https://encrypted-tbn2.google.com/images?q=tbn:ANd9GcSwvtSk7hcAFdTCZM6CZtQj2WWGjq3DTx1SlnuBzlzfW_73nYyG

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN**

**FIEC**

**LABORATORIO DE MICROCONTROLADORES**

**Proyecto Final:**

Lectura de valores análogos de un potenciómetro – Ahorcado

**Profesor:**

Ing. Carlos Valdivieso.

**Estudiante:**

Pacheco Pico Johanna

**Paralelo:**

# 3

**Fecha de presentación:**

Jueves, 23 de agosto del 2012

**I TÉRMINO**

**2012 – 2013**

1. **Enunciado del Proyecto.**

Para realizar este proyecto había que dividir en 2 partes, puesto que como especificaciones nos pedían que hagamos un ejercicio propuesto por el profesor que consistía en la lectura de los valores analógicos del potenciómetro que se encuentra conectado en la patita RE0 del PIC16F887, y presentar las variaciones de voltajes en valores binarios y decimal. Para este caso convertimos el valor en bytes, eso es lo que se muestra en la pantalla LCD 4X20.

Cuando se inicia la simulación del proyecto en Proteus sale un mensaje inicial donde se indica que es el proyecto de microcontroladores, y las iniciales JKPP junto con el paralelo al que pertenezco, esto pasa rápidamente de ahí en la LCD se verá el menú principal.

Se debe indicar que durante toda la ejecución del proyecto permanece el display multiplexado de 7 segmentos y 2 dígitos encendido, mostrando las iniciales de mi nombre JP, este display solo se apaga cuando termina la ejecución o cuando se realiza el reseteo correspondiente.

Ya expliqué que se realiza en la parte uno de proyecto, ahora se indicará el funcionamiento de la segunda parte: al entrar a la opción 2 de menú principal el juego comenzará, para este juego se usó el teclado 4x4, la pantalla LCD de 4X20.

El juego consiste en una simulación del ahorcado, la manera más fácil que encontré fue usar una sola palabra y el jugador solo tiene 5 oportunidades de equivocarse a la sexta vez que introduce mal una letra pierde, en el LCD solo se muestra las letras correctas y se puede ver si el jugador está perdiendo cuando se van los leds del ahorcado.

1. **Diagrama de Bloques.**

**ENTRADAS**

**MICROCONTROLADORES**

PIC16F887

(Programación)

**SALIDAS**

**TECLADO 4X4**

**RESET**

**POTENCIOMETRO**

**LCD 4X20**

**LEDS**

**DISPLAY MULTIPEXADO DE 7 SEGMENTOS 2 DIGITOS.**

Figura 1.- Diagrama de bloques de proyecto.

1. **Descripción del algoritmo o estrategia utilizada.**

**Descripción del Algoritmo:**

1. Se realiza la asignación de las terminales del puerto B para la LCD.
2. Se realiza la asignación de las terminales del puerto A para el conjunto de resistencias que forman al ahorcado.
3. Se realiza la asignación de las terminales del puerto C para el display de 7 segmentos y 2 dígitos donde aparecerá las iniciales del nombre.
4. Se realiza la asignación de las terminales del puerto D para el teclado 4X4.
5. Declaración de variables y seteo de TMR0
6. Configuración del pin AN5 del RE0 como entrada analógica.
7. Se configuran todos los pines de los puertos A, B, C, D y E como salidas digitales.
8. Se deshabilita los comparadores.
9. Se realiza la inicialización del keypad y del LCD.
10. Se apaga el cursor para tener una mejor visibilidad en el LCD.
11. Se muestra en el LCD los mensajes iniciales.
12. Se realiza la presentación de las letras iniciales del nombre en el display multiplexado.
13. Se muestra en el LCD un pequeño menú y se espera a que el usuario presione alguna tecla (1 o 2).
14. Si presiona y suelta en el teclado 1, entonces se realizará la primera parte que consiste en mostrar en el display valores del convertidor de ADC en binario y decimal. De ahí hay que presionar D para salir de esta parte.
15. Si presiona y suelta la tecla 2, entonces se inicia el juego.
16. En la LCD se muestra un presione # para continuar, si no presiona # no podrá acceder al juego.
17. Se muestra en la LCD una serie de 13 “\*” asteriscos, los mismos que indican que es una palabra de 13 letras.
18. Presionamos cualquier letra en el teclado, si es correcta se muestra la letra en el LCD, si no lo es entonces se prende un led de la parte del ahorcado donde indica que el usuario ha fallado, solo se tiene 6 intentos de fallar.
19. Si el usuario llegó al número de intentos máximos de fallas entonces en el LCD se muestra de forma intermitente que perdió y tampoco se muestran que letras están ocultas. Mientras el usuario no acierte no verá ninguna letra en el display son verá asteriscos.
20. Si el usuario acierta se mostrará que letra es la correcta y se borrarán los asteriscos, una vez que el usuario ha completado toda la palabra correctamente entonces el display rápidamente cambia de estado y muestra el mensaje que el usuario ha ganado.
21. Después que esto ocurre el juego termina pero siempre nos mantendremos jugando, si deseamos salir entonces tenemos que pulsar el botón de reset, con lo que el proyecto de nuevo se ejecuta manteniendo sus valores iniciales.
22. **Diagrama de flujo funcional del programa principal**



Fig. 2.- Diagrama Funcional del primer ejercicio usando LCD.

1. **Listado del programa fuente en lenguaje ensamblador con comentarios en las líneas de código que considere fundamentales.**

***Programa principal:***

#include "mask.c"

char keypadPort at PORTD;

// LCD module connections

sbit LCD\_RS at RB4\_bit;

sbit LCD\_EN at RB5\_bit;

sbit LCD\_D4 at RB0\_bit;

sbit LCD\_D5 at RB1\_bit;

sbit LCD\_D6 at RB2\_bit;

sbit LCD\_D7 at RB3\_bit;

sbit LCD\_RS\_Direction at TRISB4\_bit;

sbit LCD\_EN\_Direction at TRISB5\_bit;

sbit LCD\_D4\_Direction at TRISB0\_bit;

sbit LCD\_D5\_Direction at TRISB1\_bit;

sbit LCD\_D6\_Direction at TRISB2\_bit;

sbit LCD\_D7\_Direction at TRISB3\_bit;

// End LCD module connections

unsigned short dig\_no=0;

int name2, name1;

const char \*txt1[] = {"PROYECTO2 UC","JKPP - P2","MENU PRINCIPAL","1)ADC ","2)AHORCADO"};

const char \*txt2[] = {"el ADC es: ","Valor leido por","binario: ","decimal: ","voltage: "};

const char \*txt3[] = {"Nota: Si desea salir ","presione la tecla D"};

const char \*txt5[] ={"PRESS: #","Para comenzar"};

const char \*txt9[] = {"USTED HA","PERDIDO EL JUEGO","GANADO EL JUEGO"};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char\* codetxt\_to\_ramtxt(const char\* ctxt)

{

static char txt[20];

char i;

for(i =0; txt[i] = ctxt[i]; i++);

return txt;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void conv(){

int j;

long adc;

char temp1[10];

adc= ADC\_Read(5); // Conversión A/D. Pin RE0 es una entrada.

//MOSTRAR EL VALOR ANALOGICO EN BINARIO

for(j=0;j<10;j++){

temp1[j]=(int)adc%2;

Lcd\_Chr(4,20-j,48+temp1[j]);

adc=adc/2;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void POT() {

unsigned short tecla2=0;

int temp\_res;

unsigned char txt[4];

unsigned char txt15[11];

Lcd\_cmd(\_lcd\_clear);

delay\_ms(40);

Irp\_bit=0;

while(tecla2!=16){

Lcd\_out(1,1,codetxt\_to\_ramtxt(txt2[1]));

Lcd\_out(2,1,codetxt\_to\_ramtxt(txt2[0]));

temp\_res = ADC\_Read(5); // Get 10-bit results of AD conversion

delay\_ms(10);

wordtostr(temp\_res,txt);

Lcd\_out(3,12,txt);

Lcd\_out(3,1,codetxt\_to\_ramtxt(txt2[3]));

Irp\_bit=1;

conv();

delay\_ms(10);

Lcd\_out(4,1,codetxt\_to\_ramtxt(txt2[2]));

Lcd\_out(4,10,txt15);

IRP\_bit = 0;

tecla2=keypad\_key\_click();

if(tecla2!=0 && tecla2!=16)

{

Lcd\_cmd(\_Lcd\_clear);

delay\_ms(40);

Lcd\_out(2,1,codetxt\_to\_ramtxt(txt3[0]));

Lcd\_out(3,1,codetxt\_to\_ramtxt(txt3[1]));

delay\_ms(1000);

Lcd\_cmd(\_lcd\_clear);

}

}

Lcd\_cmd(\_lcd\_clear);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void JUEGO(){

unsigned short kp, cnt,cnt2,i,n,band,band2,band3;

char FR[16];

cnt = 0;

i=0;

n=13;

kp = 0; // Reset key code variable

Lcd\_cmd(\_lcd\_clear);

// Wait for key to be pressed and released

do{

do{

Lcd\_cmd(\_lcd\_clear);

Lcd\_out(2,1,codetxt\_to\_ramtxt(txt5[0]));

Lcd\_out(3,1,codetxt\_to\_ramtxt(txt5[1]));

kp = Keypad\_Key\_Click(); // Store key code in kp variable

Delay\_ms(100);

}while (kp!=15);

FR[0]='C';

FR[1]='O';

FR[2]='D';

FR[3]='I';

FR[4]='G';

FR[5]='O';

FR[6]='D';

FR[7]='I';

FR[8]='F';

FR[9]='I';

FR[10]='C';

FR[11]='I';

FR[12]='L';

Lcd\_Cmd(\_LCD\_CLEAR);

for (i =0;i < n; i++){

Lcd\_Chr(1,i+1,'\*');

}

i=0;

band=0;

band2=0;

band3=1;

PORTC=0;

cnt=0;

kp=0;

cnt2=0;

do{

kp=0;

do{

kp = Keypad\_Key\_Click(); // Store key code in kp variable

Delay\_ms(100);

}while (!kp);

kp=kp+64;

band=0;

for (i =0;i < n; i++){

if(FR[i]==kp){

Lcd\_Chr(1,i+1,FR[i]);

band=1;

}

}

if(band==0){

band2++;

cnt++;

if(cnt==1){

porta=1;

}

if(cnt==2){

porta=3;

}

if(cnt==3){

porta=7;

}

if(cnt==4){

porta=15;

}

if(cnt==5){

porta=31;

}

if(cnt==6){

porta=63;

}

}

else{

cnt2++;

}

if(band2==6 || cnt2>=7 ){

band3=0;

}

}while (band3);

Lcd\_Cmd(\_LCD\_CLEAR);

if(cnt2<=6){

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(1, 1,codetxt\_to\_ramtxt(txt9[0]));

Lcd\_Out(2, 1,codetxt\_to\_ramtxt(txt9[1]));

Delay\_ms(2000);

}

else{

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(1, 1,codetxt\_to\_ramtxt(txt9[0]));

Lcd\_Out(2, 1,codetxt\_to\_ramtxt(txt9[2]));

Delay\_ms(2000);

}

}while(kp!=15);

}

void interrupt(){

if (dig\_no==0) {

PORTE = 0; // Turn off both displays

PORTC = name1; // Set mask for displaying ones on PORTD

PORTE = 2; // Turn on display for ones (LSD)

dig\_no = 1;

}

else {

PORTE=0;

PORTC = name2; // Set mask for displaying tens on PORTD

PORTE = 4; // Turn on display for tens (MSD)

dig\_no = 0;

}

TMR0 = 0; // Reset counter TMRO

INTCON = 0x20; // Bit T0IF=0, T0IE=1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

unsigned short tecla=0;

ANSEL = 0x01; // Configure AN5 pin as analog

ANSELH = 0; // Configure other AN pins as digital I/O

C1ON\_bit = 0; // Disable comparators

C2ON\_bit = 0;

OPTION\_REG = 0x80; // Set timer TMR0

TMR0 = 0;

INTCON = 0xA0; // Disable interrupt PEIE,INTE,RBIE,T0IE

TRISA = 0x00; // PORTA is output

TRISC = 0x00; // PORTC is output

TRISD = 0x00; // PORTD is output

TRISB = 0x00; // PortB is output

PORTE = 0;

TRISE = 0x01;

PORTA = 0x00;

PORTC = 0;

LCD\_init();

Keypad\_init();

Lcd\_cmd(\_Lcd\_clear);

Lcd\_cmd(\_lcd\_cursor\_off);

do {

name1 = mask(9); // Prepare mask for

name2 = mask(14); // Prepare mask for

Lcd\_out(2,6,codetxt\_to\_ramtxt(txt1[0]));

Lcd\_out(3,6,codetxt\_to\_ramtxt(txt1[1]));

delay\_ms(500);

Lcd\_cmd(\_lcd\_clear);

Lcd\_out(1,4,codetxt\_to\_ramtxt(txt1[2]));

Lcd\_out(3,2,codetxt\_to\_ramtxt(txt1[3]));

Lcd\_out(4,2,codetxt\_to\_ramtxt(txt1[4]));

while(tecla!=1 && tecla!=2)

{

tecla=keypad\_key\_click();

delay\_ms(100);

}

if (tecla==1)

{

POT();

delay\_ms(80);

tecla=0;

}

if (tecla==2)

{

JUEGO();

delay\_ms(80);

tecla=0;

}

}while (1); // Endless loop

}

**Mask.c**

unsigned short mask(unsigned short num)

{

switch (num)

{ // gfedcba

case 0 : return 0b1110111;//a

case 1 : return 0b1111100;//b

case 2 : return 0b0111001;//c

case 3 : return 0b1011110;//d

case 4 : return 0b1111011;//e

case 5 : return 0b1110001;//f

case 6 : return 0b1101111;//g

case 7 : return 0b1110100;//h

case 8 : return 0b0000110;//i

case 9 : return 0b0011110;//j

case 10 : return 0b0111000;//L

case 11 : return 0b1001111;//m

case 12 : return 0b1010100;//n

case 13 : return 0b0111111;//O

case 14 : return 0b1110011;//P

case 15 : return 0b1100111;//q

case 16 : return 0b0110001;//r

case 17 : return 0b1101101;//S

case 18 : return 0b1111000;//t

case 19 : return 0b0111110;//U

case 20 : return 0b0111110;//V

case 21 : return 0b1101110;//y

}

}

1. **Copia impresa del circuito armado en PROTEUS para la simulación en el momento de su ejecución.**

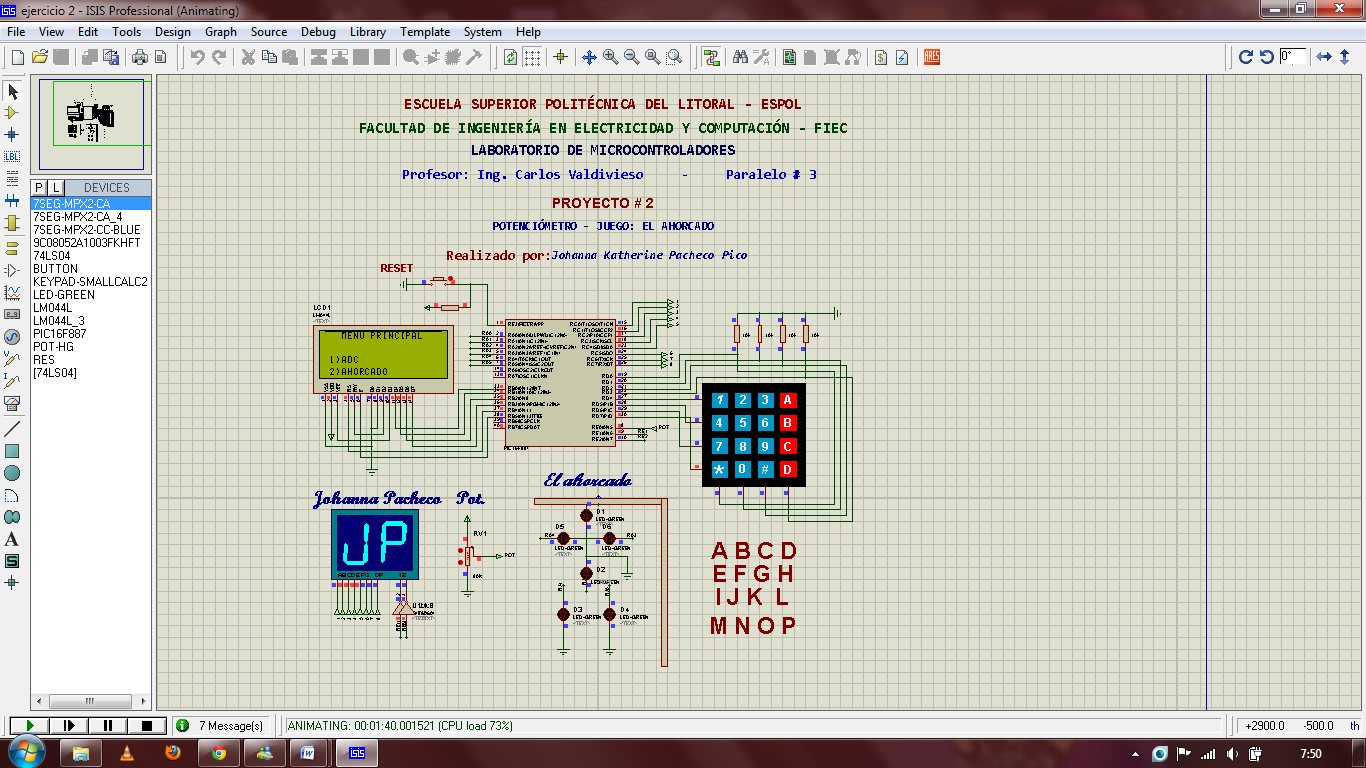
****

Fig. 3.- Muestra el menú principal y las iniciales de mi nombre

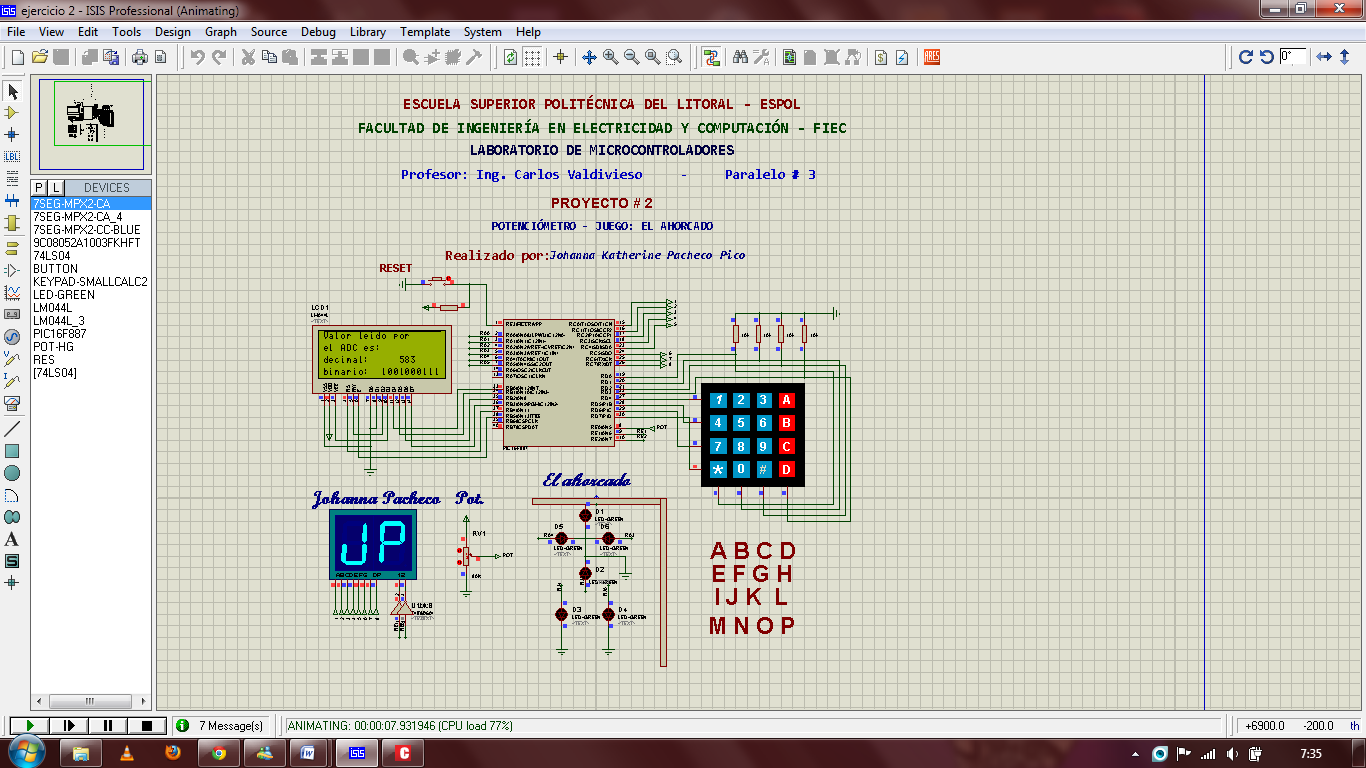
****

Fig. 4.- Se muestra en la LCD los valores del potenciómetro en binario y decimal (bytes).

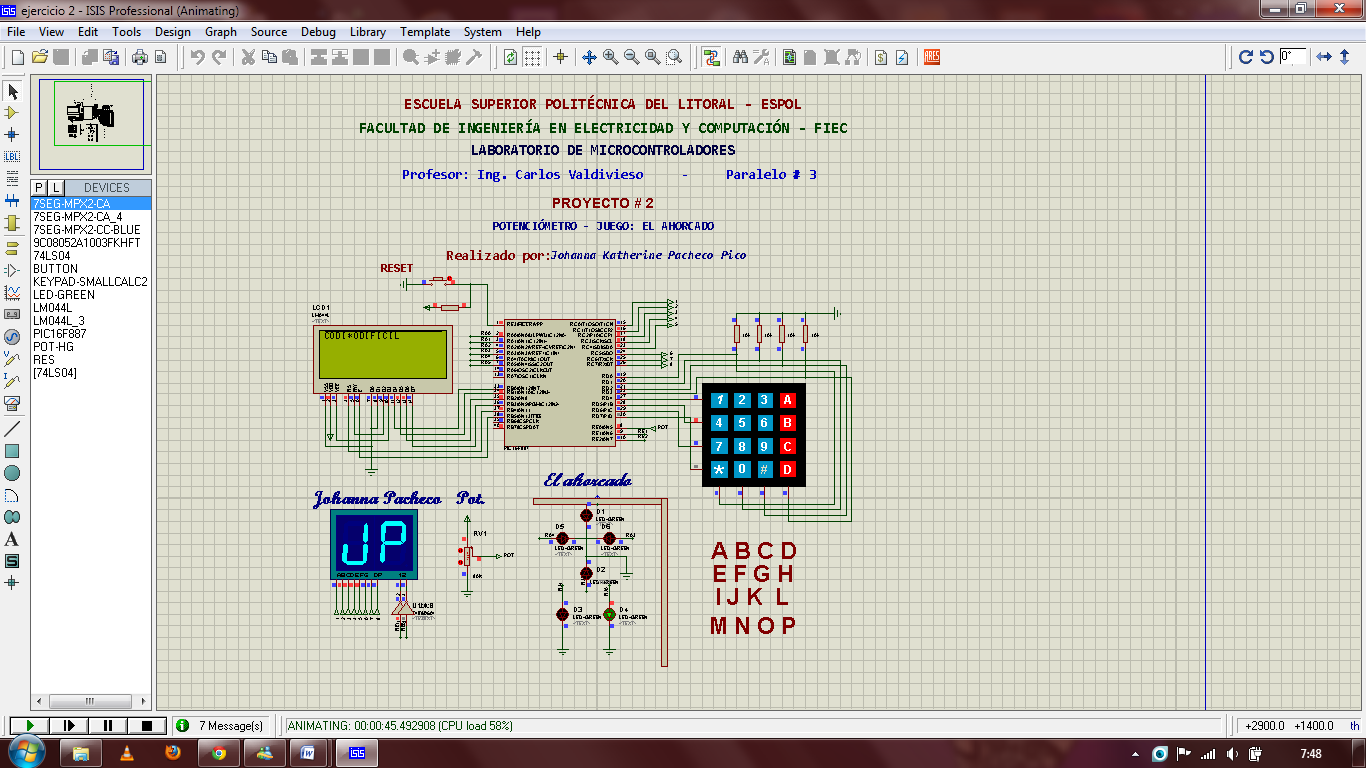


Fig. 5.- Esta es la segunda parte donde se muestra que el usuario está a punto de ganar y que ha tenido una falla.

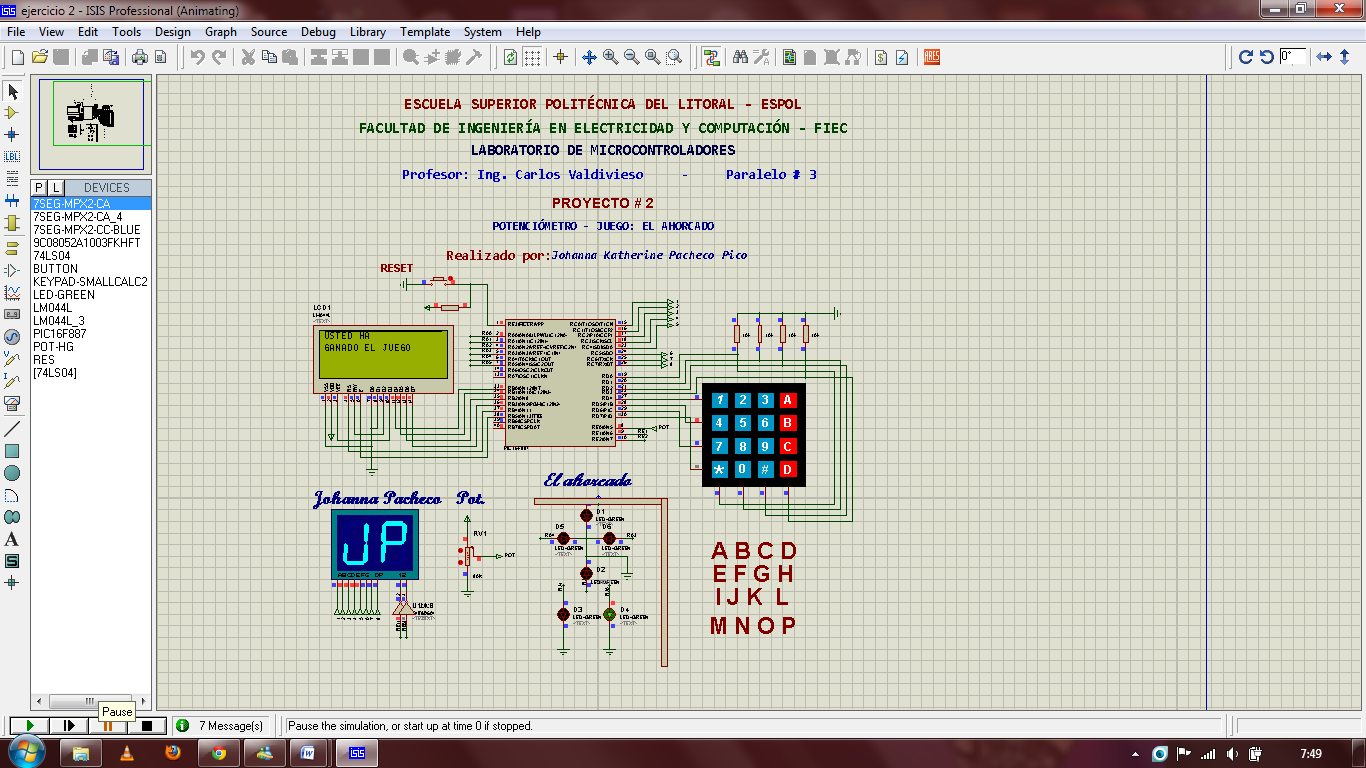
****

Fig. 6.- Se muestra en el LCD que el usuario ha ganado.

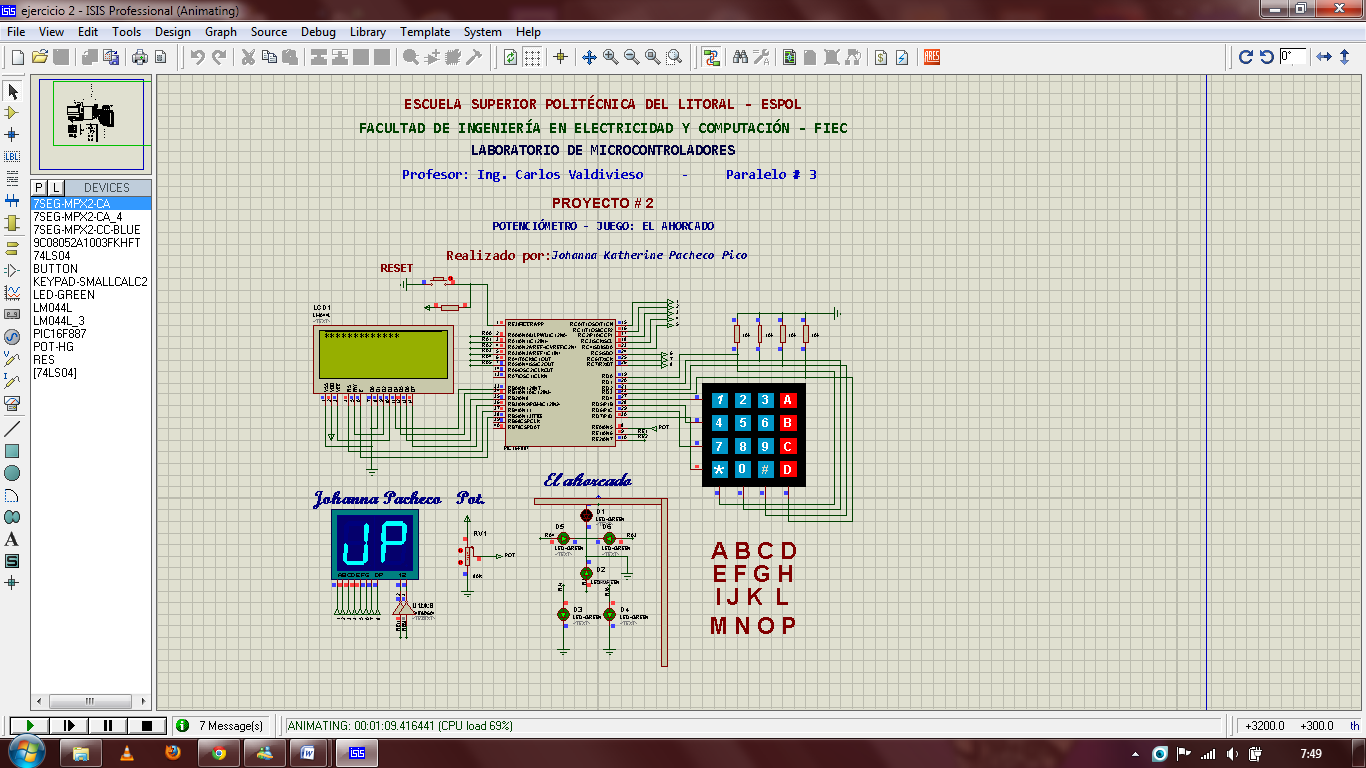
****

Fig. 7.- No se muestra nada en el LCD solo asteriscos por lo que se ve que el jugador va a perder pronto, además tiene un último intento porque ya solo falta un 1 led de encender.

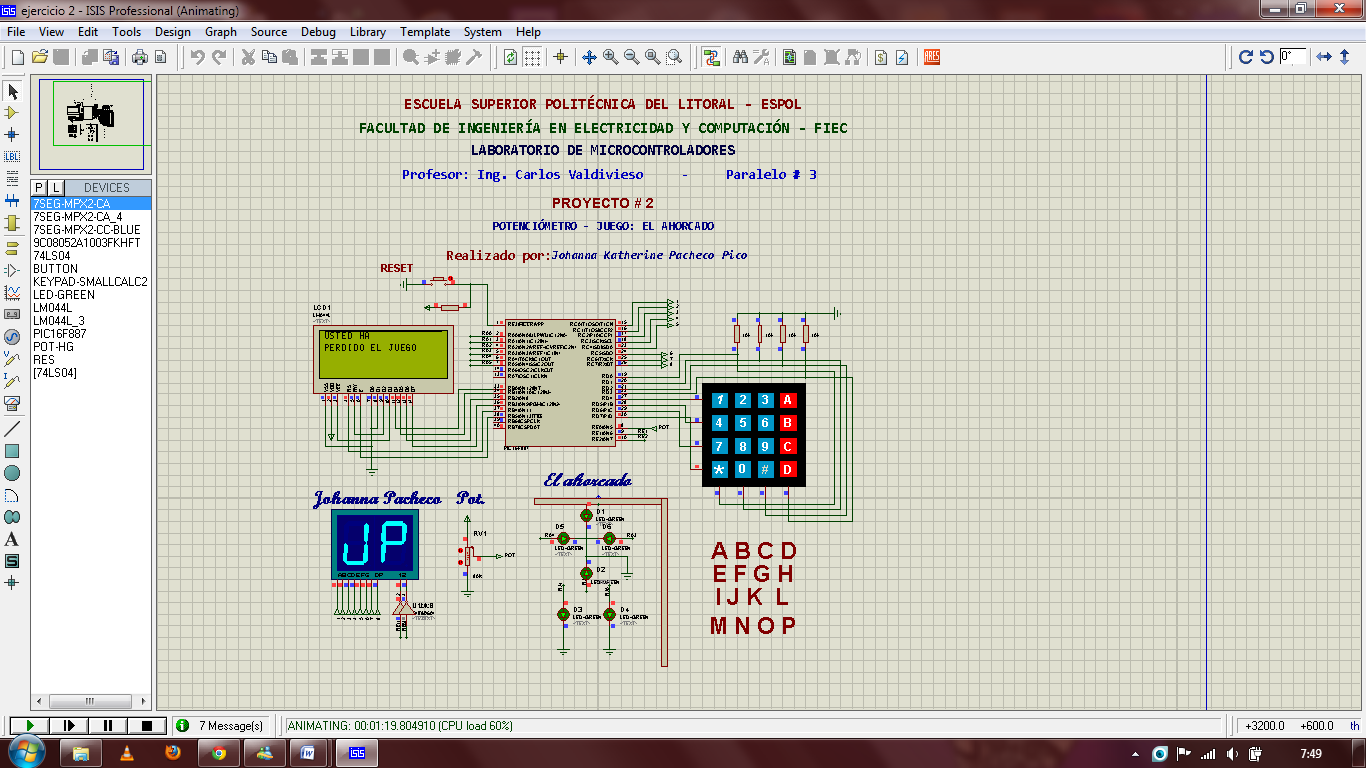
****

Fig. 8.- El usuario perdió el juego.

1. **CONCLUSIONES:**

* Trabajar en un proyecto de esta magnitud me ha permitido aprender tareas que no sabía y a familiarizarme más con el programa simulador Proteus, aprendí que a todos los elementos les puedo cambiar los colores y valores. En este caso experimenté con los colores del teclado 4x4, no le hice cambios a los números de las teclas porque quería los números para el menú.
* Comprendí mejor la utilización y funcionamiento del TMR0 para la presentación de datos en el display multiplexado, configurando en el registro INTCON y en el option\_reg, esto es configurado en la programación. Para la presentación en la LCD se debe poner el cursor off porque si no lo ponemos estará molestando a cada momento.
* Para corregir el problema de muestreo de datos basura en la LCD, tuve que implementar una función que me permite manejar punteros con el objetivo de optimizar la presentación de datos en la LCD, en esta implementación tenemos un arreglo que accede a la posición de memoria de cada texto, y como los textos también los manejo como arreglo cuando llamo a la función entonces me muestra los caracteres que necesito y no me sobrelapa la información y tampoco me muestra datos extraños como chinos.

1. **Recomendaciones:**

* Tenemos que tener en cuenta que cuando declaremos las variables y funciones, en mikroC Pro estas tienen que ser únicas porque si se repiten entonces me saldrán errores en a programación.
* Al crear un proyecto en mikroc pro tenemos que establecer una misma frecuencia tanto en este programador como en el simulador, si no lo hacemos así entonces no funcionará de manera correcta.
* Si optamos por usar el oscilador interno del PIC, también hay que configurarlo en mikroc pro en el edit\_project y escojo el INTOSCIO.