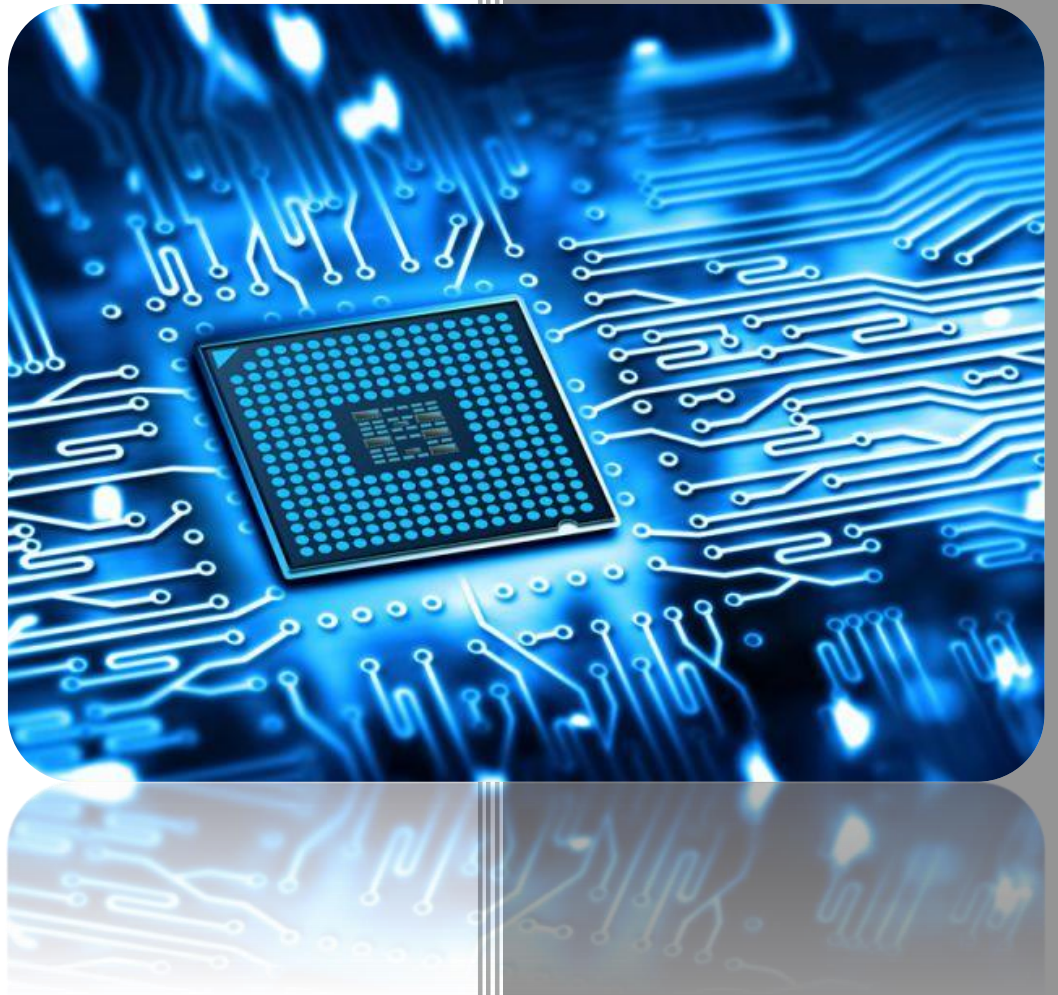


TP2

Circuitos digitales y microcontroladores



UNIVERSIDAD
NACIONAL
DE LA PLATA

Arreche, Cristian Carlos 01515/4
Blasco, Federico Matías 01678/4
20-5-2019

Cerradura electrónica

Índice

1. Problema.....	2
2. Introducción.....	2
2.1. Hardware.....	2
2.1.1. Conexionado display LCD.....	3
2.1.2. Conexionado teclado matricial.....	3
2.2. Software.....	4
2.2.1. Display LCD.....	4
2.2.2. Teclado matricial.....	4
2.2.3. Reloj y time-out.....	4
2.2.4. Máquina de estados.....	5
3. Resolución.....	6
3.1. Teclado matricial.....	6
3.2. Reloj y time-out.....	7
3.3. Máquina de estados.....	8
3.4. Programa principal.....	10
4. Anexos.....	11

1. **Problema**

Implementar con el MCU una cerradura electrónica. Para esto se dispone de un display LCD de 2 líneas, un teclado matricial 4x4 y el HS08. La implementación deberá hacerse con máquinas de estados temporizadas con el módulo RTC.

Requerimientos:

- a. Cuando el equipo se inicia deberá mostrar en la primer línea del LCD un reloj funcionando con el formato HH:MM:SS (horas, minutos y segundos) inicializado en la hora de compilación y en la segunda línea el estado de la cerradura "CERRADO".
- b. El sistema debe tener la clave numérica por defecto 2580 de manera de poder activar o desactivar la cerradura. Si el usuario presiona la clave correcta se mostrará en la segunda línea "ABIERTO" durante 5 seg y luego volverá automáticamente al estado por defecto. Si la clave es incorrecta se mostrará el estado "DENEGADO" durante 2 seg y luego volverá automáticamente al estado por defecto. No se mostrarán en el LCD las teclas presionadas.
- c. Para modificar la contraseña se deberá presionar la tecla 'D' en el estado por defecto. Se deberá mostrar el mensaje "clave actual:" y esperar que ingrese la clave actual. Si la clave es incorrecta se mostrará el estado "DENEGADO" durante 2 seg y luego volverá automáticamente al estado por defecto. Si la clave es correcta se deberá mostrar el mensaje "nueva clave:" y leer cada tecla presionada hasta que se presione 'D'. Se mostrará el mensaje "Fin ingreso nueva clave" 5 seg y se volverá al estado por defecto. A partir de aquí esta será la nueva clave para abrir la cerradura. Si desea cancelar se puede presionar '#' lo que interrumpe el ingreso y vuelve al estado por defecto sin guardar cambios. Cuando se presione una tecla numérica se deberá mostrar '*'.
- d. Las acciones de ingreso de claves por parte del usuario requieren un time-out de manera de cancelar la operación si la misma tarda demasiado y volver al estado por defecto. Agregar esta funcionalidad.

2. **Introducción**

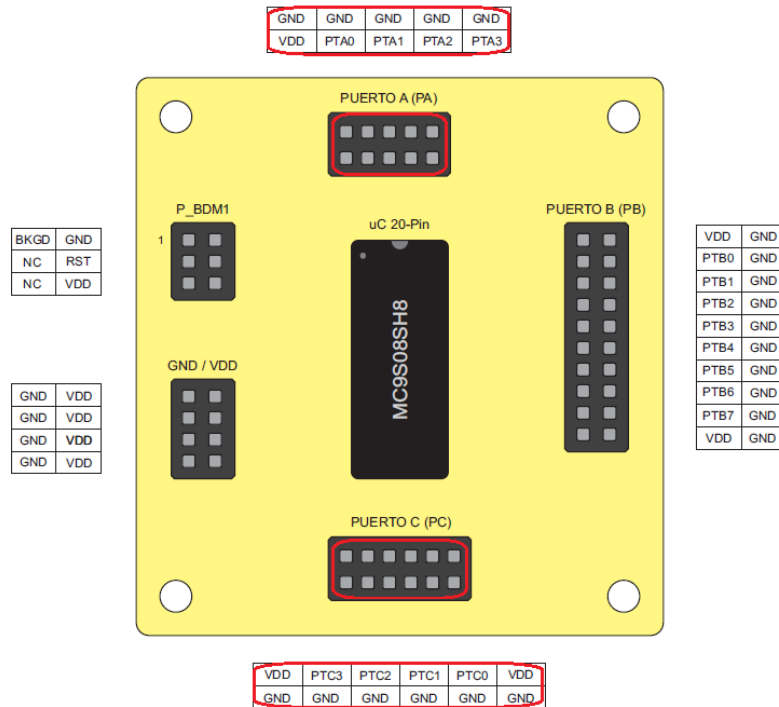
2.1. **Hardware**

Para la resolución del problema planteado anteriormente, contaremos con diferentes componentes de hardware, estos serán:

- Display LCD de 2 líneas, cada una de 16 caracteres.
- Teclado matricial de 4x4 caracteres.
- Microcontrolador MC9S08SH8 CPJ.

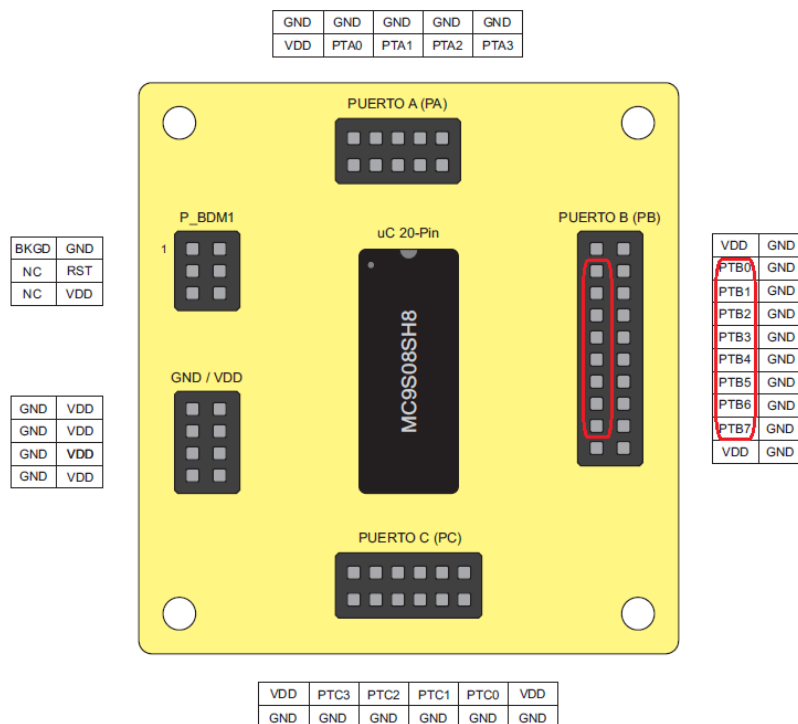
2.1.1. Conexionado display LCD

El conexionado del display se realizará tanto en el puerto A como en el puerto C de la placa, de la forma que se ve a continuación:



2.1.2. Conexionado teclado matricial

El conexionado del teclado matricial se realizará en el puerto B de la placa, como podemos ver a continuación:



2.2. Software

2.2.1. Display LCD

Para la utilización del display contamos con las librerías *lcd.c* y *lcd.h*, donde podemos encontrar una variedad de funciones que nos serán muy útiles a la hora de trabajar con el display. Algunas de ellas, y las más importantes son:

- *LCD_init (param1, param2)*, se encarga de inicializar el LCD para poder comenzar a utilizarlo, donde sus parámetros te permiten setear algunas configuraciones básicas del mismo.
- *LCD_pos_xy(posX, posY)*, como su nombre lo indica, permite escribir en la posición deseada del display.
- *LCD_write_char(carac)*, permite escribir un caracter en la posición indicada por *LCD_pos_xy()*, o por defecto, lo hará en la posición (0,0).
- *LCD_write_string(cadena)*, análogo al *LCD_write_char()*, pero nos permite escribir una cadena de caracteres.

2.2.2. Teclado matricial

A diferencia del display, no contamos con librerías para su procesamiento, por lo que la implementación interna dependerá de nosotros. Para esto se realizarán 2 funciones, donde sus objetivos serán:

- Una primera función que se encargará de hacer el barrido por la matriz de 4x4 del teclado, corroborando si fue o no presionada alguna tecla, y si así lo fuera habría que verificar cuál de ellas fue.
- Una segunda función que se encargará, cuando la función anterior detecte que una tecla fue presionada, de devolver el valor específico de dicha tecla presionada.

La implementación y resolución de estas funciones, las veremos en la sección 3. (*resolución*).

2.2.3. Reloj y time-out

Para implementar un reloj con el formato HH:MM:SS (horas, minutos y segundos), se utilizó una interrupción llamada *isrVrtc*, la cual es llamada por el mismo microcontrolador cada cierto período de tiempo (a determinar en la resolución).

Este período de tiempo puede ser regulado por nosotros según la “resolución” de tiempo que busquemos. En nuestro caso, como se trata de un reloj donde las unidades mínimas son los segundos, nos alcanzará con que las interrupciones sean cada 1 segundo.

Para el caso del “time-out”, el mismo consiste en limitar el tiempo de permanencia en un determinado estado sin realizar ninguna acción. De la misma manera, se hará uso de la interrupción mencionada anteriormente.

2.2.4. Máquina de estados

Para la resolución de la cerradura electrónica fue necesario contar con una máquina de estados finita, de manera de poder realizar un seguimiento del problema en cuestión, y así poder ir resolviéndolo por partes, es decir, dividir el problema en subproblemas más pequeños.

Los estados a tener en cuenta fueron determinados por el análisis del problema, donde, dependiendo de la etapa donde nos encontremos, serán las decisiones que se deberán tomar. Consideramos 6 estados diferentes, donde algunos de ellos pueden llegar a tener la misma salida.

Estos 6 estados describirán el sistema de la siguiente manera:

- 1) El estado por defecto, donde se mostrará el reloj funcionando y la cerradura se encontrará cerrada.
- 2) Ingreso de clave correcta, donde se mostrará el reloj funcionando y la cerradura pasará a estar abierta.
- 3) Ingrese de clave incorrecta, donde se mostrará el reloj funcionando y la cerradura notificará un acceso denegado.
- 4) Solicitud de cambio de clave, donde, en un primer paso, se pedirá que se ingrese la clave actual de la cerradura.
- 5) Cambio de clave, donde, una vez cumplido el paso anterior, se pedirá que se ingrese la nueva clave de la cerradura.
- 6) Fin cambio de clave, donde se notificará que la clave fue cambiada con éxito.

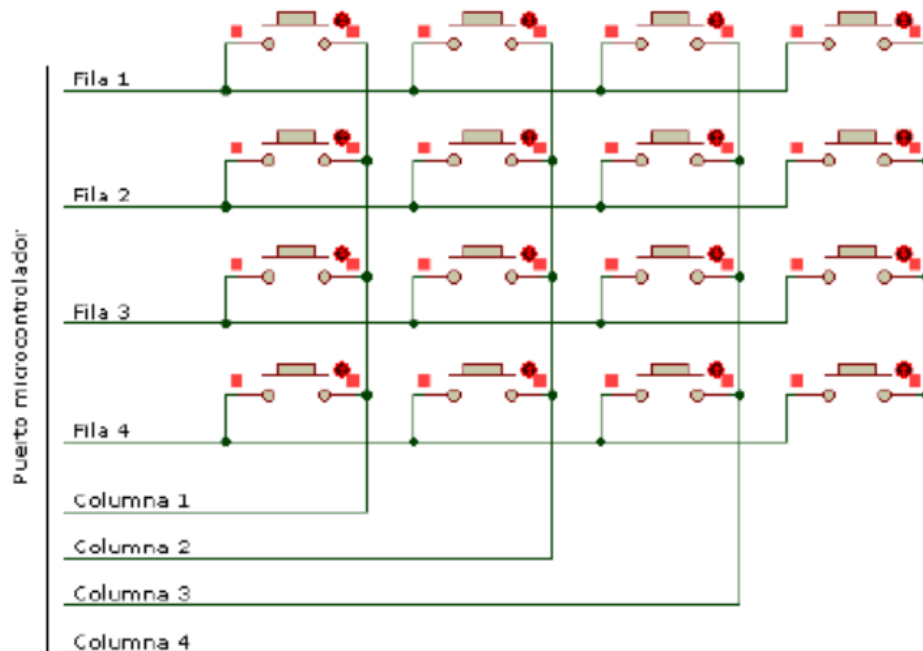
Cada uno de estos estados, tendrá distintas características específicas que, valga la redundancia, son las mencionadas en el problema.

La máquina de estados fue planteada como una máquina de moore, es decir, que la salida del sistema solo depende del estado actual.

3. Resolución

3.1. Teclado Matricial

Las filas y columnas del mismo están dispuestas de la siguiente manera:



Nuestro objetivo es detectar cuándo se presiona una tecla, y determinar qué tecla fue la presionada.

Las columnas del teclado son las entradas, configuradas con una resistencia pull-up interna que es activa en bajo (0 lógico). Por lo que, cuando se presione una tecla, será activado el bit correspondiente a la columna que pertenezca dicha tecla.

Por otro lado, las filas del teclado son las salidas, donde, para cada columna, se va induciendo un 0 a la vez hasta probar todas las combinaciones posibles.

Para la implementación del teclado, hay que tener en cuenta la conexión del mismo al puerto B (PTBD), como se ve a continuación:

C4	C3	C2	C1	F4	F3	F2	F1
----	----	----	----	----	----	----	----

Donde, como podemos observar, los 4 bits menos significativos corresponden a las salidas (filas), y los 4 bits más significativos corresponden a las entradas (columnas). De esta manera, la relación entre el valor cada una de las teclas y el valor del puerto B es la siguiente:

TECLA PRESIONADA	TECLA PRESIONADA	VALOR EN PTBD (8 bits)
1	C1F1	0b 1110 1110
2	C2F1	0b 1101 1110
3	C3F1	0b 1011 1110
A	C4F1	0b 0111 1110
4	C1F2	0b 1110 1101
5	C2F2	0b 1101 1101
6	C3F2	0b 1011 1101
B	C4F2	0b 0111 1101
7	C1F3	0b 1110 1011
...

Teniendo en cuenta todo lo mencionado anteriormente, contaremos con dos funciones, una primera función que será la encargada de hacer el barrido por las filas y columnas detectando cuándo una tecla sea presionada, y una segunda función que será la encargada de determinar qué tecla fue la que se presionó. Por lo tanto, los pseudocódigos para dichas funciones serán los siguientes:

- Función *KEYPAD_Scan(*char)*:
Para cada fila
Para cada columna
Si se presionó la tecla
Llamar función devolverTecla
Retornar verdadero
Retornar falso
- Función *devolverTecla(char)*:
Verificar valor de la variable recibida
Retornar la tecla que corresponda

A su vez, para el correcto funcionamiento del teclado, al tratarse de un teclado con el mismo problema de efecto rebote que todos los pulsadores (como ya se explicó en el TP1), se utilizará un delay de 20ms.

3.2. Reloj y time-out

Como mencionamos en la introducción, el funcionamiento tanto del reloj como del time-out depende exclusivamente del uso de la interrupción del flag RTC. Para lograr que la interrupción se realice cada un período de tiempo de 1 segundo, se debió configurar los ajustes del reloj e interrupciones de la siguiente manera:

Name	Value
Component name	RTC1
Device	RTC
▼ Settings	
▼ Clock settings	
Clock select	1-kHz low power oscillator
Internal ref. clock for perip	Disabled
External ref. clock for perip	Disabled
Prescaler	100
RTC modulo value	9
Period	1 s
▼ Interrupts	
Interrupt	Vrtc
RTC Interrupt	Enabled
ISR name	isrVrtc

Como podemos ver, la interrupción RTC se encuentra habilitada, y con los valores seteados de "Prescaler" y "RTC modulo value" se consiguió un período de interrupción de 1 segundo. La utilidad de esta interrupción, quedará más clara en la sección 3.4 (*programa principal*).

3.3. Máquina de estados

El diagrama de estados es el siguiente:

$S_0 \rightarrow$ Estado por defecto, "CERRADO"

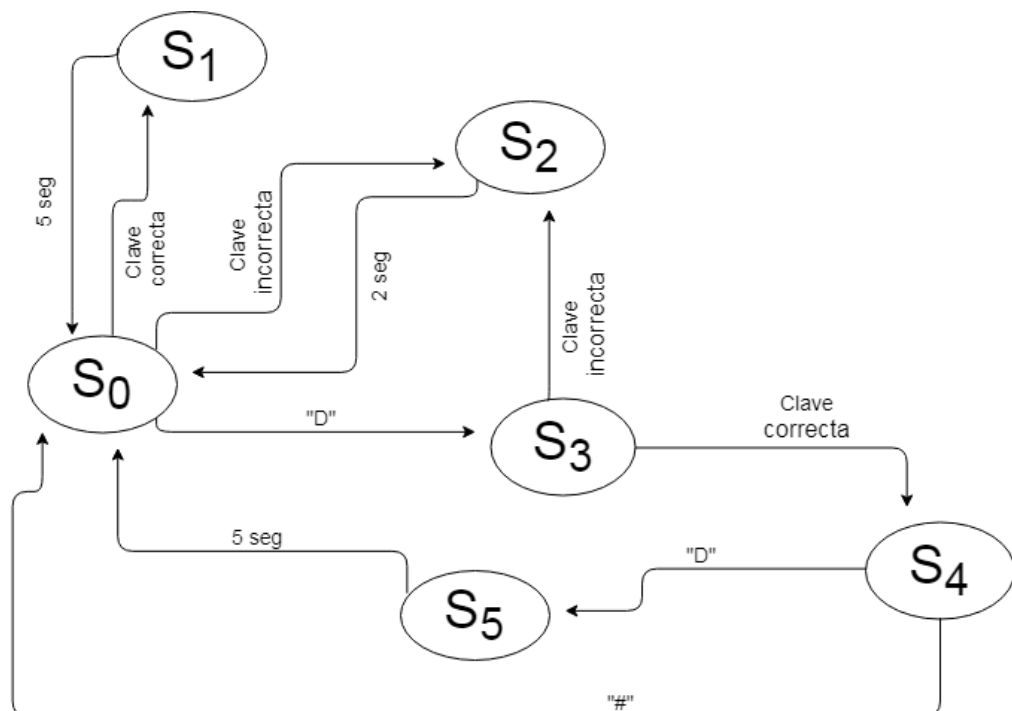
$S_1 \rightarrow$ Clave correcta, "ABIERTO"

$S_2 \rightarrow$ Clave incorrecta, "DENEGADO"

$S_3 \rightarrow$ Cambio de clave, "CLAVE ACTUAL: "

$S_4 \rightarrow$ Clave correcta, "NUEVA CLAVE: "

$S_5 \rightarrow$ Clave modificada, "FIN INGRESO NUEVA CLAVE"



A su vez, para los ingresos de claves por parte del usuario es necesario tener en cuenta un time-out, es decir, se contará con determinado tiempo para poder realizar el ingreso, pasado este tiempo, automáticamente se volverá al estado por defecto sin guardar ningún tipo de cambio. El tiempo de time-out fue fijado en 15 segundos.

La implementación será realizada mediante una máquina de estados temporizada con el módulo RTC mencionado anteriormente. A su vez, para la resolución, tenemos varias funciones:

- `cambiarEstado(state)`:
 - Se encarga de cambiar el estado al nuevo estado recibido por parámetro. Además, se encarga de volver a inicializar todas las variables en sus valores por defecto, como, por ejemplo, la variable que mantiene el tiempo de permanencia en un mismo estado (`timeInState`), y de limpiar la segunda línea de la pantalla LCD para no arrastrar salidas anteriores.
- `comprobarClave(unsigned char)`:
 - Comprueba que la clave ingresada mediante el teclado matricial sea correcta, y retorna si es correcta o no, una vez ingresadas las 4 teclas. De esta manera se evita que se pueda adivinar la clave mediante prueba y error. Luego de la comprobación de clave, esta misma función es la encargada de cambiar el estado actual al estado que corresponda.
- `timeOut()`:
 - En esta función, si el tiempo de permanencia en un mismo estado es igual al tiempo de timeout prefijado (15 segundos) se cambia del estado actual al estado por defecto (CERRADO).
- `setearClave(unsigned char)`:
 - Esta función es utilizada para el ingreso de una nueva clave para la cerradura. Esto solo es posible presionando la tecla "D" en el estado por defecto, y seguido a esto, ingresar correctamente la clave actual de la cerradura. Se recibe por parámetro cada una de las teclas presionadas, una a la vez, y en función del valor que tenga esta tecla, realiza distintos procesamientos:
 - Si se presiona la tecla '#' se cancela la operación, por lo tanto, se cambia el estado actual (`nuevaClave`) al estado por defecto, ya que significa que el usuario se arrepintió de cambiar la clave.
 - Si se presiona la tecla 'D' y la clave tiene un total de 4 caracteres, se guarda la clave y se pasa al estado "finNuevaClave".
 - Si la tecla ingresada es válida y no se llegó al límite de 4 caracteres de largo, se ingresa en la posición correspondiente del vector de caracteres que contiene el ingreso de la nueva contraseña.
 - Si no se ingresó a ninguna de las condiciones anteriormente mencionadas, quiere decir que ocurrió algún tipo de error. Estos pueden ser que se haya excedido el largo permitido de caracteres en la clave o que se haya presionada la tecla "*" (carácter inválido para la clave). En estos casos se vuelve al estado `cambiarClave`, para poder volver a intentarlo.

- ejecutarEstado():
 - Esta función es la base del sistema y es llamada constantemente por el programa principal. Se encarga de hacer el procesamiento y el llamado a los módulos correspondientes dependiendo del estado en el que nos encontremos en esa ejecución.

Además, la salida del LCD se ve reflejada por esta misma función, es decir, también imprime en el display el texto que corresponda según el estado actual.

3.4. **Programa principal**

El código utilizado para la resolución del problema, fue modularizado de la mejor manera posible, de manera de ser flexible ante cambios y tener un programa principal corto y entendible. El pseudocódigo para el programa principal es el siguiente:

```

Configurar variables, puertos y periféricos
Si hay interrupción del flag RTC
    Incrementar tiempo de permanencia en el estado
    Incrementar en un segundo el reloj
Procesamiento correspondiente según el estado actual
Chequear time-out para el estado
  
```

4. Anexos

4.1. Main.c

```
#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include "stdio.h"
#include "lcd.h"

#ifdef __cplusplus
extern "C"
#endif

typedef
enum{abierto, cerrado, denegado, cambiarClave, nuevaClave, finNuevaClave}state;
#define TIMEOUT 15

extern volatile unsigned char RTC_flag; //Extern porque se
inicializa en el init, volatile por ser global

void MCU_init(void); /* Device initialization function
declaration */

//Inicializacion de variables globales para el timer
unsigned char dig1='1';
unsigned char dig2='6';
unsigned char dig3='5';
unsigned char dig4='1';
unsigned char dig5='0';
unsigned char dig6='0';
unsigned char seg=0;
unsigned char min=51;
unsigned char hora=16;
state estado=cerrado;
unsigned char pass[] = "2580";
unsigned char passAux[] = " ";
unsigned short dimL = 4;
unsigned short indice = 0;
unsigned short indice2 = 0;
unsigned char probando;
unsigned short timeInState = 0;
unsigned short cond = 1;

////////////////////////////////////
////////////////////////////////////
void reloj(void){
    seg++;
    dig5=(seg/10)+'0';
    dig6=(seg%10)+'0';
    timeInState++;
    if(seg!=60){
        LCD_pos_xy(6,0);
        LCD_write_char(dig5);
        LCD_write_char(dig6);
    }
    else{
        LCD_pos_xy(6,0);
        LCD_write_char(dig5);
        LCD_write_char(dig6);
    }
}
```

```

        min++;
        seg=0;
        dig3=(min/10)+'0';
        dig4=(min%10)+'0';
        if(min != 60){
            LCD_pos_xy(3,0);
            LCD_write_char(dig3);
            LCD_write_char(dig4);
        }
        else{
            min=0;
            hora++;
            dig1=(hora/10)+'0';
            dig2=(hora%10)+'0';
            if(hora!=24){
                LCD_pos_xy(0,0);
                LCD_write_char(dig1);
                LCD_write_char(dig2);
            }
            else{
                hora=0;
            }
        }
    }
}

////////////////////////////////////
////////////////////////////////////
void delay(unsigned short n){//Funcion que recibe por parámetro
la cantidad de miliseg de delay
    unsigned short temp;
    unsigned short cant;
    /*Aca sabemos que la ejecucion del for equivale a 23 ciclos
de reloj
y que hacemos 8000 ciclos por milisegundo, eso equivale a mas
o menos
348 ejecuciones del for por milisegundo*/
    for(cant=n; cant>0; cant--){
        for(temp=348; temp>0; temp--);
    }
}

////////////////////////////////////
////////////////////////////////////
char devolverTecla(char keyPressed){
    switch(keyPressed){
        case 0b01110111:
            return 'D';
            break;
        case 0b01111011:
            return 'C';
            break;
        case 0b01111101:
            return 'B';
            break;
        case 0b01111110:
            return 'A';
            break;
        case 0b10110111:
            return '#';
            break;
    }
}

```

```

        case 0b10111011:
            return '9';
            break;
        case 0b10111101:
            return '6';
            break;
        case 0b10111110:
            return '3';
            break;
        case 0b11010111:
            return '0';
            break;
        case 0b11011011:
            return '8';
            break;
        case 0b11011101:
            return '5';
            break;
        case 0b11011110:
            return '2';
            break;
        case 0b11100111:
            return '*';
            break;
        case 0b11101011:
            return '7';
            break;
        case 0b11101101:
            return '4';
            break;
        case 0b11101110:
            return '1';
            break;
    }
}

////////////////////////////////////
////////////////////////////////////
char KEYPAD_Scan(char * key){
    unsigned short fila_act;
    unsigned short col_act;
    char f = 0b00000001;
    char c = 0b00000001;
    char cAux;
    char Aux;
    unsigned char PTBDaux = 0b00000001; //C4C3C2C1F4F3F2F1

    for(fila_act=0; fila_act<4; fila_act++){
        PTBD = ~(PTBDaux<<fila_act);
        for(col_act=0; col_act<4; col_act++){
            cAux = c<<4;
            Aux = cAux|f;
            Aux = ~Aux;
            if(PTBD == Aux){
                (*key) = devolverTecla(Aux);
                while(PTBD==Aux); //Para que no salga hasta que
no se suelte la tecla
                return(1);
            }
            c = c<<1;
        }
    }
}

```

```

        f = f<<1;
        c = 0b00000001;
    }
    return(0);
}

////////////////////////////////////
////////////////////////////////////
void cambiarEstado(state estadoNuevo){
    estado=estadoNuevo;
    timeInState=0;
    indice=0;
    indice2=0;
    LCD_pos_xy(0,1);
    LCD_write_string("                ");
    cond = 1;
}

////////////////////////////////////
////////////////////////////////////
void comprobarClave(unsigned char tecla){
    probando=pass[indice];
    if (probando!=tecla){
        cond=0;
    }
    if((cond)&&(indice!=(dimL-1))){//Si le pega a la tecla, y
    el indice sigue siendo valido
        indice++;
    }else{
        if(!cond){
            if(indice == 3){
                cambiarEstado(denegado);
                LCD_pos_xy(0,1);
                LCD_write_char(tecla);
            }
            indice++;
        }else{
            if(estado==cerrado){
                cambiarEstado(abierto);
            }else{//Estoy en el estado cambiar clave
                cambiarEstado(nuevaClave);
            }
        }
    }
}

////////////////////////////////////
////////////////////////////////////
void timeOut(){
    if(timeInState == TIMEOUT){
        cambiarEstado(cerrado);
    }
}

////////////////////////////////////
////////////////////////////////////
void setearClave(unsigned char tecla){

```

```

        if(tecla == '#'){
            cambiarEstado(cerrado);
        }else{
            if(tecla == 'D' && indice2==4){
                pass[0]=passAux[0];
                pass[1]=passAux[1];
                pass[2]=passAux[2];
                pass[3]=passAux[3];
                cambiarEstado(finNuevaClave);
            }
            else{
                if( (tecla!='*') && (indice2 != 4) ){
                    passAux[indice2]=tecla;
                    indice2++;
                }else{
                    LCD_pos_xy(0,1);
                    cambiarEstado(cambiarClave);
                }
            }
        }
    }
}

////////////////////////////////////
////////////////////////////////////
void ejecutarEstado(){
    unsigned char tecla;
    switch(estado){
        case cerrado:
            if(KEYPAD_Scan(&tecla)){
                if(tecla=='D'){
                    cambiarEstado(cambiarClave);
                }else{
                    comprobarClave(tecla);
                    delay(20);
                }
            }
            LCD_pos_xy(0,1);
            LCD_write_string("CERRADO");
            break;
        case abierto:
            LCD_pos_xy(0,1);
            LCD_write_string("ABIERTO");
            if(timeInState == 5){
                cambiarEstado(cerrado);
            }
            break;
        case denegado:
            LCD_pos_xy(0,1);
            LCD_write_string("DENEGADO");
            if(timeInState == 2){
                cambiarEstado(cerrado);
            }
            break;
        case cambiarClave:
            //ejecutarCambiarClave();
            LCD_pos_xy(0,1);
            LCD_write_string("Clave act:");
            if(KEYPAD_Scan(&tecla)){
                LCD_pos_xy(indice+10,1);
            }
        }
    }
}

```



```

        LCD_write_char('*');
        comprobarClave(tecla);
        delay(20);
    }
    break;
case nuevaClave:
    //ejecutarNuevaClave();
    LCD_pos_xy(0,1);
    LCD_write_string("Nueva clave:");
    if(KEYPAD_Scan(&tecla)){
        setearClave(tecla);
        LCD_pos_xy(indice2+11,1);
        LCD_write_char('*');
    }
    break;
case finNuevaClave:
    //ejecutarFinNuevaClave();
    LCD_pos_xy(0,1);
    LCD_write_string("FIN NUEVA CLAVE");
    if(timeInState == 5){
        cambiarEstado(cerrado);
    }
    break;
    }
}

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

void main(void) {

    PTBDD = 0x0F;
    PTBPE = 0xF0;//Seteo la entrada PTB con una resistencia pull
up interna

    MCU_init(); /* call Device Initialization */
    LCD_init(_2_LINES|DISPLAY_8X5,DISPLAY_ON|CURSOR_ON);

    LCD_pos_xy(0,0);
    LCD_write_char(dig1);
    LCD_write_char(dig2);
    LCD_write_char(':');
    LCD_write_char(dig3);
    LCD_write_char(dig4);
    LCD_write_char(':');
    LCD_write_char(dig5);
    LCD_write_char(dig6);

    for(;;) {

        if(RTC_flag == 1){
            RTC_flag=0;
            reloj();
        }//Termina el chequeo de la hora
        ejecutarEstado();
        timeOut();
    }
}

```

```
    }  
}
```

4.2. **MCUinit.c**

```
//Se incluyen solamente las líneas modificadas  
  
/* User declarations and definitions */  
/* Code, declarations and definitions here will be preserved  
during code generation */  
volatile unsigned char RTC_flag=0;  
/* End of user declarations and definitions */  
  
__interrupt void isrVrtc(void)  
{  
    /* Write your interrupt code here ... */  
    RTC_flag=1;//Se cumplio el plazo de 100ms  
    RTCSC_RTIF=1;//Se borra en un solo paso  
}  
/* end of isrVrtc */
```