UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Escuela Técnica Superior de Ingeniería Industrial

Diseño e Implementación de un Sistema de Control Ambiental para Terrarios

Titulación: Ingeniero Técnico Industrial
Especialidad en Electrónica Industrial
Alumno/a: Jaime Fernández-Caro Belmonte
Director/a/s: Manuel Sánchez Alonso

Cartagena 26 de Noviembre de 2005

AGRADECIMIENTOS

Quiero agradecer a todas aquellas personas que han contribuido de una forma u otra a que este proyecto se haya convertido en una realidad, muy en especial a mi Madre Isabel y a mi Padre Jose María, gracias por haberme dado mi vida, y regalarme parte de las suyas, a mi hermana por estar ahí siempre, OS QUIERO. A los tres por tener la paciencia de aguantarme cuando ni yo mismo me soportaba. A toda mi familia por sus muestras de apoyo y cariño, por sus palabras de ánimo para que aunque muy despacio, pero siguiera caminando. A mis amigos por aguantar tantas y tantas disertaciones sobre 'el proyecto' y su evolución, sois los mejores!

Quiero hacer una mención especial a Marina Alcantud, por pasar esas tardes de verano haciéndome recordar la ortografía mantenida en el olvidado. A Juanmi Ayala y Alejando Forca, por su 'howto estructurar a proyect'. A los usuarios del foro de TodoPIC, especialmente a Manolo (Nocturno), Sisco (Sispic) y Alejandro (Lager) por toda la ayuda prestada para resolver mis cientos de dudas en este mundillo de la electrónica y los pics. A Jesús Trelles por darme un gran punto de apoyo y partida con las bibliotecas del SHT11 y del DS1307. A Joaquín Roca, Julio Ibarrola y Jose Antonio Villarejo por resolverme miles y miles de dudas en este arte de la electrónica y el control. Y Por último, y no por ello menos importante, a Manuel Sánchez, mi director de proyecto, gracias por confiar en mí desde ese primer momento en el que propuse convertir mi idea en un proyecto final de carrera, gracias por su interés y por hacer todo lo que ha estado en su mano para que este proyecto, mi pequeño sueño, se haya hecho realidad.

A las empresas Microchip, Atmel, Dallas y Linear por sus muestras gratuitas. A la empresa Peginfra por conseguir la tan ansiada reactancia electrónica.

GRACIAS.

"A ti, mi búsqueda infinita,
mi desencuentro absoluto,
mi media tú, tu medio yo,
atenderé cuando aparezcas,
nómbrame sino te veo."

Índice

Capítulo 1
Introducción

"Si puede pensarse, puede hacerse"

1.1 Antecedentes

El microcontrolador es uno de los logros más sobresalientes del siglo XX, año tras año, estos dispositivos se acercan más a nuestras vidas, haciéndose un sitio en el centro de cada máquina, de cada equipo electrónico. Dada su alta versatilidad, su facilidad de programación y su alto grado de integración, es posible dotar a casi cualquier objeto de cierta 'inteligencia', de sensorización y reacción con el entorno.

Es por ello que surge la idea de este proyecto, se deseaba realizar un diseño microcontrolado de un objeto que nunca ha tenido, o al menos no comercialmente, una versión microcontrolada.

Este proyecto requeriría aplicar muchos de los conceptos estudiados a lo largo de la carrera de ingeniería técnica electrónica, así como el estudio y aplicación de otros no abarcados por ella.

Se han conseguido unificar conceptos de electrónica básica y de potencia, control, programación y la utilización de varios programas de software informático, además de porqué no decirlo, carpintería y pintura para realizar el terrario.

La motivación de este proyecto es por tanto realizar un terrario microcontrolado capaz de mantener unas condiciones de temperatura y humedad establecidas previamente por el usuario, así como regular las horas de luz y de circulación de agua haciendo uso de dos temporizadores que puedan ser modificados por el usuario.

1.2 Objetivos del Proyecto

En este proyecto se desea desarrollar un sistema microcontrolado configurable por el usuario, capaz de mantener las variables de humedad y temperatura en el valor establecido, además de dos temporizadores que determinan las horas de luz y agua para el terrario.

El usuario debe introducir los valores de configuración del sistema haciendo uso de tres pulsadores; tecla mas, tecla menos, tecla selección. Siendo estos valores visualizados en un LCD.

Para hacer esto, se necesita realizar un profundo estudio de los microcontroladores y su funcionamiento. Se debe posteriormente aprender a programarlos, por lo que es necesario seleccionar el lenguaje de programación optimo para la aplicación, así como habituarse al funcionamiento del software de compilación, simulación y grabación.

Por otro lado se debe, a partir de las hojas de datos de los sensores y actuadores, determinar los más adecuados para el sistema.

Otra parte importante es la del desarrollo de las placas, desde su diseño, pasando por el insolado, atacado y terminando con la puesta en funcionamiento de todo el sistema.

1.3 Resumen

El desarrollo del trabajo se ha dividido en los siguientes capítulos:

- Capitulo 1: Introducción.

Se exponen los motivos, y objetivos del proyecto, así como una visión global del trabajo realizado en cada capítulo

- Capítulo 2: Metodología de diseño.

Se realiza un análisis de cada una de las partes que componen el sistema y el porqué han sido seleccionadas. Se presentan también los programas de software informático utilizados para la compilación y simulación del software del sistema, así como el hardware y el software necesario para realizar la grabación del microcontrolador.

- Capítulo 3: Desarrollo del sistema físico.

Se presenta y explica el funcionamiento de cada una de las placas diseñadas para el sistema, se divide en tres secciones, una para el diseño de la fuente de alimentación, otra para la etapa de microcontrol y por ultimo la de la etapa de potencia.

- Capítulo 4: Desarrollo del código fuente.

Se explican los algoritmos realizados para el control del sistema y se muestra el código desarrollado y las bibliotecas utilizadas para tal propósito.

- Capítulo 5: Vías futuras.

Se expone una lista de conclusiones, así como futuros estudios a realizar para otorgar al sistema de nuevas funcionalidades y mejoras.

Capítulo 2
Metodología de diseño

2.1 Composición del sistema

Para dotar al sistema de un cierto conocimiento del entorno y la correspondiente reacción con este, es necesario añadirle ciertos componentes para lograr tal propósito.

Este debe disponer por tanto, de sensores, para medir ciertas variables del entorno que lo rodean, en este caso, la humedad y la temperatura. Una vez realizada la medición de estas variables debe actuar de una determinada forma, por ello es necesario hacer una selección de los actuadores para poder regular las variables ambientales.

También es necesario seleccionar un reloj de tiempo real para cumplir con los requisitos de temporización del sistema.

Para realizar la configuración del sistema, además de mostrar el estado que presentan sus salidas, entradas, la fecha y la hora, debe utilizarse un display, es necesario realizar una selección del más adecuado.

Todo el sistema es gobernado por un microcontrolador, este debe ser programado para que realice tal función. A continuación se explican cada una de estas partes.

2.1.1 Sensores

Ante la necesidad de medir la temperatura y la humedad se presentan varias alternativas, el uso de sensores analógicos o digitales.

El mercado ofrece varias soluciones dependiendo del tipo que se utilice. En el caso de los analógicos se pueden encontrar sensores de temperatura como LM35, Resistencias PTC, PT100, etc. y sensores de humedad monolíticos (ej. HIH3610), capacitivos (ej. H1), resistivos (ej. NH-01).

Dentro del campo de los sensores digitales se encuentran los de temperatura como el TC77, mientras que hay otros capaces de medir tanto la temperatura como la humedad, como es el caso del SHT11, que ha sido el elegido para este diseño.

Una de las razones principales que han hecho optar por este sensor es que al usar sensores analógicos, se debe realizar un acondicionamiento de la señal y una posterior conversión analógico-digital. El acondicionamiento de señal implica el uso de una electrónica asociada a cada sensor para adaptarlo a niveles que puedan ser procesados por el microcontrolador y su convertidor AD.

Esto cuanto menos es engorroso, puesto que se deben calibrar perfectamente dichos sensores para su correcto funcionamiento. Además se debe tener en cuenta que esto también tendría como consecuencia un mayor tamaño de la placa del diseño y unos costes en componentes mayores en comparación con los de un sensor digital.

Todo esto con un precio bastante asequible (17€) comparado con las demás soluciones estudiadas.

- Descripción.

El SHT11 es un sensor integrado (SMD) con tecnología CMOS, con capacidad de medir la humedad y la temperatura (figura 2.1). Presenta una salida digital mediante bus serie síncrono de dos hilos y protocolo de transmisión específico, además este sensor viene calibrado de fábrica. En la medición de la humedad relativa (%HR) tiene una precisión de ±3,0, una resolución de 12bits, para el caso de la temperatura (ºK) presenta una precisión de ±0,4, una resolución de 14bits.

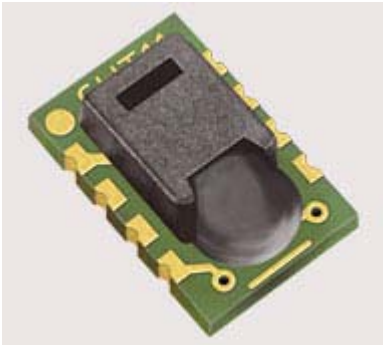


Fig. 2.1 Sensor SHT11

- Funcionamiento.

Este sensor se puede alimentar con un rango de tensiones comprendido entre 2,4 a 5v, es necesario colocar un condensador de desacoplo (100nF) lo más cercano a las patas de alimentación (VCC, GND), además de una resistencia pull-up (10K) en la línea 'Data' justo a la entrada del microcontrolador (figura 2.2).

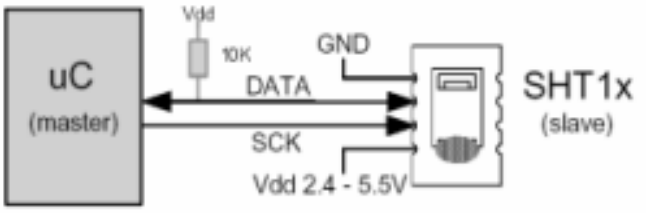


Fig. 2.2 Diagrama de conexión del SHT11

La línea "DATA" se utiliza para leer y enviar datos al sensor, es un pin triestado por lo que requiere una resistencia pull-up de 10K.

La línea "SCK" se utiliza para sincronizar el microcontrolador y el SHT11.
Como se ha indicado anteriormente, este sensor utiliza un protocolo de transmisión propio, el cuál se encuentra detallado en el APENDICE I.

En la página de sensirion (www.sensirion.com) se puede encontrar una biblioteca de ejemplo en C que implementa este protocolo, así como los cálculos necesarios para obtener la temperatura y la humedad. Sólo se ha tenido que adaptar esta biblioteca escrita en ANSI C para WIN32, a ANSI C para CCS, el compilador C para PICs elegido para la realización de este proyecto como se explica en la sección 2.2.1.

2.1.2 Actuadores

Se debe tener en cuenta, en la elección de los actuadores, que todos ellos van orientados al uso en un terrario, y que deben trabajar en unos rangos de operación dentro de las necesidades vitales del animal.

El rango de funcionamiento es bastante amplio como para que el terrario pueda ser habitable por cualquier reptil, aunque se ha centrado en la aclimatación para un camaleón, para ello se han tenido en cuenta las necesidades básicas de este animal, estas son; la iluminación, la humedad y la temperatura.

- La iluminación.

El encargado de suministrar la iluminación ultravioleta, imprescindible para las necesidades vitales del reptil, es un tubo fluorescente de la casa Exoterra, en concreto el modelo Repti Glo 5.0 con un 30% de UVA, esta radiación es la encargada de estimular el apetito, la actividad, y el comportamiento reproductivo. Un 5% de UVB, siendo esta radiación la encargada de la síntesis de vitamina D3. Este modelo cuenta con un consumo de 20W, parámetro importante a la hora de seleccionar el balasto.

Fig. 2.3 Tubo fluorescente de la casa Exoterra

Para elegir el balasto, se ha realizado la selección centrándose en dos tipos distintos, los balastos electrónicos o los balastos inductivos, también conocidos como reactancias. Se ha decidido usar un balasto electrónico regulable aunque su precio sea superior (50€ frente a los 10€ del balasto inductivo) puesto que presenta varias ventajas como son, un ahorro energético alrededor del 20%, alarga la vida útil de la lámpara un 50%, presenta un encendido instantáneo y no produce parpadeos en el tubo fluorescente que puedan llegar a estresar al animal.

El modelo elegido, Quicktronic QT-FH/230-240 (Figuras 2.4 y 2.5), de la casa Osram, tiene la posibilidad de regular la intensidad lumínica mediante un potenciómetro o bien mediante PWM (modulación del ancho de pulsos) con una señal digital, esto es útil para simular el amanecer y el ocaso del sol, aunque finalmente no ha llegado a implementarse, por lo que queda pendiente como futuro desarrollo.

Fig. 2.4 Reactancia electrónica

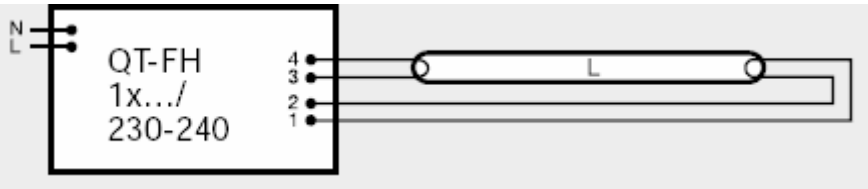


Fig. 2.5 Diagrama de conexión de la reactancia electrónica usada

- La humedad.

Se debe mantener la humedad entre unos valores adecuados para el animal, la forma más eficaz de producir esta humedad es con un generador de niebla ultrasónico, también

conocido como nebulizador, se ha elegido el modelo Coger (Figura 2.6) de la casa Exoterra, que forma parte del kit de la Cascada Natural mediana.

Fig. 2.6 Cascada y Nebulizador

Con este kit, del que forman parte la cascada, una bomba de agua para realizar la circulación de agua simulando una cascada natural y el nebulizador para generar la humedad, quedan satisfechas las necesidades de agua y humedad para el reptil.

- La temperatura.

Otro parámetro importante y que debe ser controlado es la temperatura, para realizar un control optimo sobre ella podría implementarse un PID con el microcontrolador, aunque finalmente se ha decidido implementar un control todo-nada, ya que el calefactor del sistema es una lámpara cerámica de radiación infrarroja y podría verse reducida la vida útil de esta realizando un control PID.

Para ello se ha seleccionado, como puede observarse en la figura 2.7, la lámpara de radiación infrarroja 150W de la casa Exoterra.

Fig. 2.7 Lámpara cerámica

- La ventilación.

El aire en el terrario debe ser renovado cada cierto tiempo, para ello se ha dispuesto de dos ventiladores, uno como intractor y otro como extractor. Además tienen otra función que es la de introducir y sacar aire para regular las condiciones de humedad y temperatura según convenga.

Para tal cometido se han utilizado dos ventiladores de 12V de 12cm (Figura 2.8), de aproximadamente 2000rpm y 200mA.

Fig. 2.8 Ventilador de 12V y 12Cm

2.1.3 RTC

Puesto que el diseño cuenta con dos temporizadores programables, uno para activar/desactivar la luz y otro para activar/desactivar el agua, se debe hacer uso de un reloj de tiempo real (RTC). En este campo uno de los mas usados, tanto por su facilidad de manejo e implementación, además de un coste bastante bajo (3€) es el DS1307 de la casa Dallas (www.maxim-ic.com).

- Descripción.

El DS1307 es un RTC reloj/calendario de 56 bytes de RAM, cuyas direcciones y datos son transmitidos vía serie usando el protocolo I^2C, este reloj/calendario es capaz de procesar horas, minutos,

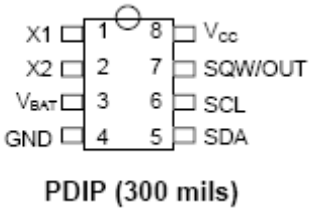


Fig. 2.9 Diagrama de pines del DS1307

segundos, mes, día del mes, día de la semana y año. Además posee la corrección por años bisiestos así como la posibilidad de cambio de formato entre 24H y 12H con indicador de AM/PM.

Tiene un detector de fallo en la alimentación, mediante el cual, conmuta automáticamente para hacer uso de la pila e impedir de esta forma, que al volver a iniciar el sistema se tenga que volver a configurar el reloj.

- Funcionamiento.

Este dispositivo debe ser alimentado a 5v. Requiere el uso de un cristal de 32,768Khz conectado entre los pines 1 y 2 (X1, X2 respectivamente). Al pin 3 (Vbat) va conectada una pila de litio de 3V (o cualquier otra fuente de energía, aunque la de litio es la mas usada por su tamaño y duración).

Como ya se ha indicado anteriormente, este RTC hace uso del bus I^2C para comunicarse con el microcontrolador. Este es un bus serie de dos líneas que corresponde con los pines 5 y 6 (Figura 2.9).

La línea “SDA” corresponde con la entrada/salida de datos, mientras que la línea “SCL” es la entrada de reloj que proviene del microcontrolador, esta se utiliza para sincronizar el flujo de datos.

Ambos pines son en colector abierto, por lo que necesitan una resistencia pull-up (1K8). El pin 7 corresponde con un generador de onda cuadrada de salidas configurables (1Hz, 4Khz, 8Khz, 32Khz). Ya que no es necesario para esta aplicación, se ha deshabilitado puesto que aumentaría el consumo innecesariamente.

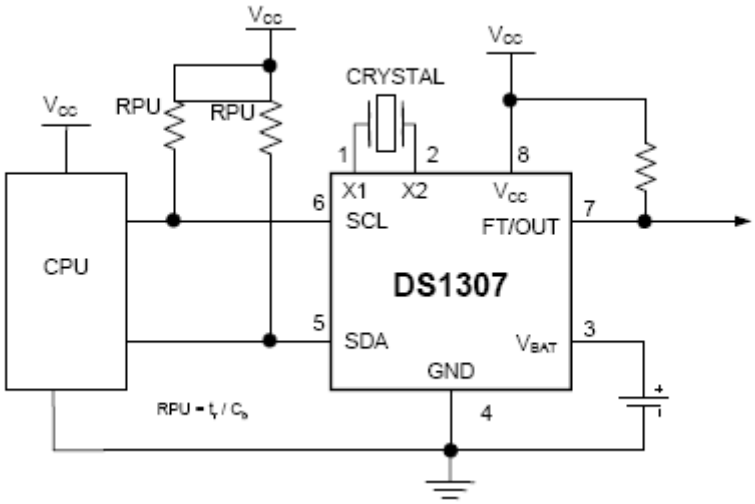


Fig. 2.10 Diagrama de conexión del DS1307

Para el control del DS1307, se ha utilizado una biblioteca en C para el compilador CCS, cedida por el Ing.Tec. Jesús Trelles, que posteriormente ha sido modificada para poder adaptarse a las necesidades de este diseño. Se puede encontrar más información sobre este dispositivo en el ANEXO II.

2.1.4 Display

Es necesario elegir un display para poder configurar los parámetros del dispositivo como son, fecha y hora, temporizadores, temperatura y humedad, etc. además de visualizar datos como la temperatura y humedad actual, los actuadores que están activos, etc.

Se ha optado por evaluar dos tipos de displays, un LCD alfanumérico de 20x4 líneas y un LCD gráfico en color, en concreto el controlado por el PCF8833, utilizado por móviles como el Nokia 6100. Se ha desarrollado la biblioteca en C para su control, realizando el control a través del bus SPI. Las primeras pruebas fueron bastante satisfactorias, se consiguió un buen control del display, de funciones gráficas, envío de imágenes y representación de caracteres.

Aunque finalmente no se ha optado por esta solución, puesto que, en primer lugar, se requería utilizar un driver para el control de los leds blancos, necesarios para la

retroiluminación y dicho componente sólo podía encontrarse en formato MicroSMD, por lo que su implementación, era algo complicada, aunque esto mas que un problema, podría considerarse como un inconveniente.

Fig. 2.11 Display Nokia 6100

El principal motivo se debe a que es necesario almacenar las imágenes en una memoria externa, ya que el microcontrolador, no tiene suficiente memoria para almacenar el código del programa y las imágenes. Se han hecho pruebas con memorias serie I^2C, en concreto con unas muestras de 512K suministradas por Microchip, con ellas ha quedado resuelto el problema de espacio. Aunque se presenta otro problema, la velocidad de representación de imágenes en el display es demasiado lenta con este tipo de memoria. El artículo publicado en Internet acerca del uso de esta biblioteca, así como unas muestras de ejemplo puede encontrarse en la página personal de quien escribe estas líneas (http://perso.wanadoo.es/j1m y http://www.hobbypic.com).

Debido a los inconvenientes anteriores y pese a que este tipo de displays son mas baratos (18€) que un alfanumérico de 20x4 (30€), finalmente se ha optado por la segunda opción, que aunque el resultado no es tan vistoso como puede serlo con el uso de un display gráfico en color, se evitan los inconvenientes antes mencionados.

El display seleccionado, como ya se ha comentado, es un LCD alfanumérico de 4x20 líneas, de esta forma se tiene el espacio suficiente en pantalla para mostrar los datos más importantes, así como todas las opciones del menú.

Se ha elegido un LCD con chip compatible con el HD44780 de Hitachi (Figura 2.12), en concreto el modelo PC2004 de la casa Powertip. Puede hallarse mas información sobre él en el ANEXO III.

Este chip es el más usado, por lo que no se ha tenido ningún problema a la hora de representar caracteres. Sin embargo, se ha decidido no utilizar la biblioteca para el

control de este tipo de LCDs, que puede encontrarse junto al compilador CCS, ya que viene ajustada para trabajar con unas patas por defecto y no puede ser modificada.

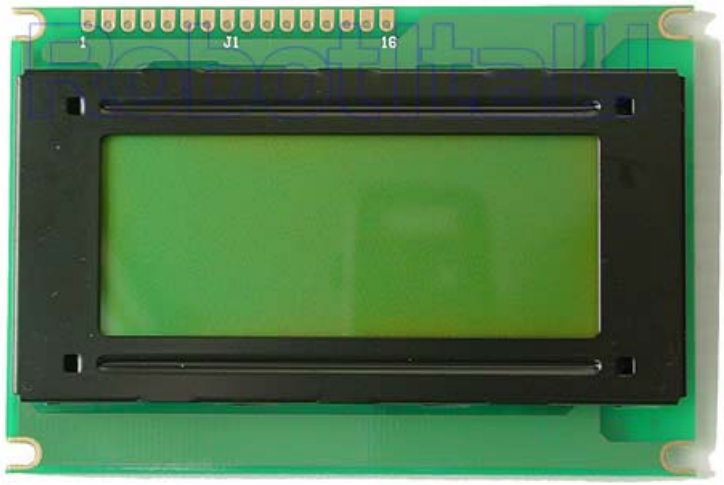


Fig. 2.12 LCD 20x4

Se ha usado una biblioteca (encontrada en los foros de TodoPIC) donde pude configurarse con total libertad que patas del microcontrolador están asociadas a cada una de las líneas del display.

2.1.5 Microcontrolador

Después de haber seleccionado los sensores, y actuadores del sistema, es el momento de elegir un dispositivo capaz de controlarlos. En el mercado pueden encontrarse gran variedad de dispositivos capaces de controlar un sistema, FPGA, DSP, Microcontroladores, etc.

Por su facilidad de uso, su bajo coste, y su amplia difusión en el sector de la enseñanza, se ha decidido afrontar el problema del control del sistema haciendo uso de los microcontroladores. Estos pequeños pero potentes dispositivos, cuentan en su interior con un computador al completo, formando parte de él se encuentra, procesador, memoria de programa (ROM), memoria de datos (RAM) unidades de entrada/salida y diversos módulos para el control de dispositivos como pueden ser, temporizadores,

conversores analógico/digital, comparadores, moduladores del ancho de pulso (PWM) y módulos de comunicación como son UART, USART, CAN, I2C, SPI, USB.

Hay una gran variedad de fabricantes de microcontroladores, entre los mas importantes podemos encontrar a Atmel, Hitachi, Intel, Microchip, Motorola, Texas, Zilog, etc. Siendo Microchip uno de los mas importantes, sobre todo en lo que al sector educación se refiere.

Las características ofrecidas por los microcontroladores de Microchip han hecho que sus microcontroladores PIC hallan sido elegidos para que se ocupen del control de este sistema, se pueden destacar algunas como son:

- Se dispone de una gran cantidad de información sobre ellos.
- Sencillez de programación, manejo y grabación.
- Dispone de gran variedad de herramientas tanto software como hardware.
- Alta velocidad, bajo consumo y tamaño reducido.
- Bajo precio.

La empresa Microchip ofrece una amplia gama de microcontroladores, su punto fuerte son los microcontroladores de 8bits, con los que se ha ido imponiéndo en el mercado con un paso lento, pero firme.

Recientemente ha presentado una gama nueva de microcontroladores (PIC) y procesadores digitales de señal (DSP) de 16bits con unas capacidades de procesamiento de 40Mips.

Dentro de la gama de PICs de 8bits se pueden encontrar varias familias. PIC10, PIC12, PIC14, PIC16, PIC18, siendo la mas potente la familia de los PIC18. Destacan por tener una gran memoria de programa y de variables, capacidad de conseguir hasta 12MIPS.

Para la elección del PIC mas adecuado en este diseño, se han tenido en cuenta varios factores:

- Número de E/S utilizadas: 19 líneas en total
- Módulos de control: 2 I2C, 3 interrupciones externas, 1 temporizador
- Memoria: A priori no puede ser calculada, pero sí se estima que es necesario un PIC con una gran memoria ROM, esto debido al menú implementado para la configuración del sistema.

Todos estos requisitos hacen posible una primera aproximación hacia el PIC elegido.

- Por el número de patas de E/S se seleccionan los PICs de 28 patas.
- Debido a la gran cantidad de memoria requerida se ha seleccionado la serie 18.
- Debido a las posibilidades del compilador, es posible emular el protocolo de transmisión I2C por software, por lo que este parámetro no se considera esencial. Igual ocurre con las interrupciones externas y el temporizador, puesto que la gran mayoría de PICs de la serie 18 incorporan estas características.
- Otra de las necesidades para el sistema es que el PIC presentara memoria EEPROM donde poder almacenar los datos introducidos por el usuario, para que en caso de que se vaya la luz, el sistema no pierda ningún dato.

Se ha terminado por seleccionar el PIC 18F2420, ya que cumple con las características necesarias, además de poseer un bajo precio, 4,5€.

Conforme ha ido evolucionando el código fuente de la aplicación, ha aumentado la memoria usada por este, haciéndose necesario utilizar un PIC con mayor memoria ROM, finalmente se ha optado por trabajar con su “hermano” mayor, el PIC 18F2520 (Figura 2.13), con un precio de 5€.

- CARACTERISTICAS GENERALES DEL PIC 18F2520.

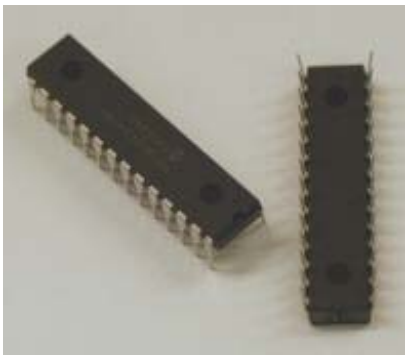


Fig. 2.13 Microcontrolador PIC

- Arquitectura del procesador modelo Harvard, donde la CPU tiene un bus para la memoria de programa y otro para la de datos.
- Procesador RISC (Reduced Instruction Set Computer) de 70 instrucciones.
- Tecnología CMOS.
- 12Mips (millones de instrucciones por segundo) para una frecuencia de trabajo de 40Mhz.
- 32 KB de memoria de programa de tipo Enhanced Flash.
- 1535 Bytes de memoria de datos de tipo SRAM.
- 256 Bytes de memoria de datos de tipo EEPROM.
- 25 puertas de Entrada/Salida.
- 3 temporizadores de 16bits y 1 temporizador de 8bits.
- 1 módulo I2C.
- Tensión de funcionamiento de 5V y 3,3V.
- 25mA de corriente suministrada/admitida por cada pata.

Estas son solo algunas de las características que ofrece este microcontrolador, se han citado las más importantes para el desarrollo de este diseño.

Una descripción mas detallada de este microcontrolador, puede ser encontrada en el ANEXO IV, donde también se describen cada uno de los módulos internos del microcontrolador.

A continuación se realiza una breve descripción de aquellos que han sido útiles para este proyecto.

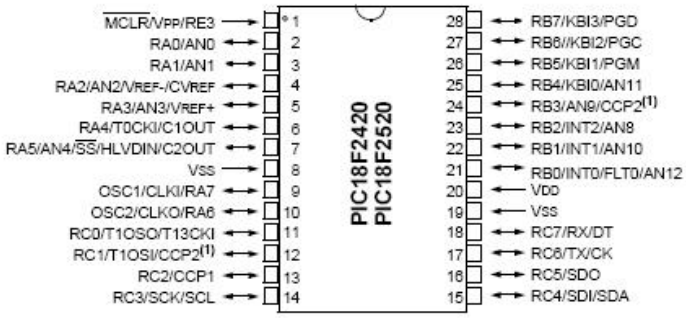


Fig. 2.14 Patillaje del PIC 18F2520

- Oscilador principal.

Para la generación de pulsos de reloj internos se dispone de 10 alternativas. Entre ellas se ha elegido la opción XT que define un cristal u oscilador externo, este irá conectado en las patas 9 y 10, con unos condensadores de desacoplo de 30 pF (Figura 2.15).

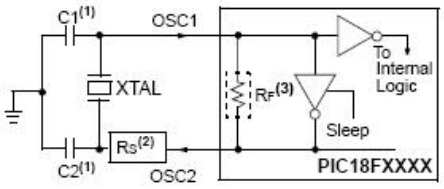


Fig. 2.15 Oscilador

- Temporizador TMR0.

El temporizador TMR0 es un contador ascendente de 16 bits programable mediante software, provocando una interrupción cuando se llega al valor de temporizado deseado. Para ello se hace uso de la siguiente formula:

Temp = 4 · Tosc · (2 ^ 16 - Valor_cargado_TMR0) · preescala_TMR0

De donde será despejado “Valor_cargado_TMR0”.

- Interrupción externa INT0/INT1/INT2.

El PIC 18F2520 tiene diversas fuentes de interrupción, entre ellas destaca la interrupción por cambio de estado en la patas RB0, RB1 y RB2 que corresponden respectivamente con INT0, INT1 y INT2. Gracias a las

interrupciones y a esta en concreto es fácil determinar, para el caso de este diseño, cuando se ha presionado un pulsador.

- Puertos de Entrada y Salida.

  Se pueden encontrar tres puertos distintos para el PIC 18F2520 (Figura 2.14), Puerto A (formado por 6 pines de comunicación bidireccional con el exterior), Puerto B (con 8 pines de comunicación bidireccional), Puerto C (con 8 pines de comunicación bidireccional). Todos actúan como pines de entrada/salida digital, adicionalmente pueden encontrarse 13 pines de entrada analógica para el módulo de conversión AD, multiplexados sobre los puertos digitales, por lo que se deberá configurar por software, la función de cada pin y que módulos, de los asociados a los pines estarán activos o no.

- Módulo I2C.

  Este tipo de interfaz serie ha sido desarrollado por Philips y utiliza solo dos hilos para la transmisión, un hilo para datos y el otro para el reloj de sincronización, se alcanza una velocidad máxima de 400 Kbps. Es capaz de interconectar hasta 128 dispositivos situados a gran distancia.

  El master es el que inicia y termina la transferencia y genera la señal de reloj. El esclavo es el dispositivo direccionado por el master, mediante 7 bits, lo que limita el número de dispositivos a 128.

  El inicio de la transmisión se determina con el bit de inicio (S) y el final con otro bit de stop (P). El bus serie de 2 hilos utiliza uno de ellos para transferir datos (SDA) y el otro para la señal de reloj (SCL).

  En el protocolo I2C cada dispositivo tiene asignada una dirección de 7 o de 10 bits que envía el master cuando comienza la transferencia con uno de ellos. Tras la dirección se añade el bit de recepción/transmisión o lectura/escritura (R/W). Los datos se transmiten con longitud byte y al finalizar cada uno se

inserta un bit de reconocimiento ACK. Debe existir un modulo de arbitraje que gestione que solo hay un maestro en cada instante sobre el bus compartido.

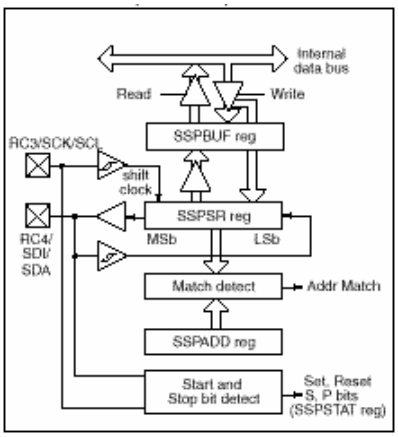


Fig. 2.16 Registros usados por I2C

SSPBUF es el registro donde se almacena el byte a transmitir o el que se recibe. SSPSR es el registro desplazamiento serie de la línea E/S. SSPADD es el registro de direcciones que identifica el dispositivo (modo esclavo) o que lo direcciona (modo master). El registro de control SSPCON selecciona las diversas funciones del modo I2C. Puede verse los registros utilizados por el bus I2C en la figura 2.16.

Cada vez que se detecta un bit de inicio o un bit de stop es posible que se active la bandera SSPIF y en el caso de estar también activado el bit de permiso correspondiente generar una interrupción.

Como antes ha sido comentado, esto es solo una breve descripción de las posibilidades que ofrece este PIC, se considera que ahondar en una explicación de la arquitectura del microcontrolador, así como la configuración de sus registros, está fuera de los objetivos de este proyecto, por lo que para una documentación mas extensa sobre el funcionamiento del PIC 18F2520, se ha incluido su datasheet en el ANEXO IV.

2.2 Entorno de trabajo

Una vez determinados los componentes que forman el sistema, se debe programar el microcontrolador para que cumpla las condiciones para las que está previsto el sistema. Para ello es necesario utilizar varias herramientas tanto software como hardware para generar el archivo que será grabado en el microcontrolador.

2.2.1 Compilador

Los microcontroladores son dispositivos programables con el fin de que realicen una determinada acción. El microcontrolador elegido para este proyecto es de la empresa Microchip y pertenece a la familia PIC de la serie 18. Para realizar la programación de este dispositivo lo primero que se debe elegir es el lenguaje de programación para a continuación seleccionar un compilador. Los lenguajes más extendidos son ASM, Basic y C, aunque también se pueden encontrar para lenguajes de programación basados en objetos, como son C++ y Java, aunque estos compiladores no están los suficientemente desarrollados y optimizados como para que puedan ser considerados como opción valida para el desarrollo de una aplicación.

El lenguaje de programación elegido para este proyecto ha sido el C, se ha elegido por su facilidad para la realización de tareas complejas, a diferencia que ocurre en ASM, por lo compacto del código generado, a diferencia del Basic y que si bien el código generado es mas compacto que generalmente el desarrollado en ASM, cada vez los compiladores de C están mas optimizados, llegando en ocasiones a generar códigos tanto o mas compactos que los desarrollados puramente en ASM.

Una vez determinado el lenguaje de programación con el que se va a trabajar, en este caso el C, el siguiente paso es seleccionar el compilador. Podemos encontrar una amplia de compiladores, unos gratuitos como puede ser el C18 de la empresa Microchip y otros de pago, como PICC de la empresa Hi-Tech ó CSCC de la empresa CCS.

El compilador elegido es el de la empresa CCS, en concreto el PCWH, soporta a toda la familia de PICs y próximamente también los DsPIC. Ha sido elegido por su gran compactación del código generado y por su alta difusión en el mundo de la enseñanza.

En este capítulo se pretende realizar un acercamiento al entorno de desarrollo del PCWH, para ello se realizará la explicación de un sencillo proyecto, este consistirá en el encendido de un led conectado a RA0, al activarse la interrupción de la pata RB0 donde será conectado un pulsador normalmente abierto.

Al ejecutar el 'PIC C Compiler' aparece una ventana como la mostrada en la figura 2.17

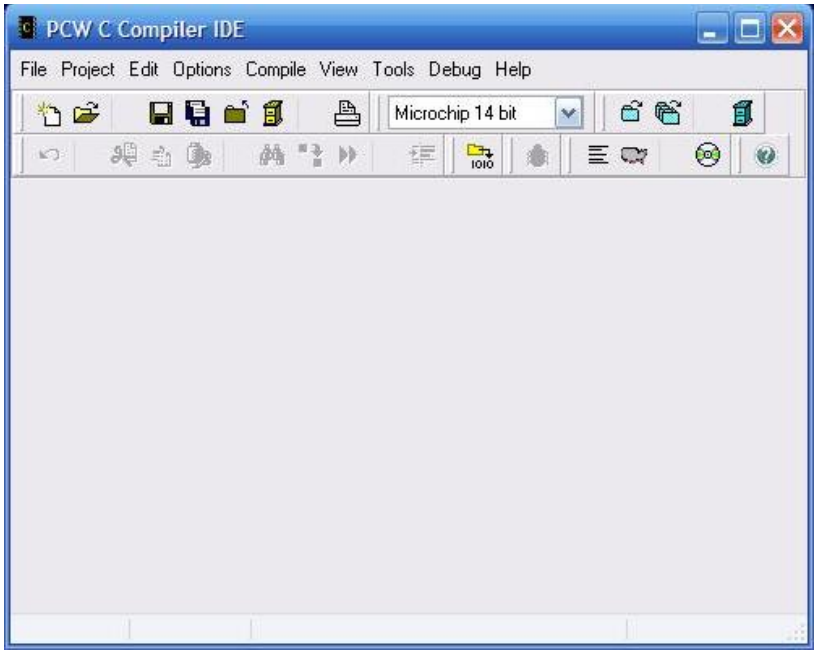


Fig. 2.17 Vista principal

A continuación se pincha sobre File/New y se guarda el archivo en la ubicación que se desee. Se dispone ahora del editor de texto dispuesto para escribir el código de la aplicación. En la figura 2.18 se muestra el código que va a ser explicado a continuación.

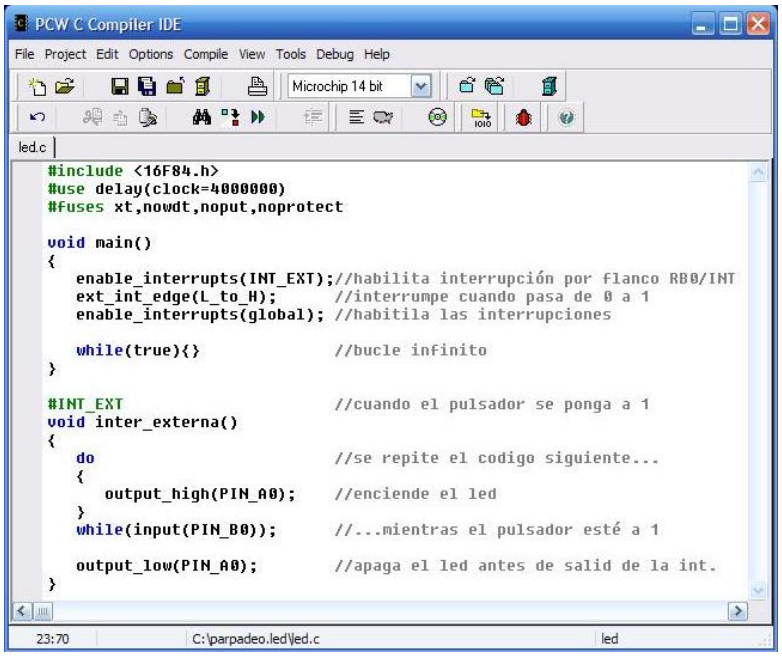


Fig. 2.18 Editor de código

Todo programa se comienza indicándole al compilador que PIC va a ser usado, para este sencillo ejemplo, se ha elegido un 16F84. Para ello se ha de incluir la directiva:

#include <16F84.h>

Siendo el 16F84.h el archivo de cabeceras del PIC en cuestión, en ese archivo se pueden encontrar las características configurables del microcontrolador por el compilador.

El siguiente paso es establecer la frecuencia de funcionamiento del sistema:

#use delay (clock=4000000)

Aquí se está configurando el sistema para trabajar a una frecuencia de 4Mhz.

Por último se configuran los fusibles del dispositivo:

#fuses xt, nowdt, noput, noprotect

La lista de fusibles disponibles para el 16F84 puede encontrarse en el 16f84.h o bien en el propio datasheet del dispositivo. Estos en concreto establecen el uso de un cristal externo (xt), la desactivación del perro guardián (nowdt), deshabilitación del Power-Up Timer (noput) y no se protege el código (noprotect).

A continuación se declaran las variables globales, en este caso no han sido necesarias, por lo que se desarrolla la función principal, es decir, la función main. Si necesitáramos alguna variable local, este sería el momento de declararlas, ya que no es el caso, se comienza con las funciones de configuración del microcontrolador.

enable_interrupts(INT_EXT);

Esta función activa la interrupción externa, se provoca por un cambio de estado de la pata RB0.

ext_int_edge(L_to_H);

Con esta función se configura el flanco de la interrupción externa, en este caso se provocará con el flanco ascendente, es decir, cuando del nivel bajo, se pase a nivel alto.

enable_interrupts(global);

Activa globalmente las interrupciones que estén habilitadas individualmente.

A continuación se entra en un bucle infinito en el que no se hace nada, el microcontrolador estará esperando a que se produzca una interrupción en la pata RB0, en el momento que eso ocurre, se entra en el servicio de tratamiento de la interrupción externa, determinado por #INT_EXT

Dentro de la rutina de interrupción, se realiza la sentencia DO-WHILE típica de ANSI C, en la que se repetirá:

output_high(PIN_A0);

Esta función establece un nivel lógico alto ‘1’ en el PIN A0, es decir, enciende el LED. Y esto se repetirá mientras:

input(PIN_B0)

Con esta función devuelve el estado lógico del PIN B0, si es ‘1’ indica que aún está el pulsador activo y volverá a repetirse el bucle, si es ‘0’ indica que el pulsador no está activo, por lo que se sale del bucle. Y se desactiva el LED haciendo uso de la función:

output_low(PIN_A0);

Se pone a nivel bajo el PIN A0, provocando un apagado del LED y se sale de la interrupción, para volver al bucle infinito donde el PIC se quedará a la espera de una nueva interrupción.

El siguiente paso consiste en compilar el código escrito, para ello se pincha en la opción del menú llamada Compile, o bien se pulsa la tecla F9. Si no se ha cometido ningún fallo al escribir el código, aparece una ventana como la mostrada en la figura 2.19

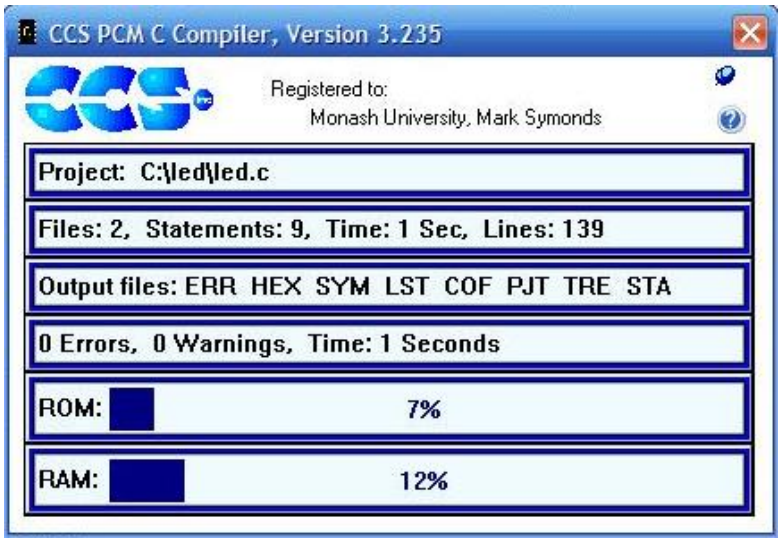


Fig. 2.19 Pantalla de compilación

En ella se muestra la cantidad de ROM y RAM utilizada por el programa que se ha compilado. Se indican también los archivos generados, siendo el .hex el que se debe cargar en el PIC ó si se prefiere, utilizar con un simulador.

Esto es solo una pequeña parte de lo que este programa es capaz de ofrecer, por ello se añade el manual de usuario del compilador en el ANEXO V, en el se explican todas y cada una de las funciones y parámetros de configuración que posee este compilador.

2.2.2 Simulador

En las primeras etapas de un diseño, es recomendable e incluso necesario, tener la posibilidad de realizar simulaciones antes de implementar el circuito en la realidad.

Pueden encontrarse varias herramientas de simulación pero quizá ninguna con las características de Proteus.

Proteus es una suite de diseño y simulación electrónica formada por dos programas, ARES e ISIS.

El primero de ellos es utilizado para diseñar la PCB del diseño y su posterior ruteado, dejando la placa lista para ser realizada en circuito impreso. El segundo programa, ISIS, es utilizado para la realización del esquemático y su simulación.

Pero es sin duda, su capacidad de simular el funcionamiento de un PIC a través del .hex, generado por el programa de compilación, la cualidad que lo hace diferente al resto.

En esta sección no se pretende explicar el funcionamiento de la suite Proteus, la explicación se ceñirá al uso del ISIS, el programa de simulación en sí mismo.

¿Y qué mejor forma de llevar a cabo una explicación que con un ejemplo práctico? Este ejemplo está basado en la simulación llevada a cabo en la primera etapa del diseño. Consiste en la simulación de la interface básica Usuario-PIC, en la que se simula el funcionamiento del menú del sistema.

Consta de un Display LCD alfanumérico de 20 caracteres y 4 líneas, y 3 pulsadores para interactuar con el PIC.

Su funcionamiento es el siguiente: El programa principal está en espera de que se produzca una interrupción por RB0 (Tecla Enter) para entrar al menú, una vez dentro, se comprueba el estado de las patas RB0, RB1, RB2 (Tecla Enter, Tecla Mas, Tecla Menos) que son las encargadas de desplazarse por los menús. En el caso de que ninguna tecla sea presionada en 5 segundos, automáticamente se sale del menú.

Una vez ejecutado el Proteus ISIS se tiene la pantalla mostrada en la figura 2.20.

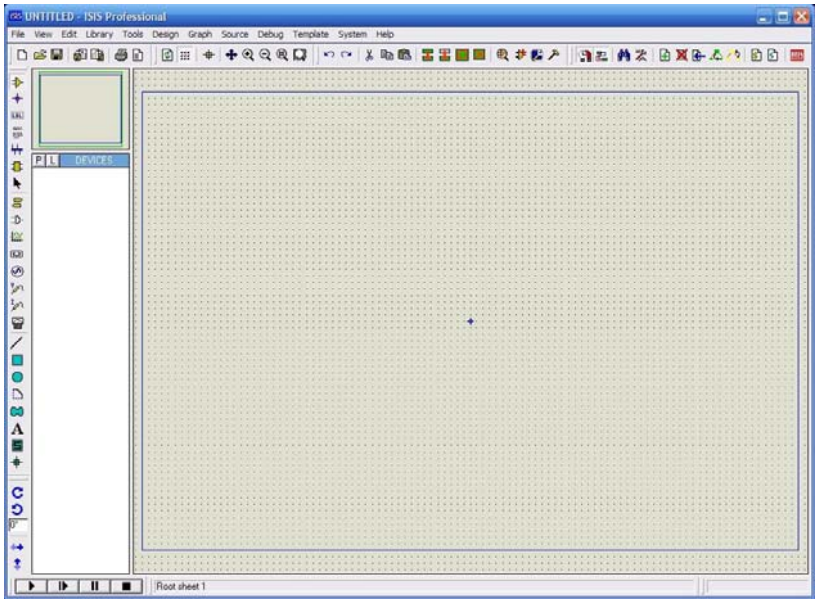


Fig. 2.20 Vista principal

Lo primero que se debe realizar, es insertar los componentes que se vayan a utilizar, para ello pinchamos sobre la "P" o pulsamos esa misma tecla, al hacerlo, aparecerá la ventana que se muestra en la figura 2.21.

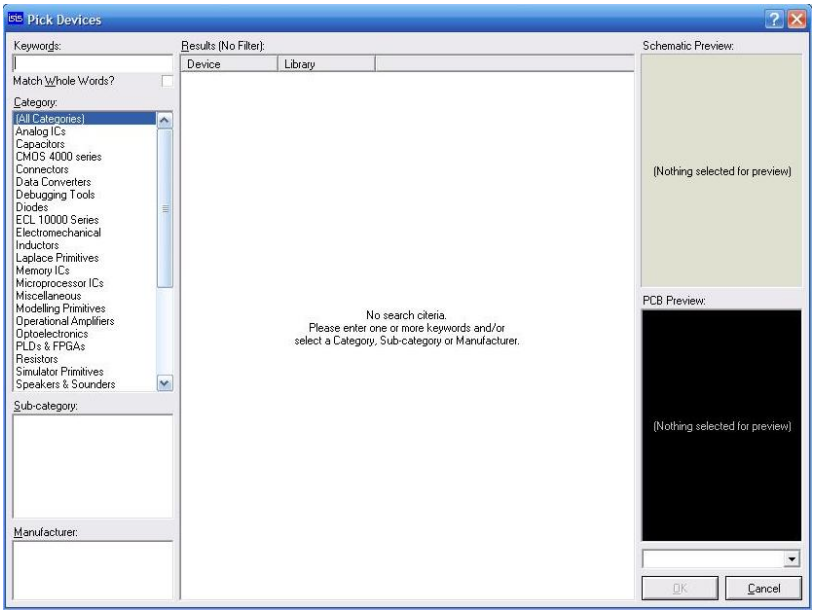


Fig. 2.21 Ventana de selección de componentes

En la ventana de selección de dispositivos, se da la opción de realizar una búsqueda por categorías del componente que se vaya a insertar ó realizar la búsqueda mediante el nombre de este haciendo uso de la caja keywords. Para este ejemplo se realiza la búsqueda por categorías (Figura 2.22) del PIC 18F2520, así como una segunda búsqueda por palabra clave (Figura 2.23) para encontrar el LCD 20x4. Se obtienen los siguientes resultados:

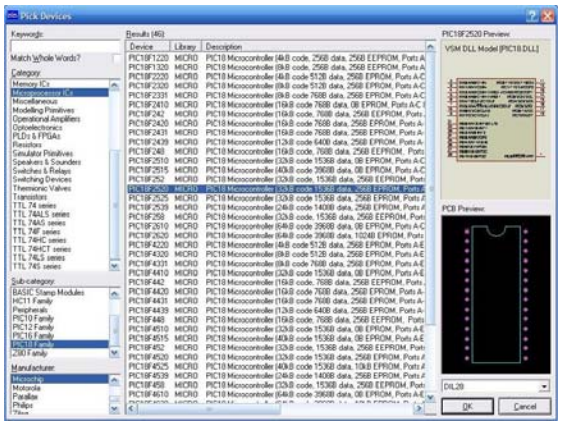


Fig. 2.22 Selección por categorías

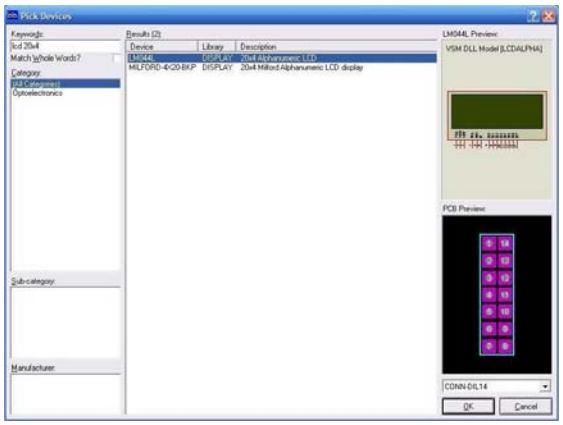


Fig. 2.23 Selección por palabra clave "keyword"

Una vez encontrado el componente, se hace doble clic sobre él y se añade a la lista de dispositivos del proyecto.

Después de haber seleccionado todos los componentes que forman parte de la simulación, se cierra la ventana de selección de componentes y se procede a la colocación del componente sobre el grid de trabajo.

Para colocar los componentes sobre el grid, basta con pinchar sobre el componente que previamente se ha añadido a la lista de dispositivos, de esta forma queda seleccionado.

A continuación se pincha sobre el lugar del grid donde se quiere colocar el componente y se ve como aparece el componente, en la posición indicada, en ese mismo instante.

Posteriormente se colocan los terminales de alimentación y masa, para ello se pincha sobre el icono ‘Inter-sheet Terminal’, es el último en la imagen que se muestra en el lateral. La lista de los componentes del proyecto ha cambiado para mostrar una lista de terminales, se selecciona el Power y el GND. Se repite el mismo procedimiento citado anteriormente para emplazar los terminales sobre el grid. En el caso de que se desee volver a la lista de componentes, bastará con pinchar sobre el icono ‘Component’, que corresponde con el primero de la imagen.

Una vez colocados los componentes y la alimentación el esquema a simular presenta un aspecto como el mostrado en la figura 2.24.

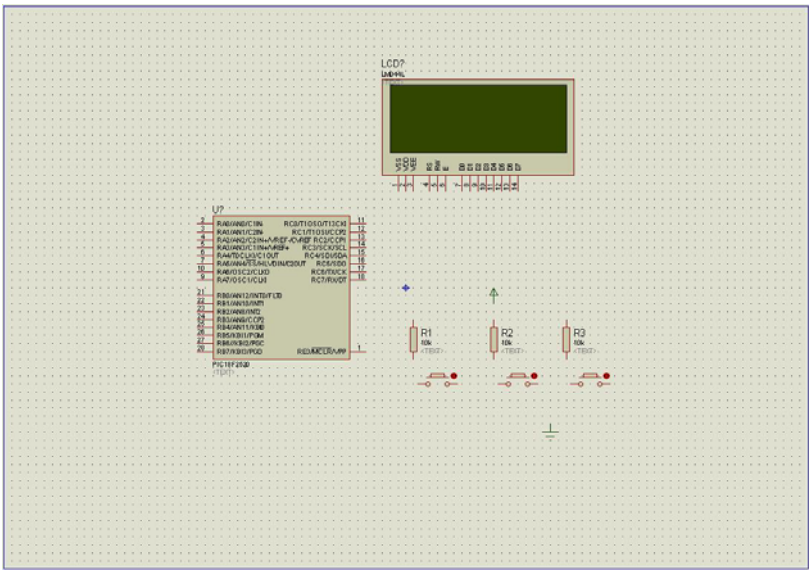


Fig. 2.24 Componentes situados en el grid

A continuación, se procede a realizar la conexión entre componentes, para ello basta con situar el cursor encima de uno de los terminales del componente elegido, pinchando sobre él y desplazando el cursor hacia la pata del componente que corresponda. De esa forma se crea una conexión virtual entre esos terminales.

Se repite el proceso para todos los componentes y se obtiene algo parecido a lo mostrado en la figura 2.25.

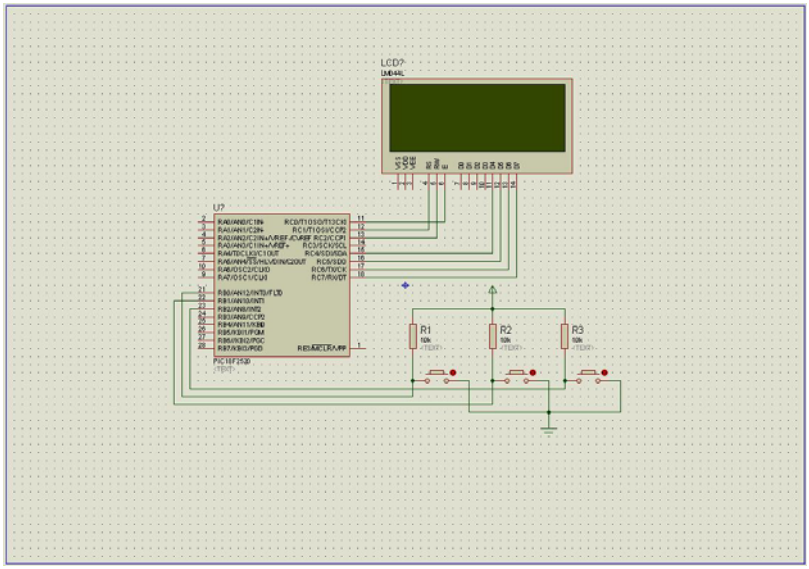


Fig. 2.25 Conexiones realizadas entre componentes

Hecho esto, se puede proceder con la simulación del diseño, para ello se selecciona el PIC, pinchándole con el botón secundario y seguidamente con el primario. Se abrirá entonces una ventana para editar la configuración del componente como la mostrada en la figura 2.26.

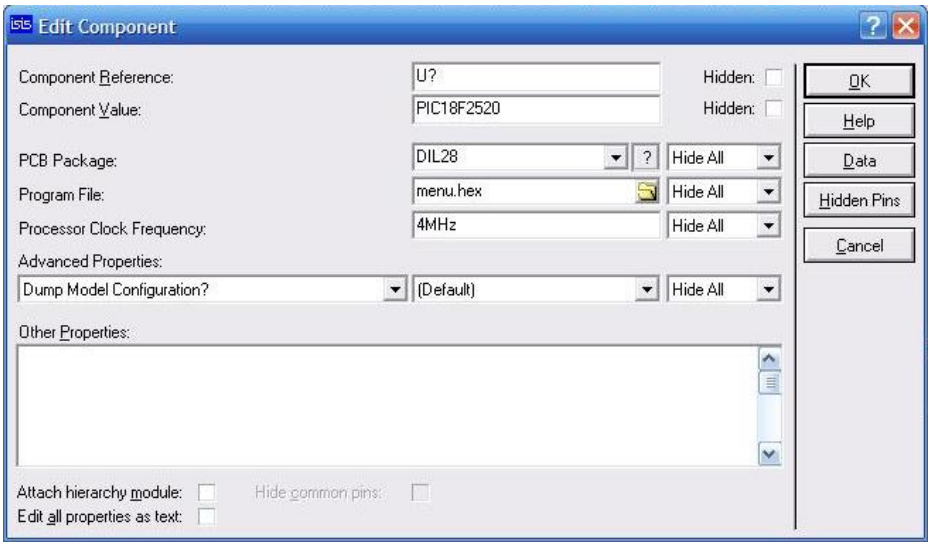


Fig. 2.26 Ventana de edición de componente

Mediante esta ventana es posible modificar una gran cantidad de parámetros del microcontrolador, siendo las mas importantes la casilla ‘Program File’, donde se debe indicar el nombre del archivo .hex generado por el compilador, y la casilla ‘Processor

Clock Frecuency' que toma un valor por defecto de 4Mhz, aunque puede ser modificado para indicar la frecuencia de reloj del cristal usado.

Una vez realizados los ajustes citados anteriormente, se puede proceder a la simulación propiamente dicha, para ello se debe pinchar sobre el icono que representa un símbolo de 'play', puede encontrarse en la esquina inferior izquierda del programa.

Si no se ha cometido ningún fallo en la colocación de los componentes ó en el programa a simular, se obtiene el resultado de la simulación (Figura 2.27)

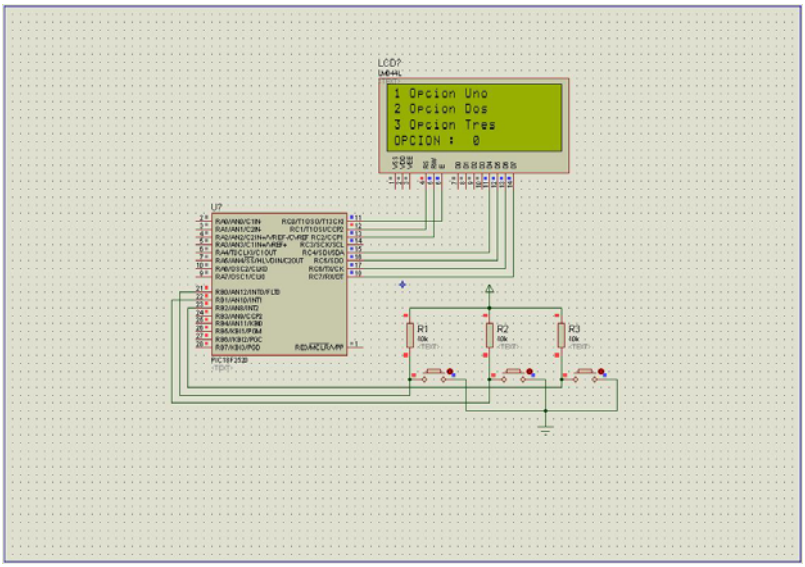


Fig. 2.27 Simulación

2.2.3 Software de grabación

Una vez se ha compilado el programa de aplicación para el PIC, haciéndose uso del compilador de la empresa CCS, se obtiene un archivo hexadecimal (.hex) que se debe grabar en la memoria FlashROM del PIC. Para realizar esta labor se pueden encontrar multitud de programas tanto comerciales, como con licencia freeware. Pueden encontrarse tanto para plataformas UNIX como Windows.

Entre esta variedad de software cabe destacar uno, tanto por sus características, como por su facilidad de uso, así como la gran cantidad de PICs y Hardware para grabación que soporta, el WinPIC800.

WinPIC800 es un software de grabación destinado a grabar la memoria ROM del PIC con la aplicación previamente desarrollada. Soporta multitud de programadores, entre ellos, los dos desarrollados en este proyecto, el GTP Lite y el GTP USB Lite, como se verá en la sección 2.2.4. Su creador Sisco Benach, sigue trabajando constantemente para incorporar al grabador los últimos PICs que van apareciendo en el mercado. Otra de sus ventajas es que posee licencia Freeware, cosa que hay que agradecer puesto que hoy en día se cobra "hasta por respirar".

El software, en constante evolución, puede ser descargado desde la página web personal de Sisco (http://perso.wanadoo.es/siscobf/)

Se muestra en la figura 2.28 la pantalla principal del WinPIC800

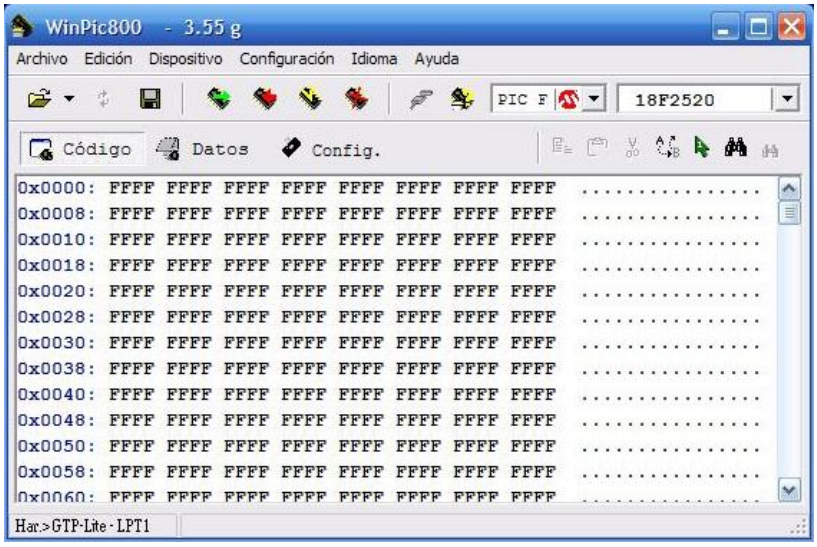


Fig. 2.28 Vista principal

Se van a explorar a continuación cada una de las opciones disponibles en su menú.

- Menú Archivo.

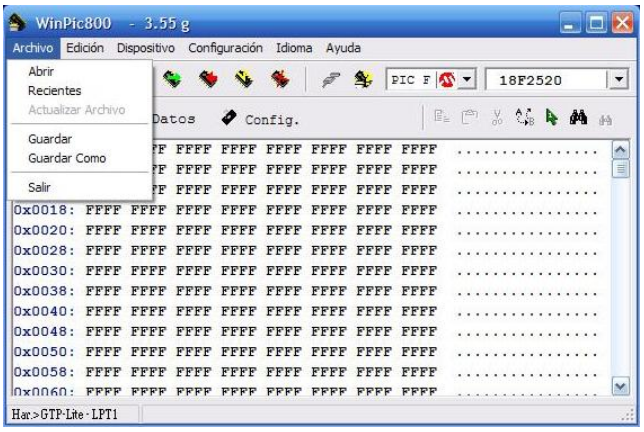


Fig. 2.30 Menú Archivo

  - Abrir: Al pinchar sobre esta opción, aparecerá la típica ventana de abrir donde se debe seleccionar el archivo .hex a programar en el PIC.
  - Recientes: Muestra una lista de los últimos .hex abiertos.
  - Actualizar Archivo: Recarga el archivo desde el disco duro.
  - Guardar: Guarda las modificaciones realizadas en el archivo .hex.
  - Guardar Como: Da la posibilidad de guardar el archivo .hex con otro nombre.
  - Salir: Cierra el programa.

- Menú Edición.

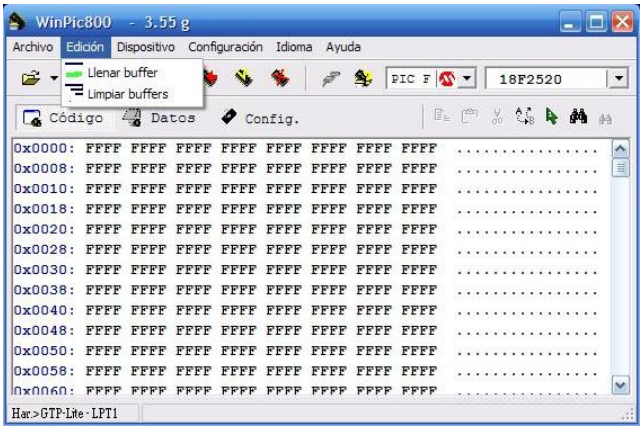


Fig. 2.31 Menú Edición

  - Llenar Buffer: Se establece el valor hexadecimal introducido en las posiciones de memoria desde el valor de inicio, hasta el valor final dados.
  - Limpiar Buffers: Establece el valor hexadecimal FFFF para todas las posiciones de memoria.

- Menú Dispositivo.

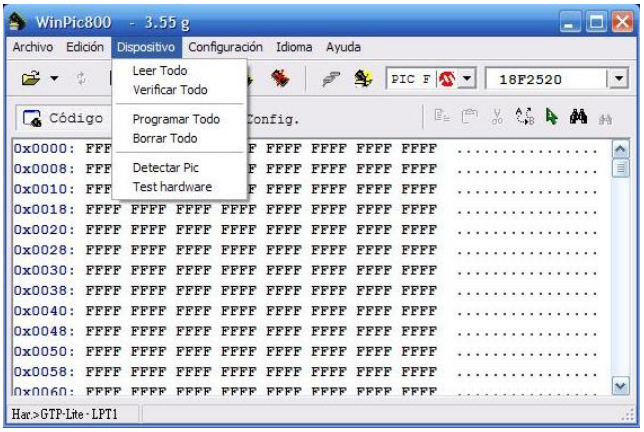


Fig. 2.32 Menú Dispositivo

  - Leer Todo: Con esta opción se lee la memoria ROM y EEPROM del PIC.
  - Verificar Todo: Comprueba si el código grabado en el PIC es igual al código abierto en el WinPIC800.
  - Programar Todo: Graba en el PIC el código que hay actualmente cargado en el WinPIC800, antes de programar se realiza un borrado del PIC automáticamente.
  - Borrar Todo: Realiza un borrado de las memorias ROM y EEPROM del PIC.
  - Detectar PIC: Al pinchar sobre esta opción se detecta el modelo del PIC que está insertado en el grabador.
  - Test Hardware: Comprueba que hay interconexión con el grabador.

- Menú Configuración.

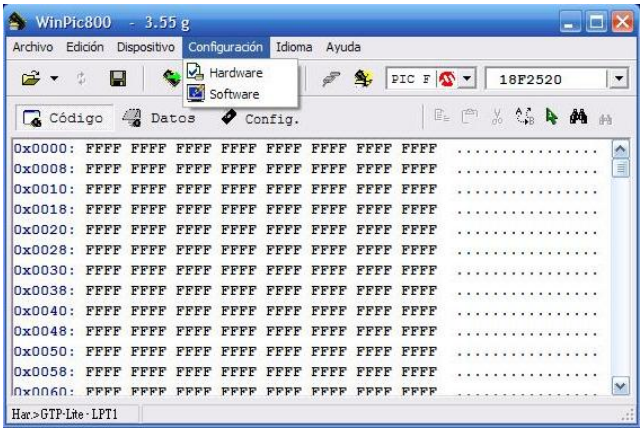


Fig. 2.33 Menú Configuración

  - Hardware: Esta opción lleva a la ventana de selección del grabador que se esté utilizando.
  - Software: Al pinchar sobre él, se pueden configurar algunas de las opciones del WinPIC800.

Al seleccionar la opción Hardware aparece una ventana como la mostrada en la figura 2.34.

Fig. 2.34 Ventana de configuración de Hardware

En esta ventana se muestran los programadores actualmente soportados por el WinPIC800, se debe seleccionar de los disponibles en la lista. Se selecciona el programador GTP Lite, el programador creado para este proyecto y que gracias a Sisco está incorporado en el WinPIC800. La casilla de Bloqueo Configuración debe ser desmarcada cuando por ejemplo, se quiera realizar un Test en las líneas VPP, VCC, Data, Clock, para verificar su correcto funcionamiento. El icono de Información muestra algunos datos sobre el programador, su creador, y donde puede ser descargado.

La opción Software muestra la ventana de la figura 2.35.

- Verificar tras la programación: Con esta opción seleccionada se verifica automáticamente el código después de haber sido grabado en el PIC.
- Avisar antes de borrar y programar: Esta opción es útil si por error seleccionamos una de esas dos opciones, ya que de otra forma se podría eliminar la memoria del PIC, sin posibilidad de recuperación.

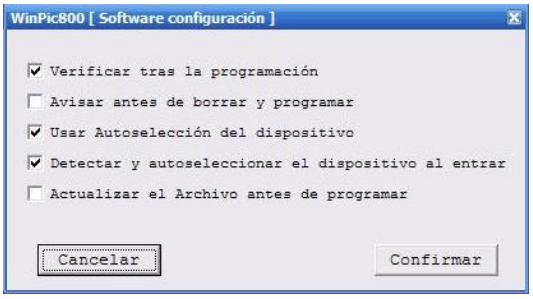


Fig. 2.35 Ventana de configuración de Software

- Usar auto selección del dispositivo: Al seleccionar esta opción, el software guarda cual ha sido el último PIC usado y lo auto selecciona la siguiente vez que se ejecute el software.
- Detectar y auto seleccionar el dispositivo al entrar: Con esta opción se detecta el PIC insertado en el grabador y se auto selecciona para su posterior grabación/borrado, etc.
- Actualizar el archivo antes de programar: De esta forma se asegura que el .hex grabado al PIC sea el último que se haya compilado.

- Menú Idioma:

- Recientemente Sisco, programador del WinPIC800, ha añadido traducción para el Software en distintos idiomas, cabe recordar que el Software es originalmente Español.

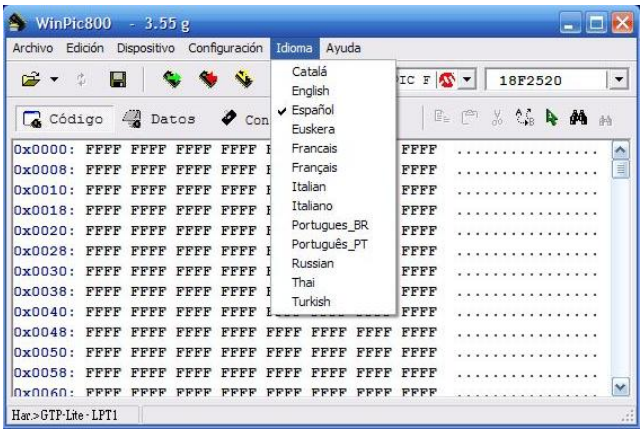


Fig. 2.36 Menú Idioma

- Menú Ayuda:

- Parámetros: Al entrar en esta opción del menú, se muestran los parámetros válidos para utilizar el software desde la consola de Windows.
- Acerca de WinPIC800: Al pinchar sobre esta opción se abre una ventana con información sobre el programador de este software, página personal, correo, etc.

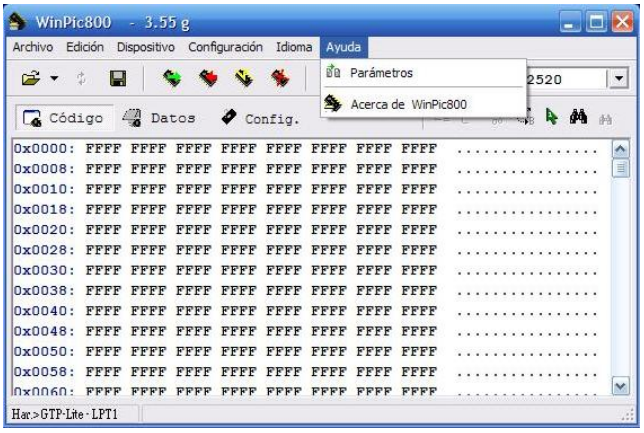


Fig. 2.37 Menú Ayuda

2.2.4 Hardware de grabación

- **GTP Lite**

Una vez se ha compilado el programa de la aplicación haciendo uso del compilador CCS, se obtiene un archivo *.hex, el cual debe ser cargado por el software de grabación WinPIC800. El siguiente paso consiste en transferir esa información hexadecimal, desde el PC hacia la memoria FlashROM del PIC, donde es grabada en forma binaria.

La información transmitida desde el PC puede hacerse a través del puerto serie, el puerto paralelo o el puerto USB. En el mercado se pueden encontrar varios grabadores para cada uno de los posibles puertos de comunicación. Se ha diseñado un programador acorde a las necesidades del proyecto y a la vez que pueda ser utilizado por la gran mayoría de PICs del mercado además de barato de realizar.

GTP Lite (Grabador TodoPIC Lite) es un grabador para PICs por el puerto paralelo y con alimentación externa, se ha decidido no incluir los zócalos de programación en la misma placa, puesto que esto agrandaría la placa del diseño y su vida quedaría limitada hasta el momento que Microchip sacara al mercado nuevos PICs con otro método ó patillas de grabación. Actualmente el GTP Lite soporta todos los PICs de las series: 10, 12, 16, 18, y 30 (DsPIC).

Ya que, como se ha dicho, no tiene placa de zócalos, como puede tener cualquier programador comercial, Micro´Pic Programmer, Ludipipo II, T20-SE, por citar algunos ejemplos, este presenta una salida ICSP (In-Circuit Serial Programming).

Gracias a esta salida ICSP, se puede programar el PIC en la propia placa del proyecto, de esta forma no se tiene que estar sacando el PIC cada vez que deba ser programado. Otra posibilidad es hacer un pequeño zócalo sobre el que vaya colocado el PIC a programar y al que vayan conectadas cada una de las líneas de programación del conector ICSP, siendo estas, VPP (Tensión de programación), VCC (Tensión de alimentación del circuito), GND (Masa del circuito), PGC (Señal de reloj de entrada al PIC), PGD (Señal de datos de entrada/salida del PIC).

En la figura 2.38 se muestra el esquemático del GTP Lite:

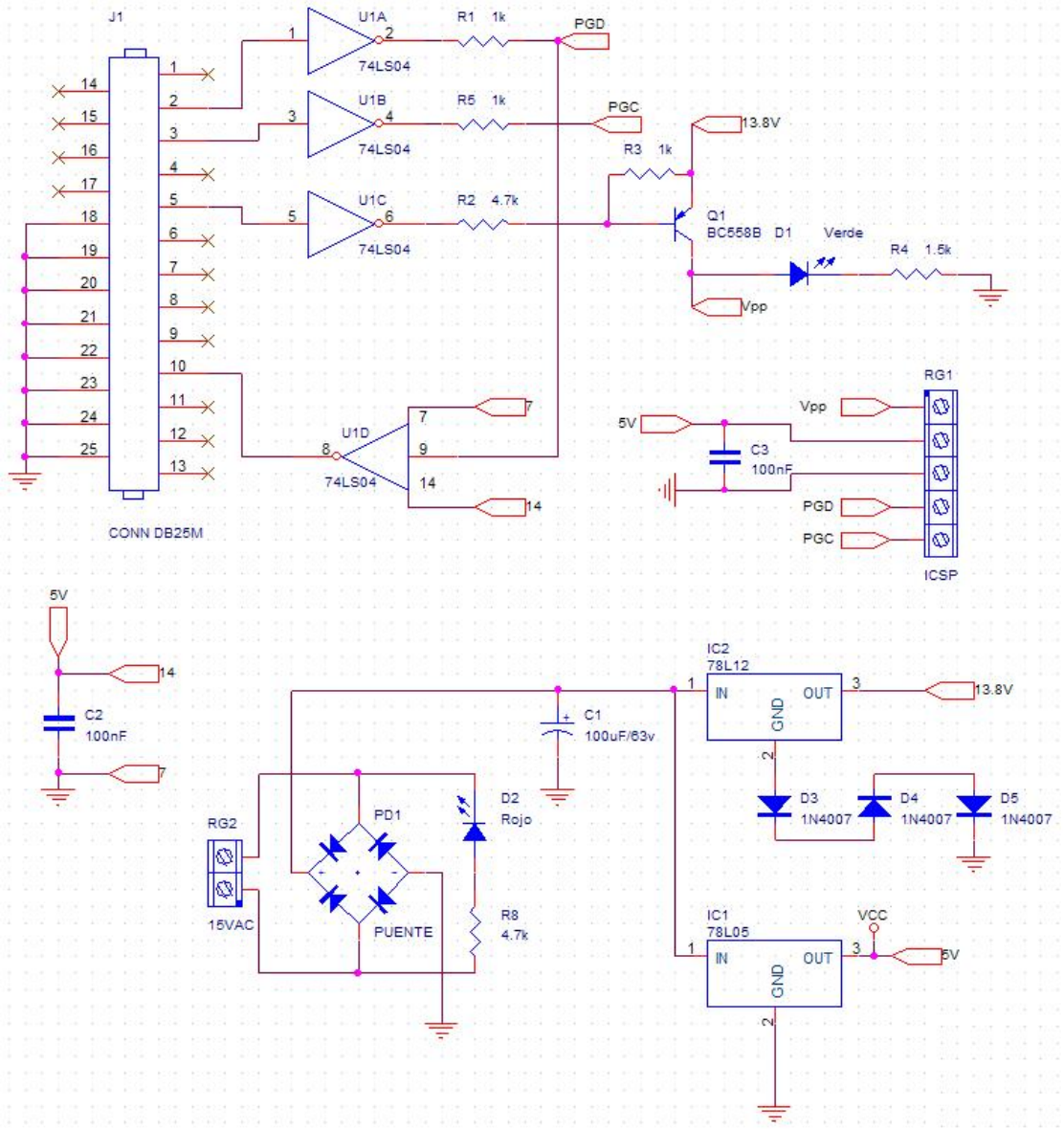


Fig. 2.38 Esquemático del GTP Lite

Este programador se ha realizado tomando ideas sobre el diseño de otros programadores, GTP, PP2, MicroPic Trainer, Kit182, ellos han sido los “padres” del GTP Lite.

Se ha tratado de realizar un grabador compacto, de fácil uso y construcción, además de que fuera compatible con el que actualmente es el mejor software de grabación para PICs, el WinPIC800.

En la actualidad, el programador de este software libre, Sisco Benach, ha añadido el GTP Lite a la lista de programadores compatibles con el WinPIC800, por lo tanto puede ser seleccionado como hardware en la configuración de este programa. Ha añadido también un link para la descarga del programador desde la página personal de quien escribe estas líneas.

La mayoría de transformadores que se pueden encontrar en una tienda normal, dan una salida máxima de 12VCC y la tensión requerida para programar un PIC es de unos 13VCC, dependiendo del modelo del PIC y para el modo de grabación HVP (High voltage programming). Se debe, por lo tanto, hacer una pequeña modificación en el transformador.

Está modificación consiste en anular la parte correspondiente a la rectificación, teniendo de esta forma un transformador de 15VCA a partir del de 12VCC. La empresa Microsystem Engineering suministra estos transformadores modificados, por si no se deseara realizar por cuenta propia.

- Descripción del circuito:

Como se acaba de comentar, la entrada al circuito es de 15VCA, esta tensión es rectificada utilizando un puente de diodos de 1A y convenientemente filtrada para eliminar el rizado (Figura 2.39), para ello se utiliza el condensador electrolítico C1 de 100uF/63v. Se ha incluido también un diodo led rojo, el cual se enciende al alimentar el circuito. A la salida del condensador de filtrado se tiene una tensión que depende de las características del transformador de 6€ y que suelen ser algo inestables, aunque no es crítico para esta aplicación, ya que como ahora se verá, se hace uso de reguladores de tensión. La salida de este bloque será de aproximadamente: $\left(15*\sqrt{2}\right)-\left(0{,}6*2\right)=20V$

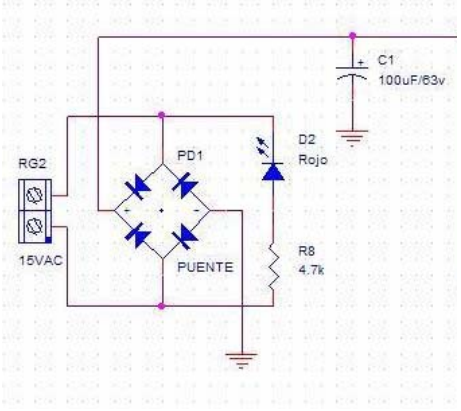


Fig. 2.39 Rectificador

La salida del bloque anterior, es la entrada de este nuevo bloque, del que se obtienen la tensión de programación (VPP) del PIC y la de alimentación (VCC). Para obtener la tensión de entrada al modo de programación (VPP), que es de aproximadamente 13V, se utiliza uno de los montajes típicos del datasheet del 7812. El cuál, consiste en conectar diodos en serie en el terminal GND, elevando de esta forma la tensión a la salida del regulador de tensión 7812, a un valor a la salida, igual a la suma de las caídas de tensión en cada uno de los diodos (Figura 2.40). Puesto que son 1N4007 y la caída de tensión en cada uno de ellos, es aproximadamente 0,6V, se tiene por tanto, un valor a la salida del regulador de tensión de 13,8V. Para obtener la tensión de alimentación (VCC) se utiliza un 7805 en su modo de configuración normal.

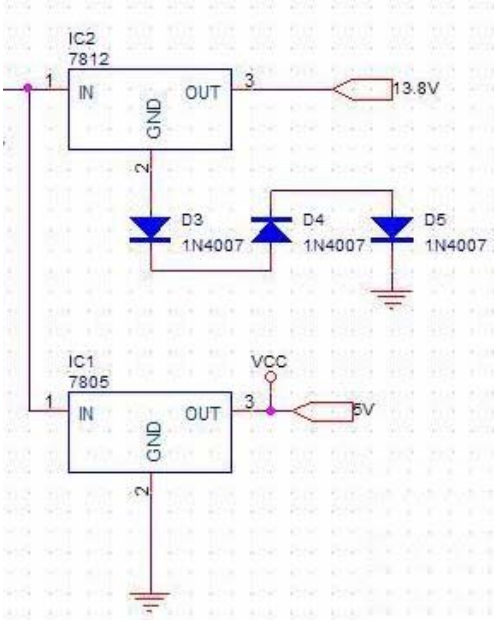


Fig. 2.40 Reguladores

Para no demandar demasiada corriente al puerto paralelo y a la vez obtener los niveles de tensión TTL necesarios para el circuito, se utiliza el inversor 74LS04. De esta forma se consigue una compatibilidad de uso tanto en PCs de sobremesa, como en portátiles, donde la corriente suministrada por el puerto paralelo, en ocasiones, no llega a satisfacer los niveles requeridos para la programación del PIC. Puesto que se debe mantener una compatibilidad total con el WinPIC800, se debe realizar la misma conexión con los pines del puerto paralelo que en el GTP original. El inversor “A” del 74LS04 es utilizado por la línea de datos (PGD) a introducir en el PIC a programar, esta línea de datos vuelve a ser introducida al puerto paralelo a través del inversor “D”, de esta manera se puede detectar el modelo del PIC y verificar su correcta grabación.

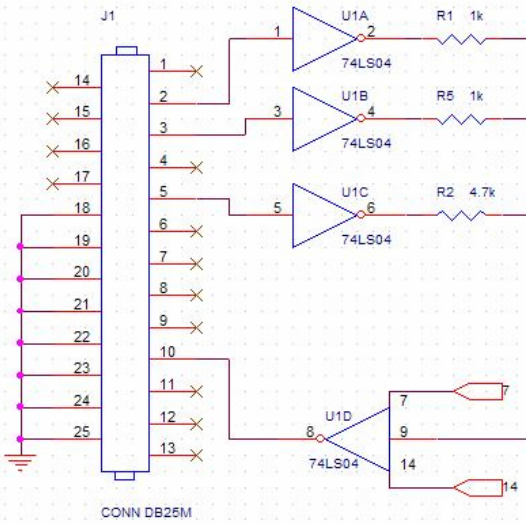


Fig. 2.41 Inversor 74LS04

El inversor “B” es el utilizado por la línea de reloj (PGC) para la sincronización en el envío de los datos al PIC a grabar.

El inversor “C” es utilizado para excitar la base del transistor PNP, de propósito general, BC558. De esta forma se consigue activar/desactivar la salida VPP (Figura 2.42), así como encender/apagar el led verde indicando cuando se ha entrado en el modo de grabación. La caída de tensión entre emisor-colector es de unos 0,7V, teniendo de esta forma, los 13V requeridos para la grabación del PIC.

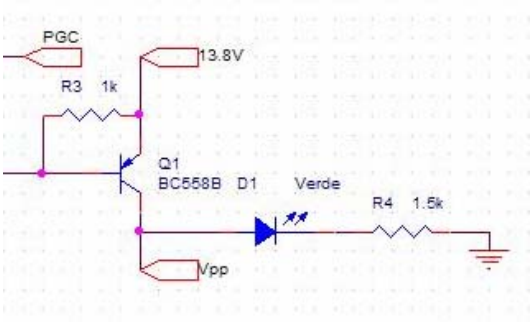


Fig. 2.42 Activación de VPP

Se muestra en la figura 2.43 el diseño de la placa y en las figuras 2.44 y 2.45 imágenes de la placa ya terminada:

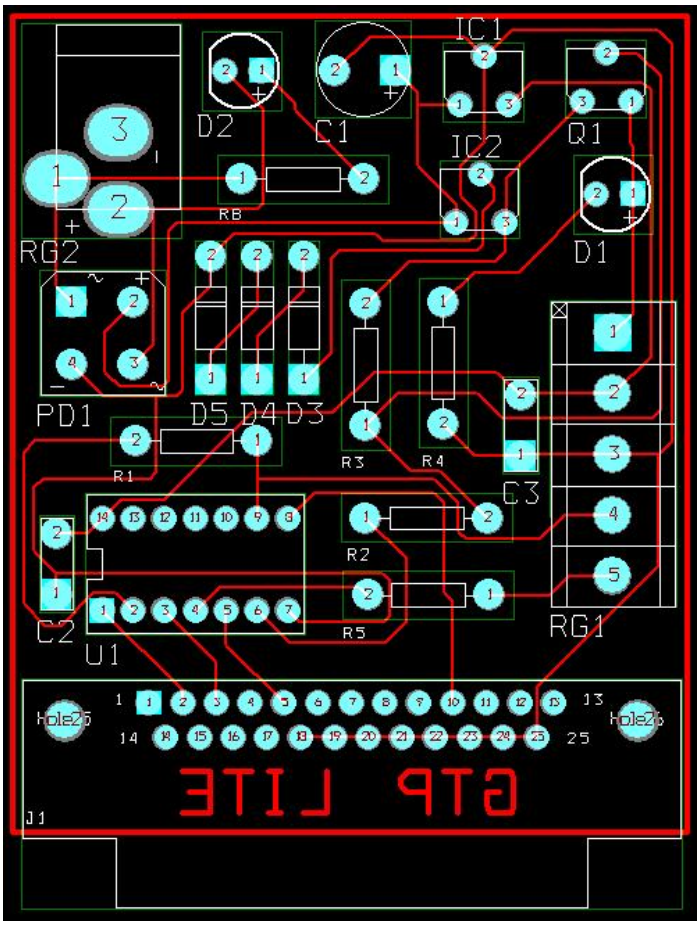


Fig. 2.43 Diseño de la placa

Fig. 2.44 Placa terminada. Vista Superior

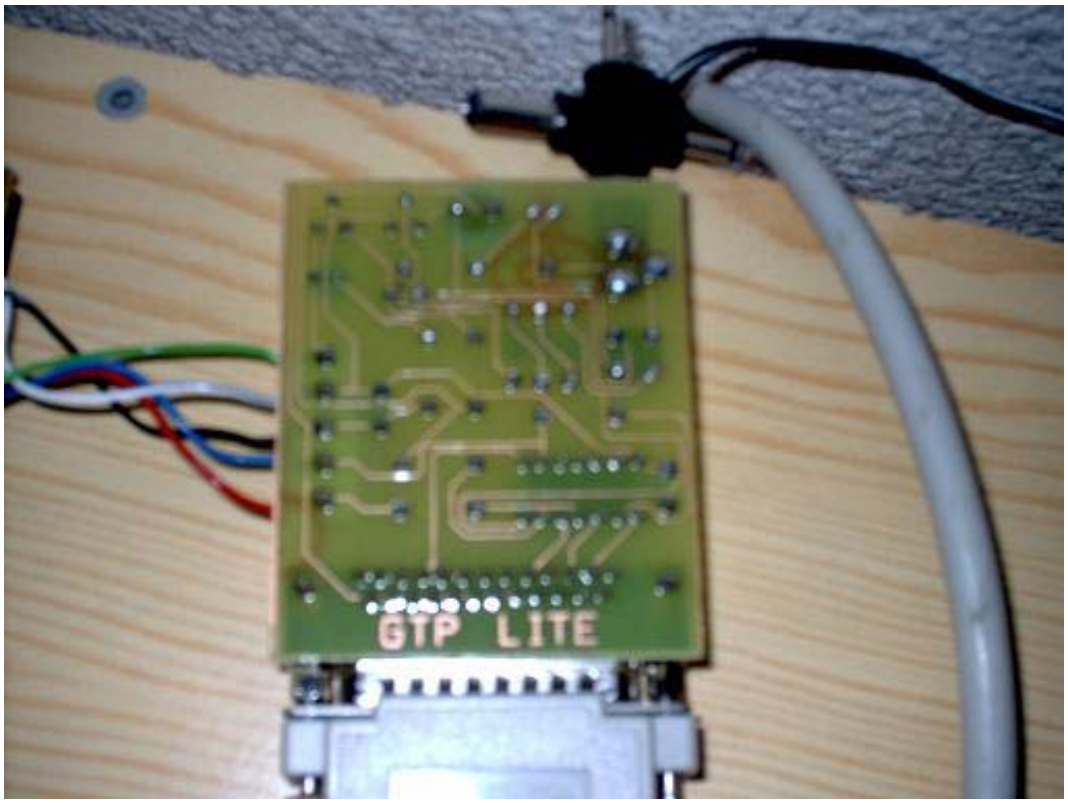


Fig. 2.45 Placa terminada. Vista Inferior

- **GTP USB Lite**

El grabador GTP Lite cumple perfectamente con los requerimientos para los que ha sido diseñado, no obstante cabe mencionar el programador que ha sido utilizado en la última etapa de realización de este proyecto fin de carrera.

GTP USB Lite es un grabador ICSP para PICs con memoria FlashROM basado en el GTP USB de Sisco Benach. No requiere de alimentación externa, ya que se alimenta a través del puerto USB. Para la comunicación con el PC vía USB se utiliza un PIC 18F2550, este PIC perteneciente a la serie 18Fxx5x, tiene la particularidad de llevar integrado el interfaz USB, para ser utilizado en modo SLAVE.

El firmware del GTP USB está realizado haciendo uso del compilador C18 de Microchip, debido a unas incompatibilidades con los chipset SIS de algunas placas base que provocan la perdida de comunicación entre el GTP USB y el PC, Sisco ha optado por desarrollarlo haciendo uso del CCS, consiguiendo eliminar el error.

Para el desarrollo del firmware con CCS, concretamente la parte del USB, se ha apoyado en un ejemplo llamado PicUSB, en el se introduce al envío/recepción de datos por USB haciendo uso de un PIC 18F2550 utilizando el compilador CCS y Visual C# para la aplicación del PC. Este ejemplo ha sido realizado por quien redacta estas líneas y me enorgullece enormemente haber sido de ayuda a quien considero un maestro. Puesto que no es objetivo de este proyecto ahondar en ese tema, se expone a continuación la página web donde se puede obtener más información sobre el PicUSB (http://perso.wanadoo.es/j1m y http://www.hobbypic.com).

Cabe destacar que el proyecto GTP USB es Freeware, tanto el propio programador, como su firmware, así como el WinPic800. Puesto que es Freeware, pero no Código Abierto, se explicará a continuación el funcionamiento del circuito del GTP USB Lite sin poder ahondar demasiado.

En la imagen 2.46 se puede ver el esquemático del GTP USB Lite:

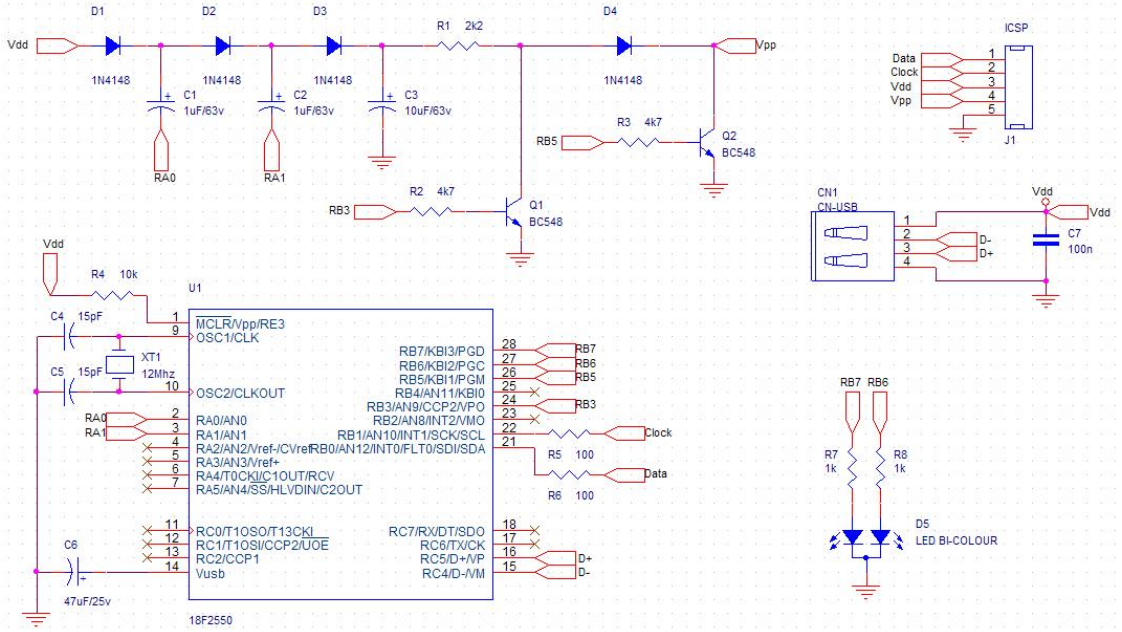


Fig. 2.46 Imagen global del esquemático GTP USB Lite

Como se ha mencionado anteriormente, GTP USB Lite es un grabador ICSP para PICs con memoria FlashROM. Se ha realizado basándose en el GTP USB y el GTP USB F1 (versión SMD del GTP USB).

- Descripción del circuito.

El GTP USB Lite hace uso de un PIC 18F2550 para realizar la comunicación a través del puerto USB del PC. (Figura 2.47) El cristal usado para el grabador es de 12Mhz, pero haciendo uso del PLL y divisor interno del microcontrolador se consigue alcanzar los 48Mhz necesarios para establecer la conexión característica del USB 2.0 de 12Mbps, o lo que es lo mismo 1,5 MBytes/s. El condensador C6 de 47uF/25 se utiliza para estabilizar el regulador de 3,3V interno, siendo este el encargado de alimentar el transceptor del USB. De los pines RB0 y RB1

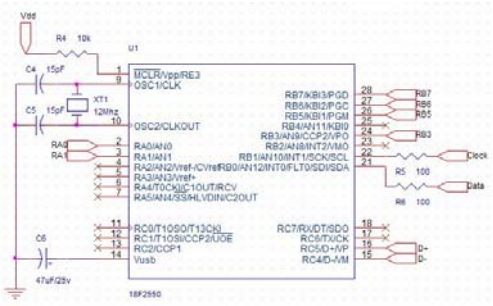


Fig. 2.47 Conexiones del 18F2550

salen respectivamente las líneas de Data y Clock hacia el conector ICSP, las resistencias de 100Ohm se encargan de limitar la corriente entregada.

Para conseguir la tensión necesaria para la programación se utiliza un elevador de tensión (Figura 2.48) controlado desde el PIC, de esta forma generando la secuencia 00, 10, 01, 11 para las patas RA0 y RA1 respectivamente, se consigue la tensión de aproximadamente 13V en la salida VPP.

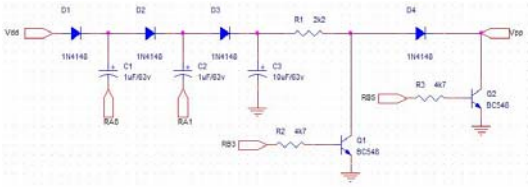


Fig. 2.48 Elevador de tensión

Cada uno de los terminales de la salida ICSP se conecta a las patas correspondientes del PIC a programar, en la Fig. 2.49 puede apreciarse un esquema de conexión para los PICs de uso más frecuente.

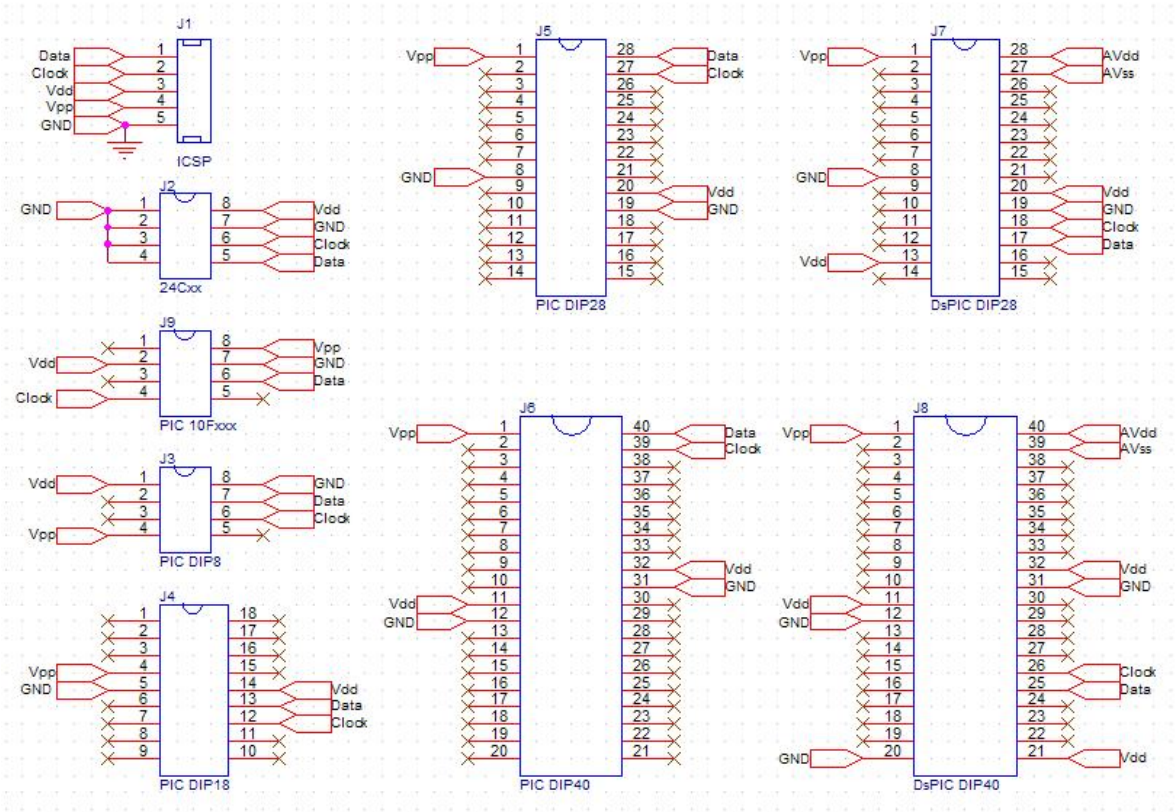


Fig. 2.49 Conexiones al PIC desde el conector ICSP

- Funcionamiento:

Como se ha comentado anteriormente el firmware que usa el GTP USB Lite para el 18F2550, es el del GTP USB, siendo este propiedad de Sisco Benach. Por lo que para conseguirlo se debe contactar con él.

Al conectar el GTP USB Lite por primera vez al PC a través del puerto USB, se debe instalar el driver del dispositivo. El driver básico es suministrado por Microchip para los PICs de la serie 18Fxx5x, que son los que llevan integrado el controlador de USB. Este driver va junto al software Winpic800ir. Si la instalación del driver ha sido realizada correctamente, el led bicolor de estado del grabador pasa de estar de color rojo, a estar verde. Esto significa que Windows ha enumerado correctamente el dispositivo y está listo para ser utilizado, además en el administrador de dispositivos aparece el nuevo dispositivo.

Fig. 2.50 GTP USB Lite instalado

Para poder utilizar este programador en el WinPIC800, se selecciona como hardware al GTP USB #Plus.

Se muestra en la imagen 2.51 el diseño de la placa y en las imágenes 2.52 y 2.53 la placa terminada.

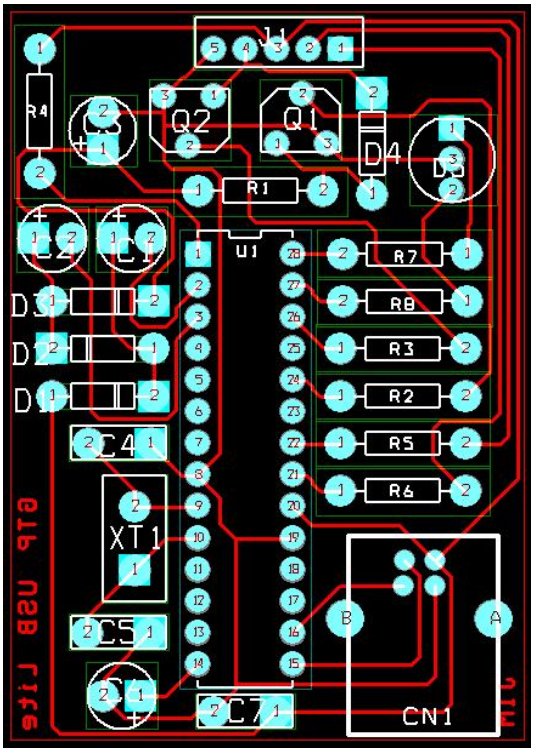


Fig. 2.51 Diseño de la placa

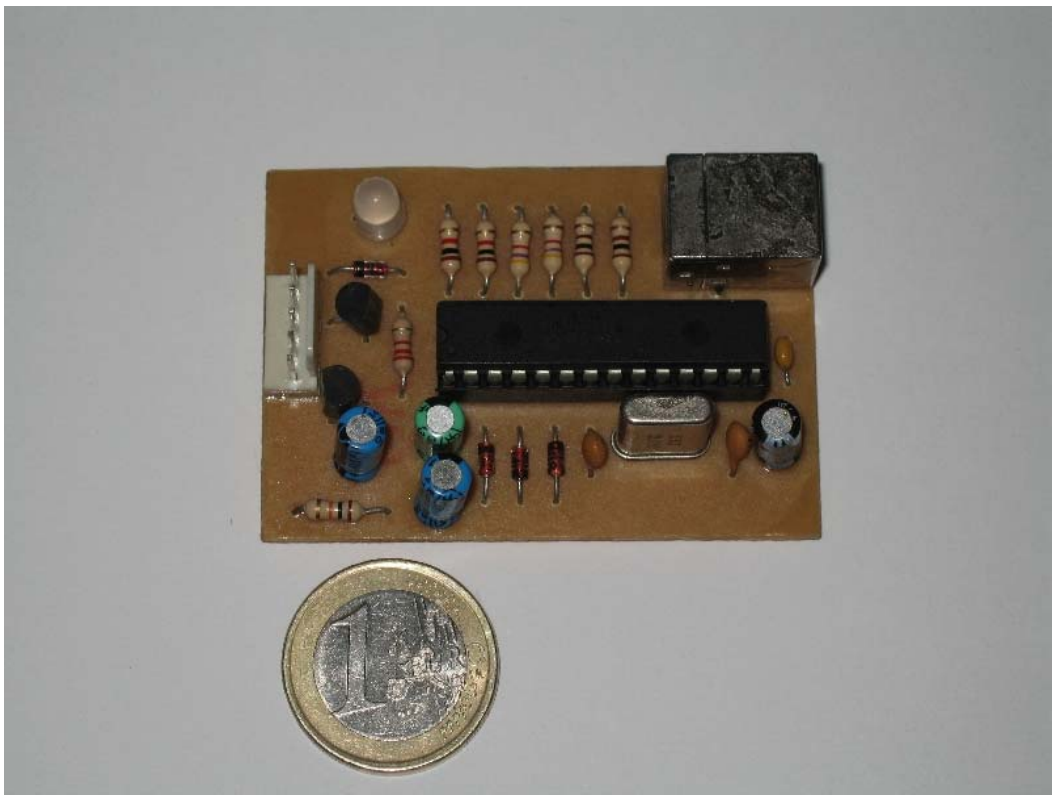


Fig. 2.52 Placa terminada. Vista Superior

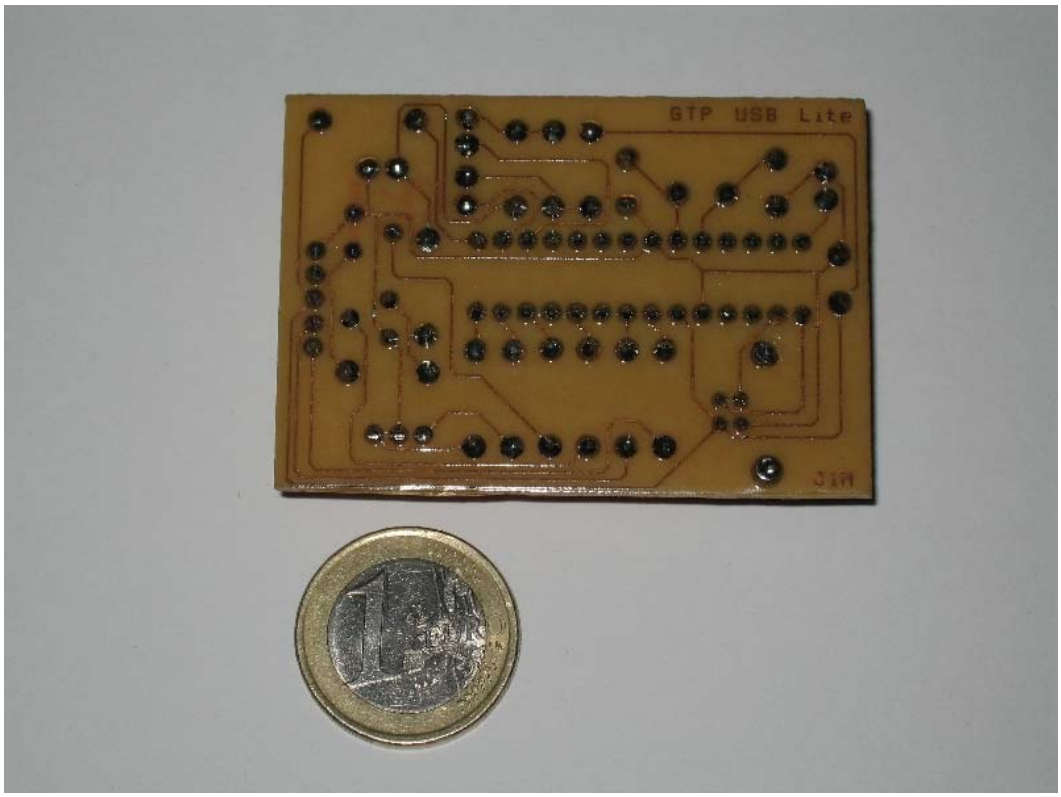


Fig. 2.53 Placa terminada. Vista Inferior

Capítulo 3
Desarrollo del diseño físico

3.1 Fuente de Alimentación

La etapa de alimentación (Figura 3.1) es una de las más importantes, por lo que es necesario un correcto modelado de esta para evitar posibles fallos en el circuito debidos a ruidos, cortes en la alimentación, corriente suministrada por el transformador insuficiente, etc.

Para la selección del transformador se han tenido en cuenta, las tensiones requeridas por el circuito, así como la intensidad que debe aportar. El circuito trabaja con dos tensiones, la de 5v para la electrónica asociada al microcontrolador y sensores, y 12v para los ventiladores. Además se tienen las cargas que funcionan a 220v, aunque estas no influyen en el dimensionado de esta etapa.

Puesto que la tensión máxima requerida es de 12v, el transformador debe al menos poder aportar esta tensión, por ello se ha utilizado uno de 220VCA/12VCA. Al rectificar la tensión de 12VCA del secundario, se tiene una tensión aproximada de 15,6 VCC.

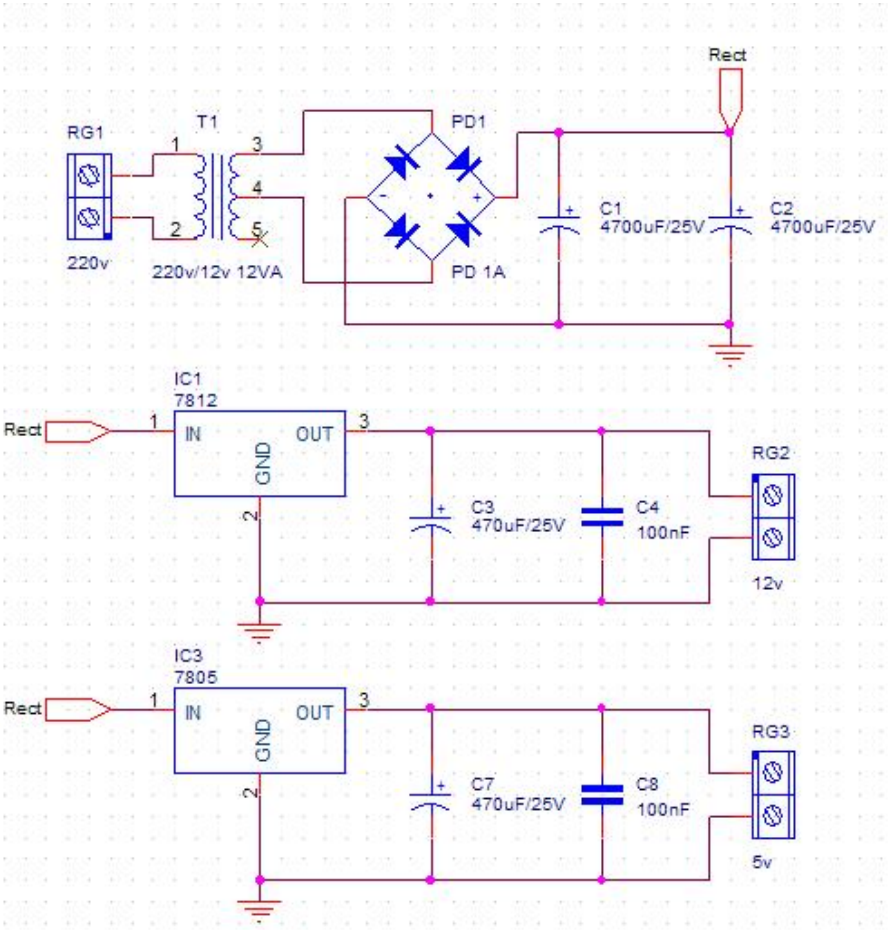


Fig. 3.1 Esquemático de la fuente de alimentación

Esta tensión rectificada y convenientemente filtrada, está lista para ser introducida en los reguladores de tensión 7812 y 7805, de esta forma se obtienen las tensiones necesarias de funcionamiento para el circuito. Los condensadores de rectificado, se han sobredimensionado para obtener una señal lo mas continua posible, y eliminar el gran rizado que pueden producir los ventiladores, debido a su gran demanda de corriente de unos 200mA.

Esta gran demanda de corriente ha sido otro factor importante a tener en cuenta en el momento de la selección del transformador. Los dos ventiladores requieren una intensidad de 360mA, sumado al consumo del resto de componentes, hacen que se requiera un transformador capaz de suministrar unos 500mA. El transformador que se ha podido encontrar, es uno capaz de aportar hasta 1A, el cual cumple perfecta y sobradamente con los requisitos del sistema.

Se ha seleccionado un puente de diodos de 1A para realizar el rectificado de la tensión senoidal entregada por el secundario del transformador.

Los reguladores de tensión, 7805 y 7812 se han montado siguiendo la configuración característica descrita en su datasheet. Cada uno de ellos es capaz de aportar hasta 1A, por lo que en este aspecto también son suficientes para cubrir las necesidades de intensidad del sistema. A cada uno de los reguladores se les acompaña a la salida con dos condensadores, un cerámico de 100nF para eliminar el ruido de alta frecuencia y otro de 470uF, sobredimensionado, para eliminar el ruido de baja frecuencia, y para que la tensión a la salida sea lo más lineal posible.

3.2 Etapa de Microcontrol

Esta etapa es la encargada de vincular todos las partes que conforman el diseño y enlazarlas al microcontrolador, para ello y puesto que el diseño se ha realizado como un prototipo, esta etapa la conforman tres partes.

Una placa base para el 18F2520 (Figura 3.2) donde todas sus líneas de E/S están conectadas al exterior, mediante terminales fácilmente intercambiables. Esto se ha realizado con la intención de facilitar el cableado hacia las otras dos placas de control.

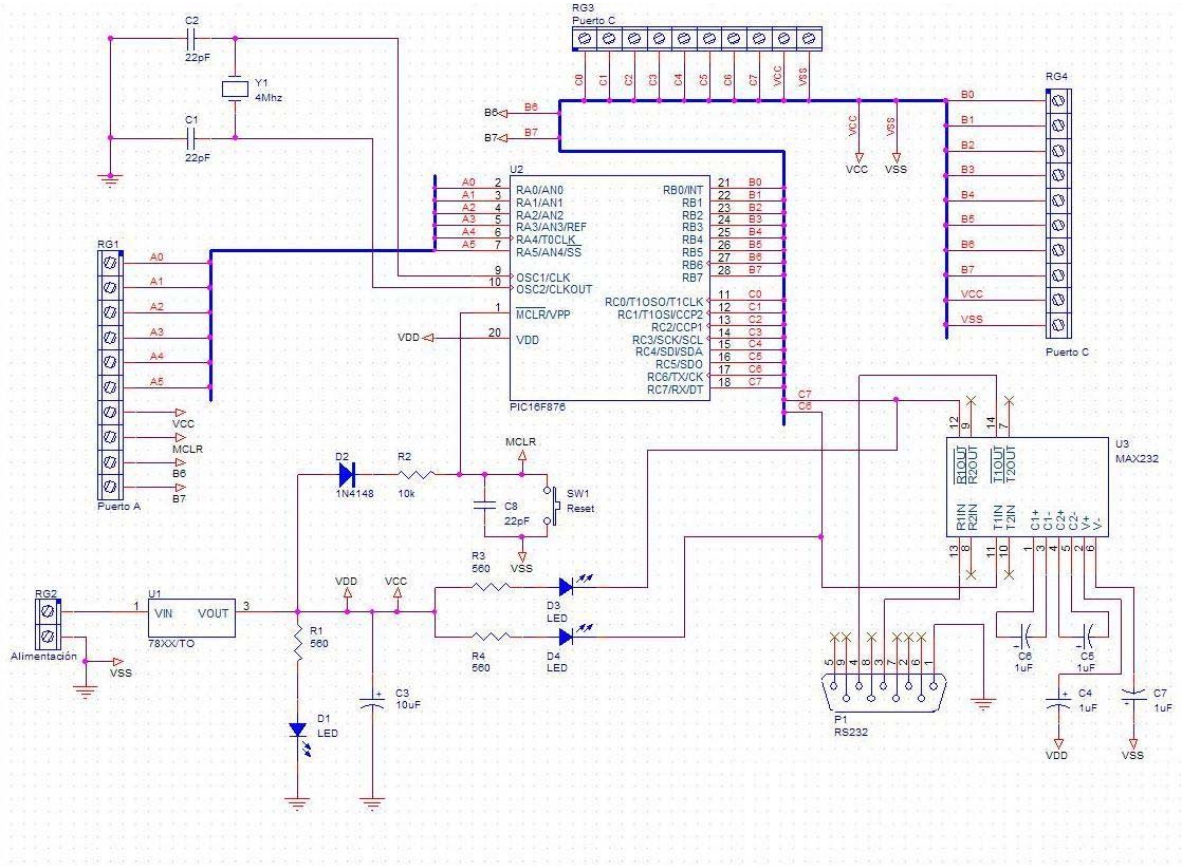


Fig. 3.2 Esquemático de la placa base del microcontrolador

Esta placa base tiene todas las E/S del microcontrolador conectadas al exterior para poder conectar a ellas fácilmente los dispositivos. Se puede observar en la parte superior izquierda el cristal de cuarzo de 4Mhz, junto a los condensadores de desacoplo de 22pF, conectados tal y como indica el datasheet de este microcontrolador, que puede ser consultado en el ANEXO IV.

En la parte inferior derecha se encuentra un MAX232 para realizar la comunicación serie RS-232 al PC, aunque finalmente no se ha utilizado para este proyecto, y será comentado como una opción en el capitulo 5.2 de vías futuras.

Adicionalmente se ha conectado un pulsador de reset del sistema. La placa ruteada se muestra en la figura 3.3.

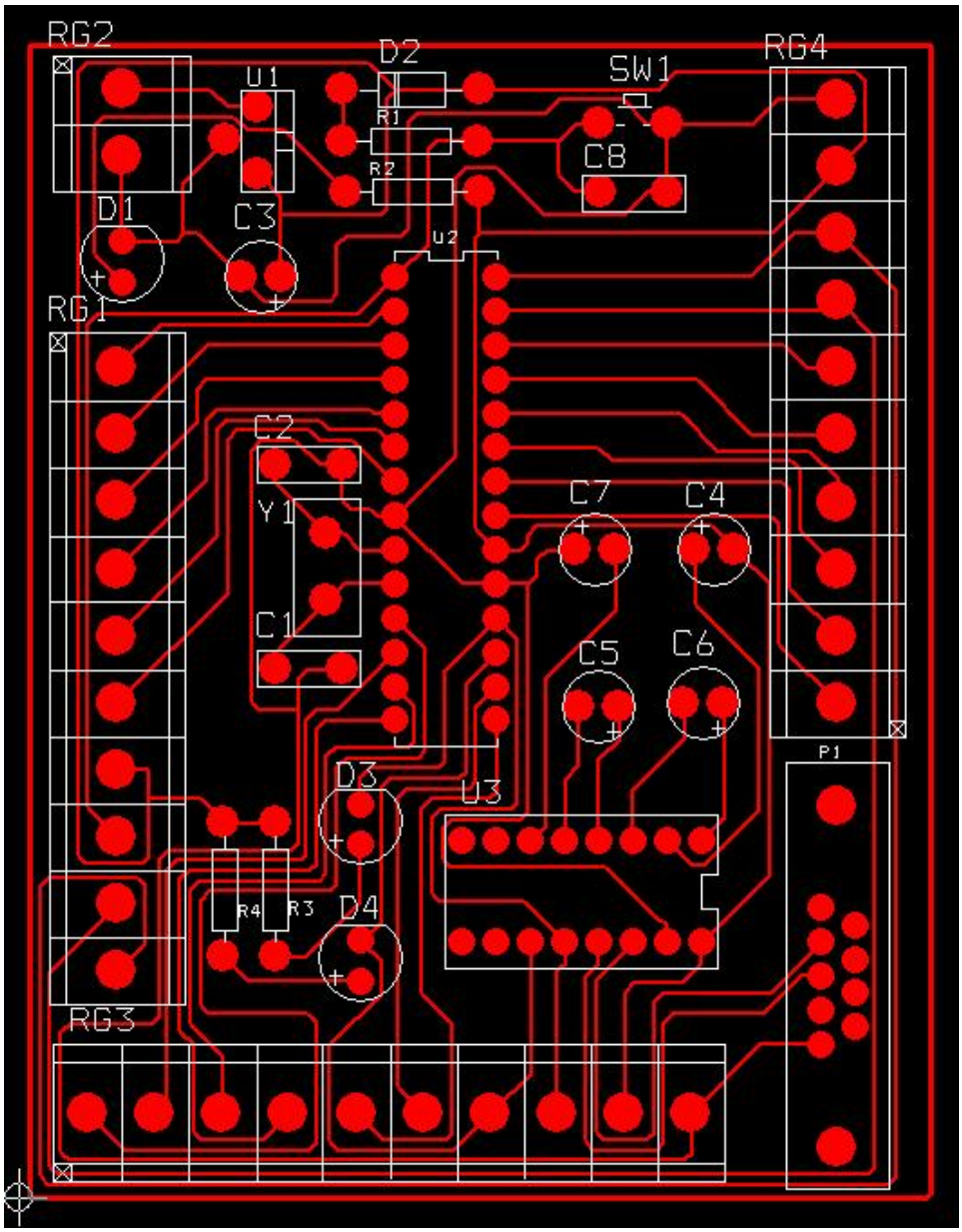


Fig. 3.3 Placa base ruteada

A esta placa van conectadas las dos placas que terminan de componer el sistema, la placa de interfaz con el usuario, y la pequeña placa de adaptación para el sensor SHT11. Esta es necesaria debido a su formato SMD y su imposibilidad de conexión directa a esta placa base. La placa de potencia es la encargada de conmutar las cargas de 12Vcc y 220Vca, será comentada en el apartado 3.3.

La interfaz con el usuario (Figura 3.4) consta de los pulsadores necesarios para que el usuario interactúe con el sistema, pudiendo de esta forma realizar su entera

configuración. Además a esta placa está conectado el LCD, en el cuál se puede visualizar el estado actual del sistema, así como los menús necesarios para la realizar la configuración de este.

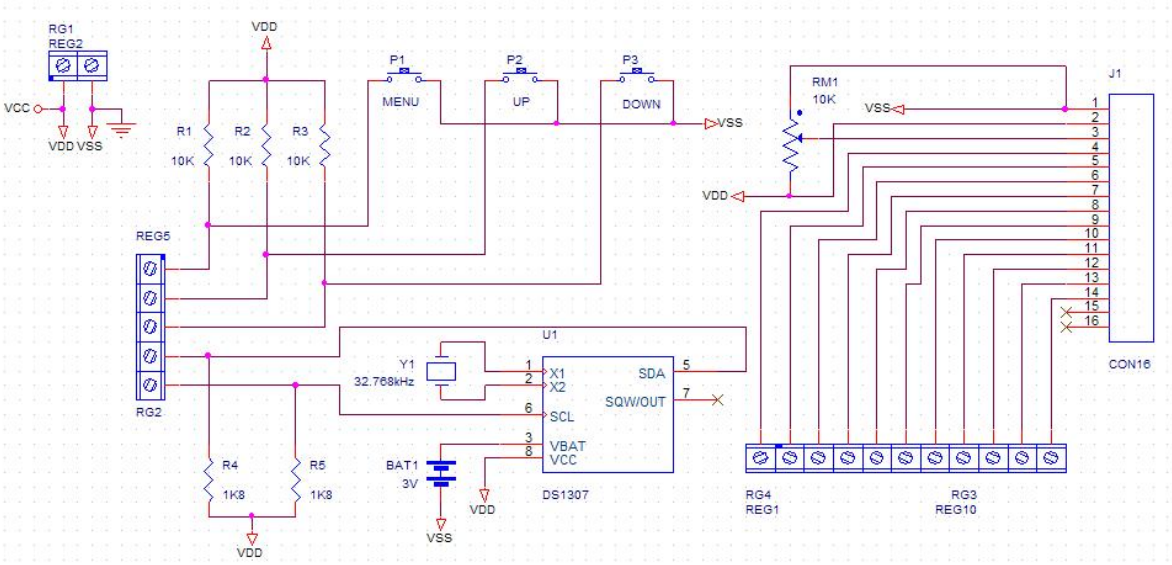


Fig. 3.4 Esquemático de placa interfaz usuario

En la imagen 3.4 se puede observar que los tres pulsadores están configurados para dar un nivel bajo al activarse. En la parte derecha se encuentra el diagrama de conexión al LCD, así como la resistencia variable para regular el nivel de contraste de la pantalla.

El RTC está situado en la parte inferior de la imagen, configurado tal y como indica su datasheet con un reloj de 32,768Khz, una batería de litio de 3V para evitar que se pierdan los datos en el caso de que se produzca un corte en la alimentación, y las resistencias pull-up para las líneas SDA y SCL.

En la figura 3.5 se muestra el resultado de la placa completamente ruteada con el OrCAD 10.

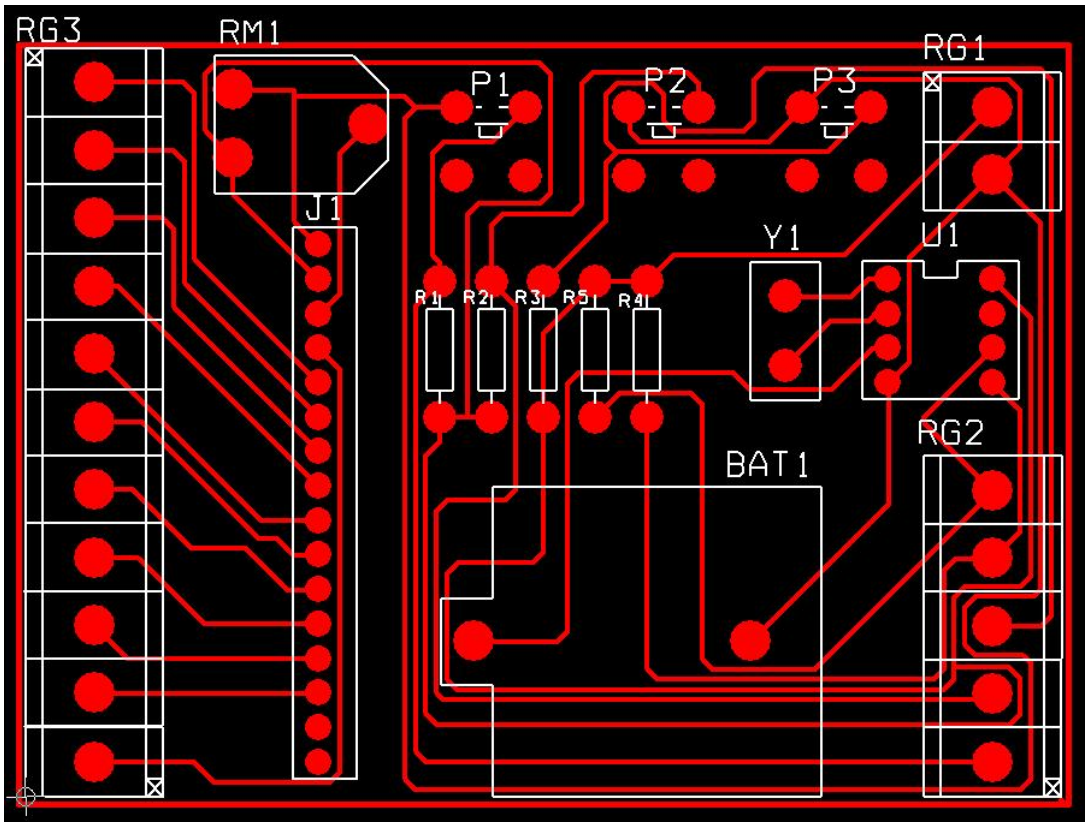


Fig. 3.5 Placa de interface con usuario ruteada

Como se ha comentado anteriormente, puesto que el sensor SHT11 viene en formato SMD, se ha realizado una placa de adaptación junto con el condensador de desacoplo de 100nF, conectado entre Vcc y Gnd tal y como indica el datasheet.

La figuras 3.6 y 3.7 muestran el esquemático y el ruteado, realizadas en Protel

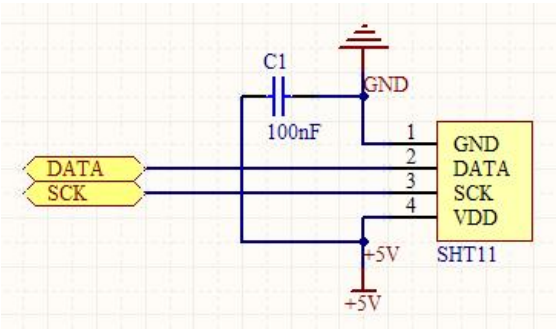


Fig. 3.6 Esquemático placa SHT11

Fig. 3.7 Placa SHT11 ruteada

Para concluir se muestran unas imágenes de las placas de microcontrol funcionando.

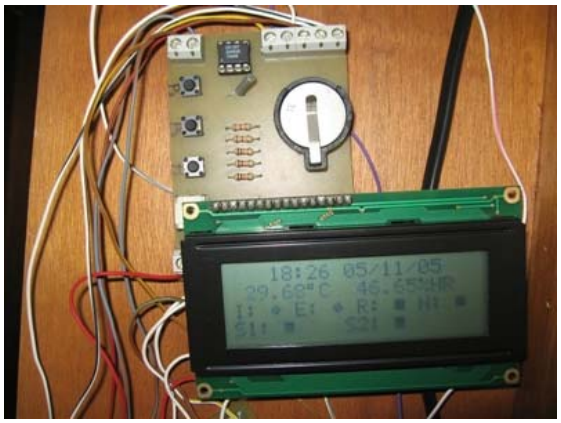


Fig. 3.8 Placa interface con usuario

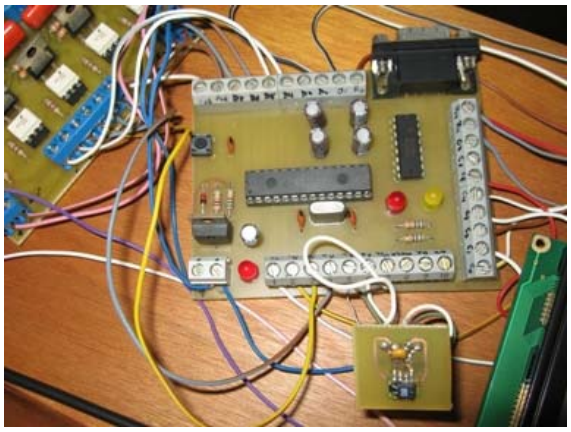


Fig. 3.9 Placa base y placa SHT11

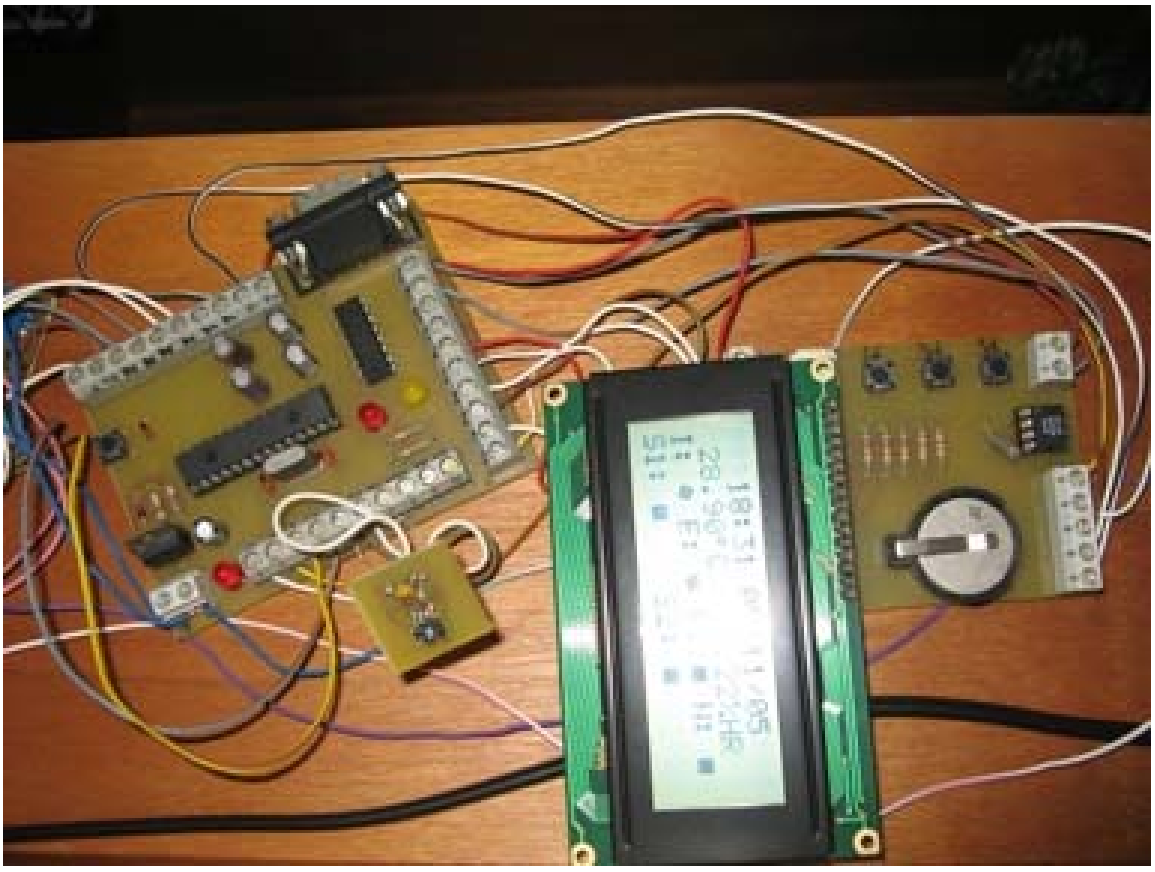


Fig. 3.10 Conjunto de placas de la etapa de microcontrol

3.3 Etapa de Potencia

Esta etapa es la encargada de conmutar las cargas de 12Vcc y 220Vca, debido a que estas tensiones, no son las normales de trabajo de microcontroladores, las cuales están entorno a los 5Vcc. Deben tomarse por lo tanto, unas medidas de protección adicionales para no destruir el circuito del microcontrolador.

A la etapa de potencia (Figura 3.11) llegan las señales de activación/desactivación de las cargas procedentes del microcontrolador, esto es, 5Vcc para nivel alto, 0Vcc para nivel bajo.

En la siguiente imagen, se muestra el esquema global de la placa de potencia, se puede ver en la mitad derecha, la regleta de entrada de las señales del microcontrolador IN1-IN7, con estas señales se controla el estado de las cargas.

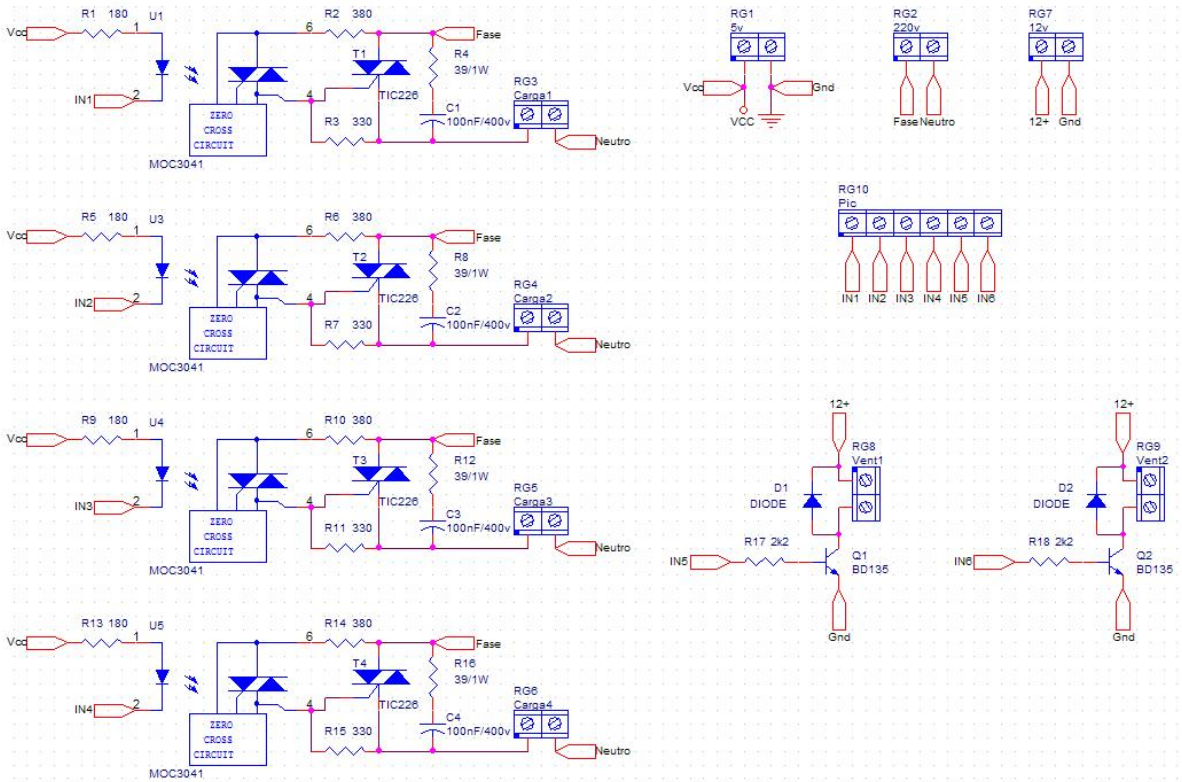


Fig. 3.11 Esquemático de la etapa de potencia

La placa se puede dividir en seis bloques, cuatro de ellos son los encargados de conmutar las cargas de 220Vca, y los dos restantes, de conmutar los ventiladores de 12Vcc.

Para conmutar los ventiladores se ha usado el esquema que se observa en la figura 3.12. Se utiliza un transistor NPN en modo de funcionamiento corte-saturación, de esta forma se realiza la conmutación de la carga de 12Vcc haciendo uso de una tensión de 5Vcc proveniente del microcontrolador. El transistor usado es un BD135, su datasheet puede encontrarse en el ANEXO VIII. Debido a que la intensidad de colector que debe soportar es de unos 200mA, y este transistor soporta intensidades de colector de hasta 1,5A, no hay problemas con la corriente que circule a través de él. Adicionalmente se ha colocado un diodo en inversa entre los terminales a conectar al ventilador, esto es debido a que al producirse el corte en la alimentación cuando el ventilador está girando, esté pasa de estar en estado 'motor' a estado 'generador', induciendo de este modo, una corriente hacia el circuito que puede ser destructiva para algún componente. Colocando el diodo en antiparalelo al producirse el corte en la alimentación, esa corriente residente en la bobina del motor del ventilador circulará a través de él hasta que se pare completamente.

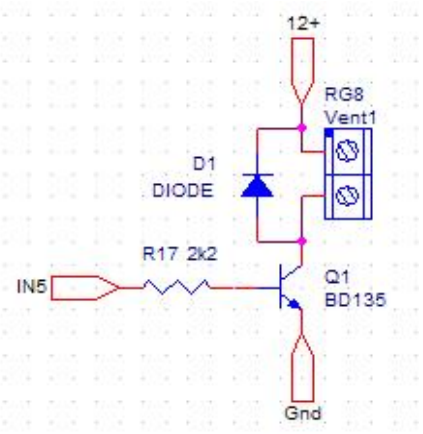


Fig. 3.12 control ventilador

Para conmutar las cargas de 220Vca se ha utilizado el circuito mostrado en la figura 3.13.

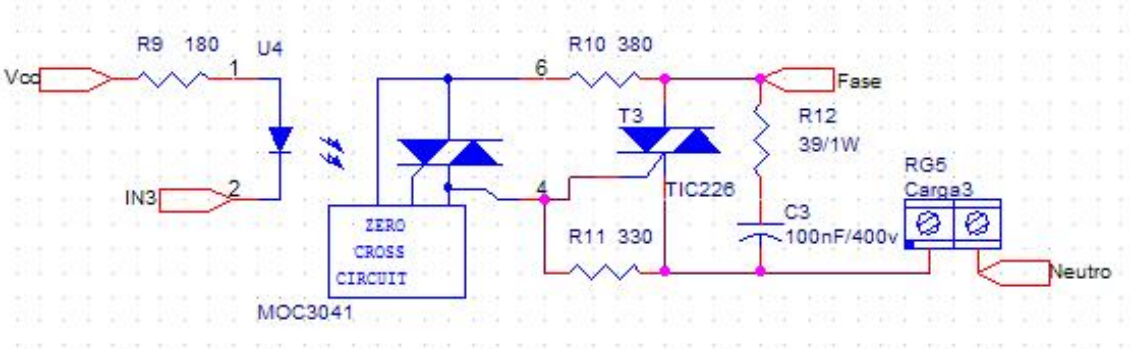


Fig. 3.13 control cargas 220v

Para aislar el circuito de continua del de alterna, se ha utilizado un optotriac, en concreto el MOC3041, en el ANEXO VI puede encontrarse su datasheet. Su bajo coste y su alto asilamiento además de sus características, hacen que este circuito integrado sea el más usado en estas situaciones. Este optotriac tiene integrado un circuito de paso por cero, por ello se evita toda la problemática causada por la conmutación de una señal cuando la senoidal de la red aún no es cero. Como dato curioso cabe mencionar que el 80% de los casos en los que se funden las bombillas, es debido a un fallo de alimentación en el momento que la tensión no es cero, lo que provoca un alto calentamiento del filamento en un bajo incremento de tiempo provocando su fundición.

Este optotriac está conectado tal y como indica su datasheet, hace uso de un triac que es quien realmente conmuta la carga de alterna, ya que como se ha indicado anteriormente, el optotriac es un elemento encargado de aislar la electrónica de microcontrol de la de potencia.

El triac usado es el TIC226, se ha añadido su datasheet como ANEXO VII. Este triac, es comúnmente utilizado en estos montajes y se ha configurado tal y como indica su datasheet. La resistencia de 39Ohm y el condensador de 220nF forman la red snubber necesaria para la conmutación de cargas inductivas, que al igual que en el caso anterior del ventilador, es necesario descargar energía acumulada en la bobina.

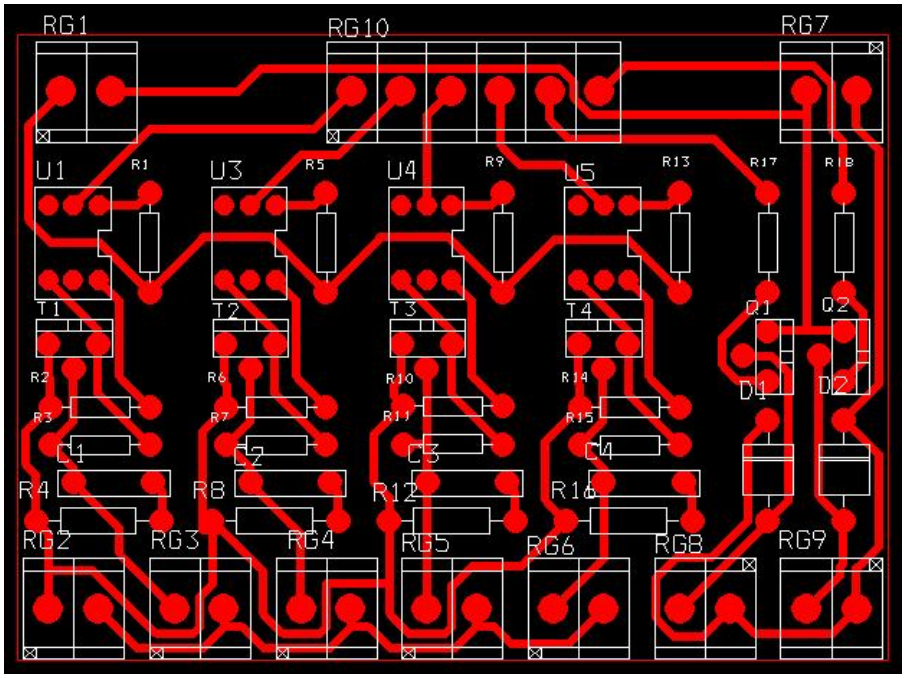


Fig. 3.14 Placa de potencia ruteada

En la figura 3.14 se puede ver una imagen de la placa ruteada, se muestra a continuación una imagen de la placa de potencia (Figura 3.15), y una imagen global del sistema en funcionamiento (Figura 3.16).

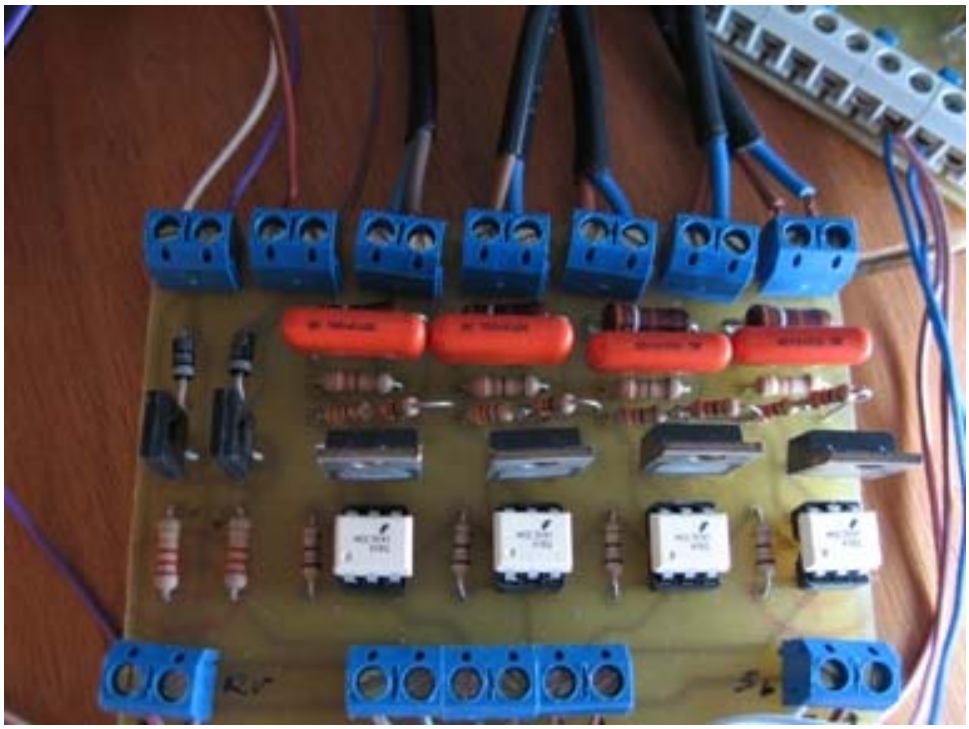


Fig. 3.15 placa potencia

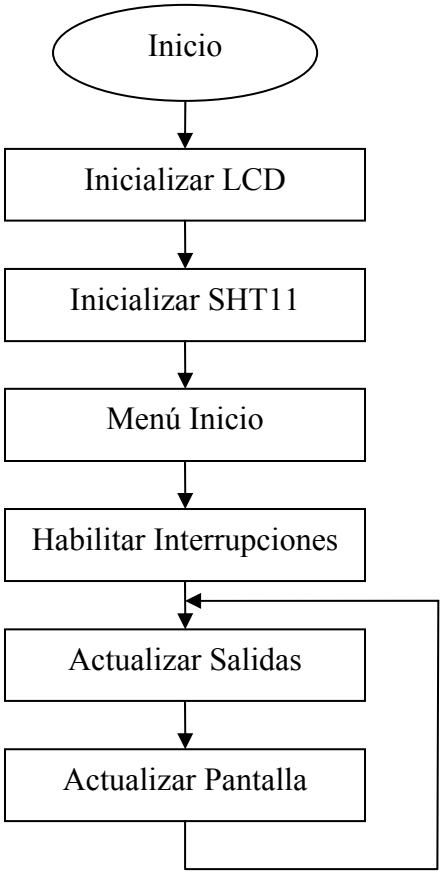
Fig. 3.16 vista global del sistema

Capítulo 4
Desarrollo del código fuente

4.1 Flujograma del sistema

En este apartado va a ser descrito el funcionamiento del sistema haciendo uso de organigramas, de esta forma los algoritmos pueden ser inteligibles de forma visual.

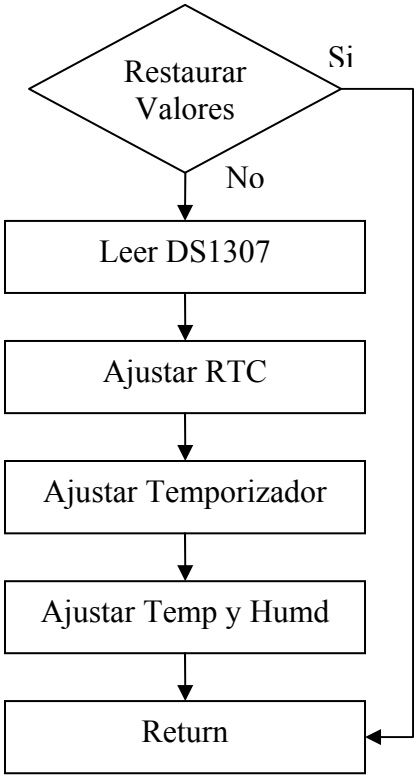
A continuación se muestra el organigrama global del sistema, para ir desmenuzándolo en las partes que lo forman.



Al alimentar el sistema se produce su inicialización, acto seguido es necesario inicializar el LCD para poder mostrar datos a través de él, seguidamente es inicializado el sensor de temperatura y humedad SHT11, de esta forma va a estar disponible para realizar mediciones de esas variables. A continuación se entra en el menú, este bloque será detallado a continuación debido a su complejidad para ser añadido en este esquema básico de funcionamiento. Seguidamente se habilitan las interrupciones TMR0, EXT, EXT1, y EXT2. Cabe destacar que el tratamiento de las interrupciones será descrito más adelante. Para terminar se entra en un bucle infinito donde son actualizadas las salidas y los datos en la pantalla, estas dos funciones serán analizadas más adelante.

- MENU INICIO

A continuación se describe el funcionamiento del bloque 'Menú Inicio' al iniciar el sistema.



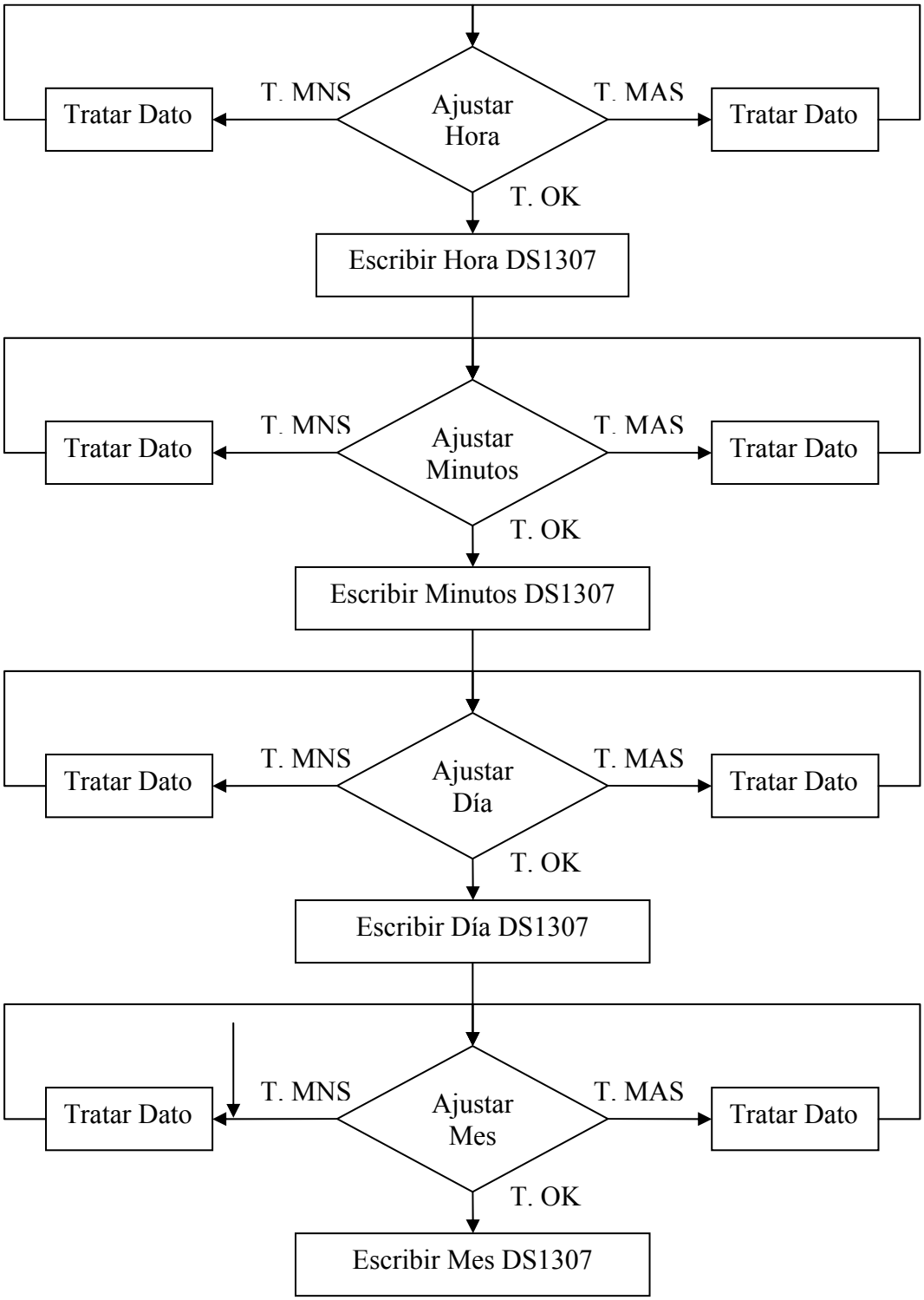
Dentro de este menú se da a elegir entre dos posibilidades, restaurar o no los valores en la EEPROM. Si es la primera vez que se inicia el sistema, se deben configurar los parámetros del sistema, por lo que debe elegirse la opción NO. En el caso de que el sistema haya sido configurado anteriormente se elige la opción SI y los valores son restaurados desde la EEPROM conforme vayan siendo necesarios para el sistema. Esto se ha hecho de esta forma para prevenir que el sistema quede inoperativo ante un corte de luz.

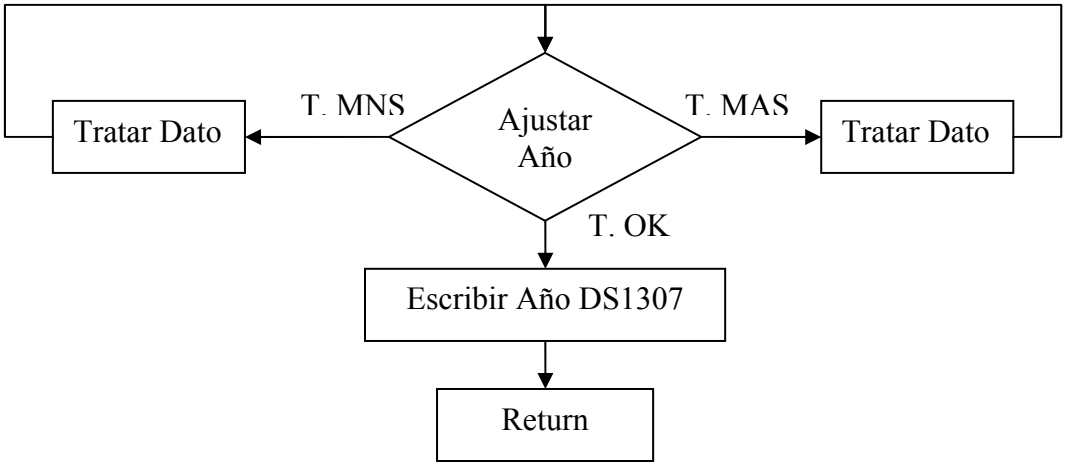
Es destacable el hecho de que, si ninguna tecla es pulsada, transcurridos 5 segundos, la opción por defecto es SI restaurar los valores.

En el caso de seleccionar la opción de NO restaurar valores, se procede a la configuración del sistema, para ello se realiza una lectura del reloj de tiempo real DS1307, esto es posible y útil realizarlo gracias a la batería de litio con la que es alimentado el reloj, tiene una duración aproximada de 10 años. Se procede entonces al ajuste del reloj de tiempo real (Real Time Clock), ajuste del temporizador, y ajuste de la

temperatura y humedad para el sistema. A continuación se detallan cada uno de los bloques de ajuste.

- AJUSTAR RTC





Para realizar el ajuste del DS1307 (RTC) se ha realizado un sencillo algoritmo para la petición de datos, su ajuste se realiza mediante el uso de tres teclas, T.Mas, T.Mns, T.Ok. Para ello, sobre el LCD se muestra el estado actual de cada una de las variables y se recorren cada una de ellas por si es necesario modificarlas, en el caso de que sea necesaria una modificación, pueden ser incrementadas/decrementadas con T.Mas/T.Mns respectivamente. Al realizar un incremento/decremento, cada variable es tratada dependiendo del valor introducido, de esta forma, por ejemplo, si se decrementa la variable Mes cuando tiene un valor de 1, pasa automáticamente al valor 12, evitando de esta forma errores por parte del usuario.

Una vez establecido el valor deseado para cada variable, se procede con la pulsación de la tecla T.Ok y el valor será enviado mediante I2C al dispositivo DS1307 y almacenado en su memoria.

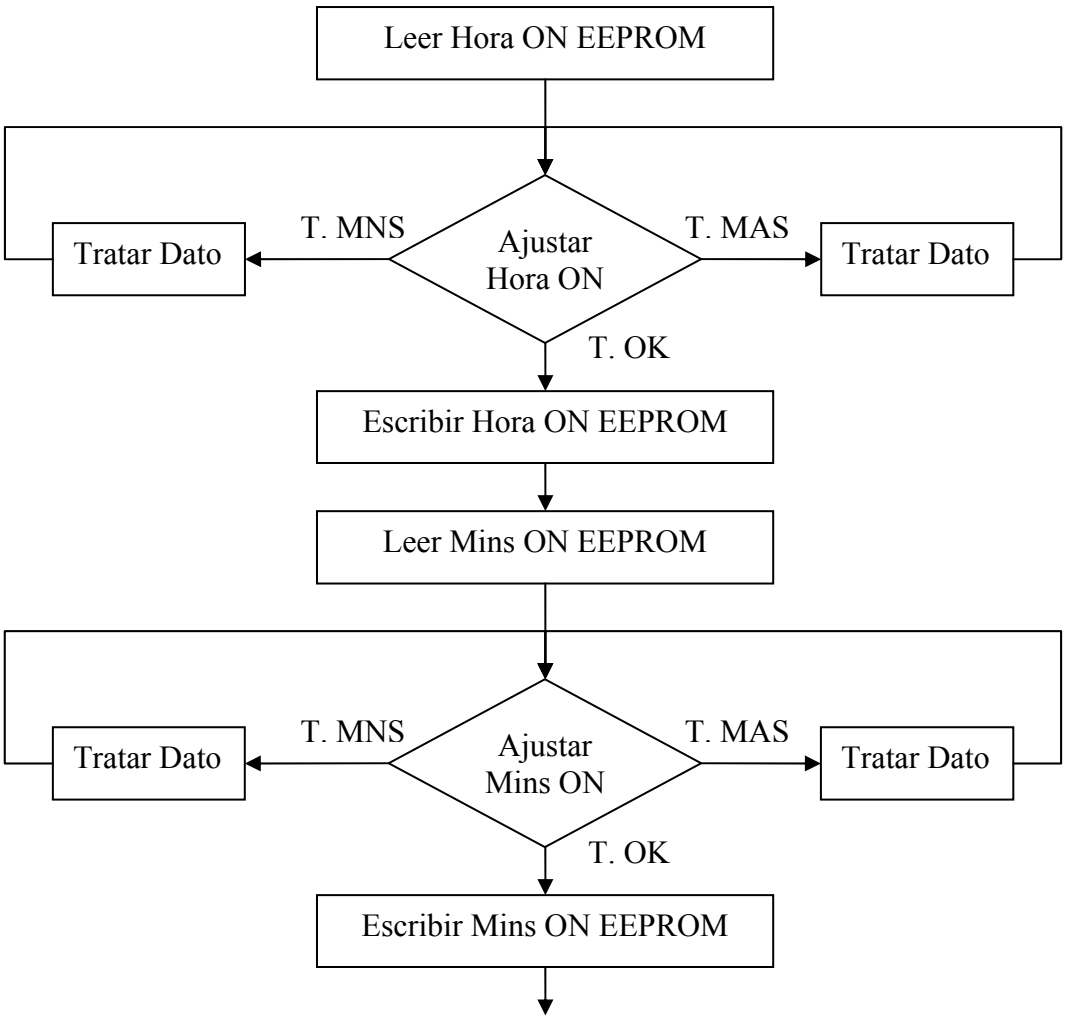
Cabe destacar que si no se detecta ninguna pulsación durante 5 segundos, se salta al ajuste de la siguiente variable, esto se hace para evitar que el sistema quede bloqueado en el caso de que el usuario no termine de introducir los datos.

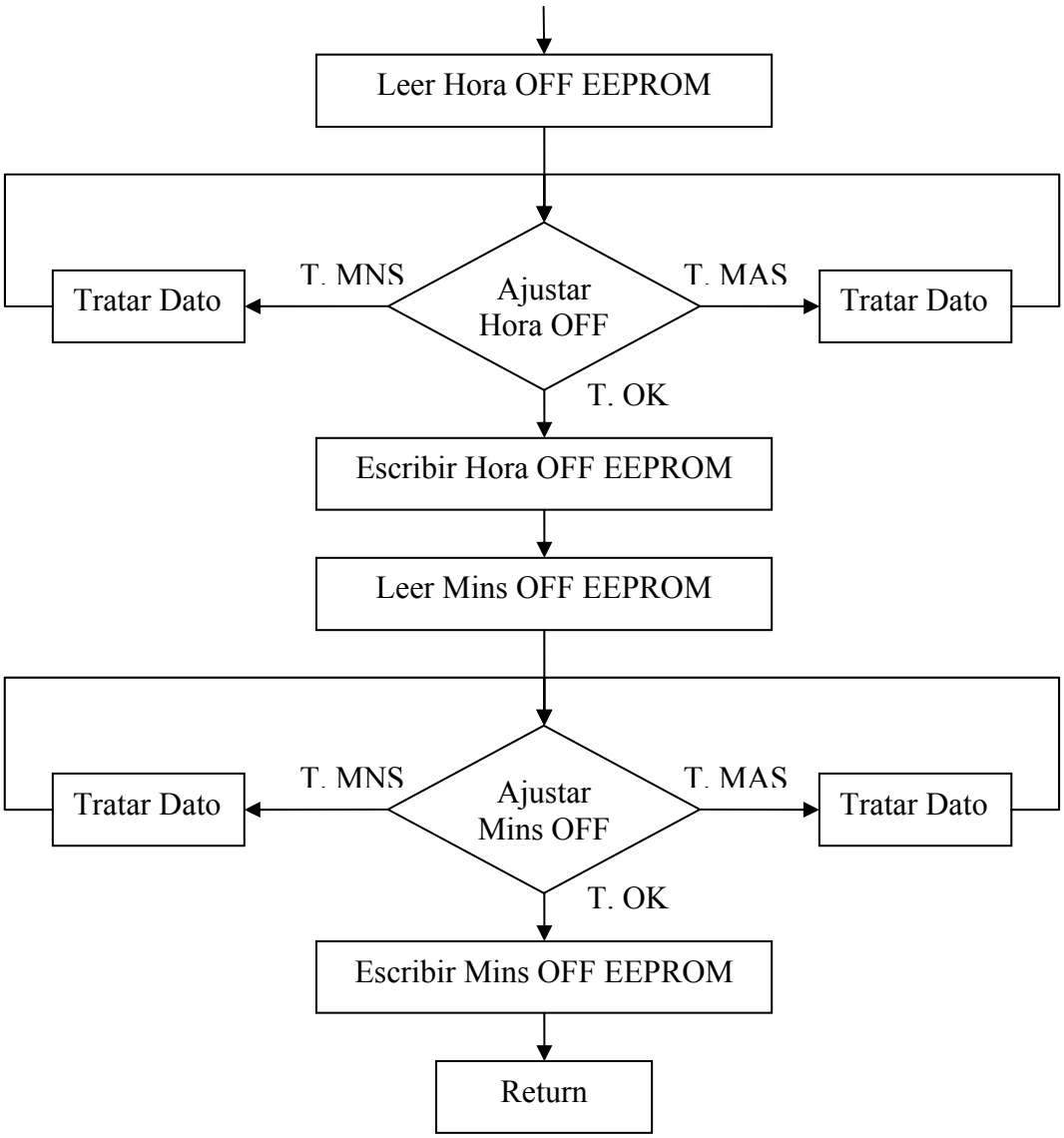
Una vez realizado el ajuste de todas las variables, se vuelve al menú inicio para continuar con el ajuste inicial del sistema.

- AJUSTAR TEMPORIZADOR

El temporizador, que en realidad son dos temporizadores, se encarga de controlar el encendido y el apagado de las cargas asociadas a ellos. El temporizador 1 controla las horas de luz del terrario mientras que el temporizador 2 controla las horas de funcionamiento de la bomba de agua.

Para realizar el ajuste de cada uno de ellos, se sigue el siguiente diagrama de flujo.



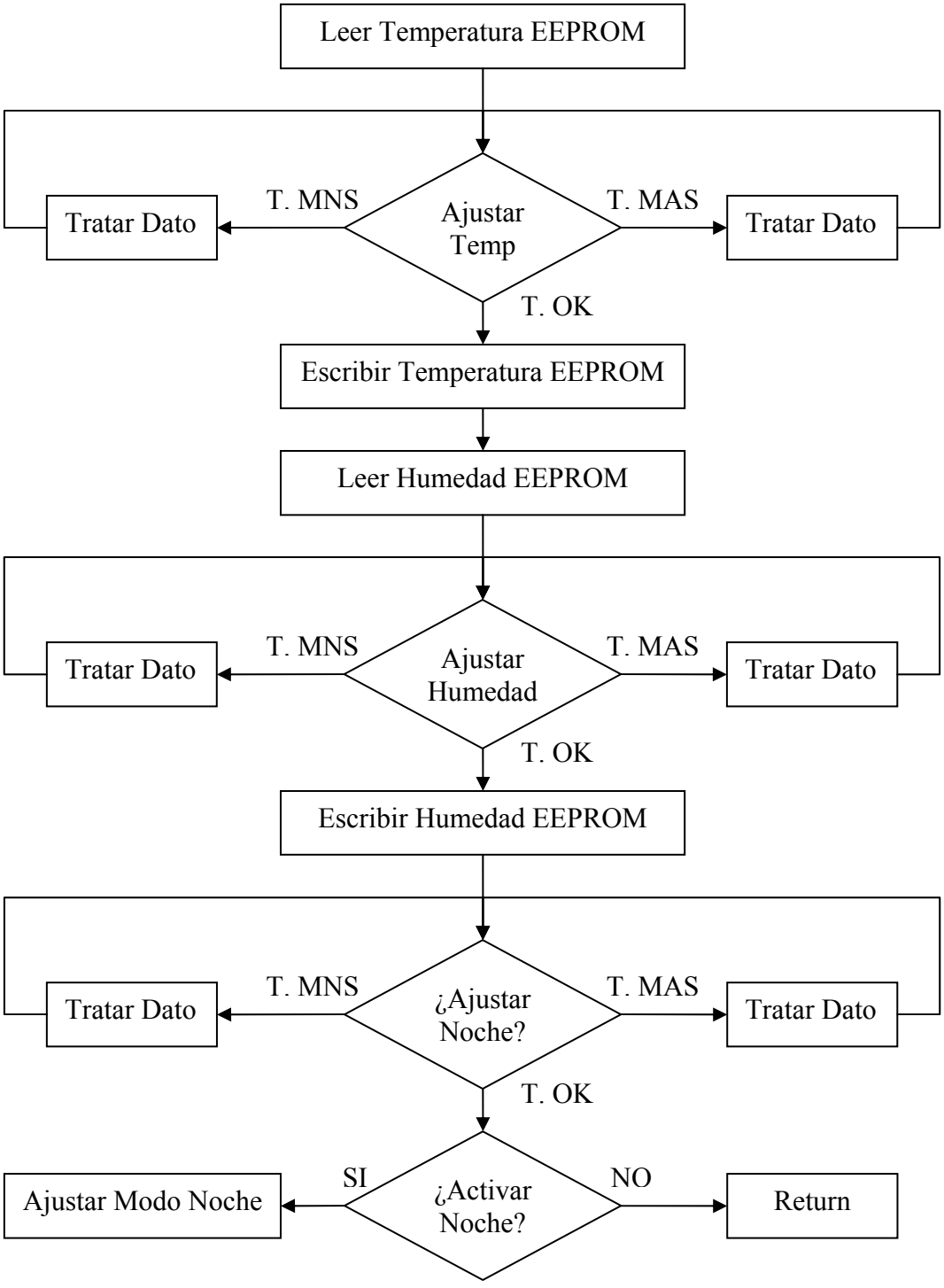


El funcionamiento es similar al caso anterior, mediante un mensaje en el display se pide la introducción de las variables. Previamente a la modificación de las variables se lee su valor desde la EEPROM para modificarlo a partir de él, de esta forma se evitan posibles errores por parte del usuario.

Una vez establecido el valor deseado de hora de encendido/apagado y minutos de encendido/apagado se almacenan en la EEPROM del PIC, de esta forma su valor puede ser recuperado posteriormente por el algoritmo del temporizador, el cual es detallado mas adelante. Además en caso de desconexión los valores son restaurados automáticamente evitando una reprogramación por parte del usuario.

- Ajustar Temperatura y Humedad

El ajuste de la temperatura y la humedad se va a llevar a cabo siguiendo un proceso semejante al indicado anteriormente.

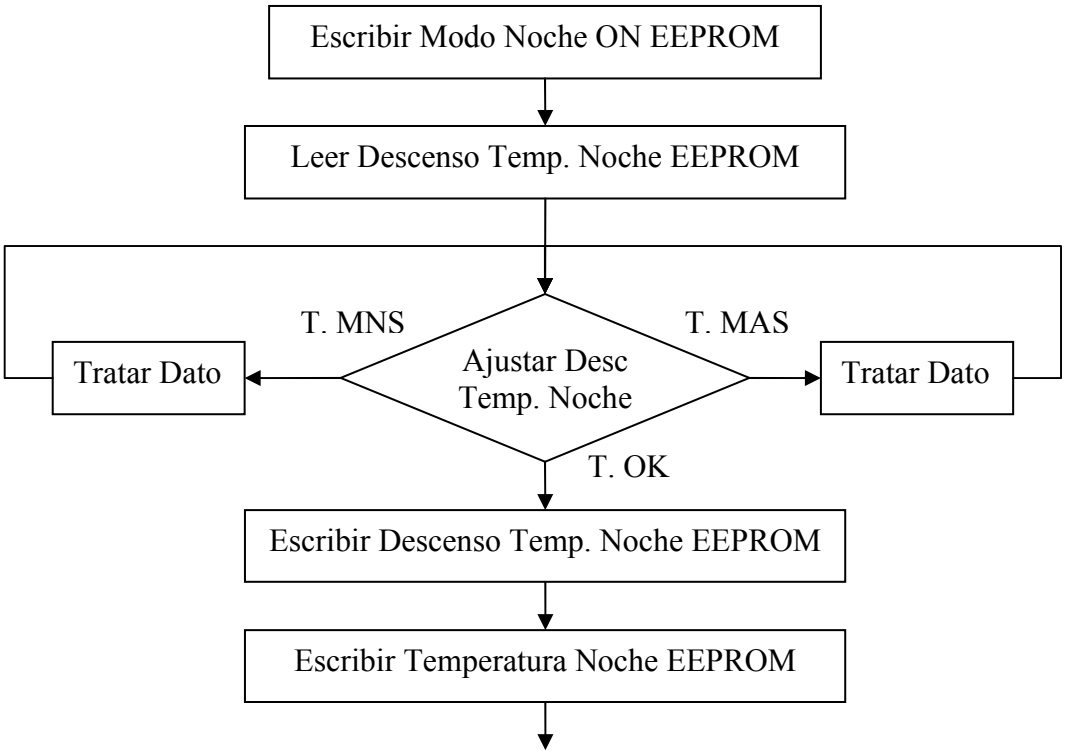


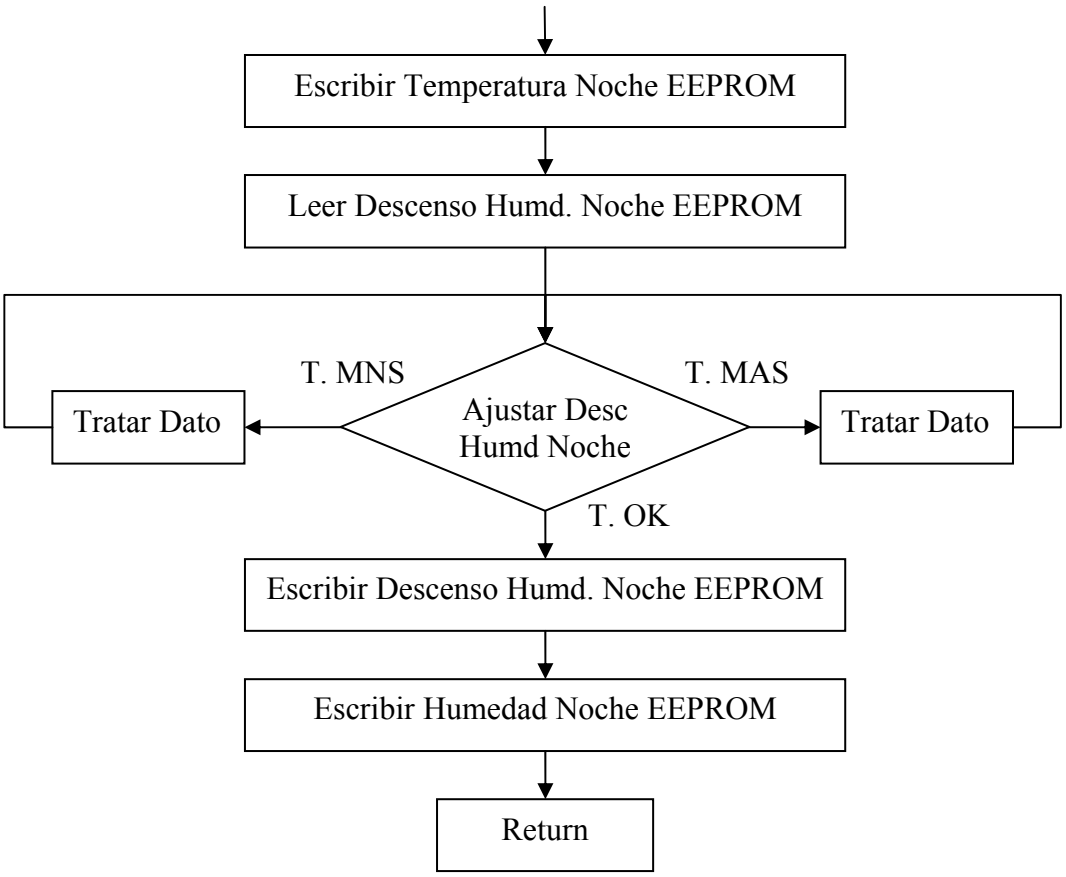
Como puede observarse en el organigrama, se lee el valor de temperatura/humedad desde la EEPROM antes de proceder a su modificación, de esta forma no se establece

ningún valor inicial por defecto que pueda llevar a una incorrecta y peligrosa configuración del sistema por parte del usuario.

Los tratamientos de datos se encargan de que las variables para temperatura y humedad estén establecidas entre unos rangos determinados y seguros de funcionamiento.

Una vez establecidos los valores óptimos de temperatura y humedad, se pide al usuario determinar la activación o no, del modo noche. En el caso de la activación de este modo, en el momento que sea desconectada la iluminación del terrario, se procede a realizar el decremento de la temperatura y la humedad respecto a los valores indicados. Esto se realiza con el fin de imitar el descenso de temperatura y humedad en la naturaleza durante la noche.





El Modo Noche se encarga, como se ha descrito anteriormente, de ajustar el descenso de temperatura y humedad llegada la noche. Para ello se utiliza un diagrama de introducción de datos como los vistos hasta ahora. Los tratamientos de datos llevan incluidos las medidas de seguridad necesarias para que el usuario solo pueda realizar un descenso sin poner en riesgo la vida del animal. Para ello se establecen unos descensos máximos de temperatura y humedad, ajustándose los descensos nocturnos con respecto a ellos.

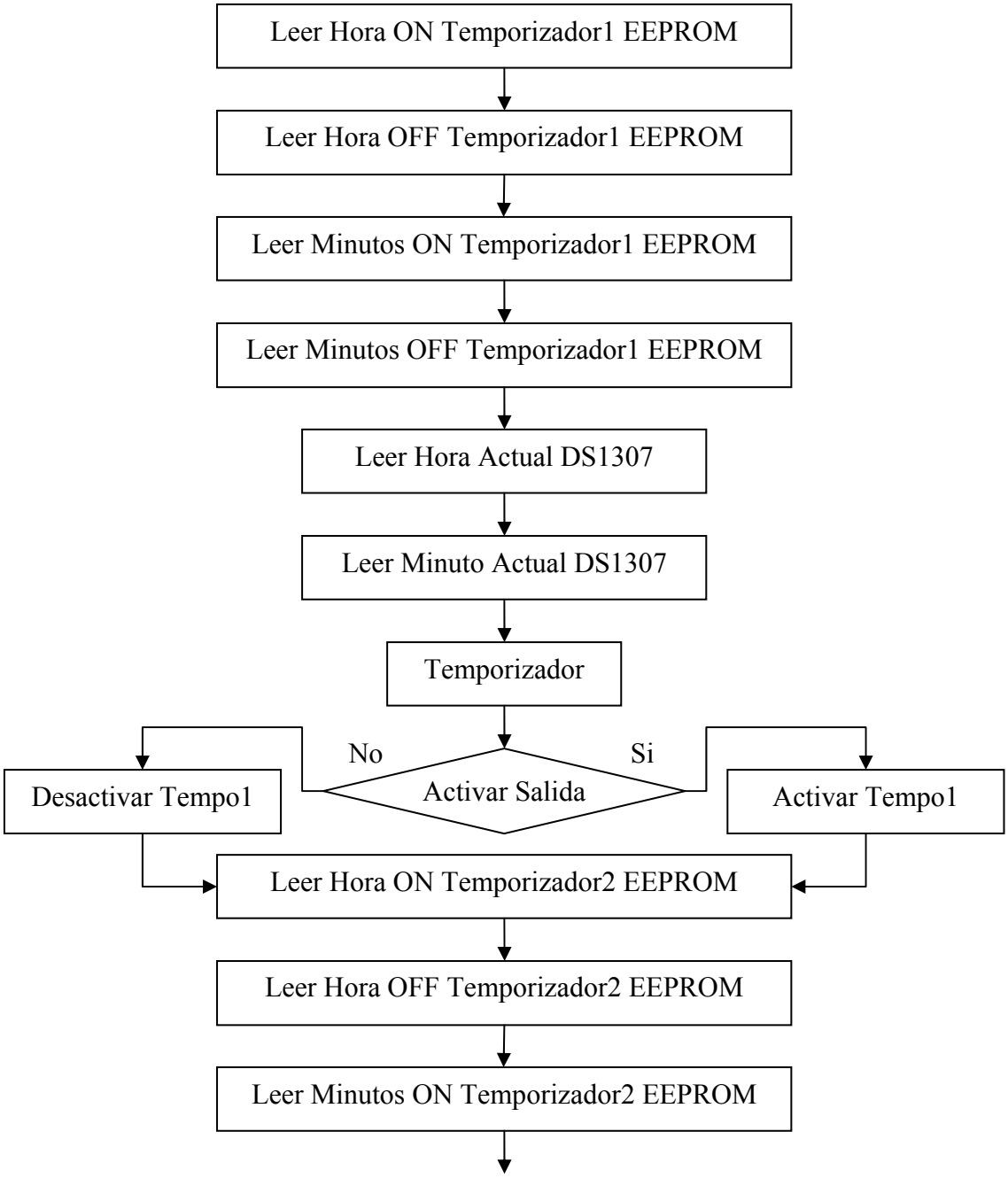
Una vez establecidas las variables de descenso de temperatura y humedad, se resta este de la temperatura diurna para determinar la temperatura de funcionamiento nocturna. Estos valores son almacenados en la EEPROM para su posterior lectura en el algoritmo del termostato e higrostato. También es almacenado en la EEPROM la activación o no del modo noche, de esta forma el sistema es capaz de identificar con que valores de temperatura o humedad actuar en cada caso.

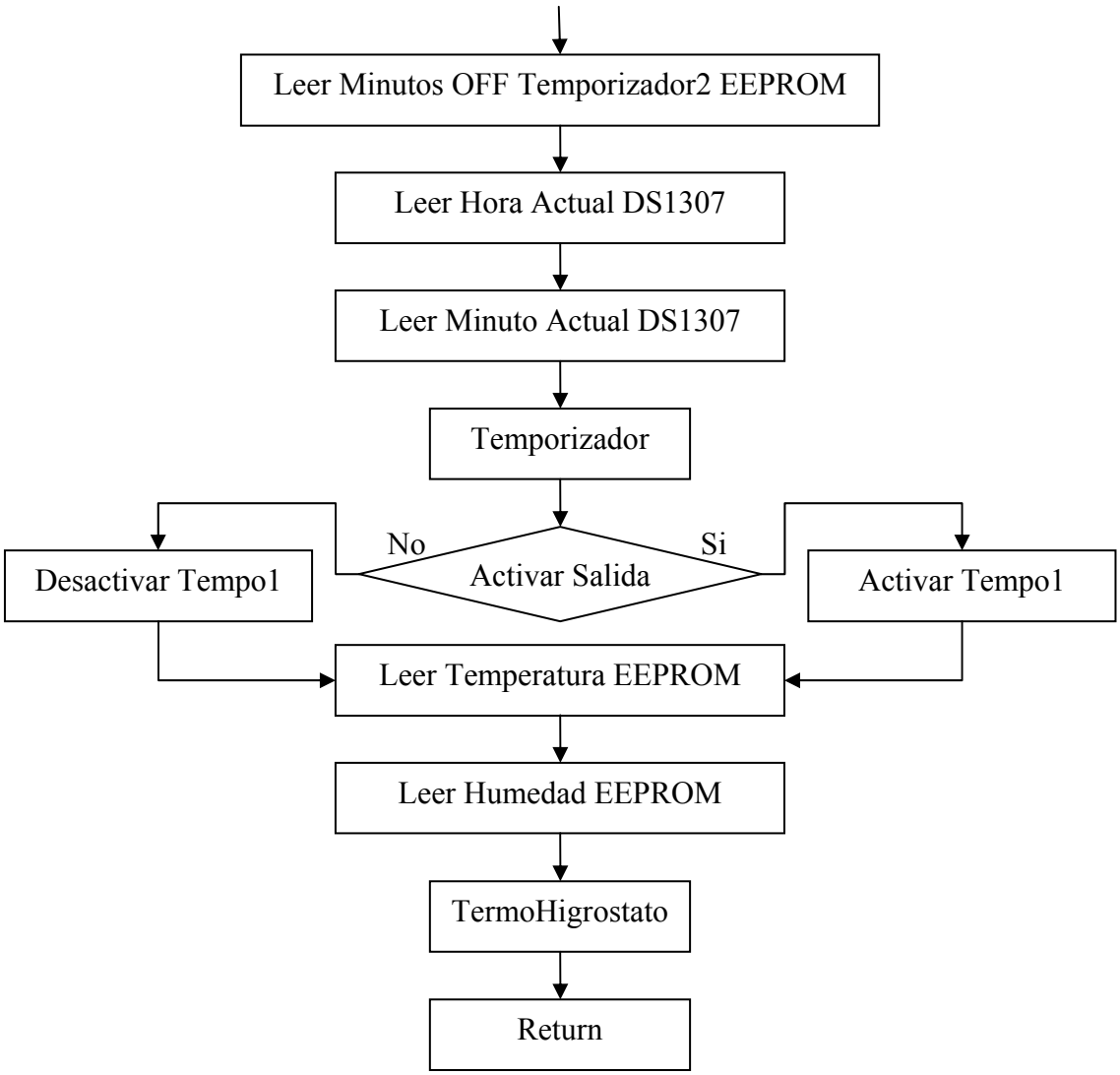
En el caso de la no activación del modo noche, o una vez ajustadas las variables de descenso, se concluye con la configuración del Menú Inicio.

- ACTUALIZAR SALIDAS

Una vez configurado el sistema por primera vez o iniciado después de un corte en la alimentación, se entra en un bucle infinito en el que se ejecutan dos funciones, Actualizar Salidas y Actualizar Pantalla. Solo se va a salir de este bucle cuando se provoque una interrupción, que son descritas mas adelante.

A continuación se muestra el diagrama de flujo de la función Actualizar Salidas.


Leer Hora ON Temporizador1 EEPROM
Leer Hora OFF Temporizador1 EEPROM
Leer Minutos ON Temporizador1 EEPROM
Leer Minutos OFF Temporizador1 EEPROM
Leer Hora Actual DS1307
Leer Minuto Actual DS1307
Temporizador
No
Si
Activar Salida
Desactivar Tempo1
Activar Tempo1
Leer Hora ON Temporizador2 EEPROM
Leer Hora OFF Temporizador2 EEPROM
Leer Minutos ON Temporizador2 EEPROM



Esta función se encarga, como puede apreciarse en el organigrama, de realizar una lectura de las variables, parámetros preestablecidos del sistema y llamar a cada una de las funciones que contienen los algoritmos para el control del sistema, en este caso, temporizador y termohigrostato.

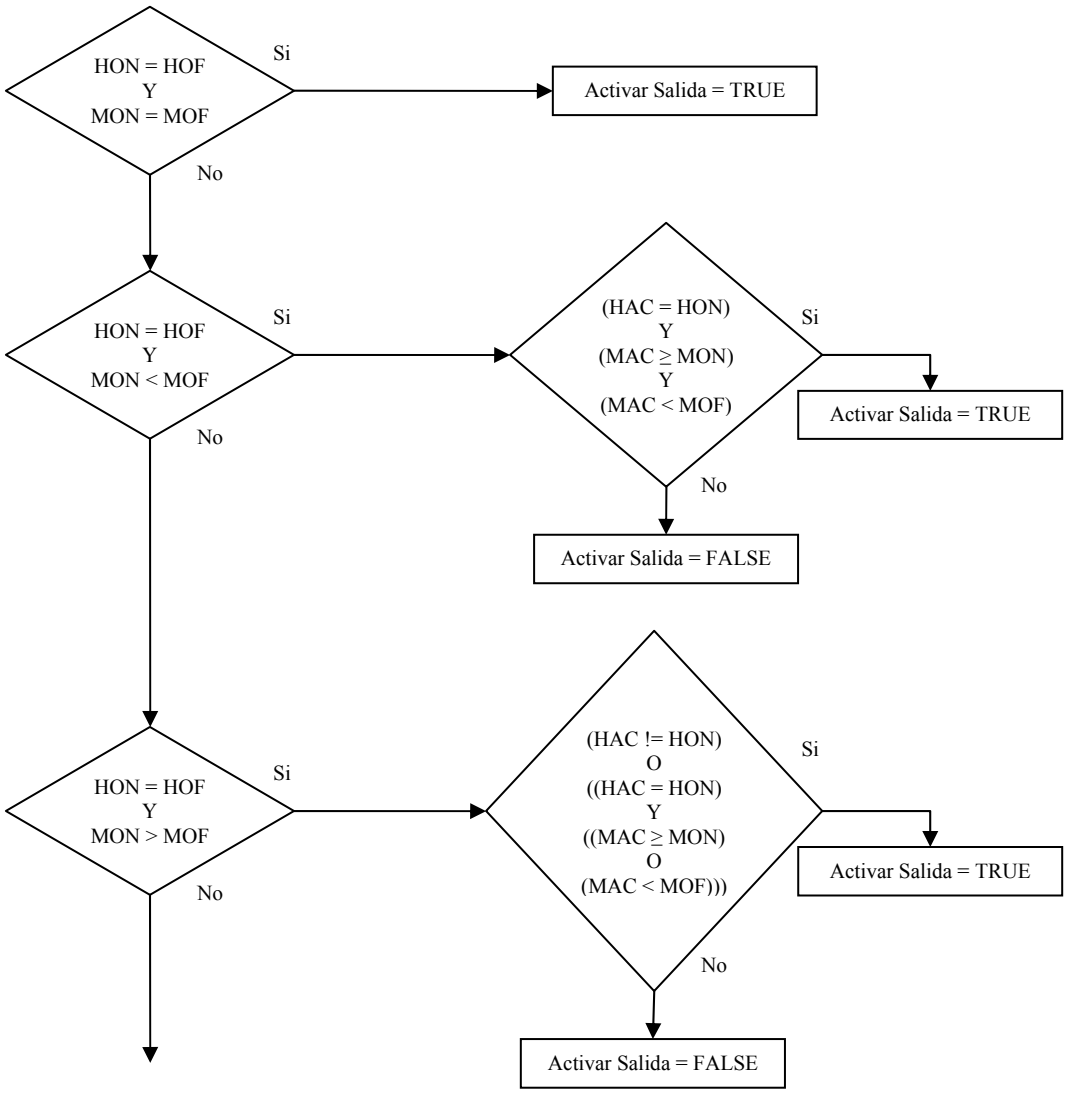
Para empezar se leen los valores establecidos en la configuración del sistema para Hora de encendido, Hora de apagado, Minuto de encendido, Minuto de apagado, los cuales se encuentran en la EEPROM. A continuación se lee del reloj de tiempo real la hora y minuto actual. Con estos valores almacenados en las respectivas variables, se llama a la función temporizador para que opere con ellos. Esto se hace tanto para el primer temporizador, como para el segundo, aunque para este último como es lógico se operará con sus valores correspondientes. Después de la llamada a la función temporizador, se activa/desactiva la salida según corresponda, y se actualiza la

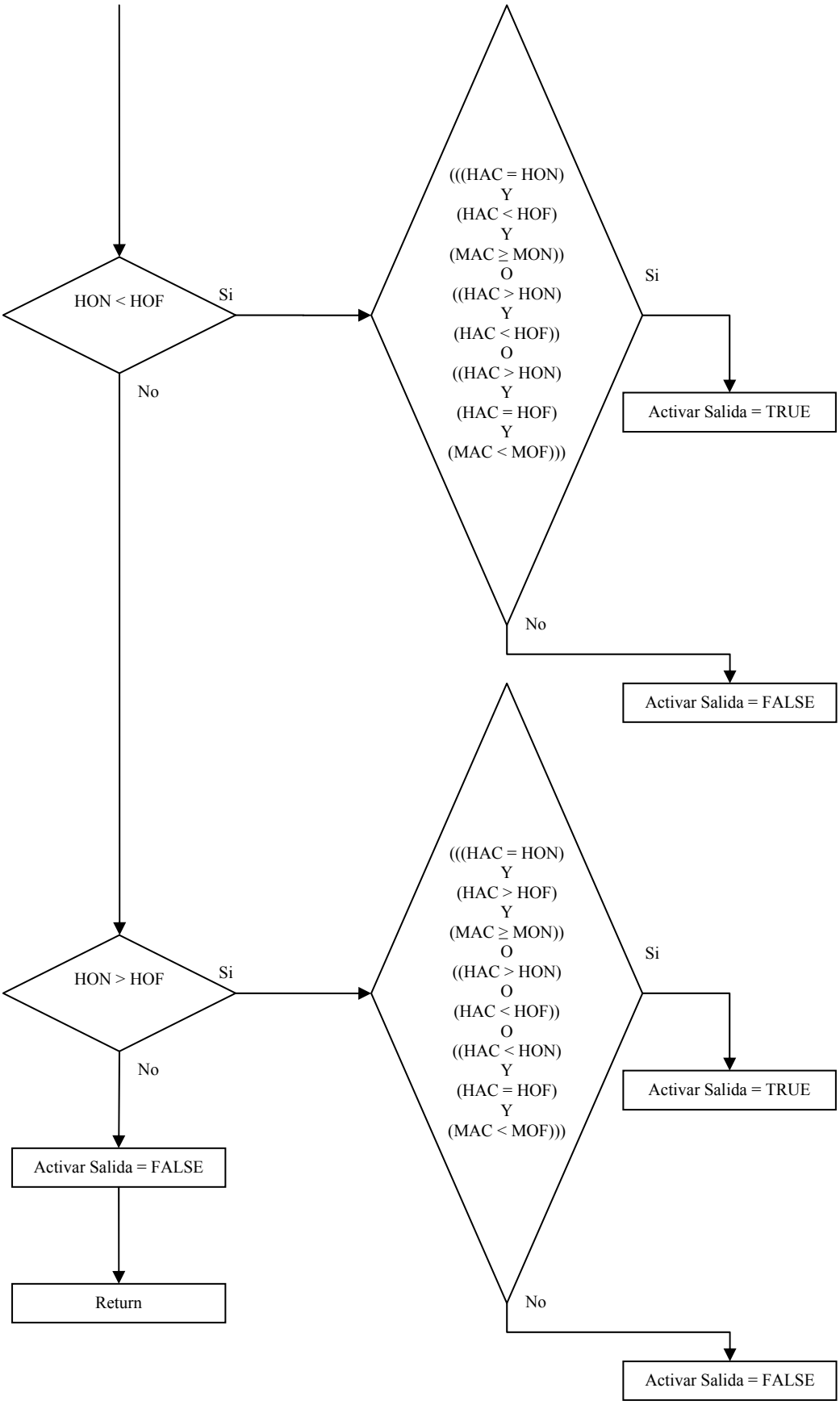
información en el LCD, mostrando este, el estado de conexión/desconexión de la carga asociada a cada uno de los temporizadores.

Una vez se ha comprobado el estado de los temporizadores, se leen los valores de temperatura y humedad fijados por el usuario de la EEPROM, y se llama a la función termohigrostato, la cual contiene el algoritmo que se encarga de activar/desactivar las cargas para el establecimiento de la temperatura y la humedad en los valores indicados.

A continuación se explica el funcionamiento de cada uno de los algoritmos principales del sistema. Siendo estos, Temporizador y TermoHigrostato.

- TEMPORIZADOR




HON < HOF
Si
No
(((HAC = HON)
Y
(HAC < HOF)
Y
(MAC ≥ MON))
O
((HAC > HON)
Y
(HAC < HOF))
O
((HAC > HON)
Y
(HAC = HOF)
Y
(MAC < MOF)))
Si
No
Activar Salida = TRUE
Activar Salida = FALSE
HON > HOF
Si
No
(((HAC = HON)
Y
(HAC > HOF)
Y
(MAC ≥ MON))
O
((HAC > HON)
O
(HAC < HOF))
O
((HAC < HON)
Y
(HAC = HOF)
Y
(MAC < MOF)))
Si
No
Activar Salida = TRUE
Activar Salida = FALSE
Return
Activar Salida = FALSE

El organigrama anterior muestra el funcionamiento del algoritmo del temporizador. El algoritmo del temporizador se encarga de determinar si la hora y minuto actual pertenece al periodo de tiempo en el cual debe estar activa o no la carga asociada a él.

Para ello se compara la hora de inicio con la hora de apagado, y los minutos de encendido y apagado en el caso de que la hora de inicio y apagado sean iguales, de esta forma se obtiene un entorno de funcionamiento para el sistema, basta entonces con realizar varias comparaciones para determinar en que entorno se está operando, y se hace uso de la hora y minuto actual para determinar si la salida debe activarse o no. A continuación se expone una lista de las abreviaturas usadas:

- HAC: Hora Actual
- HON: Hora de Encendido
- HOF: Hora de Apagado
- MAC: Minuto Actual
- MON: Minuto de Encendido
- MOF: Minuto de Apagado

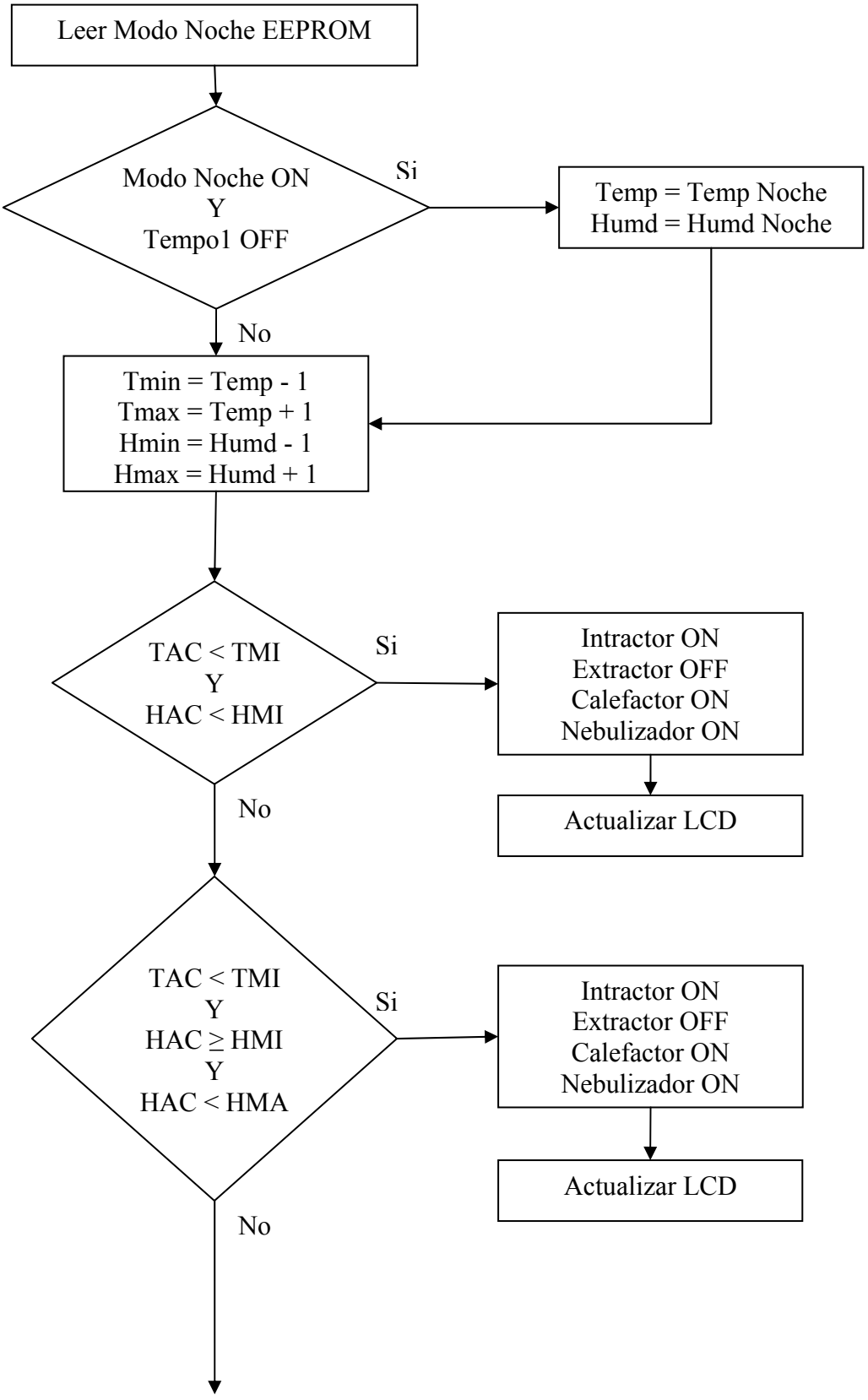
Esta función de temporizador no opera directamente con las salidas del sistema, por ello es necesario usar una variable tipo bandera, que es comprobada en la función Actualizar Salidas, tal y como ha sido descrito anteriormente.

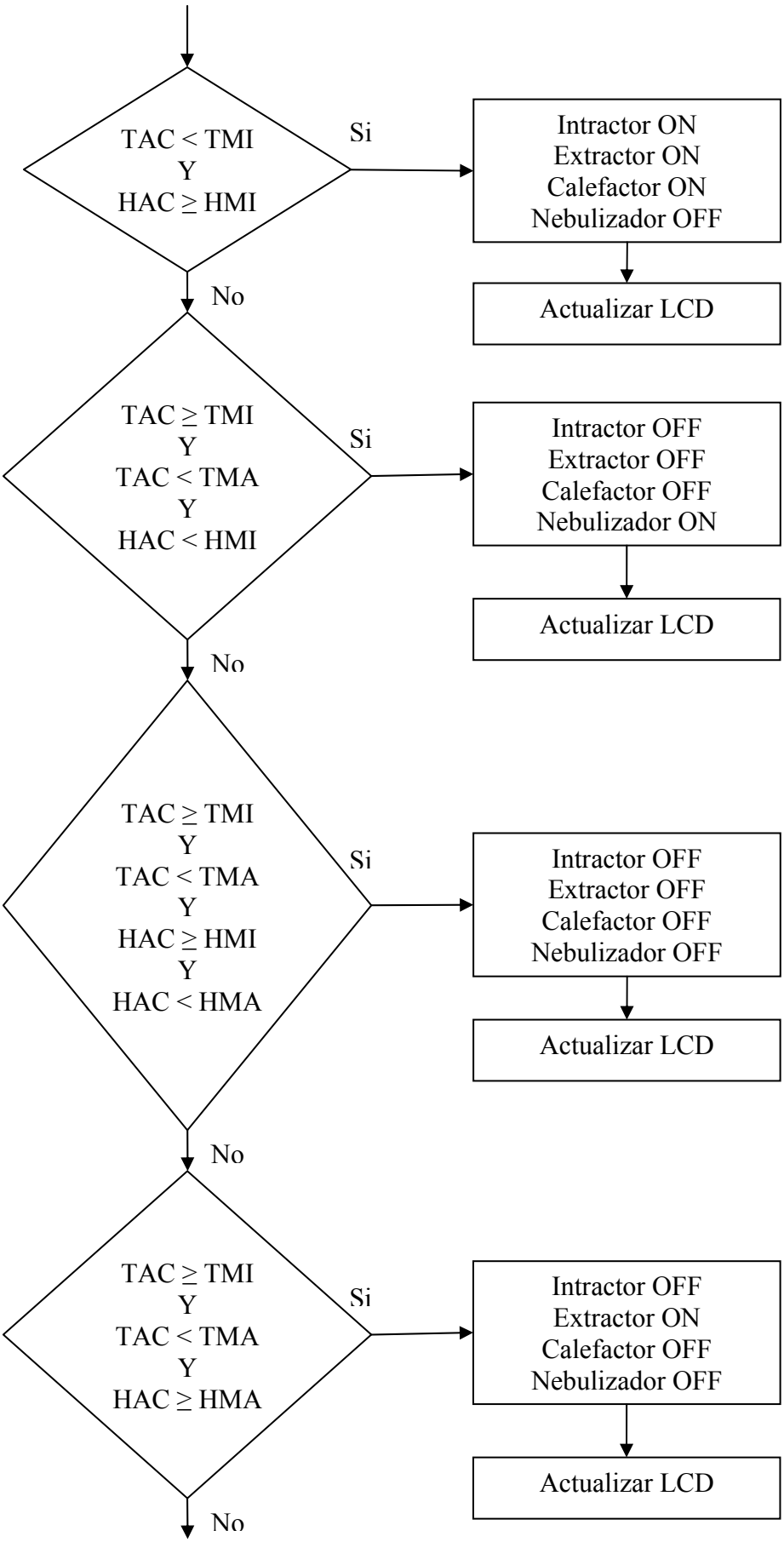
- TERMOHIGROSTATO

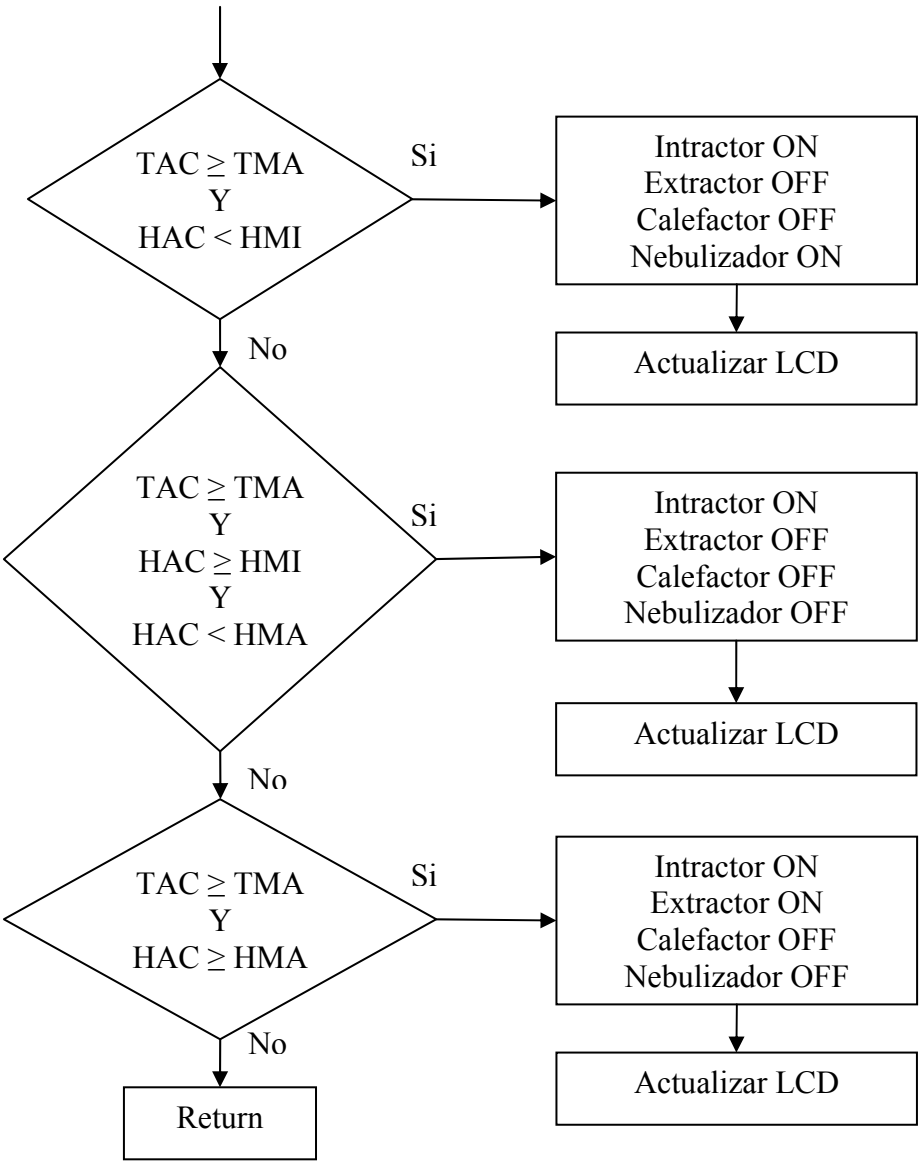
El otro algoritmo principal del sistema, es el que contiene la función termohigrostato, en este algoritmo se comprueba que la temperatura actual no sea superior o inferior a la preestablecida por el usuario, lo mismo ocurre con la humedad.

Para ello, se opera entre unos rangos de funcionamiento máximos y mínimos de ±1º C y ±1% HR. Una vez determinado en el rango de operación, donde se encuentra el sistema, se activan/desactivan las salidas para el control de ventiladores, uno de ellos infractor y el otro extractor, además del calefactor y el humidificador.

Se muestra a continuación en organigrama del algoritmo del termohigrostato.




TAC < TMI
Y
HAC ≥ HMI
Si
Intractor ON
Extractor ON
Calefactor ON
Nebulizador OFF
Actualizar LCD
No
TAC ≥ TMI
Y
TAC < TMA
Y
HAC < HMI
Si
Intractor OFF
Extractor OFF
Calefactor OFF
Nebulizador ON
Actualizar LCD
No
TAC ≥ TMI
Y
TAC < TMA
Y
HAC ≥ HMI
Y
HAC < HMA
Si
Intractor OFF
Extractor OFF
Calefactor OFF
Nebulizador OFF
Actualizar LCD
No
TAC ≥ TMI
Y
TAC < TMA
Y
HAC ≥ HMA
Si
Intractor OFF
Extractor ON
Calefactor OFF
Nebulizador OFF
Actualizar LCD
No



Se detalla a continuación la nomenclatura utilizada.

- TAC: Temperatura Actual
- TMA: Temperatura Máxima
- TMI: Temperatura Mínima
- HAC: Humedad Actual
- HMA: Humedad Máxima
- MHI: Humedad Mínima

Con esta función, se termina la descripción de la función Actualizar Salidas perteneciente al bucle infinito de la función principal del sistema. La otra función perteneciente a este bucle es Actualizar Pantalla, que simplemente envía al LCD la

fecha y la hora, el resto de información visualizada en el LCD ya ha sido añadida mediante las funciones anteriormente descritas. Se muestra a continuación una imagen del LCD mientras está operando en el modo normal, es decir, dentro del bucle principal, y sin que se produzca ninguna interrupción externa.

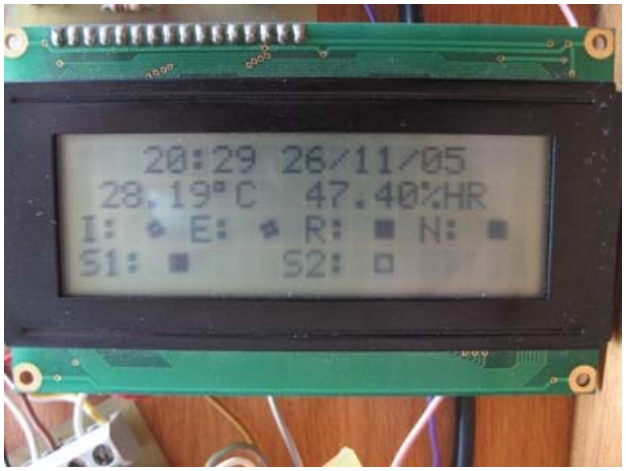


Fig. 4.1 Pantalla de estado

En la figura 4.1 se puede apreciar la fecha y la hora en la primera línea del LCD, la temperatura y la humedad relativa en la segunda línea, el estado de los dispositivos encargados de controlar la temperatura y la humedad, es decir, el ventilador intractor, que está parado, el ventilador extractor que también está parado, la resistencia calefactora que se encuentra deshabilitada y el nebulizador, que se encuentra activo. En la cuarta línea del LCD se muestra el estado de las cargas que controlan los temporizadores, estando ambas desactivadas.

- INTERRUPCIONES

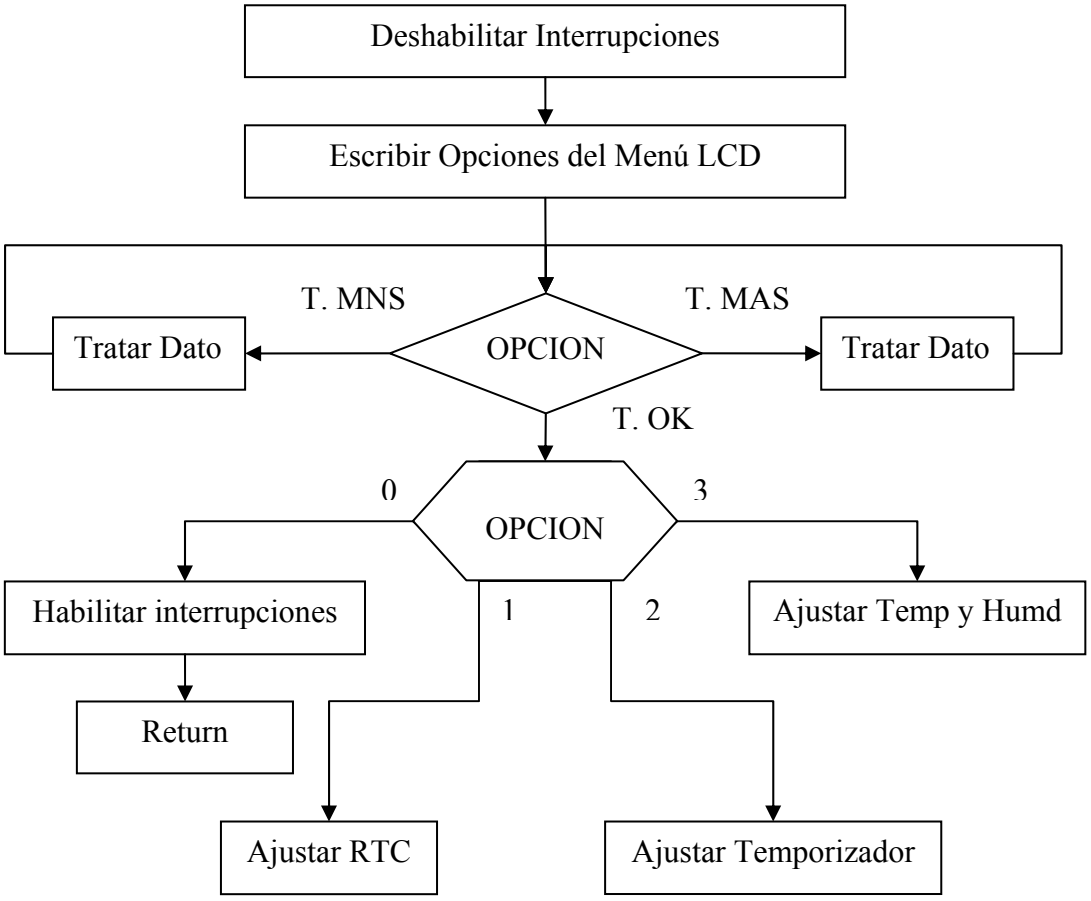
En este apartado se va a realizar una descripción de las interrupciones utilizadas en el sistema, puede hacerse una distinción entre interrupciones externas e internas.

Las externas están vinculadas a los pulsadores T.MAS, T.MNS, T.OK. Cuando es presionada alguna de ellas, el microcontrolador deja el proceso que esté realizando, y se trata a cada interrupción acorde a la función establecida para cada una de ellas.

Por otro lado se hace uso de una interrupción externa, en concreto la del Timer0. En este caso se produce una interrupción al desbordarse este contador, el cual está programado para que se realice cada segundo. La función de esta interrupción es realizar la lectura del RTC cada segundo, y cada 20 segundos, realizar una medición del sensor SHT11.

- INTERRUPCION T.OK

Este pulsador, en el modo de funcionamiento normal, es el encargado de que se entre en el Menú de Configuración del sistema, para ello al detectar la pulsación, se genera una interrupción en la que se sigue el siguiente diagrama de flujo.



Como puede observarse en la figura 4.2, al entrar en la interrupción se muestra un menú en el que se dan cuatro opciones posibles, las tres opciones de ajuste ya han sido estudiadas con anterioridad, la cuarta opción se da para salir del menú seleccionando el número 0. En el caso de que ningún pulsador se

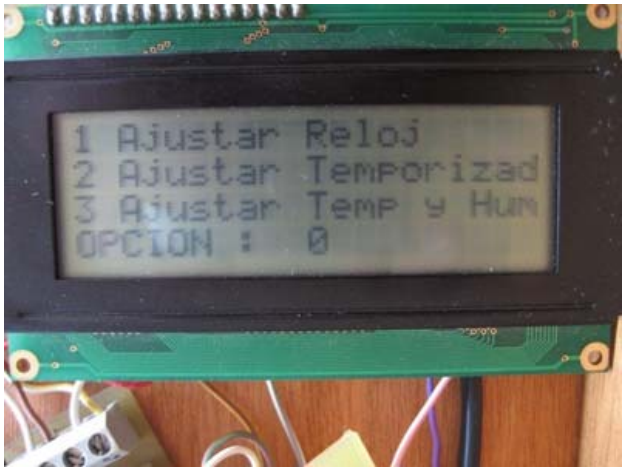
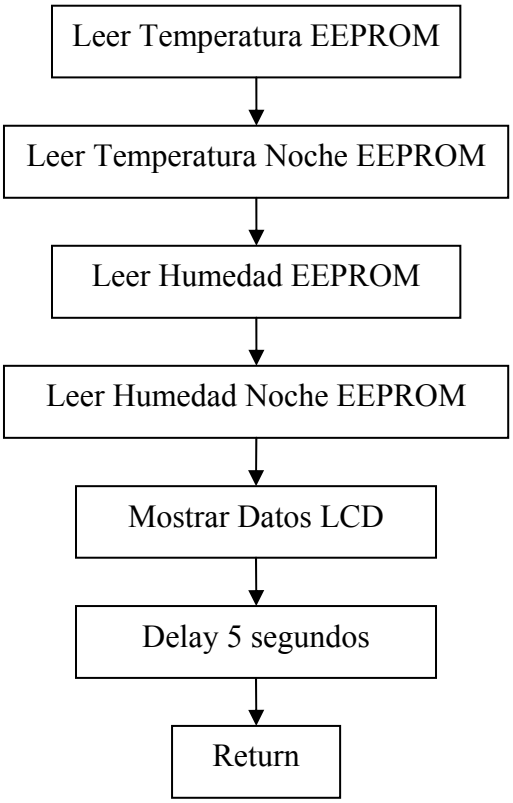


Fig. 4.2 Pantalla de menú

active en cinco segundos, el sistema automáticamente sale del menú y pasa a su modo de funcionamiento normal.

- INTERRUPCION T.MAS

Al producirse una interrupción debida a la pulsación de esta tecla, se muestra en el LCD información sobre la temperatura y la humedad definida por el usuario. Para ello se sigue el siguiente esquema.

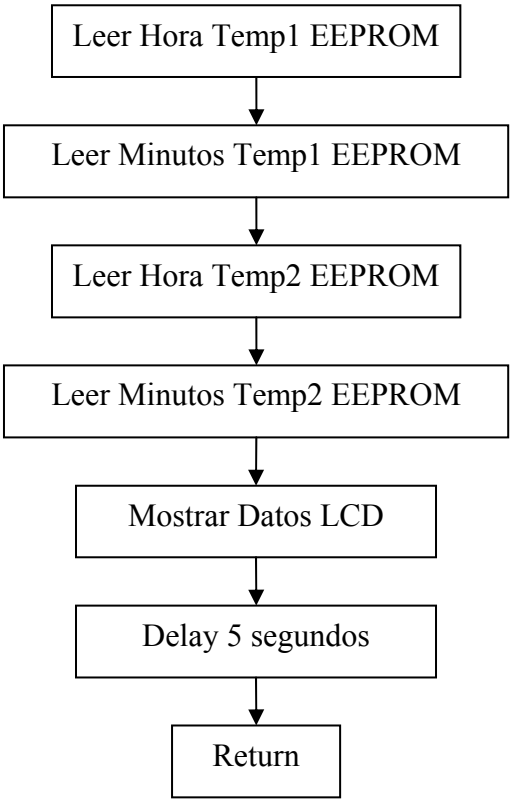


Puede observarse en el diagrama la forma en que opera esta interrupción, adicionalmente se muestra en la figura 4.3 los datos representados en el LCD al producirse la interrupción estando en el modo normal de funcionamiento.

Fig. 4.3 Pantalla de información 1

- INTERRUPCION T.MNS

Esta interrupción opera de forma análoga a la anterior, excepto que en este caso se muestra la información sobre los temporizadores, y se entra en ella mediante la pulsación de T.MNS



En la figura 4.4 se muestra la información mostrada en el LCD como consecuencia de la interrupción causada por la presión del pulsador T.MNS

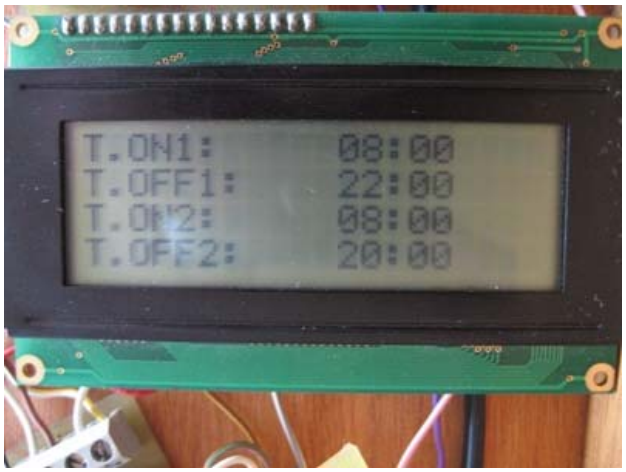
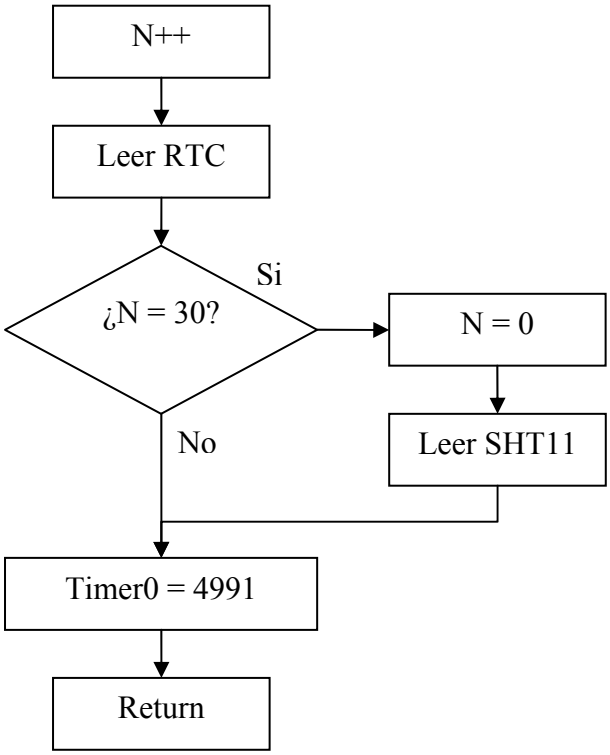


Fig. 4.4 Pantalla de información 2

- INTERRUPCION TIMER 0

Otra causa de interrupción es la ocasionada por el desbordamiento del Timer0, el cual está programado para que se realice cada segundo. Cuando se produzca la interrupción se entra en la función del tratamiento de esta, donde es leída la hora y actualizado un contador, cuando este contador llegue a 30, es decir, hayan transcurrido 30 segundos, se leerá el sensor SHT11.



El valor cargado al Timer0 se deduce de la siguiente fórmula:

Temp=4·Tosc·(2^(tamaño_registro_TMR0)-Valor_cargado_TMR0)·preescala_TMR0
1 sg=4·(1/4Mhz)·(2^16-valor)·64 → valor = 49911

A partir de ese valor, el contador ascendente del Timer0 aumenta en cada ciclo de reloj hasta que se desborda y provoca una interrupción, con ello se consigue una temporización exacta de un segundo, que es usado para generar lecturas para tiempos mayores.

4.2 Código Fuente

- TERRARIO.C

```
#include <18F2520.h>

#fuses NOWDT, WDT128, XT, NOPROTECT, BROWNOUT, BORV45, NOPUT, NOCPD
#fuses NOSTVREN, NODEBUG, NOLVP, NOWRT, NOWRTD, NOIESO, NOFCMEN, NOPBADEN
#fuses NOWRTC, NOWRTB, NOEBTR, NOEBTRB, NOCPB, NOLPT1OSC, MCLR, NOXINST

#use delay(clock=4000000)
#use i2c(Master, SDA=PIN_A0, SCL=PIN_A1)

#byte pb       =0xF81
#bit t_ok      =pb.0
#bit t_mas     =pb.1
#bit t_mns     =pb.2

#byte intcon   =0xFF2
#byte intcon3  =0xFF0
#bit  intf     =intcon.1
#bit  intf1    =intcon3.0
#bit  intf2    =intcon3.1

#define intractor    PIN_B4
#define extractor    PIN_B5
#define resistencia  PIN_B6
#define nebulizador  PIN_B7
#define tempoluz     PIN_A2
#define tempoagua    PIN_A3
#define ON           0
#define OFF          1

#include <lcd.h>
#include <ds1307.h>
#include <sht11.h>
#include <menu.h>

void actualizar_salidas(void);
void actualizar_pantalla(void);
void temporizador(void);
void termohigrostato(void);
void lectura_sht11(void);
void lectura_rtc(void);
void lcd_carga_caracteres(void);

unsigned int8 hora_on, hora_off, mins_on, mins_off, hora_actual, mins_actual;
unsigned int8 modo_noche, activar_salida, temp_actual, humedad_actual;
```

```
unsigned int8 tmin, tmax, hmin, hmax, n;
typedef union
{
   int16 i;
   float f;
} valor;
valor humedad, temperatura;
byte errorsht11,checksum;

void main()
{
   port_b_pullups(true);
   setup_adc_ports(NO_ANALOGS);

   lcd_init();
   lcd_carga_caracteres();
   sht11_hard_reset();
   menu_modo();

   intf=0; intf1=0; intf2=0;
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_64);
   set_timer0(49911);

   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
   enable_interrupts(INT_TIMER0);
   enable_interrupts(GLOBAL);

   while(true)
   {
      actualizar_salidas();
      actualizar_pantalla();
   }
}


void lectura_sht11(void)
{
   leer_ds1307();

   errorsht11=0;
      errorsht11+=sht11_medicion((byte*) &humedad.i, &checksum, HUMI);
      errorsht11+=sht11_medicion((byte*) &temperatura.i, &checksum, TEMP);
      if(errorsht11!=0)
   {
      printf(lcd_putc,"\n\rerror:%U", errorsht11);
            sht11_hard_reset();
      }
      else
   {
```

```
      humedad.f=(float)humedad.i;
         temperatura.f=(float)temperatura.i;
         sht11_calculos(&humedad.f, &temperatura.f);
      temp_actual = temperatura.f;
      humedad_actual = humedad.f;
      }
}


void lectura_rtc(void)
{
   leer_ds1307();
}


void actualizar_salidas(void)
{
   hora_on=read_eeprom(e_hora_on_1);
   hora_off=read_eeprom(e_hora_off_1);
   mins_on=read_eeprom(e_mins_on_1);
   mins_off=read_eeprom(e_mins_off_1);
   hora_actual=registros_ds1307[horas];
   mins_actual=registros_ds1307[minutos];
   temporizador();
   lcd_gotoxy(1,4);
   printf(lcd_putc,"S1: ");
   if (activar_salida == TRUE){lcd_send_byte(1,2); output_bit(tempoluz,ON);}
   else if (activar_salida == FALSE){lcd_send_byte(1,3); output_bit(tempoluz,OFF);}

   hora_on=read_eeprom(e_hora_on_2);
   hora_off=read_eeprom(e_hora_off_2);
   mins_on=read_eeprom(e_mins_on_2);
   mins_off=read_eeprom(e_mins_off_2);
   hora_actual=registros_ds1307[horas];
   mins_actual=registros_ds1307[minutos];
   temporizador();
   lcd_gotoxy(10,4);
   printf(lcd_putc,"S2: ");
   if (activar_salida == TRUE){lcd_send_byte(1,2); output_bit(tempoagua,ON);}
   else if (activar_salida == FALSE){lcd_send_byte(1,3); output_bit(tempoagua,OFF);}
   temp_=read_eeprom(e_temp);
   humd_=read_eeprom(e_humd);
   termohigrostato();
}


void temporizador(void)
{
   if ((hora_on == hora_off) && (mins_on == mins_off)) {activar_salida = TRUE; break;}
   else if ((hora_on == hora_off) && (mins_on < mins_off))
      {
```

```
         if ((hora_actual == hora_on) && (mins_actual >= mins_on) && (mins_actual <
mins_off)) {activar_salida = TRUE; break;}
         else {activar_salida = FALSE; break;}
      }
   else if ((hora_on == hora_off) && (mins_on > mins_off))
      {
         if ((hora_actual != hora_on) || ((hora_actual == hora_on) && ((mins_actual >=
mins_on) || (mins_actual < mins_off)))) {activar_salida = TRUE; break;}
         else {activar_salida = FALSE; break;}
      }
   else if (hora_on < hora_off)
      {
         if (((hora_actual == hora_on) && (hora_actual < hora_off) && (mins_actual >=
mins_on)) || ((hora_actual > hora_on) && (hora_actual < hora_off)) || ((hora_actual >
hora_on) && (hora_actual == hora_off) && (mins_actual < mins_off))) {activar_salida =
TRUE; break;}
         else {activar_salida = FALSE; break;}
      }
   else if (hora_on > hora_off)
      {
         if (((hora_actual == hora_on) && (hora_actual > hora_off) && (mins_actual >=
mins_on)) || ((hora_actual > hora_on) || (hora_actual < hora_off)) || ((hora_actual <
hora_on) && (hora_actual == hora_off) && (mins_actual < mins_off))) {activar_salida =
TRUE; break;}
         else {activar_salida = FALSE; break;}
      }
   else {activar_salida = FALSE; break;}
}


void termohigrostato(void)
{
   modo_noche=read_eeprom(e_modo_noche);
   if((modo_noche == TRUE) && (tempoluz == OFF))
   {
      temp_=read_eeprom(e_temp_noche);
      humd_=read_eeprom(e_humd_noche);
   }

   tmin=temp_-1;
   tmax=temp_+1;
   hmin=humd_-1;
   hmax=humd_+1;

   lcd_gotoxy(1,3);
   printf(lcd_putc,"I: ");
   lcd_gotoxy(6,3);
   printf(lcd_putc,"E: ");
   lcd_gotoxy(11,3);
   printf(lcd_putc,"R: ");
   lcd_gotoxy(16,3);
```

```
   printf(lcd_putc,"N: ");

   if ((temp_actual < tmin) && (humedad_actual < hmin))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,ON);
      output_bit(nebulizador,ON);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,2);
output_bit(tempoagua,ON);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,2);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,2);
   }
   else if ((temp_actual < tmin) && (humedad_actual >= hmin) && (humedad_actual < hmax))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,ON);
      output_bit(nebulizador,OFF);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,2);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
   else if ((temp_actual < tmin) && (humedad_actual >= hmax))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,TRUE);
      output_bit(resistencia,ON);
      output_bit(nebulizador,OFF);
```

```
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,2);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
   else if ((temp_actual >= tmin) && (temp_actual < tmax) && (humedad_actual < hmin))
   {
      output_bit(intractor,FALSE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,OFF);
      output_bit(nebulizador,ON);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,2);
output_bit(tempoagua,ON);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,2);
   }
   else if ((temp_actual >= tmin) && (temp_actual < tmax) && (humedad_actual >= hmin) &&
(humedad_actual < hmax))
   {
      output_bit(intractor,FALSE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,OFF);
      output_bit(nebulizador,OFF);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
```

```
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
   else if ((temp_actual >= tmin) && (temp_actual < tmax) && (humedad_actual >= hmax))
   {
      output_bit(intractor,FALSE);
      output_bit(extractor,TRUE);
      output_bit(resistencia,OFF);
      output_bit(nebulizador,OFF);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
   else if ((temp_actual >= tmax) && (humedad_actual < hmin))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,OFF);
      output_bit(nebulizador,ON);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,2);
output_bit(tempoagua,ON);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,2);
   }
   else  if  ((temp_actual  >=  tmax)  &&  (humedad_actual  >=  hmin)  &&  (humedad_actual  <
hmax))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,FALSE);
      output_bit(resistencia,OFF);
```

```
      output_bit(nebulizador,OFF);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
   else if ((temp_actual >= tmax) && (humedad_actual >= hmax))
   {
      output_bit(intractor,TRUE);
      output_bit(extractor,TRUE);
      output_bit(resistencia,OFF);
      output_bit(nebulizador,OFF);
      if((modo_noche    ==    TRUE)    &&    (tempoluz    ==    OFF)){lcd_send_byte(1,3);
output_bit(tempoagua,OFF);}
      lcd_gotoxy(4,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(4,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,0);
      delay_ms(100);
      lcd_gotoxy(9,3);
      lcd_send_byte(1,1);
      delay_ms(100);
      lcd_gotoxy(14,3);
      lcd_send_byte(1,3);
      lcd_gotoxy(19,3);
      lcd_send_byte(1,3);
   }
}


void actualizar_pantalla(void)
{
   lcd_gotoxy(4,1);
   printf(lcd_putc,"%02u:%02u
%02u/%02u/%02u",registros_ds1307[horas],registros_ds1307[minutos],registros_ds1307[dias]
,registros_ds1307[meses],registros_ds1307[anios]);
   lcd_gotoxy(1,2);
```

```
   printf(lcd_putc," %2.2f",temperatura.f);
   lcd_send_byte(1,0b11011111);
   printf(lcd_putc,"C  %2.2f%%HR",humedad.f);
}


void lcd_carga_caracteres(void)
{
   delay_ms(10);
   lcd_send_byte(0,64);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x06);
   lcd_send_byte(1,0x1E);
   lcd_send_byte(1,0x1B);
   lcd_send_byte(1,0x0F);
   lcd_send_byte(1,0x0C);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x00);
   delay_ms(10);
   lcd_send_byte(0,72);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x0C);
   lcd_send_byte(1,0x0F);
   lcd_send_byte(1,0x1B);
   lcd_send_byte(1,0x1E);
   lcd_send_byte(1,0x06);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x00);
   delay_ms(10);
   lcd_send_byte(0,80);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x00);
   delay_ms(10);
   lcd_send_byte(0,88);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x11);
   lcd_send_byte(1,0x11);
   lcd_send_byte(1,0x11);
   lcd_send_byte(1,0x1F);
   lcd_send_byte(1,0x00);
   lcd_send_byte(1,0x00);
   delay_ms(10);
}
```

```
#INT_TIMER0
interrupcion_tmr0()
{
   n++;
   lectura_rtc();
   if(n==30){n=0;lectura_sht11();}
   set_timer0(49911);
}


#INT_EXT1
visualizar_datos_meteo()
{
   delay_ms(300);
   temp_=read_eeprom(e_temp);
   temp_noche=read_eeprom(e_temp_noche);
   humd_=read_eeprom(e_humd);
   humd_noche=read_eeprom(e_humd_noche);
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   printf(lcd_putc,"Temp Dia:   %02u",temp_);
   lcd_send_byte(1,0b11011111);
   printf(lcd_putc,"C");
   lcd_gotoxy(1,2);
   printf(lcd_putc,"Temp Noche: %02u",temp_noche);
   lcd_send_byte(1,0b11011111);
   printf(lcd_putc,"C");
   lcd_gotoxy(1,3);
   printf(lcd_putc,"Humd Dia:   %02u%%HR",humd_);
   lcd_gotoxy(1,4);
   printf(lcd_putc,"Humd Noche: %02u%%HR",humd_noche);
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      if(!t_ok){delay_ms(300); break;}
   };
   intf=0;
   intf1=0;
   lcd_putc("\f");
   return;
}


#INT_EXT2
visualizar_datos_tempo()
{
   delay_ms(300);
   lcd_putc("\f");
   hora=read_eeprom(e_hora_on_1);
   mins=read_eeprom(e_mins_on_1);
```

```
   lcd_gotoxy(1,1);
   lcd_putc("T.ON1:");
   lcd_gotoxy(12,1);
   printf(lcd_putc,"%02u:%02u",hora,mins);
   hora=read_eeprom(e_hora_off_1);
   mins=read_eeprom(e_mins_off_1);
   lcd_gotoxy(1,2);
   lcd_putc("T.OFF1:");
   lcd_gotoxy(12,2);
   printf(lcd_putc,"%02u:%02u",hora,mins);
   hora=read_eeprom(e_hora_on_2);
   mins=read_eeprom(e_mins_on_2);
   lcd_gotoxy(1,3);
   lcd_putc("T.ON2:");
   lcd_gotoxy(12,3);
   printf(lcd_putc,"%02u:%02u",hora,mins);
   hora=read_eeprom(e_hora_off_2);
   mins=read_eeprom(e_mins_off_2);
   lcd_gotoxy(1,4);
   lcd_putc("T.OFF2:");
   lcd_gotoxy(12,4);
   printf(lcd_putc,"%02u:%02u",hora,mins);
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      if(!t_ok){delay_ms(300); break;}
   };
   intf=0;
   intf2=0;
   lcd_putc("\f");
   return;
}
```

- MENU.H

```
void menu_modo(void);
void menu_ajustes(void);
void menu_ajustar_rtc(void);
void menu_ajustar_temporizador(void);
void menu_ajustar_temp_y_humd(void);
void horass(void);
void minss(void);
void func_modo_noche(void);

static unsigned int8 testado, opc, i, tok, tmas, tmns;
static unsigned int8 hora, hora_on_1, hora_off_1, hora_on_2, hora_off_2;
static unsigned int8 mins, mins_on_1, mins_off_1, mins_on_2, mins_off_2;
static unsigned int8 anio, mes, dia, segs;
static unsigned int8 temp_, temp_noche;
static unsigned int8 humd_, humd_noche;
static unsigned int8 dest, dest_m, desh, desh_m;
static unsigned int1 flag;

unsigned int8 e_hora_on_1  = 0;
unsigned int8 e_hora_off_1 = 1;
unsigned int8 e_mins_on_1  = 2;
unsigned int8 e_mins_off_1 = 3;
unsigned int8 e_hora_on_2  = 4;
unsigned int8 e_hora_off_2 = 5;
unsigned int8 e_mins_on_2  = 6;
unsigned int8 e_mins_off_2 = 7;
unsigned int8 e_temp       = 8;
unsigned int8 e_humd       = 9;
unsigned int8 e_temp_noche = 10;
unsigned int8 e_humd_noche = 11;
unsigned int8 e_dest       = 12;
unsigned int8 e_desh       = 13;
unsigned int8 e_modo_noche = 14;

#INT_EXT
menu()
{
   disable_interrupts(GLOBAL);
   disable_interrupts(INT_EXT);
   disable_interrupts(INT_EXT1);
   disable_interrupts(INT_EXT2);
   delay_ms(300);
   testado=0;
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("1 Ajustar Reloj");
   lcd_gotoxy(1,2);
   lcd_putc("2 Ajustar Temporizad");
```

```
   lcd_gotoxy(1,3);
   lcd_putc("3 Ajustar Temp y Hum");
   lcd_gotoxy(1,4);
   lcd_putc("OPCION :");
   for(i=0;i!=255;i++)
   {
      delay_ms(20);

      lcd_gotoxy(11,4);
      printf(lcd_putc,"%u",testado);
      if (!t_ok){delay_ms(300); break;}
      else if (!t_mas){i=0; delay_ms(300); testado++; if(testado>3){testado=0;};}
      else if (!t_mns){i=0; delay_ms(300); testado--; if(testado==255){testado=3;};}
   };
   if(testado==0)
   {
      intf=0; intf1=0; intf2=0;
      enable_interrupts(GLOBAL);
      enable_interrupts(INT_EXT);
      enable_interrupts(INT_EXT1);
      enable_interrupts(INT_EXT2);
      lcd_putc("\f");
      break;
   }
   else if(testado==1){menu_ajustar_rtc(); lcd_putc("\f");}
   else if(testado==2){menu_ajustar_temporizador(); lcd_putc("\f");}
   else if(testado==3){menu_ajustar_temp_y_humd(); lcd_putc("\f");}
}


////////////////////////////////////////////////////////////////////////////
//    Submenu Modo
//
////////////////////////////////////////////////////////////////////////////

void menu_modo(void)
{
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Restaurar Valores?");
   lcd_gotoxy(1,2);
   lcd_putc("0 SI");
   lcd_gotoxy(1,3);
   lcd_putc("1 NO");
   lcd_gotoxy(1,4);
   lcd_putc("OPCION :");
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(10,4);
      printf(lcd_putc,"%u",opc);
```

```
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); opc++; if(opc>1) opc=0;}
      else if(!t_mns){i=0; delay_ms(300); opc--; if(opc==255) opc=1;}
   };
   intf=0; intf1=0; intf2=0;
   if(opc==0){lcd_putc("\f"); break;}
   else if(opc==1){menu_ajustes();}
}

void menu_ajustes(void)
{
   leer_ds1307();
   menu_ajustar_rtc();
   menu_ajustar_temporizador();
   menu_ajustar_temp_y_humd();
   lcd_putc("\f");
}

////////////////////////////////////////////////////////////////////////////
//    Submenu Ajuste RTC
//
////////////////////////////////////////////////////////////////////////////

void menu_ajustar_rtc(void)
{
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Ajustar Hora/Fecha:");
   lcd_gotoxy(1,3);
   printf(lcd_putc,"%02u:%02u
%02u/%02u/%02u",registros_ds1307[horas],registros_ds1307[minutos],registros_ds1307[dias]
,registros_ds1307[meses],registros_ds1307[anios]);
   hora=registros_ds1307[horas];
   if (hora > 23) hora = 0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(1,3);
      printf(lcd_putc,"%02u",hora);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); hora++; if(hora>23){hora=0;};}
      else if(!t_mns){i=0; delay_ms(300); hora--; if(hora==255){hora=23;};}
   };
   escribir_ds1307(horas,hora);
   mins=registros_ds1307[minutos];
   if (mins > 59) mins = 0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(4,3);
      printf(lcd_putc,"%02u",mins);
```

```
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); mins++; if(mins>59){mins=0;};}
      else if(!t_mns){i=0; delay_ms(300); mins--; if(mins==255){mins=59;};}
   };
   escribir_ds1307(minutos,mins);
   dia=registros_ds1307[dias];
   if (dia > 31) dia = 0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(7,3);
      printf(lcd_putc,"%02u",dia);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); dia++; if(dia>31 || dia==0){dia=1;};}
      else if(!t_mns){i=0; delay_ms(300); dia--; if(dia==255 || dia<=0){dia=31;};}
   };
   escribir_ds1307(dias,dia);
   mes=registros_ds1307[meses];
   if (mes > 12) mes = 0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(10,3);
      printf(lcd_putc,"%02u",mes);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); mes++; if(mes>12 || mes==0){mes=1;};}
      else if(!t_mns){i=0; delay_ms(300); mes--; if(mes==255 || mes<=0){mes=12;};}
   };
   escribir_ds1307(meses,mes);
   anio=registros_ds1307[anios];
   if (anio > 99) anio = 0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(13,3);
      printf(lcd_putc,"%02u",anio);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); anio++; if(anio>99){anio=0;};}
      else if(!t_mns){i=0; delay_ms(300); anio--; if(anio==255){anio=99;};}
   };
   escribir_ds1307(anios,anio);
   escribir_ds1307(0,0);
   intf=0; intf1=0; intf2=0;
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
}
```

```
////////////////////////////////////////////////////////////////////////////
//    Submenu Ajuste Temporizador
//
////////////////////////////////////////////////////////////////////////////

void menu_ajustar_temporizador(void)
{
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Ajustar Temporiz.1:");
   lcd_gotoxy(1,3);
   lcd_putc("Introduce T.ON:");
   lcd_gotoxy(10,4);
   lcd_putc(":");
   hora=read_eeprom(e_hora_on_1);
   if(hora==255){hora=0;}
   horass();
   write_eeprom(e_hora_on_1,hora);
   mins=read_eeprom(e_mins_on_1);
   if(mins==255){mins=0;}
   minss();
   write_eeprom(e_mins_on_1,mins);
   lcd_gotoxy(1,3);
   lcd_putc("Introduce T.OFF:");
   lcd_gotoxy(8,4);
   lcd_putc("  :  ");
   hora=read_eeprom(e_hora_off_1);
   if(hora==255){hora=0;}
   horass();
   write_eeprom(e_hora_off_1,hora);
   mins=read_eeprom(e_mins_off_1);
   if(mins==255){mins=0;}
   minss();
   write_eeprom(e_mins_off_1,mins);
   hora_on_1=read_eeprom(e_hora_on_1);
   hora_off_1=read_eeprom(e_hora_off_1);
   mins_on_1=read_eeprom(e_mins_on_1);
   mins_off_1=read_eeprom(e_mins_off_1);

   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Ajustar Temporiz.2 :");
   lcd_gotoxy(1,3);
   lcd_putc("Introduce T.ON:");
   lcd_gotoxy(10,4);
   lcd_putc(":");
   hora=read_eeprom(e_hora_on_2);
   if(hora==255){hora=0;}
   horass();
   write_eeprom(e_hora_on_2,hora);
   mins=read_eeprom(e_mins_on_2);
```

```
   if(mins==255){mins=0;}
   minss();
   write_eeprom(e_mins_on_2,mins);
   lcd_gotoxy(1,3);
   lcd_putc("Introduce T.OFF:");
   lcd_gotoxy(8,4);
   lcd_putc("  :  ");
   hora=read_eeprom(e_hora_off_2);
   if(hora==255){hora=0;}
   horass();
   write_eeprom(e_hora_off_2,hora);
   mins=read_eeprom(e_mins_off_2);
   if(mins==255){mins=0;}
   minss();
   write_eeprom(e_mins_off_2,mins);
   hora_on_2=read_eeprom(e_hora_on_2);
   hora_off_2=read_eeprom(e_hora_off_2);
   mins_on_2=read_eeprom(e_mins_on_2);
   mins_off_2=read_eeprom(e_mins_off_2);

   intf=0; intf1=0; intf2=0;
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
}

void horass(void)
{
   for (i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(8,4);
      printf(lcd_putc,"%02u",hora);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); hora++; if(hora>23){hora=0;};}
      else if(!t_mns){i=0; delay_ms(300); hora--; if(hora==255){hora=23;};}
   };
}

void minss(void)
{
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(11,4);
      printf(lcd_putc,"%02u",mins);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); mins++; if(mins>59){mins=0;};}
      else if(!t_mns){i=0; delay_ms(300); mins--; if(mins==255){mins=59;};}
   };
```

```
}
////////////////////////////////////////////////////////////////////////////
//    Submenu Ajuste Max y Min
//
////////////////////////////////////////////////////////////////////////////

void menu_ajustar_temp_y_humd(void)
{
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Introduce Temp.:");
   temp_=read_eeprom(e_temp);
   if(temp_<15 || temp_>40){temp_=30;}
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(1,2);
      printf(lcd_putc," %02u",temp_);
      lcd_send_byte(1,0b11011111);
      printf(lcd_putc,"C");
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); temp_++; if(temp_>40){temp_=15;};}
      else     if(!t_mns){i=0;     delay_ms(300);     temp_--;     if(temp_==255     ||
temp_<15){temp_=40;};}
   };
   write_eeprom(e_temp,temp_);

   lcd_gotoxy(1,3);
   lcd_putc("Introduce Humedad:");
   humd_=read_eeprom(e_humd);
   if(humd_<30 || humd_>99){humd_=70;}
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(1,4);
      printf(lcd_putc," %02u%%HR",humd_);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); humd_++; if(humd_>99){humd_=30;};}
      else     if(!t_mns){i=0;     delay_ms(300);     humd_--;     if(humd_==255     ||
humd_<30){humd_=99;};}
   };
   write_eeprom(e_humd,humd_);

   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("Activar Modo Noche?");
   lcd_gotoxy(1,2);
   lcd_putc("0 SI");
   lcd_gotoxy(1,3);
   lcd_putc("1 NO");
   lcd_gotoxy(1,4);
```

```
   lcd_putc("OPCION :");
   opc=0;
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(10,4);
      printf(lcd_putc,"%u",opc);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); opc++; if(opc>1){opc=0;};}
      else if(!t_mns){i=0, delay_ms(300); opc--; if(opc==255){opc=1;};}
   };
   if(opc==0){func_modo_noche();}
   else
   {
   intf=0; intf1=0; intf2=0;
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
   break;
   }
}

void func_modo_noche(void)
{
   write_eeprom(e_modo_noche,1);
   lcd_putc("\f");
   lcd_gotoxy(1,1);
   lcd_putc("La Temp desciende:");
   dest_m = temp_ - 15;
   dest=read_eeprom(e_dest);
   if(dest==255 || dest>dest_m){dest=dest_m;}
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%02u",dest);
      lcd_send_byte(1,0b11011111);
      printf(lcd_putc,"C");
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); dest++; if(dest>dest_m){dest=0;};}
      else if(!t_mns){i=0; delay_ms(300); dest--; if(dest==255){dest=dest_m;};}
   };
   temp_noche=temp_-dest;
   write_eeprom(e_dest,dest);
   write_eeprom(e_temp_noche,temp_noche);
   lcd_gotoxy(1,3);
   lcd_putc("La Hum desciende:");
   desh_m = humd_ - 30;
   desh=read_eeprom(e_desh);
   if(desh==255 || desh>desh_m){desh=desh_m;}
```

```
   for(i=0;i!=255;i++)
   {
      delay_ms(20);
      lcd_gotoxy(1,4);
      printf(lcd_putc,"%02u%%HR",desh);
      if(!t_ok){delay_ms(300); break;}
      else if(!t_mas){i=0; delay_ms(300); desh++; if(desh>desh_m){desh=0;};}
      else if(!t_mns){i=0; delay_ms(300); desh--; if(desh==255){desh=desh_m;};}
   };
   humd_noche=humd_-desh;
   write_eeprom(e_desh,desh);
   write_eeprom(e_humd_noche,humd_noche);
   intf=0; intf1=0; intf2=0;
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
}
```

- LCD.H

```
#define PORTC 0xf82
#define TRISC 0xf94

#bit lcd_en = PORTC.0
#bit tris_lcd_en = TRISC.0
#bit lcd_rs = PORTC.1
#bit tris_lcd_rs = TRISC.1
#bit lcd_db4 = PORTC.2
#bit tris_lcd_db4 = TRISC.2
#bit lcd_db5 = PORTC.3
#bit tris_lcd_db5 = TRISC.3
#bit lcd_db6 = PORTC.4
#bit tris_lcd_db6 = TRISC.4
#bit lcd_db7 = PORTC.5
#bit tris_lcd_db7 = TRISC.5

#define LCD_DATO 1
#define LCD_INST 0

#define LCD_LINEA1 0x80
#define LCD_LINEA2 0xc0
#define LCD_LINEA3 0x94
#define LCD_LINEA4 0xd4

#define LCD_FUNCTION_SET   0b00101000
#define LCD_DISPLAY_CURSOR 0b00001100
#define LCD_ENTRY_MODE     0b00000110
#define LCD_CLEAR_DISPLAY  0b00000001


void lcd_set_write()
{
   tris_lcd_db4 = 0;
   tris_lcd_db5 = 0;
   tris_lcd_db6 = 0;
   tris_lcd_db7 = 0;
}


void lcd_send_nibble(int8 n)
{
   if (bit_test(n,0))
      lcd_db4 = 1;
   else
      lcd_db4 = 0;
   if (bit_test(n,1))
      lcd_db5 = 1;
   else
```

```
      lcd_db5 = 0;
   if (bit_test(n,2))
      lcd_db6 = 1;
   else
      lcd_db6 = 0;
   if (bit_test(n,3))
      lcd_db7 = 1;
   else
      lcd_db7 = 0;
   delay_cycles(1);
   lcd_en = 1;
   delay_us(2);
   lcd_en = 0;
}


void lcd_send_byte (int1 select, int8 n)
{
   lcd_rs = 0;
   delay_us(10);
   lcd_rs = select;
   delay_cycles(1);
   lcd_en = 0;
   lcd_send_nibble(n >> 4);
   lcd_send_nibble(n);
}


void lcd_init()
{
   int8 i, count=0;

   lcd_set_write();
   tris_lcd_en = 0;
   tris_lcd_rs = 0;
   lcd_en = 0;
   lcd_rs = 0;
   delay_ms(15);

   for(i=1; i<=3; ++i)
   {
      lcd_send_nibble(0b0011);
      delay_ms(5);
   }
   lcd_send_nibble(0b0010);
   lcd_send_byte(LCD_INST, LCD_FUNCTION_SET);
   lcd_send_byte(LCD_INST, LCD_DISPLAY_CURSOR);
   lcd_send_byte(LCD_INST, LCD_ENTRY_MODE);
   lcd_send_byte(LCD_INST, LCD_CLEAR_DISPLAY);
}
```

```
void lcd_gotoxy(int8 x, int8 y)
{
   int8 const address[4]={LCD_LINEA1,LCD_LINEA2,LCD_LINEA3,LCD_LINEA4};
   int8 pos;

   pos=address[y-1]+(x-1);
   lcd_send_byte (LCD_INST, pos);
}


void lcd_putc(char c)
{
   switch (c)
   {
     case '\f'   : lcd_send_byte(0,1);
                   delay_ms(2);
                   break;
     case '\n'   : lcd_gotoxy(1,2);
                   break;
     case '\b'   : lcd_send_byte(LCD_INST,0x10);
                   break;
     case '\t'   : lcd_send_byte(LCD_INST,0x14);
                   break;
     case '\r'   : lcd_send_byte(LCD_INST,0x18);
                   break;
     case '\v'   : lcd_send_byte(LCD_INST,0x1C);
                   break;
     default     : lcd_send_byte(LCD_DATO,c);
                   break;
   }
}


void lcd_clear()
{
   lcd_send_byte(LCD_INST,0x01);
}


void lcd_home()
{
   lcd_send_byte(LCD_INST,0x02);
}


void lcd_erase_line(int8 x)
{
   int8 i;

   for(i=1;i<=x;++i)
```

```
   {
      lcd_send_byte(LCD_DATO,32);
   }
}
```

- DS1307.H

```
void escribir_ds1307(unsigned int8 direccion, unsigned int8 val);
void leer_ds1307(void);
unsigned int8 bin2bcd(unsigned int8 valor_binario);
unsigned int8 bcd2bin(unsigned int8 valor_bcd);

#define escribir_ds1307_cmd           0xd0
#define leer_ds1307_cmd               0xd1
#define segundos                      0
#define minutos                       1
#define horas                         2
#define dia_semana                    3
#define dias                          4
#define meses                         5
#define anios                         6
#define registro_de_control           7
#define DS1307_DATE_TIME_BYTE_COUNT   7
#define DS1307_NVRAM_START_ADDR       8
#define DS1307_CONTROL_REG_INIT_VALUE  0x80
//#define DS1307_CONTROL_REG_INIT_VALUE 0x13

unsigned int8 registros_ds1307[registro_de_control];


void escribir_ds1307(unsigned int8 direccion, unsigned int8 val)
{
   disable_interrupts(GLOBAL);

   val = bin2bcd(val);

   i2c_start();
   i2c_write(escribir_ds1307_cmd);
   i2c_write(direccion);
   i2c_write(val);
   i2c_stop();

   enable_interrupts(GLOBAL);
}


void leer_ds1307(void)
{
   unsigned int8 i;

   i2c_start();
   i2c_write(escribir_ds1307_cmd);
   i2c_write(segundos);
   i2c_start();
   i2c_write(leer_ds1307_cmd);
```

```
   registros_ds1307[segundos] = i2c_read() & 0x7f;
   registros_ds1307[minutos] = i2c_read() & 0x7f;
   registros_ds1307[horas] = i2c_read() & 0x3f;
   registros_ds1307[dia_semana] = i2c_read() & 0x07;
   registros_ds1307[dias] = i2c_read() & 0x3f;
   registros_ds1307[meses] = i2c_read() & 0x1f;
   registros_ds1307[anios] = i2c_read(0);

   i2c_stop();

   for(i = 0; i < 7; i++)
   {
      registros_ds1307[i] = bcd2bin(registros_ds1307[i]);
   }
}


unsigned int8 bin2bcd(unsigned int8 valor_binario)
{
   unsigned int8 temp;
   unsigned int8 retval;

   temp = valor_binario;
   retval = 0;

   while(1)
   {
      if(temp >= 10)
      {
         temp -= 10;
         retval += 0x10;
      }
      else
      {
         retval += temp;
         break;
      }
   }

   return(retval);
}


unsigned int8 bcd2bin(unsigned int8 valor_bcd)
{
   unsigned int8 valor_bin;

   valor_bin = valor_bcd;
   valor_bin >>= 1;
   valor_bin &= 0x78;
```

```
   return(valor_bin + (valor_bin >> 2) + (valor_bcd & 0x0f));
}
```

- SHT11.H

```
#define DATOS  PIN_A4
#define CLOCK  PIN_A5

#define SHT11_noACK 0
#define SHT11_ACK   1

#define SHT11_STATUS_REG_W 0x06
#define SHT11_STATUS_REG_R 0x07
#define SHT11_MEASURE_TEMP 0x03
#define SHT11_MEASURE_HUMI 0x05
#define SHT11_RESET        0x1E

enum {TEMP,HUMI};


byte sht11_escribir_byte(byte value)
{
      byte i,error=0;

      for (i=128;i>0;i/=2)
   {
            if (i & value)    output_high(DATOS);
            else output_low(DATOS);
            output_high(CLOCK);
            delay_us(5);
            output_low(CLOCK);
   }
      output_high(DATOS);
      output_high(CLOCK);
      error=input(DATOS);
      output_low(CLOCK);
      return error;
}


byte sht11_leer_byte(byte ack)
{
      byte i,val=0;

      output_high(DATOS);
      for (i=128;i>0;i/=2)
   {
            output_high(CLOCK);
            if (input(DATOS)) val=(val | i);
      output_low(CLOCK);
      }
   if (ack) output_low(DATOS);
      else output_float(DATOS);
```

```
      output_high(CLOCK);
      delay_us(5);
      output_low(CLOCK);
      output_high(DATOS);
      return val;
}


void sht11_init(void)
{
      output_high(DATOS);
      output_low(CLOCK);
      delay_us(2);
      output_high(CLOCK);
      delay_us(2);
      output_low(DATOS);
      delay_us(2);
      output_low(CLOCK);
      delay_us(5);
      output_high(CLOCK);
      delay_us(2);
      output_float(DATOS);
      delay_us(2);
      output_low(CLOCK);
}


void sht11_hard_reset(void)
{
      byte i;

      output_high(DATOS);
      output_low(CLOCK);
      for(i=0;i<9;i++)
   {
            output_high(CLOCK);
            delay_us(2);
            output_low(CLOCK);
            delay_us(2);
      }
      sht11_init();
}

byte sht11_soft_reset(void)
{
      byte error=0;

      sht11_hard_reset();
      error+=sht11_escribir_byte(SHT11_RESET);

      return error;
```

```
}


byte sht11_leer_registro_estado(byte *p_valor, byte *p_checksum)
{
      byte error=0;

      sht11_init();
      error = sht11_escribir_byte(SHT11_STATUS_REG_R);
      *p_valor = sht11_leer_byte(SHT11_ACK);
      *p_checksum = sht11_leer_byte(SHT11_NOACK);

      return error;
}


byte sht11_escribir_registro_estado(byte *p_valor)
{
      byte error=0;

      sht11_init();
      error += sht11_escribir_byte(SHT11_STATUS_REG_W);
      error += sht11_escribir_byte(*p_valor);

      return error;
}


byte sht11_medicion(byte *p_valor, byte *p_checksum, byte modo)
{
      byte error=0;
      int16 i;

      sht11_init();
      switch(modo)
   {
            case TEMP   : error+=sht11_escribir_byte(SHT11_MEASURE_TEMP); break;
            case HUMI   : error+=sht11_escribir_byte(SHT11_MEASURE_HUMI); break;
            default     : break;
      }
      for (i=0;i<65535;i++) if(input(DATOS)==0) break;
      if(input(DATOS)) error+=1;
      *(p_valor+1)  =sht11_leer_byte(SHT11_ACK);
      *(p_valor)    =sht11_leer_byte(SHT11_ACK);
      *(p_checksum) =sht11_leer_byte(SHT11_NOACK);

      return error;
}


void sht11_calculos(float *p_humedad, float *p_temperatura)
```

```
{
      const float C1=-4.0;
      const float C2=+0.0405;
      const float C3=-0.0000028;
      const float T1=+0.01;
      const float T2=+0.00008;

      float rh;
      float t;
      float rh_lin;
      float rh_true;
      float t_C;

      rh = *p_humedad;
      t = *p_temperatura;

      t_C = t*0.01 - 40;
      rh_lin=C3*rh*rh + C2*rh + C1;
      rh_true=(t_C-25)*(T1+T2*rh)+rh_lin;
      if(rh_true>100)rh_true=100;
      if(rh_true<0.1)rh_true=0.1;

      *p_temperatura=t_C;
      *p_humedad=rh_true;
}
```

Capítulo 5
Vías futuras

5.1 Conclusiones

- Se ha desarrollado un sistema microcontrolado totalmente autónomo para mantener la temperatura y humedades constantes en un habitáculo cerrado. Además de dos temporizadores que controlan las horas de luz y agua para el sistema.

- El diseño utiliza un sensor digital para la medición de humedad y temperatura, este hace uso de un bus I2C para la comunicación con el microcontrolador, con una librería escrita en lenguaje C se realiza la conversión a unos valores tratables para el termostato e higrostato.

- Se utiliza un RTC que utiliza el bus I2C para enviar la fecha y hora, que han sido ajustadas por el usuario, al microcontrolador, donde estos datos son tratados para los temporizadores.

- Se ha desarrollo un interfaz con el usuario formado por un LCD de 20x4 líneas y 3 botones. El entorno consiste en una pantalla que muestra el estado de las

salidas del sistema, además de fecha, hora y humedad y temperatura. Con un menú muy intuitivo para configurar las variables del sistema.

- Realización de una etapa de potencia para controlar las salidas del sistema que exceptuando los dos ventiladores de 12Vcc, eran de 220Vca. Bombilla cerámica, nebulizador, bomba de agua y balasto electrónico.

- El desarrollo del sistema ha supuesto el aprendizaje de varios programas para el desarrollo de sistemas con microcontroladores, como son el compilador de CCS, el programa de simulación Proteus, el programa de grabación Winpic800. Se ha profundizado en el desarrollo y realización de circuitos electrónicos con la suite OrCAD y con un acercamiento hacia suites más complejas como Protel.

- Se ha conseguido realizar un sistema microcontrolado que fácilmente puede ser adaptado para operar en otros campos, acuarios, invernaderos, etc.

- Este proyecto ha supuesto una auto superación desde su inicio hasta el final. Con un resultado muy satisfactorio y con la posibilidad de realizar futuros trabajos con este diseño.

5.2 Futuros Trabajos

- Este diseño puede ser mejorado en ciertos aspectos, puesto que está diseñado para operar en terrarios, podría implementarse una base de datos con información sobre reptiles en la cual estén predefinidas las horas de luz, humedad y temperatura adecuadas para el animal. Esa base de datos también podría incluir la variación de temperatura y humedad para las distintas estaciones del año para que el reptil pueda realizar la hibernación adecuadamente, actualmente esto se debe realizar manualmente.

- Debido a la versatilidad del sistema, podría sustituir el microcontrolador por un PIC 18F2550 para realizar la conexión al PC por USB, de esta forma poder variar los parámetros del sistema e incluso actualizar la posible base de datos

con un sencillo entorno realizado en un lenguaje de programación de alto nivel como podría ser el Visual C#.

- Se podría aprovechar la capacidad del balasto electrónico sobre ser controlado con pulsos, de esta forma poder simular la puesta y el ocaso del sol con la luz ultra violeta, causando de esta forma un menor estrés al reptil.

- Adicionalmente podría terminarse el estudio para realizar la adaptación de la pantalla gráfica a color al sistema. Cabe recordar que fue descartada por la falta de instrumental y componentes electrónicos para realizar la placa en SMD, ya que como se pudo observar en la imagen, se logró realizar la librería para su control.

Capítulo 6
Referencias

Libros

- Microcontroladores PIC. Diseño práctico de aplicaciones. Primera parte. 3ª Edición. José M.ª Angulo Usategui y Ignacio Angulo Martínez. Ed. Mc Graw Hill, 2003.
- Microcontroladores PIC. Diseño práctico de aplicaciones. Segunda parte. José M.ª Angulo Usategui, Susana Romero Yesa y Ignacio Angulo Martínez. Ed. Mc Graw Hill, 2000.
- Microcontrolador 16F84. Desarrollo de proyectos. Enrique Palacios, Fernando Remiro y Lucas J. López. Ed. Ra-Ma, 2004.
- Electrónica de Potencia. Jose Antonio Villarejo Mañas, Joaquín Roca González, Joaquín Roca Dorda. Ed. Escarabajal, 2000.

Páginas Web

- Foro TodoPIC
  http://miarroba.com/foros/ver.php?id=6510

- Página de Sisco, Winpic800, GTP USB
  http://perso.wanadoo.es/siscobf
- Página de Jaime Fernández-Caro aka J1M, GTP Lite, GTP USB Lite, PicUSB
  http://perso.wanadoo.es/j1m http://www.hobbypic.com
- ePraktukum, proyectos con Pics
  http://www.epraktikum.co.yu
- Ejemplos en C para Pics
  http://www.microchipc.com
- Proyectos con Pics
  http://www.techdesign.be/projects.htm
- CCS, Compilador C
  http://www.ccsinfo.com
- Sensirion, sensor SHT11
  http://www.sensirion.com
- Maxim, RTC DS1307
  http://www.maxim-ic.com
- Powertip, LCD 20x4
  http://www.powertipusa.com
- Microchip, PIC 18F2520
  http://www.microchip.com
- Exoterra, productos para terrarios
  http://www.exo-terra.com/ES/index.html