

**PROYECTO  
SISTEMA  
ELECTRÓNICO  
DIGITAL**

*Control de una piscina*

**Universidad de Granada**

**Curso 2010/2011**

***Emilio Martínez Pérez***

## ***Índice***

- I. Descripción global del proyecto.....***
- II. Descripción hardware....***
- III. Descripción software....***
- IV. Funcionalidad....***
- V. Problemas encontrados y soluciones aportadas....***
- VI. Presupuesto....***
- VII. Planificación....***

## Descripción global del proyecto

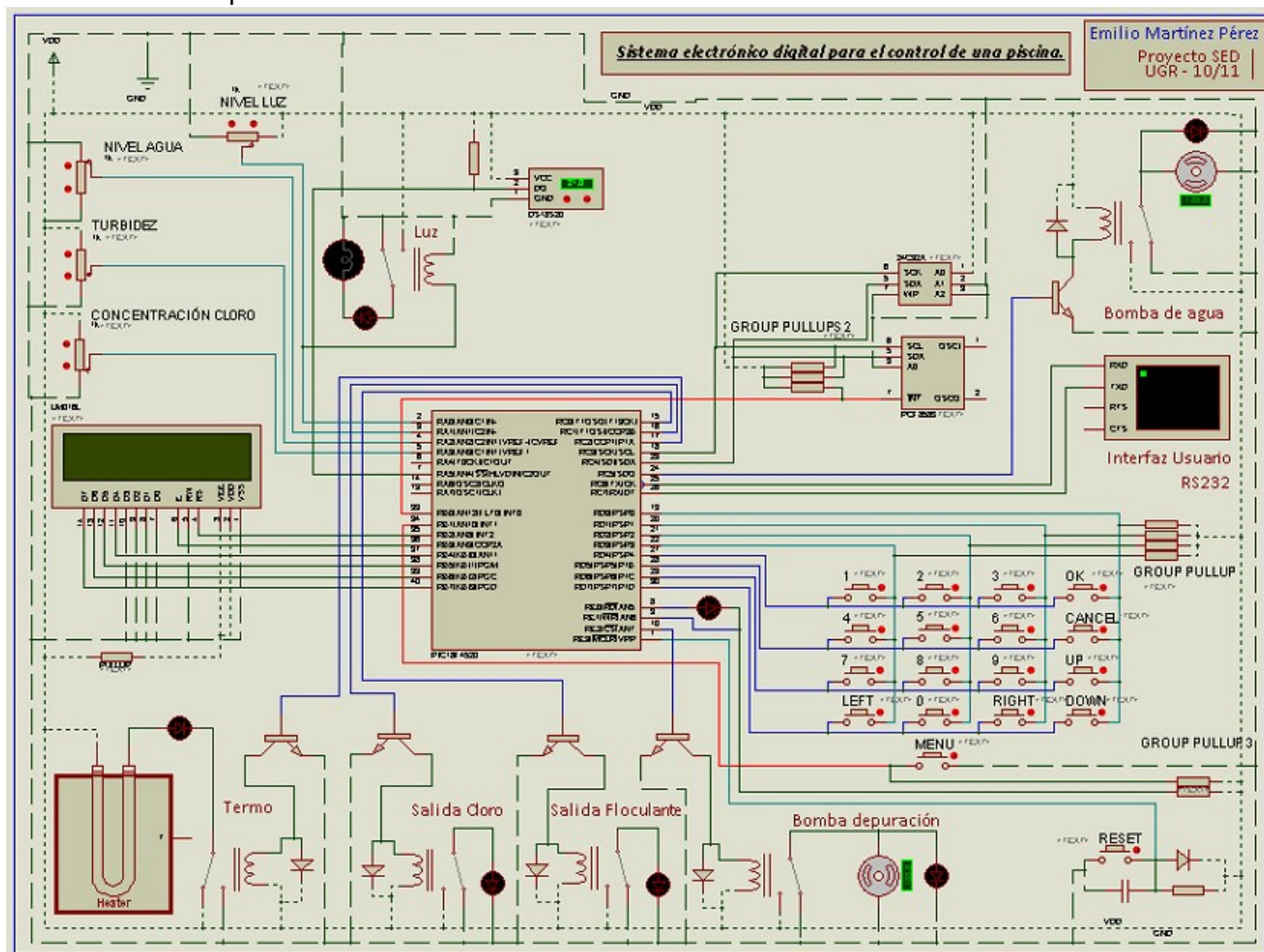
Este año, el proyecto de la asignatura ha consistido en la realización del control de una piscina cubierta mediante un sistema electrónico digital. Los requisitos mínimos pedidos a implementar por el sistema se cuentan a continuación. Se pide que el sistema sea capaz de controlar la temperatura del agua de la piscina y el proceso de depuración de la misma, siendo posible que el usuario pueda en cualquier momento modificar los valores a controlar, tales como el ciclo de depuración (hora de encendido y apagado de la bomba de depuración), niveles máximos de cloro y turbidez del agua y temperatura del agua.

Para asegurar que estos valores definidos por el usuario sean debidamente conservados ante algún problema, añadimos una memoria EEPROM externa donde éstos serán duplicados. Además, se ha agregado características a las mínimas pedidas. Tales han sido un sensor de medición de nivel de agua, control automático de llenado de la piscina en función del nivel de agua medido, sensor de luz que acciona un interruptor si la cantidad de luminosidad en la sala es menor a un mínimo, circuito de reinicio del sistema y posibilidad de control del sistema mediante un dispositivo conectado mediante un cable serie RS232.

Además, incluye una interfaz de usuario sencilla, compacta y segura para la utilización del sistema y modificación de parámetros protegida por contraseña.

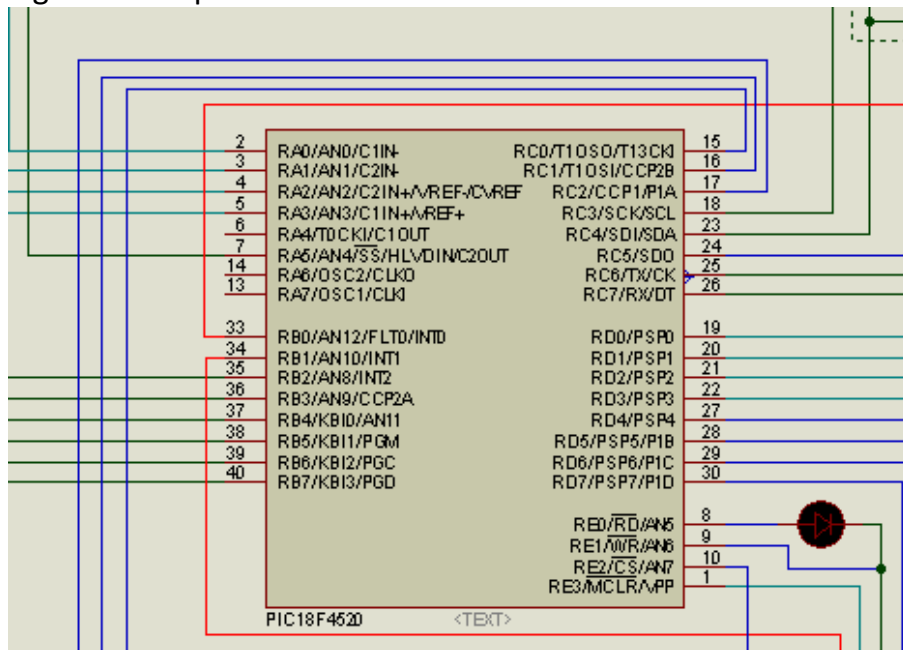
## Descripción hardware

Un vistazo rápido al sistema:



Desgranamos en los distintos componentes que lo conforman.

El sistema está gobernado por un **PIC 18F4520**



Este es el corazón del sistema controlador de la piscina cubierta. Las patillas están conectadas de tal forma:

#### *Puerta A:*

Las entradas de RA0 a RA3 se corresponden a entradas analógicas para la toma de valores de concentraciones de cloro, floculante y el nivel de agua y luz. Estos valores son tomados como valores continuos de corriente DC entre 0 y 5V. Es necesaria una codificación correcta realizada por software.

El pin RA5 está asignado para dispositivos one-wire. En nuestro caso el sensor de temperatura DS18S20.

#### *Puerta B:*

La puerta B tiene asignador los pines del RB2 al RB7 para el control de la pantalla LCD LM016L.

Los pines RB0 y RB1 están asignados para la ejecución de interrupciones externas, tanto de la alarma del reloj PCF8583 como para le botón de menú.

#### *Puerta C:*

La asignación de pines de la puerta C queda tal cual se indica a continuación.

RC0 a RC2: Salida para el control del relé de la activación del termo, bomba de salida del cloro y la bomba de salida del floculante.

RC3 y RC4: Conectores SCL y SCA para el reloj externo PCF8583 y memoria EEPROM externa 24C32A.

RC5: Salida de control del relé de la bomba de llenado de agua.

RC6 y RC7: Conectores TX y RX para la interfaz de usuario RS232.

#### *Puerta D:*

De RD0 a RD3 están conectadas las entradas del teclado matricial construido mediante Interruptores. Dicho teclado dispone de 16 teclas (4x4) distribuidas de tal forma. Valores numéricos del 0 al 9, Tecla de Ok, Cancel, Up, Down, Left y Right para poder desplazarnos por el menú de usuario.

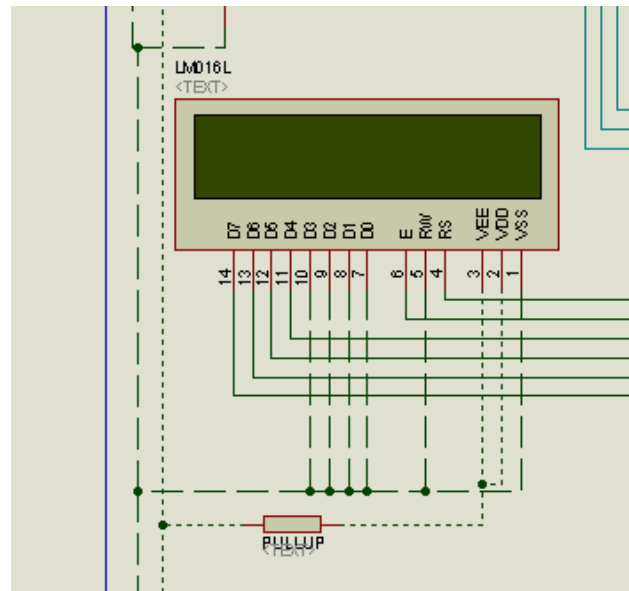
Los pines RD4 a RD7 son usados como la salida de tensión del teclado.

#### *Puerta E:*

La puerta E dispone de únicamente 4 pines, que quedan asignados como sigue:

RE0 y RE1 son usados para la señalización del estado del sistema mediante un LED con diferentes colores. Dependiendo del valor de ambas puertas tomará un color u otro. Siendo Verde en estado normal y de color Rojo cuando está en estado de ahorro de energía.

#### **Pantalla LCD LM016L**



Pantalla LCD de 2 líneas x 16 caracteres para visualización de mensajes, con campos fijos y variables que el usuario debe conocer. Las conexiones de ésta sirven para:

RS conectada a RB2 del PIC sirve para indicar el tipo de acceso al dispositivo (1: entrada a registro de datos; 0: entrada de instrucciones).

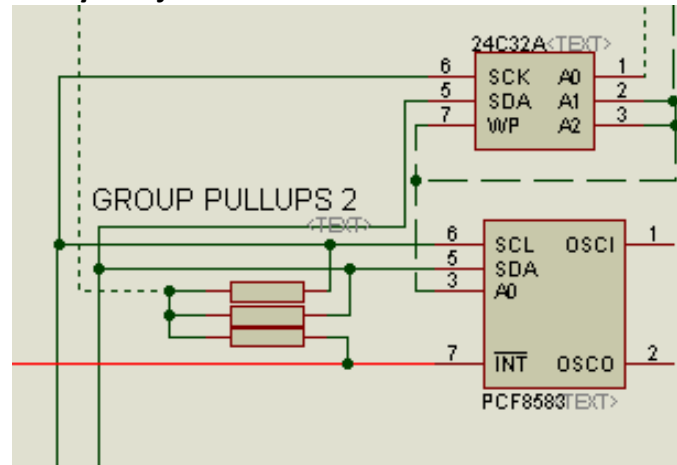
E conectada a RB3 del PIC sirve para habilitar el LCD

R/W = 0 conectada a tierra, ya que el modo será siempre de escritura.

D0, D1, D2, D3, también conectadas a tierra.

De D4 a D7 del LCD conectadas del RB4 al RB7 del PIC para la escritura de datos.

## EEPROM externa 24C32A y reloj externo PCF8583

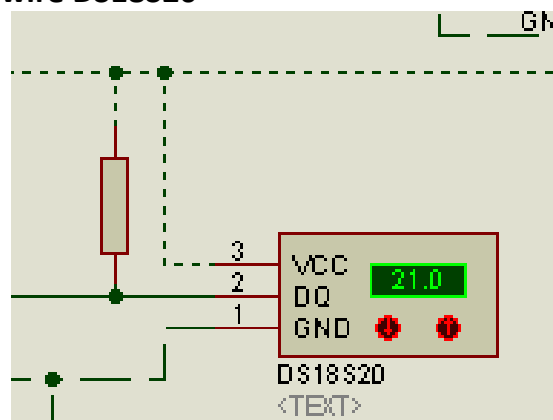


La memoria externa EEPROM nos permite tener una 'copia de seguridad' de los parámetros configurados por el usuario. Y que permitiría una recuperación de estos mismos en caso de un fallo de la memoria interna del PIC. Es una memoria EEPROM de 32K y sus pines SCK y SDA está conectado a sus correspondientes en el PIC, y también a la red eléctrica con resistencias de PULL-UP y WP a tierra.

Las conexiones Vcc y Gnd son los pines A0, A1 y A2.

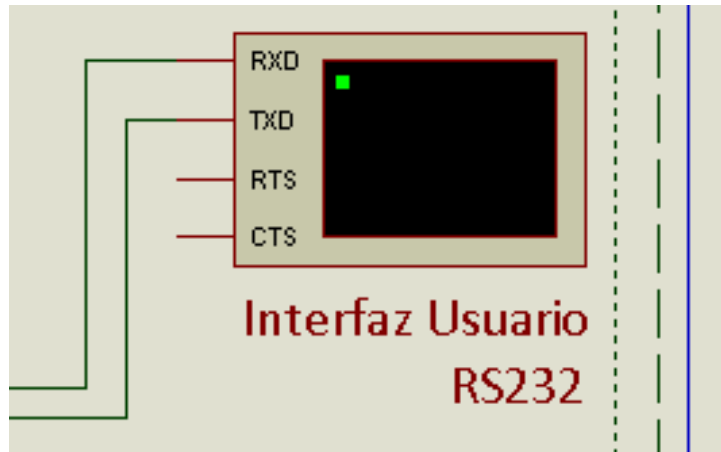
El reloj PCD8583 está conectado a los mismos cables que la memoria EEPROM esto se debe a que el acceso a ambos dispositivos se realizará de forma controlada por turnos pudiendo optimizar así las patillas de PIC. El módulo RTC se utiliza para saber la hora y el día (introduciéndola previamente) y así poder guardar fecha/hora del sistema. Este módulo dispone de una fuente de corriente externa para evitar la la pérdida de datos en caso de que se produzca un fallo de corriente. Las conexiones del módulo es SCL y SDA que van conectadas a sus correspondientes en el PIC (que son las mismas que la EEPROM), y también alimentadas por la red eléctrica con resistencias de PULL-UP. A0 va a tierra y OSC1 y OSC0 están 'desnudos'. INT con resistencia de PULLUP, es la encargada de realizar la salida de interrupcion de RB0 cuando la alarma salta.

## Sensor temperatura one-wire DS18S20



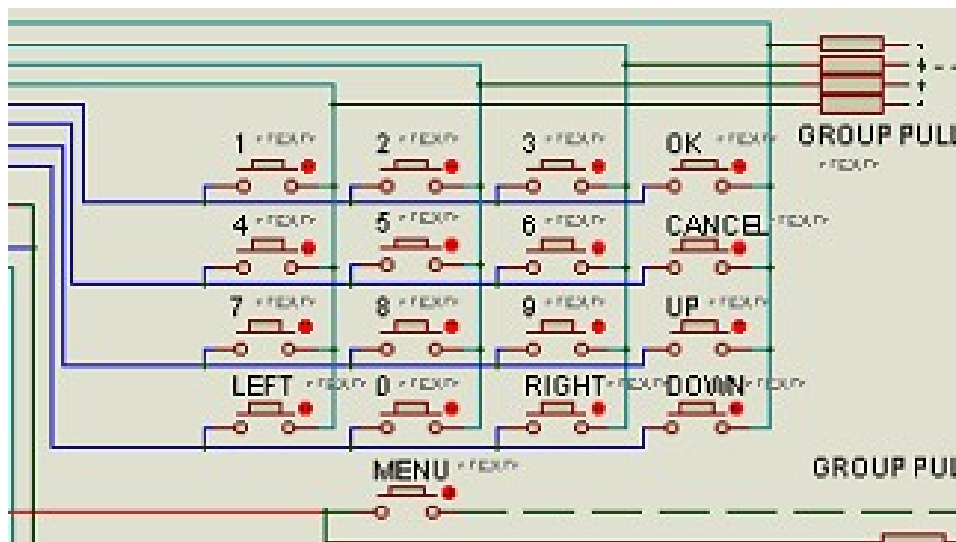
Es un sensor de temperatura one-wire. En este caso se utiliza para detectar la temperatura del agua de la piscina. Su conexión al PIC se realiza mediante un solo cable (esa es la principal característica del protocolo one-wire). También está conectado a la red eléctrica mediante resistencia de PULL-UP. El pin GND está a tierra.

## Interfaz usuario RS232



Se ha introducido la posibilidad de tener una interfaz de usuario alternativa a la de la pantalla LCD. Esta es implementada mediante un cable serie RS232 conectando de forma alterna los puertos RX-TX del PIC y ordenador correspondiente.

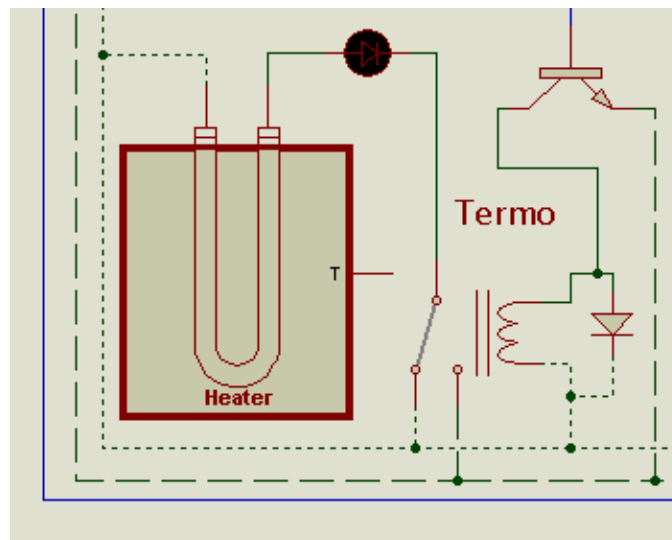
## Teclado matricial



Se ha construido un teclado matricial (4x4) para introducir los datos, conectada al puerto D tal como se ha explicado en el apartado del PIC. Las teclas han sido asignadas tal y como se observa en la imagen.

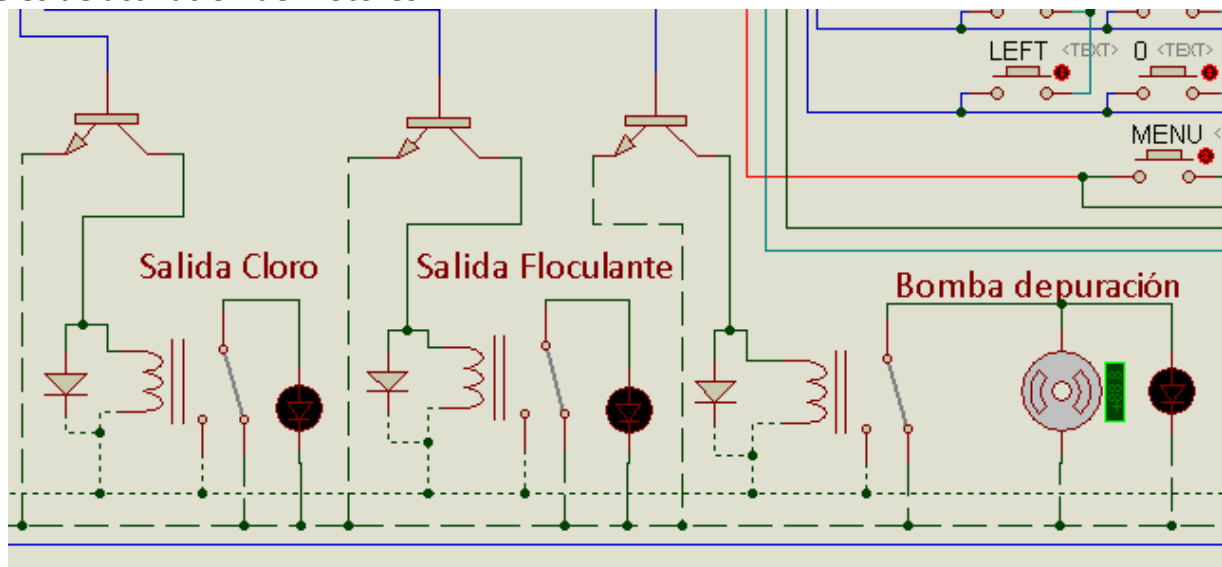
Además, el botón menú permite la activación del menú del sistema. La detección del pulsado del botón se realiza mediante interrupciones.

## Termo



El termo se activará cuando la temperatura del agua tomada por el sensor sea menor a la establecida. El mecanismo de activación es bien sencillo. El PIC manda una señal en alta (en el pin asignado para ello) al BJT que activa un relé conectado a la resistencia a usar. Se activará un LED para indicar que se está calentando el agua.

## Relés de activación de motores



Al igual que en el caso del termo, el PIC lanza una señal en alto en el pin correspondiente en función de la actividad a realizar. En el caso de la bomba de depuración, ésta se pondrá en marcha cuando la alarma del RTC crea una interrupción y el PIC la procese. El tiempo que esté activada la bomba dependerá del tipo de depuración seleccionada.

La salida de cloro y floculante se activarán al final del proceso de depuración un tiempo determinado que dependerá de los niveles de concentración de cloro y turbidez medida por los sensores.



Están suficientemente comentados para entender el funcionamiento de cada parte.

```
/* \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*-----*/\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ * \\
```

/\* \\ SISTEMAS ELECTRÓNICOS DIGITALES UNIVERSIDAD DE GRANADA 2010 - 2011 // \* \\

```
/* \\\\\\\\\\\ SISTEMA ELECTRÓNICO DIGITAL DE UNA PISCINA CUBIERTA ////////// * \\
```

```
/* \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*---*/\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ * \\
```

```
/* \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ EMILIO MARTÍNEZ PÉREZ ////////// * \\
```

```
/* \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\ * \\
```

```
float cloro, turbidez, temperatura, nivelAgua, nivelLuz; //variables para resultaDOs de conversiones
```

A/D, se supone ADC = 8 en la directiva DEVICE del archivo 18F4520.h

```
void comprobarNiveles();
```

```
void mostrar();
```

```
/////////////////////////////////FUNCIONES USUALES////////////////////////////////
```

```
void mostrar(){
```

```
    strcpy(weekday, semana[dt.weekday]); // ptr=strncpy (s1, s2, n) Copy up to n characters s2->s1
```

```
    //Si el dígito pasado por dt.XXXX es de una sola unidad, lo detectamos y colocamos un cero antes de la difra.
```

```
    //Si no estamos gestionando el sistema por la consola
```

```
    if(gestionMenuRS232==FALSE){
```

```
        //Escribimos la fecha
```

```
        printf("Fecha: %s, ", weekday);
```

```
        if (dt.day<10) printf("0%u/", dt.day);
```

```
        else printf("%2u/", dt.day);
```

```
        if (dt.month<10) printf("0%u/", dt.month);
```

```
        else printf("%2u/", dt.month);
```

```
        if (dt.year<10) printf("0%u", dt.year);
```

```
        else printf("%2u", dt.year);
```

```
        //Escribimos la hora
```

```
        printf(" - Hora: ");
```

```
        if (dt.hours<10) printf("0%u:", dt.hours);
```

```
        else printf("%2u:", dt.hours);
```

```
        if (dt.minutes<10) printf("0%u:", dt.minutes);
```

```
        else printf("%2u:", dt.minutes);
```

```
        if (dt.seconds<10) printf("0%u \n\r", dt.seconds);
```

```
        else printf("%2u \n\r", dt.seconds);
```

```
        printf("Temperatura agua: %3.1f - ", temperatura);
```

```
        if(temperatura >= (tempUs+2)) printf("Agua demasiado caliente ");
```

```
        else if( temperatura >= tempUs && temperatura < (tempUs+2)) printf("Temperatura optima. ");
```

```
        else printf("Temperatura baja. ");
```

```
        printf("CLORO: %3.1f mgr/l, TURBIDEZ: %3.1f kpart/l\n\r", cloro, turbidez);
```

```
        fputc('\n',FT232);
```

```
        fputc('\r');
```

```
    }
```

```
}
```

```
//Si la temperatura del sistema es menor que la establecida, se enciende la caldera. Sino, no.
if(temperatura<=TempUs) output_bit(PIN_C2, 1);
else {output_bit(PIN_C2, 0);}

//Si el nivel del agua es menor al establecido, se encienden los motores y se llena la piscina.
if(nivelAgua>conversionInt2NivelAgua(NivelUs)){output_bit(PIN_C5, 1);}
else {output_bit(PIN_C5, 0);}

//Se comprueban los niveles de cloro y turbidez para actuar en consecuencia en el depurado
//Siempre y cuando ya no se esté depurando
if(!depurado){
    //Si el cloro es menor al recomendado se introduce a la piscina una cantidad proporcional al mismo
    if(cloro<conversionInt2Cloro(CloroUs)){
        tiempoCloro=(conversionInt2Cloro(CloroUs)-cloro)*10/1.4;
    }

    //Si el nivel de turbidez es mayor, se introduce a la piscina una cantidad proporcional de floculante
    if(turbidez>conversionInt2Turbidez(TurbidezUs)){
        tiempoFloculante=(turbidez-conversionInt2Turbidez(TurbidezUs))*10/4;
    }
}
}
```

```
#int_ext
```

```
//bajamos el flag de alarma
```

//Al producirse la alarma del RTC se provoca una interrupción.

```
//Comprobamos cuando tiempo se quiere depurando.
```

```
//aux=read_eeprom(13);
```

```
delay_ms(20);
```

```

if (aux==0) tiempo_depurado=200;
else if (aux==1) tiempo_depurado=50;
else tiempo_depurado=10;
depurado=TRUE;  output_bit(PIN_E2, 1);

```

```

}

```

```

#int_ext1
menu_interrupcion(){
    clear_interrupt(int_ext1);
    if (!gestionmenu) bajoconsumo=FALSE; gestionmenu=TRUE;
    if (gestionmenu) return;
}

```

```

////////////////////INTERUPCCIONES INTERNAS////////////////////

```

```

//Rutina de Interrupcion del timer0, para llevar el contador para el modo sleep y el tiempo de
funcionamiento del procesado de la piscina.

```

```

//Se produce un desbordamiento cada 8,5 segundos aproximadamente.

```

```

#INT_TIMER0

```

```

rutina_timer0(){///COMPLETAR
    if (tiempoinactividad >= (maximoinactividad-5)) bajoconsumo=TRUE;
    tiempoinactividad += 8.5;
    if(depurado && !bajoconsumo){
        tiempoinactividad=0;
        tiempo_depurado -= 8.5;
        if(tiempo_depurado<0){
            output_bit(PIN_E2, 0);
            //PCF8583_write_byte(PCF8583_CTRL_STATUS_REG,
PCF8583_START_COUNTING_WITH_ALARM);
            depurado=FALSE;
        }
    }
}
}
}

```

```

#INT_TIMER1

```

```

tratamiento()

```

```

{
    if (contador_timer1 == 237){
        contador_timer1=0;
    }

    //Si el timer se desborda señalamos para que el loop principal haga las mediciones
    if (contador_timer1 == 0 ) medicion = TRUE;

    //Contador del tiempo de cloro y floculante
    if (depurado && tiempo_depurado<20){
        if(tiempoCloro >0) output_bit(PIN_C1,1);
        else output_bit(PIN_C1,0);
        if(tiempoFloculante>0) output_bit(PIN_C0,1);
        else output_bit(PIN_C0,0);

        tiempoCloro-=0.23;
        tiempoFloculante-=0.23;

    }
    contador_timer1++;
    restart_wdt ( );
    set_timer1 ( ~25000 ); //Repone TMR1 con el complemento a uno de 25000 [( 2**16 ) - 1 - 25000]
}

```

///FUNCION MAIN///

```

void main()

```

```

{

```

```

    //CONFIGURACIÓN DE LOS PINES

```

```

    //PUERTO A

```

```

    setup_adc_ports( AN0_TO_AN3 ); //RA0 y RA1 entradas analógicas (ver 18F4520.h)

```

```

    //PUERTO B 00000011

```

```

    set_tris_b(0x03); // Fijamos RB0 - RB1 como entrada y RB2-RB7 salida

```

```

    //PUERTO C 00000000

```

```

    set_tris_c(0xFF); // Fijamos RC0 - RC7 como entrada.

```

```

//PUERTO D 00001111
set_tris_d(0x0F); // Fijamos RD0 - RD3 entradas, RD4-RD7 salidas
//PUERTO E 1100
set_tris_e(0x0C); //Fijamos RE0 - RE1 como salida y RE2 - RE3 como entrada

setup_adc ( adc_clock_div_32 ); //Ajusta tiempo de conversión de cada bit

/*
|=====| Ponemos valores lógicos en
| E1 | E0 | Luz | Modo | las puertas que gobiernan
|=====| en led siguiendo la tabla.
| 0 | 0 | Apagado | Apagado |
|-----|-----|
| 0 | 1 | Verde | Normal |
|-----|-----|
| 1 | 0 | Roja | Económico |
|-----|-----|
| 1 | 1 | Apagado | Apagado |
|=====|=====| */
output_bit(PIN_E1,0);
output_bit(PIN_E0,1);
output_bit(PIN_E2,0);


//Mensaje inicial
inicioSistema( );
ajuste_ini();
//Comprobamos el estado de la memoria EEPROM
comprobarMemoria( );
//Iniciamos el RTC PCF8583
iniciarRTC();
//Activamos la alarma del RTC para el ciclo diario de depurado
iniciarAlarmaRTC();
//Tomamos de la eeprom los niveles configurados
actualizarNivelesUs();

maximoInactividad=TiempoUs;

//Mensaje para conexión serie

```

```
printf ( "Comenzamos el log.\n\r" );
printf ( "Presiona q para finalizar el log.\n\r" );
printf ( "La toma de medidas se produce cada minuto, tenga paciencia.\n\r" );
```

```
enable_interrupts ( int_timer1 ); //Activa interrupción del Timer1
enable_interrupts ( int_timer0 ); //Activa interrupción del Timer0
```

```
//Interrupciones Externas
```

```
enable_interrupts(int_ext); // habilitacion de interupcion externa
ext_int_edge(H_TO_L); // interrupcion para comparacion flanco de subida
enable_interrupts(int_ext1); // habilitacion de interupcion externa
ext_int_edge(H_TO_L); // interrupcion para comparacion flanco de subida
```

```
clear_interrupt(int_ext);
clear_interrupt(int_ext1);
```

```
/* El TMR1 trabaja con oscilador interno y un preescaler de 1:8. Con un reloj
de 8 MHz ( 0, 5us por instrucción ), el TMR1 deberá desbordarse cada 25000 cuentas
para producir una interrupción cada 0.1s ( 25000 * 8 * 0, 5us = 100000uS = 0.1" ) */
set_timer1 (~25000); //Carga TMR1 con el complemento a 1 de 25000
setup_timer_1(T1_INTERNAL | T1_DIV_BY_8); //TMR1 ON y 1:8
setup_timer_0(RTCC_INTERNAL);
enable_interrupts(global);
setup_counters(RTCC_INTERNAL,WDT_TIMEOUT);
```

```
MAIN:
```

```
in_rs232='a';
while(1/!(in_rs232 == 'e') || in_rs232 != 'm'){
    //Si el timer se ha desbordado se realiza la medición de niveles.
    if (medicion){
        temperatura=medirTemperatura();
        cloro=medirNivelCloro();
        turbidez=medirNivelTurbidez();
        nivelAgua=medirNivelAgua();
        nivelLuz=medirNivelLuz();
        comprobarNiveles();
        mostrar();
        medicion=FALSE;
    }
}
```

```

//APARTADO PARA EL MODO SLEEP()
LOOP_BAJOCONSUMO:
//Comprobar el estado de bajo consumo para entrar en modo económico
if(bajoconsumo){
    //Indicamos por el led que estamos en modo ahorro de energía (Color rojo)
    output_bit(PIN_E0, 0);
    output_bit(PIN_E1, 1);
    aux=1;
    //Indicamos por pantallas
    printf("<<<MODO BAJO CONSUMO>>>\n\r");
    lcd_send_byte(0,LCD_CLEAR);
    while(bajoconsumo && !gestionmenu && !depurado) {//mientras no se indique lo contrario
seguiamos en el modo de bajo consumo

        //Actualizamos la información del LCD
        printf(lcd_putc, "<<BAJO CONSUMO>>");
        lcd_gotoxy(2,2);
        printf(lcd_putc, "%.3.1f%cC ", temperatura, 223);
        lcd_gotoxy(11,2);
        if (dt.hours<10) printf(lcd_putc, "0%u:", dt.hours);
        else printf(lcd_putc, "%2u:", dt.hours);
        if (dt.minutes<10) printf(lcd_putc, "0%u", dt.minutes);
        else printf(lcd_putc, "%2u", dt.minutes);
        //Por el perro guardián, el micro se despertará cada 65,5 segundos (65536 ms =
16384*4ms).
        sleep();
        //Aprovechamos este desbordamiento del WDT para tomar algunas medidas y luego volver
a llamar a sleep()
        //Consultamos la hora
        PCF8583_read_datetime (&dt);
        //Ponemos el contador del timer1 a cero para que simular un desbordamiento para la toma
de medidas.
        contador_timer1=0;
        //Añadimos un pequeño retardo para que el timer1 pueda realizar las sentencias
        delay_ms(500);

        //Si está el depurado activo
        if(aux!=1){
            if (depurado) tiempo_depurado -= 65,5;

```



```

    }
    if (tiempo_depurado<0 && depurado){
        output_bit(PIN_E2, 0);
        depurado=FALSE;
    }
}
//Al salir del sleep ponemos verde el led, el indicador de funcionamiento 'normal'.
output_bit(PIN_E0, 1);
output_bit(PIN_E1, 0);
//Ponemos el contado de inactividad a cero
tiempoinactividad=0;
}

```

LOOP\_GESTIONMENU:

```

//Comprobamos si se ha pedido entrar en menú
if(gestionmenu) {
    menu();
    tiempoinactividad=0;
    gestionmenu=FALSE;
    actualizarNivelesUs();
}

```

//HORA

```

lcd_send_byte(0, LCD_CLEAR);
PCF8583_read_datetime (&dt); //Lectura del RTC

```

//Si el dígito pasado por dt.XXXX es de una sola unidad, lo detectamos y colocamos un cero antes de la difra.

```

strcpy(weekday, semana[dt.weekday]); // ptr=strncpy (s1, s2, n) Copy up to n characters s2->s1

```

//Escribimos la fecha

```

printf(lcd_putc, "%s,", weekday);
if (dt.day<10) printf(lcd_putc, "0%u/", dt.day);
else printf(lcd_putc, "%2u/", dt.day);
if (dt.month<10) printf(lcd_putc, "0%u/", dt.month);
else printf(lcd_putc, "%2u/", dt.month);
if (dt.month<10) printf(lcd_putc, "0%u", dt.year);
else printf(lcd_putc, "%2u", dt.year);

```

//Ahora la hora.

```

lcd_gotoxy(1,2);
printf(lcd_putc, "Hora: ");

```

```
if (dt.hours<10) printf(lcd_putc, "0%u:", dt.hours);
else printf(lcd_putc, "%2u:", dt.hours);
if (dt.minutes<10) printf(lcd_putc, "0%u:", dt.minutes);
else printf(lcd_putc, "%2u:", dt.minutes);
if (dt.seconds<10) printf(lcd_putc, "0%u", dt.seconds);
else printf(lcd_putc, "%2u", dt.seconds);
delay_ms(2000);
```

```
if (kbhit(FT232))in_rs232=fgetc();
if (in_rs232=='m' || in_rs232=='e') break;
if (gestionmenu) goto LOOP_GESTIONMENU;
if (bajoconsumo) goto LOOP_BAJOCONSUMO;
```

```
//TEMPERATURA
```

```
lcd_send_byte (0, LCD_CLEAR);
printf (lcd_putc, "TEMP: %3.1f ", temperatura);
lcd_putc (223);
lcd_putc ("C ");
lcd_gotoxy (1, 2);
if(temperatura >= (tempUs+2)) printf(lcd_putc, "Agua caliente  ");
else if( temperatura >= tempUs && temperatura < (tempUs+2)) printf(lcd_putc, "Temp optima
");
else printf(lcd_putc, "Agua fria      ");
```

```
delay_ms (2000);
```

```
if (kbhit(FT232))in_rs232=fgetc();
if (in_rs232=='m' || in_rs232=='e') break;
if (gestionmenu) goto LOOP_GESTIONMENU;
if (bajoconsumo) goto LOOP_BAJOCONSUMO;
```

```
//CONCENTRACIONES AGUA
```

```
lcd_send_byte (0, LCD_CLEAR);
printf (lcd_putc, "TURB: %1.1f kpar/l", turbidez);
lcd_gotoxy (1, 2);
printf (lcd_putc, "CLORO: %1.1f mgr/l", cloro);
delay_ms (2000);
```

```
if (kbhit(FT232))in_rs232=fgetc();
```

```

    if (in_rs232=='m' || in_rs232=='e') break;
    restart_wdt();
}

if(in_rs232=='m'){
    lcd_send_byte(0, LCD_CLEAR);
    gestionMenuRS232=TRUE;
    printf(lcd_putc, "INTERCAMBIO DE\nDATOS RS232");
    menuRS232();
    gestionMenuRS232=FALSE;
    //Actualizamos los niveles por si hay algún cambio
    actualizarNivelesUs();
    //Nos gustaría que una vez salimos del sistema, sepamos como nos encontramos
    PCF8583_read_datetime (&dt);
    //Ponemos el contador del timer1 a cero para que simular un desbordamiento para la toma de
    medidas.
    contador_timer1=0;
    goto MAIN;
}
printf("fin");
}

```

### **funciones.c**

//Funciones auxiliares para el Sistema Electrónico Digital de una Piscina Cubierta  
//Por Emilio Martínez.

```

/*Declaración de variables*/
#include <stdio.h>
#define LCD_CLEAR 0x01
#define LCD_HOME 0x02
#define LCD_DISPLAY_OFF 0x08
#define LCD_DISPLAY_ON 0x0C
#define LCD_CURSOR_ON 0x0E
#define LCD_CURSOR_BLINK 0x0F
short equal=FALSE, estadocorrecto=FALSE;
char *dir;
//Creamos una tabla con la forma que le damos a la memoria EEPROM y saber en cada momento
donde tenemos que leer o escribir.
char tabla_EEPROM[10]={79,75,48,48,48,48,50,51,54,55}; //{"O","K","0","0","0","2","3"}//{"cabecera1",
cabecera2, clave [x4], temp [x2], })
char ascii[10]={48,49,50,51,52,53,54,55,56,57};
char const mes[12][11]={"Enero", {"Febrero"}, {"Marzo"}, {"Abril"}, {"Mayo"}, {"Junio"}, {"Julio"},
{"Agosto"}, {"Septiembre"}, {"Octubre"}, {"Noviembre"}, {"Diciembre"}};
char const semana[7][10]={"Domingo"}, {"Lunes"}, {"Martes"}, {"Miercoles"}, {"Jueves"}, {"Viernes"},
{"Sabado"}};
char const tabla_menu[6][14]={" CAMB CONTRA "}, {" GEST PISCIN "}, {" CAMB RELOJ "}, {"
CICLO DEPUR "}, {" MODO AHORRO "}, {" SALIR "}};

```

```

char const tabla_1[6][14]={{" TEMPERATURA "},{ " CLORO  "},{ " TURBIDEZ "},{ " NIVEL AGUA
"}, {" LUZ  "},{ " PASO ATRAS "}};
char const tabla_2[2][14]={{" CAMBIA HORA "},{ " PASO ATRAS "}};
char const tabla_3[5][14]={{" DEPURAR YA  "},{ " PARAR DEPUR "},{ " MODIF CICLO "},{ " MODO
TRATAM "},{ " PASO ATRAS "}};
char const tabla_4[3][14]={{" TIEM ESPERA "},{ " PASO ATRAS "}};
int i, TempUs, CloroUs, TurbidezUs, NivelUs, TiempoUs;
char weekday[10];
char tecla=NULL, tecla2;
int ind_menu=1;
date_time_t dt; //El tipo de estructura date_time_t está definido en PCF8583.c
date_alarm_t al; // Definido en PCF8583.c

```

```

/*Listado de funciones*/
void iniciosistema();
void comprobarMemoria();
void establecerOK();
void iniciarRTC();
void modificarHoraRTC();
void iniciarAlarmaRTC();
void modificarAlarmaRTC();
void configurarVariables();
short compararMemorias(char direccion);
void establecercontras();
void establecerTemperatura();
void establecerCloro();
void establecerTurbidez();
void establecerNivel();
void establecerTiempo();
void establecerTipo();
void menu();
char teclado();
long int conversionCloro2Int(float valor);
float conversionInt2Cloro(long int valor);
long int conversionTurbidez2Int(float valor);
float conversionint2Turbidez(long int valor);
long int conversionNivelAgua2Int(float valor);
float conversionint2NivelAgua(long int valor);
long int conversionLuz2Int(float valor);
float conversionint2NivelLuz(long int valor);
float medirTemperatura();
float medirNivelCloro();
float medirNivelTurbidez();
float medirNivelAgua();
float medirNivelLuz();
void nivelesUS();
void RS232();

```

```

/* Funciones auxiliares a usar por el sistema */

```

```

void iniciosistema(){
    lcd_init();
    lcd_send_byte(0, LCD_CLEAR); //Borra LCD
    lcd_gotoxy(2,1);
    printf(lcd_putc, "INICIALIZANDO.");
    printf("Iniciando el sistema");
    lcd_gotoxy(1,2);
}

```

```

    for (i=0; i<8; i++){
        delay_ms(30);
        printf(lcd_putc, "..");
        printf("..");
    }
    delay_ms(400);
    lcd_send_byte(0, LCD_CLEAR);
    lcd_send_byte(0, LCD_CURSOR_ON);
}

```

```

void comprobarMemoria(){

```

```

    short diferencia=FALSE;
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Comprobando \nmemoria");
    printf("\rComprobando la memoria...");
    //Ver asignación de direcciones de memoria en manual.
    //Comprobamos si es la primera vez que se arranca el sistema.
    //Si las dos primeras direcciones no contiene 'O', 'K' se considera que es la primera vez que se
    arranca el sistema

```

```

    for (dir=0; dir<2; dir++){
        *dir=read_eeprom(dir);
        if (dir==0){ //para la dirección 0 debe haber un valor 'O'
            if(*dir !='O') break;
            //Si en la memoria interna está el valor deseado pasamos a comprobar que en la externa
            también lo tenga
            equal=compararmemorias(dir);
            if(!equal) break;
        }
        if (dir==1){ //para la dirección 1 debe haber un valor 'K'.
            if(*dir=='K') {
                equal=compararmemorias(dir);
                if(!equal) break;
                //Si las dos primeras direcciones de ambas memorias satisfacen las condiciones pasamos
                a la siguiente comprobación
                estadocorrecto=TRUE;
            }
        }
    }
}

```

```

    //Si estado vale 1, es porque se ha comprobado que ambas eeprom ya han sido escritas por el
    sistema

```

```

    //Comparamos si los valores de ambas eeprom coinciden, si no lo hicieran se pueden introducir
    //de nuevo las variables, o se le da prioridad a la eeprom externa.

```

```

    if(estadocorrecto){
        for(dir=0; dir<50; dir++){
            equal=compararMemorias(dir);
            if(!equal) diferencia=TRUE;
        }
        if(diferencia){
            for(dir=0; dir<50; dir++){
                *dir=read_ext_eeprom(dir);
                delay_ms(15);
                write_eeprom(dir, *dir);
            }
        }
    }
}

```

```

    else return;
}
else configurarVariables();
}

```

```

void establecerOK(){
    write_eeprom(0, 'O');
    delay_ms(15);
    write_ext_eeprom(0, 'O');
    delay_ms(15);
    write_eeprom(1, 'K');
    delay_ms(15);
    write_ext_eeprom(1, 'K');
    delay_ms(15);
}

```

```

void establecercontras(){
    char intento[4];
    char intento2[4];

    PSW:
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "ESTABLECER CLAVE:");
    lcd_gotoxy(1,2);
    for(dir=0; dir<4; dir++){
        do{
            tecla=teclado();
            //Las teclas pulsadas deben ser números o cancelar.
        }while(tecla=='L' || tecla=='R' || tecla=='O' || tecla=='U' || tecla=='D');
        //En el caso en que se cancelara se volvería a pedir que se introdujera una contraseña
        if (tecla=='C') goto PSW;
        //Se crea un vector con la contraseña
        intento[dir]=tecla;
        printf(lcd_putc, "**");
    }
    lcd_gotoxy(7,2);
    printf(lcd_putc, "Aceptar?");

    do{
        tecla=teclado();
    }while(!(tecla=='O' || tecla=='C'));
    if (tecla=='C') goto PSW;

    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "REPITA:      ");
    lcd_gotoxy(1,2);

    PSW2:
    for(dir=0; dir<4; dir++){
        do{
            tecla=teclado();
        }while(tecla=='L' || tecla=='R' || tecla=='O' || tecla=='D' || tecla=='U');
        if (tecla=='C') goto PSW2;
        intento2[dir]=tecla;
        printf(lcd_putc, "**");
    }
    lcd_gotoxy(7,2);
}

```

```

printf(lcd_putc, "Aceptar?");

do{
    tecla=teclado();
}while(!(tecla=='O' || tecla=='C'));
if (tecla=='C') goto PSW2;

i=0;
for(dir=0; dir<4; dir++){
    if(intento2[dir]==intento[dir]) i++;
}

lcd_gotoxy(1,2);

if (i!=4) {
    printf(lcd_putc, " NO COINCIDEN ");
    delay_ms(800);
    goto PSW;
}

for(dir=0; dir<4; dir++){
    write_eeprom(dir+2, intento[dir]);
    delay_ms(10);
    write_ext_eeprom(dir+2, intento[dir]);
    delay_ms(10);
}

lcd_send_byte(0, LCD_CLEAR);
lcd_gotoxy(1,2);
printf(lcd_putc, " CLAVE GUARDADA");
delay_ms(1000);
}

void establecerTemperatura(){
    long int temp=20;

    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "TEMPERATURA AGUA");
    lcd_gotoxy(8,2);
    printf(lcd_putc, "%cC", 223);
    do{
        lcd_gotoxy(5,2);
        //Mostromos por pantalla el valor la cadena
        printf(lcd_putc, "%2ld", temp);
        tecla=teclado();

        //Establecemos los límites
        if (tecla == 'R') temp++;
        if (tecla == 'L') temp--;
        if (temp<15) temp=40;
        if (temp>40) temp=15;
        //Mientras que no se pulse ok...
    }while(tecla != 'O');

    //Si se acepta ese valor guarda en la eeprom
    write_eeprom(6, temp);
    delay_ms(15);
}

```

```

    write_ext_eeprom(6, temp);
    delay_ms(15);
}

void establecerCloro(){
    float cloro=1;
    long int cloro2;

    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "CONCENT. CLORO: ");
    lcd_gotoxy(9,2);
    printf(lcd_putc, "mgr/l");
    do{
        lcd_gotoxy(5,2);
        //Mostromos por pantalla el valor la cadena
        printf(lcd_putc, "%1.1f", cloro);
        tecla=teclado();

        //Establecemos los límites
        if (tecla == 'R') cloro += 0.1;
        if (tecla == 'L') cloro -= 0.1;
        if (cloro<0.1) cloro=1.5;
        if (cloro>1.6) cloro=0.1;
        //Mientras que no se pulse ok...
    }while(tecla != 'O');

    cloro2 =conversionCloro2Int(cloro);
    //Si se acepta ese valor guarda en la eeprom
    write_eeprom(7, cloro2);
    delay_ms(15);
    write_ext_eeprom(7, cloro2);
    delay_ms(15);
}

void establecerTurbidez(){
    float turbidez=4;
    long int turbidez2;

    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "TURBIDEZ DE AGUA:");
    lcd_gotoxy(9,2);
    printf(lcd_putc, "Kpart/l");
    do{
        lcd_gotoxy(5,2);
        //Mostromos por pantalla el valor la cadena
        printf(lcd_putc, "%1.1f", turbidez);
        tecla=teclado();

        //Establecemos los límites
        if (tecla == 'R') turbidez+=0.2;
        if (tecla == 'L') turbidez-=0.2;
        if (turbidez<0.2) turbidez=4;
        if (turbidez>4) turbidez=0.2;
        //Mientras que no se pulse ok...
    }while(tecla != 'O');

    turbidez2 =conversionTurbidez2Int(turbidez);

```



```

//Si se acepta ese valor guarda en la eeprom
write_eeprom(8, turbidez2);
delay_ms(15);
write_ext_eeprom(8, turbidez2);
delay_ms(15);
}

```

```

void establecerNivel(){

float agua=0;
long int agua2;

lcd_send_byte(0, LCD_CLEAR);
printf(lcd_putc, "NIVEL AGUA:");
lcd_gotoxy(9,2);
printf(lcd_putc, "cm neg");
do{
    lcd_gotoxy(5,2);
    //Mostromos por pantalla el valor la cadena
    printf(lcd_putc, "%1.1f", agua);
    tecla=teclado();

    //Establecemos los límites
    if (tecla == 'R') agua+=5;
    if (tecla == 'L') agua-=5;
    if (agua<0) agua=30;
    if (agua>30) agua=0;
    //Mientras que no se pulse ok...
}while(tecla != 'O');

agua2 =conversionNivelAgua2Int(agua);
//Si se acepta ese valor guarda en la eeprom
write_eeprom(9, agua2);
delay_ms(15);
write_ext_eeprom(9, agua2);
delay_ms(15);
}

```

```

void establecerTiempo(){
int aux;
long int tiempo=90;
char intento[3];

lcd_send_byte(0, LCD_CLEAR);
printf(lcd_putc, "TIEMPO DE ESPERA:");
lcd_gotoxy(7,2);
printf(lcd_putc, "segundos");

do{
    lcd_gotoxy(2,2);
    //Mostromos por pantalla el valor la cadena
    printf(lcd_putc, "%3ld", tiempo);
    tecla=teclado();

    //Establecemos los límites
    if (tecla == 'R') tiempo+=30;
    if (tecla == 'L') {

```

```

        if(tiempo==30) tiempo=450;
        else tiempo-=30;
    }
    if (tiempo>450) tiempo=30;
    //Mientras que no se pulse ok...
}while(tecla != 'O');

write_eeprom(10, tiempo);
delay_ms(15);
write_ext_eeprom(10, tiempo);
delay_ms(15);
}

void establecerTipo(){
    char const tipo[3][8]={"Intenso"}, {"Medio  "}, {"Bajo  "};
    int aux=0;
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "TIPO DEPURADO:");
    lcd_gotoxy(1,2);
    do{
        lcd_gotoxy(5,2);
        for (i=0; i<8; i++){
            //Mostramos por pantalla el mensaje entero
            printf(lcd_putc, "%c", tipo[aux][i]);
        }
        tecla=teclado();
        //Establecemos los límites
        if (tecla == 'R') aux++;
        if (tecla == 'L') {
            if(aux==0) aux=2;
            else aux--;
        }
        if (aux>2) aux=0;
        //Mientras que no se pulse ok...
    }while(tecla != 'O');

    write_eeprom(13, aux);
    delay_ms(15);
    write_ext_eeprom(13, aux);
    delay_ms(15);
}

void configurarVariables(){
    //Preguntamos los valores de las diferentes variables a tratar.
    establecerOK();
    establecerContras();
    establecerTemperatura();
    establecerCloro();
    establecerTurbidez();
    establecerNivel();
    establecerTiempo();
    modificarAlarmaRTC();
    establecerTipo();
}

//Compara el valor guardado en la eepron interna y externa de direccion de memoria pasada.
short compararMemorias(char direccion){

```

```

char interna, externa;
interna = read_eeprom(direccion);
delay_ms(10);
externa = read_ext_eeprom(direccion);
delay_ms(10);
//lcd_send_byte(0, LCD_CLEAR);
//printf(lcd_putc, "%d %d - %d", direccion, interna, externa);
//delay_ms(1000);
if(externa==interna) return TRUE;
else return FALSE;
}

```

//HAY QUE REALIZAR BIEN ESTAS 4 FUNCIONES

```

long int conversionCloro2Int(float valor){
    float cloro;
    long int cloro2;
    cloro=((valor*255)/1.5);
    cloro2=(long int)cloro;
    return cloro2;
}

```

```

float conversionint2Cloro(long int valor){
    float cloro;
    cloro=(valor*1.5)/255;
    return cloro;
}

```

```

long int conversionTurbidez2Int(float valor){
    float turbidez;
    long int turbidez2;
    turbidez=((valor*255)/4);
    turbidez2=(long int)turbidez;
    return turbidez2;
}

```

```

float conversionint2Turbidez(long int valor){
    float turbidez;
    turbidez=(valor*4)/255;
    return turbidez;
}

```

```

long int conversionNivelAgua2Int(float valor){
    float agua;
    long int agua2;
    agua=((valor*255)/20);
    agua2=(long int)agua;
    return agua2;
}

```

```

float conversionint2NivelAgua(long int valor){
    float agua;
    agua=(valor*20)/255;
    return agua;
}

```

```

long int conversionNivelLuz2Int(float valor){
    float luz;

```

```

    long int luz2;
    luz=((valor*255)/450);
    luz2=(long int)luz;
    return luz2;
}

float conversionint2NivelLuz(long int valor){
    float luz;
    luz=(valor*450)/255;
    return luz;
}

void iniciarRTC(){
    PCF8583_init_al(); //Iniciación del RTC con función de alarma

    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Hora automatica?");
    lcd_gotoxy(1,2);
    printf(lcd_putc, "OK/CANCEL");
    do{
        tecla=teclado();
        if (tecla=='O'){
            lcd_send_byte(0, LCD_CLEAR);
            printf(lcd_putc, "Hora ajustada.");
            delay_ms(800);
            lcd_send_byte(0, LCD_CLEAR);
            return;
        }
        if (tecla=='C'){
            modificarHoraRTC();
            return;
        }
    }while(1);
}

void modificarHoraRTC(){

    int d=1, m=0, w=0, h=1, min=1;
    int dlimit;
    int limite[3]={31, 30, 28};
    unsigned int a=10;

    //SELECCIÓN DE MES
    lcd_send_byte(0, LCD_CLEAR);
    do{
        printf(lcd_putc, "MES: ");
        for (i=0; i<12; i++){
            //Mostramos por pantalla el mes entero
            printf(lcd_putc, "%c", mes[m][i]);
        }
        tecla=teclado();
        lcd_send_byte(0, LCD_CLEAR);
        //Establecemos los límites
        if (tecla == 'R') m++;
        if (tecla == 'L') {
            if(m==0) m=11;
            else m--;
        }
    }while(1);
}

```

```

    }
    if (m>11) m=0;
    //Mientras que no se pulse ok...
}while(tecla != 'O');
dt.month=(m+1);

//SELECCIÓN DE DÍA
lcd_send_byte(0, LCD_CLEAR);
do{
    //identificamos los días que tiene el mes seleccionado
    if (m==3 || m==5 || m==8 || m==10) dlimit = limite[1];
    else if (m==1) dlimit = limite[2];
    else dlimit = limite[0];
    //Establecemos el DÍA
    printf(lcd_putc, "DIA: ");
    printf(lcd_putc, "%d", d);
    tecla=teclado();
    lcd_send_byte(0, LCD_CLEAR);
    if (tecla == 'R') d++;
    if (tecla == 'L') d--;
    //según que mes, habrá un límite diferente
    if (d>dlimit) d=1;
    if (d<1) d=dlimit;
    //Hasta que no se pulsa Ok...
}while(tecla != 'O');
dt.day = d;

//SELECCIÓN DE DIA DE LA SEMANA
lcd_send_byte(0, LCD_CLEAR);
do{
    printf(lcd_putc, "DIA SEMANA: ");
    lcd_gotoxy(1,2);
    for (i=0; i<10; i++){
        //Mostramos por pantalla el mes entero
        printf(lcd_putc, "%c", semana[w][i]);
    }
    tecla=teclado();
    lcd_send_byte(0, LCD_CLEAR);
    //Establecemos los límites
    if (tecla == 'R') w++;
    if (tecla == 'L'){
        if(w==0) w=6;
        else w--;
    }
    if (w>6) w=1;
    //Mientras que no se pulse ok.
}while(tecla != 'O');
dt.weekday = w;

//SELECCIÓN DE AÑO
lcd_send_byte(0, LCD_CLEAR);
do{
    printf(lcd_putc, "A%cO: ", 238);
    printf(lcd_putc, "%d", a);
    tecla=teclado();
    lcd_send_byte(0, LCD_CLEAR);
    if (tecla == 'R') a++;

```

```

    if (tecla == 'L') a--;
    if (a>50) a=10;
    if (a<10) a=50;
    //hasta que no se pulse OK...
}while(tecla != 'O');
dt.year=a;

```

```

//SELECCIÓN DE HORA
lcd_send_byte(0, LCD_CLEAR);
do{
    //Establecemos la hora
    printf(lcd_putc, "HORA: ");
    printf(lcd_putc, "%d", h);
    tecla=teclado();
    lcd_send_byte(0, LCD_CLEAR);
    if (tecla == 'R') h++;
    if (tecla == 'L'){
        if(h==0) h=23;
        else h--;
    }
    //formato 24 horas
    if (h>23) h=0;
    //Hasta que no se pulse OK...
}while(tecla != 'O');
dt.hours=h;

```

```

//SELECCIÓN DE MINUTO
lcd_send_byte(0, LCD_CLEAR);
do{
    //Establecemos los minutos
    printf(lcd_putc, "MINUTO: ");
    printf(lcd_putc, "%d", min);
    tecla=teclado();
    lcd_send_byte(0, LCD_CLEAR);
    if (tecla == 'R') min++;
    if (tecla == 'L') min--;
    if (min>59) min=1;
    if (min<1) min=59;
    //Hasta que no se pulse OK...
}while(tecla != 'O');
dt.minutes=min;

```

```

dt.seconds = 00; // Segundos por defecto
PCF8583_set_datetime(&dt); //Puesta en fecha-hora
return;

```

```

}

```

```

void iniciarAlarmaRTC(){
    al.hours=read_eeprom(11);
    delay_ms(15);
    al.minutes=read_eeprom(12);
    delay_ms(15);
    al.seconds =00; // Segundos por defecto
    PCF8583_set_datealarm(&al); //Puesta en hora
}

```

```

void modificarAlarmaRTC(){

```

```

int h=1, min=1;
//SELECCIÓN DE HORA
lcd_send_byte(0, LCD_CLEAR);
printf(lcd_putc, "HORA COMIENZO \nDEPURADO:");
do{
    //Establecemos la hora

    lcd_gotoxy(11,2);
    printf(lcd_putc, "%2d", h);
    tecla=teclado();
    if (tecla == 'R') h++;
    if (tecla == 'L'){
        if(h==0) h=23;
        else h--;
    }
    //formato 24 horas
    if (h>23) h=0;
    //Hasta que no se pulse OK...
}while(tecla != 'O');

write_eeprom(11, h);
delay_ms(15);
write_ext_eeprom(11, h);
delay_ms(15);

//SELECCIÓN DE MINUTO
lcd_send_byte(0, LCD_CLEAR);
printf(lcd_putc, "MINUTO COMIENZO \nDEPURADO:");
do{
    //Establecemos los minutos
    lcd_gotoxy(11,2);
    printf(lcd_putc, "%d", min);
    tecla=teclado();
    if (tecla == 'R') min++;
    if (tecla == 'L') min--;
    if (min>59) min=1;
    if (min<1) min=59;
    //Hasta que no se pulse OK...
}while(tecla != 'O');

write_eeprom(12, min);
delay_ms(15);
write_ext_eeprom(12, min);
delay_ms(15);

return;
}

char teclado(){
    char tecla_teclado;
    //Iniciamos con un valor nulo
    tecla_teclado = NULL;
    do{
        tecla_teclado=kbd_getc();
        //Se muestrea el teclado mientras ninguna tecla sea pulsada
    }while (tecla_teclado==NULL);
}

```

```

    return tecla_teclado;
}

float medirTemperatura(){
    //Medimos la temperatura (ver ds1820.c)
    return ds1820_read ();
}

float medirNivelCloro(){
    set_adc_channel(3); //Selección del canal 3 ( pin RA3 )
    delay_us(10);
    return conversionInt2Cloro(read_adc());
}

float medirNivelTurbidez(){
    set_adc_channel(2); //Selección del canal 2 ( pin RA2 )
    delay_us(10);
    return conversionInt2Turbidez(read_adc());
}

float medirNivelAgua(){
    set_adc_channel(1); //Selección del canal 2 ( pin RA1 )
    delay_us(10);
    return conversionInt2NivelAgua(read_adc());
}

float medirNivelLuz(){
    set_adc_channel(0); //Selección del canal 2 ( pin RA0 )
    delay_us(10);
    return conversionInt2NivelLuz(read_adc());
}

}

void menu(){

    char clave[4];
    char intento[4];
    clear_interrupt(int_ext);

    LOOP_INGRESO:
    lcd_send_byte (0, LCD_CLEAR);
    printf (lcd_putc, "CLAVE: ");
    //Pedimos la clave de usuario y la guardamos en un vector
    for(dir=0; dir<4; dir++){
        do{
            tecla=teclado();
            //Las teclas pulsadas deben ser números o cancelar.
        }while(tecla!='L' || tecla!='R' || tecla!='O' || tecla!='U' || tecla!='D');
        if (tecla=='C') return;
        intento[dir]=tecla;
        printf(lcd_putc, "** ");
    }

    lcd_gotoxy(1,2);
    printf(lcd_putc, "Aceptar?");

    do{

```



```

    tecla=teclado();
}while(!(tecla=='O' || tecla=='C'));
if (tecla=='C') goto LOOP_INGRESO;

```

```

for(dir=0; dir<4; dir++){
    clave[dir]=read_eeprom((dir+2));
    delay_ms(15);
}

```

```

i=0;
for(dir=0; dir<4; dir++){
    if(clave[dir]==intento[dir]) i++;
}
lcd_gotoxy(1,2);

```

```

if (i!=4) {
    printf(lcd_putc, "INCORRECTO");
    delay_ms(800);
    goto LOOP_INGRESO;
}
lcd_putc("CORRECTO");
delay_ms(800);
lcd_send_byte (0, LCD_CLEAR);

```

//CABECERA MENU

LOOPMENU:

```

do{
    printf(lcd_putc, "--MENU USUARIO--");
    lcd_gotoxy(1,2);
    lcd_putc(127);
    for (i=0; i<14; i++){
        //Mostramos por pantalla el mensaje entero
        printf(lcd_putc, "%c", tabla_menu[ind_menu][i]);
    }
    lcd_putc(126);
    tecla=teclado();
    //Establecemos los límites
    if (tecla == 'R') ind_menu++;
    if (tecla == 'L'){
        if (ind_menu==0) ind_menu=5;
        else ind_menu--;
    }
    if (tecla == 'C') return;
    if (ind_menu>5) ind_menu=0;
    //Mientras que no se pulse ok...
}while(!(tecla == 'O' || tecla == 'D'));

```

//SUB MENU-CAMBIO CONTRASEÑA

```

if (ind_menu==0){
    ind_menu=1;
    lcd_send_byte (0, LCD_CLEAR);

```

LOOP\_CONTRA:

```

printf(lcd_putc, "-CAMBIO CONTRAS-");
lcd_gotoxy(1,2);
printf(lcd_putc, "ACTUAL:      ");

```

```

    lcd_gotoxy(9,2);
    for(dir=0; dir<4; dir++){
        do{
            tecla=teclado();
        }while(tecla=='L' || tecla=='R' || tecla=='O' || tecla=='D');
        if (tecla=='C' || tecla=='U') goto LOOPMENU;
        intento[dir]=tecla;
        printf(lcd_putc, "");
    }

    for(dir=0; dir<4; dir++){
        clave[dir]=read_eeprom((dir+2));
        delay_ms(15);
    }

    i=0;
    for(dir=0; dir<4; dir++){
        if(clave[dir]==intento[dir]) i++;
    }

    if (i!=4) {
        lcd_gotoxy(8,2);
        printf(lcd_putc, "INCORRECTO");
        delay_ms(800);
        goto LOOP_CONTRA;
    }
    establecercontras();
    goto LOOPMENU;
}

//SUBMENU GESTION PISCINA
else if(ind_menu==1){
    LOOPGESTION:
    ind_menu=0;
    do{
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "-GESTION PISCIN-");
        lcd_gotoxy(1,2);
        lcd_putc(127);
        for (i=0; i<14; i++){
            //Mostramos por pantalla el mensaje entero
            printf(lcd_putc, "%c", tabla_1[ind_menu][i]);
        }
        lcd_putc(126);
        tecla=teclado();
        //Establecemos los límites
        if (tecla == 'R') ind_menu++;
        if (tecla == 'L'){
            if (ind_menu==0) ind_menu=5;
            else ind_menu--;
        }
        if (tecla=='C' || tecla=='U') goto LOOPMENU;
        if (ind_menu>5) ind_menu=0;
        //Mientras que no se pulse ok...
    }while(!(tecla == 'O' || tecla == 'D'));

    if(ind_menu==0){

```

```

//mostramos la temeperatura establecida y daremos la oprtunidad de modificarla.
lcd_send_byte(0, LCD_CLEAR);
printf(lcd_putc, "%d", tempUs);
lcd_putc (223);
lcd_putc ("C ");
lcd_gotoxy(1,2);
printf(lcd_putc, "Modificar?");
tecla=teclado();
if (tecla=='O'){
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Nueva temp:");
    establecerTemperatura();
}
else goto LOOPGESTION;
}
else if(ind_menu==1){
    //mostramos el nivel de cloro establecido y daremos la oprtunidad de modificarlo
    lcd_send_byte(0, LCD_CLEAR);

    printf(lcd_putc, "%1.1f mgr/l", conversionInt2Cloro(cloroUs));
    lcd_gotoxy(1,2);
    printf(lcd_putc, "Modificar?");
    tecla=teclado();
    if (tecla=='O'){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Nuevo nivel?");
        establecerCloro();
    }
    else goto LOOPGESTION;
}
else if(ind_menu==2){
    //mostramos el nivel de turbidez establecido y daremos la oprtunidad de modificarlo
    lcd_send_byte(0, LCD_CLEAR);

    printf(lcd_putc, "%1.1f Kpart/l", conversionInt2Turbidez(turbidezUs));
    lcd_gotoxy(1,2);
    printf(lcd_putc, "Modificar?");
    tecla=teclado();
    if (tecla=='O'){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Nuevo nivel?");
        establecerTurbidez();
    }
    else goto LOOPGESTION;
}
else if(ind_menu==3){
    //mostramos el nivel de agua establecido y daremos la oprtunidad de modificarlo
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "%1.1f cm neg", conversionInt2NivelAgua(nivelUs));
    lcd_gotoxy(1,2);
    printf(lcd_putc, "Modificar?");
    tecla=teclado();
    if (tecla=='O'){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Nuevo nivel?");
        establecerNivel();
    }
}

```

```

else goto LOOPGESTION;

}
else if(ind_menu==4){
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Luminos. actual:");
    lcd_gotoxy(4,2);
    printf(lcd_putc, "%1.1f lux", medirNivelLuz());
    lcd_gotoxy(1,2);
    tecla=teclado();
    goto LOOPGESTION;
}
else{GOTO LOOPMENU;}
}
else if(ind_menu==2){
    LOOPHORA:
    ind_menu=0;
    do{
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "--GESTION HORA--");
        lcd_gotoxy(1,2);
        lcd_putc(127);
        for (i=0; i<14; i++){
            //Mostramos por pantalla el mensaje entero
            printf(lcd_putc, "%c", tabla_2[ind_menu][i]);
        }
        lcd_putc(126);
        tecla=teclado();
        //Establecemos los límites
        if (tecla == 'R') ind_menu++;
        if (tecla == 'L'){
            if (ind_menu==0) ind_menu=1;
            else ind_menu--;
        }
        if (tecla=='C' || tecla=='U') goto LOOPMENU;
        if (ind_menu>1) ind_menu=0;
        //Mientras que no se pulse ok...
    }while(!(tecla == 'O' || tecla == 'D'));
    if(ind_menu==0){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Cambiar hora?");
        tecla=teclado();
        if (tecla=='O'){
            modificarHoraRTC();
        }
        else goto LOOPHORA;
    }
    else goto LOOPMENU;
}
else if(ind_menu==3){
    LOOPDEPURACION:
    ind_menu=0;
    do{
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "-GESTION DEPURA-");
        lcd_gotoxy(1,2);
        lcd_putc(127);

```

```

for (i=0; i<14; i++){
    //Mostramos por pantalla el mensaje entero
    printf(lcd_putc, "%c", tabla_3[ind_menu][i]);
}
lcd_putc(126);
tecla=teclado();
//Establecemos los límites
if (tecla == 'R') ind_menu++;
if (tecla == 'L'){
    if (ind_menu==0) ind_menu=4;
    else ind_menu--;
}
if (tecla=='C' || tecla=='U') goto LOOPMENU;
if (ind_menu>4) ind_menu=0;
//Mientras que no se pulse ok...
}while(!(tecla == 'O' || tecla == 'D'));

if(ind_menu==0){
    //posibilita comenzar el ciclo de depuración
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Comenzar");
    lcd_gotoxy(1,2);
    printf(lcd_putc, "depuracion?");
    tecla=teclado();
    if (tecla=='O'){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Depurando");
        output_bit(PIN_E2, 1);
        delay_ms(300);
    }
    else goto LOOPDEPURACION;
}
else if(ind_menu==1){
    //posibilita detener el ciclo de depuración
    lcd_send_byte(0, LCD_CLEAR);

    printf(lcd_putc, "Parar");
    lcd_gotoxy(1,2);
    printf(lcd_putc, "depuracion?");
    tecla=teclado();
    if (tecla=='O'){
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Parado");
        output_bit(PIN_E2, 0);
        delay_ms(300);
    }
    else goto LOOPDEPURACION;
}
else if(ind_menu==2){
    //posibilita modificar la hora de comienzo del ciclo
    lcd_send_byte(0, LCD_CLEAR);

    printf(lcd_putc, "Modificar hora");
    lcd_gotoxy(1,2);
    printf(lcd_putc, "comienzo?");
    tecla=teclado();
    if (tecla=='O'){

```

```

        lcd_send_byte(0, LCD_CLEAR);
        modificarAlarmaRTC();
        iniciarAlarmaRTC();
    }
    else goto LOOPDEPURACION;
}
else if(ind_menu==3){
    lcd_send_byte(0, LCD_CLEAR);
    printf(lcd_putc, "Modificar tiempo");
    lcd_gotoxy(1,2);
    printf(lcd_putc, "depurado?");
    tecla=teclado();
    if (tecla=='O'){
        establecerTipo();
    }
    else goto LOOPDEPURACION;
}
else goto LOOPMENU;
}
else if(ind_menu==4){
    LOOPAHORRO:
    do{
        ind_menu=0;
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "-GESTION SLEEP-");
        lcd_gotoxy(1,2);
        lcd_putc(127);
        for (i=0; i<14; i++){
            //Mostramos por pantalla el mensaje entero
            printf(lcd_putc, "%c", tabla_4[ind_menu][i]);
        }
        lcd_putc(126);
        tecla=teclado();
        //Establecemos los límites
        if (tecla == 'R') ind_menu++;
        if (tecla == 'L'){
            if (ind_menu==0) ind_menu=1;
            else ind_menu--;
        }
        if (tecla=='C' || tecla=='U') goto LOOPMENU;
        if (ind_menu>1) ind_menu=0;
        //Mientras que no se pulse ok...
    }while(!(tecla == 'O' || tecla == 'D'));
    if(ind_menu==0){
        //posibilita comenzar el ciclo de depuración
        lcd_send_byte(0, LCD_CLEAR);
        printf(lcd_putc, "Mod. tiempo");
        lcd_gotoxy(1,2);
        printf(lcd_putc, "de espera?");
        tecla=teclado();
        if (tecla=='O'){
            lcd_send_byte(0, LCD_CLEAR);
            establecerTiempo();
        }
        else goto LOOPAHORRO;
    }
    else goto LOOPMENU;
}

```

```

    }
    else return;
}

void actualizarNivelesUs(){
//Se actualizan los niveles al salir del menu
    TempUs=read_eeprom(6);
    delay_ms(15);
    CloroUs=read_eeprom(7);
    delay_ms(15);
    TurbidezUs=read_eeprom(8);
    delay_ms(15);
    NivelUs=read_eeprom(9);
    delay_ms(15);
    TiempoUs=read_eeprom(10);
    delay_ms(15);
}

void menuRS232(){
    char tecla;
    printf("Pulse la tecla 'e' en cualquier momento para salir\n\r");
    while(1){
        MENURS232:
        printf("<<MENU DE CONTROL PISCINA>> \n\r1. Gestion de la piscina\n\r2. Cambio del\n\r3. Cambio de ciclo de depuracion\n\r4. Modo ahorro de energia\n\r");
        tecla = getchar();
        if(tecla=='1'){
            printf("<<MENU DE GESTION DE LA PISCINA>>\n\r1. Temperatura\n\r2. Cloro\n\r3. Turbidez\n\r4. Nivel de agua\n\r5. Luz\n\r0. Paso atras\n\r");
            tecla = getchar();
            if(tecla=='1'){
                printf("\tTEMPERATURA\n\r");
                tecla = getchar();
                if(tecla=='0') goto MENURS232;
                if(tecla=='e') return;
            }
            else if(tecla=='2'){
                printf("\tCLORO\n\r");
                tecla = getchar();
                if(tecla=='0') goto MENURS232;
                if(tecla=='e') return;
            }
            else if(tecla=='3'){
                printf("\tTURBIDEZ\n\r");
                tecla = getchar();
                if(tecla=='0') goto MENURS232;
                if(tecla=='e') return;
            }
            else if(tecla=='4'){
                printf("\tNIVEL AGUA\n\r");
                tecla = getchar();
                if(tecla=='0') goto MENURS232;
                if(tecla=='e') return;
            }
            else if(tecla=='5'){
                printf("\tLUZ\n\r");
                tecla = getchar();
                if(tecla=='0') goto MENURS232;
            }
        }
    }
}

```

```

        if(tecla=='e') return;
    }
    else if(tecla=='0') goto MENURS232;
    else if(tecla=='e') return;
}
else if(tecla=='2'){
    printf("<<MENU DE GESTIÓN RELOJ>>\n\r1. Cambiar hora\n\r0. Paso atras\n\r");
    tecla = getchar();
    if(tecla=='1'){
        printf("\tCAMBIAR HORA\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='0') goto MENURS232;
    else if(tecla=='e') return;
}
else if(tecla=='3'){
    printf("<<MENU DE DEPURACION>>\n\r1. Depurar ahora\n\r2. Parar depuración\n\r3.
Modificar hora ciclo\n\r4. Modo tratamiento\n\r0. Paso atras\n\r");
    tecla = getchar();
    if(tecla=='1'){
        printf("\tDEPURAR AHORA\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='2'){
        printf("\tPARAR DEPURACION\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='3'){
        printf("\tMODIFICAR CICLO\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='4'){
        printf("\tMODO TRATAMIENTO\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='0') goto MENURS232;
    else if(tecla=='e') return;
}
else if(tecla=='4'){
    printf("<<MENU DE MODO AHORRO>>\n\r1. Tiempo de espera\n\r2. Activar\n\r0. Paso
atras\n\r");
    tecla = getchar();
    if(tecla=='1'){
        printf("\tTiempo de espera\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;

```



```

        if(tecla=='e') return;
    }
    else if(tecla=='2'){
        printf("\tActivar\n\r");
        tecla = getchar();
        if(tecla=='0') goto MENURS232;
        if(tecla=='e') return;
    }
    else if(tecla=='0') goto MENURS232;
    else if(tecla=='e') return;
}
}
if(tecla=='e') return;
}
}

```

## ***Funcionalidad***

El sistema es de fácil uso. Una vez arrancado comprueba si es la primera vez que se usa mediante la comparación de memoria interna y externa (si ambos valores coinciden, entonces se considera que no es el primer uso del sistema).

Siempre pedirá la hora, que podrá ser automática (no recomendable ya que el año no lo obtiene correctamente por fallo del RTC o manualmente).

Se tomará una medida de datos cada minuto en estado normal y cada minuto y medio cuando se encuentre en modo SLEEP.

El tiempo de inactividad para pasar a modo sleep puede ser modificado en el menú, al igual que el resto de variables del sistema, como la temperatura mínima, el nivel de cloro del agua, etc.

Para acceder al menú, hay que pulsar el botón de menú e introducir la contraseña por defecto, esta es: 0000. Acto seguido se debe pulsar Ok. Una vez dentro del menú debemos movernos por el mismo usando las teclas de Left y Right, si queremos entrar en esa opción pulsamos Ok o Up. Para salir o para ir al apartado inmediatamente anterior debemos pulsar Cancel o Down

## ***Problemas encontrados y soluciones aportadas***


Durante la realización del proyecto no ha surgido ningún problema reseñable que no fuera solucionado a los pocos minutos tomando como referencia los apuntes de clase.

## ***Presupuesto***

Los precios son sin IVA. Con IVA se calculan después, en la tabla general de precios.

Los precios se han tomado de diferentes páginas web de vendedores reales.

	<p>EEPROM</p> <p>Fabricante: Microchip</p> <p>Modelo: 24LC32A/SN</p> <p>Precio: 0.60€</p>
	<p>PIC</p> <p>Fabricante: Microchip</p> <p>Modelo: PIC18F4520-I/P</p> <p>Precio: 4.90€</p>
	<p>Sensor de temperatura</p> <p>Fabricante: Maxim</p> <p>Modelo: DS18S20+</p> <p>Precio: 5.68€</p>
	<p>Módulo RTC</p> <p>Fabricante: NXP</p> <p>Modelo: PCF8583P</p> <p>Precio: 3.62€</p>
	<p>LCD</p> <p>Fabricante: Everbouquet</p> <p>Modelo: MC16021E8-SYL</p> <p>Precio: 12.02€</p>
	<p>Botón</p> <p>Fabricante: Multicomp</p> <p>Modelo: R13-24A-05-BB</p> <p>Precio: 0.65€</p>
	<p>Motor</p> <p>Fabricante: Maxon Motors</p> <p>Modelo: 110044</p> <p>Precio: 47.44€</p>
	<p>Interruptor</p> <p>Fabricante: Multicomp</p> <p>Modelo: R13-112F-02-BB-0A</p> <p>Precio: 1.39€</p>
	<p>Pull Ups</p> <p>Fabricante: VISHAY BC COMPONENTS</p> <p>Modelo: 2312 142 75602</p> <p>Precio: 0.54€ (pack de 5)</p>

	<p>Potenciómetro</p> <p>Fabricante: Tyco Electronics</p> <p>Modelo: 1174082</p> <p>Precio: 1.56€</p>
---	--

## Presupuesto

Un posible conjunto de tareas a realizar desde el punto de vista del desarrollo mediante un equipo de diseño sería el siguiente:

*Planificación hardware:* sería lo primero que haríamos. Hay que tener en cuenta la base sobre la que se construirá el proyecto; en nuestro caso, la tarjeta EasyPIC3. A continuación, habría que estudiar qué módulos vamos a utilizar y cómo se van a conectar al PIC: display LCD, reloj-calendario RTC, EEPROM externa, sensor de temperatura 1-wire, motor, componentes lógicos, teclado matricial, etc. Para ello es necesario leer las hojas características de los componentes, cosa que también requiere de más tiempo. Por lo tanto, se ha estimado que sería necesario un tiempo de 2 días.

*Planificación software:* una vez tenemos los componentes hardware, es más sencillo hacer un estudio del programa y su funcionamiento. Es necesario adquirir las librerías de los componentes que vamos a utilizar. El tiempo estimado es de 2 días.

*Código en C:* elaboración del programa que hace funcionar los dispositivos. Se deben implementar todas las funciones expuestas en la planificación software y, además, modificar las librerías si fuera necesario. En este apartado es en donde se suelen encontrar más dificultades y en donde se decide si cambiar o no alguna de las partes del hardware y/o software. El tiempo estimado es de 2 semanas.

*Proteus:* para terminar, se debe probar el programa en el software “Proteus” y verificar que todo funciona correctamente. Tiempo estimado: 3 horas.

Esto es una planificación. En realidad el tiempo que he tardado en hacerlo y el orden de planificación no ha sido el mismo, puesto que debido a mi inexperiencia he tenido que hacer las cosas por partes e ir mezclando software con hardware y programación del código. He empleado alrededor de un mes para realizar el proyecto. Además, después de la defensa tuve que añadir y modificar cosas que no estaban bien, y esto ha hecho que haya tardado más días en terminarlo.