**OBJETIVOS**

* Comprobar conocimientos obtenidos en el desarrollo final del curso bajo el entorno de Micro C for PIC.
* Verificar destrezas en la implementación de los elementos en la tarjeta PCB.
* Utilizar el PIC16F887 como controlador que conmutará la Lectura del valor analógico de un Potenciómetro y un juego de Tennis.

**DESCRIPCION DEL PROYECTO**

**Este proyecto tiene como primera parte la visualización digital en 2 Displays del valor analógico ajustado mediante un potenciómetro.**

**La segunda parte se basa en un juego implementado que utilizará los recursos de la PCB que tiene como tema la emulación de un juego de Tennis.**

Se programará la función ADC\_Read para la conversión del valor analógico configurando al PUERTO A con entrada analógica en el PIN AN0, este valor será visualizado en los Displays luego de pasar por las funciones ***ver\_decenas*** y ***ver\_unidad*** que toma el valor de la función ADC\_Read y lo codifica para mostrar digitalmente el valor en los Displays y también se podrá verificar este cambio a través de los leds.

Se usará procedimientos de lazos WHILE e IF para controlar la conmutación de la lectura del valor del potenciómetro y el juego de tennis cada vez que el usuario lo requiera, la ejecución inicia al presionar las botoneras colocadas en RA1 o RA2 cada cambio de estado será monitoreado mediante la función RA#\_bit. Al presionar la botonera en RA2 se mostrará el valor del potenciómetro, cuando se desee pulsar la botonera en RA1 se empezará a jugar.

Cuando se ejecute el juego de tennis se debe esperar hasta que se muestren en los Displays los marcadores de cada jugador (ambos encerados debido a que empieza la partida) luego se pregunta mediante IF(RA#\_bit) por estado de la entrada colocada en RA4 al ser presionada se produce una secuencia implementada mediante un lazo FOR que emulará el recorrido de la pelota hasta llegar al otro extremo en donde se preguntará si la tecla puesta en RA3 se ha aplastado de ser así la pelota regresa al otro lado. Cuando la pelota visualizada a manera de una secuencia de un led encendido cada vez que avanza llegue al siguiente extremo al jugador que presionó la última vez su respectiva pulsadora se le adjudicará un punto a su marcador y este a su vez tiene el derecho de sacar, se mostrará en los Displays el marcador de cada jugador previo al nuevo saque.

Al llegar el conteo a 6 de cualquier jugador se dará por finalizado el ciclo While y mostrará el resultado final del juego de tennis e indicara cual jugador fue el ganador.

**DIAGRAMA DE BLOQUES**

**ENTRADAS**

BOTONERAS

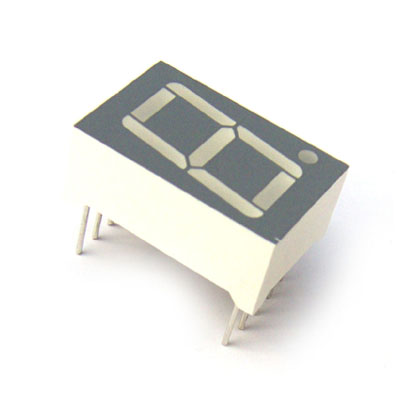
POTENCIÓMETRO

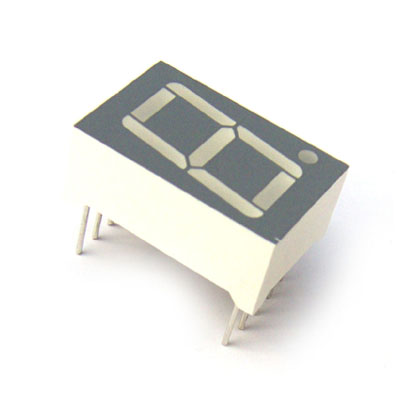
**MICROCONTROLADOR**

PIC16F887

AJUSTE VALOR ANALÓGICO JUEGO DE TENNIS

**SALIDAS**

****

****

****

Valor Digital

Marcador de juego

Ganador de Tennis

Pulsadores

**DIAGRAMA DE FLUJO PROGRAMA PRINCIPAL**

INICIO

temp\_res, i1, i2, G1, G2

Entrada Digital: PORTA

Excepto AN0 Analógico

Salidas Digitales: PORTB, PORTC, PORTD

WHILE

NO

PORTC = ~0x0E;

PORTD = ~0x46;

NO

SI

RA2\_bit

RA1\_bit

SI

SI

load()

(G1orG2)>5

WHILE

NO

WHILE

temp\_res=ADC\_Read(0)

RA3\_bit

SI

show\_result(G1,G2)

PORTC=ver\_unidad(temp\_res)

PORTD=ver\_decena(temp\_res)

NO

RA1\_bit

FOR

RA4\_bit

SI

NO

G2++

RA4\_bit

FOR

NO

SI

RA2\_bit

RA3\_bit

G1++

**DIAGRAMA DE FLUJO**

**SUBRUTINAS**

ver\_decena()

Ver\_unidad()

temp\_res codificado a 7 segmentos

temp\_res codificado a 7 segmentos

PORTD=decena

PORTC= unidad PORTB =temp\_res>>2

Return

Return

show\_result()

PORTD = “J” PORTC = “1” PORTD = “0” PORTC = G1 PORTD = “J” PORTC = “2” PORTD = “0” PORTC = G2

Ganador “J1”

SI

SI

G1>G2

(G1orG2)>5

NO

NO

Ganador “J2”

Return

**DESCRIPCIÓN DEL ALGORITMO**

Incluir funciones y máscaras:

***ver\_unidad.c,***codifica el valor de salida de la función ADC\_Read para visualizar la unidad correspondiente al valor ajustado por el potenciómetro.

***ver\_decena.c,*** codifica el valor de salida de la función ADC\_Read visualizar la decena correspondiente al valor ajustado por el potenciómetro.

***win1.c,*** *muestra mediante codificación de 7 segmentos que el jugador número ha sido el victorioso al finalizar la partida.*

***win2.c,*** *muestra mediante codificación de 7 segmentos que el jugador número ha sido el victorioso al finalizar la partida.*

Definir las variables:

***unsigned int temp\_res,*** la cual servirá para guardar el valor convertido de analógico a digital por la función ADC\_Read(0).

***unsigned short digit1,*** recepta el valor codificado por la máscara *ver\_decena*.

***unsigned short digit,*** recepta el valor codificado por la máscara *ver\_unidad.*

***unsigned short i1,*** itera el encendido del led que simula el recorrido hacia un extremo.

***unsigned short i2,*** itera el encendido del led que simula el recorrido hacia un extremo.

***unsigned short G1***, almacena el puntaje para el jugador 1 en el partido de Tennis.

***unsigned short G2,*** almacena el puntaje para el jugador 1 en el partido de Tennis.

Colocar puerto A como entrada digital, excepto el pin AN0 que se configura como entrada analógica.

Colocar puerto B, C, D como salida digital.

Se espera por el pulso generado en la botonera colocada en el bit1 del puerto A o en RA2 para el arranque del proyecto caso contrario se visualiza en los Displays “C” “F”.

Dentro de un lazo While se monitoreará el estado de las entradas RA1 y RA2.

Si la botonera en RA2 es presionada se observa en los Displays y mediante el encendido de los leds el valor actual que entrega el potenciómetro; esto se logra codificando el valor de la función ADC\_Read a 7 segmentos a través de las funciones ver\_unidad y ver\_decena.

Si la botonera que es presionad es la de RA1 se ejecutará el juego, simulando mediante leds “loading” donde se encienden led de uno a uno programando iteraciones en un lazo for.

Dentro de esta parte del proyecto se controla mediante un lazo While el estado de las 2 últimas botoneras, es decir las colocadas en RA3 y en RA4.

Luego de emular la carga del juego se muestran en los Displays los marcadores iniciales, después se espera por el pulso enviado por la tecla en RA3 que inicia el juego y mediante un FOR se logra que el led encendido cambie de lugar hasta llegar al siguiente extremo cuando el led encendido es el último se espera 250 milisegundos y se verifica el estado en la patita RA4, de haber sido presionada el recurrido del led encendido se invierte desplazándose hacia el lado inicial caso contrario, si el bit en RA4 no fue cambiado entonces se incrementa en 1 la variable que guarda el marcador correspondiente al jugador 1 (G1).

Se puede retornar a la visualización del valor del potenciómetro cada vez que se desee, presionando la botonera en RA2.

**Reglas del juego:**

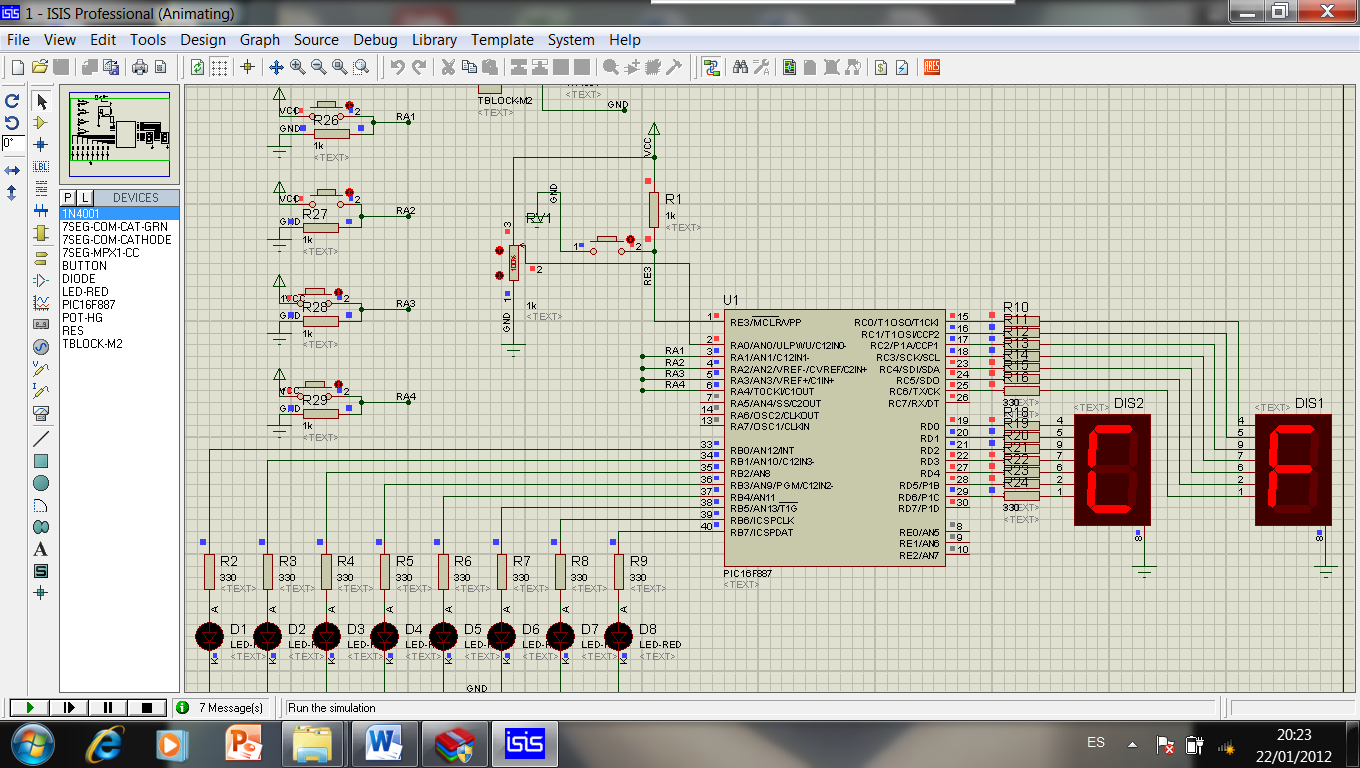
El juego emula el golpe de la pelotita de tennis presionando la botonera perteneciente a cada jugador RA3 para el jugador 1 y RA4 para el jugador 2.

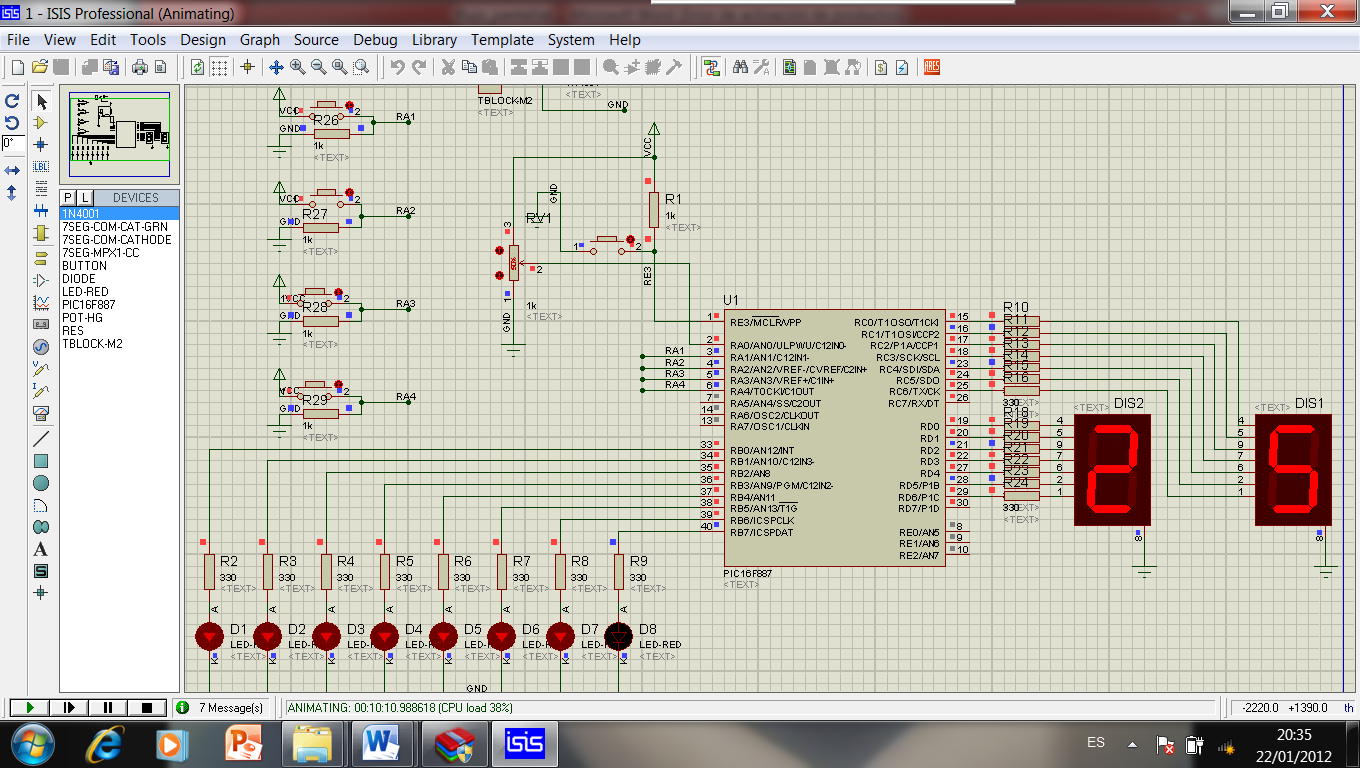
Cada vez que la pelotita llegue al extremo del jugador contrario y este a su vez no pulse su botón el jugador contrario recibirá un punto.

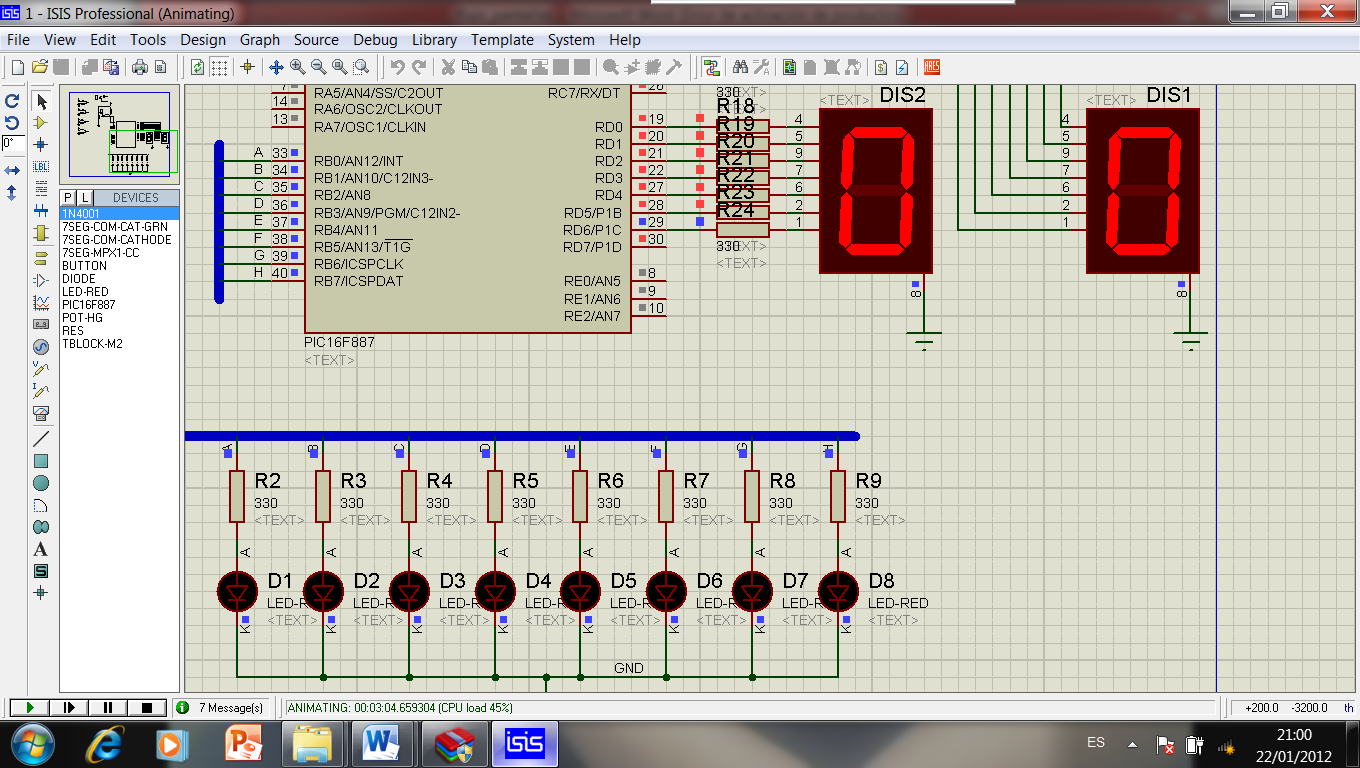
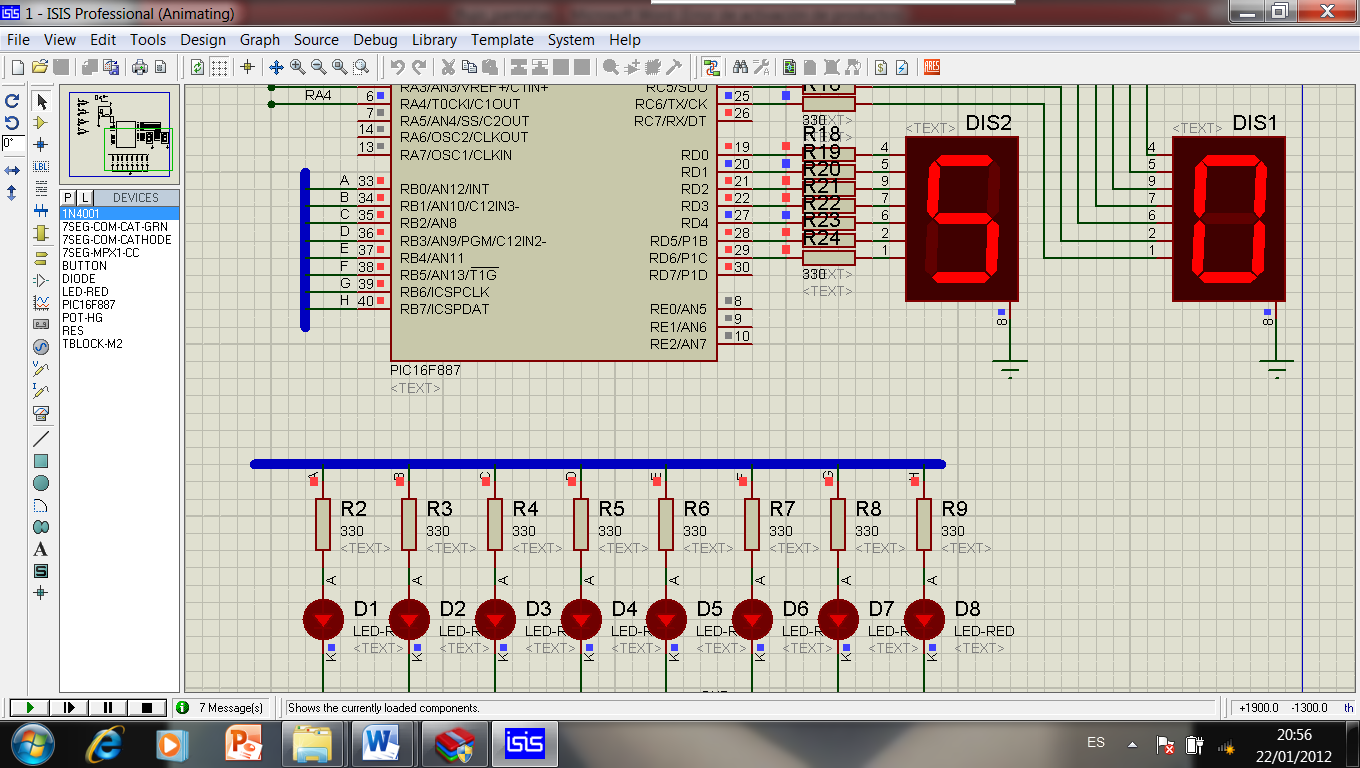
El resultado puede ser observador en los Displays cada vez que algún jugador obtenga un punto y este jugador puede sacar solo después de verse los puntajes.

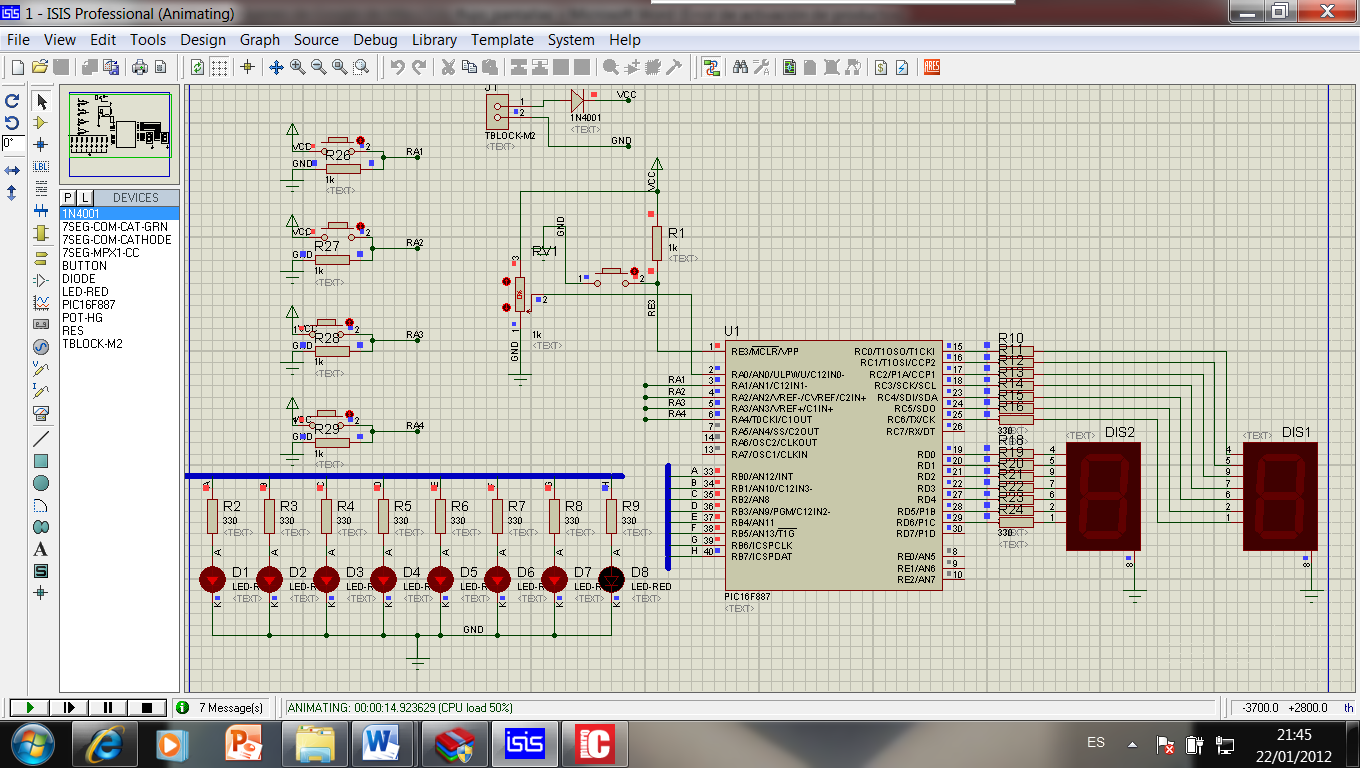
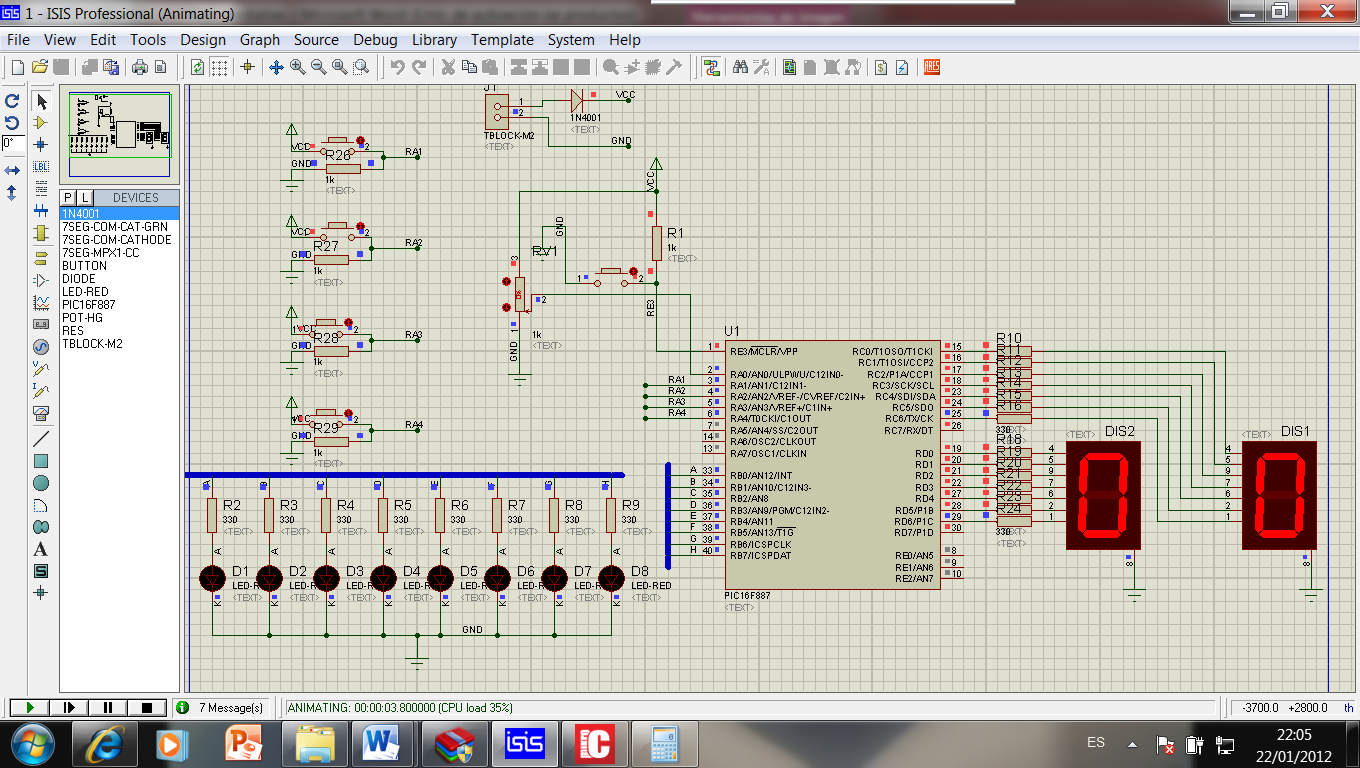
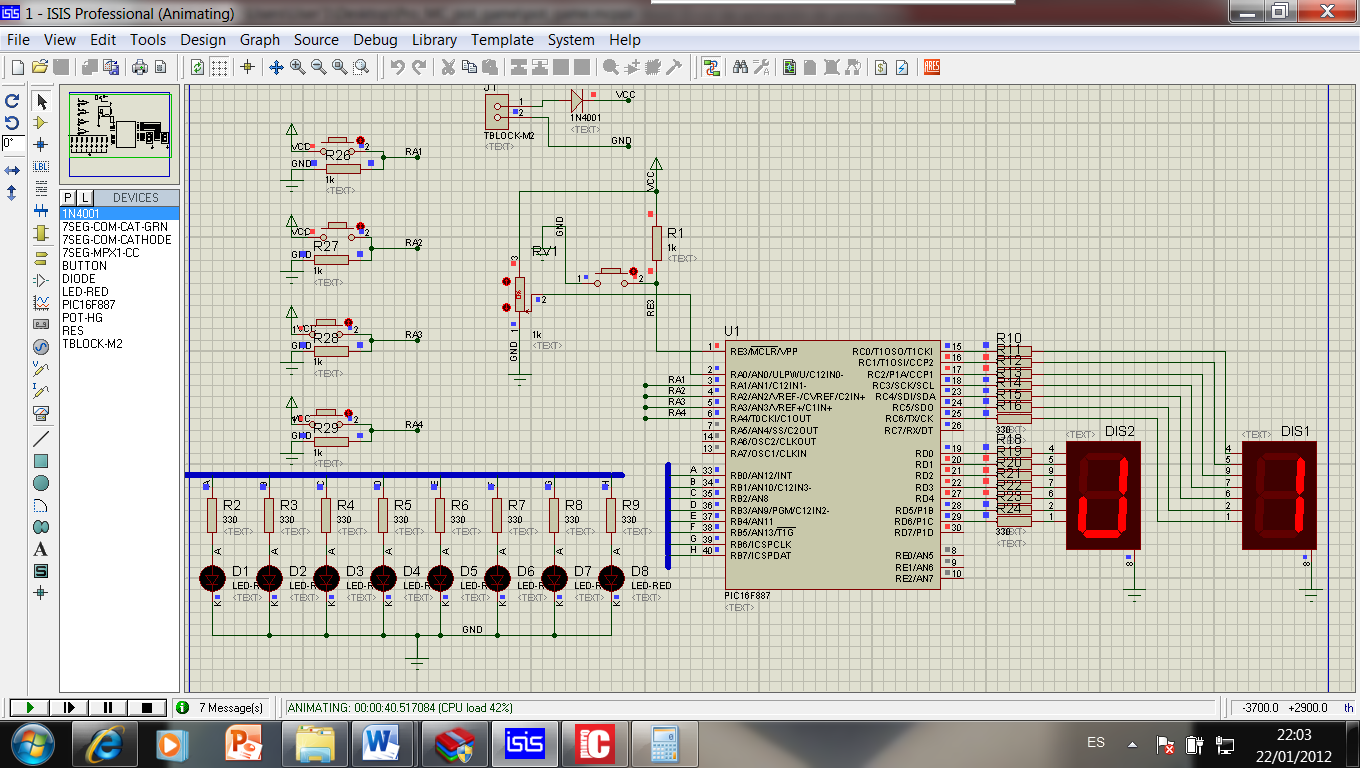
El juego termina cuando cualquier jugador puntúa 6 y el programa retorna donde se verifica el valor del potenciómetro.

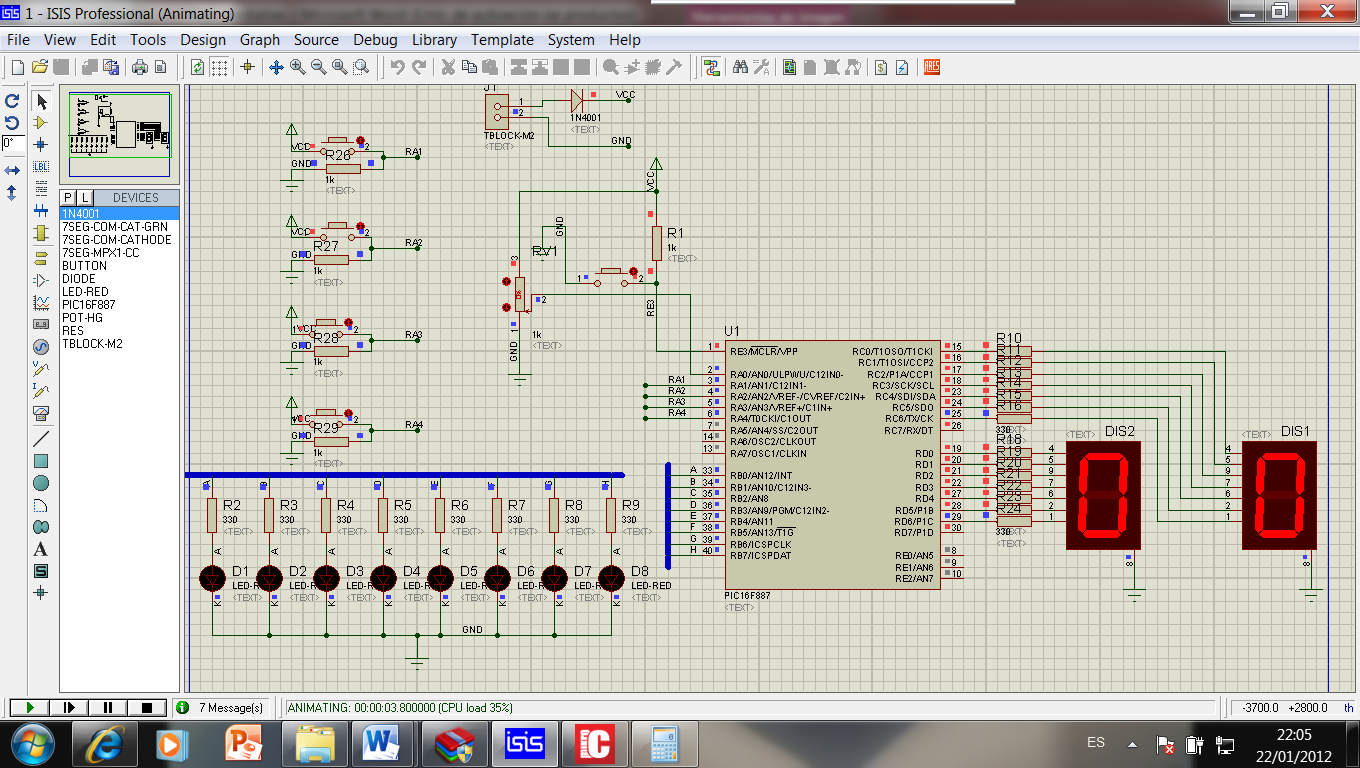
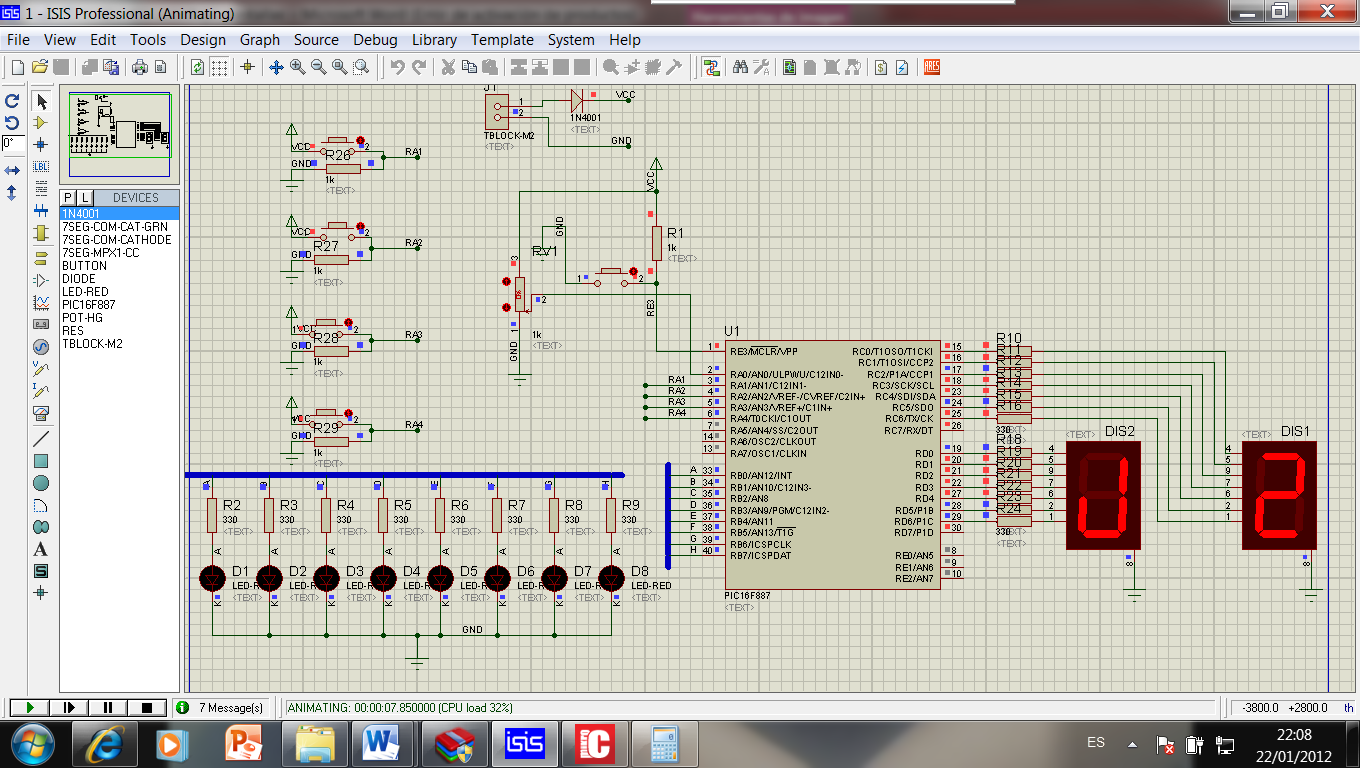
**SIMULACIÓN EN PROTEUS**

Inicialmente se visualiza las iniciales del programador.

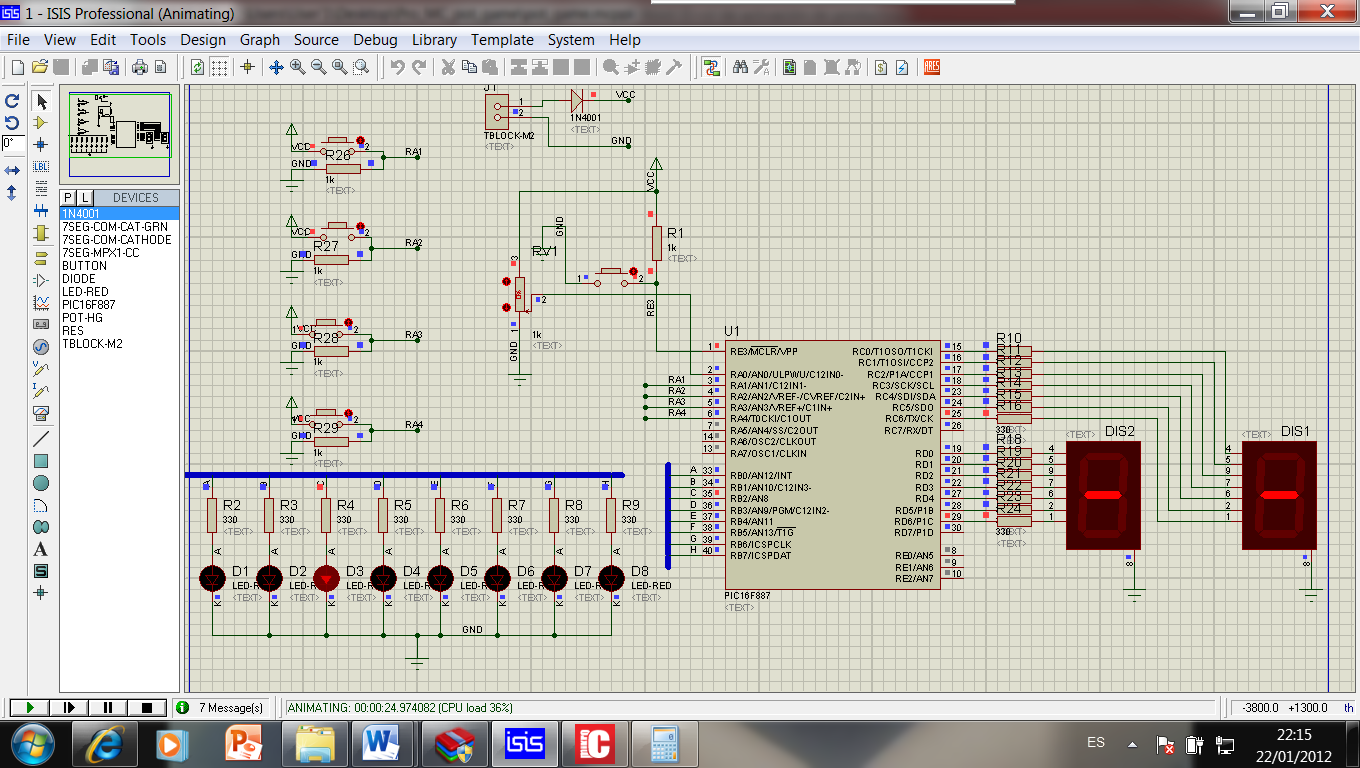
Se requiere presionar cualquiera de las botoneras colocadas en RA1 y RA2 para iniciar el funcionamiento del proyecto. Si se ha presionado la botonera ubicada en RA2 se logra observar la operación de conversión del valor analógico manipulado con el potenciómetro a un valor digital que se muestra en los Displays.

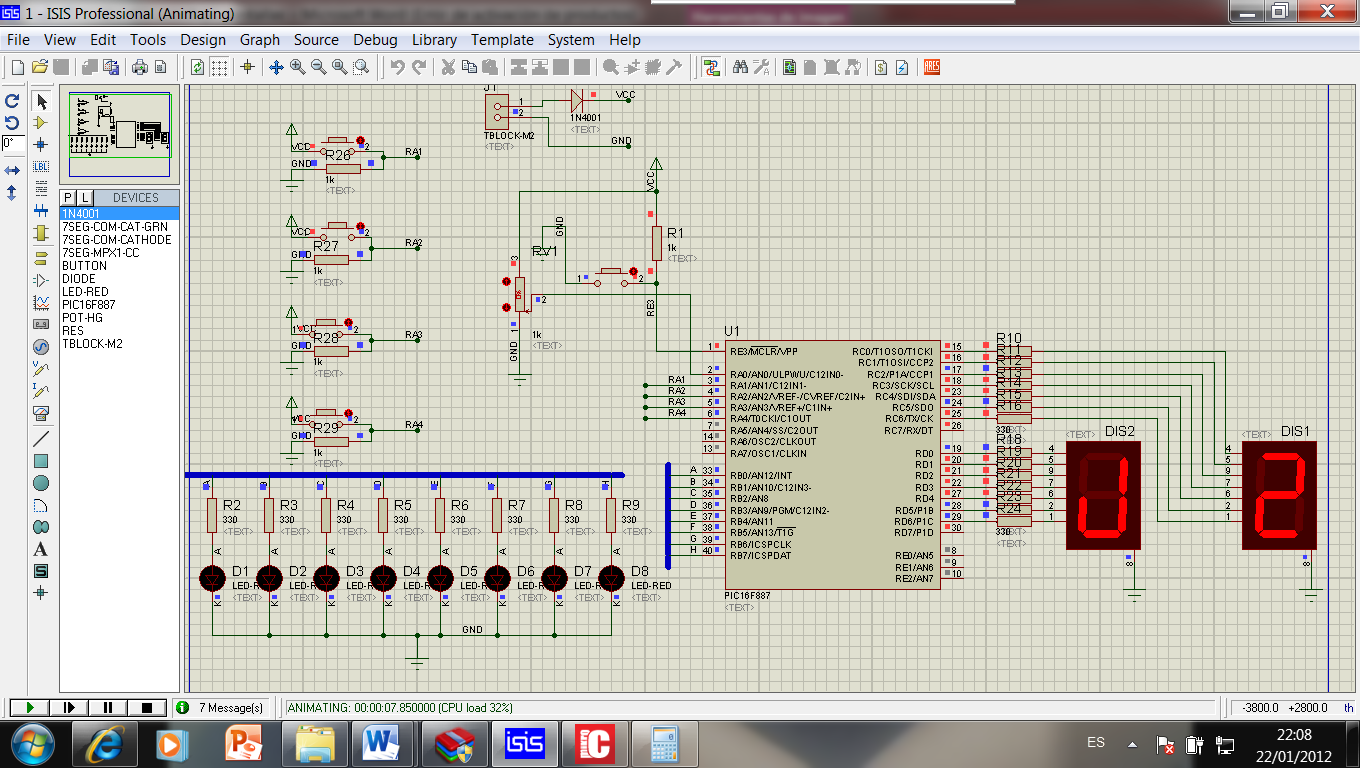
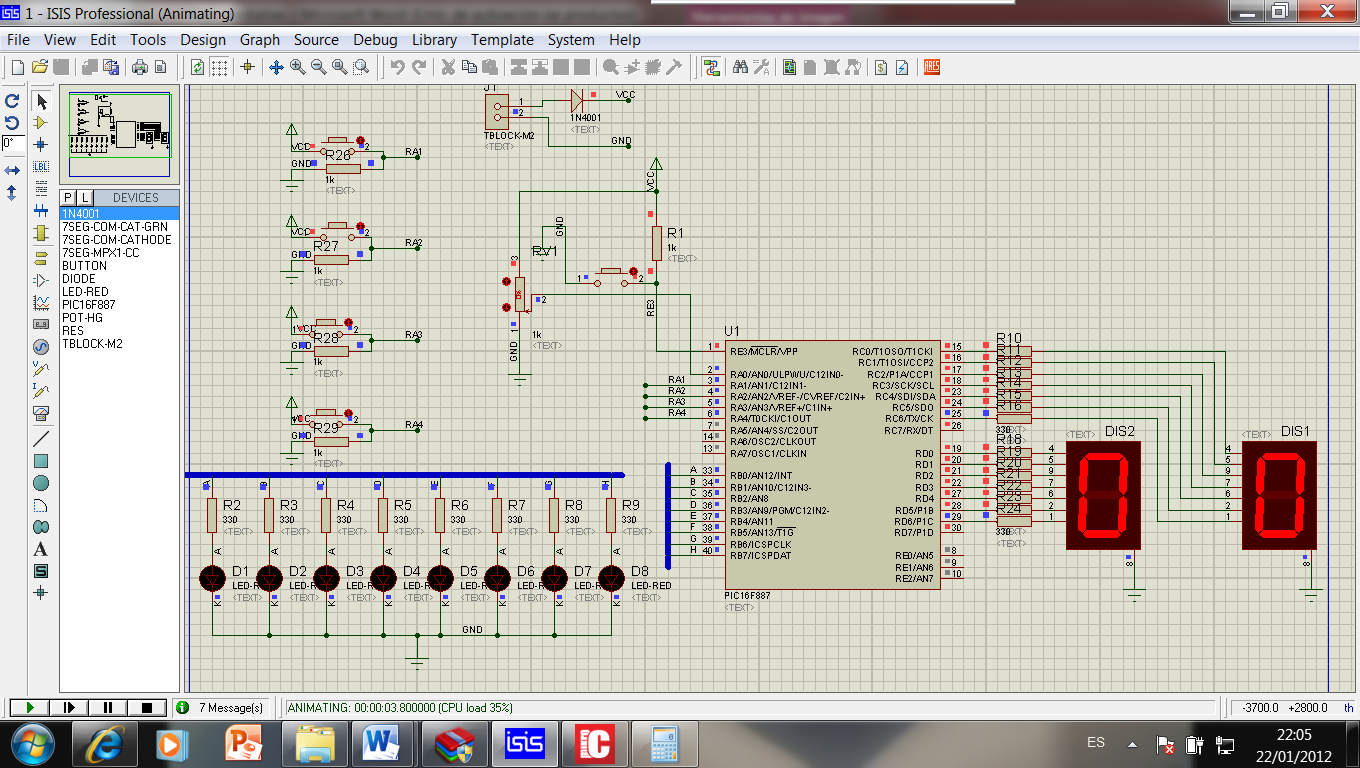
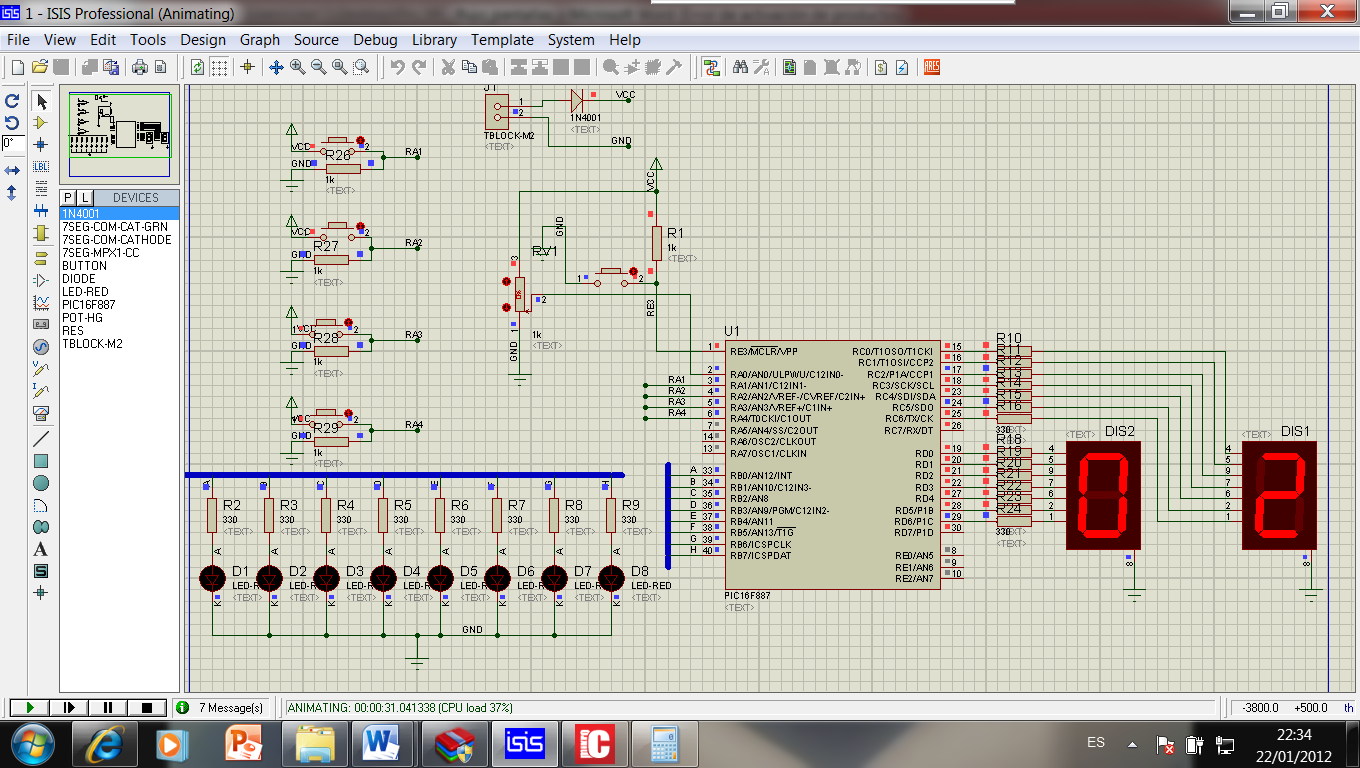
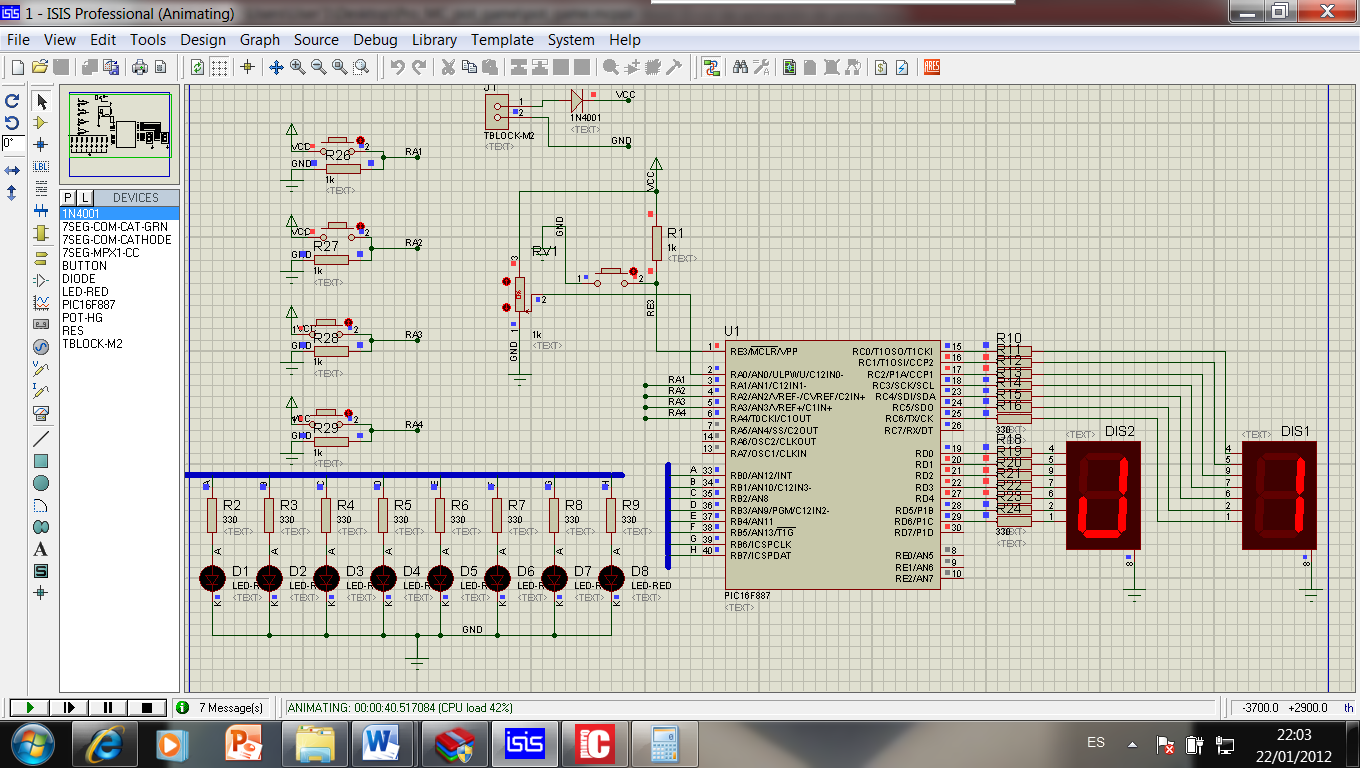
Puede notarse que los límites ajustables tienen como un valor máximo en 50 y mínimo en 0.

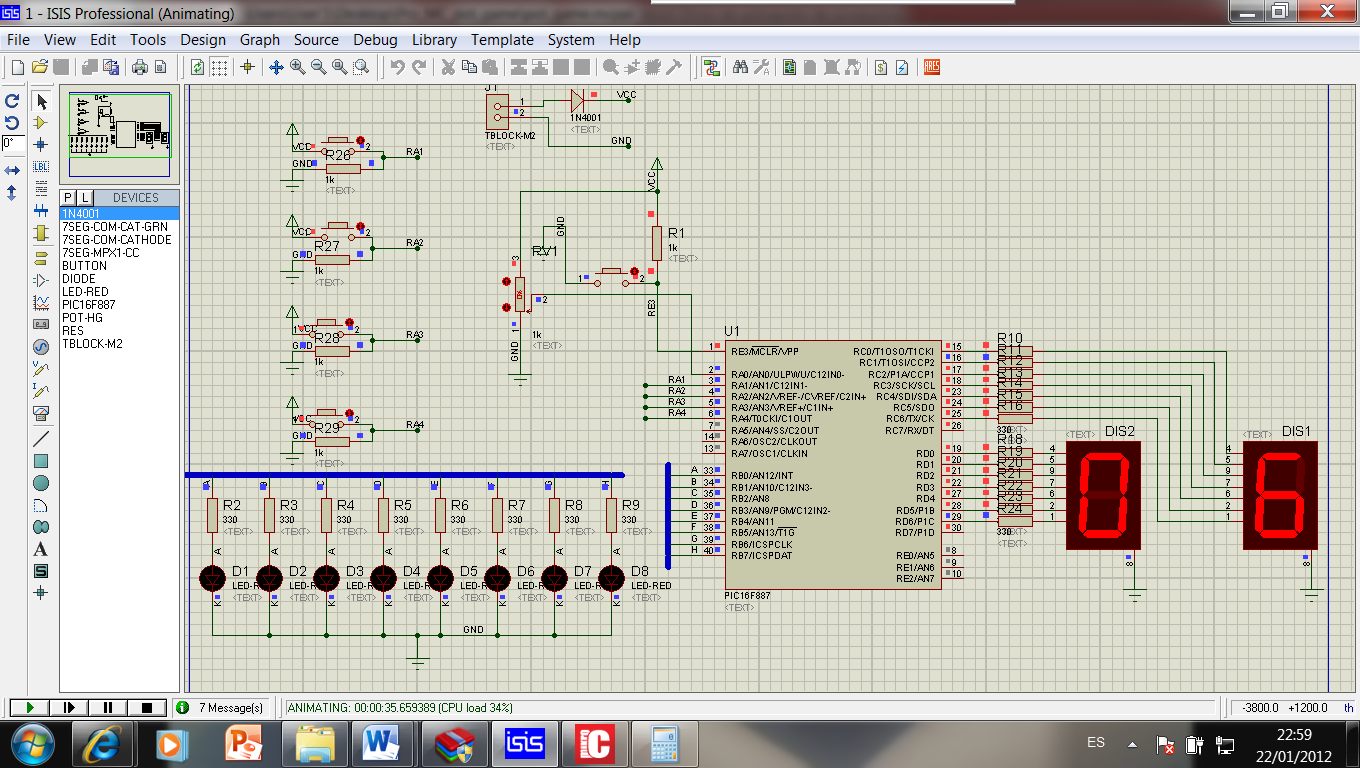
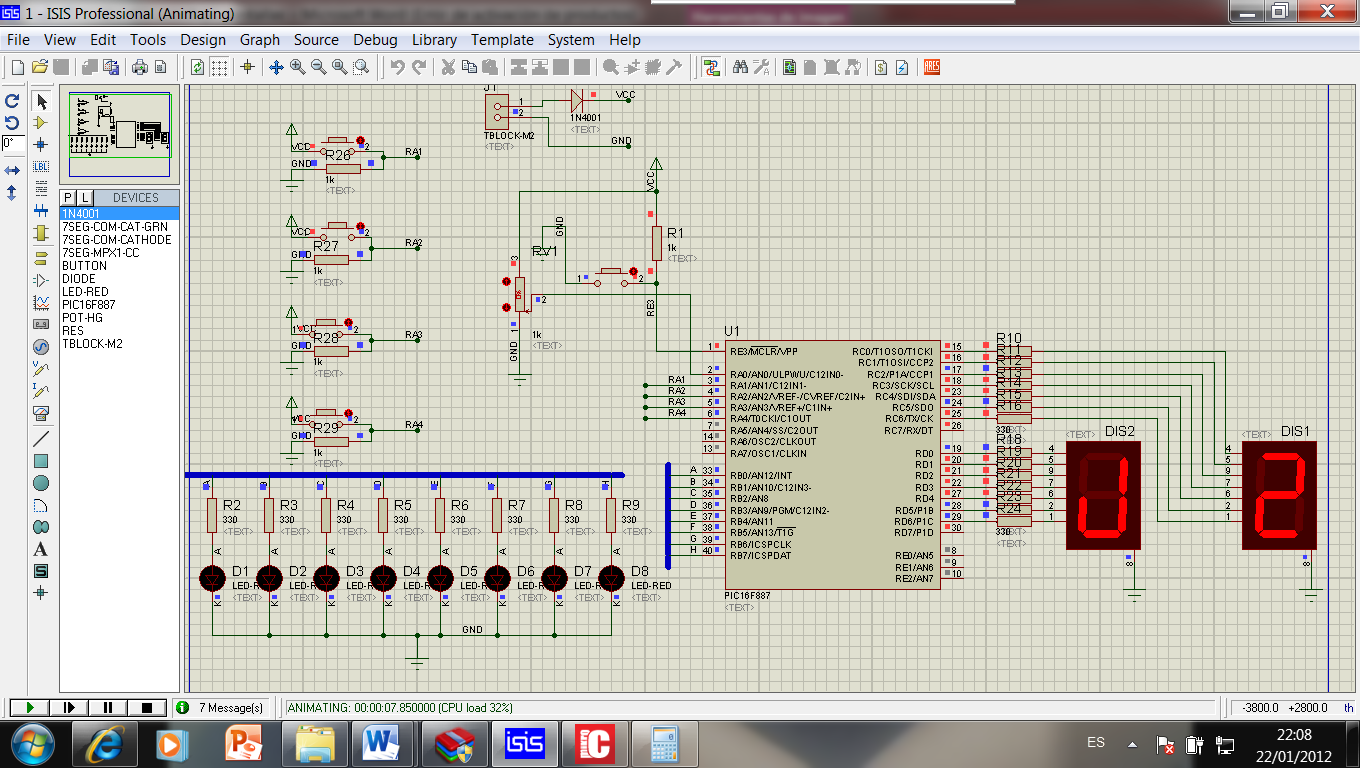
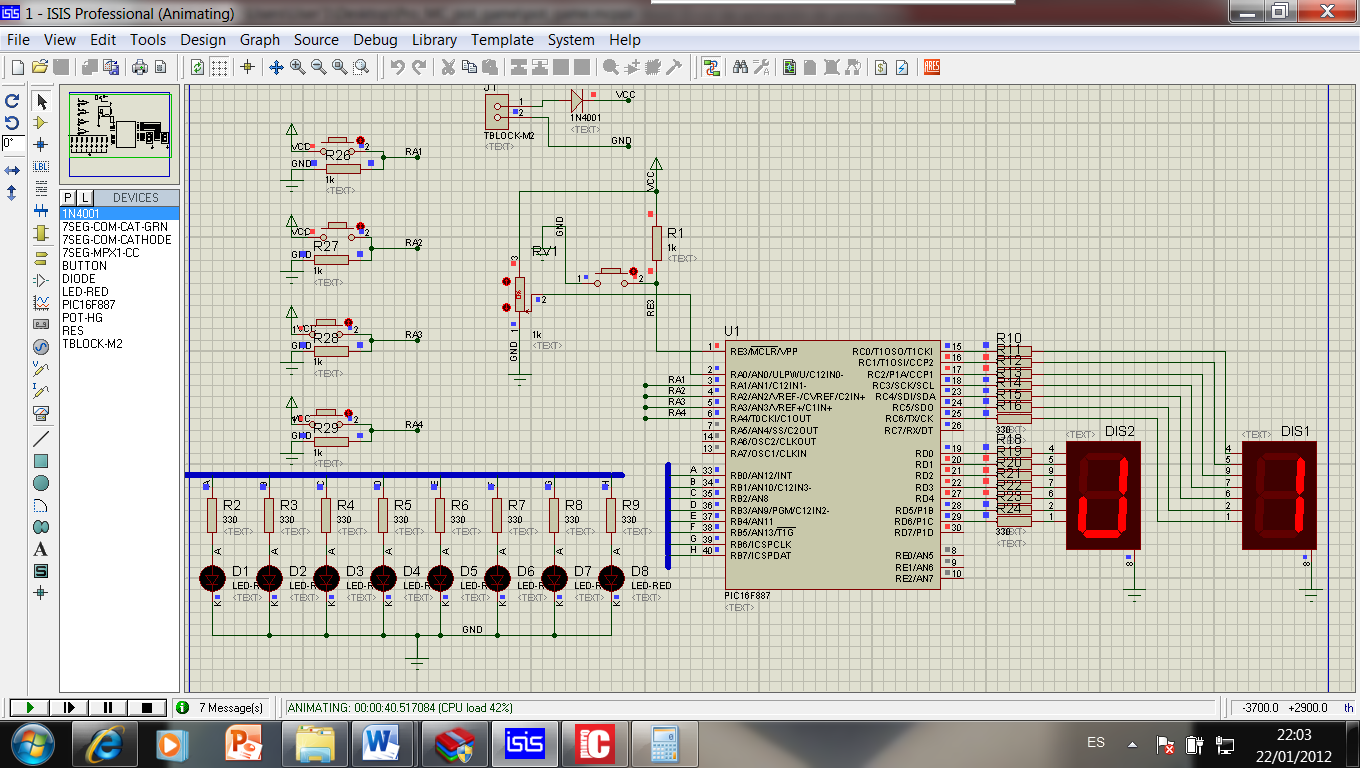
Se conmuta al juego de Tennis presionando la botonera situada en RA1 donde se emula la espera encendiendo los leds de izquierda a derecha para luego mostrar mediante displays el marcador inicial de cada jugador

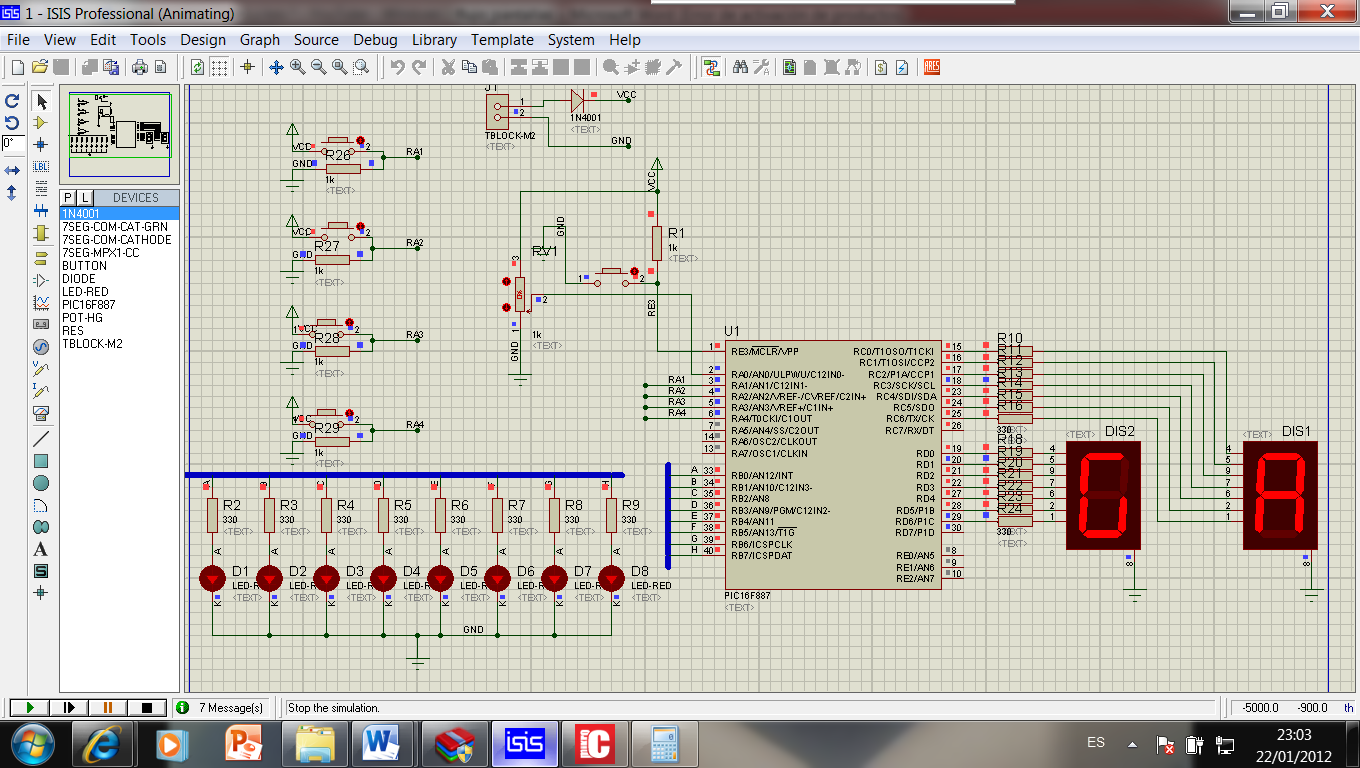


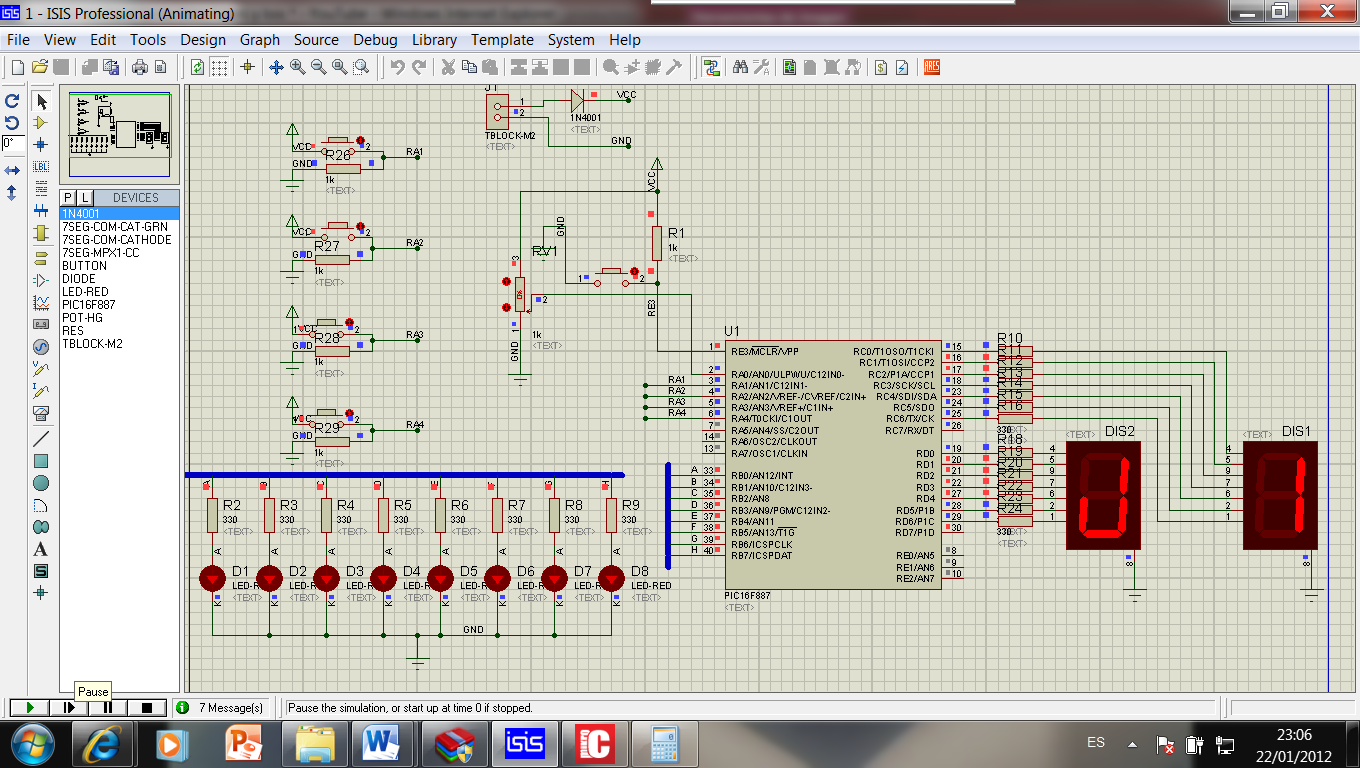
Para comenzar la partida se debe presionar la botonera que se encuentra en RA3 con lo que se genera el encendido de un led que avanza al lado contrario donde se espera que el otro jugador presione la pulsadora que está en la entrada RA4 con lo que se produce un desplazamiento hacia el extremo inicial encendiendo de igual forma un led conforme avanzaría la pelota de Tennis.



Si cualquiera de los jugadores no presiona su botonera que controla el choque que recepta la contestación visualizada al encenderse el ultimo led en su respectivo extremo opuesto, se incrementará el marcador del jugador que golpeó la última vez la pelota(led encendido).

El juego termina cuando cualquiera de los jugadores logra alcanzar primero 6 puntos mostrando cual fue el jugador ganador a través de los Displays.





**PROGRAMA FUENTE EN LENGUAJE C**

#include "ver\_unidad.c"

#include "ver\_decena.c"

#include "win1.c"

#include "win2.c"

unsigned int temp\_res;

unsigned short convertir(unsigned short num2);

unsigned short digit1, digit, i1, i2, G1, G2;

unsigned int num;

void load()

{

PORTD = ~0xFF;

PORTC = ~0xFF;

for(i2=3;i2>0;i2--)

{

for(i1=128;i1>0;i1=i1/2)

{

PORTB = i1;

Delay\_ms(40);

}

PORTB = 0x00;

}

PORTB = 0x00;

PORTD = 0xBF;

PORTC = 0xBF;

}

unsigned short convertir(unsigned short num2)

{

switch (num2) {

case 0 : return 0x01;

case 1 : return 0x02;

case 2 : return 0x04;

case 3 : return 0x08;

case 4 : return 0x10;

case 5 : return 0x20;

case 6 : return 0x40;

case 7 : return 0x80;

}

}

unsigned short convertir2(unsigned short numx)

{

switch (numx) {

case 7 : return 0x01;

case 6 : return 0x02;

case 5 : return 0x04;

case 4 : return 0x08;

case 3 : return 0x10;

case 2 : return 0x20;

case 1 : return 0x40;

case 0 : return 0x80;

}

}

unsigned short Marcador(unsigned short gamer)

{

switch (gamer) {

case 0 : return ~0x40;

case 1 : return ~0xF9;

case 2 : return ~0x24;

case 3 : return ~0x30;

case 4 : return ~0x19;

case 5 : return ~0x12;

case 6 : return ~0x02;

case 7 : return ~0xF8;

case 8 : return ~0x00;

}

}

void show\_result(unsigned short gamer1,unsigned short gamer2)

{

PORTD = ~0x0C;

PORTC = ~0xF9;

Delay\_ms(300);

PORTD = ~0x40;

PORTC = Marcador(gamer1);

Delay\_ms(400);

PORTD = ~0xBF;

PORTC = ~0xBF;

Delay\_ms(250);

PORTD = ~0x0C;

PORTC = ~0x24;

Delay\_ms(300);

PORTD = ~0x40;

PORTC = Marcador(gamer2);

Delay\_ms(400);

PORTD = ~0xFF;

PORTC = ~0xFF;

Delay\_ms(50);

PORTD = ~0xBF;

PORTC = ~0xBF;

}

void main() {

ANSEL = 0x01; // Configure AN2 pin as analog

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

C1ON\_bit = 0; // Disable comparators

C2ON\_bit = 0;

TRISA = 0xFF; // PORTA is input

TRISE = 0xFF; // PORTA is input

TRISC = 0; // All port A pins are configured as outputs

PORTB = 0; // Turn off both displays

TRISB = 0; // All port A pins are configured as outputs

PORTD = 0; // Turn off all display segments

TRISD = 0; // All port D pins are configured as outputs

do {

salir:

PORTC = ~0x0E;

PORTD = ~0x46;

opcion1:

if(RA2\_bit){

Delay\_ms(40);

do {

regresar:

temp\_res = ADC\_Read(0); // Get 10-bit results of AD conversion

PORTB = temp\_res>>2; // Send lower 8 bits to PORTD

num = temp\_res;

digit = ver\_unidad(num);

digit1= ver\_decena(num);

PORTC = digit;

PORTD = digit1;

if(RA1\_bit){

Delay\_ms(40);

goto opcion2;

}

} while(1);

}

opcion2:

if(RA1\_bit){

Delay\_ms(40);

G1=0;

G2=0;

PORTB=0x00;

load();

show\_result(G1,G2);

do {

do{

while(1)

{

if(RA3\_bit){

Delay\_ms(40);

for(i1=0;i1<8;i1++)

{

PORTB = convertir2(i1);

Delay\_ms(250);

}

if(RA4\_bit){

Delay\_ms(40);

break;

}

else{

PORTB = 0x00;

G1++;

show\_result(G1,G2);

}

if((G1>5)||(G2>5))

{

if(G1>G2)

{

gano1:

win1();

}

else

{

gano2:

win2();

}

goto regresar;

}

}

if(RA2\_bit){

Delay\_ms(40);

goto opcion1;

}

if(RE3\_bit){

Delay\_ms(40);

goto salir;

}

}

while(1)

{

if(RA4\_bit){

Delay\_ms(40);

for(i2=0;i2<8;i2++)

{

PORTB = convertir(i2);

Delay\_ms(250);

}

if(RA3\_bit){

Delay\_ms(40);

break;

}

else{

PORTB = 0x00;

G2++;

show\_result(G1,G2);

}

if((G1>5)||(G2>5))

{

if(G1>G2)

{

goto gano1;

}

else

{

goto gano2;

}

goto regresar;

}

}

if(RA2\_bit){

Delay\_ms(40);

goto opcion1;

}

if(RE3\_bit){

Delay\_ms(40);

goto salir;

}

}

} while(1);

if(RA2\_bit){

Delay\_ms(40);

goto opcion1;

}

} while(1);

}

} while(1);

}

**CONCLUSIONES**

Al emplear cada conocimiento asimilado en el transcurso final del Laboratorio se consiguió programar el controlador del proyecto en Lenguaje C y del mismo modo se pudo constatar el sencillo ambiente que provee el Software Micro C for PIC debido a que describe de manera detallada como se debe utilizar cada una de sus librerías y comandos reforzando las enseñanzas comprendidas.

El implementar los elementos en la PCB proporcionó habilidades para solucionar inconvenientes que se presenten en el proceso de soldado como lo son generación de cortos que impidan el buen desempeño del controlador siendo esta destreza fundamental en el campo laboral.

Al realizar este proyecto y en base al desarrollo de las prácticas en el Laboratorio se verificaron las excelentes características que posee el PIC16F887 para controlar un sin número de tareas indicando la gran herramienta que se ha suministrado a nosotros los estudiante y que puede ser utilizada en la vida real.

**RECOMENDACIONES**

Se aconseja ir probando el funcionamiento deseado paso a paso del programa en lenguaje C en proteus para disminuir el tiempo de detección de errores en la lógica del programa.

Se debería tener sumo cuidado al momento de soldar los elementos en la placa para evitar posibles cortos e inconvenientes que retarden la consecución del proyecto.

Se recomienda considerar el tamaño de la memoria que contiene al programa controlador y demás subprogramas ya que esta tiene un límite y al excederlo generará errores y afectará al funcionamiento del proyecto.