Universidad Nacional de La Plata

Trabajo Práctico 2 - Grupo 20

Juego interactivo
“ENCUENTRE EL NÚMERO”

Circuitos Digitales y Microcontroladores

Tarifa, Carla.

Carballo Ormaechea, Lucas.

Índice

1. Problema

Implementar el juego interactivo “ENCUENTRE EL NÚMERO”. Para esto se dispone de un display LCD de 2 líneas, un teclado matricial 4x4 y el MCU Atmega328p. La implementación deberá hacerse con máquina de estados temporizadas con Timer. A continuación se enumeran los requerimientos que debe satisfacer el sistema:

a) Cuando el equipo se inicia deberá mostrar en la primer línea del LCD un mensaje de bienvenida y en la segunda línea el texto: “P/ JUGAR PULSE A”.

b) Si el usuario presiona ‘A’ se mostrará en la primera línea “JUGANDO”, se iniciará un temporizador interno, y se generará un número aleatorio entre 0 y 99 sin mostrarlo. En la segunda línea se mostrará “INGRESE NUM”

c) El jugador deberá ingresar por teclado un número entre 0 y 99. Si está fuera de rango se le solicitará otro. Si el número ingresado coincide con el generado internamente se mostrará en la primera línea “GANADOR” y en la segunda línea se mostrará el tiempo que tardó en resolverlo, por ejemplo “TIME: 1:45:682”. Luego de 3 segundos el sistema volverá al estado inicial.

d) Si el número ingresado es menor que el generado internamente, se mostrará al lado del número el símbolo ‘>’ para indicar que debe ingresar un número más alto que el anterior. En caso contrario se mostrará ‘<’ para indicar que debe ingresar un número más bajo que el anterior.

e) Por último, si el jugador quiere descartar la partida puede presionar la tecla ´D´ y volver al estado inicial.

1.1. Interpretación

Se busca implementar un juego en el cual el usuario debe adivinar un número generado aleatoriamente por el sistema. Este se ejecuta en un MCU Atmega328p, un display LCD de 2 líneas y un teclado matricial 4x4. La lógica del juego se implementa utilizando máquinas de estado temporizadas con Timer0.
En resumen:

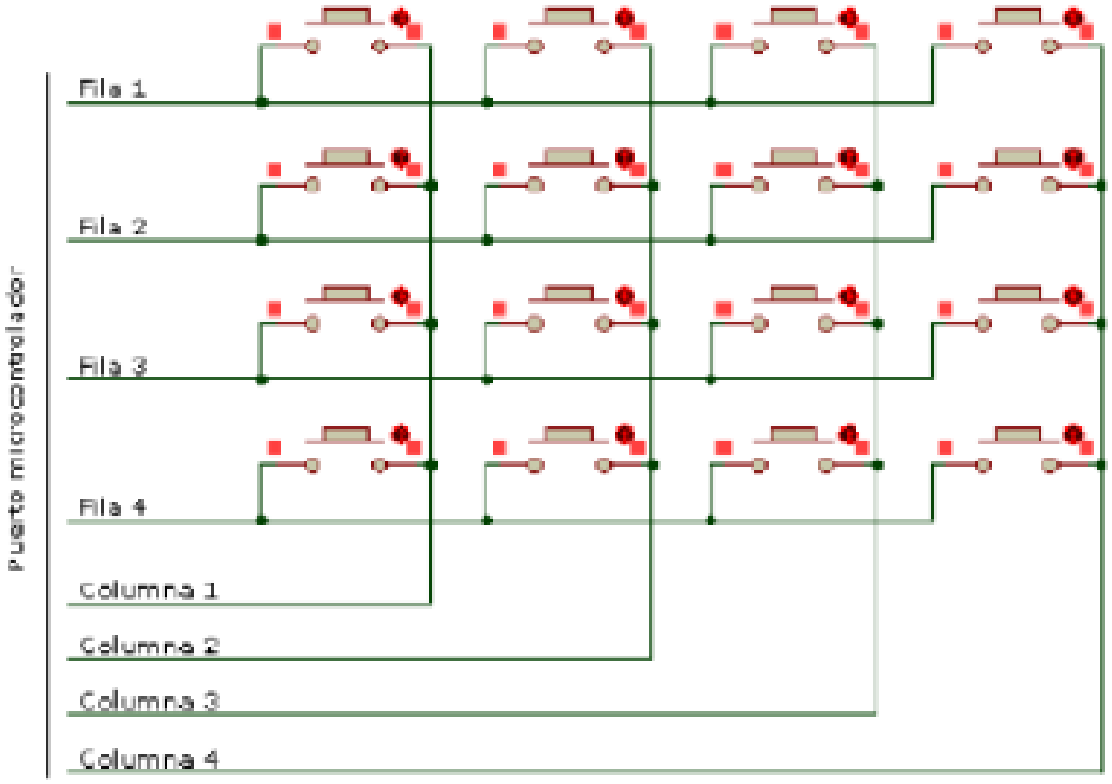
-La pantalla LCD guía al jugador.
-El jugador intenta adivinar un número aleatorio entre 0 y 99.
-Usa timer0 y un clock para medir el tiempo de juego.
-Teclado4x4 para ingresar números y comandos(A/D).

2. Resolución del problema

Para llevar a cabo la resolución del problema , habrá que abordar varios aspectos, ya sea comprensión, desarrollo y configuración de las herramientas provistas por la cátedra.
A continuación se encuentran las explicaciones del desarrollo de las herramientas

2.1 Teclado matricial 4x4

Para la implementación del juego se tendrá un teclado de 4x4 el cual será conectado al microcontrolador y para manipularlo se utilizará la librería “Teclado4x4.h” la cual actúa como driver del teclado. El teclado matricial funciona como una matriz de botones que conecta cada fila con cada columna de la siguiente manera:



Para conocer el boton que es presionado, se utilizara el método de barrido, el cual implica en primer lugar colocar las 4 filas en alto, e ir iterando una a una cambiándola a nivel bajo, dentro de cada iteración chequeamos qué columna está en bajo y de esta forma conocemos fila y columna del botón presionado, información suficiente para saber que botón es, en caso de no haberse presionado ningun boton se retornara el valor cero.
El conexionado del teclado se realizará mediante los puertos B y D,utilizando los pines 4, 3 y 0 del puerto B como salida, además de el pin 7 del puerto D para la última fila, como entradas se setean los pines 3, 5, 4 y 2 del puerto D. Esta conexión fue dada por la cátedra en conjunto con el conexionado del display, por lo que el barrido , contará con un for que itera sobre cada fila del puerto B consultando por cada columna del puerto D y al final una última iteración de cada columna en el puerto D.
Haciendo la implementación en pseudocódigo, la implementación del barrido tendría la siguiente lógica:

```
int Barrido( número de 8 bits)
        Para cada columna(pines 3,5,4 y 2 del puerto D)
            Configuración de entradas con pull-ups internos
            Configuración Salidas en alto
            Setteo la columna correspondiente en bajo
            Para cada fila (pines 4, 3 y 0 puerto B)
                        Si la entrada es 0
                                guardar valor de la tecla en un vector
```

retorno 1
Si la entrada en pinD7 es cero
guardar valor de la tecla en un vector
retorno 1
Setear la columna correspondiente
Si no encuentro tecla presiona retorno 0

El archivo “teclado4x4.c”cuenta con un arreglo estático que contiene una matriz de 16 caracteres, en donde se respeta el orden de las teclas de dicho teclado. Además se cuenta con una función que nos permite garantizar el antirebote o evitar detecciones múltiples llamada KEYPAD_Scan(), esta función recibe como parámetro un puntero a una variable de 8 bits, esta variable tiene el valor del caracter presionado para poder ser almacenado.

2.2 LCD

Las librerías y conexiones de este periférico fueron otorgados por la cátedra, pero aun así haremos algunos comentarios al respecto.
El display cuenta con 2 renglones de 16 caracteres cada 1 y una serie de funciones, por ejemplo : LCDsendChar(c) es una función utilizada para enviar caracteres o LCDstring(s) utilizada para enviar mensajes largos , estas funciones se encuentran en la librería “lcd.h” y “lcd.c”.
En el presente trabajo se tomó la decisión de no utilizar la función LCDclr() , debido a que tiene un retardo mayor que otras funciones y ocasionalmente no funcionaba como era deseado. En su reemplazo se utilizarán caracteres “ “ sobre lo que se desee borrar.

2.3 Conexión

En la siguiente figura se podrá observar la conexión de todos los elementos a utilizar en este proyecto.

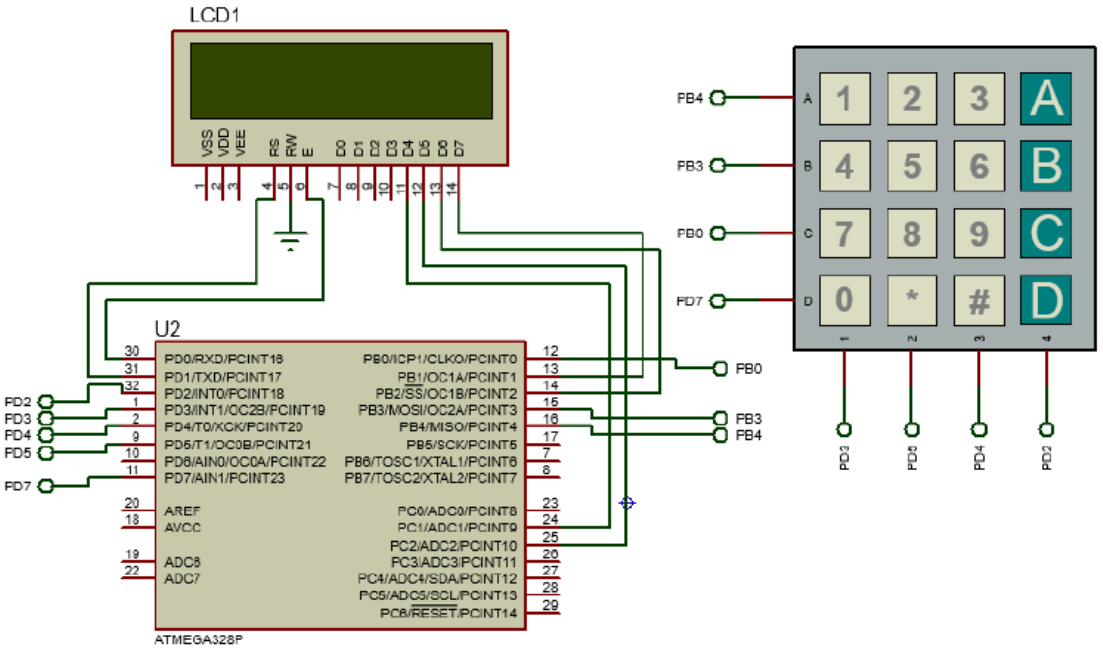


Figura 2.3.1 Conexión de elementos al MCU

2.4 Reloj

Para la implementación del reloj del juego, se escribirá una librería que englobe su comportamiento(“RELOJ.c” y “RELOJ.h”), para ello se definen las siguientes funciones:

- RELOJ_Init(): Inicializa el reloj en cero, éste será llamado cuando el jugador presione la letra ‘A’ para iniciar el juego.
- RELOJ_Update(): Actualiza el reloj , es decir , que avanzara 1 milisegundo el reloj, si llega a 100 suma 1 segundo, si llega a 60 segundos, suma 1 minuto.
- RELOJ_GetTiempo(uint8_t*): Devolverá el tiempo en un arreglo de caracteres [MM:SS:MSMSMS]
- getMili():Devolverá los milisegundos utilizados hasta el momentos para ser utilizados en la generación del número aleatorio

Además de las funciones, la librería cuenta con 3 variables internas estáticas que corresponden al almacenamiento de minutos, segundos y milisegundos.

2.5 TIMER

Se utilizará el **Timer 0**, el cual cuenta con dos registros, TCNT0(contador) y OCR0(registro de comparación) y dos banderas TOV(Overflow) y OCF0(Flag de comparación).

Este fue configurado en modo CTC, en el cual el temporizador cuenta desde 0 hasta un valor de comparación predefinido y luego se reinicia a 0. Es decir, cuando el temporizador alcanza el valor de comparación, se produce una coincidencia y se ejecuta una acción específica, como una interrupción o un cambio en la salida del temporizador.
Podemos conocer el tiempo entre cada interrupción mediante una simple función

$$\frac{(Valor\ de\ comparación + 1)*Preescalar}{Frecuencia\ del\ microprocesador}$$

● La frecuencia del microprocesador se define en 16MHz.
● El valor de comparación es asignado a la variable OCR0A la cual se estableció en 249.
● El preescalar se define en el registro TCCR0B el cual seteamos CS01 y CS00 en 1 mientras que CS02 en 0, indicando al temporizador que utilizara el preescalar de 64.
● El modo CTC se debe almacenar en el registro TCCR0A seteando en 1 el bit 6 de dicho registro.
● Finalmente configuramos las interrupciones habilitando la interrupción COMPA

Una vez inicializado el TIMER 0, tendremos interrupciones cada

$$\frac{(249+1)*64}{16MHz} = 1ms$$

Las cuales hacemos uso de ellas en la función Manejador de interrupción del comparador del TIMER 0 y actualizaremos el reloj cada 1 segundo (1000 ms)

2.6 Máquina de estado(MEF)

Para poder plantear el problema dado, se deberá utilizar una máquina de estado. El uso de este permitirá:

● Gestión de eventos: El juego deberá responder a eventos como la introducción de la letra A, ingresar números, mostrar si el número es mayor o menor al que se está buscando y el cancelamiento de la operación. Una máquina de estados permite gestionar estos eventos de manera ordenada y controlada, definiendo cómo el sistema responde a cada uno de ellos en función del estado en el que se encuentre.

● Comportamiento secuencial: Una máquina de estados proporciona una estructura lógica que permite representar y controlar esta secuencia, asegurando que los pasos se realicen en el orden correcto y según las reglas establecidas.

● Cambios de estado: El juego deberá mantener un estado determinado, como "INICIO",”INICIO_JUGANDO” ,"JUGANDO", y "GANADOR", y cambiar de un estado a otro en función de las acciones del usuario y los eventos que ocurran. La máquina de estados facilita el control y la gestión de estos cambios de estado, asegurando que el sistema responda correctamente y se mantenga en el estado adecuado en todo momento.

● Modularidad y mantenibilidad: El uso de una máquina de estados permite dividir la lógica del programa en diferentes estados y transiciones, lo que mejora la modularidad y facilita el mantenimiento del código.

Para realizar la implementación de la máquina de estados , se deberá saber que

1. El sistema tendrá un estado default que será “INICIO”, el cual indicará el siguiente mensaje en el LCD ,“P/ JUGAR PULSE A”.
2. Si el usuario llegara a presionar una tecla, se pasará al estado “INICIO_JUGANDO”, este se encargará de chequear la tecla presionada . Si la tecla es “A” , entonces deberá generar el número aleatorio , inicializar el reloj y avanzar al estado “JUGANDO” , caso contrario mostrará un mensaje de error y volverá al estado “INICIAL”.
3. Si el estado es “JUGANDO”, se deberá mostrar los siguientes mensajes : “Jugando” e “Introducir num” ,chequear que las teclas presionadas sean números y que estén dentro del rango permitido(0 a 99).Si se cumplen las condiciones , el estado verificará si el numero ingresado es igual al número generado aleatoriamente para poder pasar al estado “GANADOR” , si no es igual , deberá mostrar un mensaje con el número ingresado y el signo de mayor o menor según corresponda para poder ayudar al usuario a adivinar el número aleatorio .
   Si no se cumplen las condiciones esperará hasta que se ingrese un número dentro del rango o verificará si la tecla presionada es ‘D’, en este caso se eliminará la partida y volverá al estado “INICIAL”.
4. Si el estado es “GANADOR” , se mostrará el siguiente mensaje “GANADOR” ,el tiempo que duró la partida (MM:SS:MSMSMS) y esperara 3 seg para poder pasar al estado “INICIAL”

- Cabe aclarar que el número a ingresar deberá ser de dos cifras , es decir , si se quiere ingresar el número 1 , se deberá ingresar 01.

A continuación se muestra el diagrama de la MEF

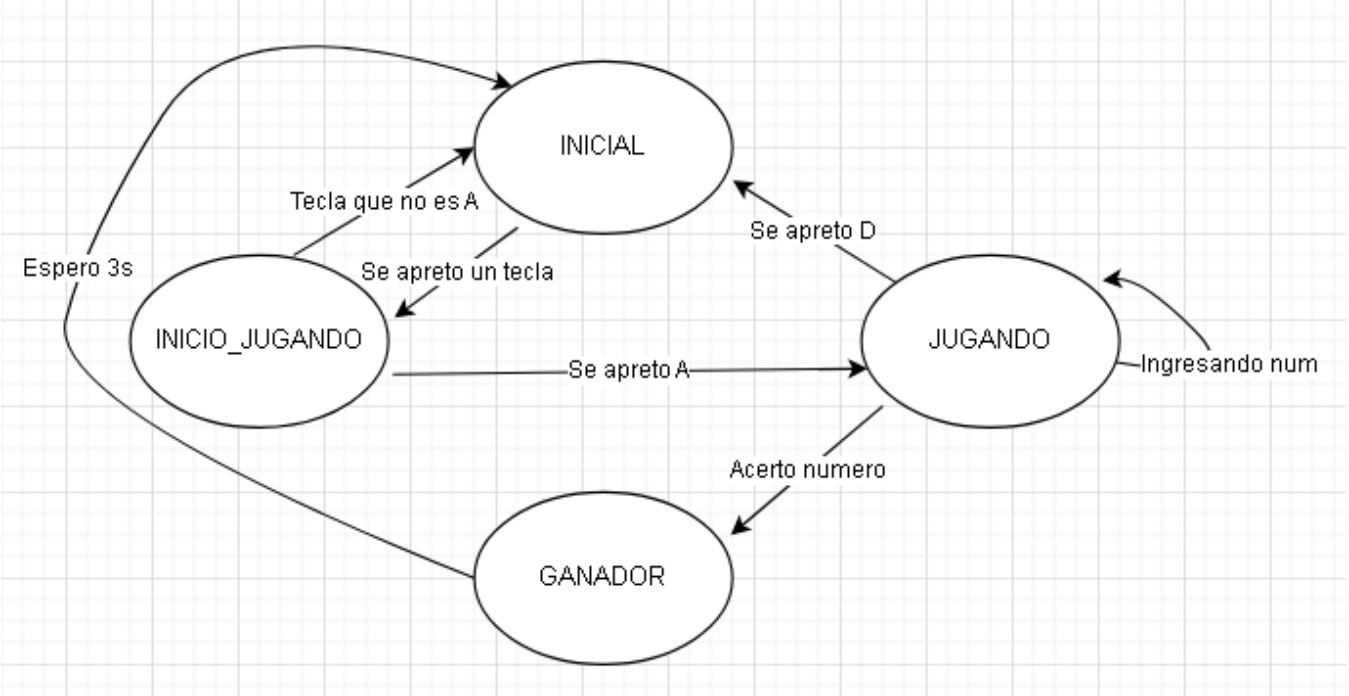


Figura 2.6.1. Diagrama de la MEF.

2.6.1 Implementación de la MEF

Para implementar la MEF que controlará todo el juego se desarrollará una librería ("mef.h").Esta se comunicará con otras para recibir sus entradas y enviar sus salidas.
Se usarán las cuatro librerías vistas en las secciones anteriores: Para determinar si el usuario ha presionado una tecla (entrada de la MEF), para actualizar el display LCD (salida de la MEF), para las operaciones con el tiempo y para las interrupciones.

Para las variables internas se deberán declarar, los estados posibles en un enumerador, ya que es un tipo de dato que puede tomar valores finitos que solo son modificables a la hora de su creación llamado MEF_STATE, un puntero que poseerá el valor de la entrada por teclado (key), la cantidad de iteraciones que tendrá la MEF a cada llamado de actualización de la misma y que se reinicia al cambiar de estado (State_call_cout), el propio estado actual del sistema (estado),la cantidad de teclas presionadas(cantTecla) , un arreglo en donde se irán guardando las teclas presionadas (teclasPresionadas) , el número generado aleatoriamente(numRandom) y por último ,número que contendrá la conversión de ascii a número entero de teclasPresionadas(num).

La librería tendrá 2 métodos públicos: Para inicializar la MEF llamado JUEGO_Init() y para actualizar la MEF llamado JUEGO_Update().

Para el caso de JUEGO_Init(), lo que hace es iniciar la propia MEF en el estado por defecto (es decir, en el estado “INICIO”) y reiniciar la cantidad de llamadas de actualización de la MEF.

En lo que respecta al segundo método (JUEGO_Update()), este será el encargado de leer las entradas y transicionar entre estados en caso de ser necesario.

2.6.2 Implementación del Pseudocódigo

La implementación de la máquina de estados (MEF) en software, es del tipo SWITCH-CASE, donde el método JUEGO_Update() consulta en qué estado está el sistema actualmente. Una vez que encuentra el estado actual, procesa las posibles entradas/salidas.

JUEGO_Update()
begin
    Incremento State_call_count;
      Pregunto en qué estado estoy
        Si estoy en INICIO: //
            Hago lo correspondiente al estado INICIO
        Si estoy en INICIO_JUGANDO: //
            Hago lo correspondiente al estado INICIO_JUGANDO
        Si estoy en JUGANDO: //
            Hago lo correspondiente al estado JUGANDO
        Si estoy en GANADOR: //
            Hago lo correspondiente al estado GANADOR
end

A continuación, se mostrará el pseudocódigo de cada estado:

Si estoy en el estado **INICIO**
    Se inicializa State_call_count
      Escribo en la primera línea del led “BIENVENIDO”.
      Escribo en la segunda línea del led “P/JUGAR PULSE A”.
      Si se presiono una tecla
            Cambio al estado INICIO_JUGANDO

Si estoy en el estado **INICIO_JUGANDO**
      Si presiono la tecla “A”.
            Generar un número aleatorio entre 0 y 99.
          Inicializo reloj de la partida

Cambio al estado JUGANDO

Sino

Aviso por pantalla que fue denegado el acceso

Cambio al estado INICIO

Si estoy en el estado **JUGANDO**

Si State_call_count ==1

Muestro en la primera línea del lcd “JUGANDO”

Muestro en la segunda línea del lcd “INGRESE NUM”

Si la función IntroducirNro* =1

Si el numero ingresado es igual al aleatorio

Cambio al estado GANADOR

Sino

Si el numero ingresado es más grande que el aleatorio

Posicionar el cursor del led al lado del mensaje “jugando”

Muestro el número ingresado y el carácter ‘<’ para que ingrese un numero mas chico

Me mantengo en el estado JUGANDO

Sino

Si el numero ingresado es mas chico que el aleatorio

Posicionar el cursor del led al lado del mensaje “jugando”

Muestro el número ingresado y el carácter ‘>’ para que ingrese un numero mas grande

Me mantengo en el estado JUGANDO

Sino

Si se presiono “D”

Cambio estado INICIO

Si estoy en el estado **GANADOR**

Muestro en la primera línea del led “GANADOR”

Llamo a la función para obtener el tiempo hasta ese momento

Muestro el tiempo en en la segunda línea del led

Espero 3 segundos

Cambio al estado INICIO

*Función IntroducirNro: es la encargada de verificar si se ingresaron los dos dígitos del número , caso contrario retorna 0 , a continuación se mostrará su pseudocódigo

IntroducirNro()

Begin

Si no se presiono ninguna tecla

Devuelvo 0

Si se presiono solo una tecla

Renicicio State_call_count

Reinicio la cantidad de teclas ingresada

Devuelvo 0

Si la tecla presionada es un dígito entre 0 y 9

Guardo la tecla presionada en un arreglo

Si la cantidad de teclas es 2

Reinicio la cantidad de teclas

Llamo a la función DentroDelRango , esta función es la encargada de convertir el contenido del arreglo de ascii a entero y verificar que este dentro del rango (0 a 99)

Sino

Devuelvo 0

End

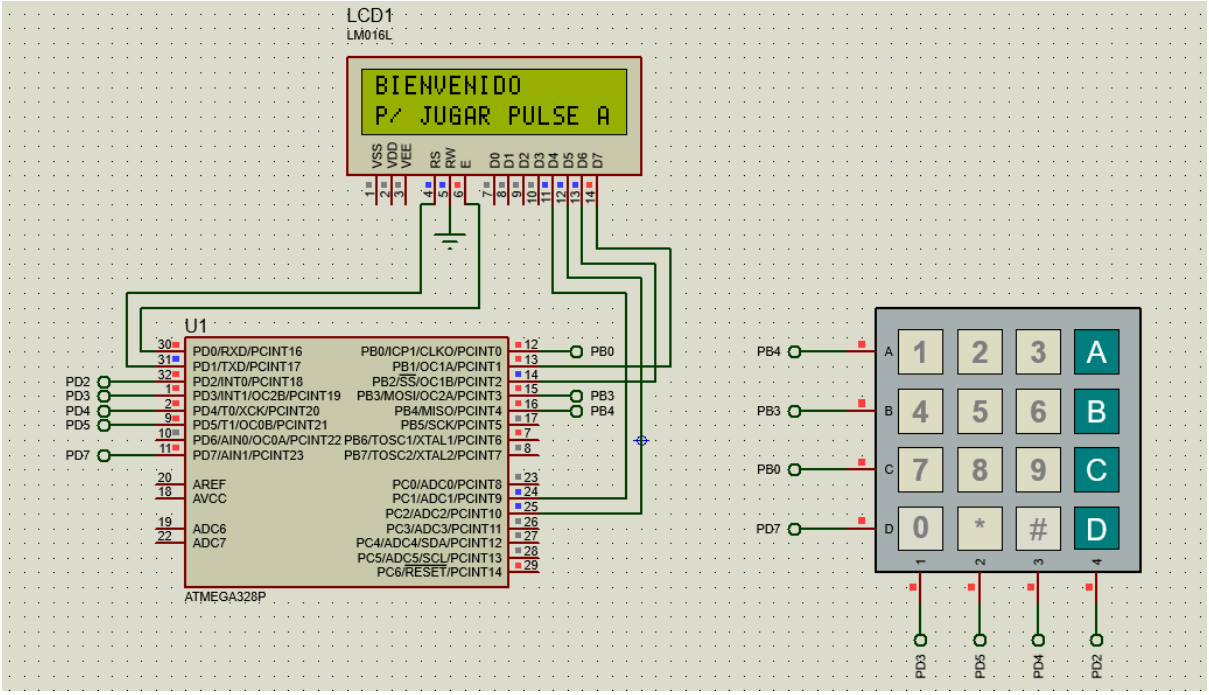
Todos los cambios de estados y salidas anteriormente descritos se encuentran implementados en el MEF.c como funciones privadas, el objetivo de esto es obtener encapsulación y ocultamiento de datos en la implementación del código. Es decir, desde afuera del archivo MEF.c solamente se conocen los métodos JUEGO_Init() y JUEGO_Update(), pero dentro de ella si aparecen todas las funciones para simular las salidas y los cambios de estados.

Tanto para declarar una función o una variable privada dentro de un archivo .c estas deben declararse como ‘static’ y el prototipo de función estar presente en ese mismo archivo.

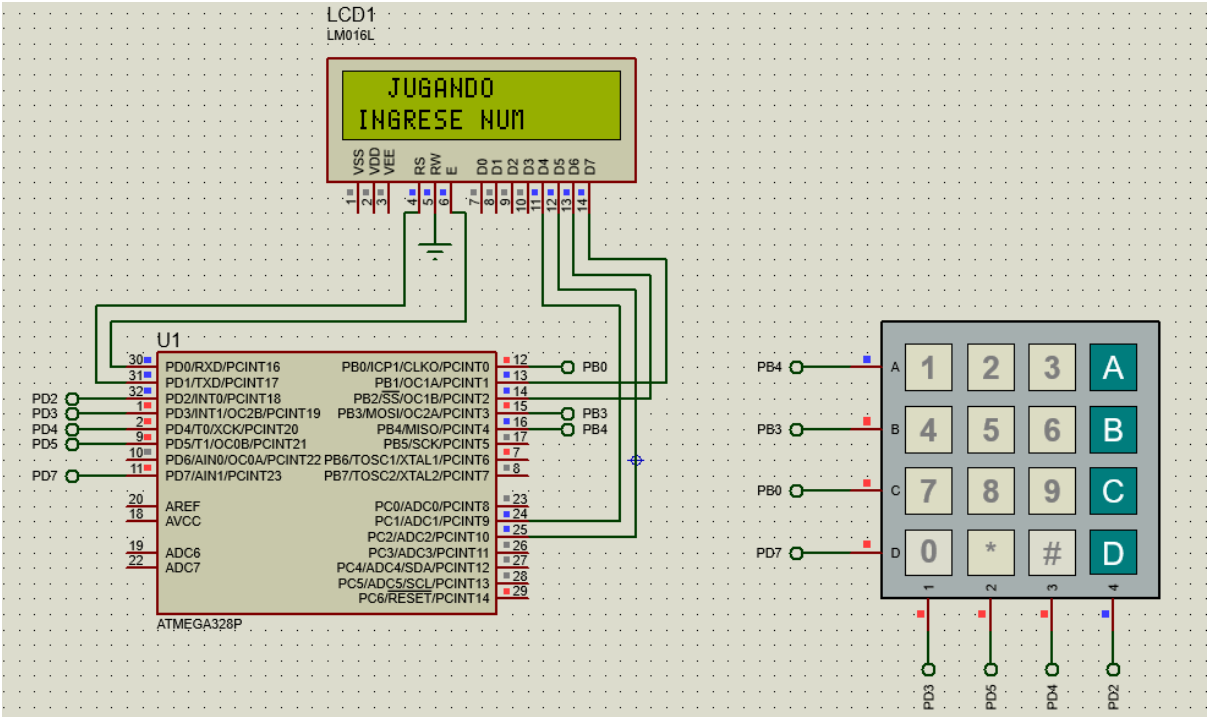
3. Validación

Para validar el correcto funcionamiento del juego , procederemos a mostrar los resultados de algunas pruebas realizadas en el simulador las cuales demuestran la satisfactoria ejecución del trabajo.

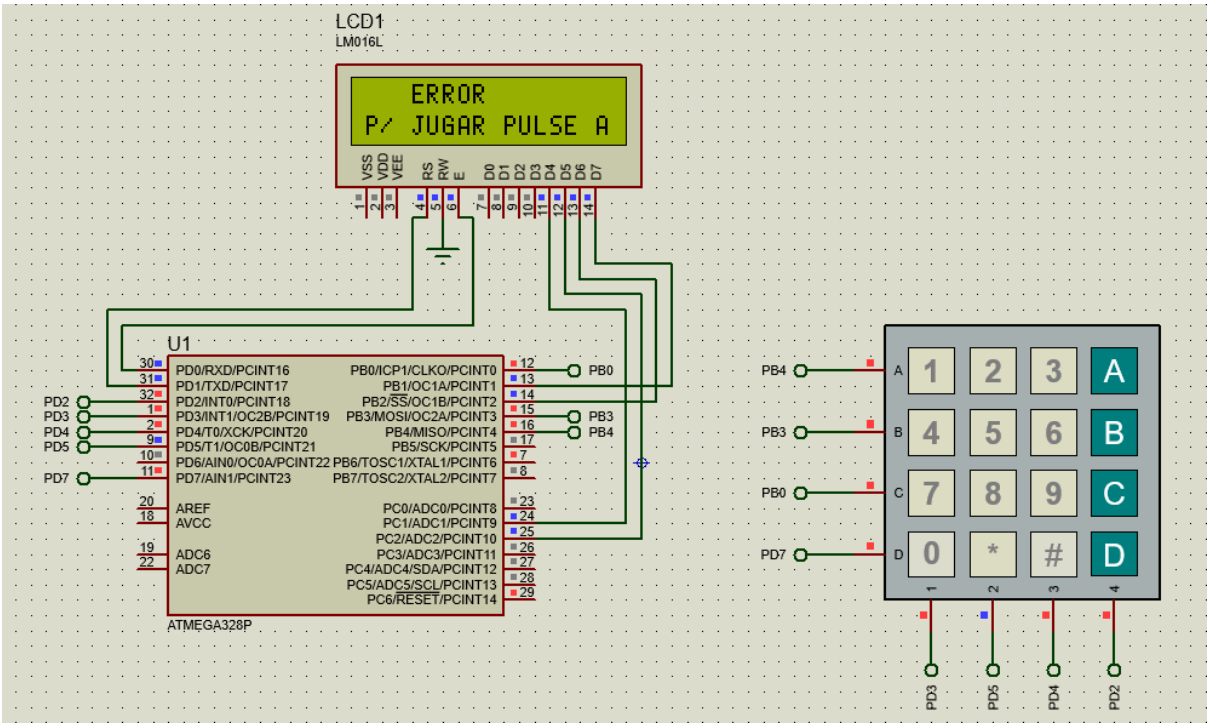
3. 1. Encendido del juego



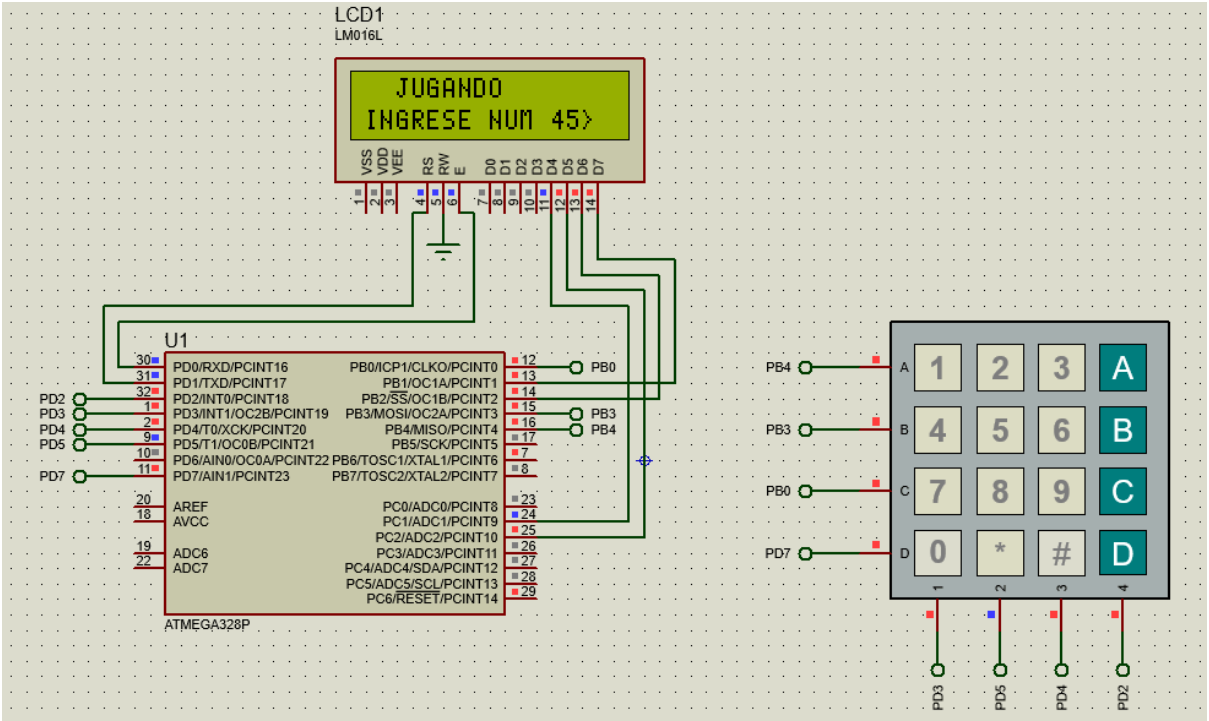
3.1.1 Si se presionó A , inicia el juego



3.1.2 Si se presiono una tecla diferente

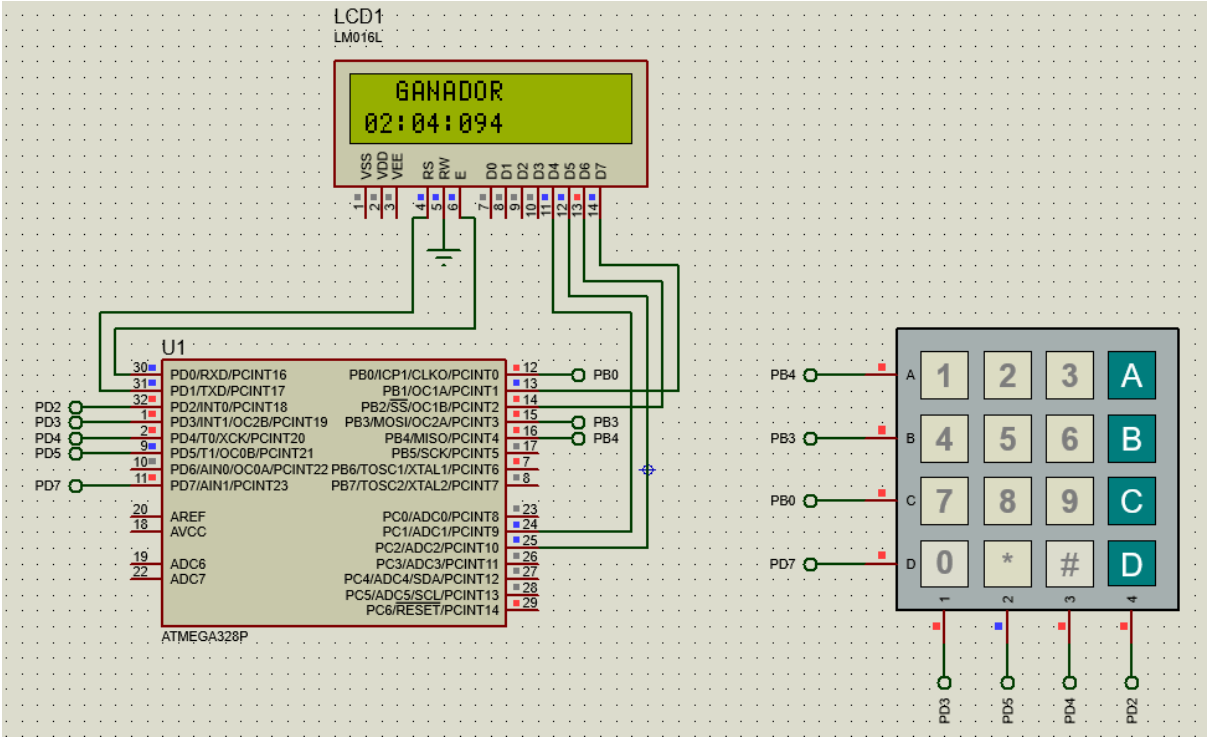


3.2 Juego iniciado , procedemos a ingresar un número



El cartel indica que se debe ingresar un número mayor que 45 y así sucesivamente hasta que se encuentre el número .

3.3 Cuando se encuentra el número buscado



4.Conclusiones

4.1 Enunciado

Saque conclusiones sobre el funcionamiento del programa, sobre las ventajas de utilizar modularización, abstracción y planificación periódica para temporizar tareas. Hágase preguntas como por ejemplo: ¿afecta el tiempo de respuesta a la acción del usuario

4.2 Desarrollo

Una de las ventajas de utilizar la modularización es poder dividir el programa en módulos independientes , estos se podrán reutilizar en otros programas y también facilitar el mantenimiento del código . En cuanto a la abstracción , facilita a que diferentes desarrolladores trabajen en diferentes módulos sin necesidad de conocer todos los detalles , esto evita posibles errores que se puedan provocar y por último la planificación periódica para tareas temporizadas , permite que el MCU realice tareas específicas a intervalos regulares, optimizando el uso del tiempo de CPU.

5. Código

Librería Timer (timer.h y timer.c)

```c
/*
 * timer.h
 *
 * Created: 6/5/2024 16:10:04
 *  Author: LENOVO
 */


#ifndef TIMER_H_
#define TIMER_H_
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include "Reloj.h"
void timer_init();



#endif /* TIMER_H_ */
```

```c
static uint8_t contMEF=0;
void timer_init(){
      //Primero me aseguro que esten deshabilitadas las interrupciones
         TIMSK0 &=~((1<<OCIE0A));

     // Modo CTC
      TCCR0A|=(1<<WGM01);
      TCCR0A|=(1<<COM0A0);//Lo pongo el modo toggle

    //Configuramos TCCR0B con preescaler en 64
      TCCR0B|=(1<<CS00);
      TCCR0B|=(1<<CS01);

    //Inicializo contador
     TCNT0=0;
    //Valor para la comparacion
    OCR0A=249;

    //Hablito interrupciones por compareA
    TIMSK0=(1<<OCIE0A);
    //Habilito interrupciones globales
     sei();
}
        ISR(TIMER0_COMPA_vect){
        if(++contMEF==10){
            contMEF=0;
            RELOJ_Update();
        }
    }
```

Librería Reloj (Reloj.h y Reloj.c)

```c
/*
 * Reloj.h
 *
 * Created: 11/5/2024 17:15:46
 *  Author: LENOVO
 */


#ifndef RELOJ_H_
#define RELOJ_H_

#include <avr/io.h>
void RELOJ_Init();
void RELOJ_Update();
void RELOJ_GetTiempo(uint8_t* );
#include <stdint.h>
uint8_t getMili();



#endif /* RELOJ_H_ */
```

```c
/*
 * Reloj.c
 *
 * Created: 11/5/2024 17:16:08
 *  Author: LENOVO
 */

#include "Reloj.h"
 static uint8_t mili;
 static uint8_t segundo;
 static uint8_t minuto;

 //inicializacion
void RELOJ_Init(){
    mili=0;
    segundo=0;
    minuto=0;
}
void RELOJ_Update(){
    mili++;
    if( mili==100){
        mili=0;
        segundo++;
    }
    if(segundo==60){
        segundo=0;
        minuto++;
    }
}

void RELOJ_GetTiempo(uint8_t* tAct){ //Guarda en el puntero de parametro [MM:SS:MS]
    tAct[0] = (minuto/10) + '0';
    tAct[1] = (minuto%10) + '0';
    tAct[2] = ':';
    tAct[3] = (segundo/10) + '0';
    tAct[4] = (segundo%10) + '0';
    tAct[5] = ':';
    tAct[6] = (mili/100) + '0';
    tAct[7]=((mili-(mili/100)*100)/10 )+ '0';
    tAct[8] = (mili%10) + '0';
}

uint8_t getMili()
{
    return mili;
}
```

.

Librería teclado (teclado4x4.h y teclado4x4.c)

```c
#ifndef TECLADO4X4_H_
#define TECLADO4X4_H_

//#include "main.h"
#include <avr/io.h>

//Inicializacion del teclado
void KEYPAD_Init();

//Para ver si se presiono alguna tecla y devuelvo 1 si si o si no cero
uint8_t KEYPAD_Scan(uint8_t *key);



#endif /* TECLADO4X4_H_ */
```

```c
/*
 * teclado4x4.c
 *
 * Created: 6/5/2024 11:45:48
 *  Author: LENOVO
 */

#include "Teclado4x4.h"

static const uint8_t teclado[4][4]= {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'0','*','#','D'}
};

static const uint8_t filas[4]={PINB4,PINB3,PINB0,PIND7};
static const uint8_t columnas[4]={PIND3,PIND5,PIND4,PIND2};


//Configuro las entradas y salidas de las filas y columnas
void KEYPAD_Init(){
    //filas como entradas
    DDRB &=  ~((1<< PORTB4)|(1<<PORTB3)|(1<<PORTB0));
    DDRD &=  ~((1<<PORTD7)|(1<<PORTD3)|(1<<PORTD4)|(1<<PORTD2)|(1<<PORTD5));
    //seteo de las resistencias internas
    PORTB |= (1<<PORTB4)|(1<<PORTB3)|(1<<PORTB0);
    PORTD |= (1<<PORTD7);

    //COLUMNAS COMO SALIDAS
    //DDRD |=  (1<<PORTD3)|(1<<PORTD4)|(1<<(1<<PORTD2) |PORTD5);

    PORTD |=(1<<PORTD2)|(1<<PORTD3)|(1<<PORTD4)|(1<<PORTD5); // pongo en uno las salidas
}
```

```
uint8_t KepadUpdate(){
    //escanear columnas
    for (uint8_t c=0;c<4;c++){
        DDRB &=  ~((1<< PORTB4)|(1<<PORTB3)|(1<<PORTB0));
        DDRD &=  ~((1<<PORTD7)|(1<<PORTD3)|(1<<PORTD4)|(1<<PORTD2)|(1<<PORTD5));
        DDRD|= (1<<columnas[c]); // pongo como salidas
        PORTD &=~(1<<columnas[c]);// escribo un cero

        //escanear filas
        for (uint8_t f=0;f<3;f++){
            if (!(PINB & (1<<filas[f]))){
                return teclado[f][c];// calculo el valor de la tecla
            }
        }

        if (!(PIND & (1<<PIND7))){
            return teclado[3][c];// calculo el valor de la tecla
        }

        // Vuelve a la configuracion inicial
        DDRD &= ~(1 << columnas[c]);
        PORTD |= (1 << columnas[c]);


    }
    return 0;// no encontre
}
```

```
uint8_t KEYPAD_Scan(uint8_t *pkey)
{
    static uint8_t Old_key, Last_valid_key=0xFF; // no hay tecla presionada;
    uint8_t Key;
    Key= KepadUpdate();
    if(Key==0xFF){
        Old_key=0xFF;// no hay tecla presionada
        Last_valid_key=0xFF;
        return 0;
    }
    if(Key==Old_key) { //2da verificación
        if(Key!=Last_valid_key){ //evita múltiple detección
            *pkey=Key;
            Last_valid_key= Key;
            return 1;
        }
    }
    Old_key=Key; //1era verificación
    return 0;
}
```

Librería MEF (mef.h y mef.c)

```c
/*
 * mef.h
 *
 * Created: 6/5/2024 16:10:25
 *  Author: LENOVO
 */


#ifndef MEF_H_
#define MEF_H_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define F_CPU 16000000UL
#include <util/delay.h>

#include "Reloj.h"
#include "Teclado4x4.h"
#include "lcd.h"
#include "timer.h"

typedef enum { INICIO,INICIO_JUGANDO,JUGANDO,GANADOR} MEF_STATE;
 MEF_STATE estado;

uint8_t State_call_count;

void JUEGO_Init();
void JUEGO_Update();
uint8_t ValidarTecla();
uint8_t IntroducirNro();


#endif /* MEF_H_ */
```

```c
/*
 * mef.c
 *
 * Created: 6/5/2024 16:11:34
 *  Author: LENOVO
 */
#include "mef.h"

static uint8_t cantTecla=0;
static uint8_t num;
static uint8_t numRandom;
uint8_t key;
static uint8_t teclasPresionada[2];


void JUEGO_Init(){ //Inicializa en estado INICIO
    estado = INICIO;
    State_call_count = 0;
}
```

```
void JUEGO_Update(void){//llamada cada 100 ms
    //Contar numero de interrupciones
    State_call_count++;
    switch(estado){
        case INICIO:
                    State_call_count=0;
                    LCDGotoXY(0,0);
                    LCDstring("BIENVENIDO",10);
                    LCDGotoXY(0,1);
                    LCDstring("P/ JUGAR PULSE A",16);
                    if(KEYPAD_Scan(&key)!=0){//Si presiono alguna tecla avanza al siguiente estado
                         estado=INICIO_JUGANDO;
                    }
               break;

         case INICIO_JUGANDO:
                          //Chequeamos tecla presionada
                           if(key=='A'){
                              srand(getMili());// Semilla para la generación de números aleatorios
                              numRandom = rand() % 100;  // Genera un número aleatorio entre 0 y 99
                              RELOJ_Init();
                              estado=JUGANDO;
                           }
                           else {//Ingresa tecla incorrecta
                              //LCDclr();
                              LCDGotoXY(0,0);
                              LCDstring("   ERROR  ",10);
                              _delay_ms(3000);
                              estado=INICIO;
                           }
                           break;
```

```
    case JUGANDO:
    if(State_call_count==1){ //Muestra en el display "JUGANDO"
        LCDGotoXY(0,0);
        LCDstring("  JUGANDO ",10);
        LCDGotoXY(0,1);
        LCDstring("INGRESE NUM     ",16);
        if(IntroducirNro()==1){
            if ( num==numRandom){
                estado = GANADOR;
            }
            else if (num > numRandom){//Si el numero ingresado es mayor al numero random
                LCDGotoXY(12,1);
                LCDstring(teclasPresionada,2);
                LCDGotoXY(14,1);
                LCDsendChar('<');//EL numero ingresado debe ser menor
                _delay_ms(1000);
                estado = JUGANDO;
            }
            else if ( num < numRandom){//Si el numero ingresado es menor al numero random
                LCDGotoXY(12,1);
                LCDstring(teclasPresionada,2);
                LCDGotoXY(14,1);
                LCDsendChar('>');//el numero ingresado debe ser mayor
                _delay_ms(1000);
                estado = JUGANDO;
            }
        }

        else if (key == 'D'){
            //Vuelvo al inicio luego de presionar D en el teclado.
            estado = INICIO;
        }
```

```
                    break;

        case GANADOR:
                    //LCDclr();
                    LCDGotoXY(0,0);
                    LCDstring("  GANADOR ",10);
                    uint8_t tAct[9];

                    RELOJ_GetTiempo(tAct);
                    LCDGotoXY(9,1);
                    LCDstring("       ",6);
                    LCDGotoXY(0,1);
                    LCDstring(tAct,9);


                    //Luego de 3 segundos volvemos al inicio.
                    _delay_ms(3000);
                    estado = INICIO;
                    break;
        }
    }

uint8_t IntroducirNro(){
    // Hay nuevo dato en el buffer de teclado?
    if(KEYPAD_Scan(&key) == 0){
        return 0; // No Hay dato nuevo
    }

    // Se ha apretado una tecla, se espera 10seg mas para ver si se ingresa otra tecla
    if(State_call_count> 300)
    {
        // timeout, no se completo el ingreso de la clave, reiniciar
        State_call_count= 0;
        cantTecla=0;
        return 0;
    }
     //guardo tecla presionada
     if ((key>='0')&&(key <='9'))// verifico que es un digito
     {
     teclasPresionada[cantTecla++] = key;
     }
    // tengo las 2teclas?
    if(cantTecla==2) {
        cantTecla=0;
        return  DentroDelRango();//Verifico que sea un numero y que este dentro del rango
    } else
    return 0; //no tengo las dos teclas aún
}
```

Librería mai.h y código principal

```h
#ifndef MAIN_H_
#define MAIN_H_

#include "mef.h"
#include "Teclado4x4.h"
#include <avr/io.h>
#include "timer.h"
#include <avr/interrupt.h>



#endif /* MAIN_H_ */
```

```c
/*
 * Entregable2.c
 *
 * Created: 5/5/2024 21:38:01
 * Author : LENOVO
 */


#include "main.h"

int main(void)
{
 //Inicializacion de equipo

   LCDinit();//Inicializo LCD
   KEYPAD_Init();
   JUEGO_Init();
   timer_init();
    while (1)
    {
         JUEGO_Update();

    }
}
```