**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y**

**COMPUTACIÓN**

****

**LABORATORIO DE MICROCONTROLADORES**

**PROYECTO # 2**

**Lectura de valor analógico de un potenciómetro y Tema Libre:**

**juego de 40**

**Nombre:**

Luis Alfonso Loayza Feijoo

**Profesor:**

Ing. Villavicencio

**Paralelo: 7**

**Fecha de Presentación:**

27 de Enero del 2012

**Especificaciones técnicas del proyecto:**

El siguiente proyecto es desarrollado en lenguaje C y ejecuta dos programas:

* Primer Programa: Nos da la lectura de los valores analógicos de un potenciómetro presentando las variaciones en código binario a través de 8 LED’s además de dos DISPLAY’s de 7 segmentos que se encargan de mostrar los valores en base 10.
* Segundo Programa: Consiste en el clásico juego del 40 dando la opción de dos jugadores nos muestra a través de los display’s s valores generados aleatoriamente dándonos la opción de retirarnos o de seguir el juego por medio de las botoneras, el valor que me muestra en los display’s me indicara si sigo en el juego o si perdi al pasarme de 40.

**Descripción del Proyecto:**

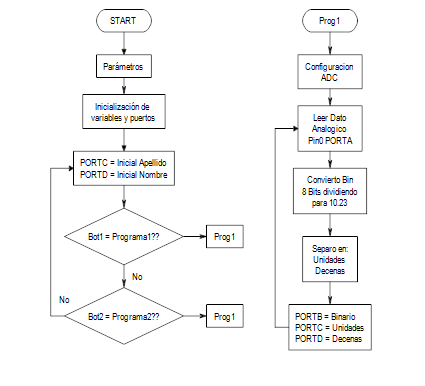
Para efectuar el ejercicio 1 hacemos uso del modula ADC que se encuentra en el micro controlador, el valor del potenciómetro que se ubica en el PORTA es leído y mostrado su equivalente digital en decimal de la siguiente manera: Unidades y Decenas respectivamente en cada displays conectados a los puertos C y D. No olvidando que su valor binario se muestra en los leds de colores conectados al PortB

Con el siguiente ejercicio consiste en el famoso juego del 21 en el cual primero empieza un jugador que es referenciado en los display’s como jugador 1 una vez que el juego empieza me asigna valores aleatorio en cada display estos valores al final se suman y si me paso del 40 indica que perdí

**Diagrama de Bloques:**



**Diagrama de Flujo del programa principal y del programa1:**



**Diagrama de Flujo del juego o ejercicio 2**

Suma1= 0, suma2 =0, A=0

Empieza el jugador1

Seleccion carta

0

1

Jugador2

Suma1=suma1+A

**0 0**

retira

Pasar otro juga

Selecciona carta

Pasar otro juga

retira

**1 1**

retira

Suma2=suma2+A

**Descripción del Algoritmo**

Se escriben los parámetros de configuración y inicializan las variables y los puertos a ser utilizados. Se muestran en los puertos C y D las iniciales del estudiante. Quedamos encerrador en un lazo infinito que pregunta por la BOTONERA1 o la BOTONERA2.

Si se presiona la BOTONERA2, se ejecuta el PROGRAMA1: Se configura el modulo ADC y se leen los Datos Analógicos desde el PIN0 del PORTA. Una vez obtenido el valor digital se lo divide para 10.23 para obtener un rango de 0 a 99. Seguido separamos el número en unidades y decenas y Finalmente mostramos el valor binario en el PORTB y el valor Digital Decimal en los PUERTOS C y D. Usando un arreglo de códigos de 7 segmentos para poder mostrar el número correctamente.

Si se presiona la BOTONERA3, se ejecuta el PROGRAMA2: Se inicializan las variables y puertos a usar; Luego de esto se espera a que el usuario inicia un conteo regresivo para comenzar la batalla, la misma que depende de las teclas R1 para jugador 1 y R4 para jugador 2. Cada vez que se presiona la tecla R1 se realiza un desplazamiento hacia la derecha y cuando se presiona la tecla R4 el desplazamiento ocurre a la izquierda, de tal manera que si uno de los jugadores logra sacar el LED encendido del ring (Desborde) por el lado del oponente gana el juego.

**Programa Fuente:**

**#line 1 "C:/Users/admin/Documents/2011 - II termino/Proyecto\_Luis/pe.c"**

unsigned short np, ng, digit, digit1;

signed short i;

unsigned short flag, ind, tmp, cont, suma1, suma2, oldstate = 0;

unsigned short mask(unsigned short num);

unsigned short mask1(unsigned short num);

unsigned short invert (unsigned short num);

unsigned short carta(unsigned short num);

void ejercicio1 ();

void ejercicio2 ();

unsigned short conversion(float n);

const inicial = 0x38;

const inicial1 = 0x3E;

const E = 0x79;

const G = 0x3D;

const J = 0x1E;

const P1 = 0x73;

void main() {

// principal:

ANSEL = 0x01; // All I/O pins are configured as digital except AN0

ANSELH = 0;

OSCCON = 0x75;

OPTION\_REG = 0x80;

// PORTA = 0; // Turn off both displays

TRISA = 255; // All port A pins are configured as inputs

PORTB = 0; // Turn off all display segments

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

PORTC = 0; // Turn off all display segments

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

PORTD = 0; // Turn off all display segments

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

do {

PORTB = 0;

PORTD = inicial;

PORTC = inicial;

if(RA2\_bit){

ejercicio1();

}

if(RA3\_bit){

ejercicio2();

// Delay\_ms(500);

}

} while (1); // Endless loop

}

void ejercicio1 (){

float digitp, temp\_res;

unsigned short result, digit10, digit1, digit, i;

do {

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

digitp = (temp\_res\*5)/1023;

i = conversion(digitp);

digit = (char)(i % 10u);

digit1 = mask(digit); // Prepare mask for displaying ones

digit = (char)(i / 10u) % 10u;

digit10 = mask(digit); // Prepare mask for displaying tens

Delay\_ms(100);

PORTC = digit1; // Set mask for displaying ones on PORTD

PORTD = digit10; // Set mask for displaying tens on PORTD

result = i & 0X0F;

result = invert(result);

result = result << 4;

digit1 = i & 0XF0;

digit1 = digit1 >> 4;

digit1 = invert(digit1);

PORTB = result + digit1;

} while(1);

}

void ejercicio2(){

PORTC = 0;

PORTD = 0;

PORTB = 0;

Delay\_ms(250);

suma1 = 0;

suma2 = 0;

tmp = 0;

ind = 0;

do{

if(tmp != 1 && ind != 1) {

PORTC = 0;

PORTD = 0;

PORTB = 0;

Delay\_ms(250);

PORTD = J;

PORTC = mask(1);

Delay\_ms(500);

PORTC = 0;

PORTD = 0;

cont = 0;

flag = 0;

do{

PORTB = suma1;

if(cont <= 1 || tmp == 2){

if (Button(&PORTA, 1, 1, 1)) {

do{

oldstate = 1;

i= (rand () % 12) + 2;

} while (Button(&PORTA, 1, 1, 1));

}

if (oldstate && Button(&PORTA, 1, 1, 0)) {

//ng1 = i;

if(cont == 0)

PORTD = carta(i);

else

PORTC = carta(i);

Delay\_ms(500);

suma1 = suma1 + i;

oldstate = 0;

cont++;

}

}

if(suma1 > 40) {

ind = 1;

flag = 1;

}

if(suma1 != 0){

if (Button(&PORTA, 2, 1, 1)) {

do{

oldstate = 1;

} while (Button(&PORTA, 2, 1, 1));

}

if (oldstate && Button(&PORTA, 2, 1, 0)) {

oldstate = 0;

flag = 1;

}

}

if(suma1 != 0){

if (Button(&PORTA, 3, 1, 1)) {

do{

oldstate = 1;

} while (Button(&PORTA, 3, 1, 1));

}

if (oldstate && Button(&PORTA, 3, 1, 0)) {

oldstate = 0;

flag = 1;

if(tmp == 2)

tmp = 3;

else

tmp = 1;

}

}

}while (flag.F0 == 0);

}

if(tmp != 2 && ind != 1) {

PORTC = 0;

PORTD = 0;

PORTB = 0;

Delay\_ms(250);

PORTD = J;

PORTC = mask(2);

Delay\_ms(500);

PORTC = 0;

PORTD = 0;

cont = 0;

flag = 0;

do{

PORTB = suma2;

if(cont <= 1 || tmp == 1){

if (Button(&PORTA, 4, 1, 1)) {

do{

oldstate = 1;

i= (rand () % 12) + 2;

} while (Button(&PORTA, 4, 1, 1));

}

if (oldstate && Button(&PORTA, 4, 1, 0)) {

//ng1 = i;

if(cont == 0)

PORTD = carta(i);

else

PORTC = carta(i);

Delay\_ms(500);

suma2 = suma2 + i;

oldstate = 0;

cont++;

}

}

if(suma2 > 40) {

ind = 1;

flag = 1;

}

if(suma2 != 0){

if (Button(&PORTA, 2, 1, 1)) {

do{

oldstate = 1;

} while (Button(&PORTA, 2, 1, 1));

}

if (oldstate && Button(&PORTA, 2, 1, 0)) {

oldstate = 0;

flag = 1;

}

}

if(suma2 != 0){

if (Button(&PORTA, 3, 1, 1)) {

do{

oldstate = 1;

} while (Button(&PORTA, 3, 1, 1));

}

if (oldstate && Button(&PORTA, 3, 1, 0)) {

oldstate = 0;

flag = 1;

if(tmp == 1)

tmp = 3;

else

tmp = 2;

}

}

}while (flag.F0 == 0);

}

}while((tmp!=3) && (suma1 < 40) && (suma2 < 40) && (ind != 1) );

PORTC = 0;

PORTD = 0;

PORTB = 0;

Delay\_ms(500);

if(suma1 > 40) {

PORTD = P1;

PORTC = mask(1);

Delay\_ms(500);

PORTD = G;

PORTC = mask(2);

}

else if(suma2 > 40) {

PORTD = P1;

PORTC = mask(2);

Delay\_ms(500);

PORTD = G;

PORTC = mask(1);

}

else if(suma2 == suma1) {

PORTD = E;

PORTC = E;

}

else if(suma1 > suma2) {

PORTD = G;

PORTC = mask(1);

}

else {

PORTD = G;

PORTC = mask(2);

}

Delay\_ms(2000);

}

unsigned short mask(unsigned short 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;

}

}

unsigned short conversion(float n){

unsigned short final;

float tmpf;

tmpf = n;

if(n > 1 && n <=2)

tmpf = tmpf - 1;

if(n > 2 && n <=3)

tmpf = tmpf - 2;

if(n > 3 && n <=4)

tmpf = tmpf - 3;

if(n > 4 && n <=5){

tmpf = tmpf - 4;

}

if (tmpf > 0 && tmpf<= 0.05)

final = 1;

else if (tmpf > 0.05 && tmpf<=(0.05\*2))

final = 2;

else if (tmpf > (0.05\*2) && tmpf<=(0.05\*3))

final = 3;

else if (tmpf > (0.05\*3) && tmpf<=(0.05\*4))

final = 4;

else if (tmpf > (0.05\*4) && tmpf<=(0.05\*5))

final = 5;

else if (tmpf > (0.05\*5) && tmpf<=(0.05\*6))

final = 6;

else if (tmpf > (0.05\*6) && tmpf<=(0.05\*7))

final = 7;

else if (tmpf > (0.05\*7) && tmpf<=(0.05\*8))

final = 8;

else if (tmpf > (0.05\*8) && tmpf<=(0.05\*9))

final = 9;

else if (tmpf > (0.05\*9) && tmpf<=(0.05\*10))

final = 10;

else if (tmpf > (0.05\*10) && tmpf<=(0.05\*11))

final = 11;

else if (tmpf > (0.05\*11) && tmpf<=(0.05\*12))

final = 12;

else if (tmpf > (0.05\*12) && tmpf<=(0.05\*13))

final = 13;

else if (tmpf > (0.05\*13) && tmpf<=(0.05\*14))

final = 14;

else if (tmpf > (0.05\*14) && tmpf<=(0.05\*15))

final = 15;

else if (tmpf > (0.05\*15) && tmpf<=(0.05\*16))

final = 16;

else if (tmpf > (0.05\*16) && tmpf<=(0.05\*17))

final = 17;

else if (tmpf > (0.05\*17) && tmpf<=(0.05\*18))

final = 18;

else if (tmpf > (0.05\*18) && tmpf<=(0.05\*19))

final = 19;

else if (tmpf > (0.05\*19) && tmpf<=(0.05\*20))

final = 20;

else

final = 0;

if(n > 1 && n <=2)

return (final + 20);

else if(n > 2 && n <=3)

return (final + 40);

else if(n > 3 && n <=4)

return (final + 60);

else if(n > 4 && n <=5) {

if(n > 4.85 && n <= 4.925)

return 98;

else if(n > 4.925)

return 99;

else

return (final + 80);

}

else

return final;

}

unsigned short invert (unsigned short num) {

switch (num) {

case 0 : return 0x00;

case 1 : return 0x08;

case 2 : return 0x04;

case 3 : return 0x0C;

case 4 : return 0x02;

case 5 : return 0x0A;

case 6 : return 0x06;

case 7 : return 0x0E;

case 8 : return 0x01;

case 9 : return 0x09;

case 10 : return 0x05;

case 11 : return 0x0D;

case 12 : return 0x03;

case 13 : return 0x0B;

case 14 : return 0x07;

case 15 : return 0x0F;

}

}

unsigned short carta(unsigned short num) {

switch (num) {

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;

case 10 : return 0x77;

case 11 : return 0x1E;

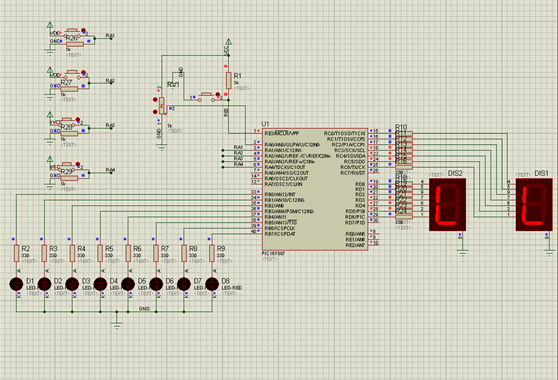
case 12 : return 0x67;

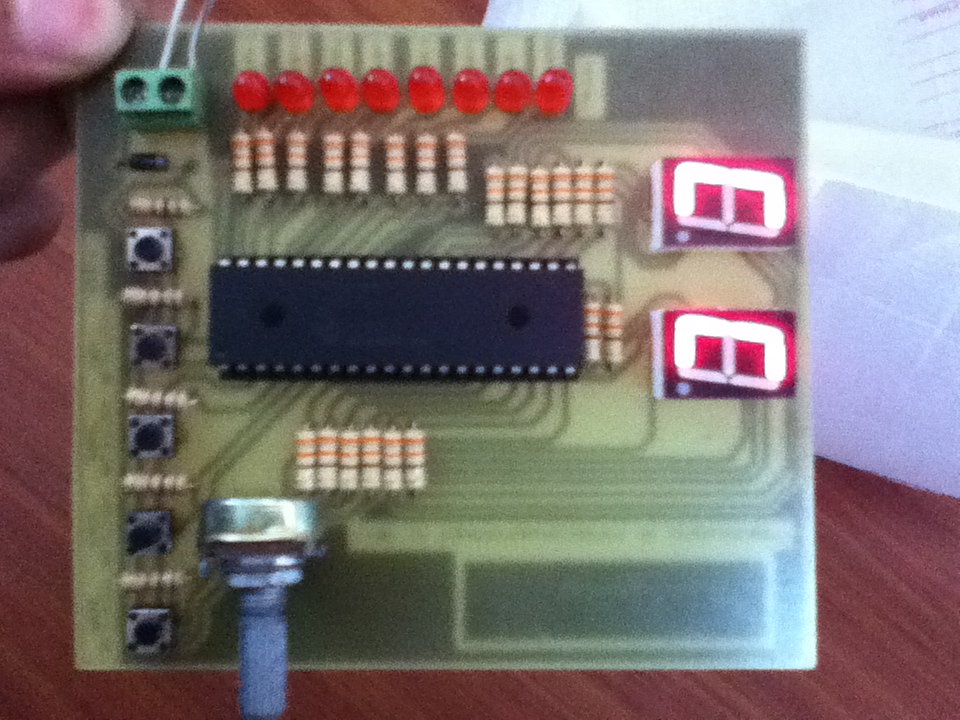
case 13 : return 0x76;

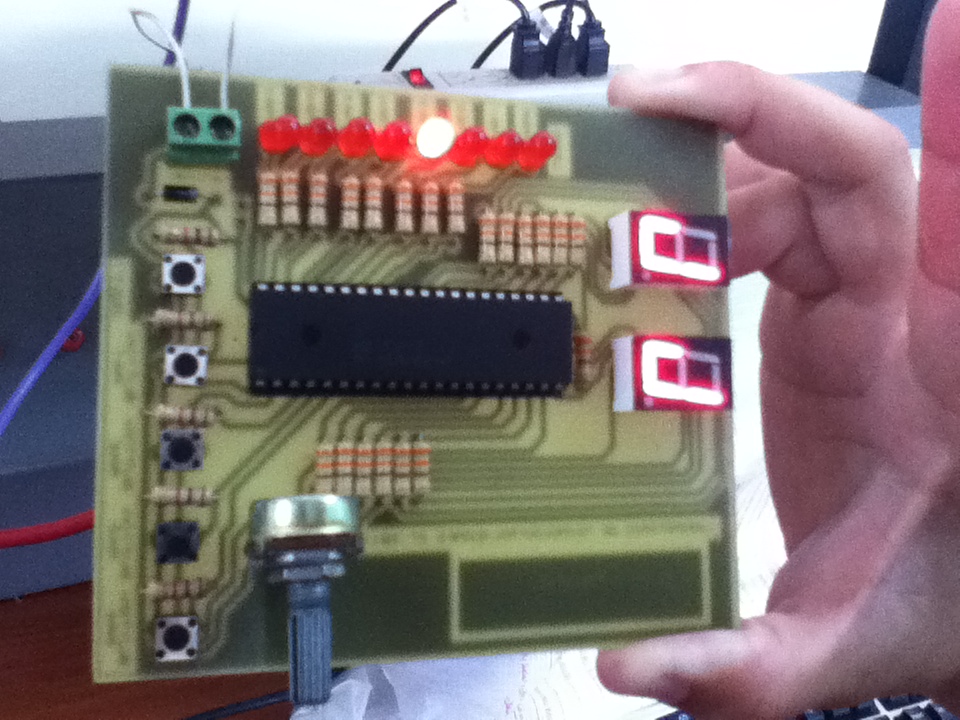
}

}

**Circuito impreso:**





****

**Conclusiones**

* Para la realización del proyecto fue necesario el uso de banderas e indicadores ya que estas variables nos permiten implementar un algoritmo más eficiente debido a que nos ayudan a controlar las diferentes instancias de nuestro proyecto y de esta manera se evita el desperdicio de líneas de código que hacen redundante el mismo, simplificando la sintaxis repetitiva y compactándolo como un solo bloque controlador para diferentes instancias.
* En este proyecto se amplió el conocimiento de las aplicaciones de control de los PIC. Se comprobó el funcionamiento de programas que están implementados para familiarizarnos con la utilización de los módulos de Conversión Analógico-Digital (ADC), los cuales en la teoría son un poco abstractos hasta que se plasman en una práctica o proyecto.
* La realización de este proyecto nos es de gran utilidad debido a todos los conceptos que implica para la correcta realización del mismo, ya sea a la hora de configurar los registros adecuadamente hasta poner en práctica nuestros conocimientos sobre los comandos utilizados en lenguaje C para microcontroladores, los cuales varían un poco por la implementación de ciertas funciones que no están disponibles en la sintaxis del lenguaje C tradicional.

**Recomendaciones**

* Se recomienda tener en un archivo aparte guardado todas las subrutinas vistas tanto en la clase teórica como en la clase práctica, para poder realizar la resolución de problemas de manera menos redundante.
* Tener mucho cuidado a la hora de soldar, es recomendable revisar puntos de contacto sobre la placa, conexión entre pistas, tratar de mantener orden y cuidado en este proceso para evitar el mal funcionamiento al momento de probar el proyecto en la placa.
* Es recomendable usar Delays cuando se necesite observar de manera más lenta algún proceso, haciendo posible ver en más detalle el funcionamiento de una tarea o proyecto para así realizar correcciones si se detectan errores.