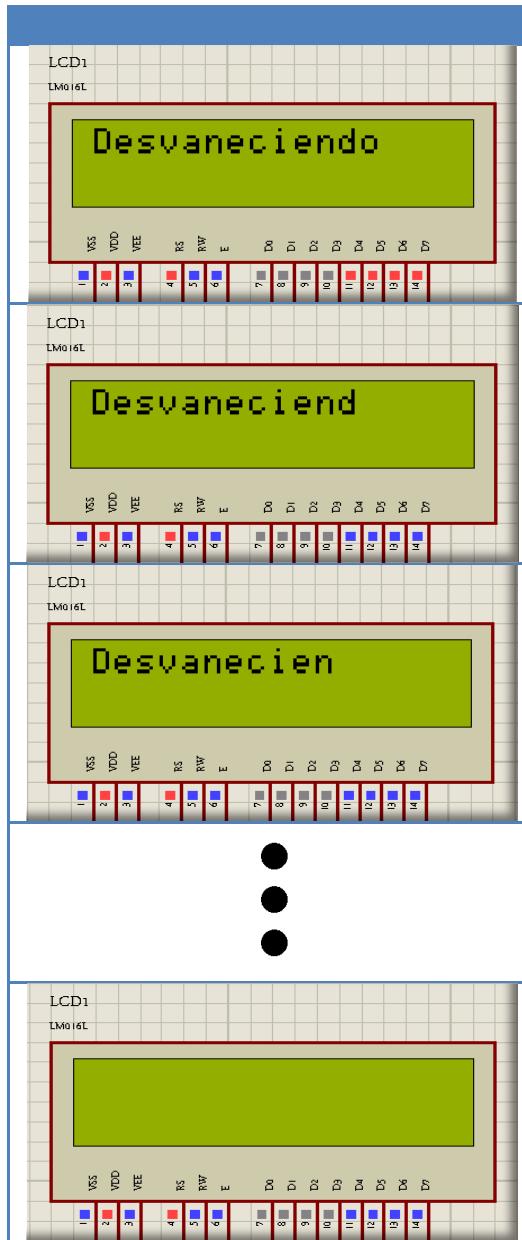


ACTIVIDAD 22 - Efecto Desvanecer.

Descripción:

Genere un programa que imprima un mensaje en un LCD. Posteriormente el letrero debe ir desapareciendo desde la última letra hasta la primera.

Resultado Esperado:





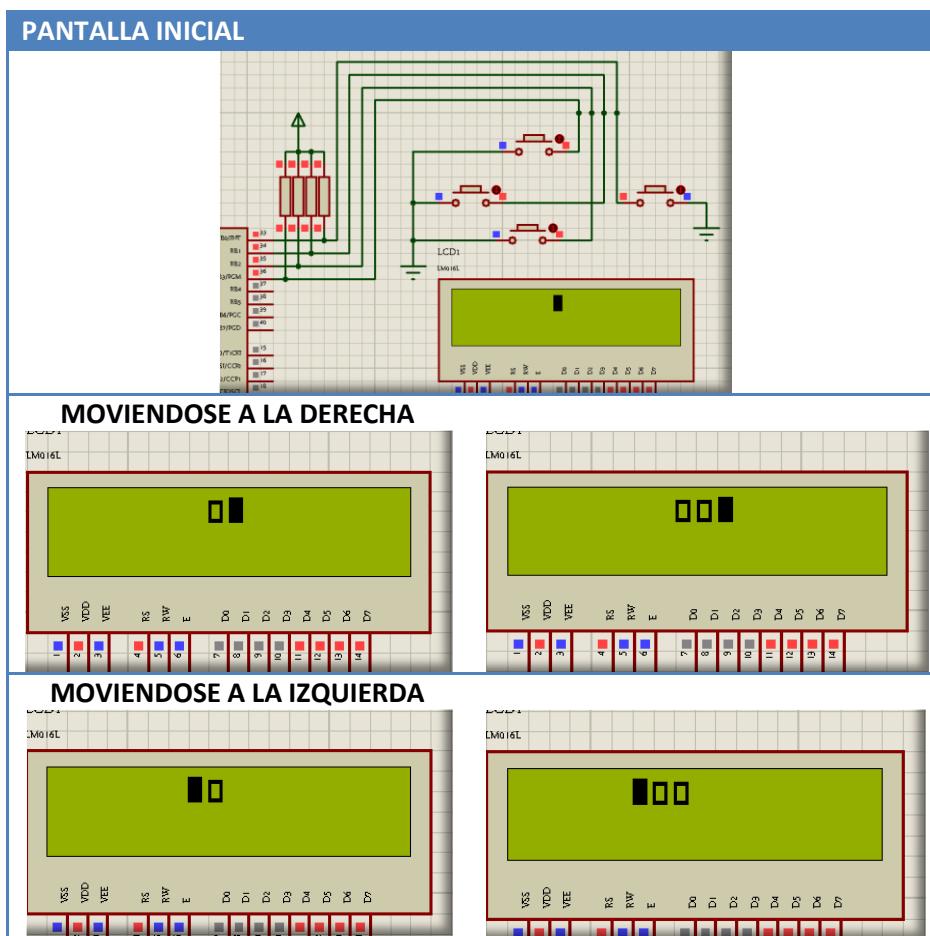
ACTIVIDAD 23 - Dejando huellas.

Descripción:

Genere un programa que muestre en un LCD el movimiento de un símbolo, este movimiento será de acuerdo al botón presionado por un joystick de 4 botones. De tal manera que si se presiona para que se mueva a la derecha, el símbolo se vaya moviendo y por cada lugar donde pasa dejará la huella de que paso por medio de otro símbolo. El símbolo que se mueve deberá tener la posibilidad de moverse por todas las posiciones del LCD.



Resultado Esperado:

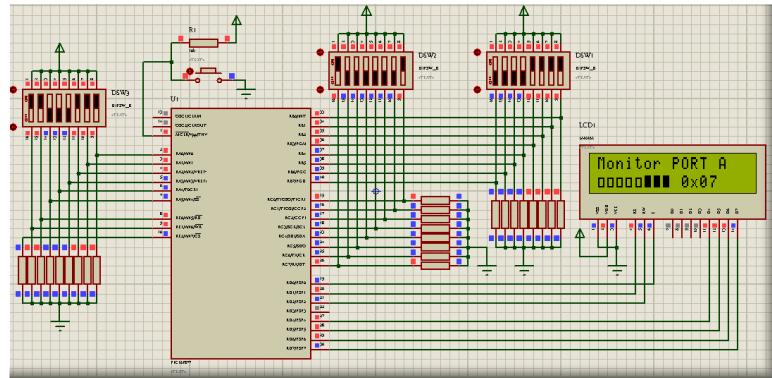




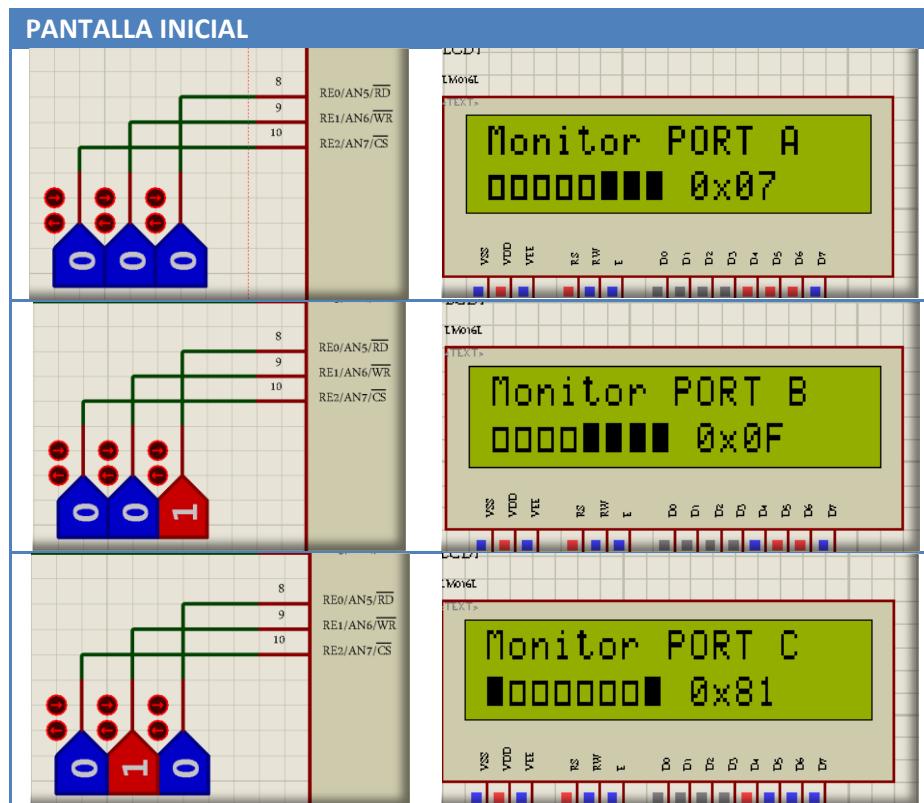
ACTIVIDAD 24 - Monitor de Puertos Digitales.

Descripción:

Genere un programa que muestre en un LCD el contenido de los puertos A, B y C, de tal forma que muestre tanto el valor hexadecimal del valor de la entrada como una representación binaria por medio del LCD. El usuario podrá elegir el puerto a monitorear por medio de tres entradas digitales colocadas en el puerto E y por medio de la combinación asignada para visualizar cada puerto.



Resultado Esperado:

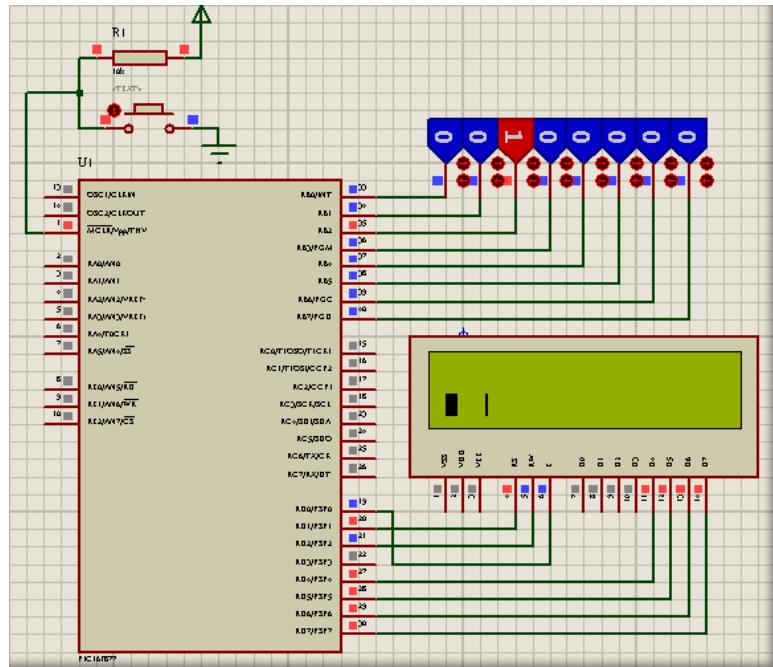




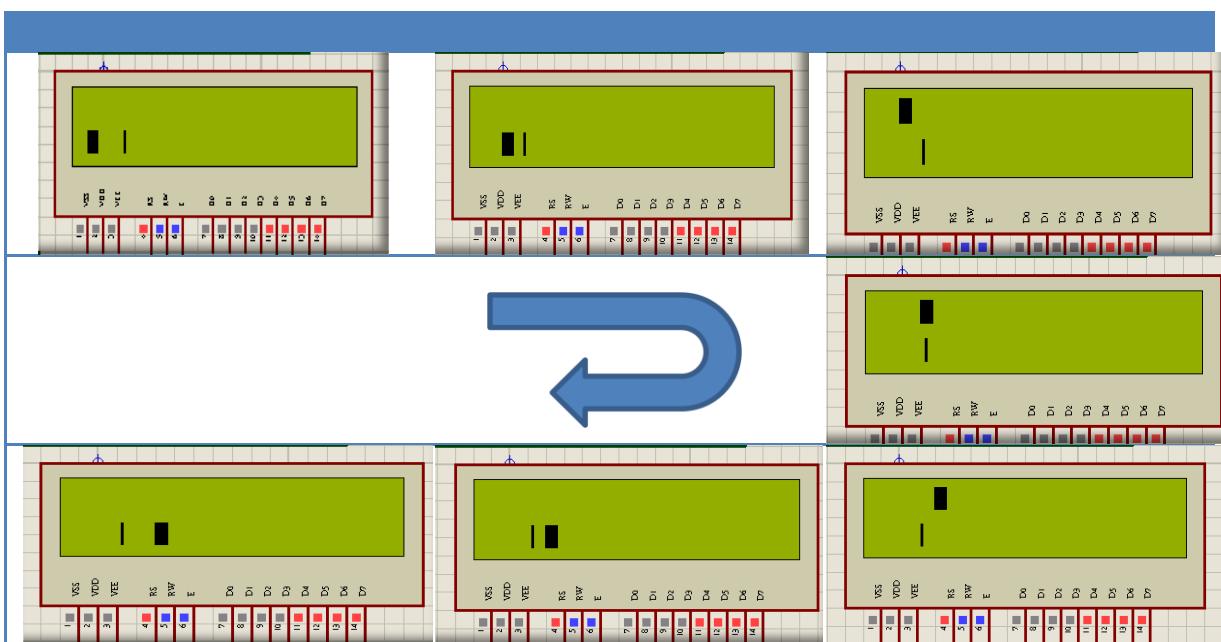
ACTIVIDAD 25 - Brincando la cerca II.

Descripción:

Genere un programa que realice el mismo efecto que la actividad 11, mostrando el recorrido y el salto de la cerca por medio de su graficación en el LCD.



Resultado Esperado:

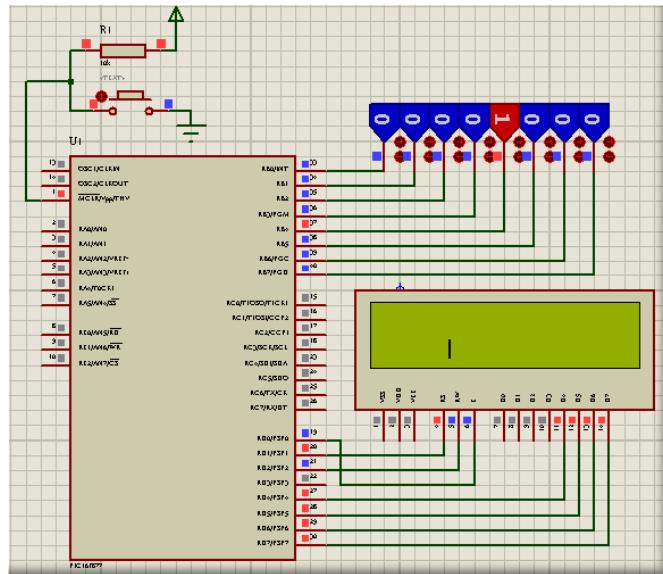




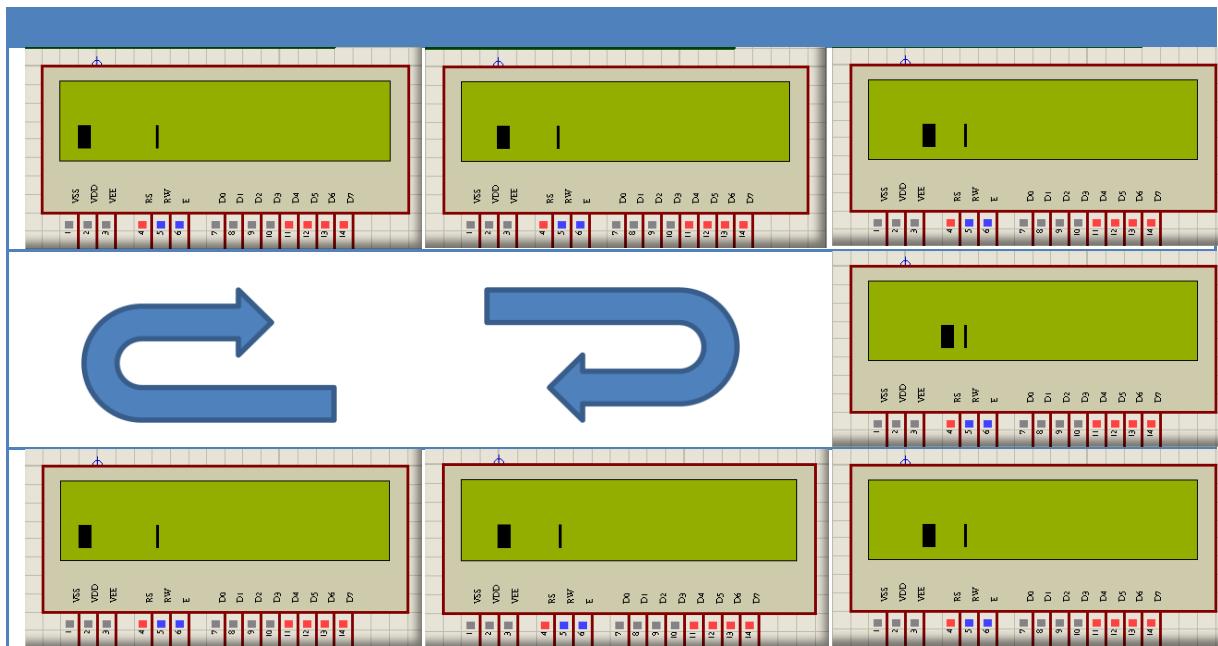
ACTIVIDAD 26 - Rebotando la pelota II.

Descripción:

Genere un programa que realice el mismo efecto que la actividad 1, mostrando el recorrido y el rebote de la pelota por medio de su graficación en el LCD.



Resultado Esperado:

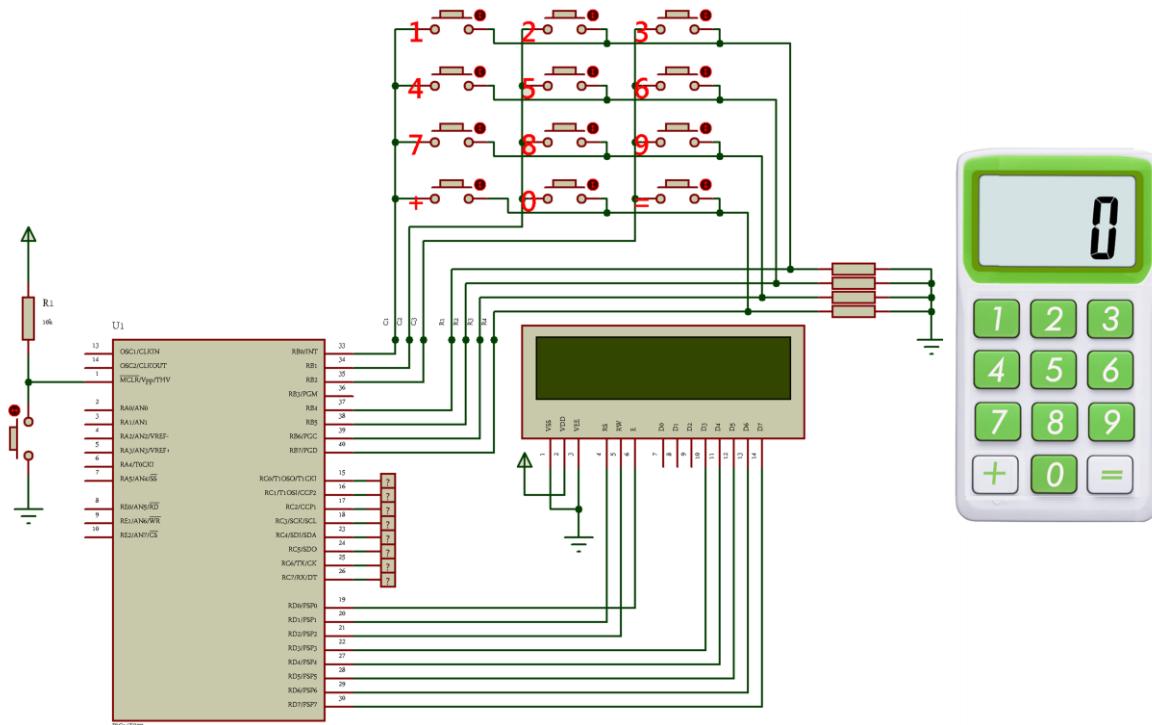




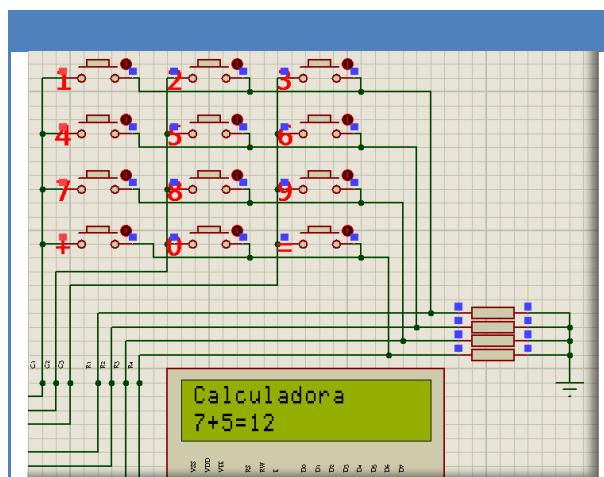
ACTIVIDAD 27 - Calculadora básica.

Descripción:

Genere un programa que realice la operación suma de una calculadora, tal que en un LCD se muestre cuando se oprime un número, la operación suma, la igualdad y el resultado de la operación. Los datos deberán ser introducidos por medio del teclado matricial.



Resultado Esperado:

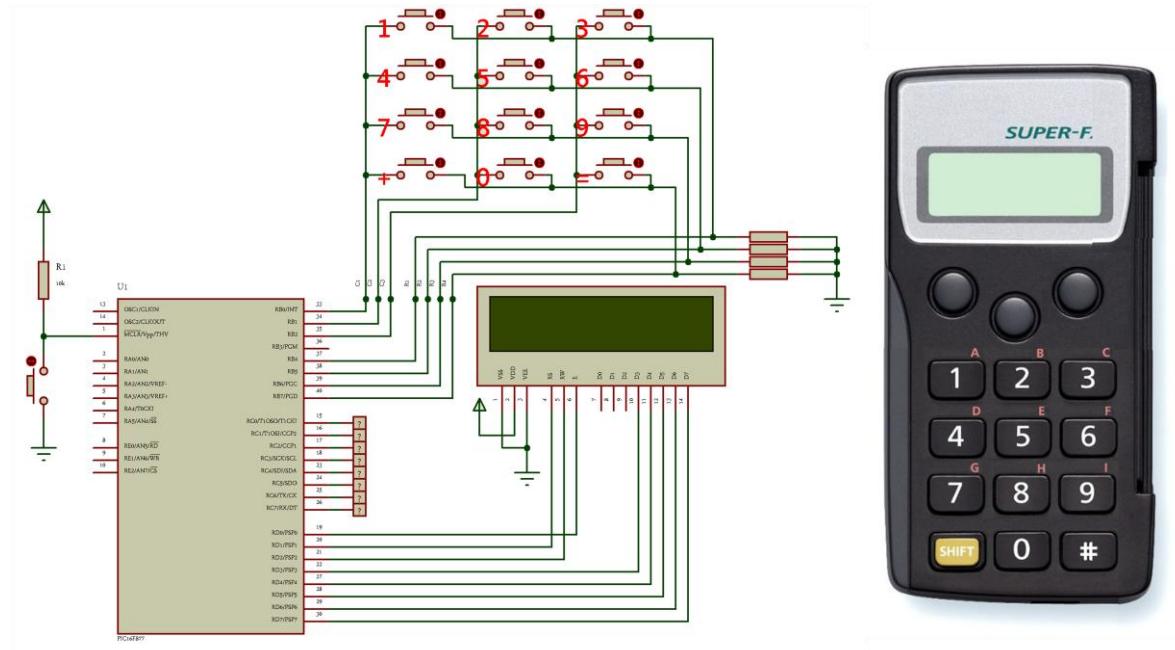




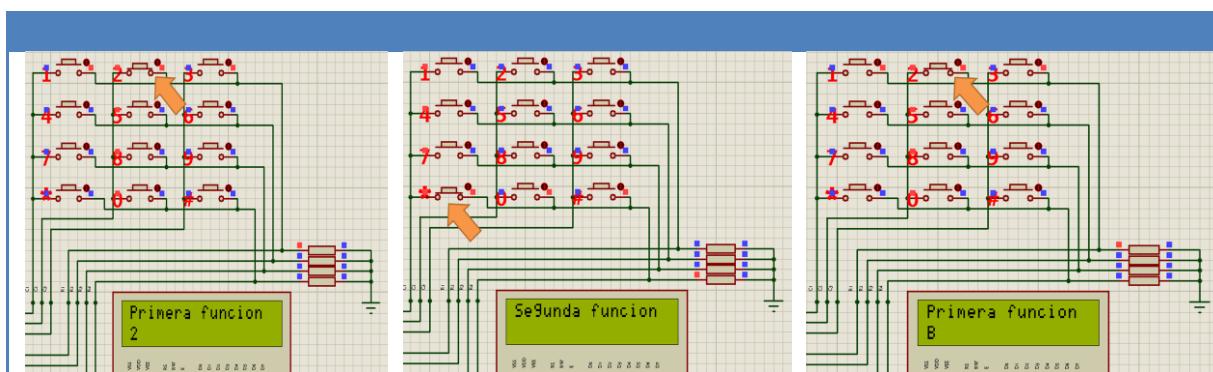
ACTIVIDAD 28 - Segundas funciones.

Descripción:

Genere un programa que realice la operación SHIFT similar a la de una calculadora, de tal manera que si solo se presiona una tecla, imprima el número correspondiente a la tecla oprimida, pero si es presionada la tecla SHIFT y después una tecla numérica, se deberá imprimir el carácter mostrado en la parte superior de esa tecla.



Resultado Esperado:

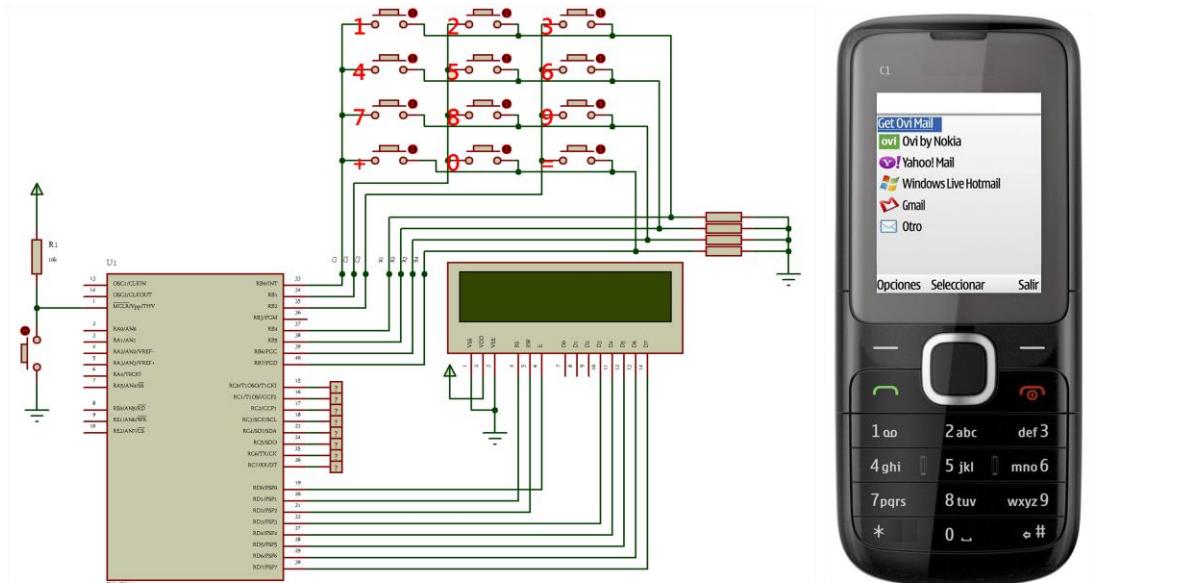




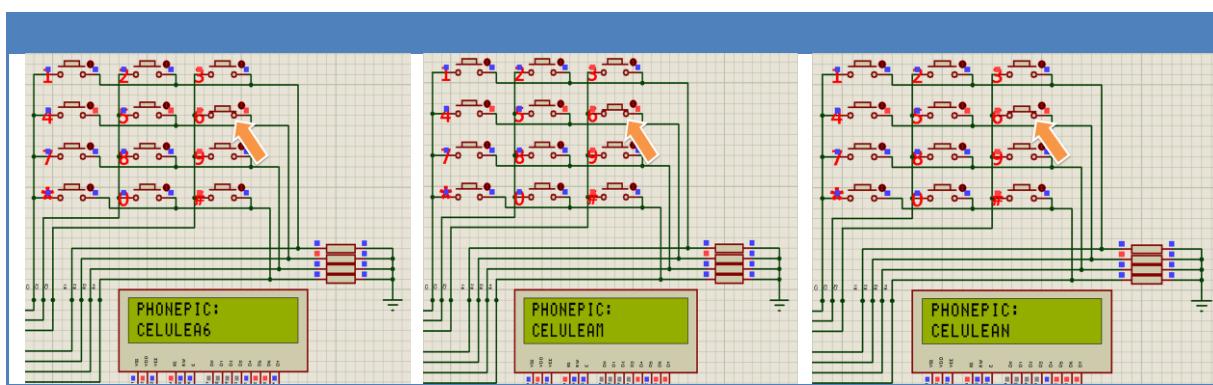
ACTIVIDAD 29 - Tecleando en el celular.

Descripción:

Genere un programa que realice la operación de un teclado de celular estándar, de tal manera que al presionar varias veces una misma tecla aparezcan los caracteres *abcd*, permitiendo escribir mensajes en la pantalla del LCD por medio del teclado matricial.



Resultado Esperado:

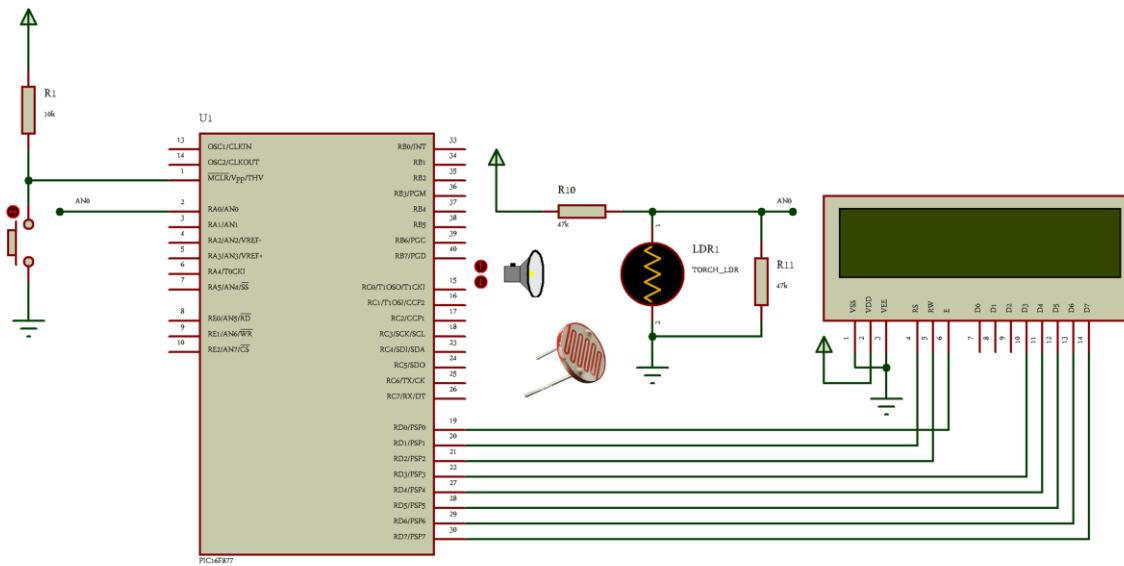




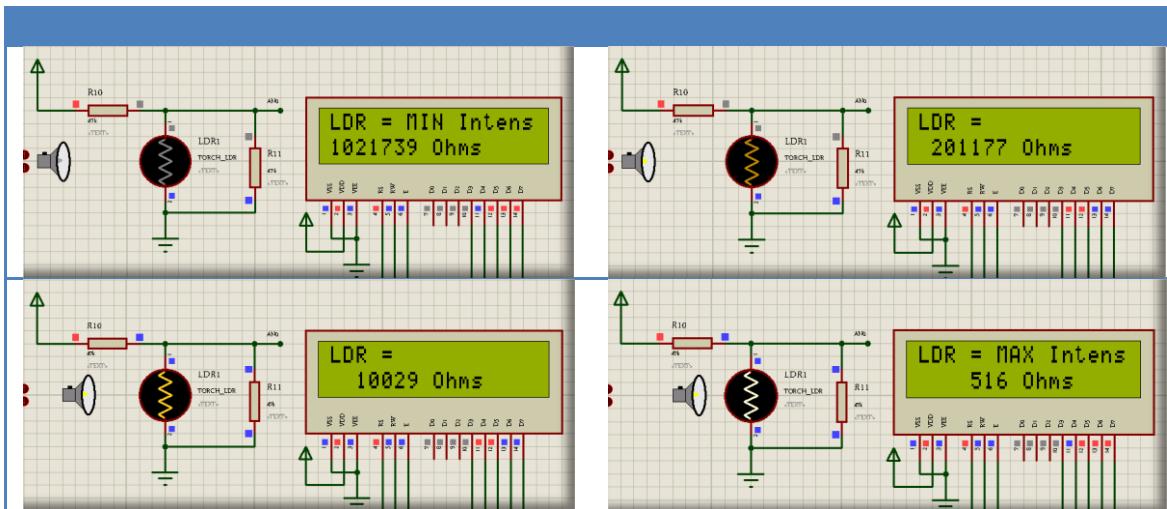
ACTIVIDAD 30 - Sensor de luminosidad.

Descripción:

Genere un programa que sea capaz de medir la resistencia de una LDR a diferentes niveles de intensidad lumínica por medio del convertidor A/D, tal que muestre su medición en un LCD.



Resultado Esperado:

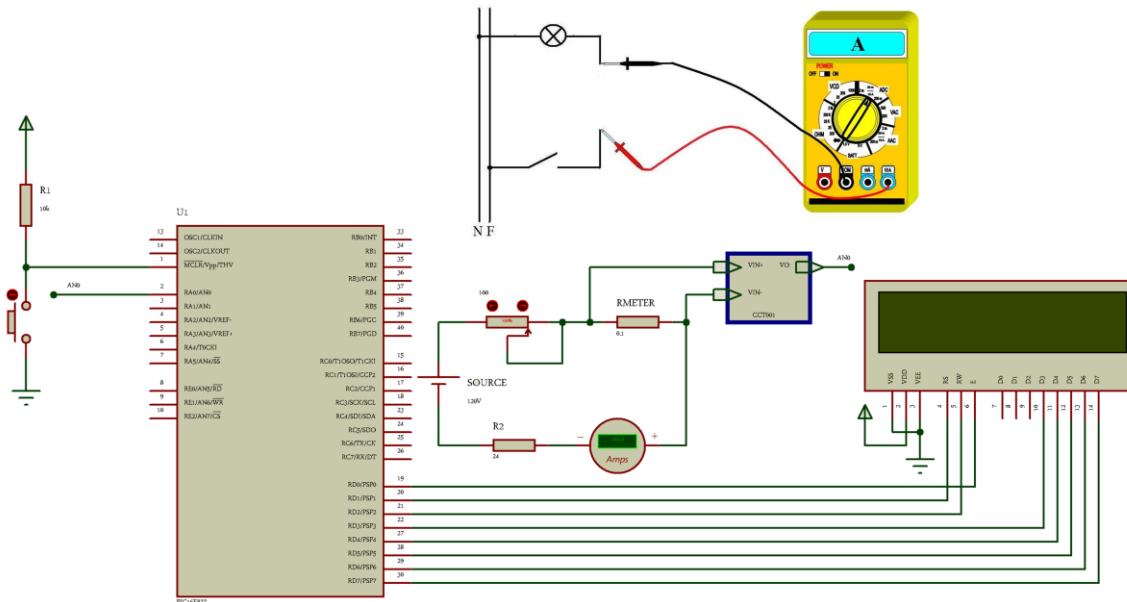




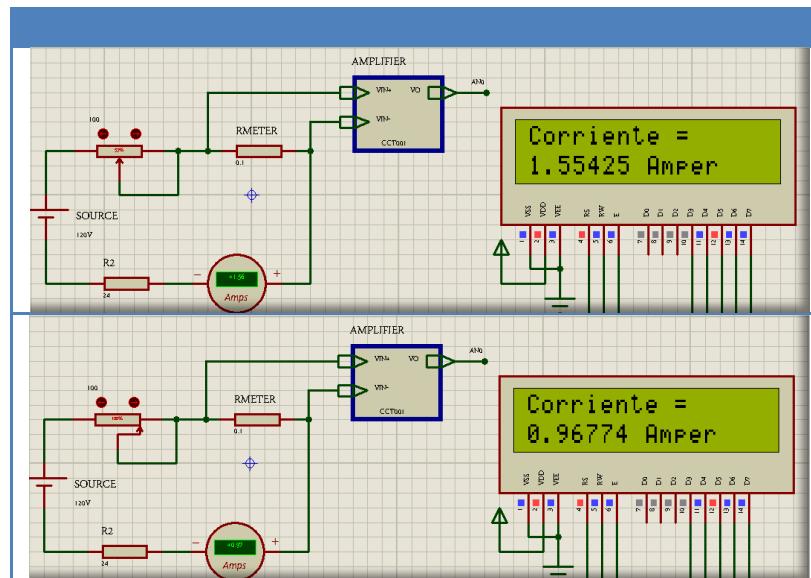
ACTIVIDAD 31 - Sensor de corriente eléctrica.

Descripción:

Genere un programa que sea capaz de medir la corriente de un circuito eléctrico por medio del convertidor A/D, tal que muestre su medición en un LCD.



Resultado Esperado:

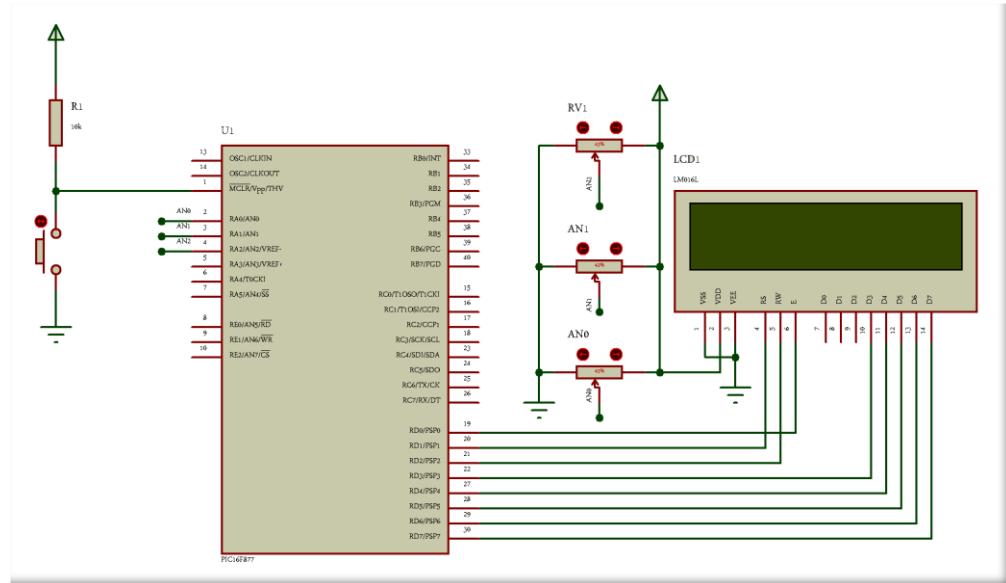




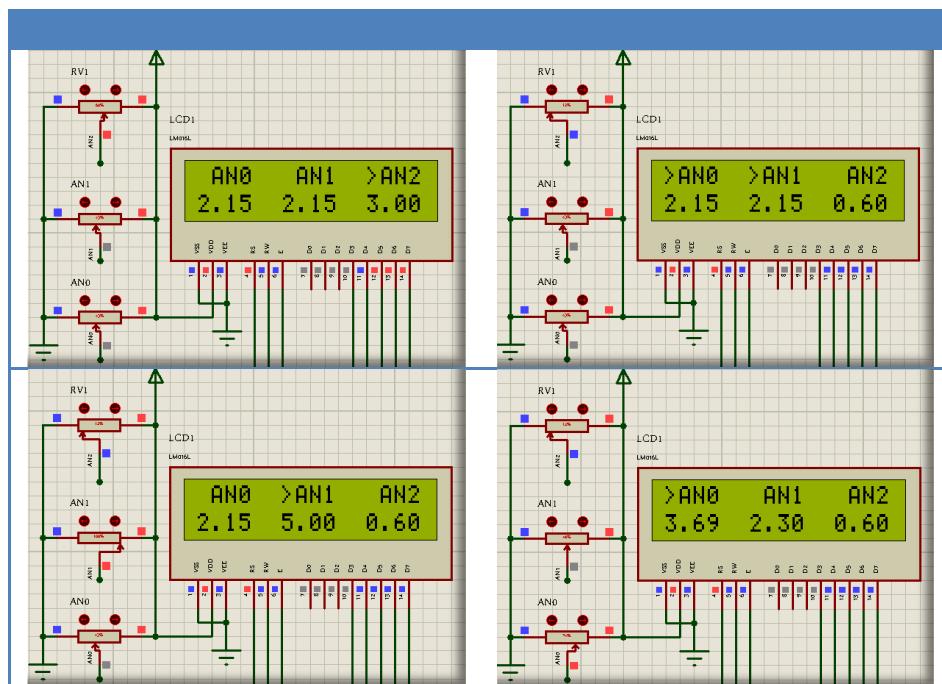
ACTIVIDAD 32 - El canal mayor.

Descripción:

Genere un programa que sea capaz de hacer la medición de 3 señales analógicas y sean mostradas en la pantalla LCD, tal que apunte cual de las señales tiene el voltaje mayor.



Resultado Esperado:

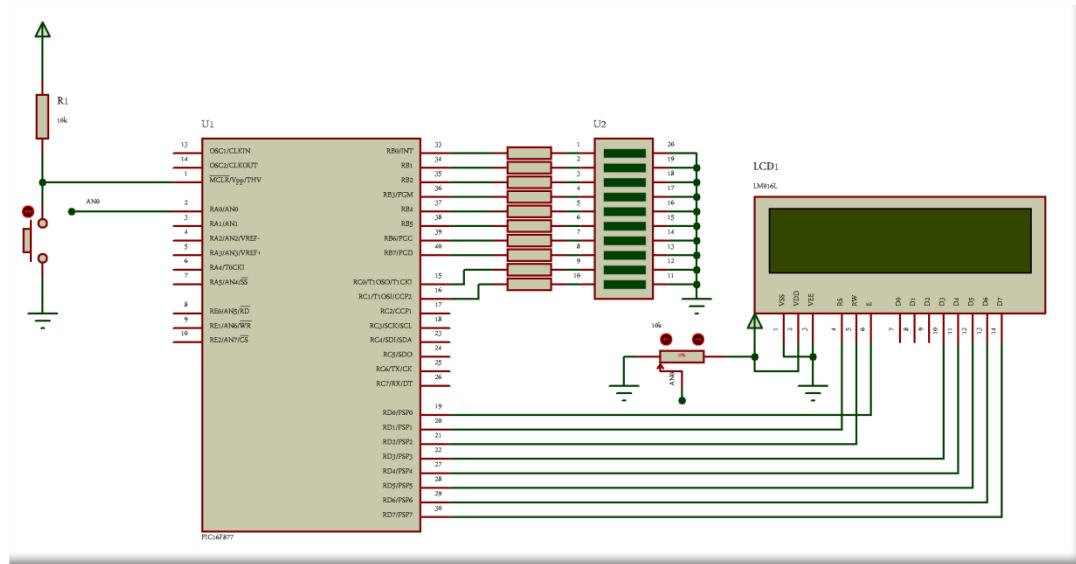




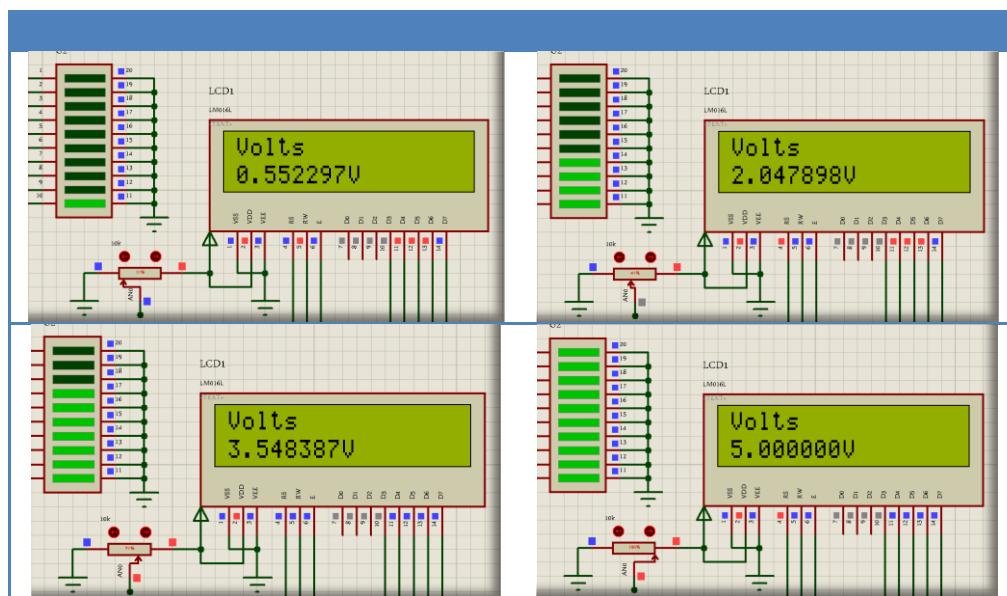
ACTIVIDAD 33 - Voltímetro a Leds.

Descripción:

Genere un programa que muestre la medición de una señal analógica en un LCD y además muestre el nivel de la señal en una barra de leds, tal que cada nivel este cambiando a una razón 0.5 voltos.



Resultado Esperado:

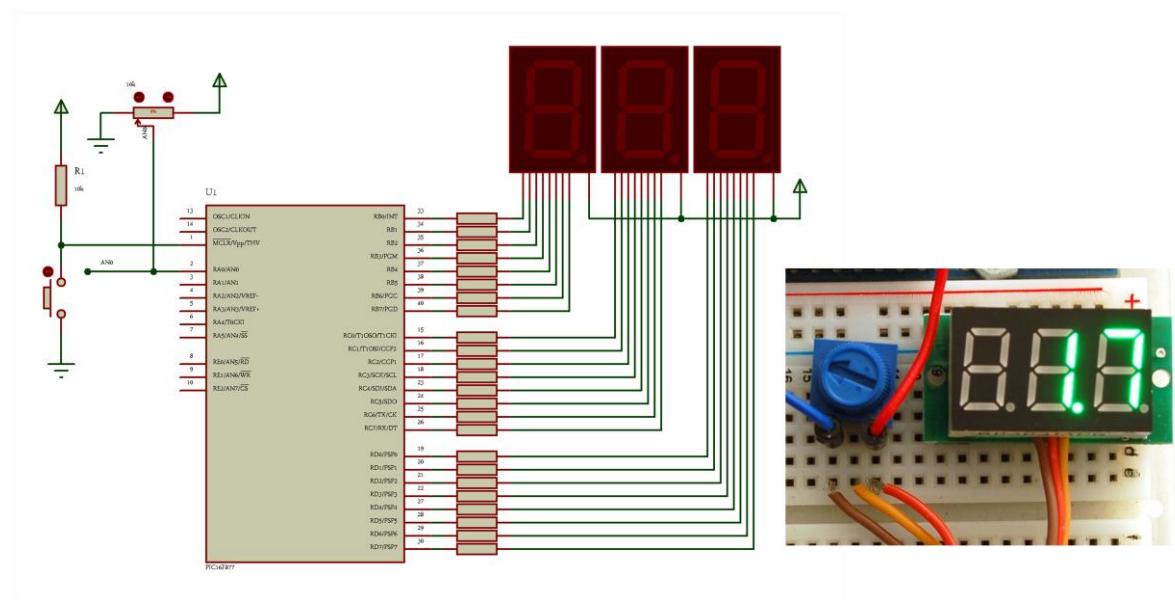




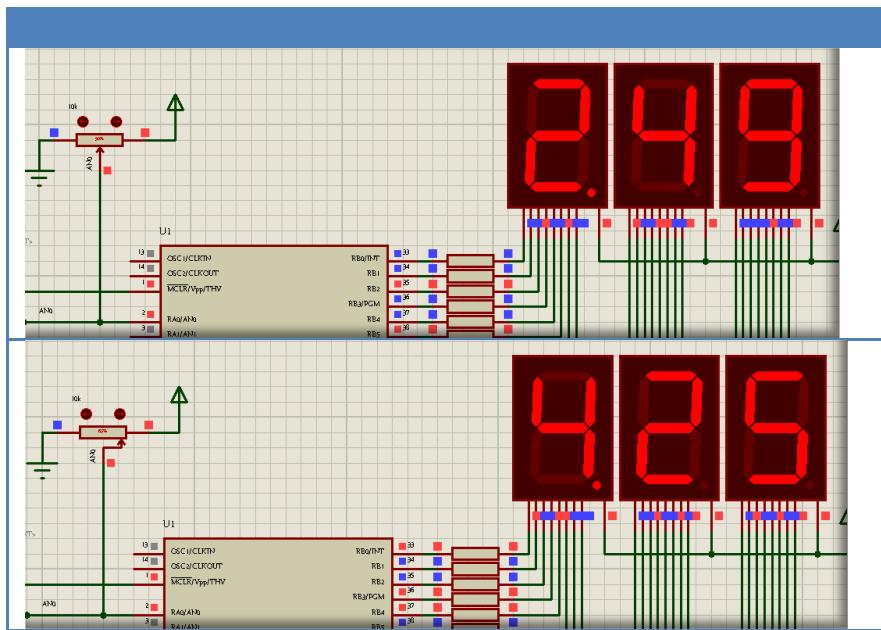
ACTIVIDAD 34 - Voltímetro en Displays

Descripción:

Genere un programa que sea capaz de medir una señal analógica y por medio de 3 displays de 7 segmentos muestre la medición en voltaje realizada.



Resultado Esperado:

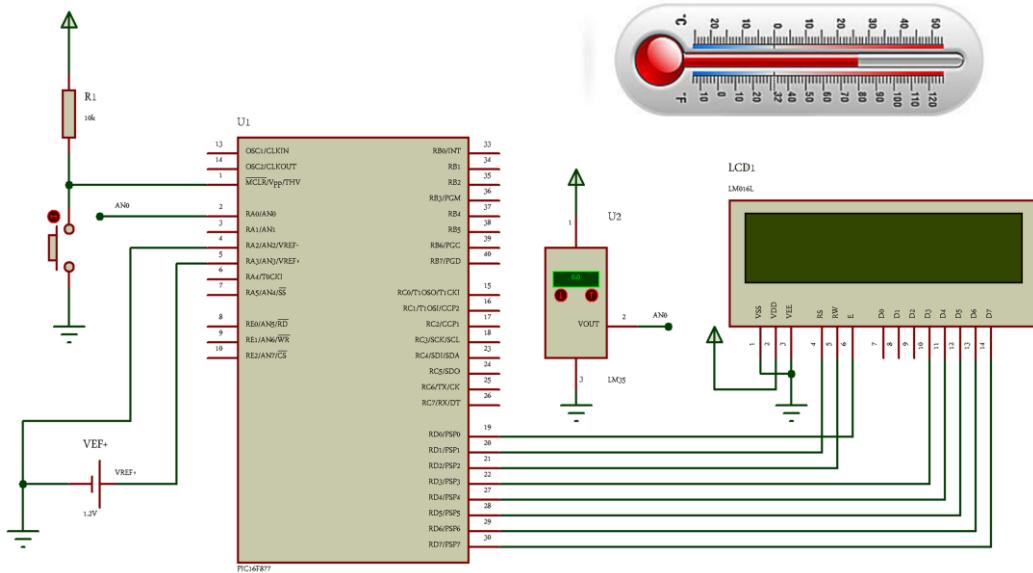




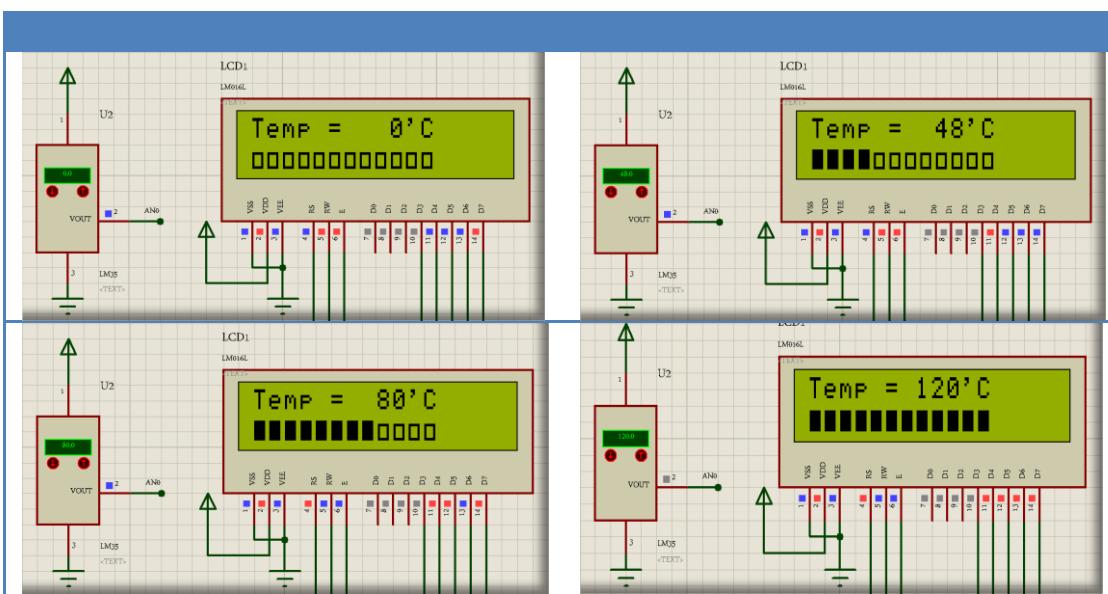
ACTIVIDAD 35 - El termómetro gráfico.

Descripción:

Genere un programa que sea capaz de medir la temperatura de un sensor, de tal manera que la temperatura medida se muestre numéricamente y por medio de una barra de nivel horizontal en la pantalla de un LCD.



Resultado Esperado:

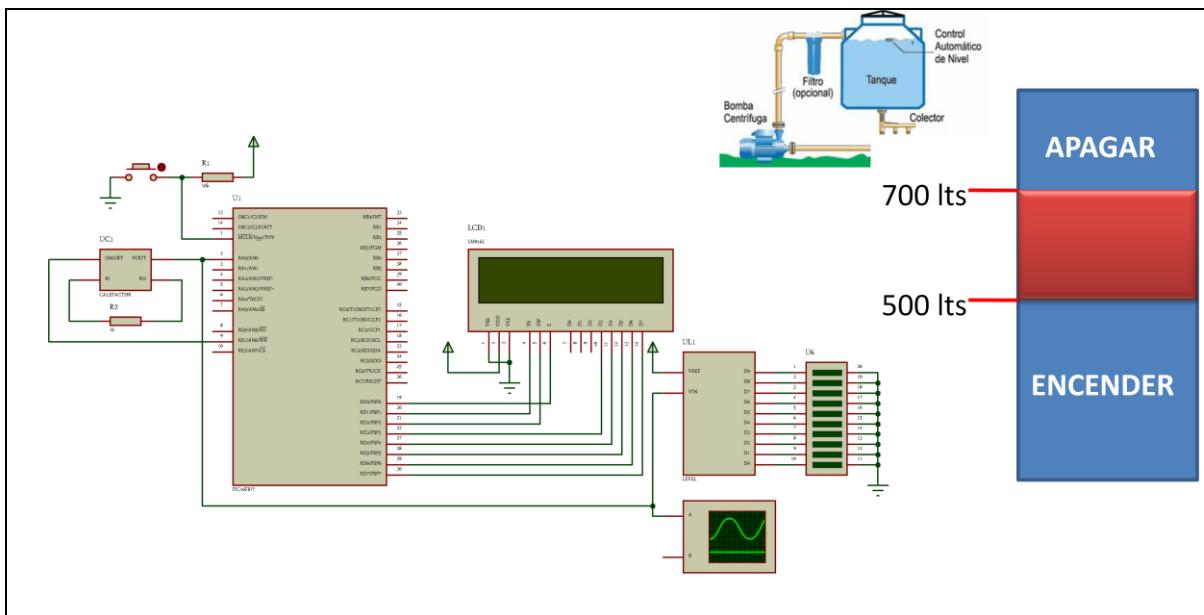




ACTIVIDAD 36 - Llenando el tinaco.

Descripción:

Genere un programa que sea capaz de controlar el nivel de agua de un tanque de 1000 lts entre dos niveles, tal que si el nivel del agua está por debajo de los 500 lts, la bomba de agua se enciende y si el nivel del agua está por arriba de los 700 lts, la bomba deberá apagarse.

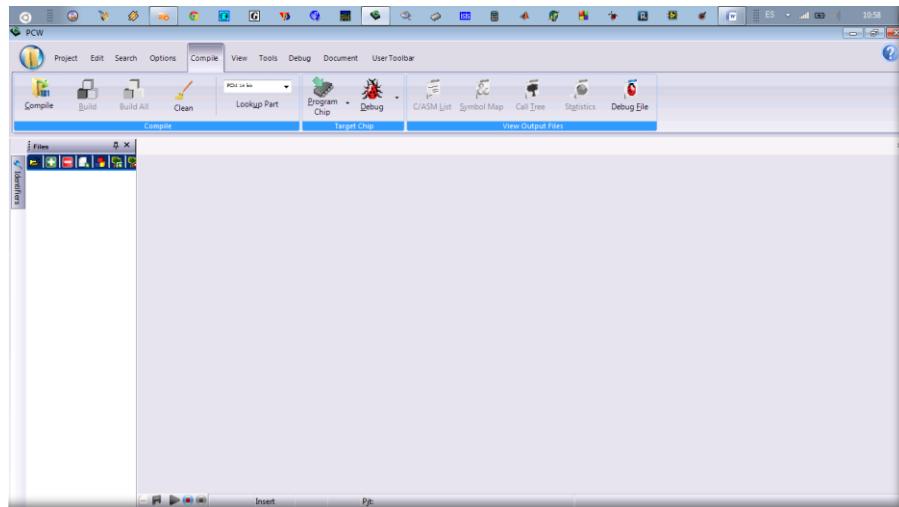




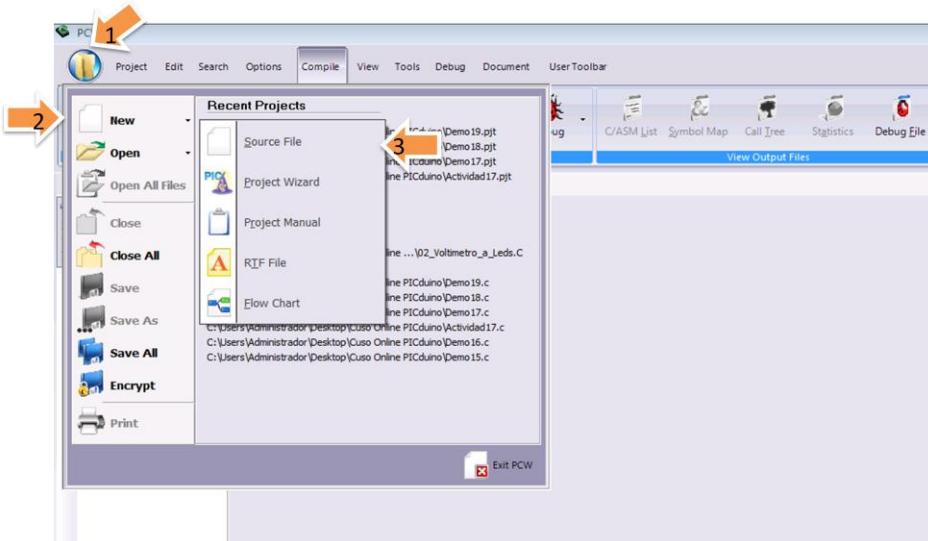
Apéndice – A

Compilando un primer programa en Pic C Compiler.

El entorno de desarrollo de Pic C Compiler posee varias herramientas como edición, búsqueda y remplazo de texto, generación de diagramas de flujo, compilación inclusive de programación para nuestro microcontrolador. En nuestro caso, la principal herramienta que usaremos será la pestaña de Compilación.



Para generar un nuevo programa iremos al menú File (1), luego New (2) y finalmente Source File (3) con el fin de generar un archivo de texto donde almacenaremos nuestro código del programa.

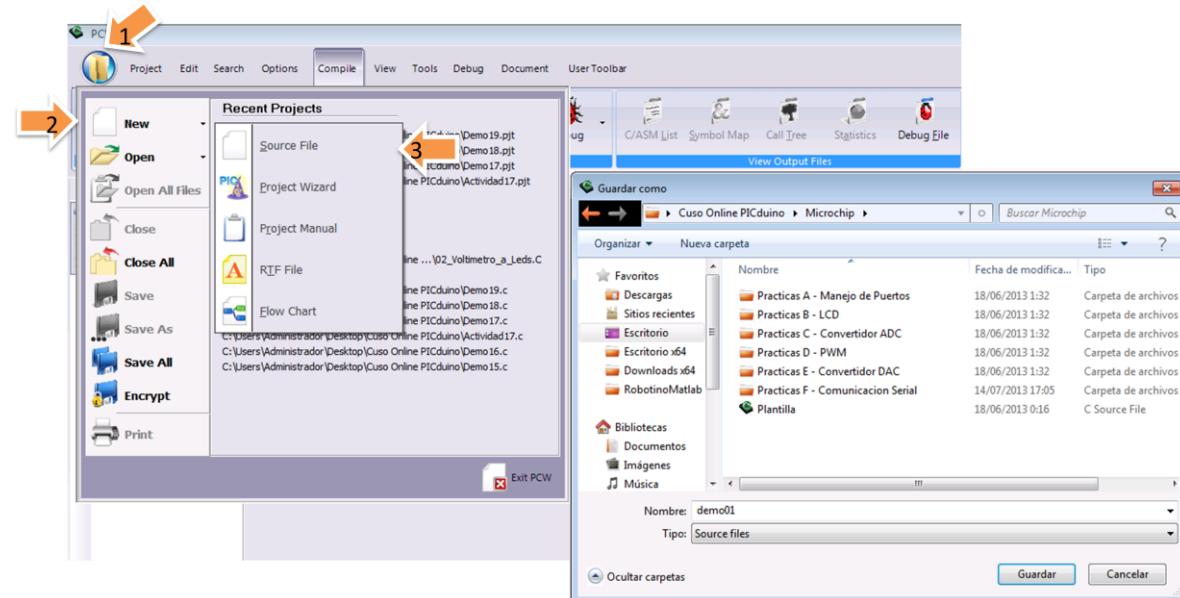


Manual – Aplicaciones Prácticas con Microcontroladores PIC

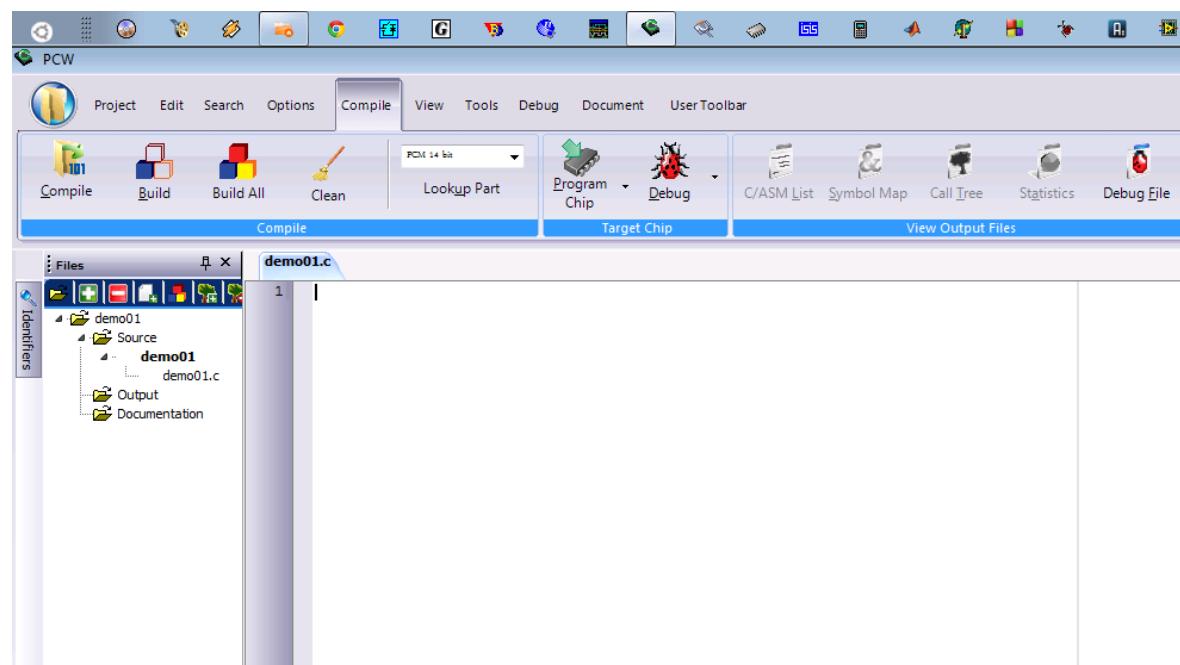
Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Una vez hecho esto nos aparecerá una ventana de dialogo donde nos pedirá una ubicación y un nombre para guardar nuestro proyecto.



Una vez guardado nuestro archivo nos aparecerá un archivo de texto en blanco, donde escribiremos nuestros programas.



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Para la compilación de nuestro primer programa transcribiremos el código del Demo 01, y le daremos al botón “Build All” para compilar nuestro código.

The screenshot shows the PCW (PIC C Compiler) interface. The toolbar at the top has buttons for Project, Edit, Search, Compile, View, Tools, Debug, Document, and User Toolbar. The 'Compile' button is highlighted with an orange arrow. Below the toolbar is a menu bar with 'File', 'Project', 'Edit', 'Search', 'Compile', 'View', 'Tools', 'Debug', 'Document', and 'User Toolbar'. The 'Compile' menu is open, showing 'Build All' as the selected option. The main window displays a code editor with the following C code:

```
877.h>
NDT,NOPROTECT,PUT
CLOCK=20000000 )

// Definiciones Globales
#define prende_pin(x) output_high(x)
#define apaga_pin(x) output_low(x)

// Definiciones de Variables Globales
// Declaración de Subrutinas o Métodos
// Programa Principal
void main ()
{
    // Definiciones de Variables Locales
    // Configuración de Puertos
    // Prueba de Definiciones globales
    prende_pin(PIN_D0);

    // Bucle Principal
    while (1){
        // Instrucciones del programa
        .
        } // end while
    } //end main
}
```

The screenshot shows the PCW interface with the 'Build All' button highlighted. A CCS C Compiler dialog box is overlaid on the main window. The dialog box title is 'CCS C Compiler' and 'PCM Compiler V4.104'. It displays the following information:

Project: C:\...ktop\Curso Online PICduino\Microchip\demo01.c

Complete
No errors
Files: 2, Statements: 3, Time: 1 Sec, Lines: 330
Output files: ERR HEX SYM LST COF PJT TRE STA

RAM: 1%
ROM: 0%

www.ccslinfo.com



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



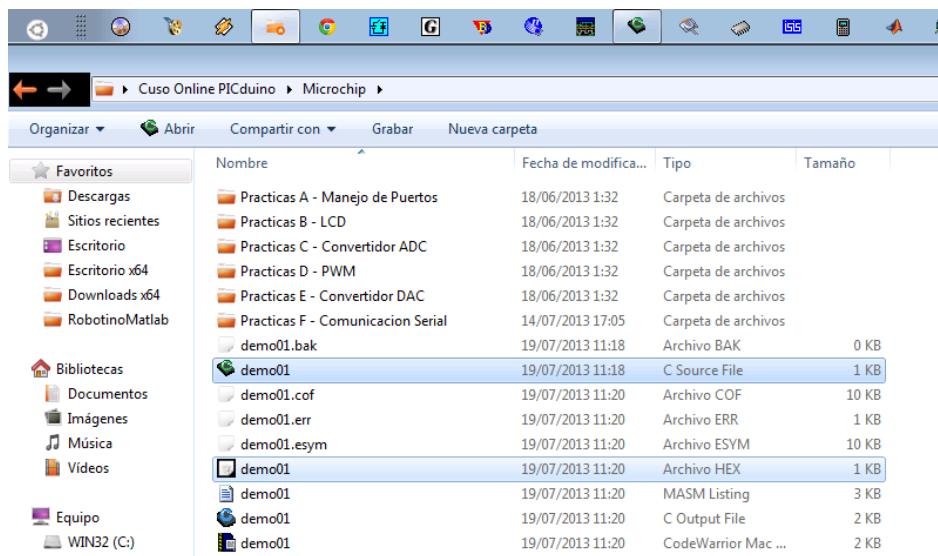
Si no hay errores en nuestro programa, nos aparecerá un mensaje similar al siguiente, donde marca cero errores.

The screenshot shows a software interface with a tab labeled "Output". Inside the window, the following text is displayed:

```
>>> Warning 203 "demo01.c" Line 23(1,1): Condition always TRUE
Memory usage: ROM=0% RAM=1% - 1%
0 Errors, 1 Warnings.
```

Below the window are two buttons: "Compiler" and "Find".

Al compilar nuestro programa se generan varios archivos en el proceso en el lugar donde guardamos nuestro código fuente. De estos, solo serán necesarios dos un archivo .C que es nuestro código fuente y un .hex que son los códigos que serán almacenados en nuestro microcontrolador para realizar nuestra aplicación.



Una vez hecho esto, estamos listos para simular nuestro microcontrolador en Proteus ISIS.



Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Ahora supongamos que por algún motivo nos equivocamos al escribir nuestro programa, por ejemplo el olvido de un punto y coma.

The screenshot shows the PCW (PIC C Workstation) software interface. The main window displays a C program named 'demo01.c'. The code includes comments and a main function. On line 20, there is a syntax error: 'prende_pin[PIN_D0];' where a semicolon is missing after 'PIN_D0'. The word 'PIN_D0' is highlighted in yellow, indicating it is an undeclared identifier.

```
demo01.c
9 // Declaración de Subrutinas o Métodos
10 // Programa Principal
11 void main ()
12 {
13     // Definiciones de Variables Locales
14
15     // Configuración de Puertos
16
17     // Prueba de Definiciones globales
18     prende_pin[PIN_D0];
19
20     // Bucle Principal
21     while (1){
22         // Instrucciones del programa
23
24     } // end while
25
26
27 }
```

En el programa eliminaremos un punto y coma a propósito generando errores para ver qué resultados da la compilación.

This screenshot shows the same PCW software interface as the previous one, but the error has been corrected. The line 'prende_pin[PIN_D0];' now includes a semicolon at the end of the assignment statement. The code runs without errors.

```
demo01.c*
9 // Declaración de Subrutinas o Métodos
10 // Programa Principal
11 void main ()
12 {
13     // Definiciones de Variables Locales
14
15     // Configuración de Puertos
16
17     // Prueba de Definiciones globales
18     prende_pin[PIN_D0];
19
20     // Bucle Principal
21     while (1){
22         // Instrucciones del programa
23
24     } // end while
25
26
27 }
```





Al compilar el programa con este error el software nos genera un mensaje “Error 76 “demo01.c” Line 23(1, 6): Expect ;” el cual nos indica un posible error, en este caso nos dice que esperaba un punto y coma (Expect ;), sin embargo el software no nos envía directamente al lugar donde está este error, sino que generalmente nos ubica una instrucción antes o una instrucción después, por lo que tendremos que verificar en ocasiones antes y después de donde nos señale.

A screenshot of the MPLAB X Integrated Development Environment. The code editor shows a C program with syntax highlighting. Line 23 contains the error: "while (1){". The output window at the bottom shows the error message: "Error 76 \"demo01.c\" Line 23(1,6): Expect ;".

```
14 // Definiciones de Variables Locales
15
16 // Configuración de Puertos
17
18 // Prueba de Definiciones globales
19 prende_pin(PIN_D0)
20
21 // Bucle Principal
22 while (1){
23     // Instrucciones del programa
24
25         } // end while
26
27 } //end main
28
29
30
```

Muchos de los posibles errores se generan por no escribir correctamente las instrucciones, otras muchas también por punto y comas omitidos, por lo que tendremos que cuidar estos detalles para la correcta compilación de nuestro programa.

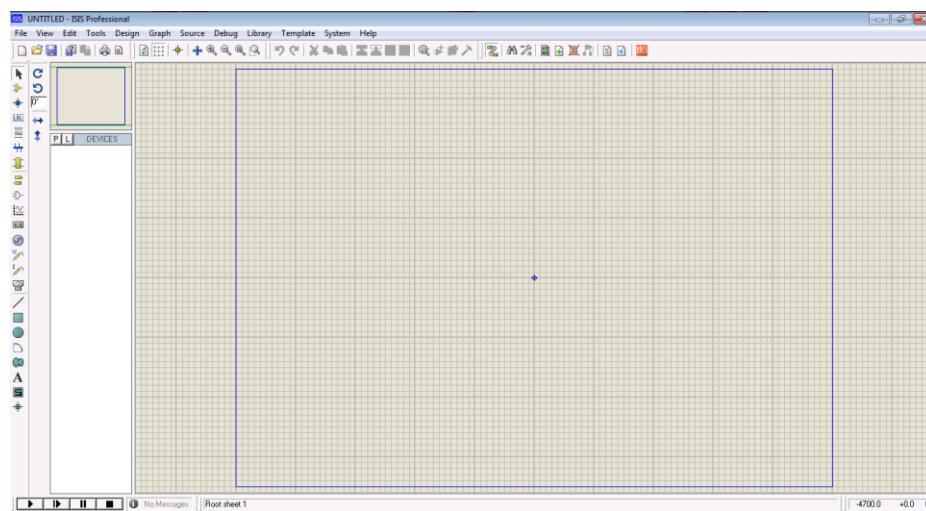




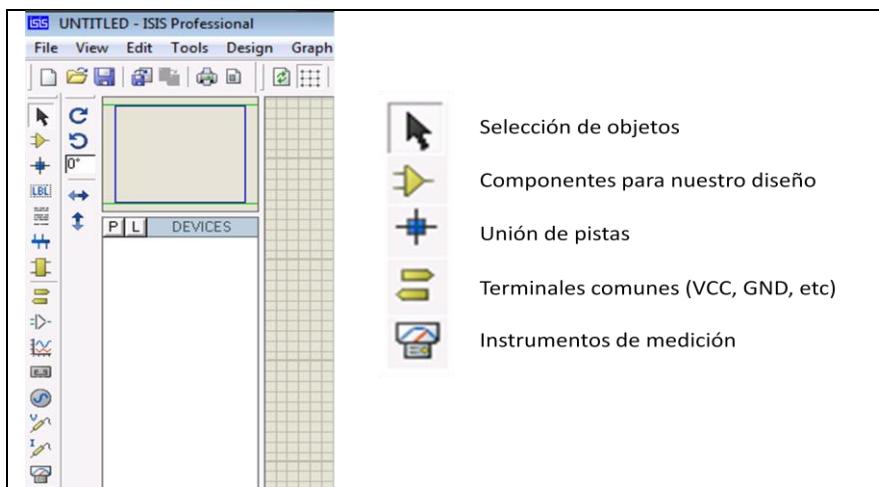
Apéndice – B

Armando un primer circuito en Proteus ISIS.

Proteus ISIS es una herramienta muy completa en la simulación de Microcontroladores, inclusive se pueden hacer prácticas donde en este se muestren valores reales que están siendo introducidos a la computadora.

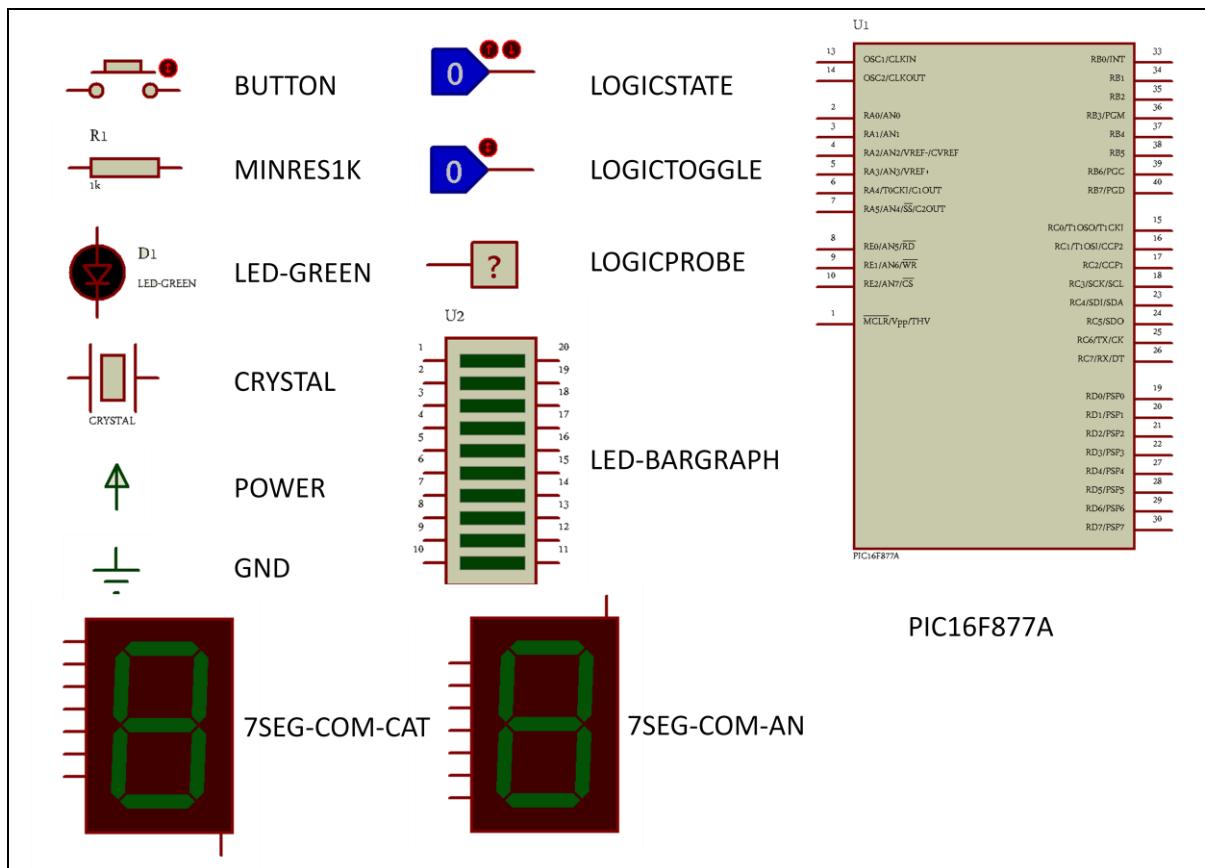


Las principales herramientas que estaremos usando para crear nuestros circuitos son las siguientes:

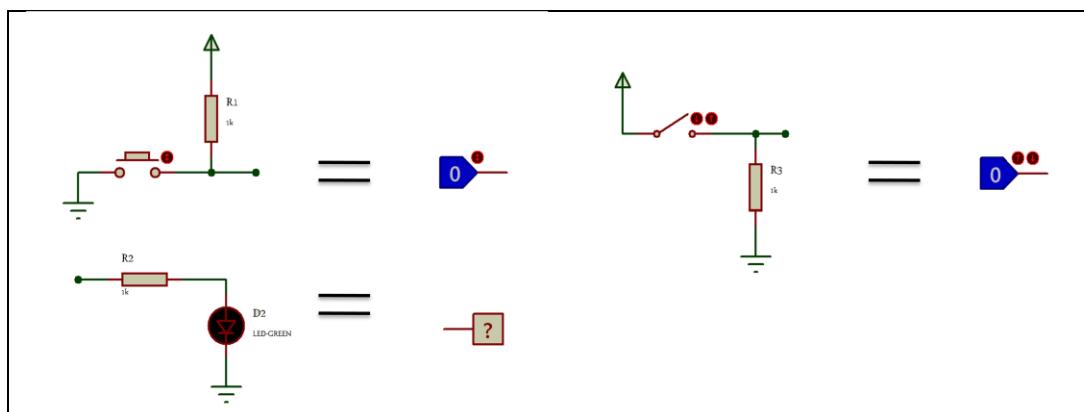




Entre los principales componentes que usaremos comúnmente se enlistan los siguientes:



A excepción de POWER y GND, todos ellos se encuentran en la Herramienta de componentes, los otros se encuentran en Terminales comunes. Algunos de estos componentes son equivalentes entre ellos, por ejemplo cuando deseamos hacer pulsadores con los BUTTON o cuando armamos un visualizador por medio de un led. Estas equivalencias se muestran a continuación.

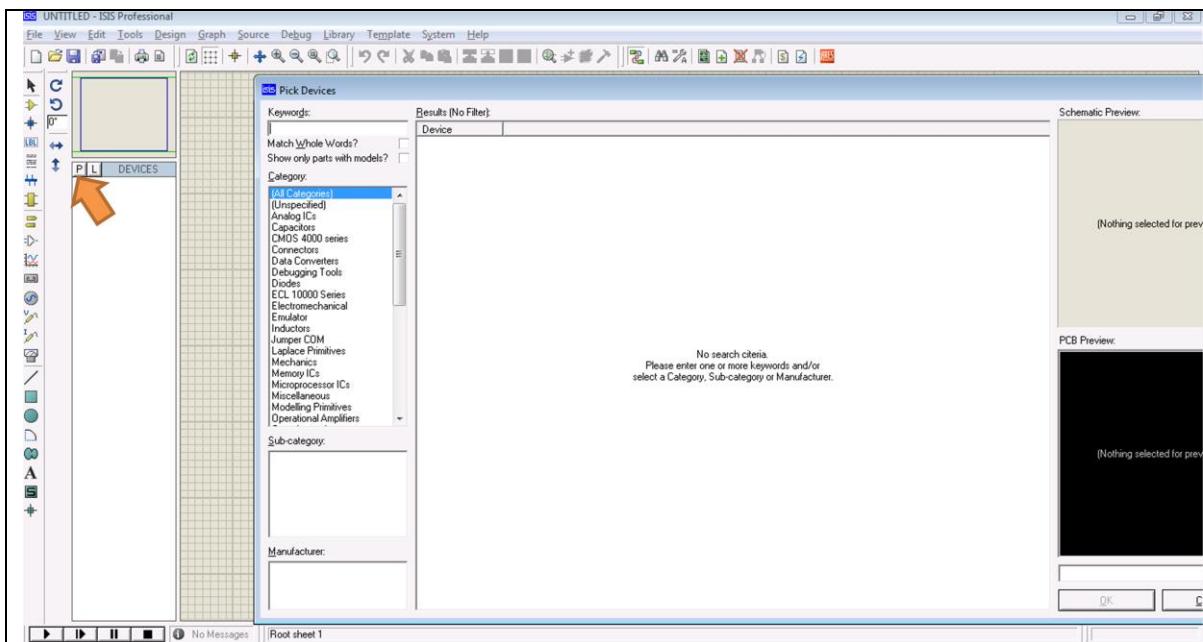


Manual – Aplicaciones Prácticas con Microcontroladores PIC

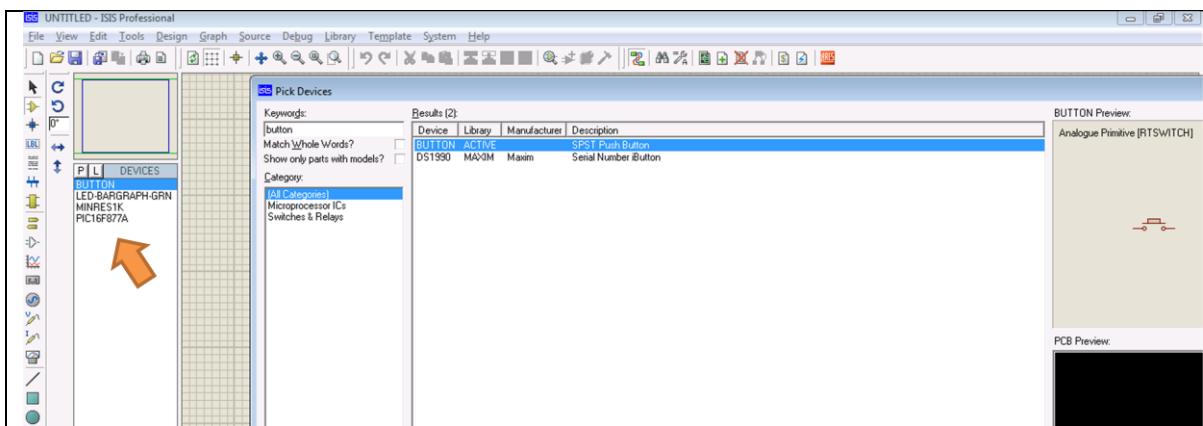
Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Para poder usar un componente en PROTEUS ISIS es necesario primero generar nuestra lista de componentes, esto se hace mediante el botón Pick en la Herramienta de componentes



donde iremos buscando por nombre o por categoría los componentes que deseamos, una vez ubicados, le daremos dos click a cada uno de ellos para importarlos a nuestra lista de componentes.

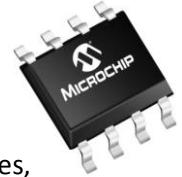


Una vez colocados en nuestra lista daremos click al botón OK y ya podremos usarlos para generar nuestro diagrama esquemático. Y simplemente tendremos que dar click sobre ellos y luego ubicarnos en alguna parte de nuestra zona de diseño y dar nuevamente click para colocarlos.

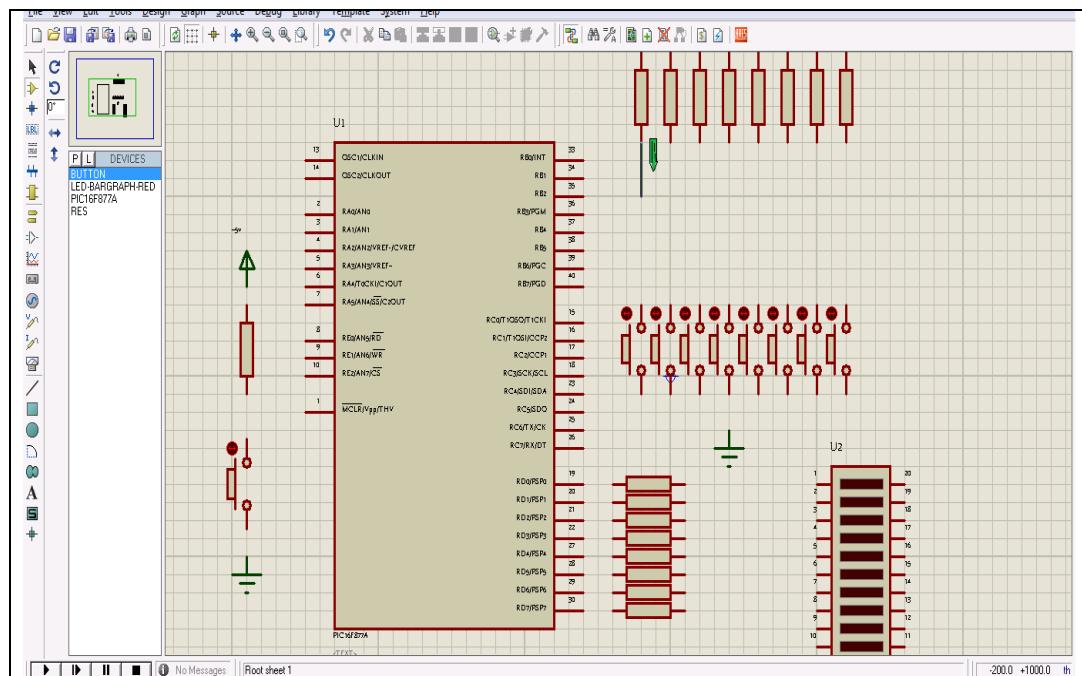
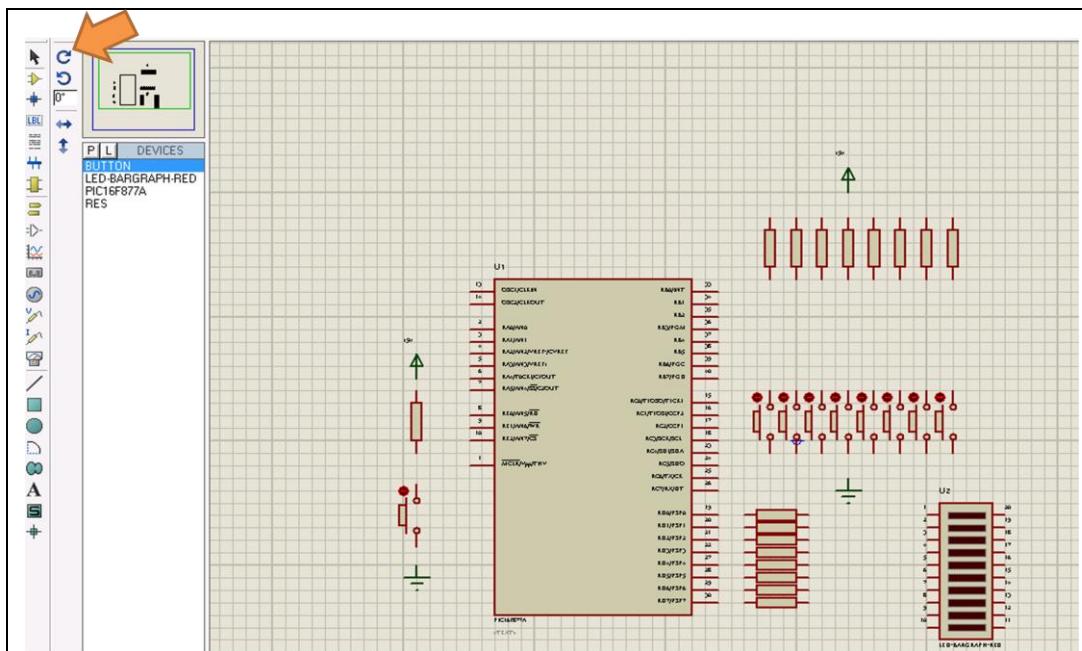


Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán

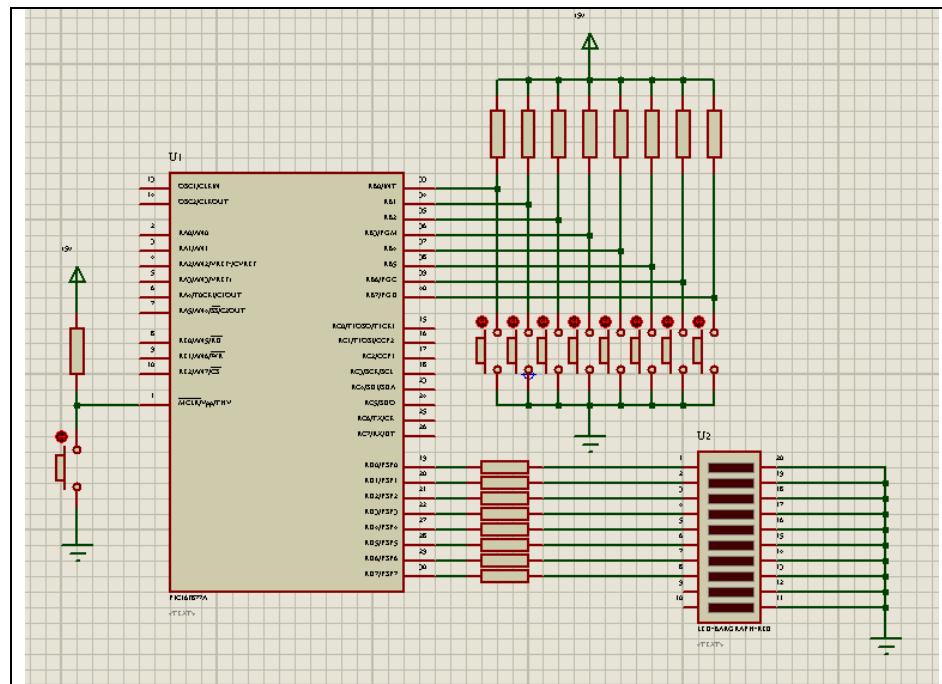


Apoyados en la herramienta de rotación y una vez colocados nuestros componentes, podemos realizar su interconexión, tan solo con poner el cursor sobre un pin, darle un click y luego poner el cursor en algún otro pin y darle click.

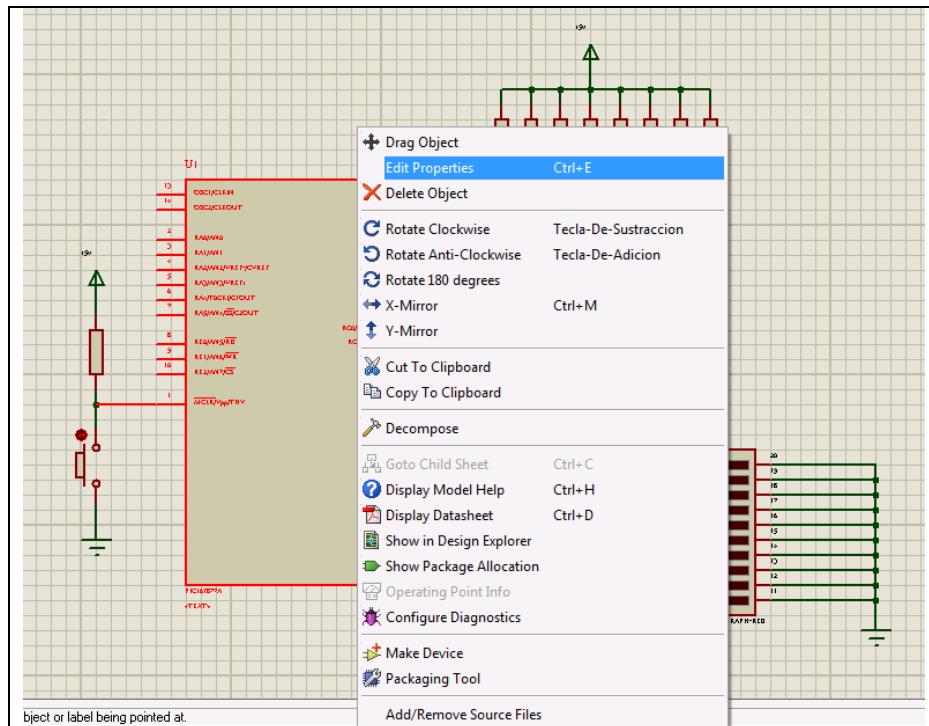


Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Una vez interconectados todos los elementos, tendremos que configurar nuestro microcontrolador para que reciba el programa que hemos compilado en Pic C Compiler, esto se logra dando click derecho en el PIC y luego en Edit Properties.

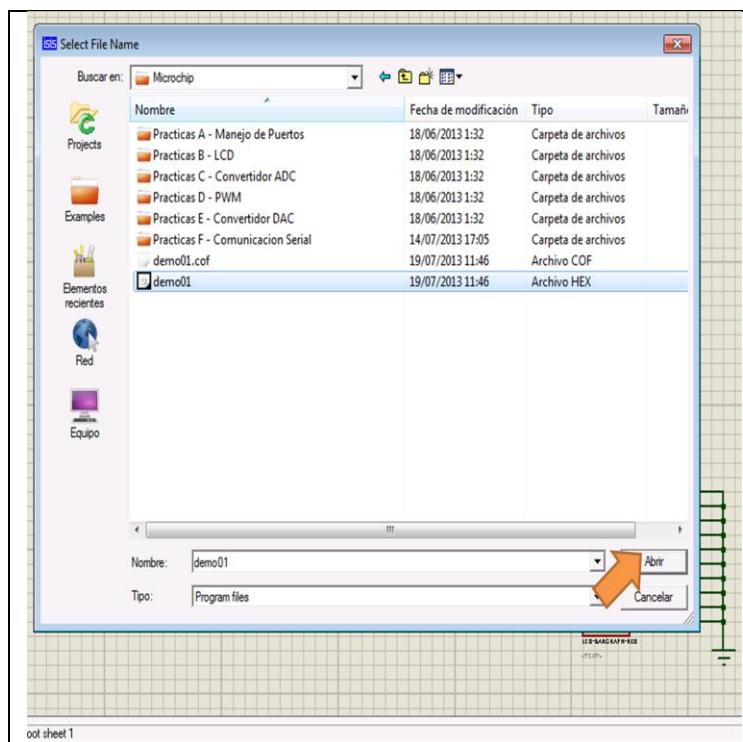
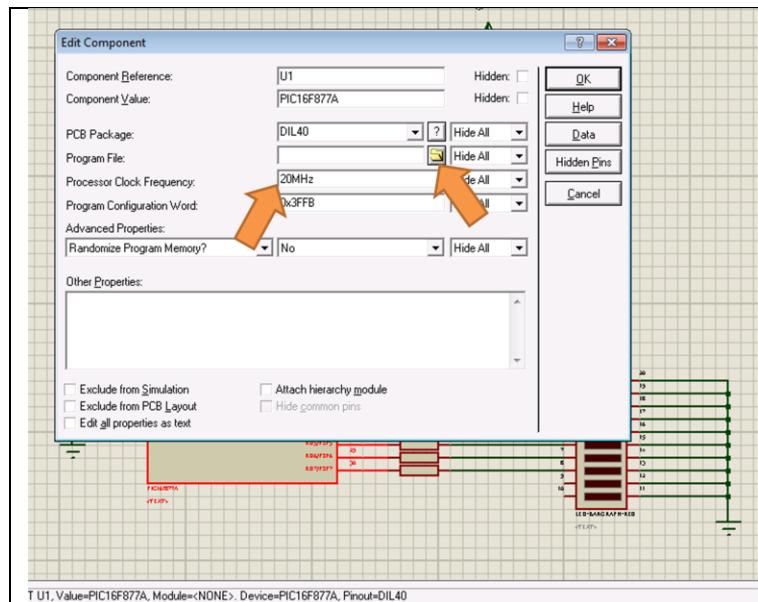


Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Enseguida nos aparecerá una ventana donde las opciones que tenemos que configurar son **Program File**, en la cual le asignaremos el archivo .hex que generamos en la compilación y **Processor Clock Frequency** donde tendremos que especificar al igual que en el programa en C la velocidad del oscilador con la cual estamos trabajando.

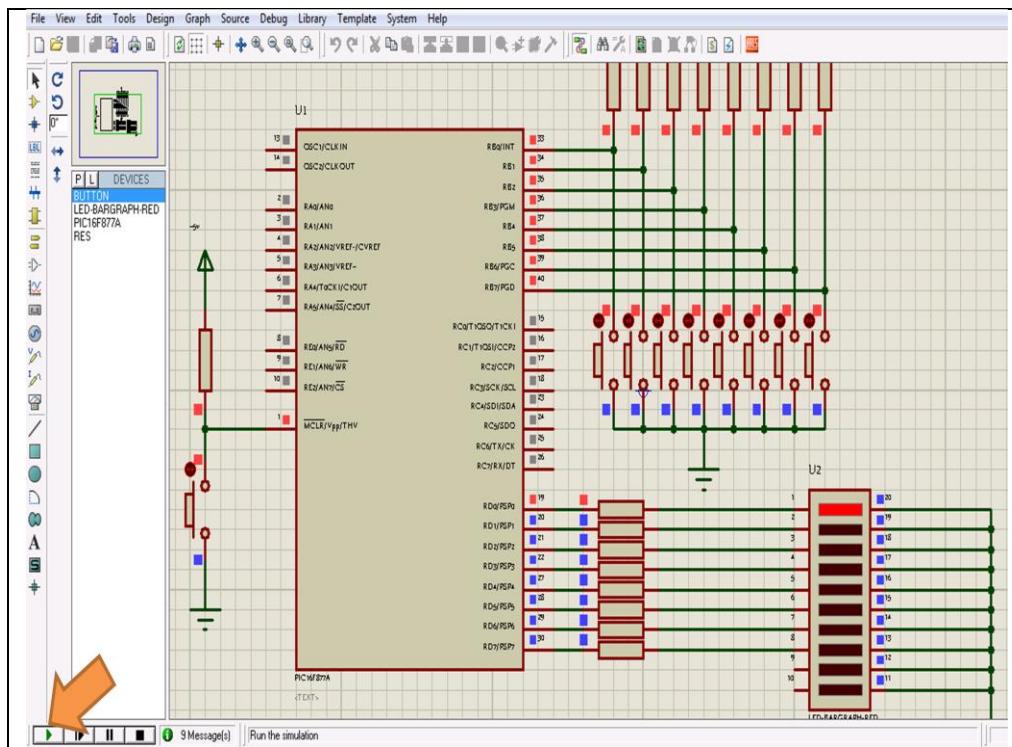
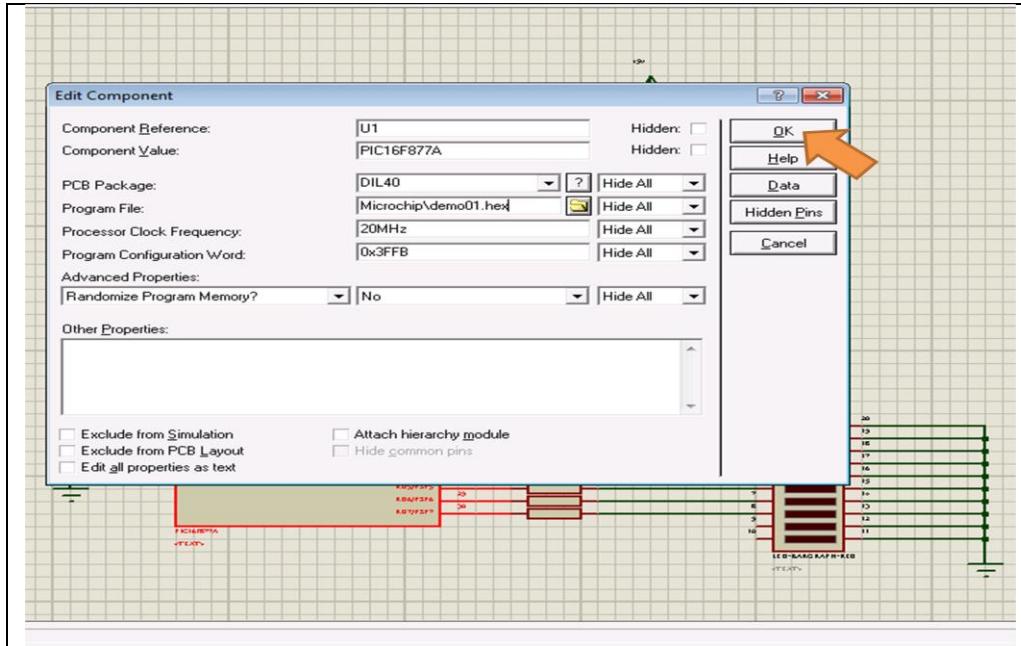


Manual – Aplicaciones Prácticas con Microcontroladores PIC

Elaborado por: M.I.E. Hugo Antonio Méndez Guzmán



Una vez configuradas estas opciones daremos click en OK y nuestro circuito estará listo para simularse con el programa en C que hayamos generado.



Nota: El cristal que se requiere para la señal de reloj no es necesario incluirlo en el esquemático, bastará con las configuraciones anteriores (físicamente sí es necesario).

