***ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL***



***FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN***

***“FIEC”***

LABORATORIO DE MICROCONTROLADORES

**PROYECTO # 2 EN LENGUAJE C CON MIKROC PRO**

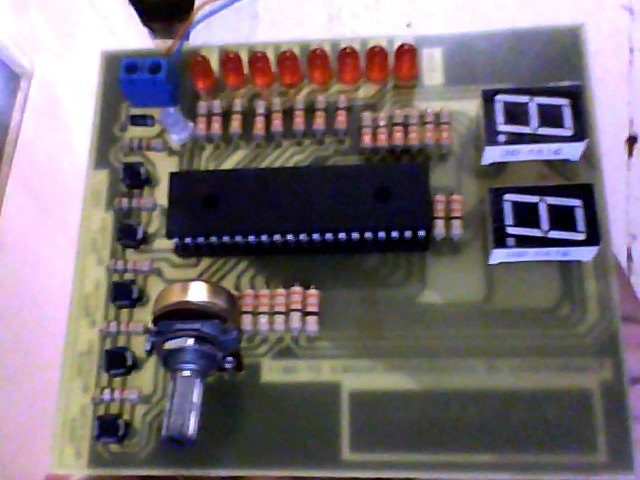
**TEMA:**

Lectura de valor analógico de un potenciómetro y juego del “AGARRA EL BIT”.

**PROFESOR:**

Ing. Carlos Valdivieso

**NOMBRE:**

****Irving Garófalo Yánez



**PARALELO:**

6

**GRUPO:**

#5

**FECHA DE PRESENTACIÓN:**

26- 01 – 12

1. **ENUNCIADO DEL PROYECTO**

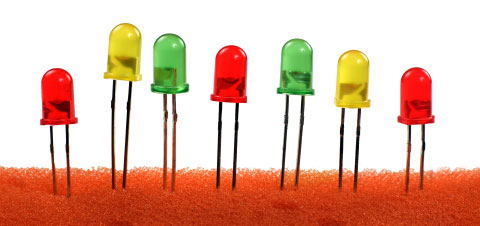
El siguiente proyecto se presentan 2 ejercicios, el primero consiste en la lectura de los valores analógicos de un potenciómetro colocado en la bornera de entrada analógica, presentando las variaciones de valores (en binario) en 8 LEDs y en dos displays de 7 segmentos (en dos dígitos decimales) disponibles.

El segundo ejercicio es un juego llamado “agarra el bit” que consiste en iniciar el movimiento del encendido de un led a la vez de un lado a otro, al momento de que el bit se encienda en uno de los dos extremos de los 8 LEDs y al mismo tiempo presionar una botonera como resultado se ganara un punto. El juego terminará cuando el jugador complete 10 puntos y luego este juego se resetea automáticamente.

El programa está hecho en lenguaje C con el programa MIKRO PRO for PIC, en el cual se usó retardos de tiempo con ayuda de la función Delay\_ms(), además el convertidor ADC para realizar el ejercicio propuesto por el profesor.

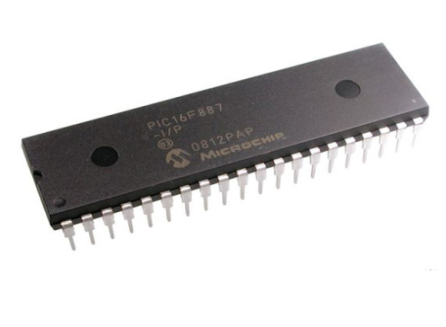
1. **DIAGRAMA DE BLOQUES.**

8 LEDS

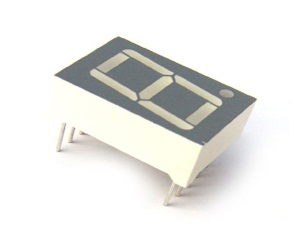


Botoneras

Microcontrolador



Potenciómetro



Displays 7 seg.

1. **Diagrama de flujo funcional del programa principal (nivel alto).**







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

* Setear los parámetros y seleccionar los puertos a usar.
* Definir puertos como salidas o entradas.
* Inicializar variables.
* Iniciar el ADC.
* Si se selecciona botón RA1 se presenta el ejercicio 1 que consiste en la lectura de los valores analógicos de un potenciómetro colocado en la bornera de entrada analógica.
* Presenta las variaciones de valores (en binario) en 8 LEDs y en dos displays de 7 segmentos (en dos dígitos decimales) disponibles.
* Si se selecciona botón RA2 se presenta el ejercicio 2 que consiste en un juego llamado “agarra el bit” que consiste en iniciar el movimiento del encendido de un led a la vez de un lado a otro con retardo de 100ms.
* Al momento de que el bit se encienda en uno de los dos extremos de los 8 LEDs y al mismo tiempo presionar una botonera como resultado se ganara un punto.
* El puntaje se presentará en los displays.
* El juego terminará cuando el jugador complete 10 puntos.
* El juego se resetea automáticamente al terminar de jugar.

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

***/\****

***\* Nombre del Proyecto:***

***proyecto2MICROCON.c***

***\* Nombre del Autor:***

***Irving Darly Garofalo Yanez, 2012.***

***\* Test configuration:***

***MCU: PIC16F887***

***Oscillator: HS, 08.0000 MHz***

***SW: mikroC PRO for PIC***

***\*/***

***unsigned int conversion(unsigned int num);***

***unsigned int digmassig, dig1, digmenossig, pot\_analg, displaymenossig, displaymassig;***

***unsigned int i,com,dig\_com,dig\_1,dig\_2,display\_me,display\_ma;***

***void ejercicio1();***

***void ejercicio2();***

***void display\_dig();***

***void main() {***

***ANSEL = 0;***

***ANSELH = 0;***

***C1ON\_bit = 0; // Disable comparators***

***C2ON\_bit = 0;***

***TRISA = 0xFF; // PORTA is input***

***TRISB = 0; // PORTB como salida***

***TRISC = 0; // PORTC is output***

***TRISD = 0; // PORTD is output***

***PORTB = 0;***

***ADC\_Init();***

***do {***

***PORTD = 0b00000110;***

***PORTC = 0b00111101;***

***if (RA1\_bit) { // button on RA0 pressed***

***Delay\_ms(40);***

***ejercicio1();***

***}***

***if (RA2\_bit) { // button on RA1 pressed***

***Delay\_ms(40);***

***ejercicio2();***

***}***

***} while(1);***

***}***

***unsigned int conversion(unsigned int num) {***

***switch (num) {***

***case 0 : return 0x3F;***

***case 1 : return 0x06;***

***case 2 : return 0x5B;***

***case 3 : return 0x4F;***

***case 4 : return 0x66;***

***case 5 : return 0x6D;***

***case 6 : return 0x7D;***

***case 7 : return 0x07;***

***case 8 : return 0x7F;***

***case 9 : return 0x6F;***

***}***

***}***

***void ejercicio1() {***

***ANSEL = 0xFF;***

***while(1){***

***pot\_analg = ADC\_read(0);***

***dig1=(pot\_analg\*5\*19.8)/1023;***

***PORTB=dig1;***

***digmenossig= dig1%10;***

***displaymenossig= conversion(digmenossig);***

***PORTC= displaymenossig;***

***digmassig=(dig1/10)%10;***

***displaymassig=conversion(digmassig);***

***PORTD= displaymassig;***

***}***

***}***

***void ejercicio2() {***

***do {***

***PORTD = 0b00011110;***

***PORTC = 0b00111110;***

***i=0;***

***dig\_com=0;***

***if (RA1\_bit) { // button on RA0 pressed***

***do {***

***display\_dig();***

***PORTB = 0b00000001;***

***if (PORTB == 0b00000001) {***

***com=1;***

***};***

***if (RA2\_bit==1 && com==1) {***

***dig\_com++;***

***display\_dig();***

***};***

***Delay\_ms(100);***

***PORTB = 0b00000010;***

***Delay\_ms(100);***

***PORTB = 0b00000100;***

***Delay\_ms(100);***

***PORTB = 0b00001000;***

***Delay\_ms(100);***

***PORTB = 0b00010000;***

***Delay\_ms(100);***

***PORTB = 0b00100000;***

***Delay\_ms(100);***

***PORTB = 0b01000000;***

***Delay\_ms(100);***

***PORTB = 0b10000000;***

***if (PORTB == 0b10000000) {***

***com=2;***

***};***

***if (RA2\_bit == 1 && com==2) {***

***dig\_com++;***

***display\_dig();***

***};***

***Delay\_ms(100);***

***PORTB = 0b01000000;***

***Delay\_ms(100);***

***PORTB = 0b00100000;***

***Delay\_ms(100);***

***PORTB = 0b00010000;***

***Delay\_ms(100);***

***PORTB = 0b00001000;***

***Delay\_ms(100);***

***PORTB = 0b00000100;***

***Delay\_ms(100);***

***PORTB = 0b00000100;***

***Delay\_ms(100);***

***PORTB = 0b00000010;***

***Delay\_ms(100);***

***PORTB = 0b00000001;***

***if (dig\_com == 10) {***

***i=21;***

***};***

***i++;***

***} while(i<20);***

***}***

***PORTC == 0b00110110;***

***PORTD == 0b01110001;***

***} while(1);***

***}***

***void display\_dig() {***

***dig\_1=dig\_com%10;***

***display\_me = conversion(dig\_1);***

***PORTC = display\_me;***

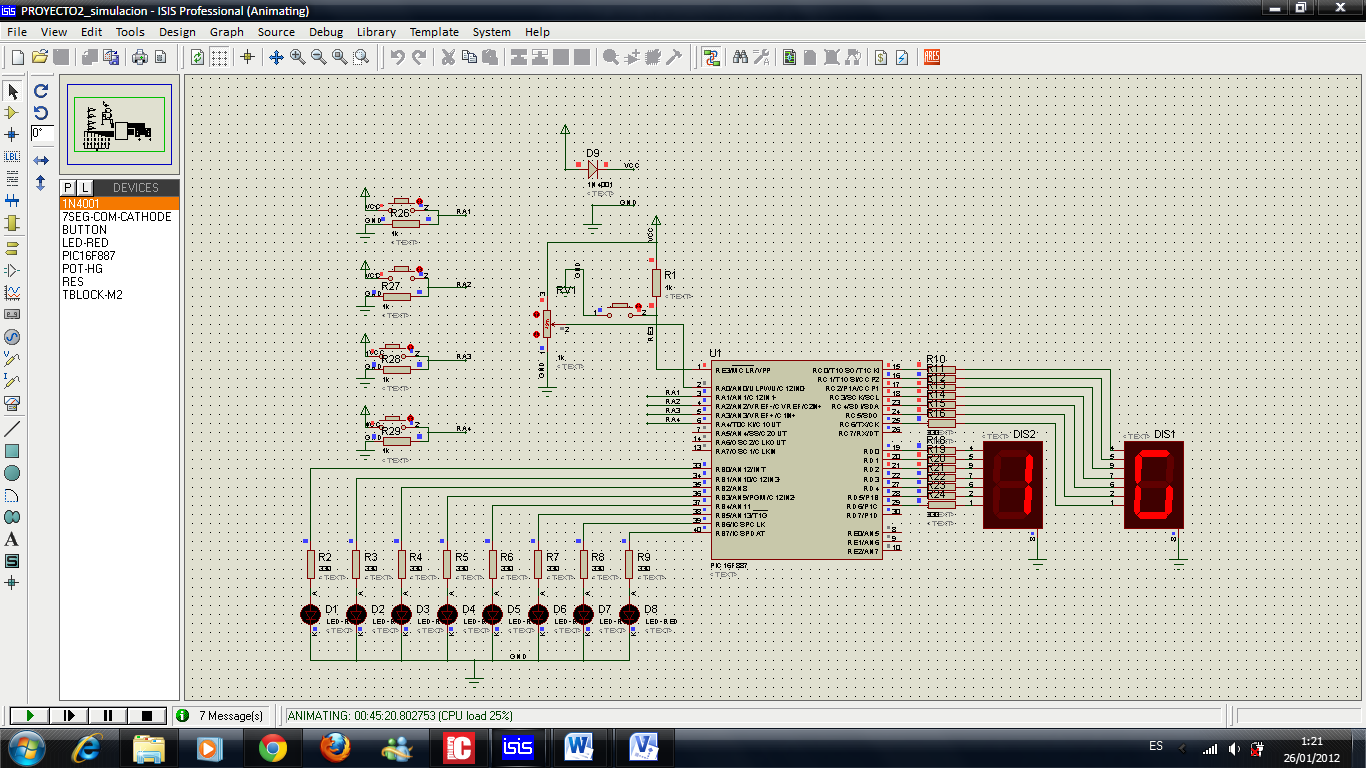
***dig\_2=(dig\_com/10)%10;***

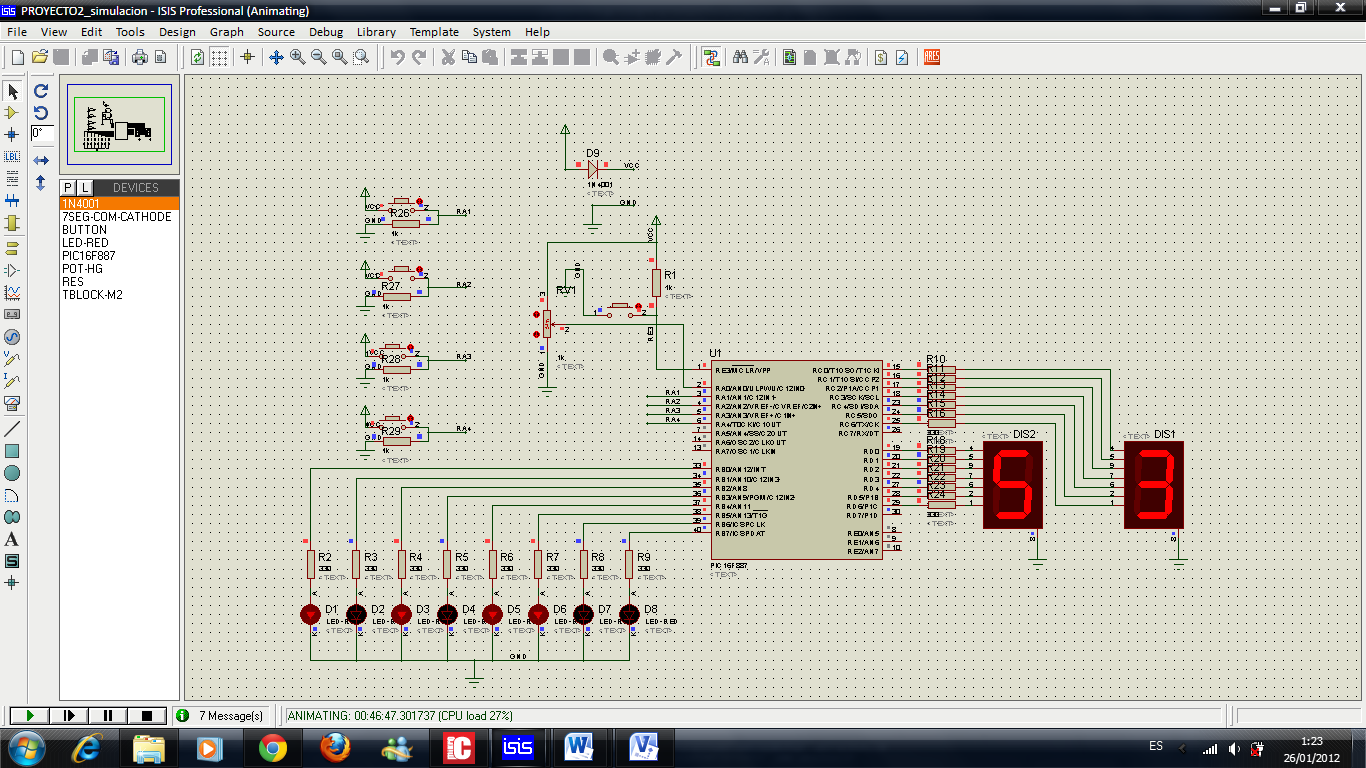
***display\_ma = conversion(dig\_2);***

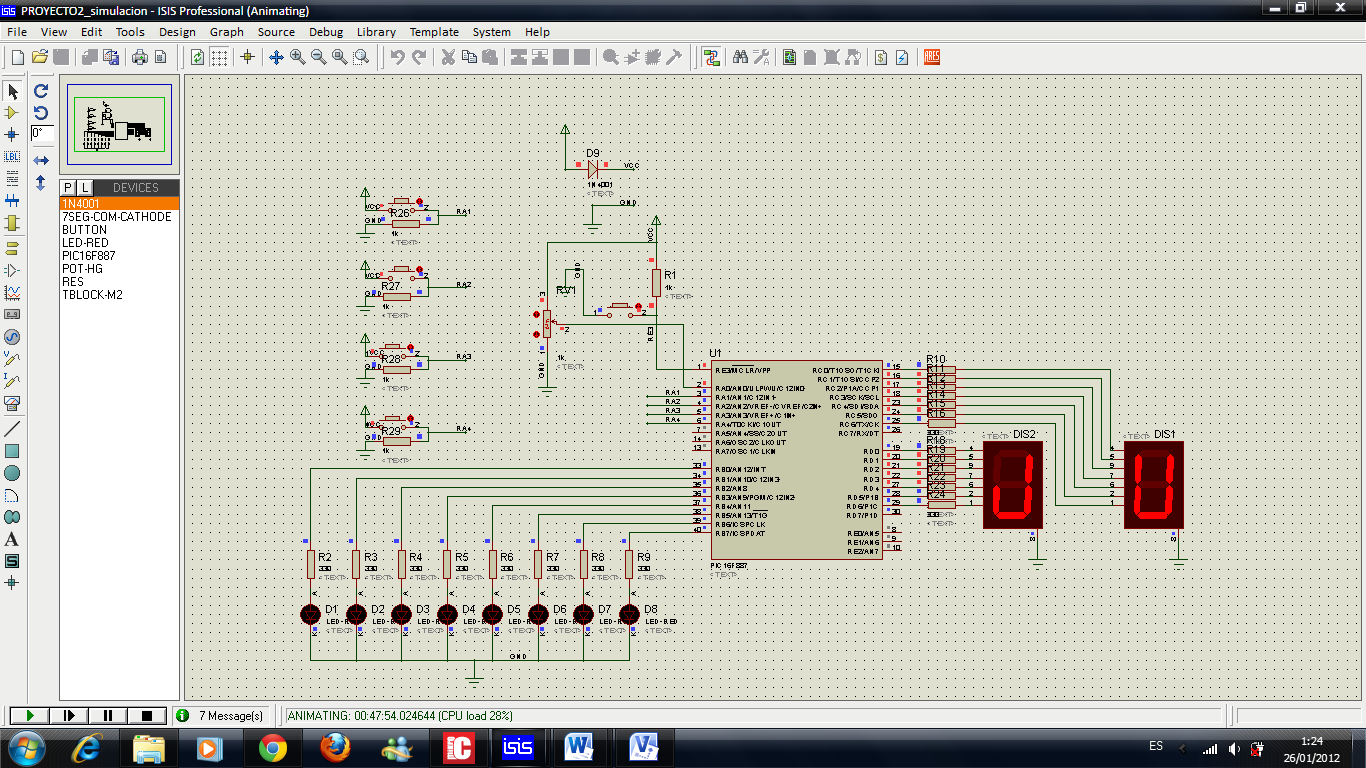
***PORTD = display\_ma;***

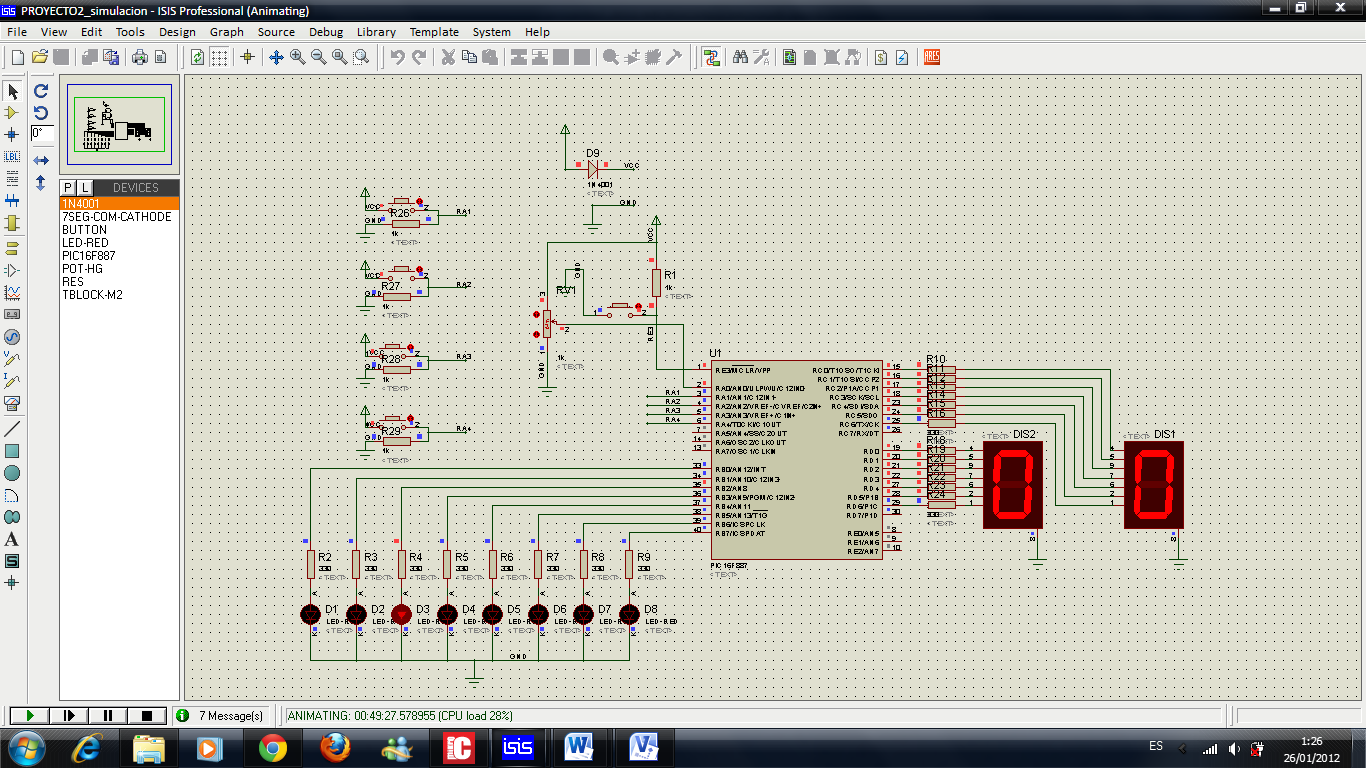
***}***

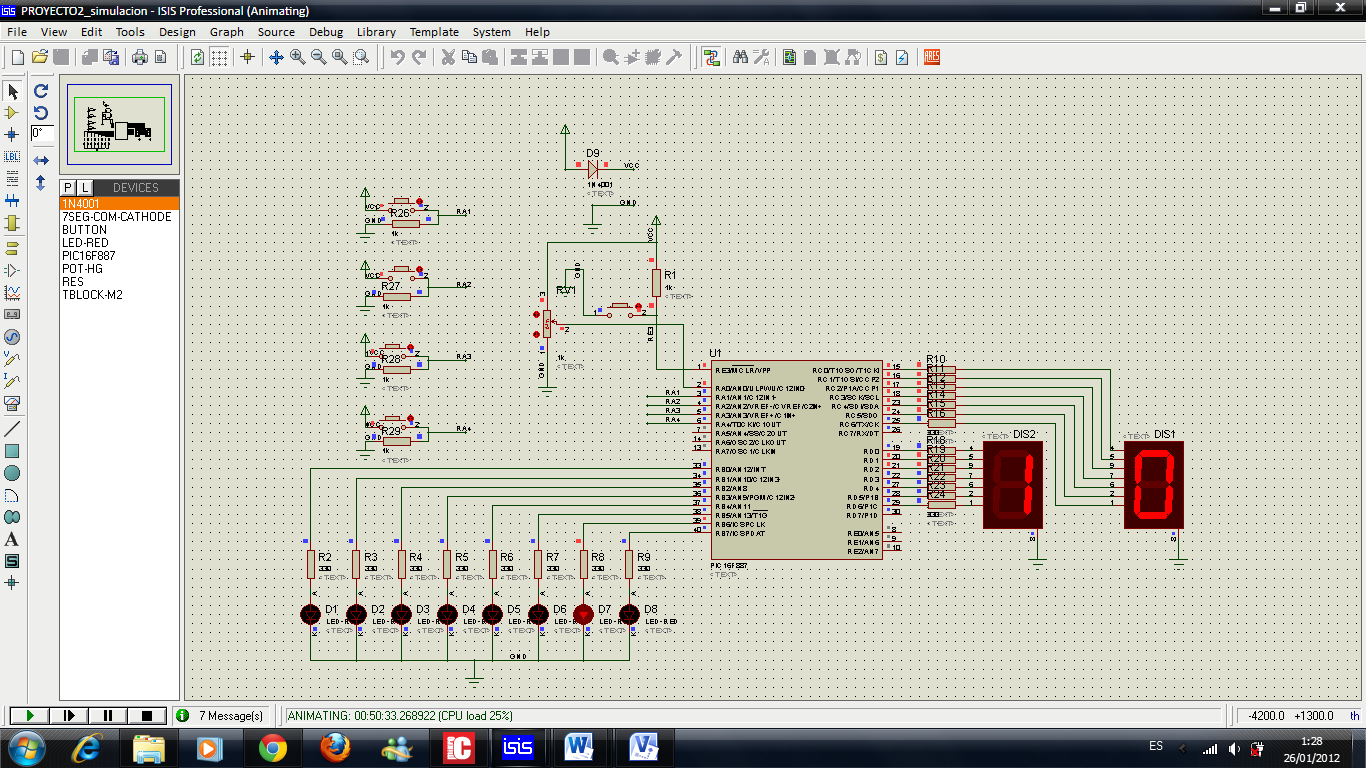
1. **Copia impresa del circuito armado en PROTEUS para la simulación en el momento de su ejecución (Por ejemplo con LEDS encendidos).**



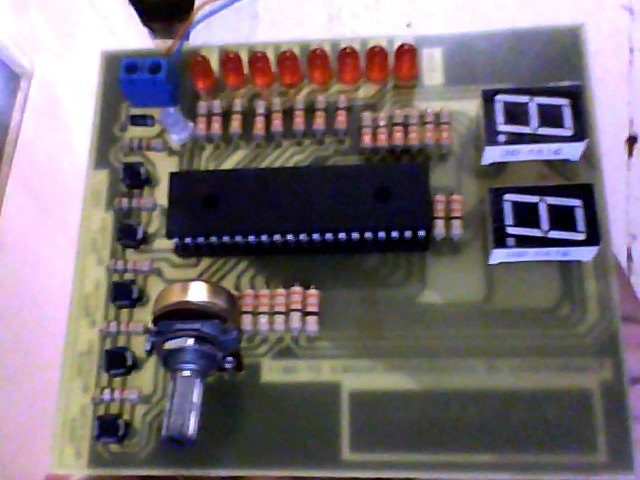








1. **circuito IMPRESO**

****

1. **CONCLUSIONES.**

* Se demostró la correcta utilización del convertidor de analógico a digital ADC usando las respectivas funciones del mismo en la programación en C, como son ADC\_Init() y esta rutina inicializa el módulo interno del PIC ADC para trabajar con el reloj RC. Reloj determina el período de tiempo necesario para realizar la conversión AD, ADC\_Read() y esta Inicializa el módulo interno del PIC ADC para trabajar con el reloj RC.
* El ADC siempre nos da valores en 10 bits, como solo necesitábamos 8 bits se tuvo que utilizar conversiones de bits utilizando las formulas aprendidas en clases en la cual teníamos en cuenta el valor de voltaje en la entrada, el voltaje referencial que este caso era +5v y el muero de bits, ya que también se necesitaba obtener valores de 0 a 99 para poder mostrar con facilidad en los 2 displays de 7 segmentos.
* Se tuvo que crear funciones para la conversión de valores en binario y así mostrar correctamente los números del 0 al 9 en cada display de 7 segmentos como es el caso de *convertir()*.
* Con ayuda de funciones básicas de programación se pudo implementar el juego “agarra el bit” utilizando retardos en milisegundos utilizando la función Delay\_ms.
* MikroC PRO for PIC es un excelente compilador para programaciones de Pics.

1. **RECOMENDACIONES.**

* Antes de realizar cualquier proyecto es necesario tener a la mano todos los materiales y verificar su buen estado antes de conectar para no tener problemas en la presentación final. Es útil tener más de un elemento de los necesarios.
* En la programación es recomendable primero poner la declaración de variables y prototipos de funciones.
* Tener muy en cuenta la utilización de entradas cuando estas sean analógicas o digitales ya que puede generarse problemas en el funcionamiento del programa
* Al terminar de soldar se recomienda medir con un multímetro la continuidad de la placa, y probar también si no existe algún corto circuito, ya que se puede dar el caso de que al soldar, se haya producido la unión de las pistas y esté haciendo corto circuito, también se recomienda limpiar la placa, para extraer causada por la pasta para soldar.