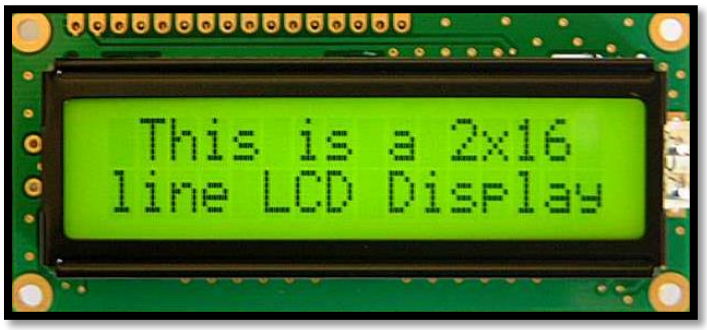
CLASE1: INTRODUCCION A LA PROGRAMACION EN MICROCONTROLADORES PIC Y MANEJO DE PUERTOS DE ENTRADA Y SALIDA

1. DESCRIPCION Y CONFIGURACION DE LA LCD
   1.1. DEFINICION

   El LCD es una pantalla delgada y plana formada por un número de pixeles, estos pixeles en conjunto forman los caracteres que usan para mostrar información.

   El LCD está formado por 2 filas y 16 columnas lo que nos da un total 32 caracteres para mostrar. Estos caracteres están definidos en la memoria interna que tiene el LCD



   1.2. DESCRIPCION DE LOS PINES

PIN	Símbolo	Función	PINES USADOS EN CLASE
1	VSS	GND (0V)	GND
2	VDD	ALIMENTACION (+5V)	+5V
3	V0	Contraste se regula con voltaje	Regular voltaje con pot. 5K
4	RS	Selección de registro	PIN D1
5	R/W	Lectura y escritura	PIN D2
6	E	Señal enable	PIN D0
7	DB0	LINEAS DE DATOS	
8	DB1		
9	DB2		
10	DB3		
11	DB4		PIN D4
12	DB5		PIN D5
13	DB6		PIN D6
14	DB7		PIN D7
15	A	Iluminación del LCD (+5V)	+5V
16	K	Iluminación del LCD (0V)	GND

1.3. FUNCIONES PARA USAR LA LCD

CCS C COMPILER tiene una librería <lcd.c>, donde incluye funciones para la manipulación del LCD.

En el siguiente cuadro se muestra los comandos necesarios para el correcto funcionamiento del LCD:

Comando	Función
#include <lcd.c>	Esta librería contiene las funciones definidas para la posición y escritura de los caracteres en el LCD.
Lcd_init()	Inicializa la configuración del LCD.
lcd_gotoxy(x,y)	Establece la posición de escritura en la pantalla. - X tomará valores de 1 a 16. - Y tomará los valores de 1 ó 2.
Lcd_putc(X)	Mostrará el valor X en la siguiente posición de pantalla del LCD. X es una variable tipo char.

Otra forma de escritura en el LCD es con el siguiente comando:

printf(lcd_putc, cadena, var1, var2,....);

Cadena: Cadena de caracteres, aquí también se definen el tipo de dato de varX.

VarX: Son variables previamente definidos el tipo de dato en la cadena de caracteres.

Ejemplo:

*Printf(lcd_putc, "ADC=**%d** Temp=**%2.2f**", var1, var2);*

Para indicar el tipo de dato en la cadena se usa el siguiente formato ***%nt***, donde **n** es opcional y puede ser:

1-9: Para especificar la cantidad de caracteres a usar.

01-09: Para indicar la cantidad ceros a la izquierda.

1.1-9.9: Para coma flotante.

En el caso de **"t"** es obligatorio y puede tomar los siguientes valores:

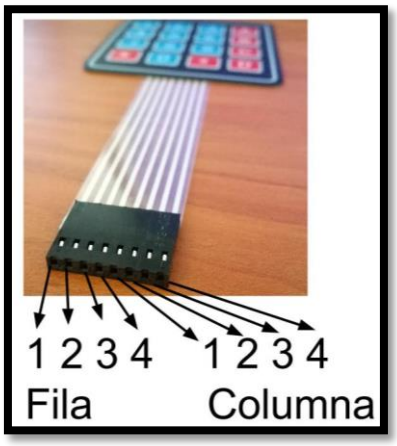
t	Tipo de dato
c	Carácter
s	Carácter o cadena de caracteres
u	Entero sin signo
d	Entero con signo
Lu	Entero largo sin signo
Ld	Entero largo con signo
x	Entero hexadecimal
Lx	Entero largo hexadecimal
f	flotante

2. DESCRIPCION Y CONFIGURACION DEL TECLADO MATRICIAL

2.1. DEFINICION

Un teclado matricial es un arreglo de pulsadores conectados en filas y columnas, podemos encontrar variedad de teclado matricial pero el más común es el teclado matricial 4x4.

El teclado matricial 4x4 solamente ocupa 4 líneas de un puerto para las filas y otras 4 líneas para las columnas, de este modo se pueden leer 16 teclas utilizando solamente 8 líneas de un microcontrolador.



2.2. DESCRIPCION DE LOS PINES

Posición	Pin
FILA 1	PIN_B0
FILA 2	PIN_B1
FILA 3	PIN_B2
FILA 4	PIN_B3
COLUMNA 1	PIN_B4
COLUMNA 2	PIN_B5
COLUMNA 3	PIN_B6
COLUMNA 4	PIN_B7

2.3. FUNCIONES PARA EL TECLADO MATRICIAL

CCS C COMPILER tiene una librería <kbd.c>, donde incluye funciones para la manipulación del teclado matricial.

Es necesario resaltar que los valores obtenidos del teclado matricial son de tipo char o carácter ya que la librería así lo define.

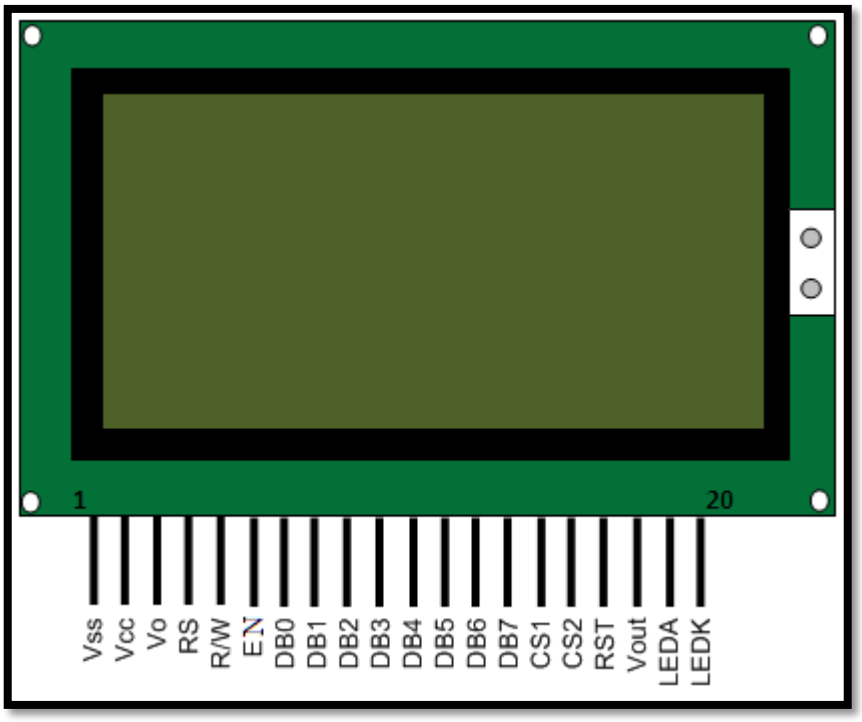
En el siguiente cuadro se muestra los comandos necesarios para el correcto funcionamiento del teclado matricial:

Comando	Función
#include <kbd.c>	Esta librería contiene las funciones definidas para la obtención de ***caracteres*** del teclado matricial.
kbd_init()	Inicializa la configuración del teclado matricial.
Kbd_getc()	Obtiene el valor seleccionado en el teclado matricial. Esta función debe ser llamada con frecuencia para que no se pierda una pulsación de tecla.
Port_b_pullups(true)	Activa las resistencias pull up del microcontrolador necesario para el funcionamiento de teclado matricial. *Se escribe luego de llamar la función de inicio kbd_init().*

3. DESCRIPCION Y CONFIGURACION DE LA GLCD

3.1. DEFINICION

Las LCDs de caracteres 16x2 tienen sus propias limitaciones; Sólo pueden mostrar caracteres de ciertas dimensiones. Las pantallas LCD gráficas se utilizan para mostrar caracteres e imágenes personalizadas. Las LCD gráficas encuentran uso en muchas aplicaciones; Se utilizan en videojuegos, teléfonos móviles, ascensores, etc. como unidades de visualización.



3.2. DESCRIPCION DE LOS PINES

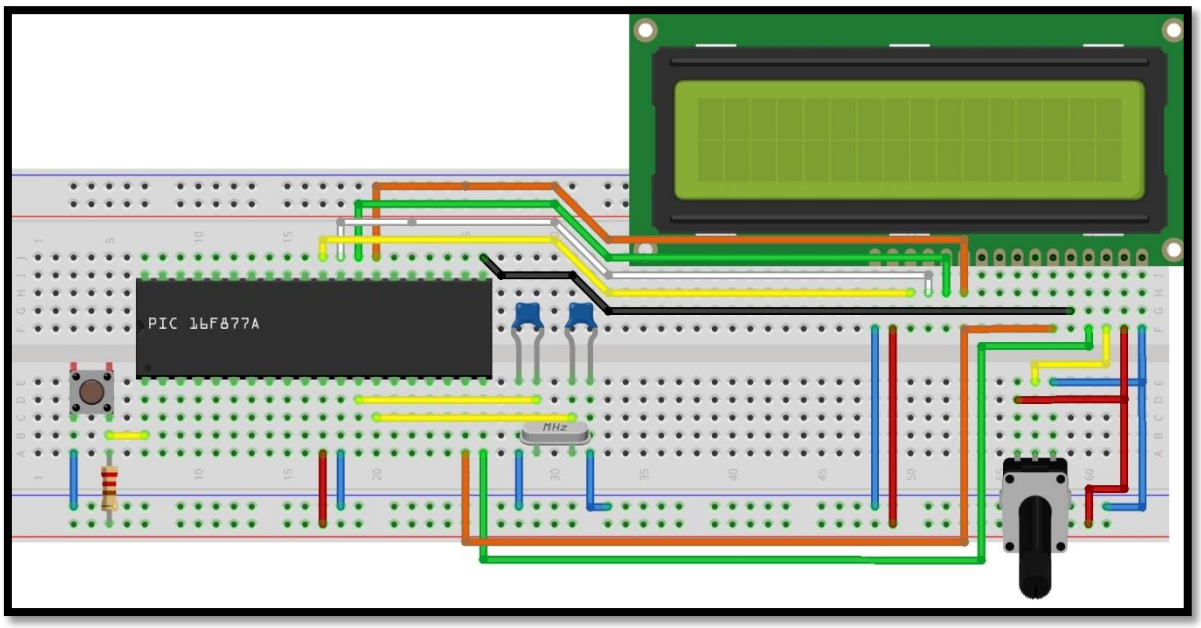
Pin no.	Function	Name
1	Ground (0 V)	Vss
2	Supply voltage; 5V	Vcc
3	Contrast adjustment	Vo
4	High to display data; Low for instruction code	Register select (RS)
5	Low to write to the register; High to read from the register	Read/Write (R/W)
6	Reads data when high; Writes data at high to low transition (falling edge)	Enable (EN)
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Chip selection for IC1; Active high	CS1
16	Chip selection for IC2; Active high	CS2
17	Reset signal; Active low	RST
18	Output voltage for LCD driving	Vout
19	Backlight V_{CC} (5V)	LED A
20	Backlight Ground (0V)	LED K

3.3. FUNCIONES PARA LA GLCD

Comando	Función
#include < HDM64GS12.c>	Esta librería contiene las funciones definidas para iniciar y configurar los pines de la GLCD
#include < graphics.c>	Funciones para realizar gráficos en la GLCD
glcd_init()	Inicializa la configuración de la GLCD.
glcd_pixel(x,y,color)	Establece el color del pixel
glcd_fillscreen(color)	rellena la GLCD de un color determinado puede activarse o desactivarse
glcd_line(x1,y1,x2,y2,color)	Dibuja un rectángulo con un vértice en (x1, y1) y el otro en (x2, y2). Puede ser rellenado o no y puede activarse un color o no.
glcd_rect(x1,y1,x2,y2,fill,color)	Dibuja un rectángulo con un vértice en (x1, y1) y el otro en (x2n, y2). Puede ser rellenado o no y puede activarse un color o no.
glcd_bar(x1,y1,x2,y2,width,color)	Dibuja una barra desde el primer punto al segundo; se puede definir el número del rango de pixeles y puede activarse el color o no.
glcd_circle(x,y,radius,fill,color)	dibuja un circulo con centro en (x,y) y con el radio especificado; puede rellenarse o no y puede activarse el color o no.
glcd_text57(x,y,textptr,size,color)	Escribe el texto empezando en (x,y); y los caracteres son de 5x7 pixeles se puede

	escalar el tamaño y puede activarse el color o no. Esta función envía los caracteres a la línea siguiente(se debe usar #define GLCD_WIDTH para definir el ancho de visualización)

4. EJEMPLOS DE LA CLASE1
   4.1. EJEMPLO1
   Enviar Mensaje “Hola LaboTEC” en el LCD
   4.1.1. CONEXIÓN

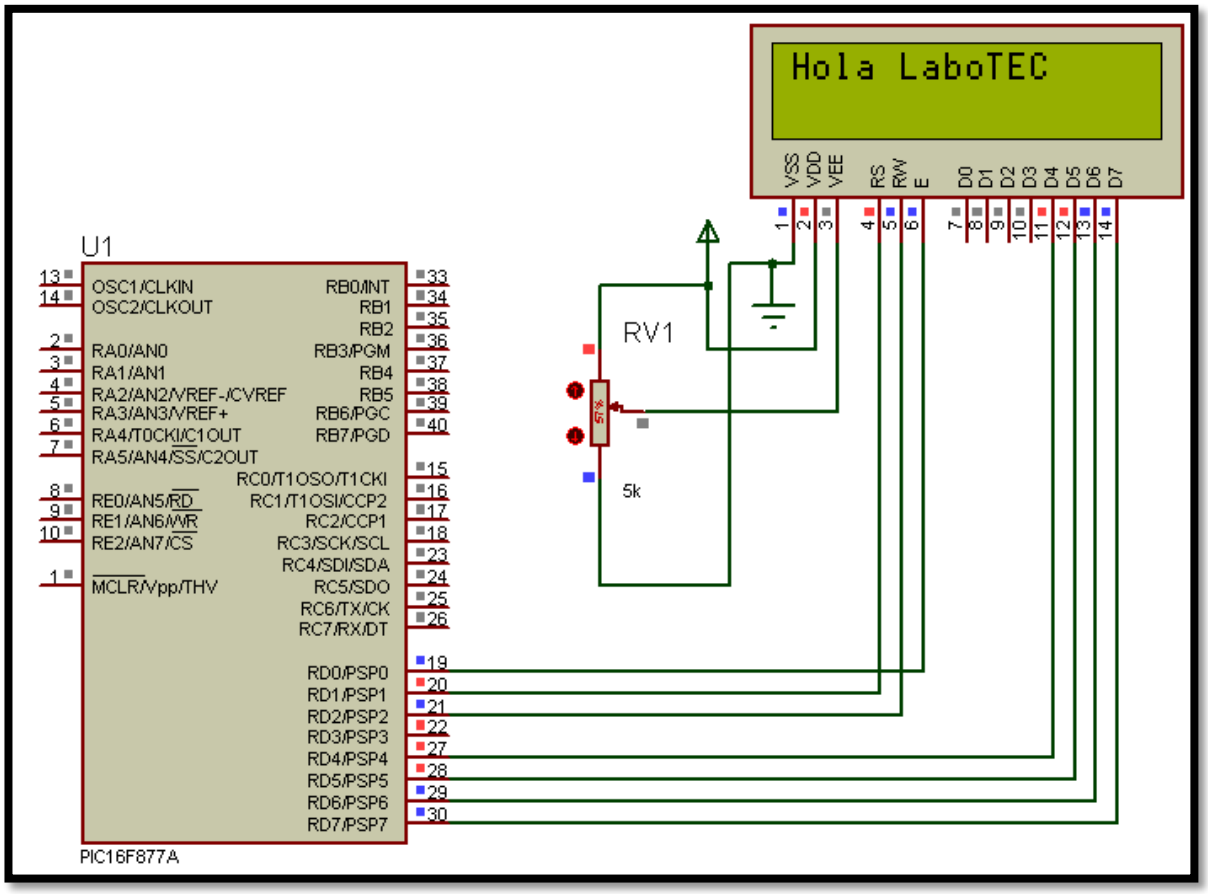


4.1.2. PROGRAMA

Ejemplo1.c

```
1  #include <16f877a.h>
2  #use delay(clock=4M)
3  #fuses XT
4  #include <lcd.c>
5
6  void main(void){
7
8     lcd_init();
9     printf(lcd_putc,"Hola LaboTEC");
10    for(;;){
11
12    }
13 }
```

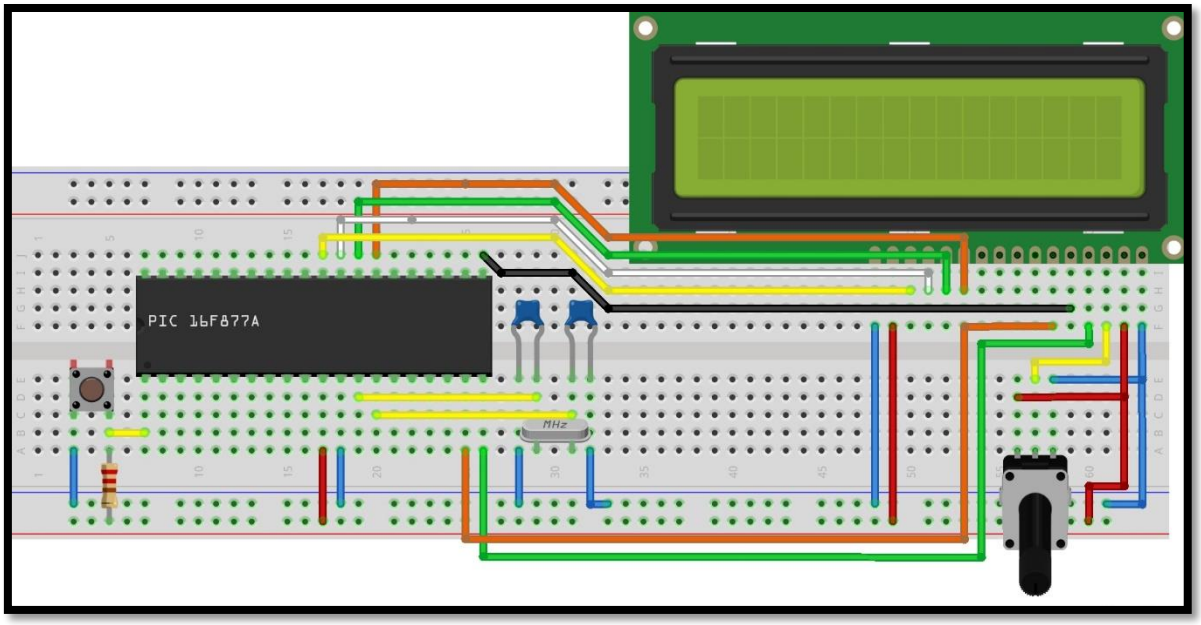
4.1.3. SIMULACION



4.2. EJEMPLO2

Realizar un Conteo de 0 a 100 y mostrarlo en la LCD

4.2.1. CONEXIÓN

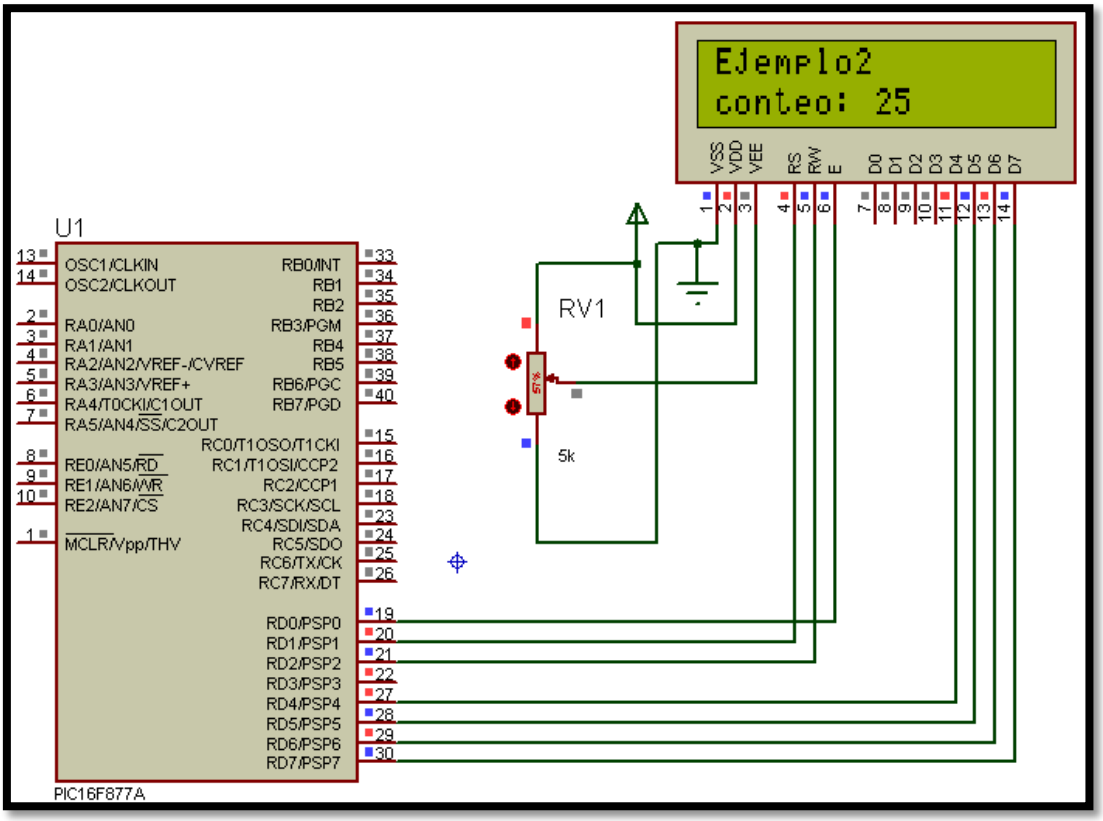


4.2.2. PROGRAMA

Ejemplo2.c

```
#include <16f877a.h>
#use delay(clock=4M)
#fuses XT
#include <lcd.c>

void main(void){

   long i=0;
   lcd_init();
   for(;;){
      for(i=0;i<=100;i++){
         lcd_putc('\f');
         printf(lcd_putc,"Ejemplo2");
         lcd_gotoxy(1,2);
         printf(lcd_putc,"conteo: %lu",i);
         delay_ms(50);
      }
   }
}
```

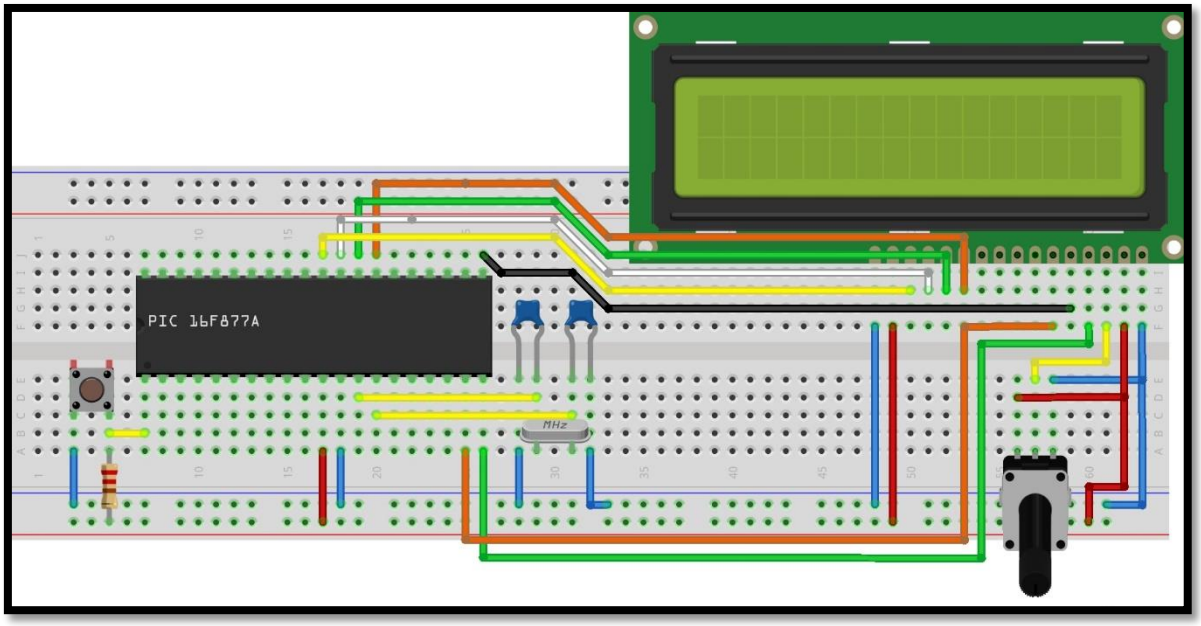
4.2.3. SIMULACION



4.3. EJEMPLO3

Desplazar el mensaje “Hola LaboTEC” en la fila1 y fila2 de la LCD

4.3.1. CONEXIÓN

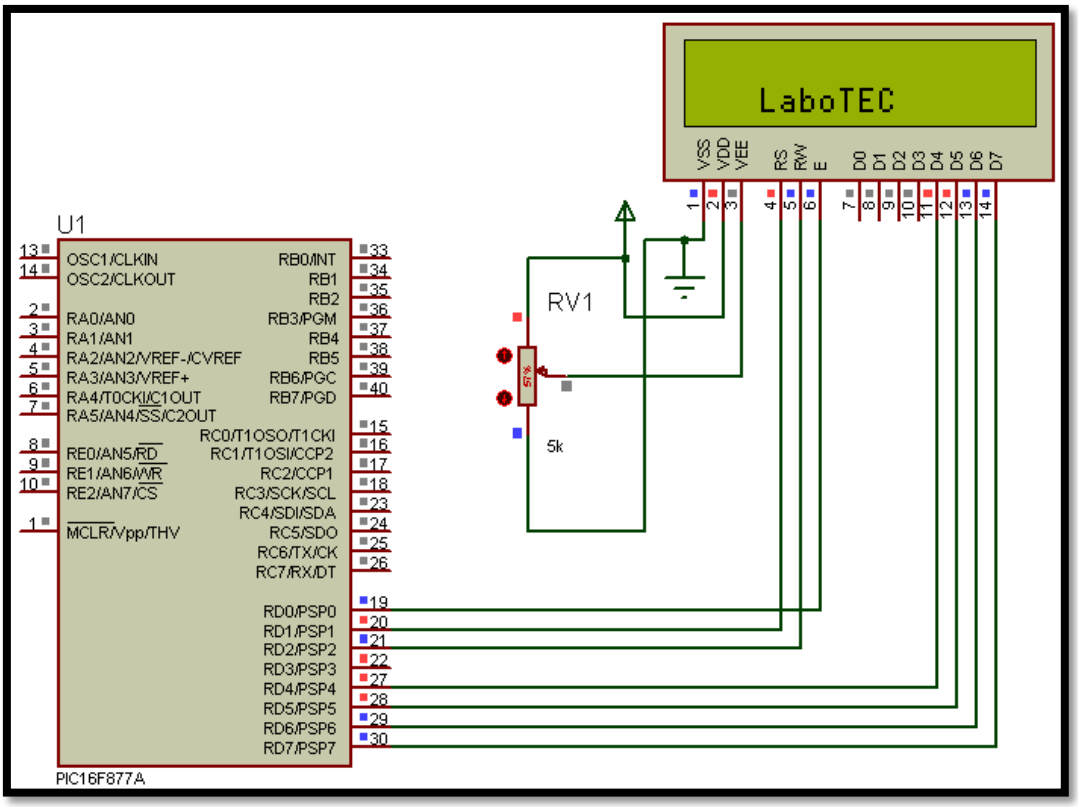


4.3.2. PROGRAMA

Ejemplo3.c

```
1  #include <16f877a.h>
2  #use delay(clock=4M)
3  #fuses XT
4  #include <lcd.c>
5
6  void main(void){
7
8     int x=0,y=0;
9     lcd_init();
10    while(1){
11       for(y=1;y<=2;y++){
12          for(x=1;x<=16;x++){
13             lcd_putc('\f');
14             lcd_gotoxy(x,y);
15             printf(lcd_putc,"LaboTEC");
16             delay_ms(300);
17          }
18       }
19    }
20 }
```

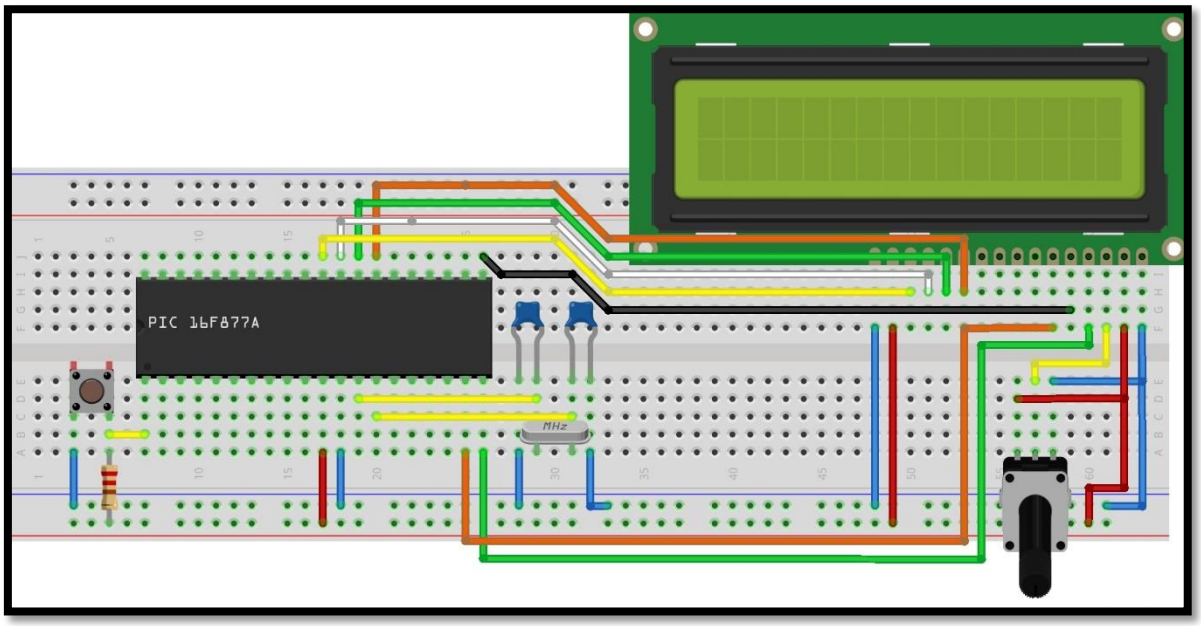
4.3.3. SIMULACION



4.4. EJEMPLO4

Crear dos nuevos caracteres y mostrarlos en la LCD

4.4.1. CONEXIÓN

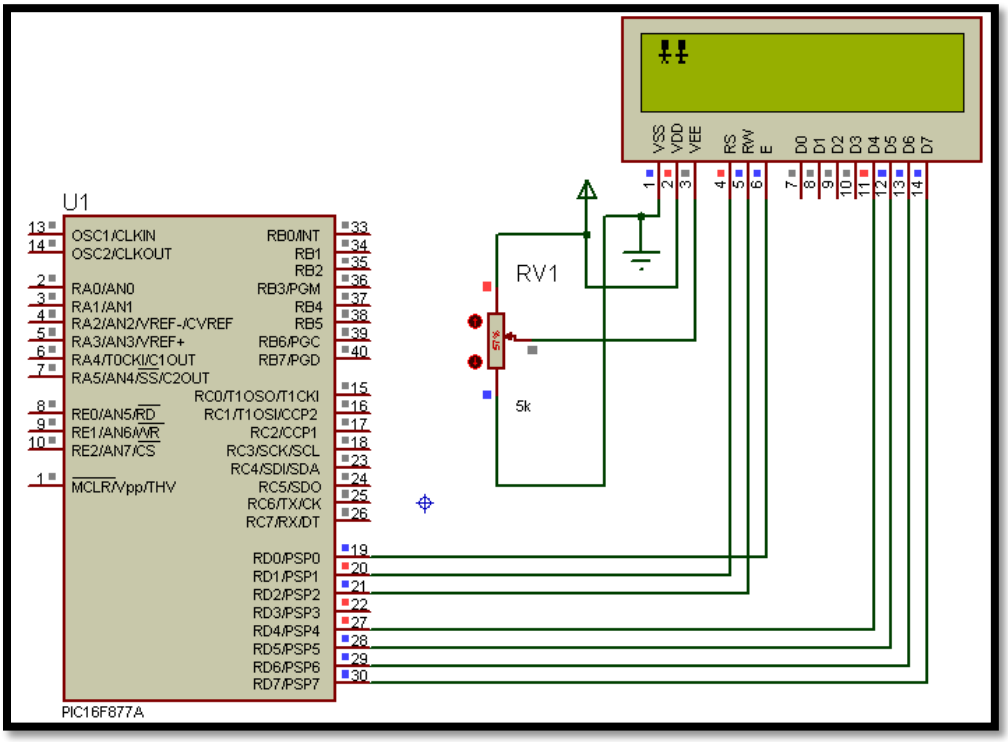


4.4.2. PROGRAMA

Ejemplo4.c

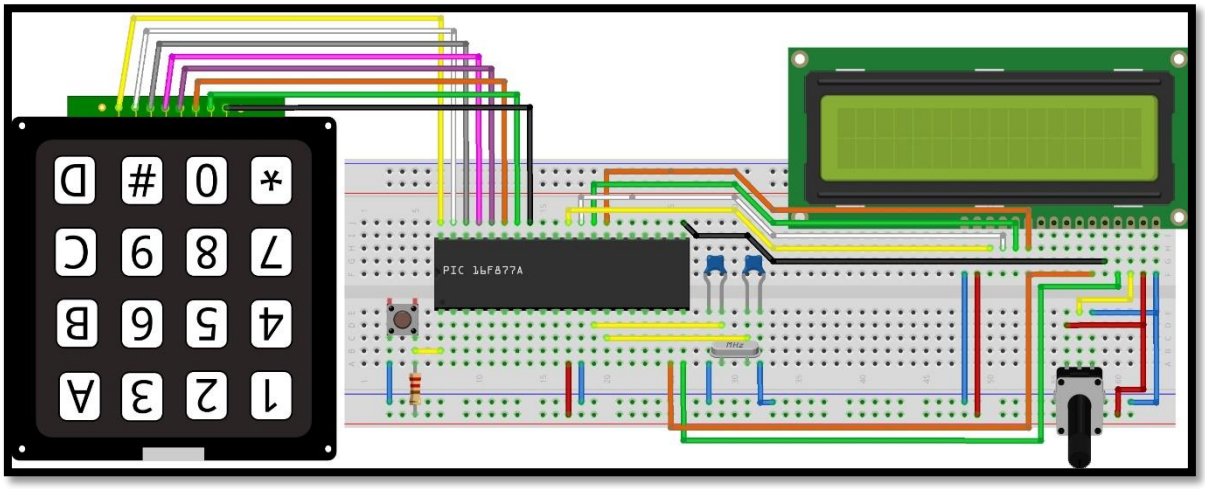
```
#use delay(clock=4M)
#fuses XT
#include <lcd.c>
void nuevo_caracter(void){
   lcd_send_byte(0,0x40);
   lcd_send_byte(1,0b00001110);lcd_send_byte(1,0b00001110);
   lcd_send_byte(1,0b00001110);lcd_send_byte(1,0b00000100);
   lcd_send_byte(1,0b00011111);lcd_send_byte(1,0b00000100);
   lcd_send_byte(1,0b00001010);lcd_send_byte(1,0b00010001);
   lcd_send_byte(0,0x48);
   lcd_send_byte(1,0b00001110);lcd_send_byte(1,0b00001110);
   lcd_send_byte(1,0b00001110);lcd_send_byte(1,0b00000100);
   lcd_send_byte(1,0b00011111);lcd_send_byte(1,0b00000100);
   lcd_send_byte(1,0b00001110);lcd_send_byte(1,0b00011111);
}

void main(void){
   lcd_init();
   nuevo_caracter();
   lcd_putc('\f');
   lcd_putc(0);
   lcd_putc(1);
   for(;;){
   }
}
```

4.4.3. SIMULACION



4.5. EJEMPLO5

4.5.1. CONEXIÓN

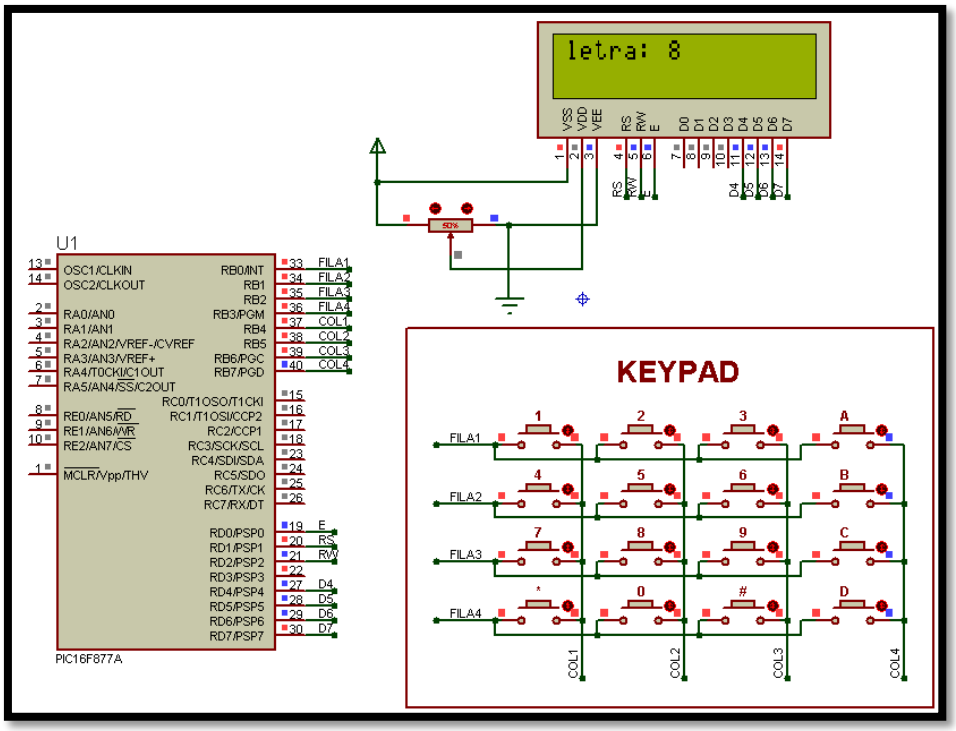


4.5.2. PROGRAMA

Ejemplo5.c

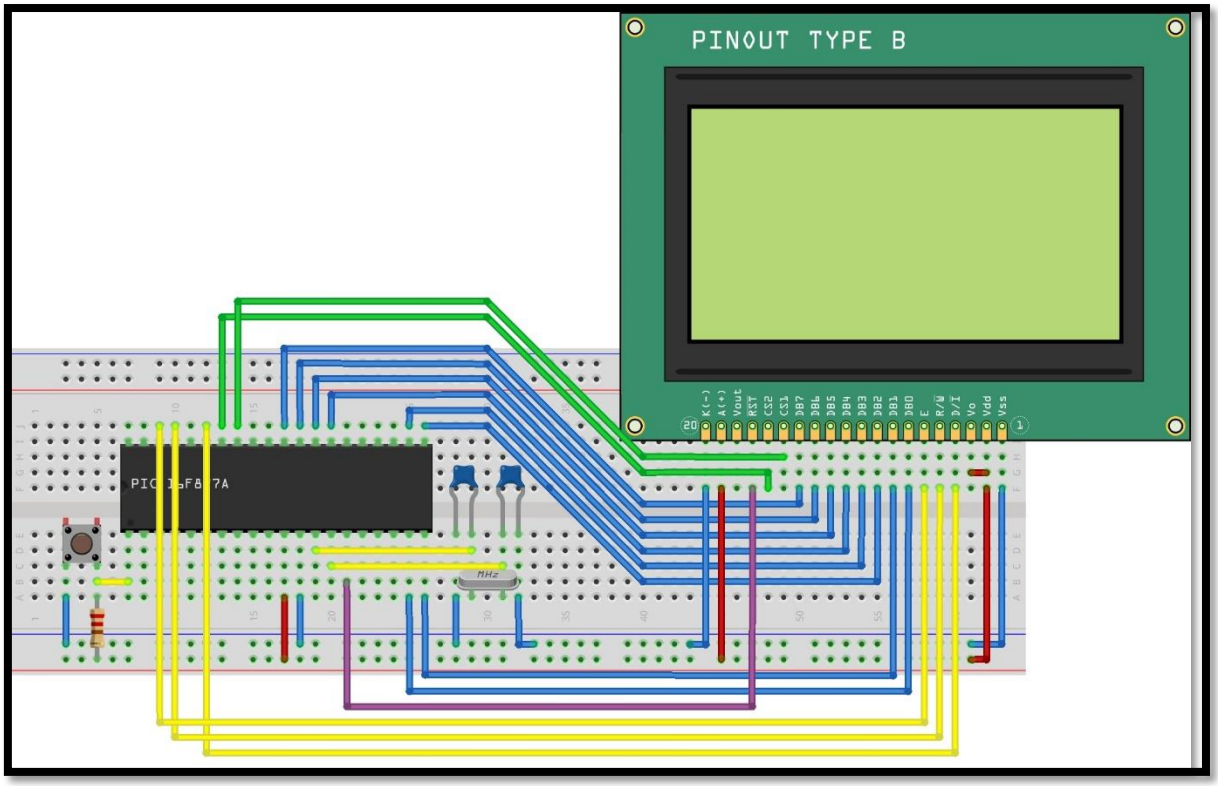
```c
#include <16f877a.h>
#use delay(clock=4M)
#fuses XT
#include <lcd.c>
#include "keypad.c"

void main(void){
   char c;
   lcd_init();
   kbd_init();
   port_b_pullups(True);
   lcd_gotoxy(1,1);
   printf(lcd_putc,"Presionar");
   lcd_gotoxy(1,2);
   printf(lcd_putc,"Una tecla");
   for(;;){
      c=kbd_getc();
      if(c != 0){
         lcd_putc('\f');
         printf(lcd_putc,"letra: %c",c);
      }
   }
}
```

4.5.3. SIMULACION



4.6. EJEMPLO6

4.6.1. CONEXIÓN



4.6.2. PROGRAMA

Ejemplo6.c

```
#fuses HS
#use delay(clock=20Mhz)
#include <HDM64GS12.c>
#include <graphics.c>

void main(){

   int data[]="LaboTEC";
   glcd_init(ON);

   glcd_line(10,10,50,50,ON);
   glcd_line(50,10,10,50,ON);
   glcd_rect(10,10,50,50,0,ON);
   glcd_text57(60,10,data,1,ON);

   for(;;){
   }
}
```

4.6.3. SIMULACION