



UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR  
DE INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA  
INGENIERO EN ELECTRÓNICA

# Horno de Soldadura por Reflujo para SMD

Autor:

**Alejandro Fernández Blanco**

Tutor:

**Jesús M. Hernández Mangas**

Valladolid, noviembre de 2008



---

TÍTULO:	Horno de Soldadura por Reflujo para SMD
AUTOR:	Alejandro Fernández Blanco
TUTOR:	Jesús M. Hernández Mangas
DEPARTAMENTO:	Electricidad y Electrónica

---

### Miembros del tribunal

---

PRESIDENTE:	Luis A. Bailón Vega
VOCAL:	Jesús M. Hernández Mangas
SECRETARIO:	Jesús Arias Álvarez

---

FECHA DE LECTURA:	
CALIFICACIÓN:	

---

### Resumen del proyecto

Hoy día el objetivo de la electrónica es disminuir el espacio ocupado por los dispositivos incluyendo en ellos más componentes y mayores prestaciones. Esto hace muy complicada la soldadura manual de los dispositivos de montaje superficial (SMD).

Desarrollando un sistema de control se consigue diseñar un horno que es capaz de controlar la temperatura de un horno que permite soldar componentes SMD mediante el proceso denominado "*Soldadura por reflujo*" que consiste en realizar una curva térmica determinada dentro de un horno, de forma que se suelden los dispositivos en la PCB, sin intervención humana durante el proceso.

### Palabras clave

PID, Soldadura, Reflujo, Horno, SMD, PICC18, LCD, SPI.

## **Abstract**

Today the goal of electronics area is to reduce the space occupied by devices including them in more components and greater features. This makes it very complicated soldering the Surface Mounted Devices (SMD).

Developing a system of control is achieved by designing an oven that is capable of controlling the temperature of an oven that allows soldering SMD components through a process called "*Reflow Soldering*" which is to make a curve within a given thermal oven, so that the devices are soldered to the PCB, without human intervention during the process.

## **Key words**

PID, Soldering, Reflow, Oven, SMD, PICC18, LCD, SPI.



*A mis padres por hacer posible que este  
proyecto sea realidad, en especial a tí  
papá. (Vicente Fernández Gabriel)*



# Índice general

<b>1. Introducción</b>	<b>15</b>
1.1. Objetivos . . . . .	15
1.2. Fases . . . . .	16
<b>2. Desarrollo Hardware</b>	<b>19</b>
2.1. Diseño del circuito . . . . .	19
2.1.1. Bloque de alimentación . . . . .	20
2.1.2. Interfaz de comunicación . . . . .	22
2.1.3. Bloque de control . . . . .	24
2.2. Fabricación de la PCB . . . . .	26
2.2.1. Diseño de la PCB . . . . .	26
2.2.2. Soldado y pruebas sobre la PCB . . . . .	28
2.3. Montaje del horno . . . . .	30
2.3.1. Resistencias . . . . .	33
<b>3. Desarrollo Software</b>	<b>37</b>
3.1. Configuración del compilador PICC18 . . . . .	37
3.2. Bootloader . . . . .	38
3.3. Programa principal . . . . .	40
3.3.1. Teclado . . . . .	41
3.3.2. Puerto Serie . . . . .	44
3.3.3. Lectura de Temperatura . . . . .	49
3.3.4. Memoria externa, M25P16 . . . . .	51
3.3.5. Otras funciones . . . . .	53
3.4. Control PID . . . . .	54
3.5. LCD Gráfico . . . . .	58
3.6. Programa Final . . . . .	61

<b>4. Manual de usuario</b>	<b>63</b>
4.1. Arranque del horno SMD . . . . .	63
4.2. Menú de Configuración . . . . .	65
4.2.1. Submenú de resistencias . . . . .	65
4.2.2. Submenú de velocidad de comunicación . . . . .	66
4.3. Menú de Selección . . . . .	66
4.4. Menú de Transferencia . . . . .	69
4.4.1. Herramienta Hyperterminal . . . . .	70
4.4.2. Transferencia del proceso al PC . . . . .	72
4.4.3. Transferencia de las curvas hacia el PC . . . . .	73
4.4.4. Transferencia de las curvas desde PC hacia el Horno . . . . .	74
4.5. Menú de Gráfica almacenada . . . . .	75
4.6. Ejecución de una curva . . . . .	76
4.7. Errores . . . . .	77
<b>5. Pruebas</b>	<b>79</b>
5.1. Proceso de Soldadura . . . . .	79
5.1.1. Primera prueba . . . . .	79
5.1.2. Segunda prueba . . . . .	85
5.2. Proceso de Desoldado . . . . .	88
5.2.1. Primera prueba . . . . .	88
5.2.2. Segunda prueba . . . . .	90
<b>6. Presupuesto</b>	<b>93</b>
6.1. Coste de materiales . . . . .	93
6.2. Coste del producto . . . . .	95
6.3. Producción en cadena . . . . .	96
<b>7. Conclusiones</b>	<b>97</b>
<b>A. Código principal: main.c</b>	<b>99</b>
<b>B. Archivo cabecera: main.h</b>	<b>143</b>
<b>C. Código para el LCD: lcd.c</b>	<b>149</b>
<b>D. Cabecera para la fuente del LCD: font.h</b>	<b>167</b>
<b>8. Bibliografía</b>	<b>182</b>

# Índice de figuras

2.1. Horno de partida . . . . .	19
2.2. Hoja principal de Proteus donde se divide el diseño en 3 partes . . . . .	20
2.3. Esquema de la alimentación del control . . . . .	21
2.4. Esquema de la interfaz de comunicaciones . . . . .	23
2.5. Esquema del circuito de control implementado . . . . .	25
2.6. Esquema del rutado de las pistas sobre la PCB . . . . .	27
2.7. Placa de circuito impreso con la serigrafía . . . . .	28
2.8. Placa de control final introducida en el horno . . . . .	29
2.9. Pieza frontal modificada . . . . .	30
2.10. Teclado modificado para el horno . . . . .	31
2.11. Silicona de alta temperatura utilizada . . . . .	31
2.12. Chasis del horno moodificado . . . . .	32
2.13. Horno con el aislante colocado y el frontal montado . . . . .	32
2.14. Ventilador añadido para refrescar la zona de control . . . . .	33
2.15. Resistencia de 600W retirada . . . . .	33
2.16. Adaptación de la nueva resistencia . . . . .	34
2.17. Horno Delonghi EO2131 modificado . . . . .	34
2.18. Puesto de trabajo . . . . .	35
3.1. Ventana de configuración del compilador y línea utilizada . . . . .	38
3.2. Ventana del programa Bootloader . . . . .	39
3.3. Distribución de las patillas del PIC18F458 . . . . .	40
3.4. Prestaciones y características del micro utilizado . . . . .	41
3.5. Diagrama de flujo del escaneo del Teclado . . . . .	42
3.6. Diagrama de flujo de la función Teclado . . . . .	43
3.7. Ecuaciones utilizadas para calcular las constantes del PWM . . . . .	44
3.8. Ecuaciones utilizadas para calcular los parámetros de velocidad . . . . .	44
3.9. Ecuaciones utilizadas para calcular los parámetros de velocidad . . . . .	46
3.10. Ecuaciones utilizadas para calcular los parámetros de velocidad . . . . .	47

3.11. Ecuaciones utilizadas para calcular los parámetros de velocidad . . . . .	48
3.12. Conexión típica de una comunicación SPI . . . . .	49
3.13. Modelo de SPI que utiliza el chip MAX6675 . . . . .	50
3.14. Modelo de SPI que utiliza la memoria M25P16 . . . . .	51
3.15. Modelo de SPI que utiliza la memoria M25P16 . . . . .	52
3.16. Diagrama de flujo la función Control . . . . .	55
3.17. Curva de máxima pendiente obtenida para el modelado de las resistencias .	56
3.18. Diagrama de flujo del control PID . . . . .	57
3.19. Curvas reales que más se aproximan a la curva teórica . . . . .	58
3.20. Cronogramas de escritura y lectura del LCD . . . . .	59
3.21. Diagrama de flujo del programa final . . . . .	62
4.1. Ventana de inicio que muestra el programa . . . . .	64
4.2. Menú principal del Horno SMD . . . . .	64
4.3. Menú de configuración del Horno SMD . . . . .	65
4.4. Submenú de selección de las resistencias . . . . .	65
4.5. Submenú de selección de la velocidad de comunicación . . . . .	66
4.6. Menú de selección de curva del Horno SMD . . . . .	66
4.7. Pantalla de confirmación para ejecutar la curva seleccionada . . . . .	67
4.8. Pantalla de confirmación para borrar la curva seleccionada . . . . .	68
4.9. Submenú de edición de la curva seleccionada . . . . .	68
4.10. Menú de transferencia de datos del Horno SMD . . . . .	69
4.11. Pantalla de confirmación de la operación y velocidad de comunicación . . .	69
4.12. Pantalla inicial de la herramienta Hyperterminal . . . . .	70
4.13. Pantalla de configuración de la conexión . . . . .	71
4.14. Pantalla de propiedades de la conexión . . . . .	71
4.15. Menú de Transferencia del Hyperterminal . . . . .	72
4.16. Pantalla del Hyperterminal en proceso de captura de datos . . . . .	72
4.17. Pantalla del Hyperterminal con la captura de las curvas . . . . .	73
4.18. Documento de texto con las curvas que se han recibido . . . . .	74
4.19. Pantalla de espera de datos del PC . . . . .	74
4.20. Documento de texto con los puntos que se quieren enviar . . . . .	75
4.21. Menú de gráfica guardada del Horno SMD . . . . .	76
4.22. Menú de la representación de la curva ejecutada en el Horno SMD . . . . .	76
4.23. Menú de la representación de la curva ejecutada en el Horno SMD . . . . .	77
5.1. Elementos utilizados en la soldadura . . . . .	80
5.2. Bolas depositadas sobre los pads de conectores . . . . .	80

---

5.3. Conectores colocados sobre las bolas de pasta . . . . .	81
5.4. Tiras depositadas en los anclajes donde irá el integrado . . . . .	81
5.5. Integrado colocado con cuidado sobre las tiras de pasta . . . . .	82
5.6. Placa dispuesta para soldar . . . . .	82
5.7. Curvas proporcionadas por el fabricante . . . . .	83
5.8. Resultado obtenido después del proceso de soldado . . . . .	83
5.9. Curva realizada por el horno vista en el aparato . . . . .	84
5.10. Comparación de las curvas teórica y real realizada . . . . .	84
5.11. Tiras depositadas en los anclajes donde irá el integrado . . . . .	85
5.12. Integrado colocado con cuidado sobre las tiras de pasta . . . . .	85
5.13. Placa dentro del horno durante el proceso . . . . .	86
5.14. Placa una vez finalizada la prueba de soldadura . . . . .	86
5.15. Muestra al microscópio del resultado de la soldadura . . . . .	87
5.16. Material utilizado para limpiar las agujas . . . . .	87
5.17. Placa que se desea desoldar . . . . .	88
5.18. Colocación de la placa en el horno . . . . .	89
5.19. Disposición de la placa durante el calentamiento . . . . .	89
5.20. Resultado tras la prueba de desoldado . . . . .	90
5.21. Forma de calentar en el desoldado . . . . .	90
5.22. Resultado del proceso de desoldado . . . . .	91





# Índice de cuadros

4.1. Solución a posibles errores . . . . .	78
6.1. Costes de los materiales utilizados en el horno . . . . .	94
6.2. Coste de los materiales extras necesarios . . . . .	94
6.3. Presupuesto de la fabricación del producto final . . . . .	95
6.4. Presupuesto de la fabricación de un equipo . . . . .	95
6.5. Presupuesto de la fabricación de un equipo . . . . .	96



# Capítulo 1

## Introducción

Hoy día la evolución de la electrónica consiste en minimizar el espacio ocupado, por ello los componente discretos de gran tamaño están dejando paso a los componentes de montaje superficial o también conocidos como SMD (Surface Mounted Device) Además, la tendencia de estos es integrar el mayor número de prestaciones lo que influye en el número de anclajes de los que dispondrá. Esto conlleva que el espacio entre patillas sea mínimo y la soldadura manual un factor de riesgo para el resultado final. Por ello, el proyecto que aquí se presenta esta encaminado a la eliminación de estos inconvenientes, de forma que se puedan realizar diseños electrónicos que incluyan componentes SMD.

### 1.1. Objetivos

El objetivo principal es controlar la temperatura interna de un horno convencional de forma que siga la curva característica de los procesos de soldadura por reflujo y ésta sea introducida por el usuario. Conseguido este paso se busca que mediante un visor gráfico se muestre curva térmica y la temperatura interna del horno.

Alcanzados estos objetivos, se orienta el proyecto hacia una interfaz de comunicación con el usuario de forma que éste pueda introducir diversas curvas térmicas mediante un teclado, las cuales sean llevadas a cabo por el horno con diferentes configuraciones dispuestas por el usuario.

Para mejorar todo el diseño se añade además un protocolo de comunicación con un PC, de forma que se pueda realizar la actualización del firmware y el intercambio de datos almacenados en la memoria, como la curva real seguida por el horno para un posterior análisis. Todo el desarrollo se realiza en base a la integración de todo el sistema de control

dentro del producto base de forma que sea lo más robusto y compacto posible, haciendo para ello un diseño acorde con las dimensiones ofrecidas por el horno a utilizar.

## 1.2. Fases

La primera fase es un estudio detallado de los componentes que se van a utilizar para realizar el diseño. Acto seguido se realiza el diseño de la placa de control que se va a crear. Como se quiere un sistema compacto, la alimentación es única para el horno, con lo que se partirá de la tensión de red para el diseño.

Para el control se toma un microprocesador que ofrezca las prestaciones que más se adecuen a los objetivos del proyecto. Se debe controlar el funcionamiento de las resistencias disponibles en el horno de forma independiente, así como disponer de un aviso sonoro que indique el funcionamiento para prevenir accidentes. Se tienen unos sensores de temperatura que toman medidas para actuar convenientemente sobre el horno y que sirvan para indicar la temperatura interna en todo momento.

Para la interfaz gráfica se dispone de un display que deberá mostrar un menú en el que el usuario elija una opción de las ofertadas, tales como realizar la curva térmica o crear una nueva para realizar. Todo el proceso de selección se realizará a partir de un teclado.

En la comunicación con el ordenador, se toma el protocolo de comunicación serie de forma que se pueda programar todo el sistema una vez montado. Además de poder pedir los datos almacenados en la memoria interna.

Una vez realizada la fase de diseño, se procede a la fase de montaje y test, donde se realiza la placa de circuito impreso y el montaje de los componentes sobre la PCB y se comprueba el correcto funcionamiento.

En la fase de programación, se desarrolla el algoritmo de control basado en un PID, las diferentes aplicaciones que permitirán interactuar con el usuario y el programa de arranque (Bootloader) para poder programar el microprocesador mediante el puerto serie, de forma que cuando el horno este montado, se pueda actualizar el programa sin necesidad de desmontarlo.

Finalmente, se pasa al montaje del producto final y a la realización de diversas pruebas

---

para ver el correcto funcionamiento, tanto del control como del circuito en un ambiente de temperatura elevada, evaluando los riesgos y añadiendo las prevenciones que puedan ser necesarias.



# Capítulo 2

## Desarrollo Hardware

El proyecto lleva una fase de diseño hardware que esta dividida entre la creación de la placa de control y la integración de ésta en el horno DeLonghi modelo EO2131.



Figura 2.1: Horno de partida

### 2.1. Diseño del circuito

El desarrollo de la primera fase consta del estudio y la evaluación de los objetivos pedidos y los dispositivos que ofrezcan una solución a estos. Partiendo de dos artículos de

la revista Elektor[1] de las que disponen estos hornos y se establecen las que se requerirán para este proyecto. Realizado el estudio se pasa al diseño en tres bloques, los cuales se detallan a continuación, utilizando para ello la herramienta PROTEUS, que es un programa de diseño electrónico.

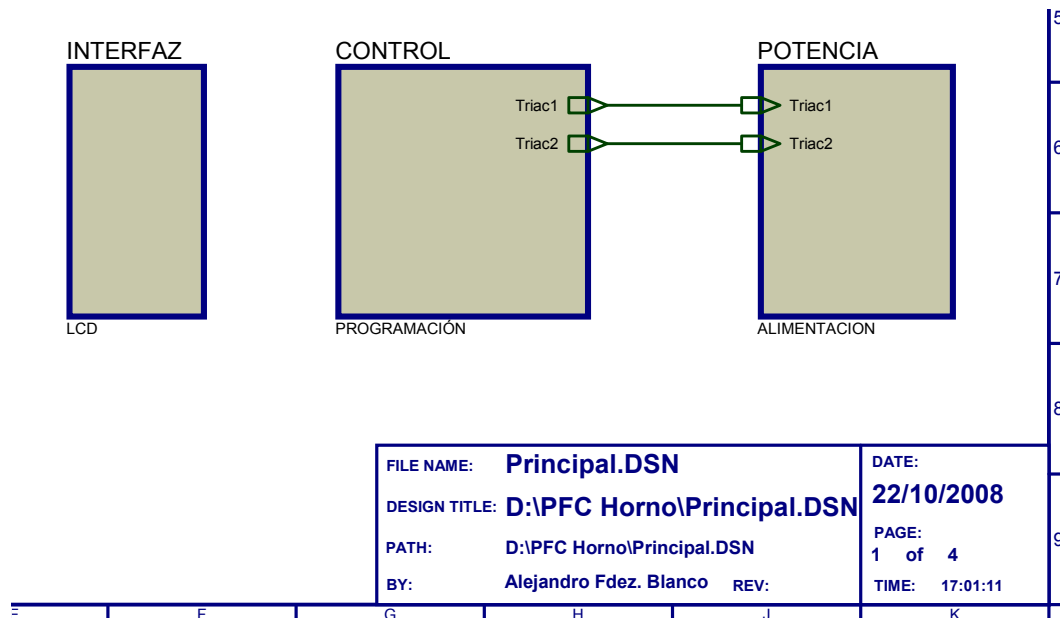


Figura 2.2: Hoja principal de Proteus donde se divide el diseño en 3 partes

### 2.1.1. Bloque de alimentación

En el bloque de alimentación se detalla el circuito que proporciona la alimentación al horno así como al circuito de control. Como se desea una sola alimentación, y el control se realiza con electrónica alimentada a 5V, se añade un transformador que proporciona la corriente y tensión necesarias. Tras estudiar diversas posibilidades, se toma un transformador[2] de 230V-9V conectado en la configuración que proporcione una corriente suficiente para la placa de control. No obstante se añade un fusible que deberá estar limitado al máximo de corriente que se requiera. Tras este se colocará un puente rectificador para obtener la tensión continua, y finalmente se añadirá un regulador de tensión a 5V que es la deseada, utilizando para ello el integrado L7805[3]. Se realiza la configuración de este integrado calculando las capacidades necesarias para su correcto funcionamiento según lo explicado en las hojas de especificaciones.





Para las resistencias, se establece un enganche fácil y seguro como los utilizados en el horno real, para dar un aspecto más robusto al diseño. El control del paso de alimentación a las resistencias, se hace a través de unos triacs[4] que soporten la máxima corriente que puede pasar (8A) y la tensión de red (220V). Estos necesitan unos optoacopladores[5] que controlan el paso por cero de la tensión de red y protege el sistema gracias al aislamiento óptico.

### **2.1.2. Interfaz de comunicación**

Para la comunicación con el usuario se emplea un LCD gráfico[6], pues el objetivo es mostrar la gráfica teórica y la real realizada. Para ellos se busca un componente acorde con el tamaño disponible en el horno seleccionado y que de una visión aceptable, siendo elegida una resolución de 128x64.

Referente a la interacción del usuario con el control se toma un teclado en matriz 4x4, para disponer de los 9 dígitos de numeración más los controles de dirección, aceptación y cancelación.

Como se ha indicado en los objetivos, se añade un puerto de comunicación serie, para poder conectar el producto final a un ordenador de forma que mediante una aplicación sencilla se puedan transferir o capturar datos del horno.

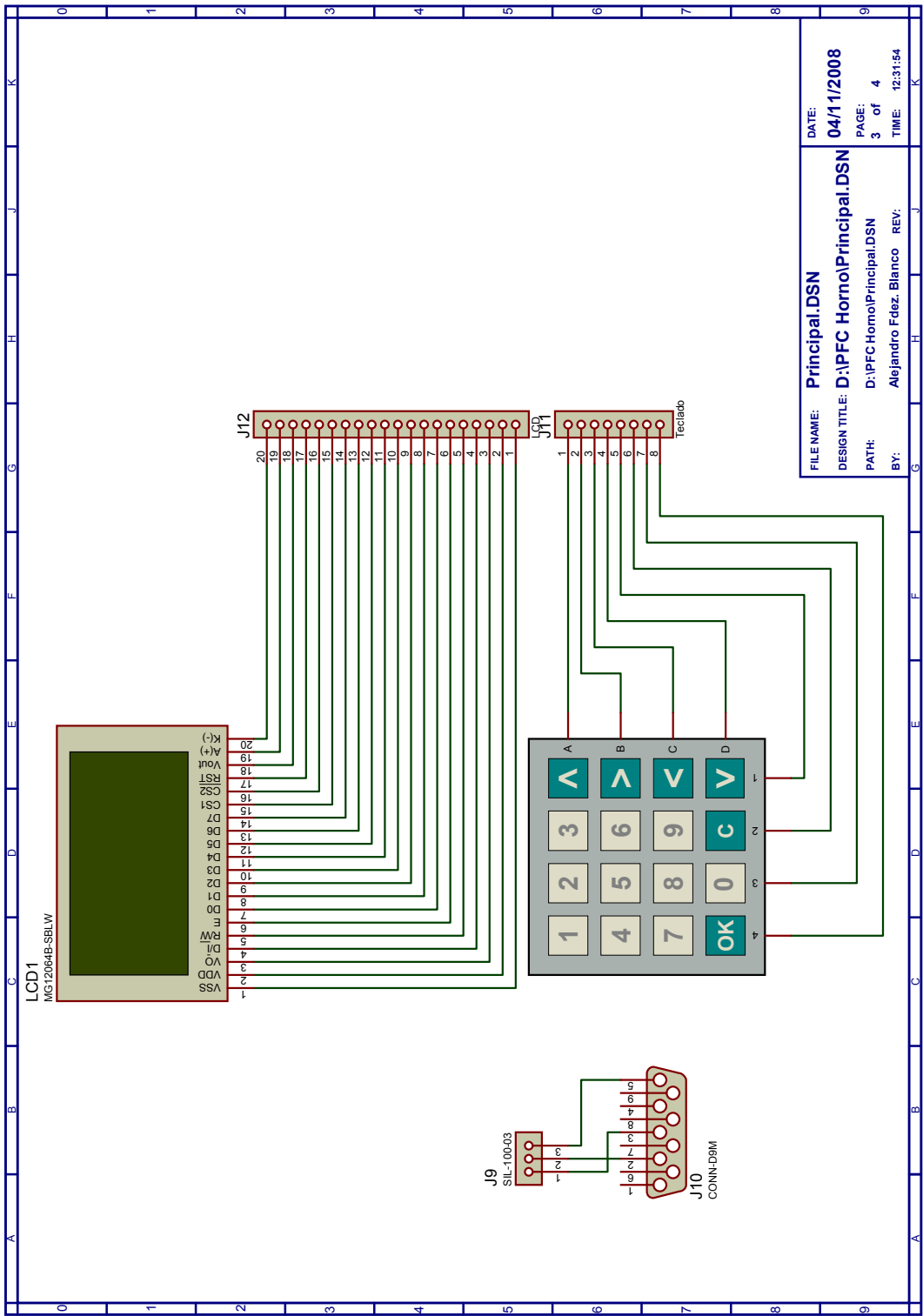


Figura 2.4: Esquema de la interfaz de comunicaciones

### 2.1.3. Bloque de control

En la parte de control será donde se tenga el microcontrolador. Estudiando todas las prestaciones deseadas, se establece el número de líneas que serán necesarias para el control de todo el sistema, eligiendo el microcontrolador PIC18F458[7], y configurándolo con las resistencias necesarias y el cristal deseado. En el caso de la frecuencia de reloj, se toma un cristal de 8MHz. Se observa en el esquema de la figura 2.5 que la patilla RA4 lleva una resistencia de pull-up debido a que no dispone de él internamente. Además como se desea un aviso sonoro para indicar finalización de proceso o apoyo a la pulsación, se añade un resonador.

Para la medida de temperatura se toman termopares tipo K debido a su bajo costo y que su rango de funcionamiento está dentro del rango de trabajo. Para el acondicionamiento de la señal, se añadirá el integrado MAX6675[8] que es un acondicionador de señales para termopares tipo K, que trabaja dentro del rango de trabajo, siendo este aproximadamente entre 0°C y 250°C. El protocolo de comunicación de este integrado es SPI (Serial Peripheral Interface), módulo que incluye el micro elegido.

Como se pretenden recoger las muestras y almacenarlas para poder transferirlas más tarde, se añade la memoria M25P16[9] que también utiliza el protocolo SPI como medio de transferencia con el micro. Esta memoria trabaja a una tensión de 3.3V por tanto se añade el regulador de tensión LD1086V33[10] para alimentar la memoria y se realizan divisores de tensión para sus patillas de entrada provenientes del micro. Ambos componentes no están disponibles en las librerías de PROTEUS, por lo que han tenido que ser diseñadas previamente.

El microcontrolador también dispone de un módulo de comunicación serie, pero se debe añadir el integrado MAX232[11] de comunicación, que acondiciona la transferencia del horno con el PC. Dentro de este esquema se tiene también el acondicionamiento de las líneas del LCD tal como indica el fabricante en la hoja de características, realizando los cálculos necesarios para obtener la corriente necesaria hacia el LCD. Se añade un potenciómetro que regule su contraste de forma que se pueda ajustar de forma adecuada.

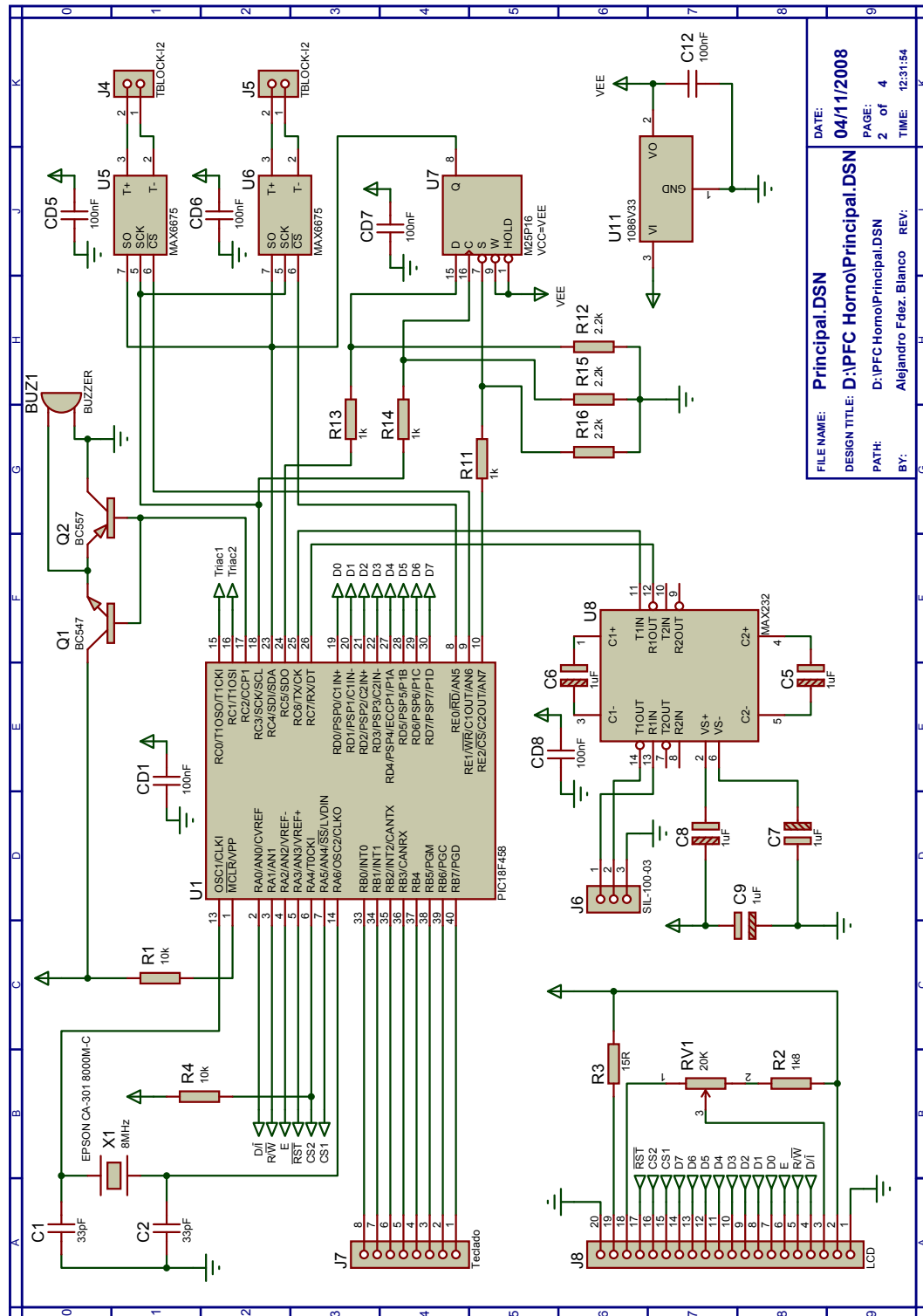


Figura 2.5: Esquema del circuito de control implementado

## 2.2. Fabricación de la PCB

### 2.2.1. Diseño de la PCB

Una vez realizado el diseño del sistema completo, se debe realizar el rutado del esquema donde irán colocados todos los componentes. Para ello se parte de la herramienta ARES que se distribuye junto con la herramienta PROTEUS. En ella primero se debe establecer cuales serán las dimensiones de las que dispondrá el diseño y donde irán colocados los taladros de sujeción de la placa, pues se deben amoldar a la estructura donde irá colocada la placa final.

Acto seguido se colocan todos los componentes de forma que se vayan situando de la manera más óptima posible, consiguiendo así un mejor rutado, intentando en todo momento que todas las pistas queden sobre la cara de soldadura. Se debe tener en cuenta que los componentes de montaje superficial se deben colocar en la cara de soldadura, así como los elementos que deben estar de ese mismo lado, debido a la situación final que tendrán la placa de control y los periféricos. Como el programa ARES no dispone de la huella de ciertos componentes, como son el transformador o el resonador, estas se diseñan con las medidas que los fabricantes dan de su producto en la hoja de datos, o mediante medida del componente.

Colocados los componentes, se pasa al rutado, que se realiza de forma manual para optimizar el grosor de aquellas pistas que serán de tensión alterna, y llevando todas las pistas de la forma más eficiente posible. Como se puede ver en la figura 2.6 no todas las pistas se consiguen por la cara de soldadura por lo que se pasan algunas por la de componentes, siendo estas las el menor número posible.

Finalizado este proceso se procede a la creación de la capa de mecanizado, que se utiliza para separar las pistas de tensión alterna, puesto que en otros proyectos se comprobó que con el método utilizado para realizar las placas, se producían arcos de tensión debido a la proximidad de las pistas de tensión de red. Por tanto se realiza una capa en el diseño que retira el cobre de aquellas zonas que forman parte de la tensión alterna, aislando aun más la parte de control de la parte de alimentación.

Una vez terminado el proceso de rutado y comprobado que se cumplen las reglas de diseño, se procede a la extracción de los ficheros tipo CAD/CAM para generar los archivos necesarios para la máquina fresadora disponible en el laboratorio de la universidad. Generados los archivos PLOT y DRILL necesarios, se pasa a realizar la placa física donde

irán situados los componentes.

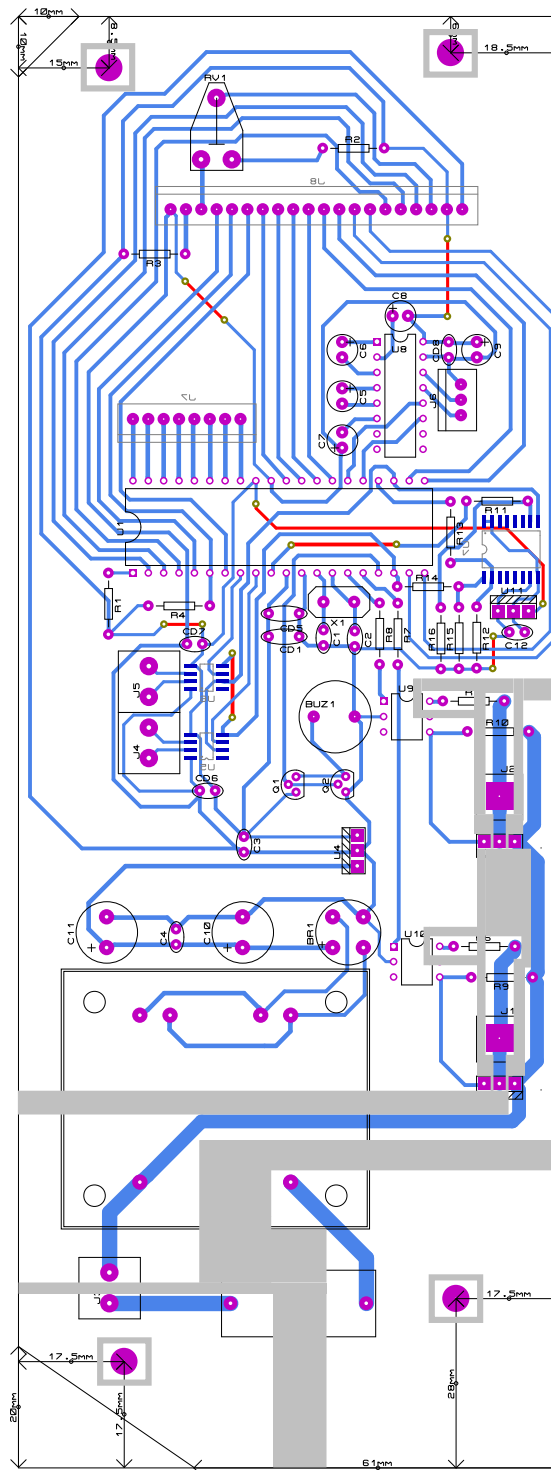


Figura 2.6: Esquema del rutado de las pistas sobre la PCB

### 2.2.2. Soldado y pruebas sobre la PCB

Cuando se tiene la placa fresada, se comprueba que no exista ningún cortocircuito en las pistas. Mediante impresión térmica con una plancha, se sitúa la serigrafía de los componentes en la cara correspondiente.

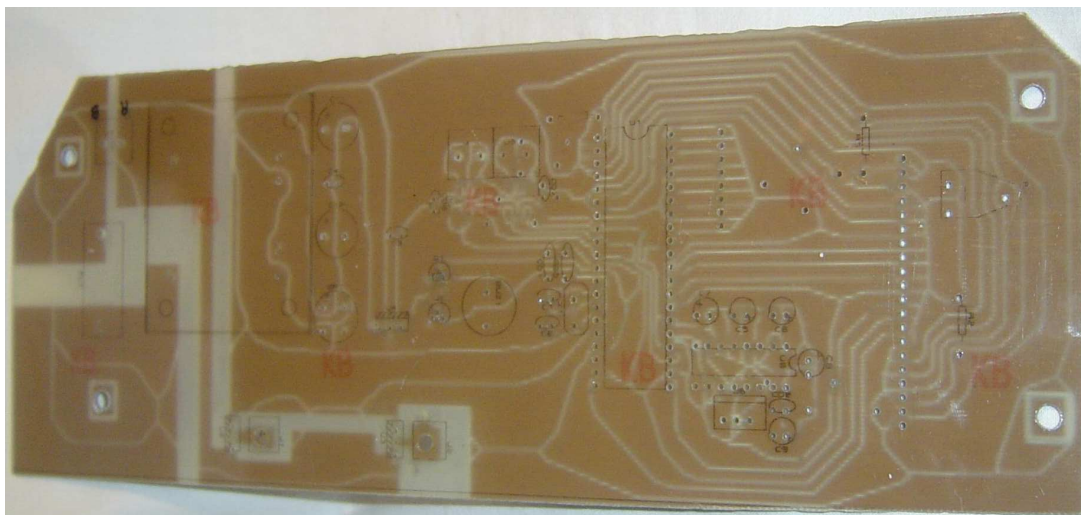


Figura 2.7: Placa de circuito impreso con la serigrafía

Se pasa después al montaje de los componentes empezando por la parte de tensión alterna, para comprobar que se obtiene a la salida de la alimentación los 5V requeridos y que no se producen arcos entre las pistas gracias a la capa de mecanizado. Se comprueba además que la corriente consumida no es desproporcionada, asegurando que no hay cortocircuitos en las pistas.

El siguiente paso es el soldado de todos los componentes que incluye el diseño, poniendo zócalos para aquellos componentes que puedan resultar dañados y tengan que ser reemplazados, como son el micro, el integrado MAX232 o los optoacopladores. El resultado final de la placa es el que se ve en la figura 2.8. Las dimensiones iniciales de la placa son las apropiadas para la introducción del diseño en el chasis del producto, pero debido a que se debe ajustar a las condiciones y contornos finales de los que se disponen, se realiza un lijado de la placa que modifica el contorno y con ello el aspecto final perdiendo parte de la forma rectangular.



Finalizado el proceso de creación del hardware, se comprueba que todo está situado y aislado para evitar derivas de tensión, se pasa a la fase de realización del software, que se explica en capítulos posteriores.

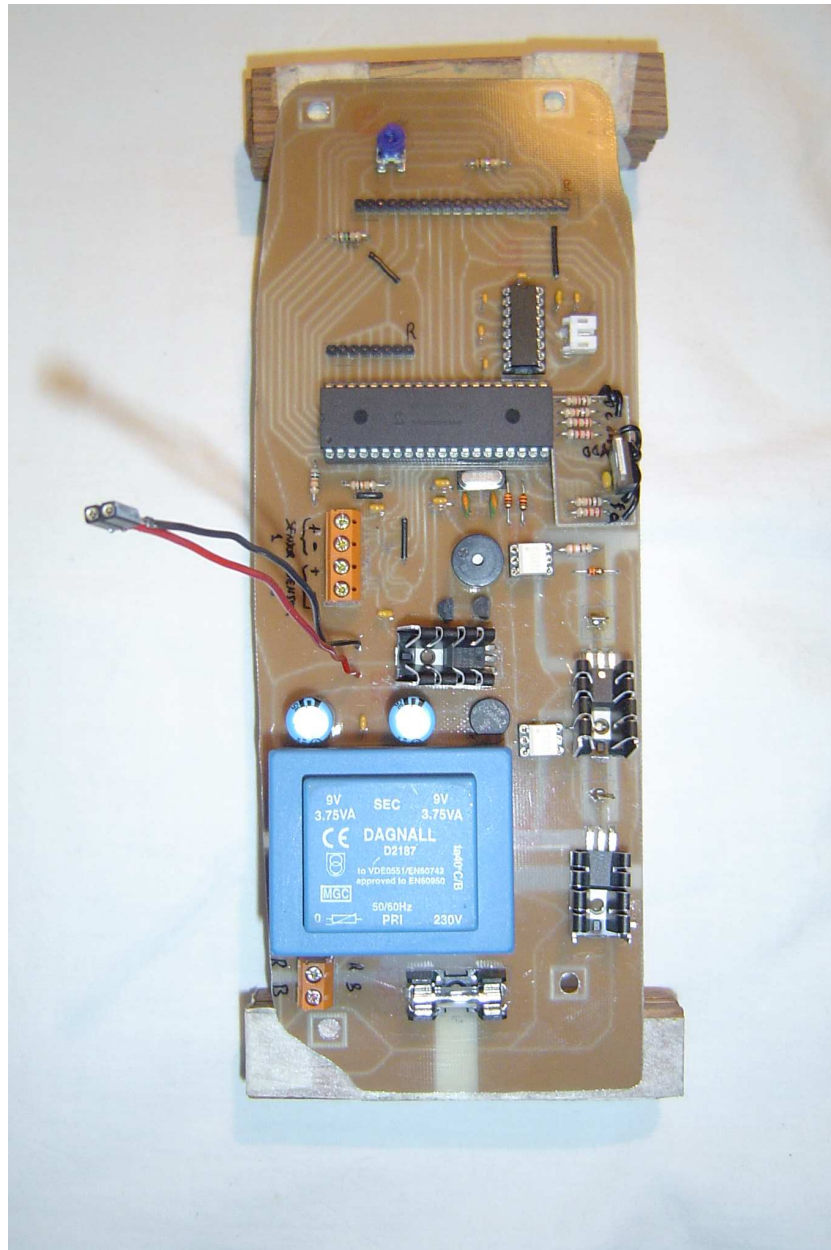


Figura 2.8: Placa de control final introducida en el horno

## 2.3. Montaje del horno

Cuando se tiene realizado todo el programa y comprobado que funciona correctamente, se procede a la preparación del chasis del horno para recibir la placa y poder cerrarlo de forma que se realicen las pruebas necesarias.

Lo primero que se realiza es la pieza que llevará sujeta toda la parte de interfaz. Partiendo de la pieza original, se realiza con productos resistentes a temperaturas elevadas las modificaciones suficientes para acoger la pantalla el teclado y el puerto serie. Para el recubrimiento se utiliza cemento de automóvil, que es un elemento pastoso, pero que su resultado final es una pieza sólida y resistente a temperaturas elevadas. Para darle uniformidad se pinta la pieza de color acorde con el chasis. También se pintan las teclas de control del teclado, para que tomen el aspecto del control que realizan, eliminando las letras que tienen por defecto y añadiendo la serigrafía de dirección, cancelar y aceptar.



Figura 2.9: Pieza frontal modificada

Realizado el frente se pasa a modificar el chasis para que acoja el nuevo control diseñado para el producto. Mediante la utilización de una máquina Dremel y su juego de herramientas, se taladra el lateral del horno donde se situarán los sensores. En el frontal del chasis se realiza la eliminación de material que obstaculiza la colocación del LCD, y en la parte posterior se abre una ventada que se utilizará para la colocación de



Figura 2.10: Teclado modificado para el horno

un interruptor de encendido y apagado del horno.

Los sensores se colocan en el interior del horno, separados del chasis una cierta distancia para que tomen la temperatura del interior del horno sin ser afectados en gran medida por la temperatura de la chapa. Para que tengan un aspecto robusto se le añade una silicona de alta temperatura. Este producto además se emplea para el sellado de pequeñas ranuras del horno, de forma que mejore la rampa de subida de temperatura.



Figura 2.11: Silicona de alta temperatura utilizada

Otra mejora de la pendiente de subida de la temperatura se realiza envolviendo el espacio dedicado al horno con una capa de aislante térmico, y además en las zonas donde no esta el control, se añade una capa de vitrofil (lana de vidrio) que ayude a la pendiente.

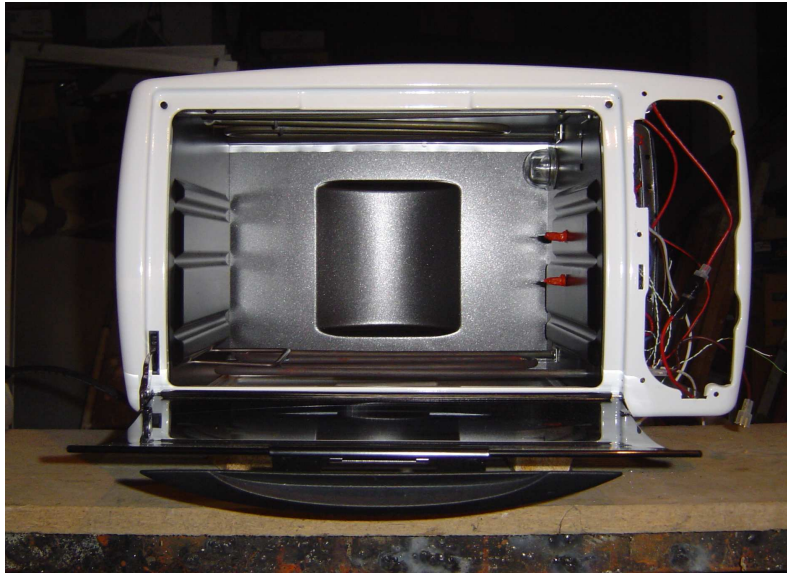


Figura 2.12: Chasis del horno modificado



Figura 2.13: Horno con el aislante colocado y el frontal montado

En una primera prueba con el control fuera de su emplazamiento se observa que en el espacio dedicado al control la temperatura se eleva pero manteniéndose dentro de un rango aceptable (inferior a  $80^{\circ}\text{C}$ ). Aun siendo aceptable, se añade en el diseño un ventilador, alimentado de la salida del puente rectificador de diodos y sujetado al chasis mediante la silicona de alta temperatura, para que esta zona se refresque más rápido y no sufran tanto los componentes del diseño.





Figura 2.14: Ventilador añadido para refrescar la zona de control

### 2.3.1. Resistencias

En las primeras pruebas se comprueba que no se alcanza la temperatura necesaria en el tiempo requerido por las curvas de soldadura, lo que hace plantearse la modificación de los elementos calefactores. Se procede por tanto a desmontar las resistencias y se comprueba que las resistencias son diferentes (1200w y 600w) cumpliendo las especificaciones de 1800w indicadas en las hojas de características del fabricante. Se adquiere una nueva resistencia de 1100w que sustituirá a la de 600w, y en el taller se procede a la adaptación de la resistencia al chasis del horno como se ve en la figura 2.16.

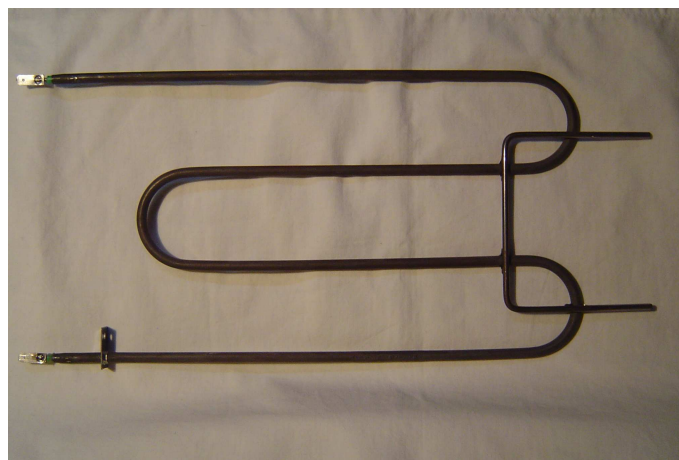


Figura 2.15: Resistencia de 600W retirada

El anclaje se realiza con varilla de hierro de 4mm mediante soldadura por arco y una vez adaptada la nueva resistencia se procede al montaje de nuevo de todo el horno, colocando las resistencias de 1100w y 1200w en la parte superior e inferior respectivamente. Con esto, la nueva potencia del aparato es de 2300w.



Figura 2.16: Adaptación de la nueva resistencia

El último paso consiste en cerrar el horno, obteniendo el producto final listo para realizar pruebas y comprobar el correcto funcionamiento.



Figura 2.17: Horno De'Longhi EO2131 modificado

Con el producto finalizado se instala en el laboratorio un puesto de trabajo que se compone de un PC con un cable puerto serie conectado al horno para poder hacer las actualizaciones de software según se vaya desarrollando.



Figura 2.18: Puesto de trabajo





# Capítulo 3

## Desarrollo Software

Para el desarrollo software se divide el problema en varias partes:

- El programa principal, que tendrá las funciones necesarias para leer el teclado, usar el puerto serie o guardar datos en memoria.
- El control de las resistencias mediante un algoritmo PID.
- La representación visual de los menús y las gráficas mediante el LCD.

Para poder desarrollar el programa se realiza la instalación del compilador PICC18 de HI-TECH necesario para la familia PIC18 de la cual forma parte el micro seleccionado.

### 3.1. Configuración del compilador PICC18

El primer paso es la configuración del compilador y el estudio de sus comandos para añadirlos al programa PROTEUS que servirá de soporte y de módulo de pruebas antes de pasar a grabar el micro. Además se busca utilizar un programa denominado bootloader que permite, una vez grabado éste en la memoria del micro, que se re programe tantas veces como se quiera el micro, sin sacarlo de la placa de control, mediante el puerto serie.

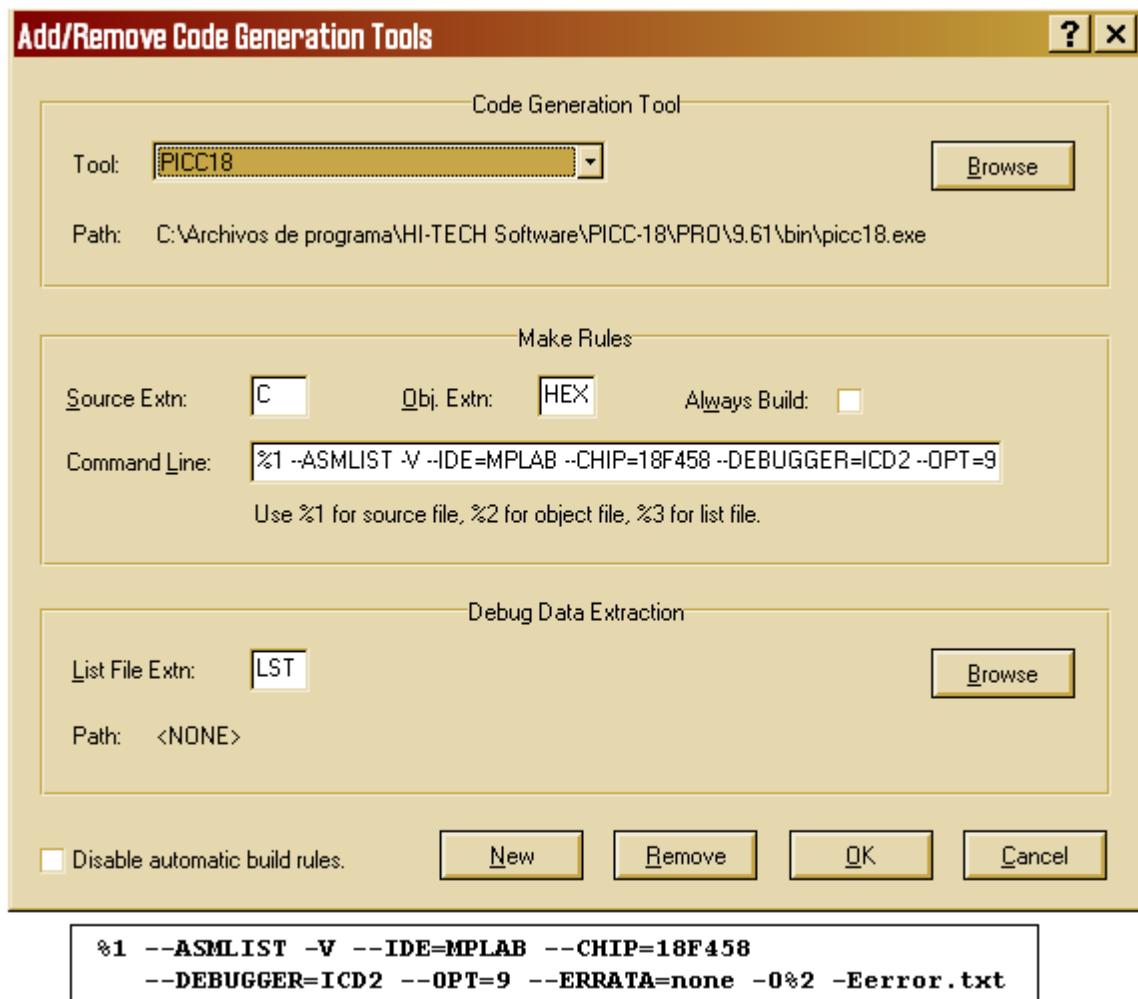


Figura 3.1: Ventana de configuración del compilador y línea utilizada

## 3.2. Bootloader

Partiendo de un pequeño proyecto desarrollado en la herramienta MPLAB, se tiene desarrollado un bootloader de reducidas dimensiones válido para el microcontrolador, de forma que, utilizando un programador serie tipo JMD y el programa ICPROG, se graba el micro con el bootloader. Esto permite que la actualización del firmware del micro se haga con este colocado siempre en la placa de control, sin tener que desmontarlo para programarle nuevamente. Con ello se consigue el objetivo de poder actualizar el firmware a través del puerto serie del que se dispone.

Este pequeño programa es grabado en la parte final del espacio de memoria del micro, de forma que cuando graba el programa principal lo hace desde la posición 4 de memoria. El programa saltará a esta dirección cada vez que no se utilice el bootloader para grabar,

haciendo que se ejecute el firmware.

El proyecto del bootloader dispone de una pequeña aplicación cuyo aspecto es el de la figura 3.2 que permite grabar rápidamente el dispositivo. Tiene dos velocidades de comunicación, de las cuales, la de 19.200 Baudios es la que se puede utilizar puesto que la velocidad de 115.200 Baudios genera demasiado error para la frecuencia del diseño.



Figura 3.2: Ventana del programa Bootloader

Antes de grabar el bootloader en el micro, se modifica el código ensamblador, para adaptar la velocidad de comunicación. Luego se compila el programa para generar el fichero objeto HEX y éste se graba en el micro. Seguidamente se introduce el micro en la placa de control y se ejecuta en el PC el programa del Bootloader. Se carga en este la ruta donde se encuentra el fichero objeto que contiene el programa principal de la aplicación desarrollada para el control, y se pulsa el botón de grabar (write flash) con lo que el programa enviará una palabra específica al micro. Éste deberá estar alimentado y al recibir la palabra de comunicación se iniciará el protocolo de comunicación y se grabará en la memoria flash de programa. Al finalizar el proceso, se arrancará la aplicación que se acaba de grabar.

Si se enciende el circuito y no se transmite nada por el puerto serie, se iniciará la aplicación que se encuentre en la memoria tras un cierto tiempo de espera. Si se pulsa el botón de grabar y no se conecta el micro, la aplicación enviará durante un tiempo la palabra de establecimiento de comunicación, tras el cual advertirá que no se ha encontrado el micro conectado y cesando el intento de conexión. Además dispone de una pequeña ventana en la cual muestra un informe de incidencias, indicando si se ha grabado correctamente el micro o ha ocurrido algún problema. Una vez grabado el Bootloader en el micro, se puede pasar a realizar el programa principal que será el desarrollado para el horno.

### 3.3. Programa principal

Para el control se tiene el micro PIC18F458 de Microchip que dispone de 40 patillas distribuidas en un encapsulado tipo DIL40 y módulos internos que ayudarán a un desarrollo más eficaz de la aplicación final que se desea tener.

Primeramente se necesita la configuración de los registros de control del micro para que durante la ejecución del programa todo funcione correctamente. El micro dispone de cinco puertos, los cuales se configuran en función del diseño hardware realizado, haciendo que sean patillas de entrada o salidas, o directamente añadidas a los módulos que se pretenden utilizar. Para todo ello se desarrolla la función **Configuracion** que contiene la inicialización de todos los registros para el micro.

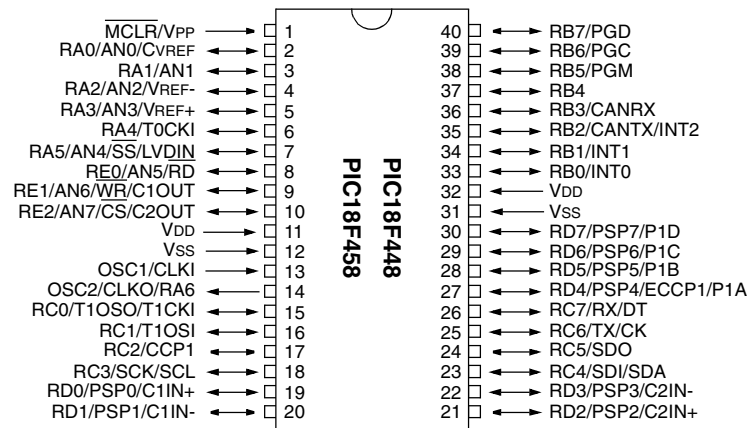


Figura 3.3: Distribución de las patillas del PIC18F458

- El puerto A se encarga del control del LCD.
- El puerto B realiza la revisión del teclado indicando la tecla pulsada.
- El puerto C se emplea para controlar las resistencias, el resonador, los sensores y la memoria.
- El puerto D es el de datos del LCD.
- El puerto E al disponer solo de 3 patillas, se utilizan para la selección de los distintos dispositivos SPI.

**TABLE 1-1: PIC18FXX8 DEVICE FEATURES**

Features		PIC18F248	PIC18F258	PIC18F448	PIC18F458
Operating Frequency		DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Internal Program Memory	Bytes	16K	32K	16K	32K
	# of Single-Word Instructions	8192	16384	8192	16384
Data Memory (Bytes)		768	1536	768	1536
Data EEPROM Memory (Bytes)		256	256	256	256
Interrupt Sources		17	17	21	21
I/O Ports		Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers		4	4	4	4
Capture/Compare/PWM Modules		1	1	1	1
Enhanced Capture/Compare/PWM Modules		—	—	1	1
Serial Communications		MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART
Parallel Communications (PSP)		No	No	Yes	Yes
10-bit Analog-to-Digital Converter		5 input channels	5 input channels	8 input channels	8 input channels
Analog Comparators		No	No	2	2
Analog Comparators VREF Output		N/A	N/A	Yes	Yes
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low-Voltage Detect		Yes	Yes	Yes	Yes
Programmable Brown-out Reset		Yes	Yes	Yes	Yes
CAN Module		Yes	Yes	Yes	Yes
In-Circuit Serial Programming™ (ICSP™)		Yes	Yes	Yes	Yes
Instruction Set		75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages		28-pin SPDIP 28-pin SOIC	28-pin SPDIP 28-pin SOIC	40-pin PDIP 44-pin PLCC 44-pin TQFP	40-pin PDIP 44-pin PLCC 44-pin TQFP

Figura 3.4: Prestaciones y características del micro utilizado

### 3.3.1. Teclado

Para el teclado se desarrolla una función dedicada exclusivamente a su evaluación. Además se configura el temporizador TMR1 para controlar el tiempo de las pulsaciones. Todas las funciones desarrolladas se pueden ver en el apéndice A. El escaneo en busca de nuevas pulsaciones, se realiza dentro de la función del Teclado y su funcionalidad se puede ver en su diagrama de flujo.

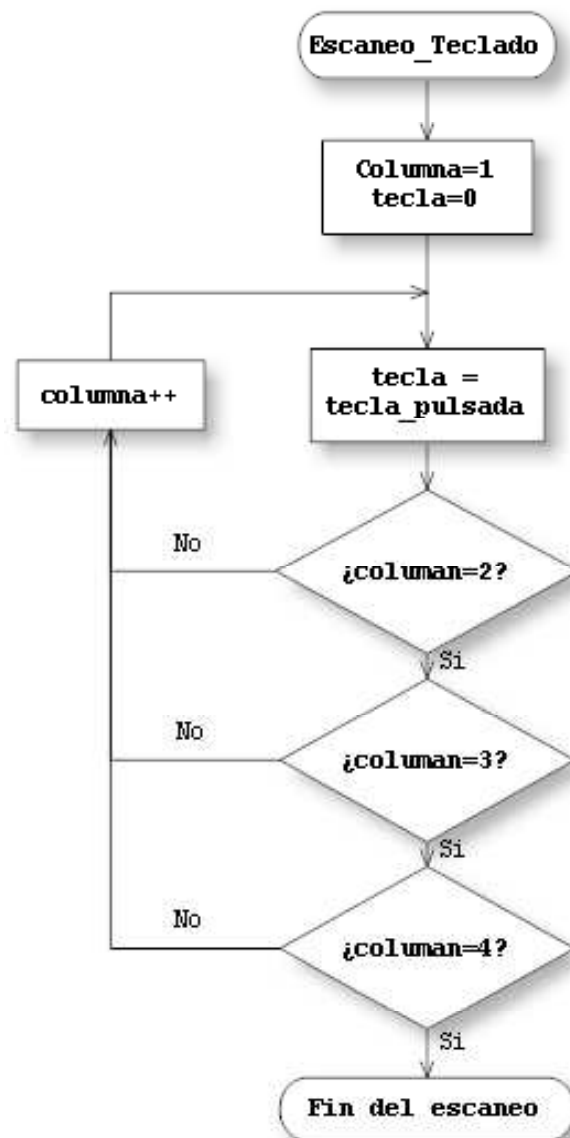


Figura 3.5: Diagrama de flujo del escaneo del Teclado

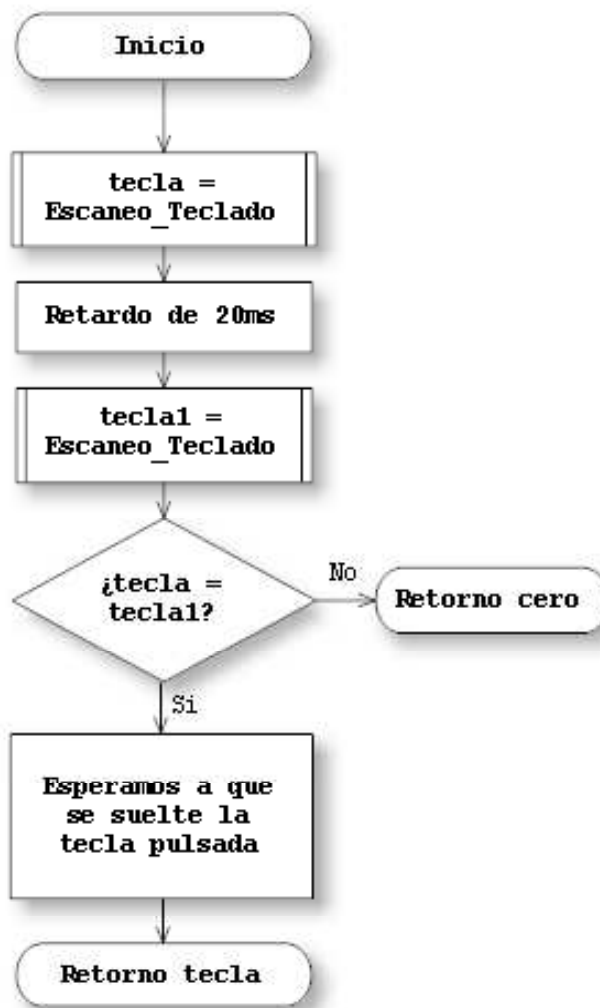


Figura 3.6: Diagrama de flujo de la función Teclado

Una vez se recoge la tecla pulsada, se configura el módulo PWM (Pulse-Width Modulation) del micro, para generar una señal de pulsos modulados, que a la frecuencia de 2.400Hz irá hacia el resonador de forma que genere un pitido. Los cálculos que se deben realizar para obtener la frecuencia de resonancia del dispositivo son las proporcionadas por el fabricante del micro. La señal modulada se controla mediante una función que recibe un número que define las repeticiones del ciclo del PWM. Con esta función se hace que haya una advertencia sonora cada vez que se pulse una tecla, se finalice el proceso o el arranque del sistema.

**EQUATION 15-1:**

$$\text{PWM Period} = [(\text{PR2}) + 1] \cdot 4 \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})$$

**EQUATION 15-2:**

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON}<5:4>) \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})$$

**EQUATION 15-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{\text{FOSC}}{\text{FPWM}}\right)}{\log(2)} \text{ bits}$$

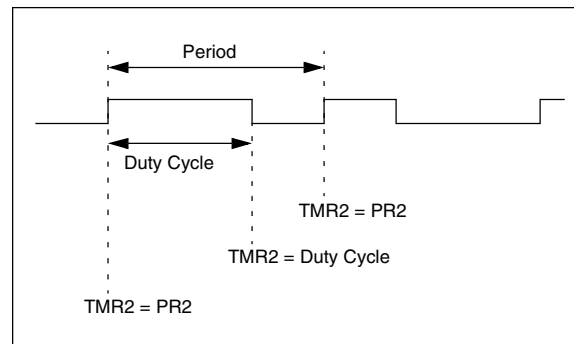
**FIGURE 15-4: PWM OUTPUT**

Figura 3.7: Ecuaciones utilizadas para calcular las constantes del PWM

**3.3.2. Puerto Serie****EXAMPLE 18-1: CALCULATING BAUD RATE ERROR**

Desired Baud Rate	=	$\text{FOSC}/(64 (X + 1))$
Solving for X:		
	X =	$((\text{FOSC}/\text{Desired Baud Rate})/64) - 1$
	X =	$((16000000/9600)/64) - 1$
	X =	$[25.042] = 25$
Calculated Baud Rate	=	$16000000/(64 (25 + 1))$
	=	9615
Error	=	$\frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}}$
	=	$(9615 - 9600)/9600$
	=	0.16%

Figura 3.8: Ecuaciones utilizadas para calcular los parámetros de velocidad

La comunicación con el puerto serie se realiza a través del módulo USART. Este se configura en modo asíncrono y se calculan el valor del registro SPBRG que guardará la relación de generación de los Baudios con la ecuación proporcionada en las hojas de car-



acterísticas del micro. Como se quiere disponer de más de una velocidad, se toma la más alta que se puede tener con una relación de error inferior al 1 %, la cual es 38.400 Baudios. Las constantes de configuración se definen en el archivo cabecera que se puede consultar en el apéndice B.

Para la captura de los datos en el PC, se dispone de la herramienta Hyperterminal con el que se captura el texto enviado desde el horno en un archivo. Para enviar datos al horno, primero se debe generar un archivo que contenga los datos en el formato correcto para que los reciba el horno sin problemas. La forma de utilizar esta herramienta se explica en manual de usuario desarrollado para el producto.

Las funciones que trabajan con el puerto serie son las siguientes y se han definido en base a los diagramas de flujo:

- La función **RS232\_Recibo** toma los datos del PC, para una nueva curva.

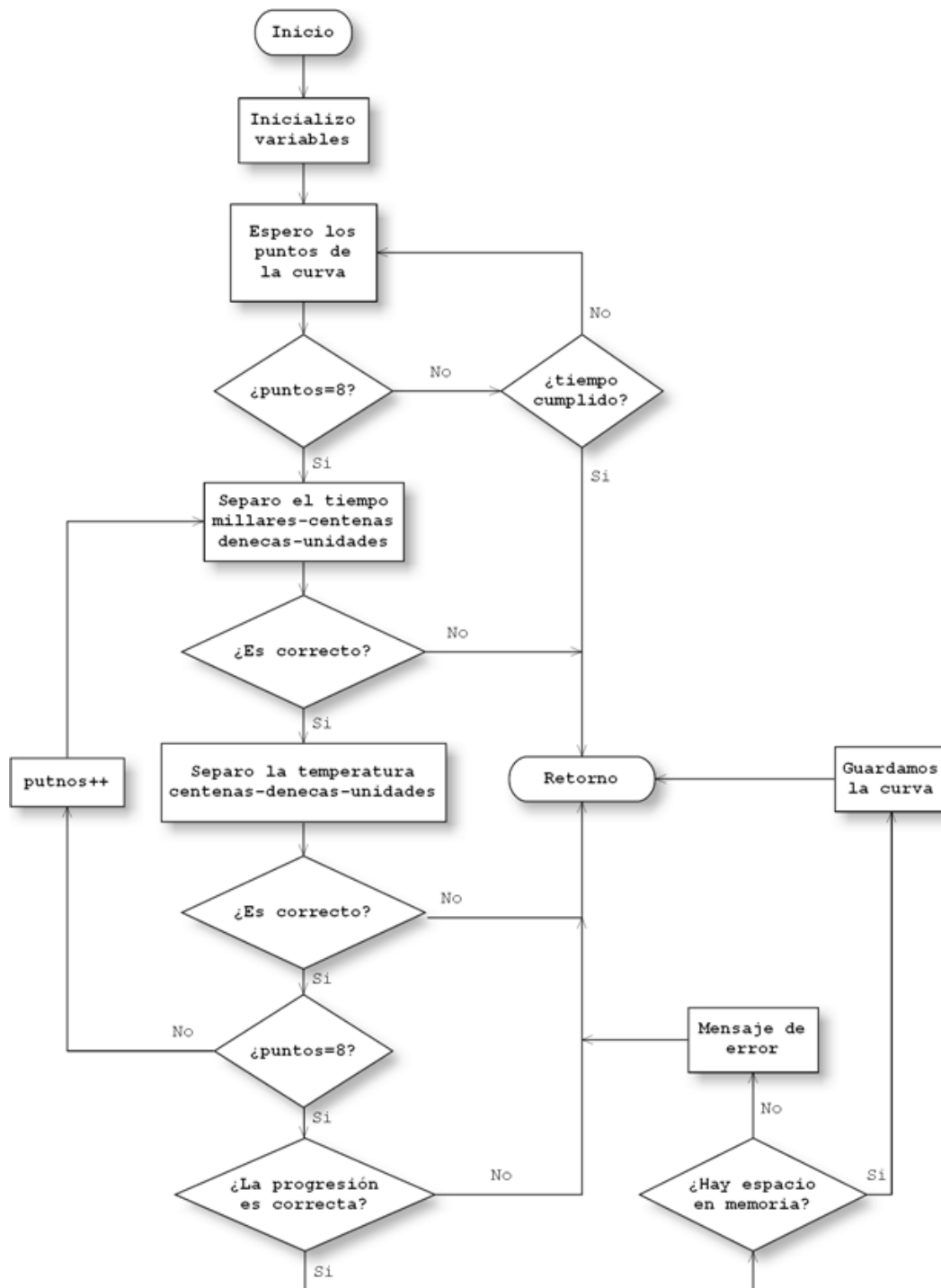


Figura 3.9: Ecuaciones utilizadas para calcular los parámetros de velocidad

- La función **Lectura Puerto** genera el formato de los datos a enviar.

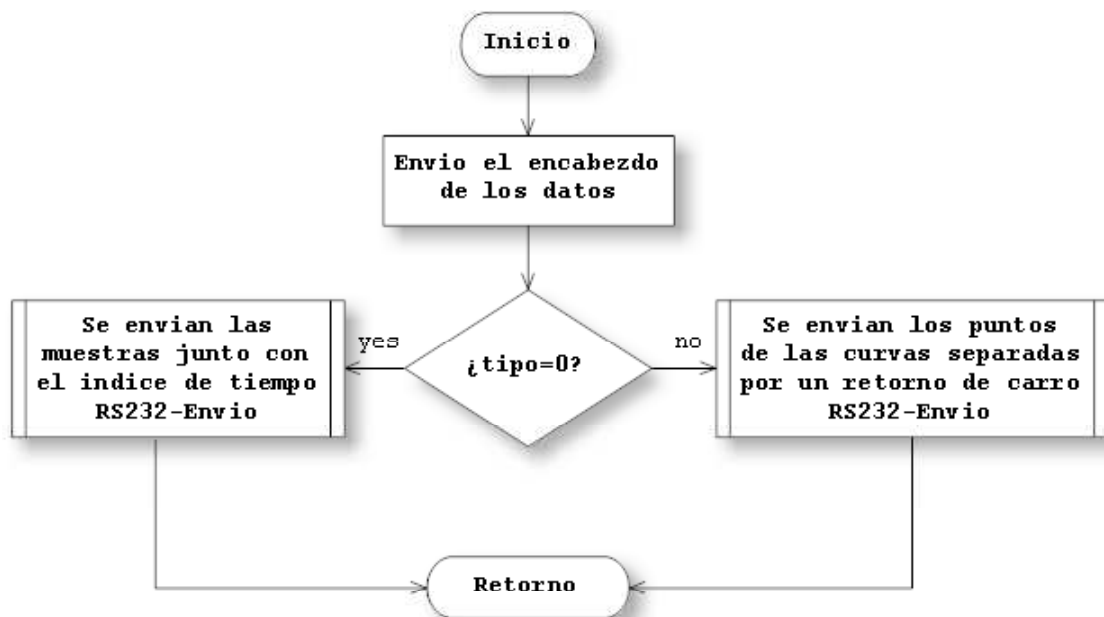


Figura 3.10: Ecuaciones utilizadas para calcular los parámetros de velocidad

- La función **RS232\_Envio** se encarga de enviar los datos por el puerto serie.

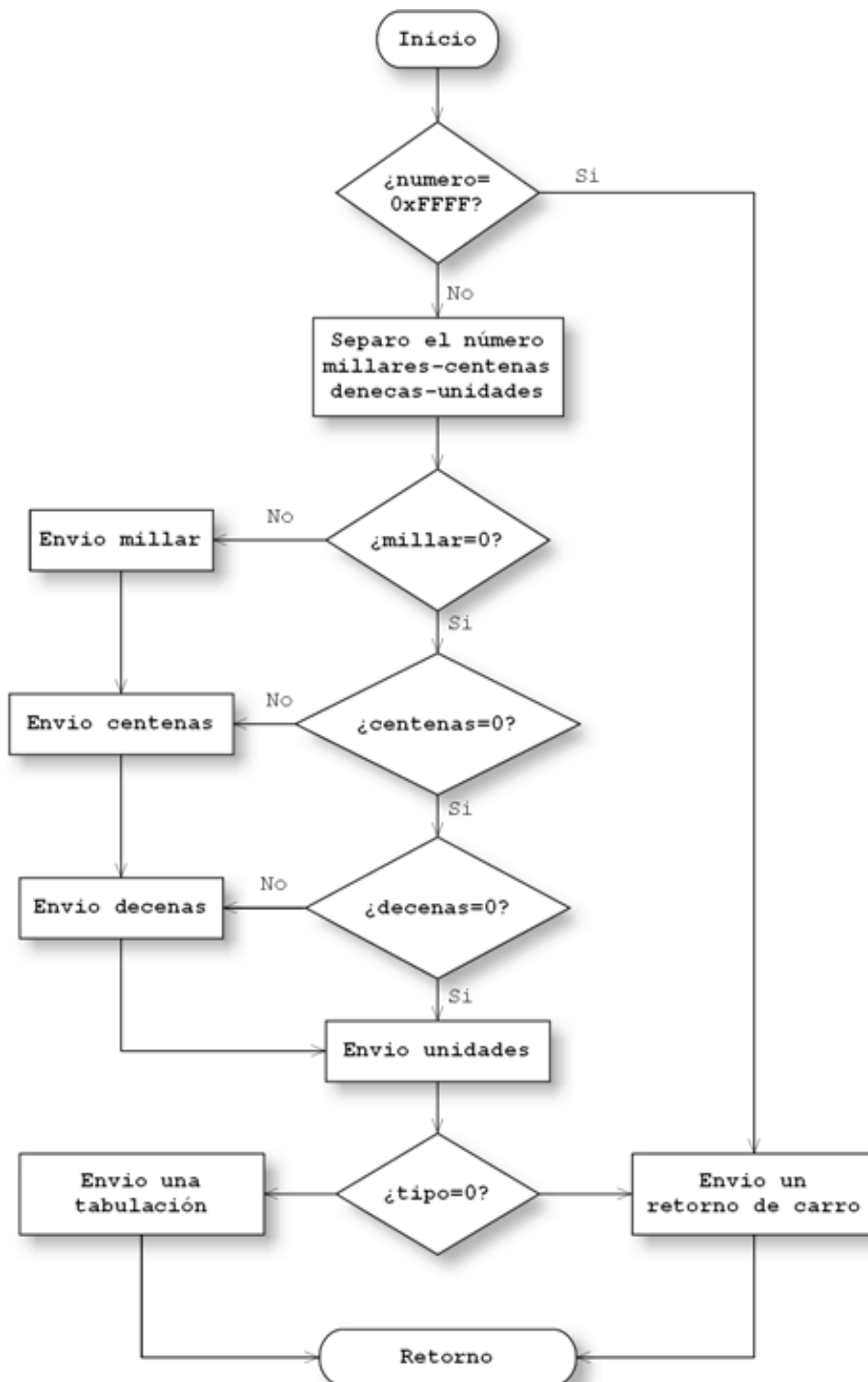


Figura 3.11: Ecuaciones utilizadas para calcular los parámetros de velocidad

### 3.3.3. Lectura de Temperatura

El control de la temperatura se hace mediante el módulo MSSP (MASTER SYNCHRONOUS SERIAL PORT) configurado en modo SPI. Es un protocolo de comunicación serie y depende del micro toda la comunicación, siendo este el responsable de la señal de reloj de sincronismo. La figura 3.12 muestra la conexión típica en este tipo de protocolo. Los integrados MAX6675 envían la información de la temperatura leída en 16 bits, divididos en dos bloques de 8 bits. De las cuatro configuraciones típicas que existen en el protocolo SPI, se toma la que corresponde a los integrados, donde se leen los 16 bits en el flanco de bajada del reloj. Por tanto se muestrea la señal en el flanco de bajada del reloj tomando como estado ocioso de la señal el cero.

**FIGURE 17-2: SPI™ MASTER/S�AVE CONNECTION**

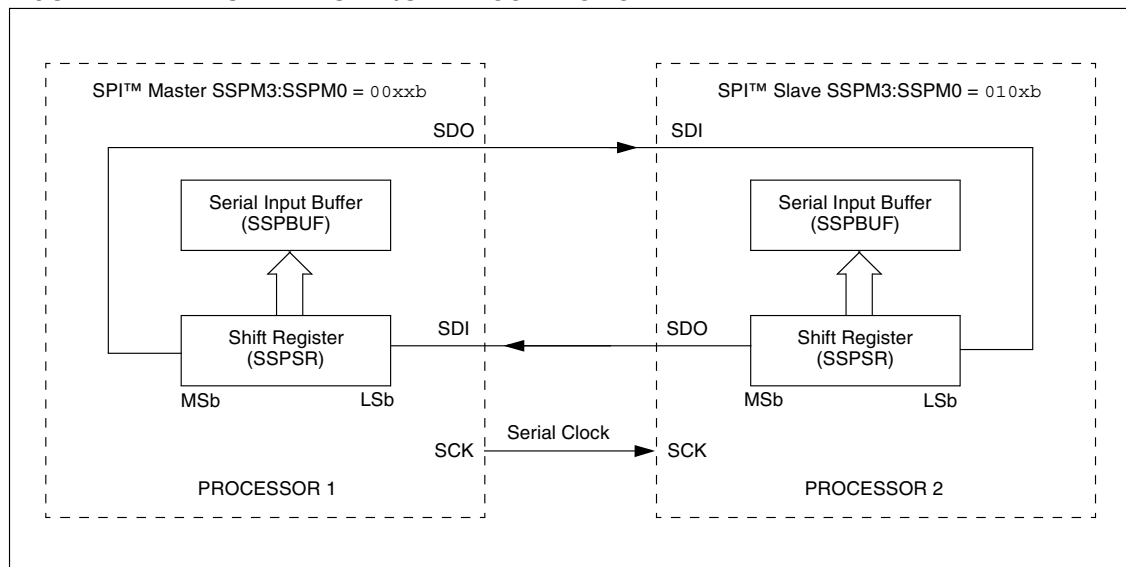


Figura 3.12: Conexión típica de una comunicación SPI

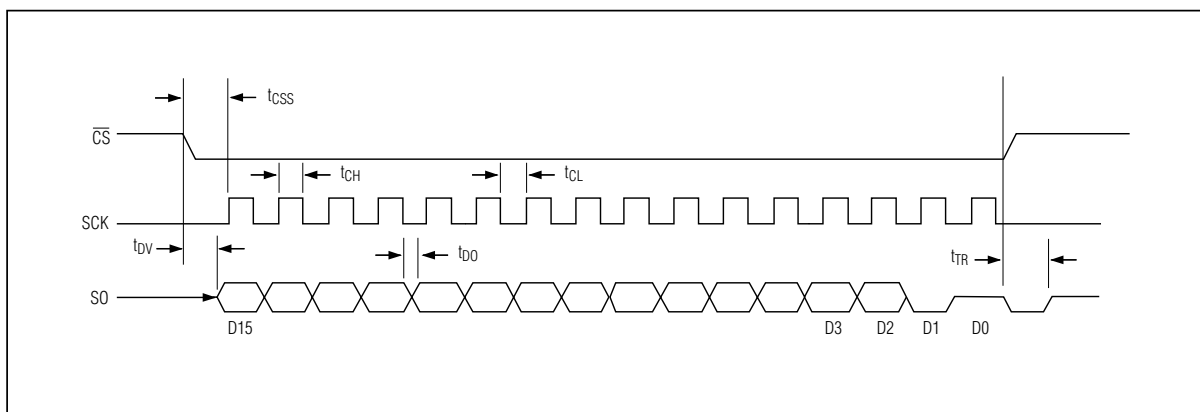


Figure 1b. Serial Interface Timing

BIT	DUMMY SIGN BIT	12-BIT TEMPERATURE READING												THERMOCOUPLE INPUT	DEVICE ID	STATE
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	MSB											LSB		0	Three- state

Figure 2. SO Output

Figura 3.13: Modelo de SPI que utiliza el chip MAX6675

Con el módulo configurado correctamente, solo es necesario seleccionar el chip correspondiente y este enviará la lectura que hace en ese momento. Las funciones encargadas de esta tarea son la de Temperatura1 y Temperatura2, que responden a la misma estructura pero seleccionando sensores diferentes. Los pasos que desarrollan son:

- Selecciona el sensor correspondiente.
- Se reciben los primeros 8 bits de medida.
- Se reciben los últimos 8 bits de medida, se concatenan en un registro de 16 bits, y se desplaza para quedarnos con el valor real de la medida.
- Se deselecta el sensor para dejar las líneas de comunicación libre para otro proceso SPI.

El integrado MAX6675 nos proporciona una precisión de 12 bits, de los cuales los dos menos significativos son decimales de temperatura, por lo que no se emplean debido a que el proceso desarrollado con precisión de 1 grado es suficiente.

### 3.3.4. Memoria externa, M25P16

Como la memoria también es SPI, se realizan sus funciones para almacenar los datos. Para su correcto funcionamiento se debe configurar nuevamente el módulo con el modo correcto de reloj, tal como indica el fabricante en las hojas de características. Como es una memoria con suficiente capacidad, se dispondrá de memoria extra útil para posteriores mejoras. La memoria M25P16 está dividida en 31 bloques de 256 MBytes, por lo que se ha establecido la siguiente configuración de memoria.

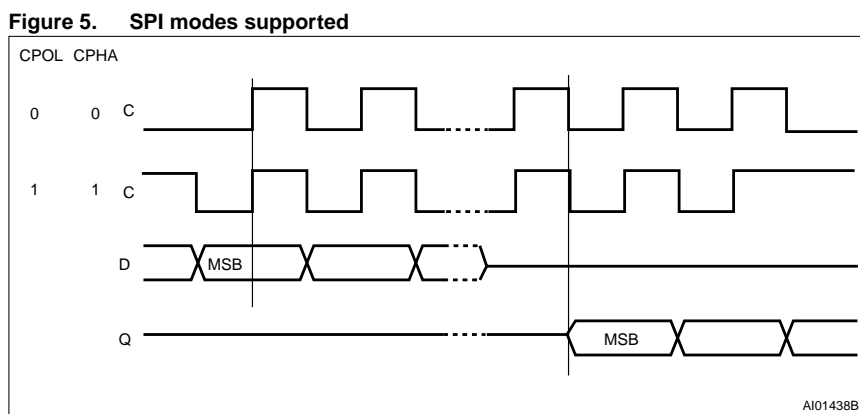


Figura 3.14: Modelo de SPI que utiliza la memoria M25P16

La configuración del horno requiere dos parámetros que son las resistencias que se van a utilizar y la velocidad de comunicación del puerto serie, por lo que se memorizan en la memoria EEPROM interna del micro. Esto confiere mayor rapidez a la hora de acceder o modificar los parámetros de configuración.

El uso de la memoria M25P16 se divide en unas funciones básicas y tan sólo difieren en las del proceso que se quiere realizar. Se tiene por tanto:

- Primero se da la dirección de memoria a la que se va acceder. Se difiere entre el proceso de lectura y el de escritura, pues se le debe pasar junto con la dirección, la instrucción propia de la memoria indicando qué se va a realizar.
- La segunda función, varía en según la instrucción. Si es una lectura se recibe un dato, se debe diferenciar si es una muestra que ocupa 1 byte (puesto que las muestras no pueden superar los 250°C), o es un punto de la curva que ocupa 2 bytes. Si la instrucción es de escritura, también diferencia entre muestra o punto de curva, pues se guarda una sola posición o dos, en función del dato a guardar.
- Finalmente se tiene una función que es común a los dos tipos de instrucciones (lecturas o escrituras) que es la de finalización de uso de memoria, cuya función es

Table 3. Memory organization

	Sector	Address range	
Espacio libre	31	1F0000h	1FFFFFFh
	30	1E0000h	1EFFFFFFh
	29	1D0000h	1DFFFFFFh
	28	1C0000h	1CFFFFFFh
	27	1B0000h	1BFFFFFFh
	26	1A0000h	1AFFFFFFh
	25	190000h	19FFFFFFh
	24	180000h	18FFFFFFh
	23	170000h	17FFFFFFh
	22	160000h	16FFFFFFh
	21	150000h	15FFFFFFh
	20	140000h	14FFFFFFh
	19	130000h	13FFFFFFh
	18	120000h	12FFFFFFh
	17	110000h	11FFFFFFh
	16	100000h	10FFFFFFh
	15	0F0000h	0FFFFFFh
	14	0E0000h	0EFFFFFFh
	13	0D0000h	0DFFFFFFh
	12	0C0000h	0CFFFFFFh
	11	0B0000h	0BFFFFFFh
	10	0A0000h	0AFFFFFFh
	9	090000h	09FFFFFFh
	8	080000h	08FFFFFFh
	7	070000h	07FFFFFFh
	6	060000h	06FFFFFFh
	5	050000h	05FFFFFFh
	4	040000h	04FFFFFFh
	3	030000h	03FFFFFFh
Sector para muestras	2	020000h	02FFFFFFh
Sector para copia	1	010000h	01FFFFFFh
Sector De Curvas	0	000000h	00FFFFFFh

Figura 3.15: Modelo de SPI que utiliza la memoria M25P16

deseleccionar la memoria de forma que las líneas de comunicación SPI quedan libres para otro dispositivo.

También está la función que nos proporciona el borrado de un sector completo, que se utiliza para borrar el sector de las muestras cada vez que se inicia un nuevo proceso, de forma que se guarden los datos del nuevo proceso. Debido a la estructura interna de la memoria, esta función es necesaria para poder borrar las curvas guardadas o modificar



alguna. La memoria sólo dispone de borrado completo de ella o por sectores, lo que hace que para modificar una curva, se deba reemplazar todo el sector, y para borrar dejando un espacio libre, se deban borrar primero todas. Por tanto se crea una función que recibe un parámetro el cual nos indica si se borra una curva o se modifica, simplificando el proceso. Los pasos seguidos son:

- Primero se copian todas las curvas excepto la que va a ser modificada o borrada en el espacio de memoria, manteniendo el espacio donde ésta estaba sin escribir.
- Después se borra el sector principal desde donde se leen las curvas existentes.
- Acto seguido se comprueba si es una modificación o se debe borrar la curva. Si se debe borrar, se copian al sector principal todas las curvas eliminando el espacio vacío que se dejó cuando se copiaron, obteniendo así el borrado de la curva seleccionada.  
Si es una modificación, cuando se ha modificado la curva, se ha guardado en su posición correspondiente del sector de copia, por lo que el espacio no está libre y será introducida la curva modificada junto con las demás.
- Para finalizar el proceso de esta función lo que se hace es borrar el sector de copia, dejándolo preparado para próximos usos.

### 3.3.5. Otras funciones

Otra función necesaria para el desarrollo del programa principal es una función que proporcione tiempos de espera determinados. Estos se consiguen con la configuración del temporizador TMR1. Como ya se ha indicado, éste queda configurado para controlar las esperas del teclado y evitar rebotes, pero definiendo el número de cuenta que se quiere realizar en función del tiempo que se quiere controlar, se crea una función que nos proporciona un tiempo de espera mediante el número de repeticiones del bucle, que multiplicado por el mínimo tiempo que se contabiliza, da el tiempo total de espera. Esta función es necesaria para controlar los tiempos de acceso a la memoria M25P16 debido a que requiere un tiempo de ejecución interna, y mientras esté ocupada no puede recibir otro tipo de instrucciones.

Una vez definidas todas las funciones generales del programa básico, se tiene un sistema capaz de recibir y enviar datos, leer del teclado advirtiéndole de su pulsación de modo audible y medir la temperatura. El siguiente paso será poder controlar la temperatura interna del horno.

### 3.4. Control PID

Para el control de las resistencias se desea un algoritmo que ajuste lo más posible la potencia suministrada a las resistencias a la curva de temperatura que se desea realizar. Por ello se decide realizar un control PID el cual se basa en la obtención de tres parámetros que controlen la temperatura. Como no se dispone de modelo concreto de la planta a controlar, en este caso las resistencias, se parte del ejemplo de creación de un PID del libro *Compilador C CCS y Simulador PROTEUS* [12] donde indica cual es la planta que se puede suponer y como se deben realizar los cálculos para el control.

Un **PID** (*Proporcional Integral Derivativo*) es un mecanismo de control por realimentación que se divide en tres parte:

- La parte **Proporcional** consiste en el producto entre la señal de error y la constante proporcional como para que hagan que el error en estado estacionario sea casi nulo.
- El modo **Integral** tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el modo proporcional.
- La acción **Derivativa** se manifiesta cuando hay un cambio en el valor absoluto del error, si el error es constante, solamente actúan los modos proporcional e integral.

El resultado del control es la suma de las tres partes aplicada sobre un actuador que en el ejemplo de partida es una señal PWM para que module el ancho de sus pulsos consiguiendo una transferencia de potencia necesaria para llevar el error (diferencia entre la temperatura teórica que se quiere y la real) a cero. Como no se dispone de más módulos PWM en el micro se debería realizar la señal modula por vía software, lo que supondría mantener en un bucle al micro controlando la señal, sin poder realizar otras tareas. La solución vendría por la realización de un **RTOS** (*Real Time Operating System*) un Sistema en tiempo real que mantuviera el tiempo de muestreo intacto, mientras el resto de los sistemas actuarán. Debido a que el compilador utilizado no dispone de esta prestación y generar el código del RTOS es muy costoso, se desestima la creación del Sistema en Tiempo Real.

El control más sencillo para procesos térmicos que no necesitan mucha precisión es el sistema On-Off, que se basa en mantener los elementos térmicos encendidos hasta rebasar la temperatura fijada apagando en ese momento los elementos calefactores. Pero uno de los objetivos es tener un control más sofisticado y preciso evitando los sobrepicos que se producen en el método On-Off, por lo que se mantiene un control PID, pero aplicado

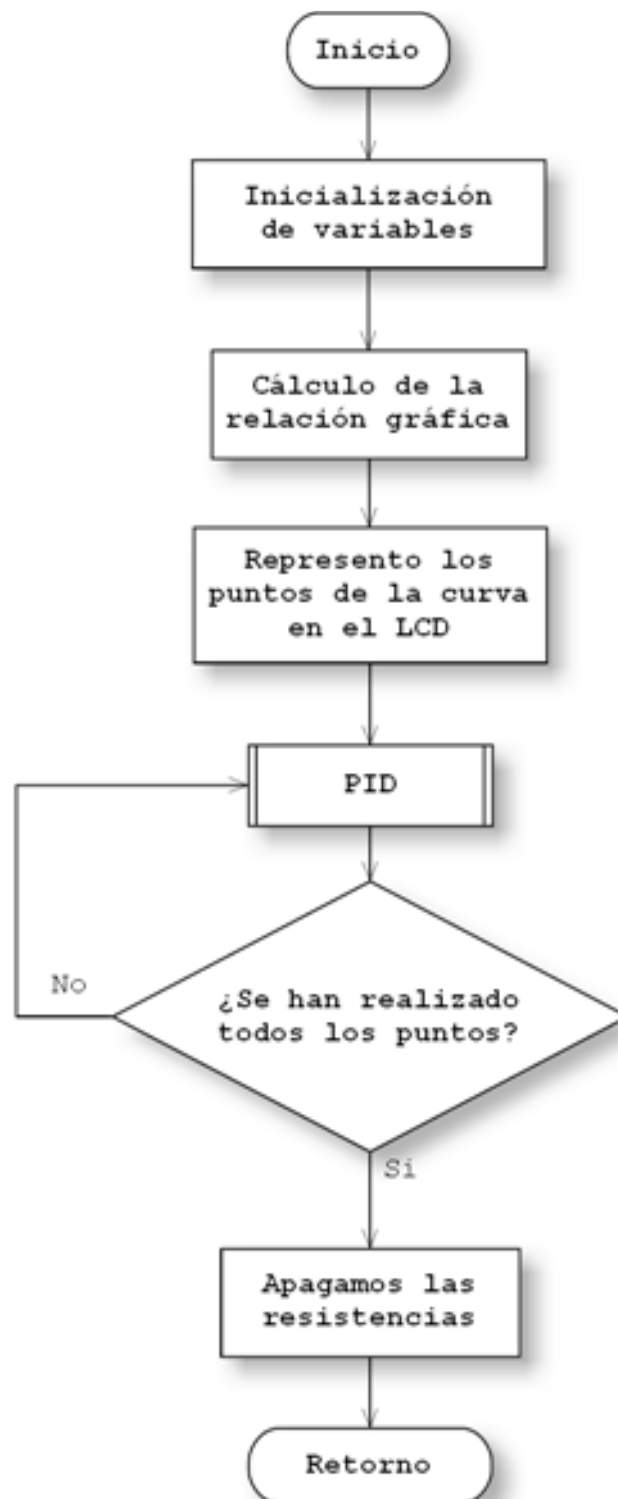


Figura 3.16: Diagrama de flujo la función Control

sobre un control On-Off. El control PID requiere tres parámetros, por lo que se procede según indica el ejemplo de referencia para su obtención. El primer problema que surge

es la obtención de una señal alterna de entre el 10 % y 20 % del valor nominal de red (220V). Los equipos generadores de los laboratorios no producen señales de 10V de alterna, con lo cual se dispone de un transformador de 220V-125V (el 57 % del valor nominal aproximadamente) con el que se realiza la prueba, pero la curva de máxima pendiente es muy lenta y tras las operaciones, se obtienen unos valores que, ejecutando una prueba del control, produce un error mucho mayor del 10 %, con lo cual se desestima el método.

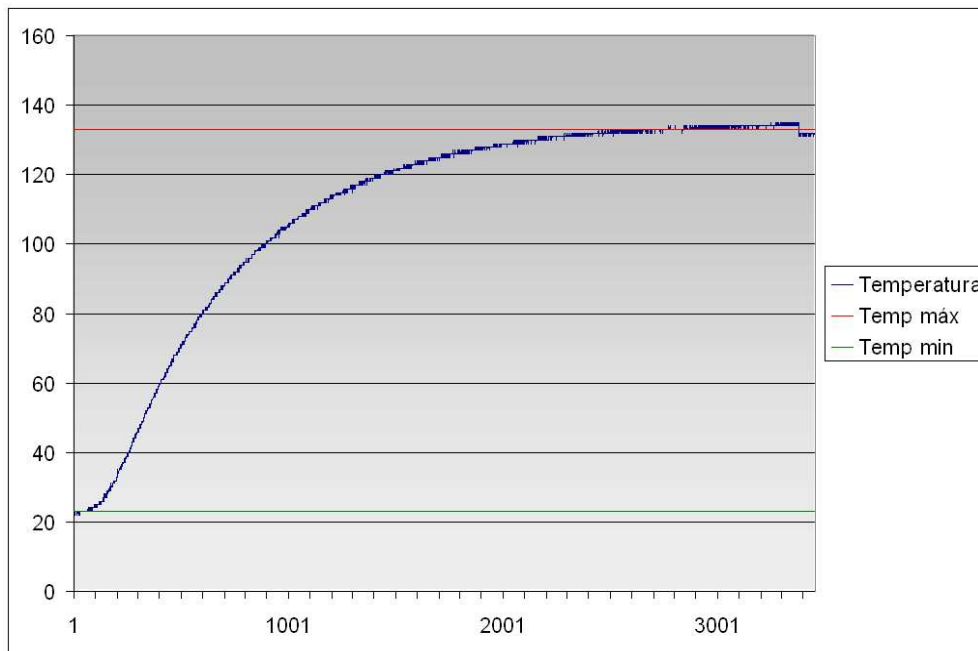


Figura 3.17: Curva de máxima pendiente obtenida para el modelado de las resistencias

Debido a que el control requiere los tres parámetros, se procede a la obtención de ellos de forma manual (método de ensayo y error) Para ello se ajustan las variables I (Integral) y D (Derivativo) a cero y se va ajustando la señal P (Proporcional) de forma que el sistema llegue a oscilar. Una vez en ese punto, se ajusta P a la mitad del valor obtenido y se incrementa D para que se adapte el control a los tiempos requeridos, sin aumentarlo mucho, debido a que puede producir inestabilidad. Finalmente se incrementa I si es necesario para que se reduzca al máximo el error estacionario y no haya un gran sobrepico.

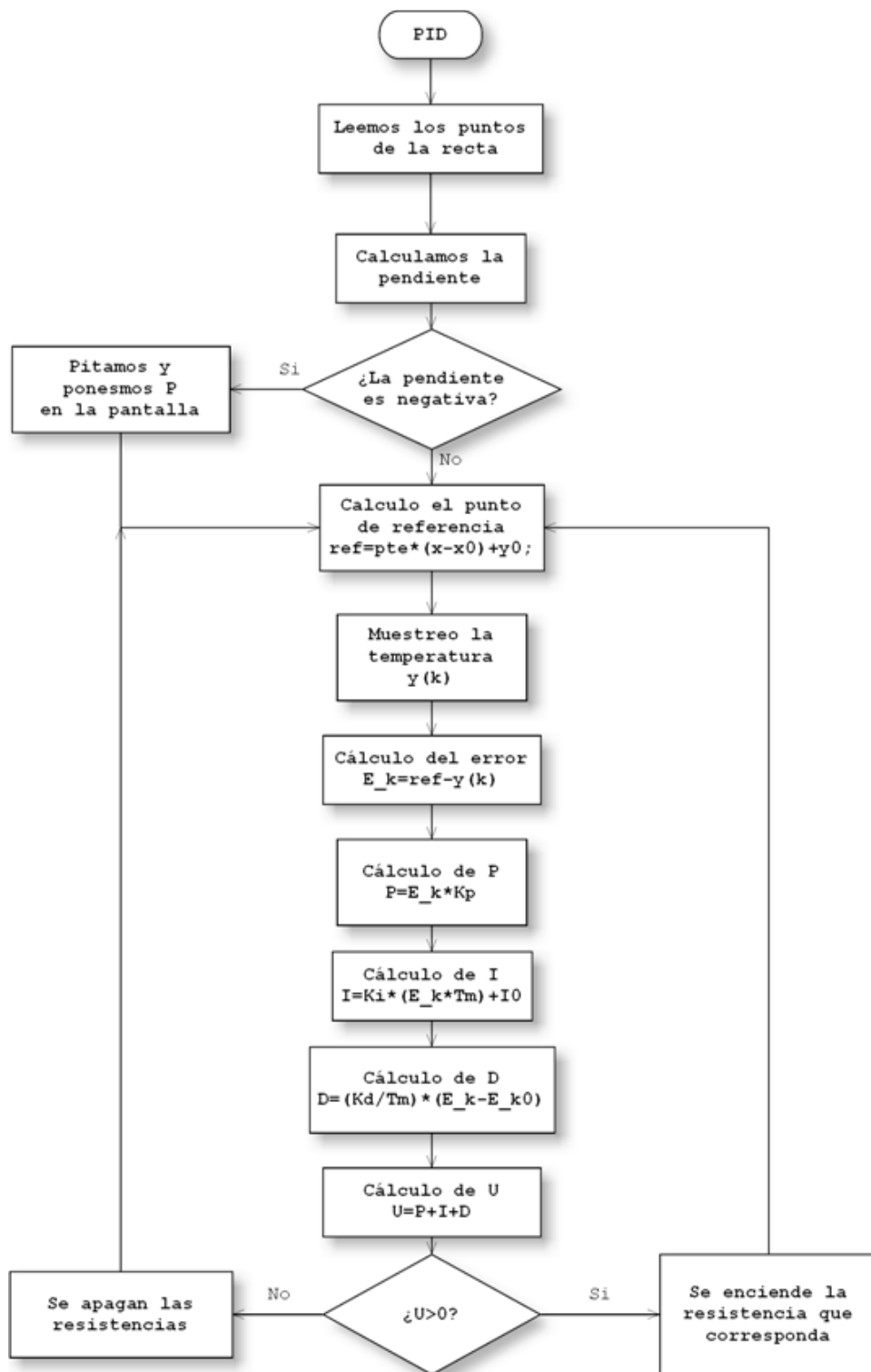


Figura 3.18: Diagrama de flujo del control PID

Después de realizar este método, se obtienen las curvas que se pueden ver en la figura 3.19 en la cual se aprecia que una curva realizada sólo con la resistencia superior es mucho

más lenta que con las dos resistencias o incluso con la inferior sólo. Cabe destacar que el aire caliente al ser menos denso tiende a subir, por lo cual es más lento el calentamiento de todo el volumen del horno por la resistencia superior.

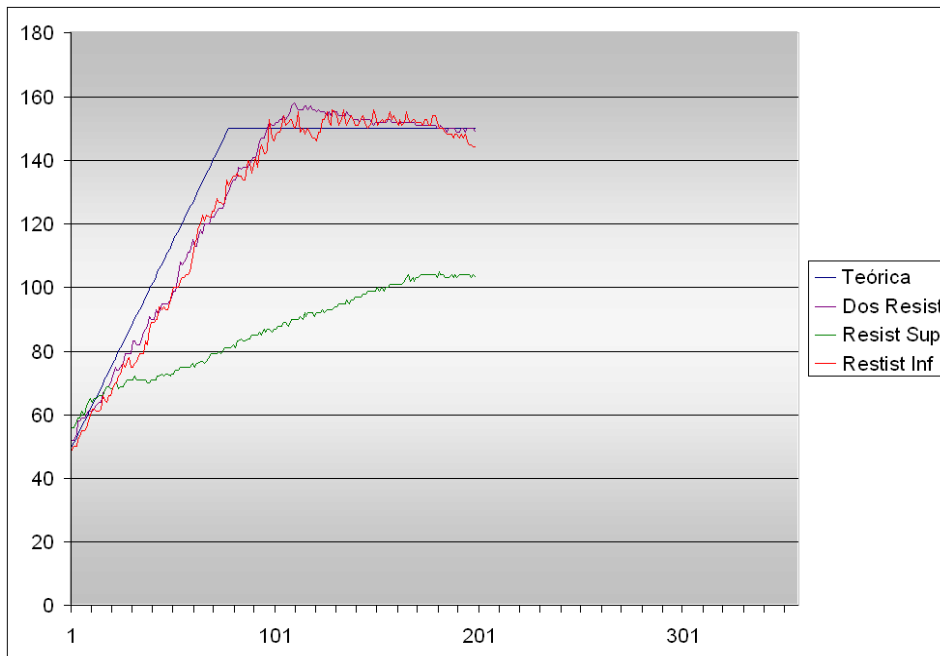


Figura 3.19: Curvas reales que más se aproximan a la curva teórica

### 3.5. LCD Gráfico

Se requiere que el horno muestre las curvas de temperatura que el usuario almacena en memoria y la curva que se realiza durante el proceso, por ello se trabaja con un LCD gráfico para el cual, se realizan diferentes funciones.

Para el correcto funcionamiento del LCD se requiere que sea habilitado y deshabilitado cada vez que se le pase un dato o una instrucción, lo cual requiere de unos tiempos de retardo entre la habilitación y la recepción de la instrucción o dato. Por ello se genera una función de retardo encargada de realizar los tiempos de espera necesarios para lectura y escritura. En la figura 3.17 se pueden apreciar los cronogramas necesarios para leer y escribir en el LCD. Para poder pintar en éste, se le debe comunicar primero cual es la posición de la RAM a la que se quiere acceder, mediante los punteros de fila y columna. Para cada uno de ellos se crea una función la cual sitúa los punteros de forma que sólo tengamos que indicar el dato a representar.

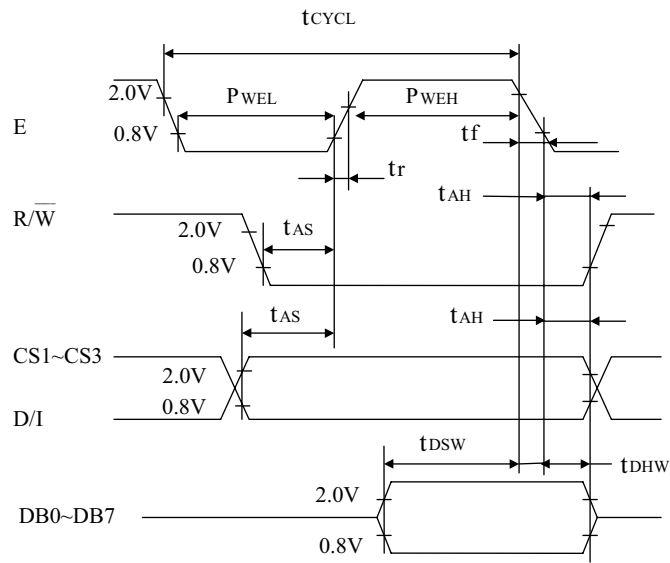


Figure 1 CPU Write Timing

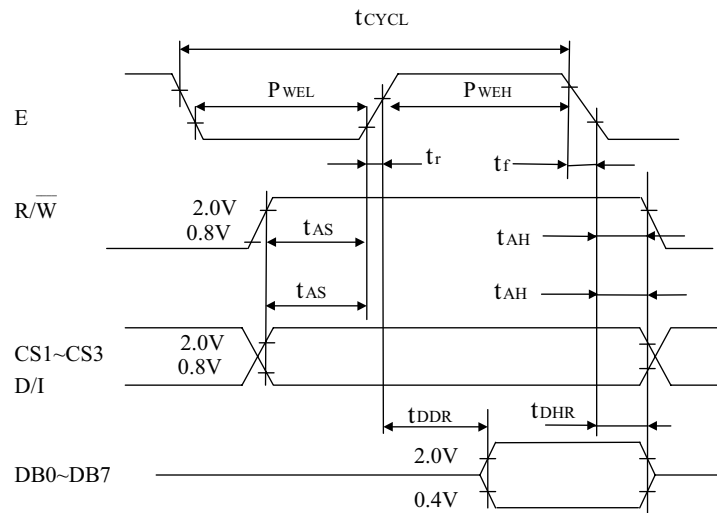


Figure 2 CPU Read Timing

Figura 3.20: Cronogramas de escritura y lectura del LCD

Los pasos seguidos para la utilización del LCD son:

- Se indica la página en la cual se quiere situar.
- Se indica la columna en la que se empezará.
- Con los punteros colocados, se puede leer o escribir.

Lo primero que se hace al iniciar el programa es inicializar el LCD poniendo todos los bytes de la RAM a cero. Para evitar que el usuario vea como se modifican los valores de la RAM, se tienen dos funciones que permiten mostrar el contenido de la memoria del LCD, o que no lo muestra.

Puesto que se quiere representar, además de píxeles, caracteres para generar diferentes menús de pantalla se necesita definir primero los caracteres que se emplearán, por ello se genera el fichero **font.h**, que contiene los bytes de cada letra, que serán de 8 bytes. La introducción de una cadena de caracteres se realiza mediante la función **LCD\_Caracter** que añadirá un carácter allí donde se le haya indicado. Definidas las funciones básicas, se pasa a generar todas las funciones complejas que harán del LCD una funcionalidad completa.

Para imprimir en pantalla cadenas de caracteres se crea la función **LCD\_Escribir** que recibe una cadena de caracteres y los imprime allí donde se hayan situado los punteros, de esta forma se crean los diferentes menús con cadenas de caracteres predefinidos en el archivo **mani.h** y sólo se debe hacer una llamada a la función de escribir para que imprima por pantalla los menús.

Otra función necesaria es aquella que permita pintar líneas en el LCD de forma que la curva teórica que se desea realizar pueda mostrarse durante su edición. En base a esto se crea la función **LCD\_Linea** que realiza el algoritmo de Bersenham, pintando una línea que discurre entre los dos píxeles que se le indican. Con ello, se puede crear una tabla gráfica donde se aprecien las curvas almacenadas en memoria indicadas con su tiempo y temperatura máximos, de forma que pueda pasar a editarse la curva, o a borrarse. Para realizar esta lectura de la memoria se crea la función **LCD\_Curvas** que muestra los datos de máxima temperatura y tiempo en el LCD.

En la edición de curvas, se tiene la función **LCD\_Edicion** que muestra la curva que se está editando permitiendo modificar puntos existentes. Si se entra en un espacio libre de memoria que no contiene ninguna gráfica, se creará una nueva gráfica y se guardará siempre y cuando se acepte la curva para realizar, aunque luego no se ejecute. Las curvas también se pueden borrar como ya se ha indicado anteriormente. Tanto la función para editar como para borrar las curvas, son accedidas mediante los cursores derecho e izquierdo respectivamente.



## **3.6. Programa Final**

Una vez desarrolladas las diferentes funciones de las que se compone el programa principal, se generan los diferentes menús de forma que el programa trabaje de la forma deseada. Para ello se establece un diagrama de flujo de forma que se pueda seguir el desarrollo de todo el código de una manera más clara. El diagrama de flujo muestra las trazas de la ejecución del programa básico, mostrando solamente el avance sobre la pulsación de la tecla aceptar.

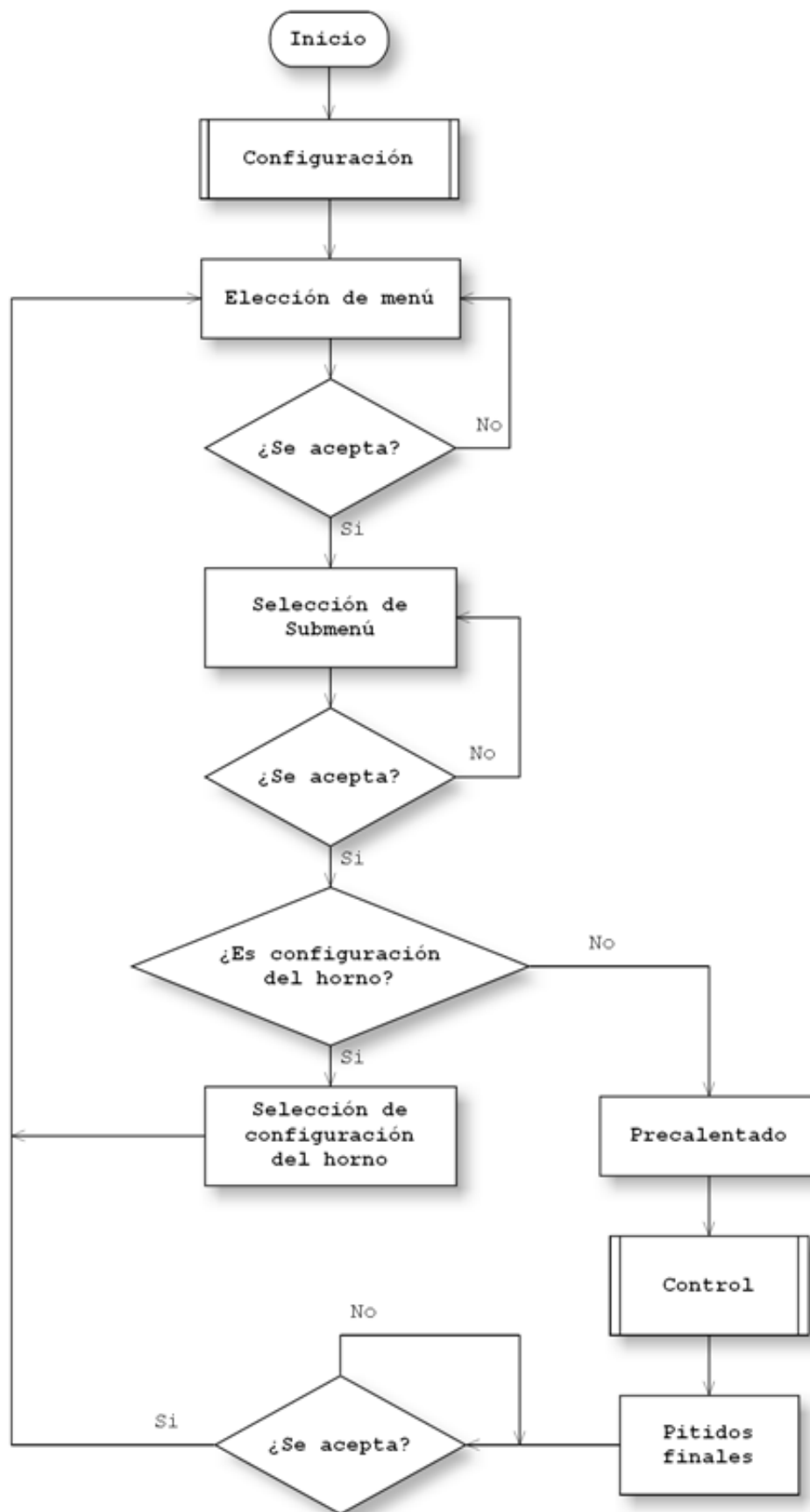


Figura 3.21: Diagrama de flujo del programa final

# Capítulo 4

## Manual de usuario

Este manual explica como realizar un uso correcto del producto conseguido. Tanto el diseño como la estructura de las diversas pantallas es simple, pues se distribuye en el encabezado, el cuerpo y la línea de los comandos útiles en cada menú.

### 4.1. Arranque del horno SMD

Coloque el horno sobre una superficie lisa al menos a unos 80cm sobre el nivel del suelo, y en un lugar amplio y ventilado, ya que los gases que pueden desprender las pastas utilizadas pueden ser tóxicos. Enchufe el horno a una tensión de 220V y sobre una línea de al menos 16A de corriente.

En la parte posterior del horno se encuentra el interruptor de encendido y apagado. Cuando esté iluminado significará que el horno está siendo alimentado y se encenderá la bombilla interna así como la pantalla. Para apagar el horno vuelva a pulsar el interruptor de la parte posterior. Por seguridad siempre desenchufe el horno si no va a proseguir con su uso.

El horno lleva un fusible interno para evitar temperaturas superiores a 300°C pero para su seguridad la temperatura máxima está limitada a 260°C. El control del horno dispone de otro fusible de 0.5A por seguridad.



Figura 4.1: Ventana de inicio que muestra el programa

Al arrancar el programa del horno se mostrará la pantalla de bienvenida y una señal acústica indicará su encendido. Durante este proceso de arranque, no será operativo el teclado.

Tras el arranque se muestra el menú principal donde se tienen cuatro opciones. Presione uno de los botones de dirección (arriba o abajo) para desplazar el cursor hacia el menú que desea abrir y presione el botón de aceptar para entrar.



Figura 4.2: Menú principal del Horno SMD

En el centro del encabezado de la pantalla se muestra el menú en el que se encuentra el programa. En la parte inferior derecha se encuentra la temperatura actual del horno junto con los comandos.

## 4.2. Menú de Configuración



Figura 4.3: Menú de configuración del Horno SMD

Dentro de este menú se selecciona con los cursores de dirección una de las dos posibilidades ofrecidas, que son las resistencias a utilizar y la velocidad de comunicación con el PC.

### 4.2.1. Submenú de resistencias



Figura 4.4: Submenú de selección de las resistencias

Para las resistencias se tiene la elección de trabajar con las dos resistencias, sólo con la superior (tipo grill) o sólo con la inferior.

### 4.2.2. Submenú de velocidad de comunicación

La velocidad de comunicación con el PC se selecciona entre tres estándares de los que dispone, que deberá ser el que se seleccione en la aplicación utilizada en el PC.



Figura 4.5: Submenú de selección de la velocidad de comunicación

Con los cursores de dirección arriba y abajo se desplaza el cursor a la opción deseada de velocidad o resistencia. Si se pulsa cancelar se vuelve al menú principal y no se modifica la configuración. Si se pulsa aceptar, se guardará la nueva configuración siendo esta la que quede guardada para posteriores usos, por tanto se debe siempre comprobar la configuración antes de iniciar cualquier proceso.

## 4.3. Menú de Selección



Figura 4.6: Menú de selección de curva del Horno SMD

En este menú se muestra la tabla con las 32 curvas que se pueden memorizar. La caracterización de estas se realiza en base a su tiempo máximo y la temperatura máxima que deben alcanzar. Aquellas donde el tiempo y la temperatura son nulas, son espacios de memoria que están libres y donde se pueden editar nuevas curvas. También se pueden editar las curvas existentes modificando sus puntos.

Utilice los cursores de desplazamiento arriba y abajo para desplazar el cursor hasta la curva que desea seleccionar. Pulse el botón de cancelar para volver al menú principal, o el botón de aceptar para pasar a ejecutar el proceso, en este caso aparecerá una pantalla de confirmación. Si se pulsa aceptar se pasará a realizar el proceso. Si se pulsa cancelar se vuelve al menú de selección de curva.

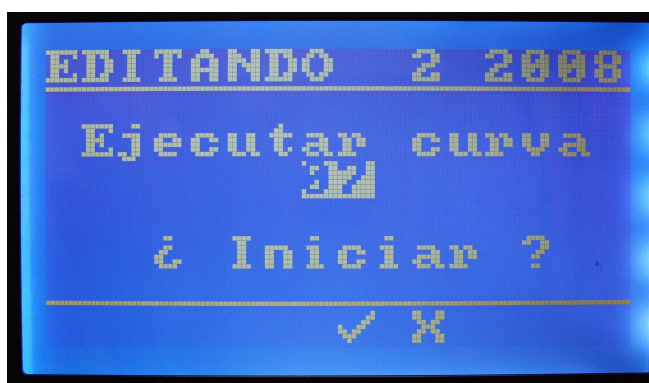


Figura 4.7: Pantalla de confirmación para ejecutar la curva seleccionada

Si se pulsa el **cursor izquierdo**, aparecerá una nueva pantalla donde se pide la confirmación del borrado de la curva seleccionada. Si pulsa aceptar se borrará dicha curva, desplazando todas las demás de forma que todas las curvas estén ordenadas sin espacios libres entre ellas. Si pulsa cancelar se regresará al menú de selección de curva.

Si se pulsa el **cursor derecho**, se entrará en la ventana de edición de la curva seleccionada. En esta se mostrará la curva teórica que se pretende conseguir. Inmediatamente debajo aparece la edición de cada uno de los 8 puntos que pueden componer la gráfica. Con los cursores de dirección arriba y abajo podemos mostrar los puntos de los que se compone la gráfica. Si se pulsa el botón de aceptar se pasa a la ventana de confirmación para ejecutar la curva, si se pulsa el botón cancelar se regresa al menú de selección de curva.



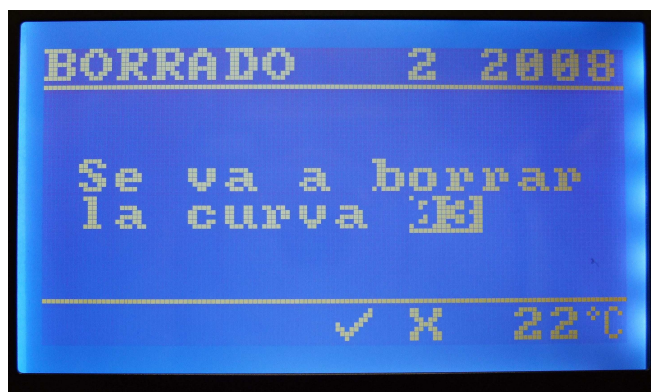


Figura 4.8: Pantalla de confirmación para borrar la curva seleccionada

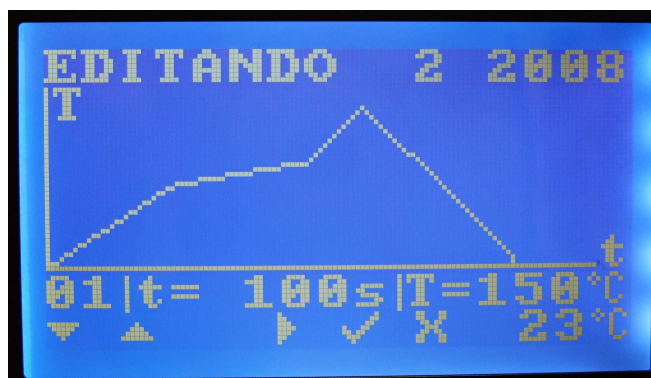


Figura 4.9: Submenú de edición de la curva seleccionada

Dentro del submenú de edición, se debe pulsar el cursor derecho para pasar a la edición del punto que se muestra en la pantalla, pudiendo modificar el tiempo de duración del tramo seleccionado. Si se pulsa cancelar se sale sin modificar el tiempo, si pulsa aceptar se guardarán los cambios realizados y se actualizará la gráfica para observar como afecta el nuevo punto a la curva teórica. Al aceptar el valor de tiempo introducido, se haya modificado o no, se comprobará, descartando los valores que retrocedan en tiempo respecto al punto anterior, y dejando el valor que había inicialmente.

Tras esto se pasa a editar la temperatura, siguiendo el mismo procedimiento que con el tiempo. Con los cursores derecha e izquierdas nos podemos mover por las distintas posiciones de los números, para editar las unidades, decenas, centenas o millares (en caso del tiempo solamente). Una vez se acepta el nuevo valor de temperatura se comprueba que sea correcto, no superando el máximo permitido y se mantendrá el menú de edición. Si se pulsa cancelar se saldrá de la edición sin guardar los cambios realizados en la gráfica



existente. Si se pulsa aceptar se pasa al menú de confirmación de inicio del proceso con la curva modificada o nueva, y se guardará en memoria, pudiendo volver al menú de selección de curvas pulsando el botón cancelar, pero esta vez con las modificaciones guardadas.

#### 4.4. Menú de Transferencia



Figura 4.10: Menú de transferencia de datos del Horno SMD

En este menú se muestra los tres tipos de transferencias que podemos hacer con el PC. Con los cursores arriba y abajo desplazamos el cursor a la opción deseada. Si se pulsa aceptar se pasa a una ventana de confirmación que nos indica la velocidad que se tiene configurada y el proceso elegido. Con el botón cancelar se regresa al menú principal.



Figura 4.11: Pantalla de confirmación de la operación y velocidad de comunicación

### 4.4.1. Herramienta Hyperterminal

El programa que se utiliza para comunicarse con el PC es la herramienta Hyperterminal. Antes de iniciar cualquier comunicación se debe configurar la herramienta utilizada en el PC de acuerdo a las características del horno. Los pasos que se deben seguir son:

- Se crea una nueva conexión y con el nombre deseado.

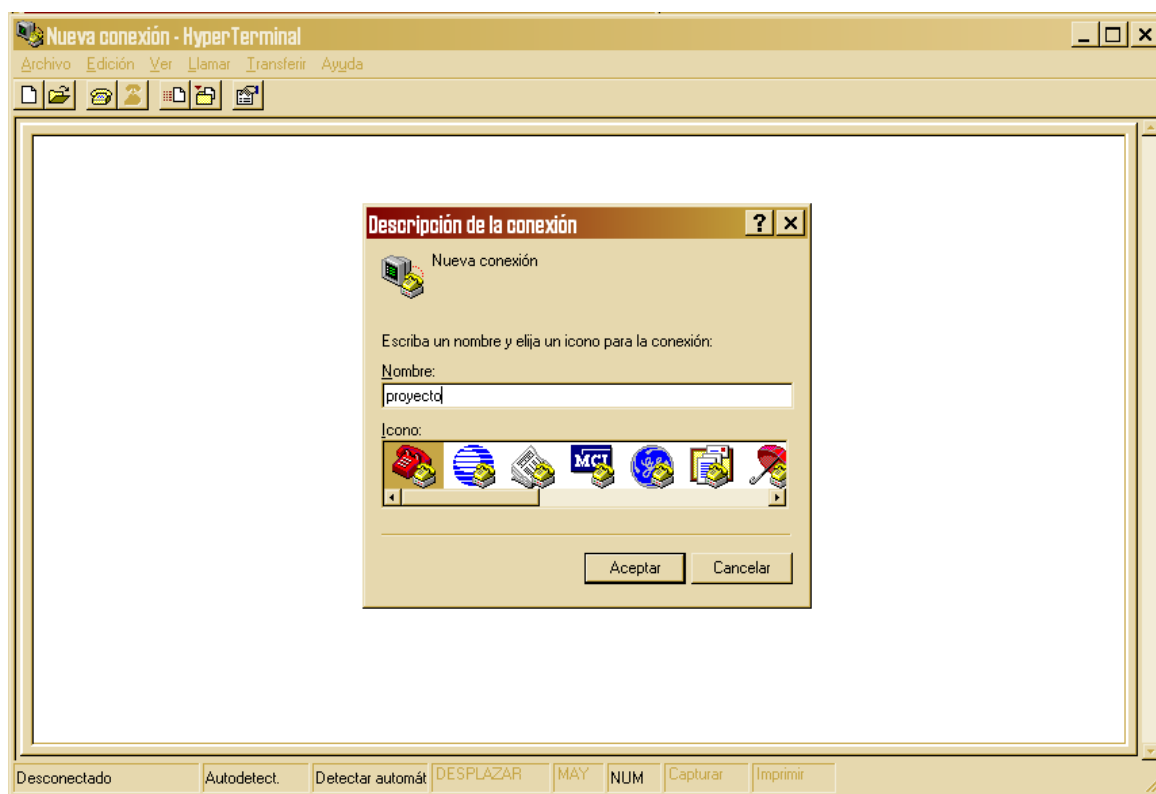


Figura 4.12: Pantalla inicial de la herramienta Hyperterminal

- Se selecciona el puerto con el cual se va a acceder al horno. Después se selecciona la velocidad de comunicación que se desea y el resto de propiedades se configuran como se muestra en la figura.
- Una vez se tiene configurada la conexión, se debe configurar la página que se muestra por pantalla, entrando en el menú **Archivo/propiedades**, y dentro de este en la configuración ASCII, para que exista retorno de carro por cada línea recibida, que muestre por pantalla las líneas recibidas sin sobre-escribirlas.
- Para poder capturar los datos recibidos en un fichero, se debe entrar en el menú **Transferir/Capturar texto...** que pedirá una ruta donde se genere el archivo de texto

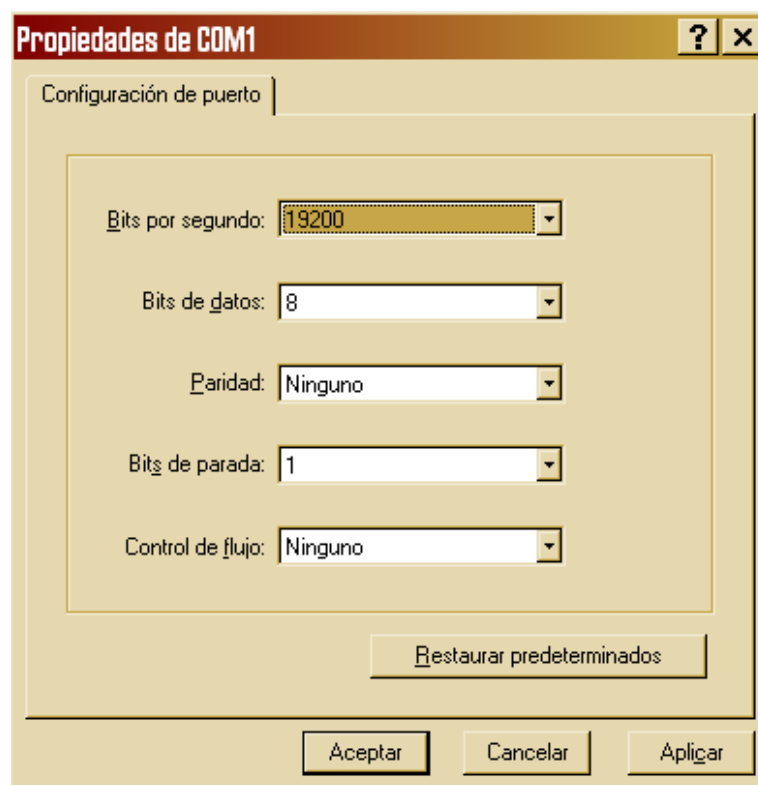


Figura 4.13: Pantalla de configuración de la conexión

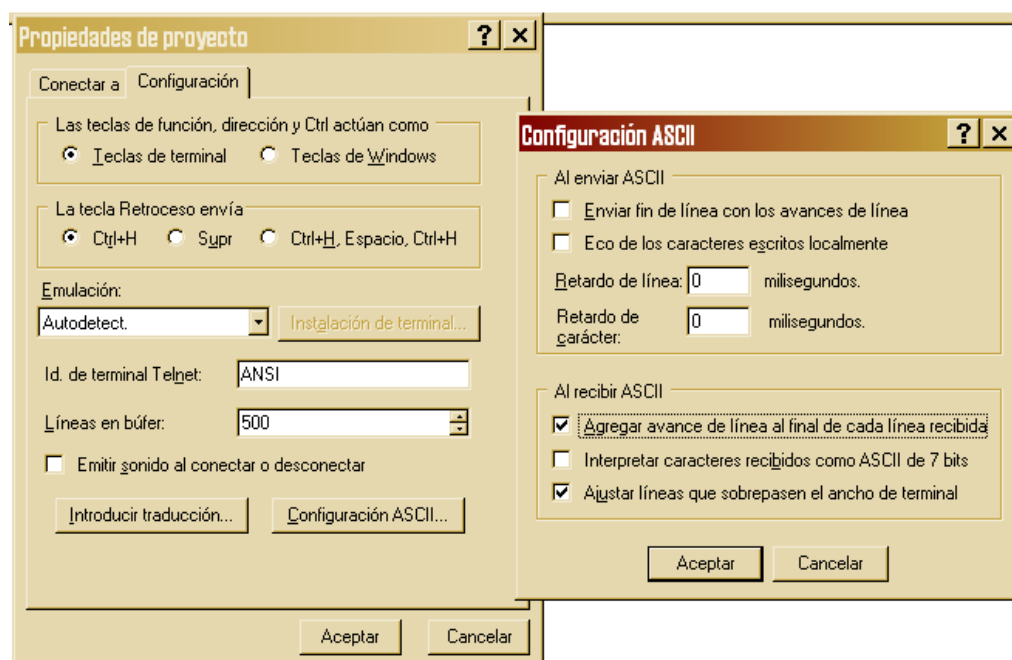


Figura 4.14: Pantalla de propiedades de la conexión

con los datos recibidos.

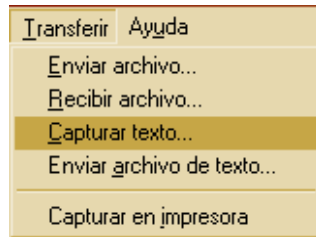


Figura 4.15: Menú de Transferencia del Hyperterminal

- Si se desea transferir un archivo, se debe seleccionar el menú **Transferir/Enviar archivo de texto...** el cual abre una ventana de exploración para seleccionar el archivo de texto que contiene la curva a transferir.
- Finalizado el proceso de configuración, la herramienta está lista para recibir y enviar datos al horno.

#### 4.4.2. Transferencia del proceso al PC

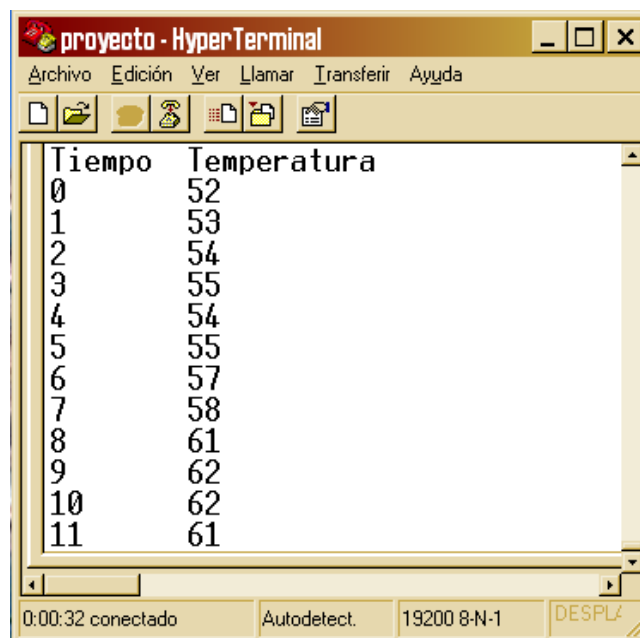


Figura 4.16: Pantalla del Hyperterminal en proceso de captura de datos

En la opción **PROCESO** → **PC** se transfieren las muestras de temperatura almacenados en la memoria durante la ejecución de la última curva realizada, de forma que se pueda representar en una gráfica junto con la curva teórica. El formato de los datos es una tabla donde se indica el tiempo y la temperatura en una cabecera.

#### 4.4.3. Transferencia de las curvas hacia el PC

La opción **CURVA HORNO** → **PC** se transfieren todas las curvas almacenadas en el horno. En este caso el formato de los datos son dos columnas con una cabecera de tiempo y temperatura, seguidas de los puntos que componen la curva y cada curva almacenada está separada por un espacio.

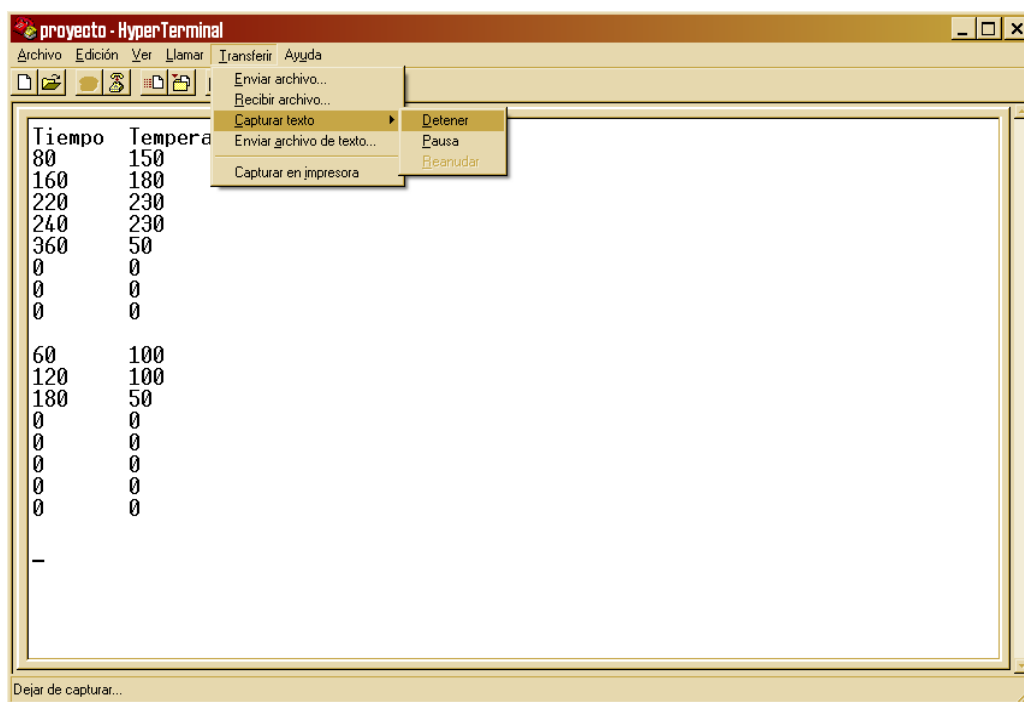
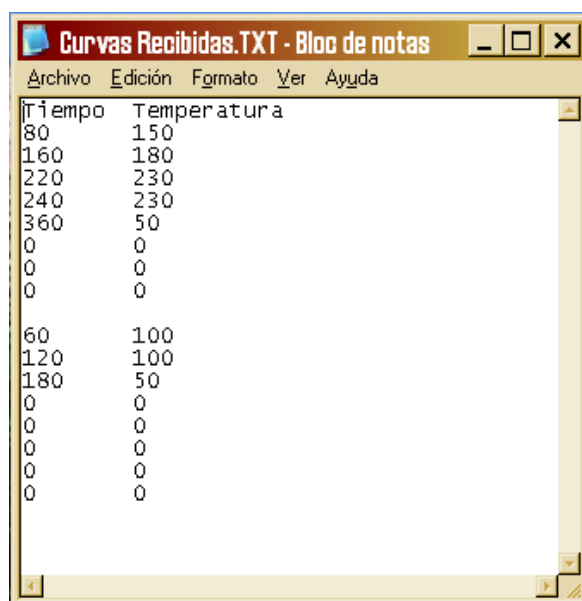


Figura 4.17: Pantalla del Hyperterminal con la captura de las curvas

Tras recibir los datos, se debe detener la captura para poder acceder al fichero que almacena los datos recibidos. Como se muestra en la figura 4.18 con un editor de textos se puede abrir el fichero recibido donde se ven las curvas recibidas, de forma que se pueda modificar alguna y enviarla de vuelta creando una nueva curva.



tiempo	Temperatura
80	150
160	180
220	230
240	230
360	50
0	0
0	0
0	0
60	100
120	100
180	50
0	0
0	0
0	0
0	0
0	0
0	0

Figura 4.18: Documento de texto con las curvas que se han recibido

#### 4.4.4. Transferencia de las curvas desde PC hacia el Horno

Finalmente la opción **CURVA PC** → **HORNO** nos realiza la transferencia de los puntos editados en un documento de texto al horno. Tras la pantalla de confirmación, se muestra una ventana que indicará la espera a la recepción de los datos.



Figura 4.19: Pantalla de espera de datos del PC

Si estos no llegan dentro del tiempo, se finaliza la espera volviendo al menú de transferencia PC. Si los datos no son correctos se mostrará una ventana que nos indica que los datos no son correctos. Si los datos son correctos, se producirá un pitido para indicar que se ha guardado la nueva curva correctamente, volviendo al menú de transferencia PC.

Los datos enviados deben estar en un documento, donde se divida en dos columnas, separadas con una tabulación y al final de la segunda columna un retorno de carro.

La primera columna (tiempo) debe contener números de 2 a 4 dígitos en progresión creciente en tantos puntos como se quieran utilizar. La segunda columna (temperatura) estará formada por números de 2 a 3 dígitos sin superar los 250°C. Los puntos no utilizados deben estar puestos a cero. Si no se cumple alguno de los requisitos la curva no será almacenada debido a un error en los datos.

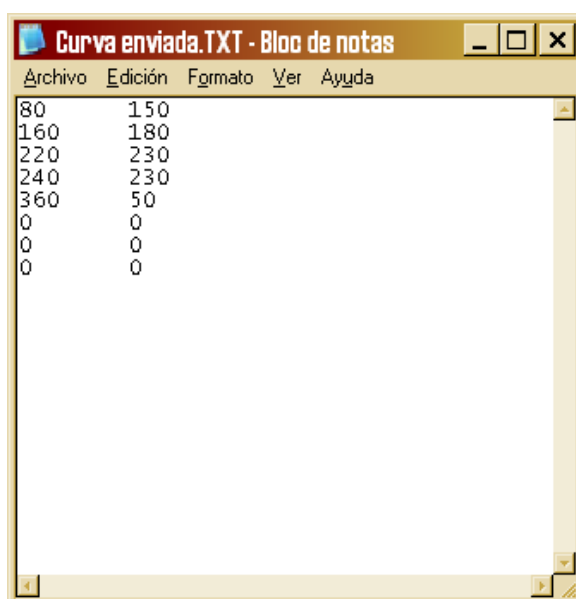


Figura 4.20: Documento de texto con los puntos que se quieren enviar

## 4.5. Menú de Gráfica almacenada

En este último menú, se puede ver la última curva ejecutada en el horno, además de mostrarse los 5 minutos posteriores a la realización de la curva, mostrando así como se ha enfriado el horno después de haber realizado el proceso, o desde el momento en que se canceló el proceso. En este caso el único comando disponible será el de cancelar para volver al menú principal.

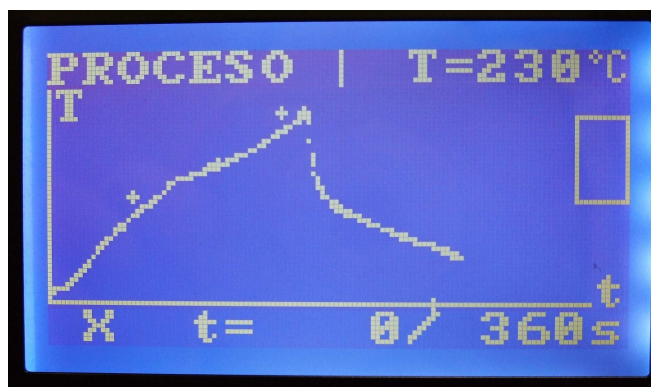


Figura 4.21: Menú de gráfica guardada del Horno SMD

## 4.6. Ejecución de una curva

Cuando se acepta la ejecución de una curva, aparece una nueva pantalla donde se indica el precalentamiento del horno. Este proceso lleva la temperatura del horno a 50°C punto del que parte la ejecución de todas las curvas. Los puntos definidos por debajo de dicha temperatura se llevarán a este valor. Esto se debe a que la temperatura desde la del ambiente hasta los 50°C es un proceso lento y no es de gran interés para el resultado final. Si se pulsa el botón de cancelar se apagarán las resistencias y se volverá al menú principal.



Figura 4.22: Menú de la representación de la curva ejecutada en el Horno SMD

Finalizado el precalentamiento, aparece en la pantalla la gráfica donde se irán actualizando la curva realizada. En esta se ven los puntos teóricos editados por donde deberá pasar la temperatura. En la parte superior se tiene la temperatura real del horno que se esta muestreando. En la parte inferior es tiene el indicador del tiempo, donde se muestra cuanto tiempo se ha ejecutado respecto al total definido en la curva. En el mar-



gen derecho aparece un rectángulo que representa el horno y dentro de este se muestran dos líneas indicativas de las resistencias. Estas nos indican cuando están encendidas, que resistencia se está utilizando y en que momento está recibiendo potencia.

Debido a que el horno no dispone de ventilador interno, el enfriamiento de la capacidad del horno es demasiado lenta para las curvas de soldadura por reflujo. La solución adoptada es la emisión de un pitido y aparición de la letra P (indicadora de puerta) cuando la pendiente definida en la curva sea negativa, de forma que el usuario advierta que se está en proceso de enfriamiento y deberá abrir la puerta. Pero no se debe abrir del todo ya que el enfriamiento sería excesivamente rápido. La puerta dispone de una posición intermedia en la cual se deberá colocar, para que descienda la temperatura pero no de forma brusca.



Figura 4.23: Menú de la representación de la curva ejecutada en el Horno SMD

Durante el proceso, si se pulsa el botón cancelar este será detenido, apagando las resistencias. Al finalizar el proceso, ya por cumplir el tiempo total o bien si se ha cancelado, el horno emitirá tres pitidos indicativos dejando en pantalla la curva real realizada. Para volver al menú principal se deberá pulsar la tecla de aceptar.

## 4.7. Errores

Debido a que al igual que cualquier producto se pueden dar mensajes de error o funcionamiento anómalo, se ha creado una tabla donde se explican los errores más comunes que se pueden producir, no siendo indicativo de que no se produzcan errores diferentes. Algunos de los siguientes errores que se han previsto que puedan suceder o pantallas de error que se pueden mostrar se exponen a continuación.

ERROR	SOLUCIÓN
Se muestra la pantalla de proceso parada y el horno no para de pitar.	Se ha excedido la temperatura máxima del horno. Apague el horno, desenchúfelo y abra la puerta.
Quiero iniciar el proceso pero no pasa a precalentar y sale sin hacer nada.	Compruebe que la temperatura del horno no supera la máxima permitida. Si la supera, desenchufe el horno y abra la puerta para que se enfríe.
Se transfieren los datos de una curva desde el pc al horno y da error.	Compruebe que el documento tiene el formato correcto, que la progresión de tiempo es correcta y la temperatura no supera los 250°C.
Pulso aceptar para iniciar una curva se ha editado y guardado en memoria da un error de datos.	Compruebe que existen puntos editados en la curva seleccionada.
Estoy pasando los puntos de la última gráfica al PC y no llega nada.	Compruebe en el menú de <b>gráfica guardada</b> existe algún punto. Al cancelar el precalentamiento no existirán valores almacenados.
Al encender el horno, se enciende el interruptor pero la pantalla permanece apagada y el teclado no suena.	Uno de los fusibles esta quemado.
Al encender el horno no se enciende la bombilla interna, pero todo lo demás funciona correctamente.	La bombilla está fundida, desenrosque el foco de cristal interior del horno, saque la bombilla y sustitúyala por una de las mismas características.
Una de las resistencias que utilizo no funciona.	La resistencia se haya quemado o no le llegue la señal de control.

Cuadro 4.1: Solución a posibles errores

# Capítulo 5

## Pruebas

En el siguiente capítulo se van a exponer las diferentes pruebas realizadas con el producto final, de forma que se tenga una clara documentación de como proceder para realizar futuros usos del horno.

Inicialmente se debe adquirir pasta de soldadura utilizada en procesos de soldadura por reflujo. Este producto suele tener curvas características propias en las cuales ha demostrado tener las propiedades adecuadas para el proceso de soldadura. Tras examinar una serie de productos, se decide comprar los materiales **96SCR15AGS84 25G-MULTICORE** de **LOCTITE** [14] (a partir de ahora RP15) y **SN62RA10BAS86-25G SYRINGE** [15] cuyas curvas típicas están dentro de las reproducidas por las características del horno.

### 5.1. Proceso de Soldadura

#### 5.1.1. Primera prueba

Se parte de una PCB en la que se desean soldar una serie de componentes. La pasta de soldadura se ha adquirido junto a un juego de agujas para depositar el producto sobre el circuito, que siguen un código de colores siendo de más ancha a más estrecha: marrón, verde, rosa y rojo. El tipo de aguja utilizado dependerá del tipo de pads que se deban soldar, siendo las más finas las que utilicen la aguja más fina.

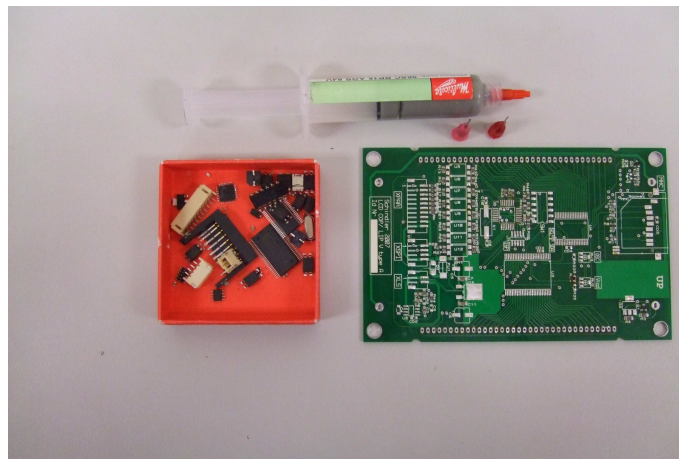


Figura 5.1: Elementos utilizados en la soldadura

Para la primera prueba se toma el rosa, y se deposita sobre los pads de los conectores bolas individuales, colocando después los conectores. Si no se quiere soldar nada más se procedería a realizar el proceso en el horno.

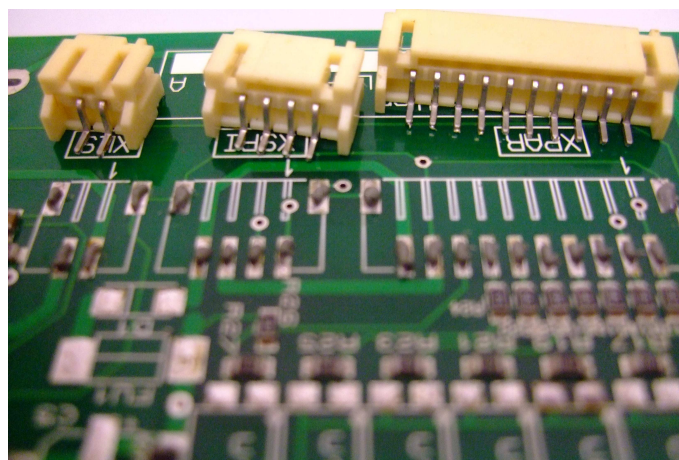


Figura 5.2: Bolas depositadas sobre los pads de conectores

Existe también otro método que consiste en depositar una tira de pasta de soldadura a lo largo de los pads. Para depositar la tira se debe hacer presión poco a poco, moviendo la punta a lo largo de los pads, de forma que el hilo de material se estire un poco sin llegar a romperse. No se debe parar mucho para que el espesor del hilo sea lo más uniforme posible. Para finalizar el hilo, se deja de hacer presión y se apoya la punta sobre el último pad, de forma que se corte el hilo de material.

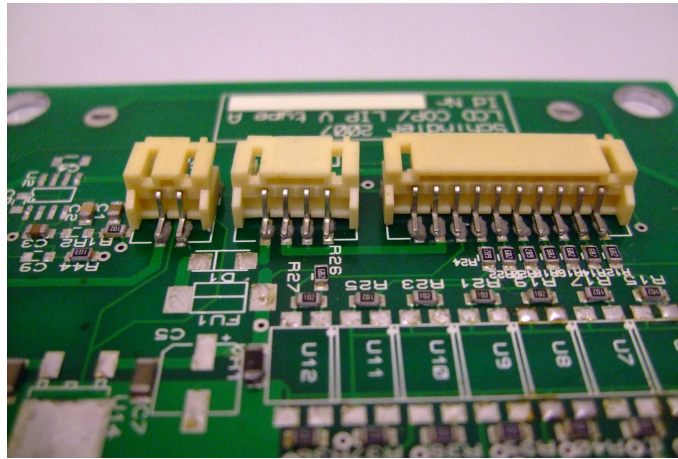


Figura 5.3: Conectores colocados sobre las bolas de pasta

Se puede pensar que producirá cortocircuito cuando se realice el proceso de soldadura por reflujo. Esto no se produce debido a las propiedades del material de soldadura. Cuando el estaño se funde, tiene a redistribuirse en aquellas zonas donde se puede adherir, y la cara de resina imprimida que funciona como aislante, actúa de barrera ante los posibles cortocircuitos. Por tanto se hace la prueba con la misma aguja, depositando una tira fina de material a lo largo de los pads de los integrados que se van a utilizar.

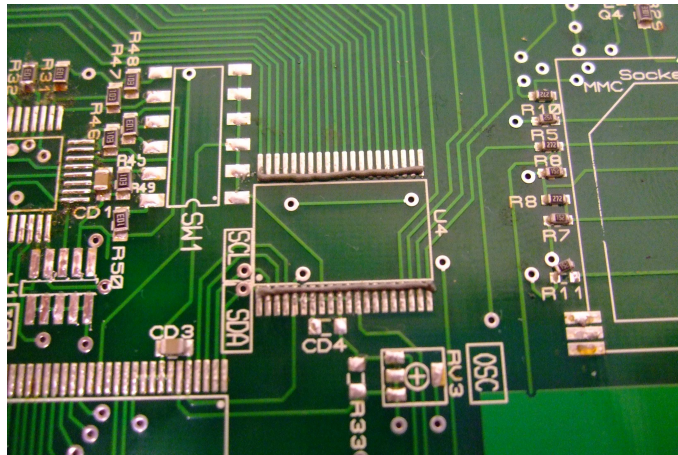


Figura 5.4: Tiras depositadas en los anclajes donde irá el integrado



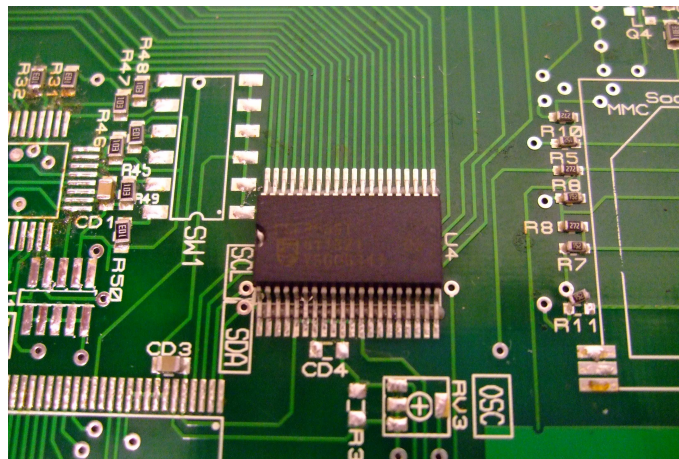


Figura 5.5: Integrado colocado con cuidado sobre las tiras de pasta

Se repite la operación de las tiras con el resto de integrados que se quieren soldar, consiguiendo el resultado que se puede ver en la figura 5.6 en la que se puede comprobar que se soldarán 3 integrados y 3 conectores.

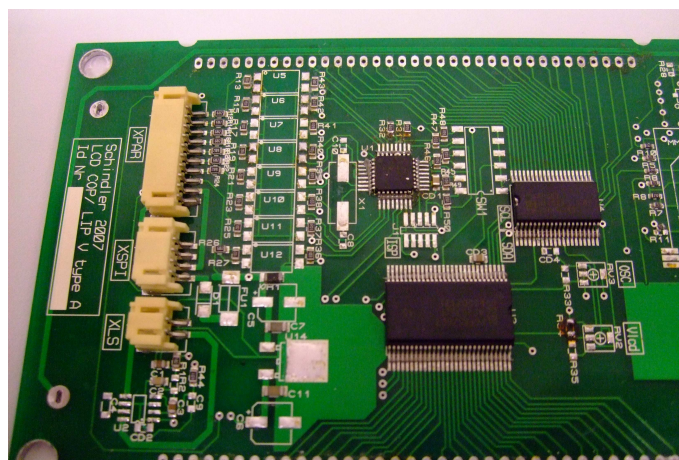
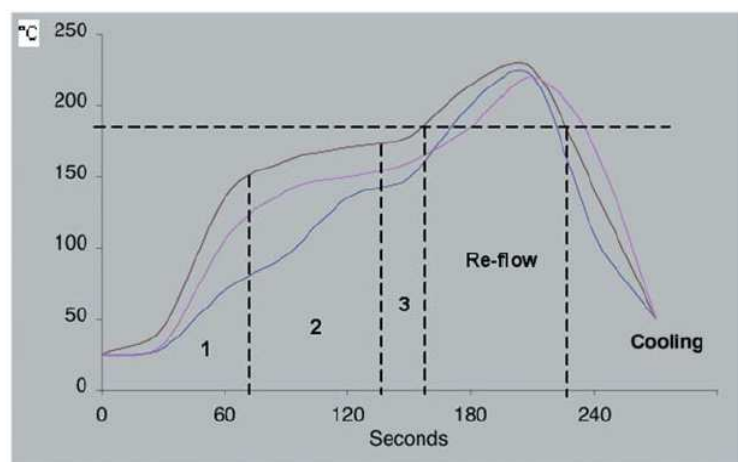


Figura 5.6: Placa dispuesta para soldar

Se introduce la placa en el horno, y se procede a configurar este utilizando para esta prueba las dos resistencias, y colocando la bandeja a media altura. La curva característica del RP15 la proporciona el fabricante en las hojas características, y en referencia a esta, se introduce una curva aproximada a las proporcionadas en el horno, dejando configurado.



**Recommended Re-Flow Profile Range – Forced Air Convection Oven**

Figura 5.7: Curvas proporcionadas por el fabricante

Una vez configurado el horno y con la curva en la memoria, se procede a la ejecución. Al finalizar el proceso, se extrae el circuito, esperando que la temperatura interna del horno este por debajo de 50°C que es lo indicado por el fabricante. El resultado en este caso es satisfactorio para la soldadura de los conectores, pero en los integrados se han producido cortocircuitos.

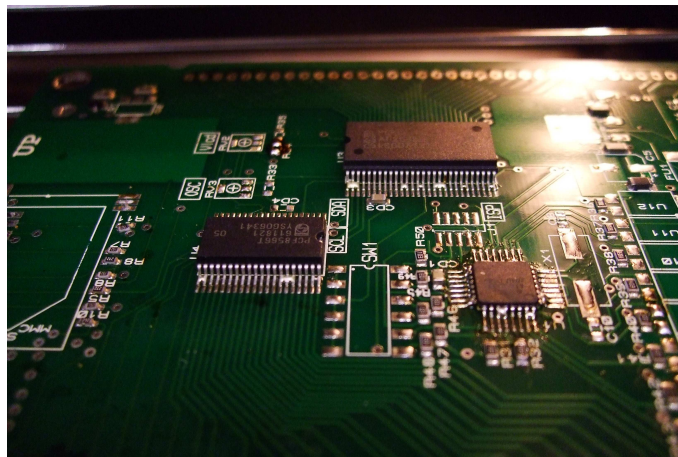


Figura 5.8: Resultado obtenido después del proceso de soldado

En las hojas características del RP15 se explican las causas de los distintos errores, comprobando que el que se ha producido se debe a un exceso de material en la placa. El resultado de esta prueba es que para deposito de bolas de pasta, la aguja rosa es acorde con las necesidades, pero para realizar la deposición con una tira en pads finos, se pro-

ducen cortos debidos a que la deposición no es uniforme, puesto que depende del pulso de la persona encargada de depositar la tira.

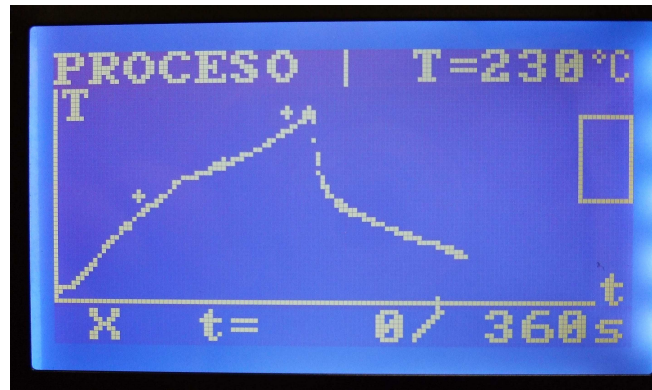


Figura 5.9: Curva realizada por el horno vista en el aparato

Para la comprobación de la curva realizada por el horno, se extrae esta mediante el puerto serie, y se introducen los datos en una representación gráfica junto con la curva teórica introducida obteniendo un resultado bastante aproximado con el proporcionado con el fabricante.

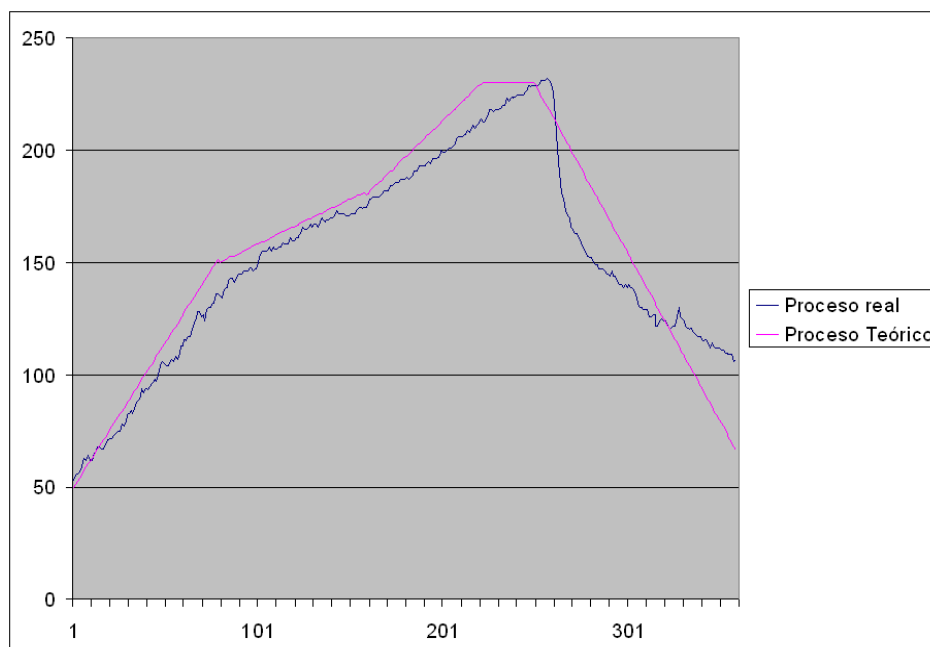


Figura 5.10: Comparación de las curvas teórica y real realizada



### 5.1.2. Segunda prueba

En este nuevo caso, se dispone del mismo material pero en este caso se coge la aguja de color rojo (la más fina) de forma que se sigan los mismos pasos que en la primera prueba. En este caso se soldará solamente un integrado, haciendo el depósito sólo de dos tiras.

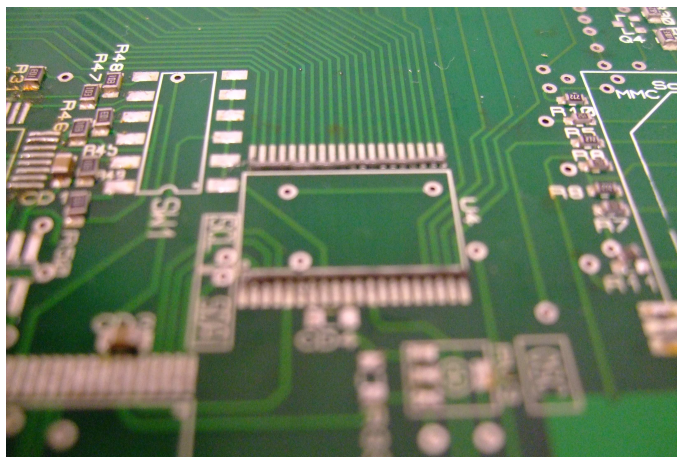


Figura 5.11: Tiras depositadas en los anclajes donde irá el integrado

Como la curva para el material utilizado es el mismo que en el caso anterior, la curva de soldadura ya está introducida en el horno. Además como la configuración del se produce de forma constante para todos los procesos, simplemente se debe introducir la placa en el horno y ejecutar el proceso hasta que finalice.

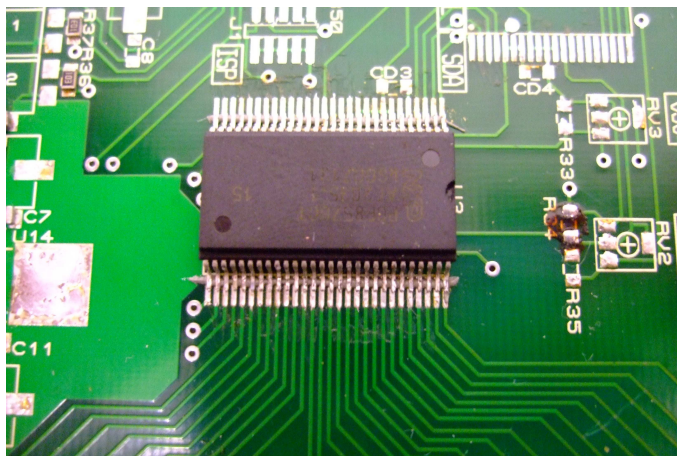


Figura 5.12: Integrado colocado con cuidado sobre las tiras de pasta

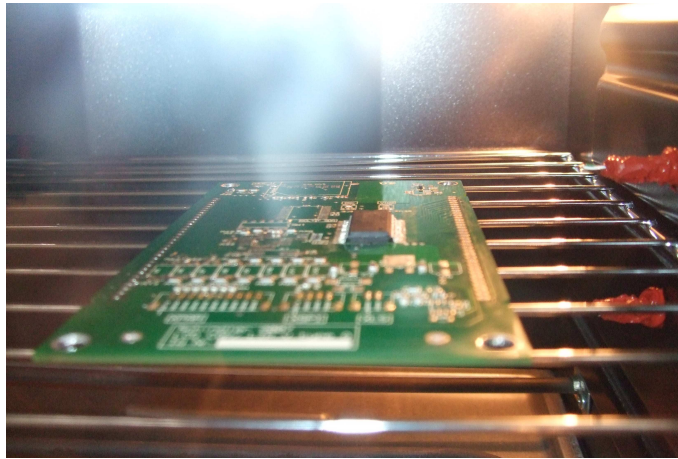


Figura 5.13: Placa dentro del horno durante el proceso

Una vez finalizado el proceso se comprueba que con esta aguja se consigue un resultado correcto en lo referente a la soldadura por separado, sin que se produzcan cortocircuitos. Para asegurar que no existen cortocircuitos se comprueba mediante un polímetro, además de comprobar la soldadura a través de microscopio. Gracias a esto se ve que se han soldado correctamente las pistas obteniendo el resultado buscado.

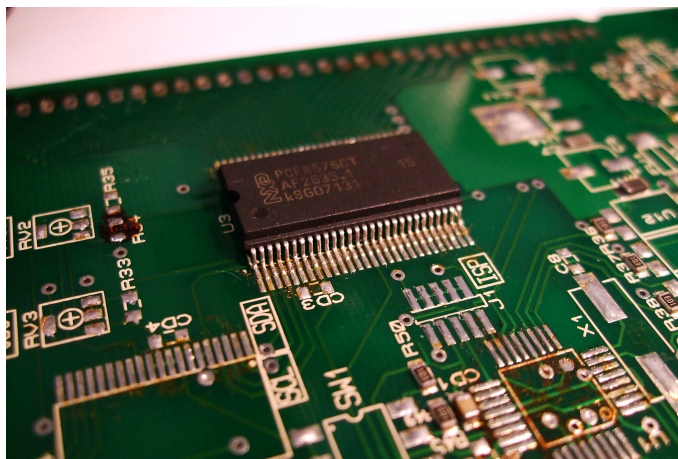


Figura 5.14: Placa una vez finalizada la prueba de soldadura

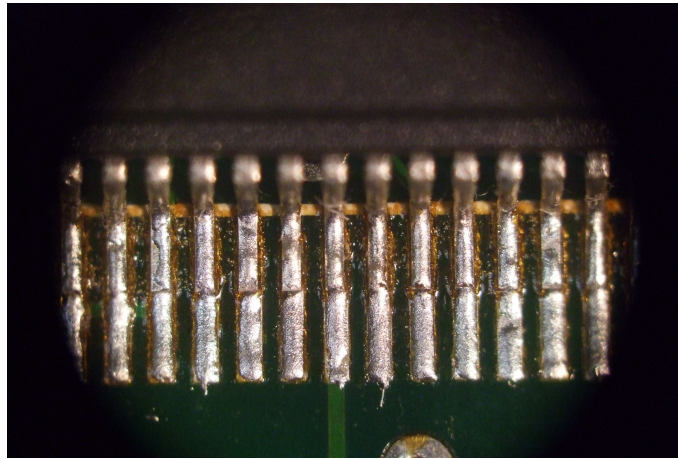


Figura 5.15: Muestra al microscopio del resultado de la soldadura

Una vez finalizado el proceso de soldadura, se debe hacer la limpieza de las agujas utilizadas. Este proceso se realiza con una jeringuilla corriente a la que se introduce agua. Después se colocan las agujas utilizadas y se hace presión de forma que salga todo el material que contienen. La presión se mantiene hasta que por la punta salga el agua introducida. De esta forma se evita la obstrucción de las agujas pudiendo reutilizarlas en futuras soldaduras.



Figura 5.16: Material utilizado para limpiar las agujas

## 5.2. Proceso de Desoldado

Puesto que el horno se utiliza para soldar componentes, una prueba que se quiere realizar es el desoldado de componentes, consiguiendo así reaprovechar los componentes. Para esta prueba se va a tomar una placa ya fabricada y con todos los elementos soldados, de forma que se someta a la curva de soldado diseñada para las pruebas de soldadura, e intentando retirar los componentes cuando la curva se encuentra en el punto más caliente.

### 5.2.1. Primera prueba

Para esta prueba se le colocan sobre la placa a desoldar unos tornillos que hagan de patas. Sobre una bandeja se coloca la placa de forma que los componentes estén mirando hacia abajo, para que se suelten por su propio peso.

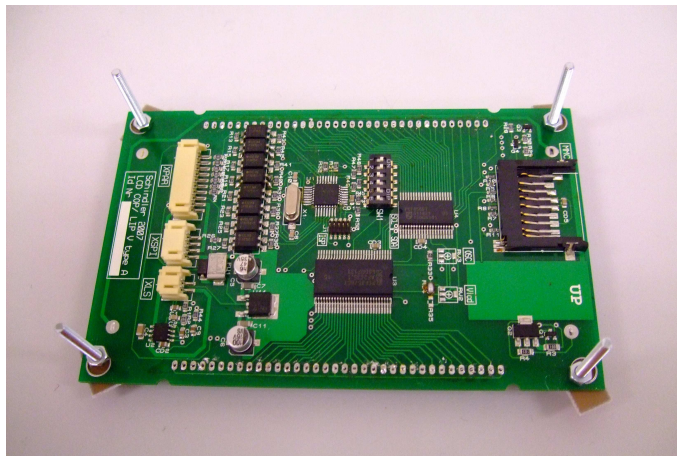


Figura 5.17: Placa que se desea desoldar

En este caso se pasa a ejecutar la curva. Cuando el horno avisa al usuario de que se debe abrir la puerta para que descienda la temperatura, lo que se debe realizar es extraer con pinzas o tenazas la placa (pues la temperatura será en torno a los 230°C) y con un golpe seco sobre la mesa, hacer que los elementos se suelten.

Tras esta prueba, se somete nuevamente la placa a la misma curva de temperatura. Entre proceso y proceso se debe dejar un periodo de al menos 10 minutos para que el horno se refrigere por seguridad para la zona de control. Realizado el segundo calentamiento se procede a quitar los elementos con muchos anclajes, pero esta vez se desplazan con una pinzas sacándolos de sus anclajes, de forma que se consiguen separar del a PCB. En



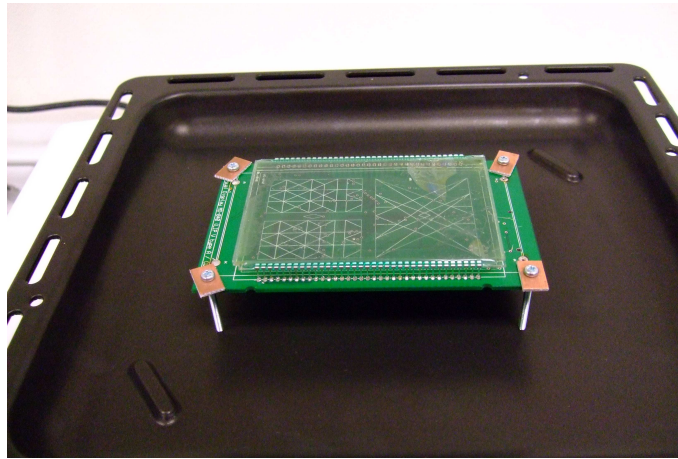


Figura 5.18: Colocación de la placa en el horno

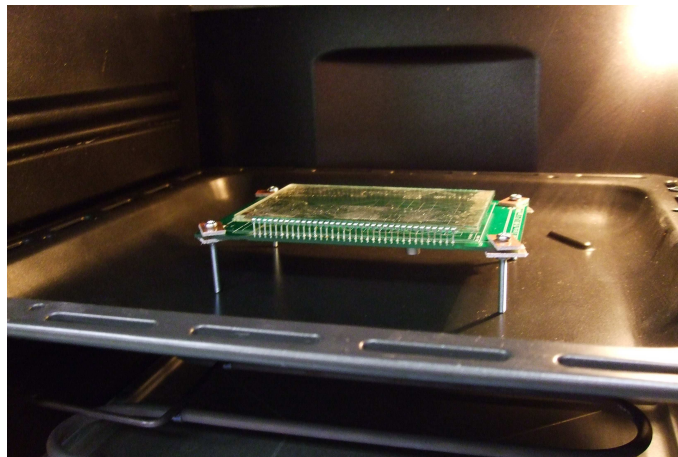


Figura 5.19: Disposición de la placa durante el calentamiento

definitiva se pueden obtener las siguientes conclusiones:

- Los elementos no se sueltan solos debido a la tensión superficial que genera el estaño.
- Tras el golpe seco contra la mesa, se soltarán aquellos componentes de grandes dimensiones y de pocos anclajes, manteniéndose los integrados debido a su gran cantidad e anclajes.
- Los componentes de gran patillaje o de muy reducidas dimensiones deben ser extraídos mediante desplazamiento, es decir, empujando con unas pinzas cuando la placa se encuentra a su máxima temperatura, de forma que se extraigan todos los componentes deseados.

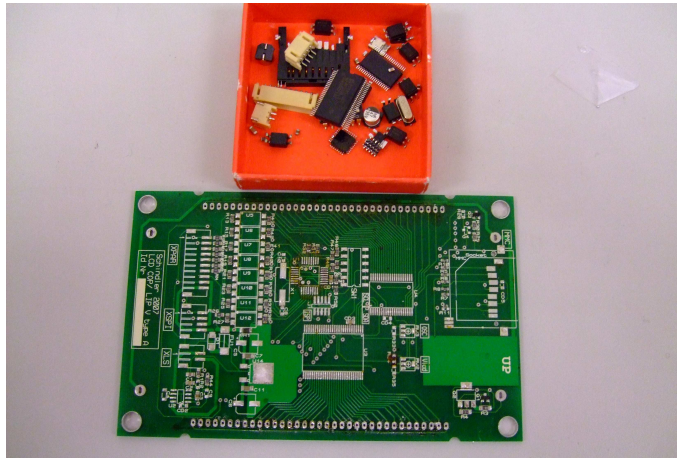


Figura 5.20: Resultado tras la prueba de desoldado

### 5.2.2. Segunda prueba

Tras la primera prueba de soldado, se procede a realizar una segunda prueba de desoldado, pero esta vez se realiza modificando las características del proceso. Para este caso se configura el horno sólo con la resistencia inferior y se le introduce la bandeja de forma que se reduzca el espacio del horno a la mitad aproximadamente.

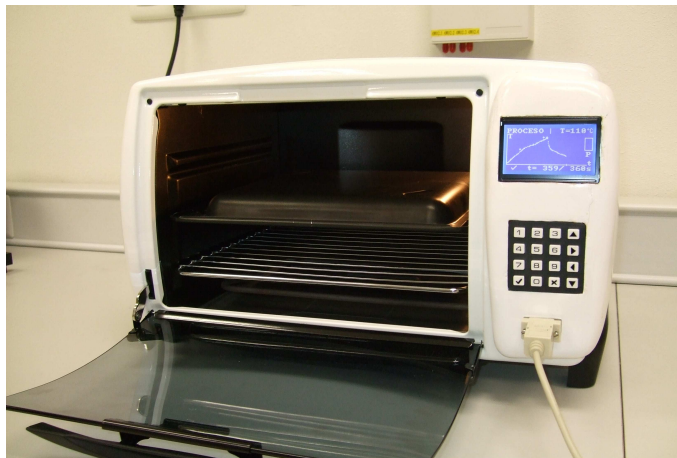


Figura 5.21: Forma de calentar en el desoldado

Con ello se consigue que la placa se caliente más, de forma que alcance esta una temperatura algo mayor que la del ambiente debida a la proximidad a la resistencia. La diferencia no es muy alta, por lo que no pelagra la integridad de la PCB ni de los componentes, ayudando a hacer que el estaño retrase su solidificación. Cuando la curva se encuentra en su temperatura máxima, se saca la placa con pinzas o tenazas y con una

espátula se arrastra por encima de la placa haciendo que todos los elementos se desplacen fuera de la PCB consiguiendo desoldar todos los componentes SMD.

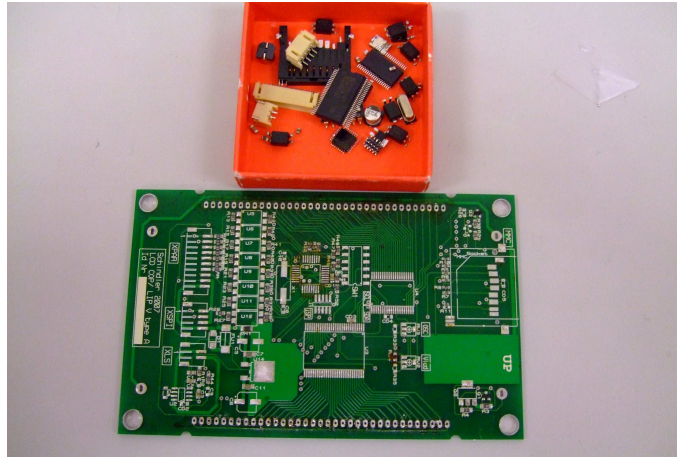


Figura 5.22: Resultado del proceso de desoldado

La espátula debe de ser calentada en la parte de arrastre para evitar que al entrar en contacto con el estaño, este se enfríe solidificando antes de desplazar los componentes. La forma de calentarla puede ser poniéndola sobre el horno, con el fin de que tome la temperatura del exterior del horno, pues es suficiente con una temperatura superior a la del ambiente. El resultado obtenido es el de una PCB sin componentes. Estos deberán de ser limpiados antes de volver a reutilizarlos, al igual que la PCB.





# Capítulo 6

## Presupuesto

### 6.1. Coste de materiales

Inicialmente el valor de los materiales que se han empleado en la construcción tanto del control como de la modificación del horno de partida es el siguiente:

UNI	MATERIAL	PRECIO/UNI	TOTAL
1	Horno de sobremesa Delonghi EO2131	100.00	100.00
1	Transformador 7.5VA	15.46	15.46
1	Puente Rectificador 1.5A, 50V	0.48	0.48
1	L7805ACV - V REG +5.0V, 7805, TO-220-3	0.60	0.60
1	Portafusibles PCB, 20X5MM	0.34	0.34
2	Triacs AVS10CBI TO-220	2.93	5.86
2	Optoacoplador, driver Triac MOC3041M	0.88	1.76
2	Conector Faston hembra	0.21	0.42
16	Resistencias	0.05	0.80
1	Potenciómetro 20K	0.80	0.80
12	Condensadores	0.20	2.40
2	Transistores, BC475 y BC557	0.065	0.13
1	BUZZER PCB	1.87	1.87
1	CRYSTAL, SM, CX-49G, 8MHZ	0.91	0.91
2	MAX6675	12.22	24.44
1	LD1086V33, TO2203	1.65	1.65
1	FLASH SERIAL 16MB M25P16	2.40	2.40
1	MAX232 DIP16, 232	4.00	4.00

UNI	MATERIAL	PRECIO/UNI	TOTAL
1	Teclado 4x4	7.15	7.15
1	Puerto serie macho	4.43	4.43
1	Módulo Gráfico LCD 128X64	93.44	93.44
1	PIC18F458-I/P MICROCHIP	6.13	6.13
3	Borna 2 vias atornillable para PCB	0.28	0.84
3	Disipadores T0-220	0.55	1.65
2	Tira de pines macho	0.53	1.06
2	Tira de pines hembra	0.53	1.06
1	Ventilador 12V dc 0.15mA	14.72	14.72
1	Fusible 100mA	0.19	0.19
1	Zócalo 16 pines	0.54	0.54
1	Zócalo 40 pines	1.84	1.84
2	Zócalo 6 pines	0.14	0.28
1	Resistencia térmica 1100W	15.00	15.00
1	Resto de materiales	20.00	20.00
		<b>TOTAL</b>	<b>332.65</b>

Cuadro 6.1: Costes de los materiales utilizados en el horno

Durante el desarrollo del proyecto, se han tenido problemas en el diseño lo que provocó que varios componentes se deterioraran teniendo que comprar nuevos componentes. Por tanto el importe de materias que se tienen que añadir a mayores es el siguiente:

UNI	MATERIAL	PRECIO/UNI	TOTAL
1	Módulo Gráfico LCD 128X64	93.44	93.44
3	PIC18F458-I/P MICROCHIP	6.13	18.90
2	Triacs AVS10CBI TO-220	2.93	5.86
4	Transistores, BC475 y BC557	0.065	0.26
2	MAX6675	12.22	24.44
1	FLASH SERIAL 16MB M25P16	2.40	2.40
2	MAX232 DIP16, 232	8.00	8.00
		<b>TOTAL</b>	<b>153.30</b>

Cuadro 6.2: Coste de los materiales extras necesarios

## 6.2. Coste del producto

El trabajo del ingeniero ha sido realizado en un periodo de 4 meses, en los cuales se ha desarrollado el diseño de la placa de control, el rutado de las pistas para la fabricación de la placa y se ha programado el firmware del horno. La placa de control se ha fabricado también en el laboratorio por el ingeniero, así como el montaje de los componentes de la placa.

En el taller se ha llevado acabo la modificación del horno para la integración del interruptor, introducción de los sensores y adaptación del chasis a los periféricos y la fabricación de la pieza que adapta los periféricos al chasis. Así también se ha realizado el montaje del producto final en el taller por un operario, contabilizando un total de una semana de trabajo.

Por tanto se pueden definir dos precios finales. El primero el precio que ha tenido el producto final obtenido. El segundo, cuadro 6.4, es el precio que se adquiere la producción de un equipo a partir del diseño que ya se tiene, donde el trabajo de fabricación de la placa y montaje de componentes sería desarrollado por el operario y contribuiría en un día añadido a la semana de trabajo sobre el horno de partida.

CONCEPTO	TIEMPO	PRECIO/H	TOTAL
Materiales			332.65
Material extra			153.30
Ingeniero	4 meses (640h)	22.00	14,080.00
Operario	1 semana (40h)	18.00	720.00
		<b>TOTAL</b>	<b>15,285.95</b>

Cuadro 6.3: Presupuesto de la fabricación del producto final

CONCEPTO	TIEMPO	PRECIO/H	TOTAL
Materiales			332.65
Operario	6 dias (48h)	18.00	864.00
		<b>TOTAL</b>	<b>1,196.65</b>

Cuadro 6.4: Presupuesto de la fabricación de un equipo

### 6.3. Producción en cadena

El proyecto desarrollado ha sido pensado para generar un sólo equipo para el departamento de Electricidad y Electrónica de la Universidad de Valladolid. Pero si se entregara el proyecto al fabricante del horno de partida para que lo desarrollara de forma industrial, se debería estimar el beneficio industrial deseado como el 20 % sobre el coste de producto inicial, de forma que se logre amortización del coste de diseño. Además se debe tener en cuenta el descuento de los materiales en función de la cantidad adquirida. Teniendo en cuenta un 10 % para 100 unidades y un 20 % para 1000 unidades, se consigue un precio de venta al público menor que hornos diseñados específicamente para soldadura por reflujo.

UNIDADES	1	10	100	1000
Costes del producto	1,196.65	11,966.50	107,698.50	957,320.00
Costes de diseño	14,080.00	14,080.00	14,080.00	14,080.00
Coste final	15,276.65	26,046.50	121,778.50	971,400.00
Beneficio	3,055.33	5,209.30	24,355.70	194,280.00
Precio	18,331.98	31,255.80	146,134.20	1,165,680.00
Precio/Unidad	18,331.98	3,125.58	1,461.34	1,165.68
16 % I.V.A	2,933.12	500.09	233.82	186.50
P.V.P (Euros)	21,265.10	3,625.67	1,695.16	1,352.19

Cuadro 6.5: Presupuesto de la fabricación de un equipo

# Capítulo 7

## Conclusiones

El desarrollo de este proyecto pretendía realizar un control sobre un horno convencional de tal forma que se pudieran realizar curvas de temperaturas concretas de soldadura por reflujo, para poder realizar futuros diseños con componentes SMD sin el factor de riesgo de soldar los componentes a través de microscopio o lupa.

Durante el desarrollo de la etapa hardware fueron apareciendo problemas como el espacio disponible para el circuito o la integración de los periféricos en un encapsulado resistente a las altas temperaturas, a los que se les fue dando solución. Además el escaso conocimiento en alguna de las partes del proyecto provocó accidentes de pequeña consideración que ha hecho replantear el diseño, modificándolo o añadiendo partes nuevas para su completo funcionamiento. También se tuvo que modificar la potencia sin que se viera alterado el calentamiento uniforme del interior, debido a que el horno adquirido tenía unas prestaciones que resultaron insuficientes para lo que se buscaba.

Con el desarrollo software se ha tenido que adaptar todo al problema del espacio de memoria de datos ocupada, puesto que el compilador limitaba su uso al 16 % de su capacidad. Puesto que el compilador es proporcionado por HI-TECH y al parecer es un error del compilador, no se ha podido solucionar dicho problema pero se ha conseguido finalizar el programa. Todo ello se ha debido al nuevo compilador utilizado, que no se había empleado antes por el tutor ni el autor. Respecto al control se ha tenido que estudiar la realización de un control PID, de una forma totalmente diferente a la estudiada, puesto que no se disponía de planta concreta ni de medios para estimar el control de forma práctica, recurriendo a un método más práctico que teórico, que ha llevado al cumplimiento del objetivo de control.

Finalmente se puede observar que el resultado ha sido una aplicación basada en un comportamiento no exclusivo para la soldadura por reflujo, sino que podría ser utilizado para otras aplicaciones como repostería, indicando curvas definidas para dichas aplicaciones. Cabe destacar que el proyecto se ha desarrollado en base a soldaduras por reflujo quedando no utilizable para preparación de alimentos debido a los componentes químicos que contienen las pastas de soldadura. Además la aplicación es clara y se puede manejar por cualquier persona, ya sea ajena a este proyecto o no, con un rápido vistazo al manual de usuario.

De valor añadido es la visión global que se ha producido del duro trabajo que supone el desarrollo de un producto, teniendo en cuenta los diversos errores que se han ido produciendo y que resultan ser el día a día del desarrollo real en cualquier empresa. Añadir también que el todo proyecto queda abierto a futuras mejoras, en este caso puede ser la sustitución del microcontrolador por uno con mayores prestaciones que permita un control mucho más fino que el desarrollado, o la adaptación al control desarrollado a un nuevo horno que permita un control electrónico sobre la refrigeración.

# Apéndice A

## Código principal: main.c

```
/** PROGRAMA PARA HORNO DE SOLDADURA POR OLA */
/** PROYECTO FIN DE CARRERA DIRIGIDO POR JESÚS HERNÁNDEZ MANGAS */
/** AUTOR: ALEJANDRO FERNÁNDEZ BLANCO */
/** FECHA: 22-10-2008 */

#include <pic18.h>
#include "main.h"
#include "font.h"
// #include "menus.h"

__CONFIG (1,HS); //18F458
__CONFIG (2,WDTDIS & BORDIS & PWRTDIS); //18F458
__CONFIG (4,DEBUGDIS & LVPDIS & STVRDIS); //18F458
__CONFIG (5,UNPROTECT); //18F458

unsigned char interrup=0;
unsigned int limite=0;

void main(){

    /* DECLARACIÓN DE LAS VARIABLES QUE SE UTILIZAN EN EL PROGRAMA */
    // Reservamos [((NPUNTOS*2)+2)] posiciones para el número
    // de puntos y el punto de partida siempre será el (0,0)
    unsigned int axis[(((NPUNTOS*2)+2))];
    unsigned int i,n; // Variables de bucles
    unsigned char tecla=0; // Variable de la tecla pulsada
    const unsigned char *texto; // Puntero para los textos de menús
    char iniciar=0; // Variable de iniciación de proceso
    char stado; // Variable de estado del proceso

    // Variables para el menú principal o menú 0
    unsigned char menu0=0; // Variable que controla la pantalla del menú
    unsigned char menu=0; // Indica el menú al que se quiere entrar.
    unsigned int fila=0; // Indicará la página que se debe pintar

    // Variables para el menú principal o menú 1
    unsigned char menu1=0;
    unsigned char submenu1=0;

    // Variables para el menú principal o menú 3
    unsigned char menu3=0;
    unsigned char submenu3=0;
```

```

// Variables del submenu 1
//Resistencias utilizadas,
//0 = dos resistencias;
//1 = resistencia superior;
//2 = resistencia inferior;
char resist=0;
// Velocidad de comunicación serie
char veloc=0;

// Variables para el Menú 2
unsigned char aux2=0; // Curva de la primera posición
unsigned char data=0;
unsigned char avan=0; // Puntero a la curva seleccionada
unsigned char index=0;
unsigned char menu2=0; // Variavle del menú 2 para el LCD

// Variable para la opción de editar;
unsigned char point=0; // Punto que se puede editar
unsigned char selec=0; // Numero seleccionado
unsigned char modif=0; // Indicación de si se modifica una curva
unsigned char a=0; // Indice de posición de aux
unsigned int aux[4]={0,0,0,0}; //Variable para nuevos valores
unsigned int valor0, valor; // Valores que se desean guardar

/* FINAL DE LAS VARIABLES UTILIZADAS EN EL PROGRAMA */

Configuracion(); // Configuración del PIC

// Se cargan de memoria los valores de configuración del horno
EEADR=RESIS; // Se carga la dirección de la resistencia
EECON1=0x00;
RD=1;
resist=EEDATA;
if(resist==255)
    resist=0;

EEADR=VELOC; // Se coloca la dirección de la velocidad
EECON1=0x00;
RD=1;
veloc=EEDATA;
if((veloc==255)|| (veloc==0)){
    veloc=0;
    SPBRG=Baud;
}
else{
    if(veloc==1)
        SPBRG=Baud1;
    else
        SPBRG=Baud2;
}

// Se inicializar el LCD
LCD_Inicializacion();
LCD_Bienvenida();

// Borra el sector de copia
Borrar_Sector(COPIACV);
// Realizamos una espera de 5 segundos para la bienvenida
Espera(200);

/** SE INICIA EL BUCLE DEL PROGRAMA PRINCIPAL **/

```



```

while(1){
    tecla=0;
    while(tecla!=12){

        if(menu0==0){
            LCD_Inicializacion(); // Inicializamos la pantalla
            LCD_Menu0();
            LCD_Escribir(conf, 16, 0,1);
            texto=conf;
            fila=16;
            menu0=1;
        }

        LCD_Temp((7*8),(127-32));

        tecla=Teclado();

        switch(tecla){

            case 10: // ACEPTAMOS LA ENTRADA AL MENÚ SELECCIONADO
                LCD_Selec(CMD, TEC_0);
                Espera(4);

                switch(fila){
                    case 16:
                        menu=0;
                        break;

                    case 24:
                        menu=1;
                        break;

                    case 32:
                        menu=2;
                        break;

                    case 40:
                        menu=3;
                        break;
                }
                tecla=12;
                menu0=0;
                break; //case 10

            case 12:
                menu=ERROR;
                submenu1=ERROR;
                break;

            //case 14:// Desplazamiento hacia la derecha
            //break;

            //case 15:// Desplazamiento hacia la izquierda
            //break;

            case 13: // Desplazamiento hacia arriba
                // Seleccionamos la tecla pulsada
                LCD_Selec(CMD, TEC_A);
                // Se elimina el cursor de la selección actual
                LCD_Escribir(texto, fila, 0,0);
                Espera(4);

```

```

switch(fila){
    case 16:
        texto=proc;
        fila=40;
        break;
        171

    case 24:
        texto=conf;
        fila=16;
        break;
        176

    case 32:
        texto=curv;
        fila=24;
        break;
        181

    case 40:
        texto=trns;
        fila=32;
        break;
        186
}
// Ponemos el cursor sobre la nueva selección
LCD_Escribir(texto,fila,0,1);
// Quitamos la selección de la tecla
LCD_Selec(CMD, TEC_A);
break; // case 13
        191

case 16: // Desplazamiento hacia abajo
    LCD_Selec(CMD, TEC_B);
    LCD_Escribir(texto,fila,0,0);
    Espera(4);
        196

    switch(fila){
        case 16:
            texto=curv;
            fila=24;
            break;
            201

        case 24:
            texto=trns;
            fila=32;
            break;
            206

        case 32:
            texto=proc;
            fila=40;
            break;
            211

        case 40:
            texto=conf;
            fila=16;
            break;
            216
    }
    LCD_Escribir(texto,fila,0,1);
    LCD_Selec(CMD, TEC_B);
    break;
        221
}
        226

switch(menu){ // SELECCIÓN DE LOS DIFERENTES MENÚS

```

```

case 0: // MENÚ DE CONFIGURACIÓN DEL HORNO                                231

    menu0=0;
    tecla=0; // Eliminamos la tecla pulsada

    LCD_Inicializacion(); // Inicializamos la pantalla
    LCD_Menu1();
    LCD_Escribir(rstr, 20, 0,1);
    texto=rstr;
    fila=20;

    while(tecla!=12){
        // COMPROBACIÓN DE LA TEMPERATURA
        LCD_Temp((7*8),(127-32));

        tecla=Teclado();

        switch(tecla){
            case 10: // ENTRADA AL SUBMENÚ ELEGIDO
                LCD_Selec(CMD, TEC_0);
                Espera(4);
                if(fila==20)
                    submenu1=0;
                else
                    submenu1=1;

                menu0=ERROR;
                tecla=12;
                break;

            case 12: // Pulasmos cancelar
                LCD_Selec(CMD, TEC_C);
                Espera(4);
                submenu1=ERROR;
                iniciar=ERROR;
                menu0=0;

                break;

            //case 14: // Desplazamiento hacia la derecha
            //break;

            //case 15: // Desplazamiento hacia la izquierda
            //break;

            case 13: // Desplazamiento hacia arriba
                LCD_Selec(CMD, TEC_A);
                LCD_Escribir(texto, fila, 0,0);
                Espera(4);
                if(fila==20){ // Comprobamos la selección
                    texto=port;
                    fila=36;
                }
                else{ // Fila =36
                    texto=rstr;
                    fila=20;
                }
                LCD_Escribir(texto, fila, 0,1);
                LCD_Selec(CMD, TEC_A);
                break;

            case 16: // Desplazamiento hacia abajo del cursor

```

```

        LCD_Selec(CMD, TEC_B);
        LCD_Escribir(texto, fila, 0,0);
        Espera(4);
        if(fila==20){ // Comprobamos la selección
            texto=port;
            fila=36;
        }
        else{ // fila = 36
            texto=rstr;
            fila=20;
        }
        LCD_Escribir(texto, fila, 0,1);
        LCD_Selec(CMD, TEC_B);
        break;
    }
}
break; // FIN MENU 0

case 1: // MENÚ DE SELECCIÓN DE CURVA
    // Inicializamos las variables
    menu0=0;
    submenu1=ERROR;
    tecla=0;
    menu2=0;
    avan=0;
    aux2=0;
    data=0;

    while(tecla!=12){

        if(menu2==0){
            GIE=0; // Porque en LCD_Curvas se accede a memoria.
            LCD_Inicializacion();
            for(i=0; i<((NPUNTOS*2)+2); i++) // Inicializo la matriz
                axis[i]=0;

            LCD_Curvas(aux2, axis);
            LCD_Numeros(avan+1, ((8*(avan-aux2))+21), 0, 3);
            LCD_Menu2();
            menu2=1;
            if(interrup==1)
                GIE=1;
        }

        LCD_Temp((7*8),(127-32));

        tecla=Teclado();

        switch(tecla){

            case 10: // ACEPTAMOS LA CURVA ELEGIDA
                LCD_Selec(CMD, TEC_0);
                Espera(4);

                LCD_Inicializacion(); // Reinicializamos la pantalla
                LCD_Menu5();
                LCD_Numeros((avan+1), 24, 56, 3);

                tecla=0;
                while(tecla!=12){
                    tecla=Teclado();

```

```

if(tecla==10){ // Se debe confirmar la opción elegida
    // Primero pasa a axis los puntos de la curva
    for(i=0; i<((NPUNTOS*2)+2); i++)
        axis[i]=0;
    356

    GIE=0;
    Memoria_Dir_Lec(REFEREN+(avan*32));
    361

    for(i=0;i<(NPUNTOS*2);i++){
        data=Memoria_Dat_Lec();
        axis[i+2]=((data<8)|Memoria_Dat_Lec());
    }
    Memoria_Fin();
    366
    if(interrupt==1)
    GIE=1;

    if(axis[2]==0xFFFF){
        LCD_Error();
        371
        Espera(120); // Esperamos tres segundos
        tecla=12;
        iniciar=0;
    }
    else{
        376
        tecla=12;
        iniciar=1;
        index=avan;
        menu0=0;
        submenu1=ERROR;
        381
    }
    } //if(tecla==10)
} //while(tecla!=12)
menu2=0;
if(iniciar!=1)
    386
    tecla=0;
break; // case 10;

case 12: // Pulsamos cancelar
    submenu1=ERROR;
    391
    menu0=0;
break;

case 14: // Desplazamiento hacia la derecha del cursor
    // SE PASA A LA EDICIÓN DE LA CURVA
    396
    LCD_Selec(CMD, TEC_D);
    // Primero pasa a axis los puntos de la curva
    for(i=0; i<((NPUNTOS*2)+2); i++) // Inicializo la matriz
        axis[i]=0;
    401

    GIE=0;
    Memoria_Dir_Lec(REFEREN+(avan*32));

    for(i=0;i<(NPUNTOS*2);i++){
        data=Memoria_Dat_Lec();
        406
        axis[i+2]=((data<8)|Memoria_Dat_Lec());
    }
    Memoria_Fin();

    if(interrupt==1)
        411
        GIE=1;

    if(axis[2]==0xFFFF){
        for(i=0; i<((NPUNTOS*2)+2); i++)

```

```

    axis[i]=0;
}

tecla=0;
point=0;
LCD_Inicializacion(); // Reinicializamos la pantalla
LCD_Edicion(point,axis);
LCD_Escalado(axis,128,38,0,17);
LCD_Menu22();

while (tecla!=12){
    // COMPROBACIÓN DE LA TEMPERATURA
    LCD_Temp((7*8),(127-32));

    tecla=Teclado();

    switch (tecla){
        case 10:
            LCD_Selec(CMD, TEC_0);
            if(modif==1){
                GIE=0;
                // Guardo la curva modificada en el espacio de copia
                Memoria_Dir_Esc(COPIACV+(avan*32));
                for(i=0;i<(NPUNTOS*2);i++)
                    Memoria_Dat_Esc(axis[i+2],0);
                Memoria_Fin(); // Fin del proceso de escritura

                // Se guardan los cambios en memoria
                // Se pasan las curvas al sector de copia
                Borrar_Curva(axis,avan,0);
                // Se borra el sector principal
                Borrar_Sector(REFEREN);
                Espera(80);
                // Reinicializamos la pantalla
                LCD_Inicializacion();
                LCD_Menu5();
                LCD_Numeros((avan+1), 24, 56, 3);
                // Se pasan las curvas al sector principal
                Borrar_Curva(axis,avan,1);
                // Se borra el sector de copia
                Borrar_Sector(COPIACV);
                Espera(80);
                if(interrup==1)
                    GIE=1; // Se habilitan las interrupciones globales

                tecla=0;
                while (tecla!=12){
                    tecla=Teclado();
                    if (tecla==10){ // Se debe confirmar el proceso

                        // Primero pasa a axis los puntos de la curva
                        for(i=0; i<((NPUNTOS*2)+2); i++) // Inicializo la matriz
                            axis[i]=0;

                        GIE=0;
                        Memoria_Dir_Lec(REFEREN+(avan*32));

                        for(i=0;i<(NPUNTOS*2);i++){
                            data=Memoria_Dat_Lec();
                            axis[i+2]=((data<<8)|Memoria_Dat_Lec());
                        }
                        Memoria_Fin();
                    }
                }
            }
        }
    }
}

```

```

        if(interrupt==1)
            GIE=1;

        if(axis[2]==0xFFFF){
            LCD_Error();
            Espera(120); // Espero 3 segundos
            tecla=12;
            iniciar=0;
        }
        else{
            tecla=12;
            iniciar=1;
            index=avan;
            menu0=0;
            submenu1=ERROR;
        }
        // if(tecla==10)
    } //while(tecla==12)
    // Modificando aquella a la que se accedio
    modif=0;
} //if(modif==1)
else{
    LCD_Inicializacion(); // Reinicializamos la pantalla
    LCD_Menu5();
    LCD_Numeros((avan+1), 24, 56, 3);

    tecla=0;
    while(tecla!=12){
        tecla=Teclado();
        if(tecla==10){ // Se debe confirmar la orden
            tecla=12;
            iniciar=1;
            index=avan;
        }
    }
} //fin else
break; // case 10;

case 12:
    LCD_Selec(CMD, TEC_C);
    menu2=0;
break;

case 14: // Desplazamiento hacia la derecha

    // EDICIÓN DEL PUNTO CORRESPONDIENTE
    LCD_Escribir(cmd4,CMD,0,0);
    tecla=0;
    selec=35;
    a=0;

    valor0=axis[(point*2)+2];
    aux[0]=valor0/1000;
    aux[1]=(valor0-(aux[0]*1000))/100;
    aux[2]=(valor0-(aux[0]*1000)-(aux[1]*100))/10;
    aux[3]=(valor0-(aux[0]*1000)-(aux[1]*100)-(aux[2]*10));

    LCD_Selec(EDI, selec);

    while(tecla!=12){
        // COMPROBACIÓN DE LA TEMPERATURA
        LCD_Temp((7*8),(127-32));

```

```

tecla=Teclado();
switch(tecla){
    case 10: // Se acepta el cambio del número
        LCD_Selec(CMD, TEC_0);
        LCD_Selec(EDI, selec); // Deselecciona el último número
        valor=(aux[0]*1000)+(aux[1]*100)+(aux[2]*10)+aux[3];

        if(valor0!=valor){
            // Se comprueba que sea correcto el tiempo
            if(valor>axis[point*2]){
                axis[(point*2)+2]=valor;
                LCD_Escalado(axis,128,38,0,17);
                modif=1;
            }
        }
        LCD_Edicion(point,axis);
        selec=94;
        tecla=12;
        LCD_Selec(CMD, TEC_0);
    break;

    case 12: // Se sale de la selección
        LCD_Selec(CMD, TEC_C);
        LCD_Edicion(point,axis);
        // Se deselecciona el comando pulsado
        LCD_Escribir(cmd3,CMD,0,0);
    break;

    case 13: // Desplazamiento hacia arriba
    break;

    case 16: // Deplazamiento hacia abajo
    break;

    case 14: // Desplazamiento hacia la derecha
        LCD_Selec(CMD, TEC_D);
        LCD_Selec(EDI, selec);

        if(selec!=59){
            selec+=8;
            a++;
        }
        else{
            selec=35;
            a=0;
        }

        LCD_Selec(EDI, selec);
        LCD_Selec(CMD, TEC_D);
    break;

    case 15: // Desplazamiento hacia la izquierda
        LCD_Selec(CMD, TEC_I);
        LCD_Selec(EDI, selec);

        if(selec!=35){
            selec-=8;
            a--;
        }
        else{
            selec=59;

```



```

        a=3;
    }

    LCD_Selec(EDI, selec);
    LCD_Selec(CMD, TEC_I);
break;

case 0: // Si tecla es 0 no se ha pulsado nada
break;

default: // Pulsar cualquier otra tecla
    if(tecla==11)
        tecla=0;

    LCD_Caracter((tecla+48),EDI,selec,0); // Pinto el nuevo número
    if(selec!=59){
        aux[a++]=tecla; // Guardo el valor de la tecla
        selec+=8; // Hacemos que el cursor avance
    }
    else // En caso de estar en la posición de unidades
        aux[a]=tecla; // Guardo el valor de la tecla

    LCD_Selec(EDI, selec); // Marco con el cursor la tecla pulsada
break;
} // fin switch
} // while(tecla!=12)
if(selec==94){
    tecla=0;
    selec=94;
    a=1;

    valor0=axis[(point*2)+3];
    aux[1]=valor0/100;
    aux[2]=(valor0-(aux[1]*100))/10;
    aux[3]=(valor0-(aux[1]*100)-(aux[2]*10));

    LCD_Selec(EDI, selec);

while(tecla!=12){
    tecla=Teclado();
    switch(tecla){
        case 10: // Se acepta el cambio en el número
            LCD_Selec(CMD, TEC_0);
            valor=(aux[1]*100)+(aux[2]*10)+aux[3];

            if((valor!=0)&&(valor0!=valor)&&(axis[(point*2)+2]!=0)){
                if(valor<(MAXIMO-10)){ // Se comprueba la temperatura
                    axis[(point*2)+3]=valor;
                    modif=1;
                    LCD_Escalado(axis,128,38,0,17);
                }
            }
            point++;
            LCD_Edicion(point,axis);
            LCD_Escribir(cmd3,CMD,0,0);
            tecla=12;
break;

        case 12: // Se sale de la selección
            LCD_Selec(CMD, TEC_C);
            LCD_Edicion(point,axis);
            LCD_Escribir(cmd3,CMD,0,0);

```

```

        break;

        case 13: // Desplazamiento hacia arriba
        break;

        case 16: // Desplazamiento hacia abajo
        break;

        case 14: // Desplazamiento hacia la derecha
            LCD_Selec(CMD, TEC_D);
            LCD_Selec(EDI, selec);

            if(selec!=110){
                selec+=8;
                a++;
            }
            else{
                selec=94;
                a=1;
            }

            LCD_Selec(EDI, selec);
            LCD_Selec(CMD, TEC_D);
        break;

        case 15: // Desplazamiento hacia la izquierda
            LCD_Selec(CMD, TEC_I);
            LCD_Selec(EDI, selec);

            if(selec!=94){
                selec-=8;
                a--;
            }
            else{
                selec=110;
                a=3;
            }

            LCD_Selec(EDI, selec);
            LCD_Selec(CMD, TEC_I);
        break;

        case 0: // Si tecla es 0 no se ha pulsado nada
        break;

        default: // Si se pulsa cualquier número
            if(tecla==11)
                tecla=0;

            LCD_Caracter((tecla+48),EDI,selec,0);
            if(selec!=110){
                aux[a++]=tecla;
                selec+=8; // Hacemos el cursor
            }
            else
                aux[a]=tecla;

            LCD_Selec(EDI, selec);
        break;
    }
} // while(tecla!=12)
} // if(selec==94)

```

```

        tecla=0;
        break;

//case 15: // Desplazamiento hacia la izquierda
//break;

case 13: // Desplazamiento hacia arriba
    LCD_Selec(CMD, TEC_A);
    if(point==0)
        point=7;
    else
        point--;

    LCD_Edicion(point,axis);
    LCD_Selec(CMD, TEC_A);
    break;

case 16: // Desplazamiento hacia abajo
    LCD_Selec(CMD, TEC_B);
    if(point==7)
        point=0;
    else
        point++;

    LCD_Edicion(point,axis);
    LCD_Selec(CMD, TEC_B);
    break;
}
} // Fin del while
// Se cambian las variables del programa
menu2=0;
if(iniciar!=1)
    tecla=0;
break; // fin case 14;

case 15: // Desplazamiento hacia la izquierda
    LCD_Selec(CMD, TEC_I);

// VENTADA DONDE CONFIRMACIÓN DE BORRADO
tecla=0;
LCD_Inicializacion();
LCD_Menu21();
LCD_Numeros((avan+1), 32, 80, 3);

while(tecla!=12){
    // COMPROBACIÓN DE LA TEMPERATURA
    LCD_Temp((7*8),(127-32));

    tecla=Teclado();

    if(tecla==10){ // Se debe confirmar el borrado

        LCD_Selec(CMD, TEC_0);
        GIE=0; // Se quitan las interrupciones globales
        Borrar_Curva(axis,avan,0); // Copiado de las curvas
        Borrar_Sector(REFEREN); // Borrado del sector
        LCD_Selec(CMD, TEC_0);
        LCD_Menu211();
        Espera(80); // Se esperan 2 segundos
        Borrar_Curva(axis,avan,1); // Copia de las curvas
        Borrar_Sector(COPIACV); // Borrado del sector
        Espera(80);
    }
}

```

```

        if(interrup==1)
            GIE=1; // Se habilitan las interrupciones globales

        tecla=12;
        avan=aux2=0;
    } //if(tecla==10)
} //while(tecla!=12)
tecla=0;
menu2=0;
break;

case 13: // Desplazamiento hacia arriba del cursor
    GIE=0;
    LCD_Selec(CMD, TEC_A);
    if(avan==0){
        avan=31;
        aux2=(avan-3);
    }
    else{
        avan--;
        if(aux2>avan)
            aux2=avan;
    }

    LCD_Curvas(aux2, axis);
    LCD_Numeros((avan+1), ((8*(avan-aux2))+21), 0, 3);

    LCD_Selec(CMD, TEC_A);
    if(interrup==1)
        GIE=1;
    break;

case 16: // Desplazamiento hacia abajo del cursor
    GIE=0;
    LCD_Selec(CMD, TEC_B);
    avan++;
    if((avan-aux2)>3){
        if(avan>31){
            avan=0;
            aux2=0;
        }
        else
            aux2++;
    }

    LCD_Curvas(aux2, axis);
    LCD_Numeros((avan+1), ((8*(avan-aux2))+21), 0, 3);

    LCD_Selec(CMD, TEC_B);
    if(interrup==1)
        GIE=1;
    break;
} //switch(tecla)
} //while(tecla!=12)
menu2=0;
break; // Fin menu de edición

case 2: // MENÚ DE TRANSFERENCIA AL PC
    menu0=0;
    submenu1=ERROR;
    tecla=0; // Eliminamos la tecla pulsada

    while(tecla!=12){

```

```

if(menu3==0){
    LCD_Inicializacion(); // Inicializamos la pantalla
    LCD_Menu3();
    LCD_Escribir(prpc, 16, 0,1);
    texto=prpc;
    fila=16;
    menu3=1;
}

// COMPROBACIÓN DE LA TEMPERATURA
LCD_Temp((7*8),(127-32));

tecla=Teclado();

switch(tecla){

    case 10: // ACEPTAMOS LA ENTRADA AL SUBMENÚ
        LCD_Selec(CMD, TEC_0);
        tecla=0; // Eliminamos la tecla pulsada

        if(fila==16)
            submenu3=0;
        else{
            if(fila==28)
                submenu3=1;
            else
                submenu3=2;
        }

        LCD_Inicializacion(); // Inicializamos la pantalla
        LCD_Menu31(submenu3,veloc);

        while(tecla!=12){
            // COMPROBACIÓN DE LA TEMPERATURA
            LCD_Temp((7*8),(127-32));

            tecla=Teclado();

            switch(tecla){

                case 10:
                    LCD_Selec(CMD, TEC_0);
                    Espera(8);
                    LCD_Selec(CMD, TEC_0);
                    GIE=0; // Se accede a la memoria en las tres funciones
                    switch(submenu3){
                        case 0: // PROCESO -> PC
                            Lectura_Puerto(MUESTRA, 0); // dirección 0
                            break;

                        case 1: // CURVAS -> PCGUARDADAS
                            Lectura_Puerto(REFEREN, 1); // 1 = lectura de puntos
                            break;

                        case 2: // PC -> HORNO
                            LCD_Aviso();
                            RS232_Recibo();
                            break;
                    }
                    if(interrup==1)
                        GIE=1;

```

```

        tecla=12;
        menu3=0;
        break;
    case 12:
        LCD_Selec(CMD, TEC_C);
        Espera(8);
        menu3=0;
        LCD_Selec(CMD, TEC_C);
        break;
    } // swicth(tecla)
} //while(tecla!=12)
tecla=0;
break; // case 10;

case 12: // Botón de cancelar
    LCD_Selec(CMD, TEC_C);
    submenu1=ERROR;
    menu3=0;
    menu0=0; // Volvemos al inicio
    LCD_Selec(CMD, TEC_C);
    break;

//case 14: // Desplazamiento hacia la derecha
//break;

//case 15: // Desplazamiento hacia la izquierda
//break;

case 13: // Desplazamiento hacia arriba
    LCD_Selec(CMD, TEC_A);
    LCD_Escribir(texto, fila, 0,0);
    Espera(4);
    switch(fila){
        case 16:
            texto=pchr;
            fila=40;
            break;

        case 28:
            texto=prpc;
            fila=16;
            break;

        case 40:
            texto=hrpc;
            fila=28;
            break;
    }
    LCD_Escribir(texto, fila, 0,1);
    LCD_Selec(CMD, TEC_A);
    break;

case 16: // Desplazamiento hacia abajo del cursor
    LCD_Selec(CMD, TEC_B);
    LCD_Escribir(texto, fila, 0,0);
    Espera(4);
    switch(fila){
        case 16:
            texto=hrpc;
            fila=28;

```

```

        break;

        case 28:
            texto=pchr;
            fila=40;
            break;

        case 40:
            texto=prpc;
            fila=16;
            break;
    }
    LCD_Escribir(texto, fila, 0,1);
    LCD_Selec(CMD, TEC_B);
    break;
} //switch(tecla)
} //while(tecla!=12)
break;

case 3: // MENÚ DE REPRESENTACIÓN GRÁFICA
    GIE=0;

    menu0=0;
    submenu1=ERROR;

    LCD_Inicializacion();
    LCD_Menu6();
    LCD_Grafica(1);

    // Primero se lee el índice de la curva
    Memoria_Dir_Lec(INDICE);
    data=Memoria_Dat_Lec();
    Memoria_Fin();

    // Segundo se pasa a axis los puntos de la curva
    for(i=0; i<((NPUNTOS*2)+2); i++) // Inicializo la matriz
        axis[i]=0;

    Memoria_Dir_Lec(REFEREN+(data*32));

    for(i=0; i<(NPUNTOS*2); i++){
        data=Memoria_Dat_Lec();
        axis[i+2]=((data<<8)|Memoria_Dat_Lec());
    }
    Memoria_Fin();

    // Se pasa a dibujar si es correcto el índice de la curva
    if(axis[2]!=0xFFFF)
        LCD_Redibujar(axis);

    if(interrup==1)
        GIE=1;

    while(Teclado()!=12);
    tecla=0;
    break;
} // fin switch(menu)

switch(submenu1){ // SUBMENÚ DE CONFIGURACIÓN

    case 0: // SELECCIÓN DE RESISTENCIAS

```

```

1036
fila=0;
tecla=0; // Eliminamos la tecla pulsada

LCD_Inicializacion();
LCD_Menu11();
1041

if(resist==0){
    fila=16;
    texto=drstr;
}
1046
else{
    if(resist==1){
        fila=28;
        texto=rsupr;
    }
1051
    else{
        fila=40;
        texto=rinfr;
    }
}
1056
LCD_Escribir(texto, fila, 0,1);

while(tecla!=12){
    // COMPROBACIÓN DE LA TEMPERATURA
    LCD_Temp((7*8),(127-32));
1061

    tecla=Teclado();

    switch(tecla){
1066

        case 10: // GUARDA LA SELECCIÓN DE RESISTENCIA
            LCD_Selec(CMD, TEC_0);
            Espera(2);
            if(fila==16)
                resist=0;
1071
            else{
                if(fila==28)
                    resist=1;
                else
                    resist=2;
1076
            }
            // Se guarda el valor de resist en su posición de EEPROM
            EEADR=RESIS;
            EEDATA=resist;
            EECON1=0x04; // FREE=1; WREN=1
1081
            GIE=0;
            EECON2=0x55;
            EECON2=0x0AA;
            WR=1;
            while(EEIF==0);
1086
            EEIF=0;
            EECON1=0x00;
            if(interrupt==1)
                GIE=1;
1091

            tecla=12;
            menu0=0;
            break;

        case 12:
1096
            LCD_Selec(CMD, TEC_C);

```



```

        Espera(2);
        menu0=0;
    break;
//case 14: // Desplazamiento hacia la derecha del cursor
//break;

//case 15: // Desplazamiento hacia la izquierda del cursor
//break;

case 13: // Desplazamiento hacia arriba del cursor
    LCD_Selec(CMD, TEC_A);
    LCD_Escribir(texto, fila, 0,0);
    switch(fila){
        case 16:
            texto=rinfr;
            fila=40;
            break;

        case 28:
            texto=drstr;
            fila=16;
            break;

        case 40:
            texto=rsupr;
            fila=28;
            break;
    }
    LCD_Escribir(texto, fila, 0,1); // Seleccionamos la nueva fila
    LCD_Selec(CMD, TEC_A);
break;

case 16: // Desplazamiento hacia abajo
    LCD_Selec(CMD, TEC_B);
    LCD_Escribir(texto, fila, 0,0); // Eliminamos el cursor
    switch(fila){
        case 16:
            texto=rsupr;
            fila=28;
            break;

        case 28:
            texto=rinfr;
            fila=40;
            break;

        case 40:
            texto=drstr;
            fila=16;
            break;
    }
    LCD_Escribir(texto, fila, 0,1);
    LCD_Selec(CMD, TEC_B);
break;
} //switch(tecla)
} //while(tecla!=12)
break; // Seleccion de resistencia

case 1: // MENÚ DE SELECCIÓN DE VELOCIDAD
    fila=0;
    tecla=0; // Eliminamos la tecla pulsada

```

```

LCD_Inicializacion(); // Inicializamos la pantalla 1161
LCD_Menu12();

if(veloc==0){
    fila=16;
    texto=vldef; 1166
}
else{
    if(veloc==1){
        fila=28;
        texto=vlmed; 1171
    }
    else{
        fila=40;
        texto=vlmax; 1176
    }
}
LCD_Escribir(texto, fila, 0,1);

while(tecla!=12){
    // COMPROBACIÓN DE LA TEMPERATURA 1181
    LCD_Temp((7*8),(127-32));

    tecla=Teclado();

    switch(tecla){ 1186

        case 10: // GUARDAMOS LA SELECCIÓN DE VELOCIDAD
            LCD_Selec(CMD, TEC_0);
            Espera(2);
            if(fila==16){ 1191
                veloc=0;
                SPBRG=Baud; // Cargamos la velocidad de 9.600.
            }
            else{
                if(fila==28){ 1196
                    veloc=1;
                    SPBRG=Baud1; // Cargamos la velocidad de 19.200.
                }
                else{
                    veloc=2; 1201
                    SPBRG=Baud2; // Cargamos el velocidad de 38.400
                }
            }
            // Se guarda el valor de resist en su posición de EEPROM
            EEADR=VELOC; 1206
            EEDATA=veloc;
            EECON1=0x04; // FREE=1; WREN=1
            GIE=0;
            EECON2=0x55;
            EECON2=0x0AA; 1211
            WR=1;
            while(EEIF==0);
            EEIF=0;
            EECON1=0x00;
            if(interrup==1) 1216
                GIE=1;

            tecla=12;
            menu0=0;

            break; 1221
    }
}

```

```

case 12:
    LCD_Selec(CMD, TEC_C);
    Espera(2);
    menu0=0;
break;

    //case 14:// Desplazamiento hacia la derecha del cursor
    //break;

    //case 15: // Desplazamiento hacia la izquierda del cursor
    //break;

case 13: // Desplazamiento hacia arriba del cursor
    LCD_Selec(CMD, TEC_A);
    LCD_Escribir(texto, fila, 0,0);
    switch(fila){
        case 16:
            texto=vlmax;
            fila=40;
            break;

        case 28:
            texto=vldef;
            fila=16;
            break;

        case 40:
            texto=vlmed;
            fila=28;
            break;
    }
    LCD_Escribir(texto, fila, 0,1);
    LCD_Selec(CMD, TEC_A);
break;

case 16: // Desplazamiento hacia abajo
    LCD_Selec(CMD, TEC_B);
    LCD_Escribir(texto, fila, 0,0);
    switch(fila){
        case 16:
            texto=vlmed;
            fila=28;
            break;

        case 28:
            texto=vlmax;
            fila=40;
            break;

        case 40:
            texto=vldef;
            fila=16;
            break;
    }
    LCD_Escribir(texto, fila, 0,1);
    LCD_Selec(CMD, TEC_B);
break;
    } //switch(tecla)
} //while(tecla!=12)
break;
} //fin switch(submenu1)

```

```

if(iniciar==1){
    GIE=0;
    limite=0;
    LCD_Inicializacion(); // Reinicializamos la pantalla
    LCD_Menu6();

    // Se borra el sector para nuevas capturas
    Borrar_Sector(MUESTRA);

    // Se inicia el precalentamiento
    stado=Precalentar();
    if(stado!=ERROR){

        LCD_Grafica(1);

        // Se guarda el índice de la curva antes de empezar
        Memoria_Dir_Esc(INDICE); // Posición del índice
        Memoria_Dat_Esc(index,1); // Se le pasa el dato
        Memoria_Fin();           // Se cierra la comunicación

        stado=Control(axis, resist);
        if(stado==ERROR){
            // Pitido continuo para indicar que hay un error
            // Y se queda en este hasta que se apague el horno
            // debido a un error de temperatura
            while(1){
                TMR2IF=0;
                while(TMR2IF==0)
                    CCPR1L=((DutC & 0xFFC)>>2);
                CCPR1L=0x00;
            }
        }
        else{
            GIE=1;
            interrup=1;
            LCD_Caracter('_',16,118,0);
            LCD_Caracter('_',24,118,0);
            // Tres pitidos para indicar fin del proceso
            for(i=0;i<3;i++){
                // Realizamos un pitido largo
                Pitido(500);
                // Espera de 1 segundo
                Espera(40);
            }

            LCD_Caracter('@', CMD, 8,0); // Cambio el comando

            while(Teclado()!=10){
                // COMPROBACIÓN DE LA TEMPERATURA
                LCD_Temp(0,(127-32));
            }
        }
        GIE=1;
        interrup=1;
        iniciar=0;
        tecla=0;
    } // Fin de if(iniciar==1)
} // Fin del while(1)
} // Fin del MAIN

```

```

void Configuracion(void){ // Configuración inicial del pic
/* CONFIGURACIÓN DE LOS PUERTOS DEL PIC18F458 */
// Configuración del puerto A para el LCD
TRISA=0x60; // RA0-RA5 son salidas
LATA=0x00; // Limpiamos los latch del puerto A
ADCON1=0x07; // Salidas digitales, no analógicas
PORTA=0x00;

//Configuración del puerto B con pull-up interno para el teclado
TRISB=0xF0; // RB0-RB3 como salidas, RB4-RB7 como entradas
LATB=0x00;
PORTB=0x00;

INTCON=0b00100000;
// GIE      0 (Deshabilitadas las interrupciones) bit7
// PEIE     0 (Deshabilitadas las prioridades)
// TMROIE   1 (Habilitada interrupción por TMR0)
// INTOIE   0 (Deshabilitada interrupción por INTO)
// RBIE     0 (Deshabilitada interrupción por puerto B)
// TMR0IF   0 (Flag de interrupción TMR0)
// INTOIF   0 (Flag de interrupción INTO)
// RBIF     0 (Flag de interrupción puerto B)

/* Todos los bits de configuración que no sean relevantes
o no tengan que declararse se pondrán por defecto a 0 */

INTCON2=0b00000000;
// -RBPU    0 (Habilitado el pull-up interno del puerto B) bit7
// INTEDG0   0 (Selección del flanco de la interrupción exterior 0)
// INTEDG1   0 (Selección del flanco de la interrupción exterior 1)
// --       0
// --       0
// TMR0IP    0 (Prioridad de la interrupción TMR0)
// --       0
// RBIP      0 (Prioridad de la interrupción puerto B)

INTCON3=0b00000000;
// INT2IP    0 (Bit de prioridad de la interrupción exterior 2) bit7
// INT1IP    0 (Bit de prioridad de la interrupción exterior 1)
// --       0
// INT2IE    0 (Deshabilitada la interrupción exterior 2)
// INT1IE    0 (Deshabilitada la interrupción exterior 1)
// --       0
// INT2IF    0 (Flag de interrupción INT2)
// INT1IF    0 (Flag de interrupción INT1)

// Configuración del puerto C para los periféricos
TRISC=0x90;
// Los periféricos sobrescriben los valores de TRISC excepto el PWM
//RC7       1  entrada del puerto de comunicación USART
//RC6       0  salida del puerto de comunicación USART
//RC5       0  salida de los datos de SPI
//RC4       1  entrada de los datos de SPI
//RC3       0  salida del reloj de SPI
//RC2       0  es la salida en PWM para el zumbador
//RC0-RC1   00 son las salidas de los triacs

LATC=0x00; // Limpiamos los latch del puerto C
//PORTC=0x00;

```

```

// Configuración del puerto D para el LCD
TRISD=0x00; // RD0-RD7 son salidas de la palabra del LCD
LATD=0x00; // Limpiamos los latch del puerto D
PORTD=0x00; 1411

// Configuración del puerto E para los dispositivos de SPI
TRISE=0b00000000;
//IBF      0      (bit de estado del buffer de entrada) bit7
//OBF      0      (bit de estado del buffer de salida)
//IBOV     0      (bit de sobreescritura del buffer)
//PSPMODE  0      modo paralelo deshabilitado
// --      0
//RE0-RE2  000    Puerto E como salidas
1421

LATE=0x00; // Limpiamos los latch del puerto E
PORTE=0x07; // El Puerto se pone a 1 porque la
              //selección de los chips en activo en baja

/* CONFIGURACIÓN DEL TEMPORIZADOR TMR0 PARA EL LCD */ 1426
// TRM0 se configurará para operar como temporizador
//que controlará 1 segundo para muestrear
TOCON=0b10000100;
//TMR0ON   1      habilita el temporizador
//T08BIT   0      contador de 16 bits (1 -> 8 bits)
//T0CS     0      Selección de reloj, 0 -> reloj interno
//T0SE     0      Selección de flanco de incremento
//PSA      0      Preescala asignada al TMR0
//TOPS2:0  100    1:32 Preescala seleccionada
1431
1436

/* CONFIGURACIÓN DEL TEMPORIZADOR TMR1 PARA EL TECLADO */
// TRM1 se configurará para operar como un contador síncrono
T1CON=0b10000001;
//RD16     1      Habilita el contador con 16bits
//BIT6      0      no usado
//T1CKPS1:0 00    1:1 pre-escala utilizada
//T10SCEN   0      Deshabilita el oscilador TMR1
//T1SYN     0      sincronización con el oscilador
//TMR1CS    0      Selección de la fuente (0 es interna)
//TMR1ON    1      Habilita el temporizador
1441
1446

/* CONFIGURACIÓN DEL PERIFÉRICO DE PWM */
// Se configura el contador TMR2 para la frecuencia deseada
T2CON=0b00000101; 1451
//bit7      -      no utilizado
//bit6-3    0000    1:1 para la Post-escala
//TMR2ON    1      Activo el contador TMR2
//bit1-0    01      01 es 4 para la Pre-escala
1456

// Configuración del control del periférico CCP1
CCP1CON=0b00001100;
//bit7-6    no se usan
//bit5-4    DC1B1-DC1B0    2 bits menos significativos del PWM
//bit3-0    CCP1M3-CCP1M0  11XX configura el modo PWM
1461

// Introduzco los bits menos significativos de DutC
CCP1CON=(CCP1CON | ((DutC & 0x003)<<4));
// Configuración del Duty Cycle junto con los bits 4-5 del CCP1CON
CCPR1L=((DutC & 0xFFC)>>2);
// Duty-Cicle=[CCPR1L:CCP1CON<5-4>].Tosc.Pre-escala
CCPR1L=0x00; // Se anula para que no pite al arrancar
1466

```

```

// Se calcula el periodo y se guarda en PR2
//Periodo=[(PR2)+1].4.Tosc.Pre-escalaTMR2
PR2=BUZZ; // Se introduce el periodo del PWM
1471

/* CONFIGURACIÓN DEL PUERTO SERIE */

TXSTA=0b00100100;
1476
//CSRC 0 Selección de la fuente del reloj
//TX9 0 Habilitación para transmisión de 8 bits (1=9-bits)
//TXEN 1 Habilitación de transmisión
//SYNC 0 Selección de modo asíncrono (1=síncrono)
//Bit3 0 no usado
1481
//BRGH 1 Velocidad de transmisión (0 -> Baja velocidad)
//TRMT 0 Estado del bit de desplazamiento
//TX9D 0 Flag de 9 bits de datos para transmitir

RCSTA=0b10010000;
1486
//SPEN 1 Habilitación del puerto serie
//RX9 0 Habilitación para recepción de 8 bits (1=9-bits)
//SREN 0 Deshabilitación de recepción simple
//CREN 1 Habilitación de recepción continua
//ADDEN 0 Deshabilita la detección de dirección (para 9 bits)
1491
//FERR 0 Bit de error de ráfaga
//OERR 0 Bit de error de sobreescritura
//RX9D 0 Flag de 9 bits de datos recibidos

SPBRG=Baud; // Cargamos el valor de la velocidad en baudios.
1496

/* CONFIGURACIÓN DEL PERIFÉRICO SPI */

SSPCON1=0b00100000;
1501
//WCOL 0 Bit de detección de colisión en transmisión
//SSPOV 0 No utilizado en modo maestro
//SSPEN 1 Habilitación del puerto serie síncrono
//CKP 0 Estado ocioso de CLK en baja (1 -> ocioso en alta)
//SSPM 0000 Modo maestro con un reloj de Fosc/4.
1506

SSPSTAT=0b11000000;
//SMP 1 Momento del muestreo del bit
//CKE 1 Bit de selección de flanco de reloj (1-> paso de 1 a 0)
//D/-A 0 Se utiliza sólo en modo I2C
//P 0 Se utiliza sólo en modo I2C
1511
//S 0 Se utiliza sólo en modo I2C
//R/-W 0 Se utiliza sólo en modo I2C
//UA 0 Se utiliza sólo en modo I2C
//BF 0 Bit de estado del buffer de recepción
1516

// Se debe reconfigurar la patilla porque la pone
//como entrada el periférico de SPI
TRISA5=0;

/* CONFIGURACIÓN DEL REGISTRO DE CONTROL PARA LA EEPROM */
1521

EECON1=0b00000000;
//EEPGD 0 Selección de memoria que se quiere (0=EEPROM)
//CFGS 0 Acceso a la memoria de datos EEPROM -> 0
//No usado 0
1526
//FREE 0 Se puede borrar y escribir otra vez ->
//WREER 0 Flag de error en escritura
//WREN 0 Habilitación de escritura
//WR 0 Se pone a 1 al iniciar escritura, se borra solo
//RD 0 Se pone a 1 al iniciar lectura, se borra solo
1531

```

```

/* FIN DE LA CONFIGURACIÓN DE LOS PERIFÉRICOS */
return;
}

char Precalentar(){
    unsigned int sensor1, sensor2;
    int offset; // Es la Ta media que existe entre las dos resistencias
    int offset1; // Es la Ta ambiente del sensor superior
    int offset2; // Es la Ta ambiente del sensor inferior
    int i=0;

    // Indicamos en el LCD que se esta precalentando
    LCD_Escribir(precal,24,0,0);

    // Realizamos un precalentamiento del horno hasta los 50 °C
    //que es necesario para que realice bien las curvas de control
    offset1=Temperatura1(); // Obtenemos la temperatura sensor 1
    offset2=Temperatura2(); // Obtenemos la temperatura sensor 2
    offset=(offset1+offset2)/2; // Hayamos la media aritmetica

    while(offset<=OFFSET){

        RCO=1;
        RC1=1;

        offset1=Temperatura1();
        offset2=Temperatura2();
        offset=(offset1+offset2)/2;

        if((Teclado()==12)|| (offset>MAXIMO)){
            RCO=0;
            RC1=0;
            return ERROR;
        }
        // MOSTRAMOS LA TEMPERATURA
        LCD_Numeros(((offset1+offset2)/2),0,(127-32),1);

        i++;
        // Realizamos una espera para tomar nuevos valores
        Espera(10);
    }
    // Fin del precalentamiento de las resistencias
    RCO=0;
    RC1=0;

    if(i<8)
        Espera(80-(i*10));

    // Indicamos en el LCD que se esta precalentando
    LCD_Escribir(finpre,24,0,0);

    return 0;
}

char Control(unsigned int *axis, char resist){

    /* El control que se realiza es un PID, donde el tiempo de muestreo
       es de 1 segundo. El control se aplica sobre un todo-nada
       y se controla el tiempo del con el temporizador TMR0 */

```



```

int offset; // Es la Ta media que existe en el horno
int offset1; // Es la Ta que mide el sensor superior
int offset2; // Es la Ta que mide el sensor inferior
int t; // Número de muestras
int i,n; // Variables para los bucles
unsigned int sensor1, sensor2;
unsigned int max[2]={0,0};
unsigned int pix[2];

// Variables para el control
unsigned int x0,x1,y0,y1;
// Variables de las acciones, el error, el control y la medida
int P, I, D, U, E_k;
unsigned int y; // Valor de la temperatura medida
int Kp=2; // Ganancia de la acción proporcional
int Ki=2; // Ganancia de la acción integral
int Kd=4; // Ganancia de la acción derivativa
int I0=0; // Valor de la acción integral anterior
int E_k0=0; // Valor del error anterior
int pte; // Es la pendiente de la curva teórica
int ref; // Valores que definirán la curva teórica

// Leemos los valores máximos de los puntos (tiempo y temperatura)
for(i=0; i<(((NPUNTOS*2)+2)/2); i++){
    if(axis[i*2]>max[0]) // Máxmio de tiempo
        max[0]=axis[i*2];

    if(axis[1+(i*2)]>max[1]) // Máximo de Temperatura
        max[1]=axis[1+(i*2)];
}

LCD_Numeros(max[0],56,88,0);

// Obtenemos la relación de escalado de los píxeles
// Para el tiempo se disponen de 115 pixeles
// multiplicamos por 10 para aumentar la precisión
if((max[0]+EX_TIEM)>115)
    pix[0]=((max[0]+EX_TIEM)*10)/115;
else
    pix[0]=10;

// Para la temperatura sólo se disponen de 46 pixeles
if(((max[1]-OFFSET)+EX_TEMP)>46)
    pix[1]=(((max[1]-OFFSET)+EX_TEMP)*10)/46;
else
    pix[1]=10;
// Una vez obtenida la relación gráfica se pasa a la representación

//Representamos los puntos de la curva teórica
for(i=0; i<NPUNTOS; i++){
    if(axis[(i*2)+2]!=0){
        // Punto de la curva
        LCD_Pixel(((axis[(i*2)+2]*10)/pix[0]),
            (((axis[(i*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto de la derecha
        LCD_Pixel(((axis[(i*2)+2]*10)/pix[0])+1,
            (((axis[(i*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto de la izquierda
        LCD_Pixel(((axis[(i*2)+2]*10)/pix[0])-1,
            (((axis[(i*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto superior
        LCD_Pixel(((axis[(i*2)+2]*10)/pix[0]),

```

```

        (((axis[(i*2)+3]-OFFSET)*10)/pix[1])+9)+1,0);
// Punto inferior
LCD_Pixel(((axis[(i*2)+2]*10)/pix[0]),
        (((axis[(i*2)+3]-OFFSET)*10)/pix[1])+9)-1,0);
}
else
    i=NPUNTOS;
}
// Se pasa a realizar la curva real

for(n=0;n<NPUNTOS;n++){
    x0=axis[(n*2)]; // x0 es el tiempo del punto inicial
    y0=axis[(n*2)+1]; // y0 es la temperatura de partida
    x1=axis[(n*2)+2]; // x1 es el tiempo del punto final
    y1=axis[(n*2)+3]; // y1 es la temperatura final a alcanzar

    if(x1==0){ // Si x1=0, la curva se ha acabado
        RC1=0;
        RCO=0;
        return 0;
    }
    else{
        // Tomamos el offset
        offset1=Temperatura1();
        offset2=Temperatura2();

        switch(resist){
            case 0: // Con las dos resistencias
                offset=(offset1+offset2)/2; // Hayamos la media aritmetica
                break;

            case 1: // Con la resistencia superior sólo
                offset=offset1; // Solo sensor superior
                break;

            case 2: // Con la resistencia inferior sola
                offset=offset2; // Solo sensor inferior
                break;
        }
        // Si partimos de una temperatura mayor
        //que la real, tomamos la teórica para los cálculos
        if(y0>=offset){
            offset=y0;
        }

        // Calculamos la pendiente de la curva que se va a seguir
        // Multiplicamos la pendiente por 10 para aumentar la precisión
        pte= ((10*(y1-offset))/(x1-x0));

        // Comprobamos que si la pendiente es negativa
        if(y1>y0){
            LCD_Caracter('P',35,115,0);
            Pitido(300);
        }

        // Si la pendiente no es nula, sumamos 1 para aumentar la precisión
        if(pte!=0)
            pte=pte+1;

        t=(x1-x0); // Número de muestras

        // Iniciamos la temporización

```

```

        // Primero se pasa la parte alta
        TMROH =(unsigned char) (((65536-SEGUNDO)& 0xFF00)>>8);
        // Se pasa la parte baja
        TMROL =(unsigned char) ((65536-SEGUNDO)& 0x00FF);
        1721

        for(i=0; i<t;i++){

// La ecuación de la recta es: ref=(pte*(i-x0)/10)+offset;
// Todas las rectas empiezan en x0=0
ref=(pte*i/10)+offset;
        1726

        if(ref>=y1)
            ref=y1;
        1731

        sensor1=Temperatura1();
        sensor2=Temperatura2();
        switch(resist){
            case 0:
                y=(sensor1+sensor2)/2;
                break;
        1736

            case 1:
                y=(sensor1&0x0FFF);
                break;
        1741

            case 2:
                y=(sensor2&0x0FFF);
                break;
        }
        1746

// Se representa en el LCD la curva
LCD_Dibujar(x0, y, i, pix);
// Se actualiza la variable de tiempo
LCD_Numeros((x0+i),56,48,4);
        1751
// Se actualiza la variable de Temperatura
LCD_Numeros(y,0,(127-32),1);

// En caso de superar los 260 °C se parará el programa
if(y>MAXIMO){
    RCO=0;
    RC1=0;
    return ERROR;
}
        1756

// La dirección es: x0+i+(posición inicial del sector)
Memoria_Dir_Esc(MUESTRA+i+x0);
// Se pasa el dato a la memoria
Memoria_Dat_Esc(y&0x00FF,1);
Memoria_Fin(); // Fin de lo comunicación
        1761

//RS232_Envio(y,0);

// Cálculo del control PID
// Tm=1 seg; Tiempo de muestreo
E_k=ref-y; // Error=(referencia - medida)
        1771

P=Kp*E_k; // Proporcional=Kp * Error

D=Kd*(E_k-E_k0); // Derivativa = (Kd/Tm)*(Error-Err_ant.)
        1776

I=Ki*E_k+IO; // Integral = (Error*Tm)*Ki+Int_ant.;

```

```

U=P+I+D;          // Control = Suma de las acciones                                1781

I0=I;             // Integral anterior = Integral actual
E_k0=E_k;         // Error_anterior = Error_actual
// Fin del cálculo del PID

if(Teclado()==12){ //Se comprueba el teclado                                1786
    RC1=0;
    RCO=0;
    return 0;
}                                                                1791

if(U>0){ // Se traduce el control a potencia
    switch(resist){
        case 0: // Encendemos las dos resistencias
            RCO=1;
            RC1=1;
            break;
            1796

        case 1: // Encendemos la resistencia superior
            RCO=1;
            break;
            1801

        case 2: // Encendemos la resistencia inferior
            RC1=1;
            break;
            1806
    }
} else { // Si el control es mayor que cero
    RCO=0;
    RC1=0;
}                                                                1811

// Mostramos las resistencias activas
if(RCO==1)
    LCD_Character('!',16,118,0);
else
    LCD_Character('_',16,118,0);
            1816

if(RC1==1)
    LCD_Character('!',24,118,0);
else
    LCD_Character('_',24,118,0);
            1821

while(TMROIF==0);
    TMROIF=0;
            1826

// Iniciamos la temporización
TMROH =(unsigned char) (((65536-SEGUNDO)& 0xFF00)>>8);
TMROL =(unsigned char) ((65536-SEGUNDO)& 0x00FF);
    } // For de i
} // fin de else
} // For de n
    RC1=0;
    RCO=0;
    return 0;
}                                                                1836

unsigned int Temperatura1(void){

    unsigned int sensor;
            1841

```

```

    REO=0; // Se selecciona el dispositivo
    // Se inicializan los registros
    SSPSTAT=0xC0;
    SSPEN=1; // Habilidad del módulo SSP
1846

    // Comprobación del sensor 1
    SSPBUF=0x00; // Se envia algo para sincronizar el envío
    while(BF==0); // Se espera a que el buffer este lleno
    sensor=SSPBUF; // Leemos los 8 primeros bits enviados
    SSPBUF=0x00; // Se repite la operación para los 8 bits restantes
1851
    while(BF==0);
    // Se desplaza el registro temporal para formar la palabra enviada
    // Se hace una OR para obtener los 16 bits en una variable
    sensor=((sensor<<8)|SSPBUF);
1856

    SSPEN=0;
    REO=1; // Se dehabilita el dispositivo

    /* De los 16bits que mandan los MAX6675, los tres inferiores no
       forman parte de la medida y el primero es el signo pero como está
       preparado para el rango positivo de 0~1023.75°C, siempre es 0 y
       no hace falta evaluarlo, por tanto lo que se hace es desplazar la
       medida tres posiciones hacia la derecha, y para eliminar los
       decimales, se debe desplazar otras dos posiciones a la derecha */
1861
    return ((sensor>>5)&0x00FF);
1866
}

unsigned int Temperatura2(void){
1871

    unsigned int sensor;

    RE1=0;
    // Se inicializan los registros
    SSPSTAT=0xC0;
    SSPEN=1;
1876

    // Comprobación del sensor 1
    SSPBUF=0x00; // Se envia algo para sincronizar el envío
    while(BF==0); // Se espera a que el buffer este lleno
1881
    sensor=SSPBUF; // Leemos los 8 primeros bits enviados
    SSPBUF=0x00; // Se repite la operación para los 8 bits restantes
    while(BF==0);
    // Se desplaza el registro temporal para formar la palabra enviada
    // Se hace una OR para obtener los 16 bits en una variable
1886
    sensor=((sensor<<8)|SSPBUF);
    SSPEN=0;
    RE1=1;

    // Se repite la operación anterior
1891
    return ((sensor>>5)&0x00FF);
}

void RS232_Envio(unsigned int enteros, char tipo){
1896

    // Tipo diferencia entre el envío de muestras o puntos de curva

    // Variable que almacena el valor de los millares
    unsigned int millar=0;
    // Variable que almacena el valor de las centenas
1901
    unsigned int centenas=0;
    // Variable que almacena el valor de las decenas

```

```

unsigned int decenas=0;
// Variable que almacena el valor de las unidades
unsigned int unidades=0;
1906

// Si el valor de la variable enteros es 0xFFFF nos indicará que
// sólo se debe enviar un retorno de carro pues estamos enviando
// los puntos de las curvas existentes en memoria.
if(enteros==0xFFFF){
1911
    while(TXIF==0);
    TXREG = 0xD; // Retorno de carro
    asm("nop");
    return;
}
1916

millar=(enteros)/1000;
centenas=(enteros-(millar*1000))/100;
decenas=(enteros-((millar*1000)+(centenas*100)))/10;
unidades=(enteros-((millar*1000)+(centenas*100)+(decenas*10)));
1921

// Enviamos por el puerto serie la temperatura que marca el sensor
// En ASCII del 48 al 57 son del 0 al 9 respectivamente

if(millar!=0){
1926
    while(TXIF==0);
    TXREG = 48+millar;
    asm("nop");
}
if((centenas!=0)|| (millar!=0)){
1931
    while(TXIF==0);
    TXREG = 48+centenas;
    asm("nop");
}
if((decenas!=0)|| (centenas!=0)|| (millar!=0)){
1936
    while(TXIF==0);
    TXREG = 48+decenas;
}
asm("nop");
while(TXIF==0);
1941
TXREG = 48+unidades;
asm("nop");

if(tipo==0){ // Tipo=0 -> envia un retorno de carro
1946
    while(TXIF==0);
    TXREG = 0xD; // Retorno de carro
    asm("nop");
}
else{ // Tipo!=0 -> envia una tabulación
1951
    while(TXIF==0);
    TXREG = 0x9; // Tabulación
    asm("nop");
}

return;
1956
}

void RS232_Recibo(void){

// Tipo diferencia entre muestras y puntos de curva
1961

// Guardamos los datos introducidos en una matriz de 8*8
unsigned char datos[(8*NPUNTOS)];
// Guardamos el índice de caracteres que tiene cada punto

```

```

unsigned char index[NPUNTOS];
unsigned char aux=0;
char n=0, i=0,t=0;
char tab=0;
long dir=0;
int tiempo[NPUNTOS];
int temp[NPUNTOS];

/* Solo se pueden recibir del PC los datos respectivos a una
   curva, colocados en un documento en el formato siguiente:
           tiempo  temperatura
           tiempo  temperatura
           ....
Solo se admitirán 8 puntos por cada transferencia como máximo,
pues es el número de puntos por curva. Además se comprobarán
que sean correctos antes de guardarlos en memoria. Si hay algún
error se indicará mediante un mensaje visual. La matriz que
recogerá los valores estará compuesta por 8 bytes en cada fila
que guardará en cada posición un caracter, dejando libres
aquellos que no se utilizan, de forma que el caracter de
tabulación nos servirá para saber cuantos dígitos tiene cada
número, siendo obligatorio que el tiempo sea al menos de unidades
y la temperatura tenga como mínimo dos dígitos (decenas y
unidades) sino es así se indicará el error en los datos. */

CREN=1; // Habilita la recepción continua para los 8 puntos

for(i=0;i<NPUNTOS;i++) // Inicializa la matriz de recepción
    tiempo[i]=0;

for(i=0;i<NPUNTOS;i++) // Inicializa la matriz de recepción
    temp[i]=0;

for(i=0;i<(8*NPUNTOS);i++) // Inicializa la matriz de recepción
    datos[i]=0;

i=0;
while(i<NPUNTOS){
    // Temporizador que controla el tiempo de espera de recepción
    TMR1H =(unsigned char) (((65536-ESPERA)& 0xFF00)>>8);
    TMR1L =(unsigned char) ((65536-ESPERA)& 0x00FF);
    t=0;
    while(RCIF==0){
        if(TMR1IF==1){
            TMR1IF=0;
            TMR1H =(unsigned char) (((65536-ESPERA)& 0xFF00)>>8);
            TMR1L =(unsigned char) ((65536-ESPERA)& 0x00FF);
            t++;
            if(t==200){
                //LCD_Inicializacion();
                LCD_Aviso2();
                Espera(100);
                return;
            }
        }
    }
    aux=RCREG;

    if(aux==0x0D){
        index[i++]=n;
        n=0;
    }
}

```

```

    else
        datos[(8*i)+(n++)]=aux;
}
CREN=0; // Deshabilitamos la recepción continua 2031

// Una vez se han recibido todos los datos, se pasan
// de caracteres a números y se guardan en memoria
for(n=0;n<NPUNTOS;n++){
    // Bucle que recorre cada fila y compone caracteres en números 2036

    for(i=0;i<index[n];i++){
        if(datos[(8*n)+i]==0x09) // Buscamos la tabulación
            tab=i;
    } 2041

    switch(tab){ //tab debe estar entre 1~4
        case 1:
            // Se comprueba que al menos existen tres puntos en la gráfica
            if((datos[(n*8)]==48)&&(n<2)){ 2046
                //LCD_Inicializacion();
                LCD_Error();
                Espera(200); // Realizamos una espera de 5 segundos
                return;
            } 2051
            tiempo[n]=(datos[(n*8)]-48); // En ASCII el cero es 48
            break;

            case 2:
                tiempo[n]=((10*(datos[(n*8)]-48))+(datos[(n*8)+1]-48)); 2056
                break;

            case 3:
                tiempo[n]=((100*(datos[(n*8)]-48))+(10*(datos[(n*8)+1]-48))
                    +(datos[(n*8)+2]-48)); 2061
                break;

            case 4:
                tiempo[n]=(1000*(datos[(n*8)]-48)+(100*(datos[(n*8)+1]-48))
                    +(10*(datos[(n*8)+2]-48))+(datos[(n*8)+3]-48)); 2066
                break;

            default:
                //LCD_Inicializacion();
                LCD_Error(); 2071
                Espera(200); // Realizamos una espera de 5 segundos
                return;
            break;
        }
    }
    switch((index[n]-tab)){ // n debe estar entre 4~8 y tab entre 1~4 2076
        case 2:
            // En el caso de introducir una temperatura de un
            // dígito y no ser cero será un error del fichero
            if(datos[(n*8)+tab+1]!=48){
                //LCD_Inicializacion(); 2081
                LCD_Error();
                Espera(200); // Espera de 5 segundos
                return;
            }
            break; 2086

            case 3:
                temp[n]=((10*(datos[(n*8)+tab+1]-48))+(datos[(n*8)+tab+2]-48));

```



```

        break;
2091
    case 4:
        temp[n]=((100*(datos[(n*8)+tab+1]-48))+
                (10*(datos[(n*8)+tab+2]-48))+(datos[(n*8)+tab+3]-48));
        break;
2096
    default:
        //LCD_Inicializacion();
        LCD_Error();
        Espera(200); // Espera de 5 segundos
        return;
2101
    break;
} // Fin switch
} // Fin for n

// UNA VEZ SE TIENEN NUMEROS, SE COMPRUEBAN
2106
for(i=0;i<(NPUNTOS-1);i++){
    if(((tiempo[i]>tiempo[i+1])&&(tiempo[i+1]!=0))||
        (temp[i]>(MAXIMO-10))||(temp[i+1]>(MAXIMO-10))||
        (tiempo[i+1]>9999)){
        //LCD_Inicializacion();
2111
        LCD_Error();
        Espera(200); // Espera de 5 segundos
        return;
    }
}
2116

// COMPROBADOS LOS DATOS, SE PASAN A MEMORIA
// Se busca una posición libre en el espacio reservado
// a las curvas y se recorrerán LAS 32 llevando el cursor
// fuera del espacio si estan ocupadas
2121
for(i=0;i<(NCURVA+1);i++){
    dir=(REFEREN+(i*32));
    Memoria_Dir_Lec(dir);
    if(Memoria_Dat_Lec()==0xFF)
        i=NCURVA;
2126
    Memoria_Fin();
}

// Si la memoria esta llena se avisa
2131
if(dir==(REFEREN+(NCURVA*32))){
    //LCD_Inicializacion();
    LCD_Mem_Full();
    Espera(200);
    return;
}
2136

// Una vez se tiene la dirección de memoria libre,
// se almacenan los puntos dentro de la memoria
Memoria_Dir_Esc(dir);
2141

for(i=0;i<NPUNTOS;i++){
    Memoria_Dat_Esc(tiempo[i],0);
    Memoria_Dat_Esc(temp[i],0);
}
2146

Memoria_Fin();

// Pitido para indicar que se ha finalizado el proceso
Pitido(200);
2151

```

```

    return;
}

void Borrar_Curva(unsigned int *axis, char curva, char paso){
    unsigned char i,n;
    unsigned char t;
    unsigned char aux;

    if(paso==0){
        // PASO=0 --> Se pasan las curvas al espacio de copia
        // menos la que se quiere borrar
        for(n=0;n<NCURVA;n++){
            if(n!=curva){
                Espera(4); // Esperamos 100 ms

                // Leemos las curvas
                Memoria_Dir_Lec(REFEREN+(n*32));

                for(i=0;i<(NPUNTOS*2);i++){
                    aux=Memoria_Dat_Lec();
                    axis[i+2]=((aux<<8)|Memoria_Dat_Lec());
                }
                Memoria_Fin();

                Espera(4); // Esperamos 100 ms

                if(axis[2]!=0xFFFF){
                    // Copiamos la curva al sector 1
                    Memoria_Dir_Esc(COPIACV+(n*32));

                    for(i=0;i<(NPUNTOS*2);i++){
                        Memoria_Dat_Esc(axis[i+2],0);
                    }
                    Memoria_Fin();
                }
                else // Si el dato es 0xFFFF no hay más curvas
                    n=NCURVA;
            } // Si n=curva deja el espacio en memoria
        }
    }
    else{ // PASO = 1 --> Se pasan las curvas al sector principal
        t=0;
        for(n=0;n<NCURVA;n++){
            Espera(4); // Esperamos 100 ms

            // Leemos las curvas
            Memoria_Dir_Lec(COPIACV+(n*32));

            for(i=0;i<(NPUNTOS*2);i++){
                aux=Memoria_Dat_Lec();
                axis[i+2]=((aux<<8)|Memoria_Dat_Lec());
            }
            Memoria_Fin();

            Espera(4); // Esperamos 100 ms

            if(axis[2]!=0xFFFF){
                // Copiamos la curva al sector 1
                Memoria_Dir_Esc(REFEREN+((n-t)*32));
            }
        }
    }
}

```

```

        for(i=0;i<(NPUNTOS*2);i++)
            Memoria_Dat_Esc(axis[i+2],0);
        Memoria_Fin();
    }
    else{
        // Si el primer dato es 0xFFFF, no hay más curvas a copiar
        if(t==0)
            t=1;
        else
            n=NCURVA;
    }
} // Fin for de n
} // Fin else
return;
}

char Teclado(void){

    int i;
    int tempo=0; // Variable de tiempo
    int tecla=0; // Variable de tecla pulsada
    int tecla1=0; // Variable para evitar rebotes

    /* Rutina para comprobar el teclado */
    TMR1H =(unsigned char) (((65536-CUENTA)& 0xFF00)>>8);
    TMR1L =(unsigned char) ((65536-CUENTA)& 0x00FF);
    tempo = 0;

    // Para temporizar 1 segundo se tienen 50 bloques de 20 milisegundos
    while(tecla == 0 && tempo < 50){

        // ESCANEADO DEL TECLADO
        LATB = FILA1; // Se pone 0 en la fila a comprobar
        for(i=0; i<4; i++){ // Bucle de las 4 filas

            if(RB7 == 0) // Primera columna
                tecla = (3*i) + 1;
            else{
                if(RB6 == 0) // Segunda columna
                    tecla = (3*i) + 2;
                else{
                    if(RB5 == 0) // Tercera columna
                        tecla = (3*i) + 3;
                    else{ // Columna de dirección
                        if(RB4 == 0)
                            tecla = i + 13;
                    }
                }
            }
        }
        if(i == 0) // Para la fila 2 -> RB2=0
            LATB = FILA2;
        else{
            if(i == 1) // Para la fila 3 -> RB1=0
                LATB = FILA3;
            else // Para la fila 4 -> RB0=0
                LATB = FILA4;
        }
    }
    // FINAL DEL ESCANEADO

```

```

// Comprobación de un segundo de pulsación
if(TMR1IF==1){
    tempo++;
    TMR1IF = 0;
    TMR1H =(unsigned char) (((65536-CUENTA)& 0xFF00)>>8);
    TMR1L =(unsigned char) ((65536-CUENTA)& 0x00FF);
}
}

if(tempo!=50){ // Se ha pulsado una tecla

    // RETARDO DE 20 MILISEGUNDOS PARA EVITAR REBOTES

    TMR1H =(unsigned char) (((65536-CUENTA)& 0xFF00)>>8);
    TMR1L =(unsigned char) ((65536-CUENTA)& 0x00FF);

    while(!TMR1IF); // Retardo de 20mseg.

    TMR1IF = 0;

    // ESCANEADO DEL TECLADO EVITAR ERRORES POR REBOTE
    LATB = FILA1; // Se pone un 0 en la fila a comprobar -> RB3=0
    for(i=0; i<4; i++){ // Bucle de las 4 filas

        if(RB7 == 0) // Primera columna
            tecla1 = (3*i) + 1;
        else{
            if(RB6 == 0) // Segunda columna
                tecla1 = (3*i) + 2;
            else{
                if(RB5 == 0) // Tercera columna
                    tecla1 = (3*i) + 3;
                else{ // Columna de dirección
                    if(RB4 == 0)
                        tecla1 = i + 13;
                }
            }
        }

        if(i == 0) // Para la fila 2 -> RB2=0
            LATB = FILA2;
        else{
            if(i == 1) // Para la fila 3 -> RB1=0
                LATB = FILA3;
            else // Para la fila 4 -> RB0=0
                LATB = FILA4;
        }

        // FINAL DEL ESCANEADO

        // Pitamos en la pulsación
        for(i=0; i<2; i++){
            TMR2IF=0;
            while(TMR2IF==0)
                CCPR1L=((DutC & 0xFFC)>>2);
            CCPR1L=0x00;
        }

        // ESPERAMOS A QUE SE SUELTE LA TECLA PULSADA

        if(tecla1 == tecla){
            // Entra si se pulsa una tecla y no hay error de rebote

```

```

TMR1H =(unsigned char) (((65536-CUENTA)& 0xFF00)>>8);
TMR1L =(unsigned char) ((65536-CUENTA)& 0x00FF);
tempo = 0;

while(tecla1 != 0 && tempo < 49){
    // Se debe temporizar 1 segundo de espera, como en los rebotes
    // ya se ha esperado 20 mseg. y un segundo son 50 veces ese
    // tiempo, ahora solo debemos contar las 49 veces restantes.

    tecla1=0; // Comprobamos que ya no esta pulsada ninguna tecla

    // ESCANEADO DEL TECLADO
    LATB = FILA1; // Se pone 0 en la fila a comprobar -> RB3=0
    for(i=0; i<4; i++){ // Bucle de las 4 filas
        if(RB7 == 0) // Primera columna
            tecla1 = (3*i) + 1;
        else{
            if(RB6 == 0) // Segunda columna
                tecla1 = (3*i) + 2;
            else{
                if(RB5 == 0) // Tercera columna
                    tecla1 = (3*i) + 3;
                else{ // Columna de dirección
                    if(RB4 == 0)
                        tecla1 = i + 13;
                }
            }
        }
        if(i == 0) // Para la fila 2 -> RB2=0
            LATB = FILA2;
        else{
            if(i == 1) // Para la fila 3 -> RB1=0
                LATB = FILA3;
            else // Para la fila 4 -> RB0=0
                LATB = FILA4;
        }
    }
    // FINAL DEL ESCANEADO

    if(TMR1IF==1){
        tempo++;
        TMR1IF = 0;
        TMR1H =(unsigned char) (((65536-CUENTA)& 0xFF00)>>8);
        TMR1L =(unsigned char) ((65536-CUENTA)& 0x00FF);
    }
} //Fin while

return tecla; // Enviamos la tecla pulsada

return 0; // Enviamos 0, pues no se ha pulsado ninguna tecla
}

void Memoria_Dir_Lec(long dir){

    RE2=0; // Se selecciona el dispositivo del que se va a leer
    // Se inicializan los registros
    SSPSTAT=0x40; // Se debe reconfigurar el periférico
    SSPEN=1;      // Habilitación del módulo SSP

    // LECTURA EN LA MEMORIA
    // Se envia la instrucción

```

```

SSPBUF=READ; // Se envia la instrucción de lectura
while(BF==0); // Se espera a que el buffer este lleno
2401

// Se envía la dirección de escritura (24 bits)
// Primeros 8 bits de la dirección que se quiere leer
SSPBUF=((dir&0x00FF0000)>>16);
while(BF==0); // Se espera a que el buffer este lleno
2406
// Se repite la operación para los 8 bits siguientes
SSPBUF=((dir&0x0000FF00)>>8);
while(BF==0);
// Se repite la operación para los 8 bits últimos
SSPBUF=(dir&0x000000FF);
2411
while(BF==0);

return;
}
2416

int Memoria_Dat_Lec(void){

    int dato=0;

    // Se envían ceros para mantener el reloj,
    // pero se recoge los datos enviados
    SSPBUF=0x00;
    // Se espera a que el buffer este lleno
    while(BF==0);
    dato=SSPBUF;
    2421
    2426

    return dato;
}

void Memoria_Fin(void){
    2431

    SSPEN=0;
    RE2=1; // Se deshabilita el dispositivo

    return;
    2436
}

void Lectura_Puerto(long dir, char tipo){

    // dir = indica la dirección desde la que se lea
    // puntos = nos indicará el número de bytes que se desean leer
    // tipo = nos indica si se quieren leer muestras
    // (1 muestra=1 byte) o puntos de curva (1 punto = 2 bytes)
    2441

    // Cabeceras de la transmisión por el puerto serie
    const unsigned char msj1[]={"Tiempo"};
    const unsigned char msj2[]={"Temperatura"};
    2446

    unsigned int dato=0;
    int i,n;
    2451

    // Se envía primero un encabezado de los datos a transmitir
    for(i=0;i<6;i++){
        while(TXIF==0);
        TXREG = msj1[i]; // Encabezado de la columna de tiempo
        2456
        asm("nop");
    }
    while(TXIF==0);
    TXREG = 0x9; // Tabulación intermedia
    2461
    asm("nop");

```

```

    for(i=0;i<11;i++){
        while(TXIF==0);
        TXREG = msj2[i]; // Encabezado de la columna de Temperatura
        asm("nop");
    }
    while(TXIF==0);
    TXREG = 0xD; // Retorno de carro.
    asm("nop");

    // enviado el encabezado, se pueden transferir los datos.
    Memoria_Dir_Lec(dir);

    if(tipo==0){ // Si tipo es igual a 0 significa que son muestras

        i=0; // Ponemos i a cero para contar el número de muestras
        dato=Memoria_Dat_Lec();

        // Se enviarán datos siempre que exista dato (sea distinto
        //de 0xFF) o no se hayan recorrido el máximo de muestras
        while((dato!=0xFF)&&(i<(CAPTURAS+BUF_T))){

            RS232_Envio(i++,1); // SE ENVIA EL TIEMPO

            RS232_Envio(dato,0); // SE ENVIA LA TEMPERATURA

            dato=Memoria_Dat_Lec();
        }
    }
    else{ // En este caso se estan enviando los puntos
        // de las curvas en el formato: " tiempo tempertarua "
        // Enviará todos los puntos de curvas que existan en la memorira
        for(i=0;i<NCURVA;i++){
            for(n=0;n<(NPUNTOS*2);n++){
                dato=Memoria_Dat_Lec();
                dato=((dato<<8)|Memoria_Dat_Lec());

                if(dato!=0xFFFF){
                    // si el bit menos significativo es distinto de uno,
                    //es un número par y estoy enviando tiempo
                    if((n&0x01)!=1)
                        RS232_Envio(dato,1); // SE ENVIA EL DATO
                    else // En este caso sería un número impar es decir,
                        // es temperatura y envío un retorno de carro
                        RS232_Envio(dato,0); // SE ENVIA EL DATO
                }
            }
            if(dato!=0xFFFF)
                RS232_Envio(0xFFFF,0); // SE ENVIA UN RETORNO DE CARRO
                // PARA INDICAR SEPARACIÓN ENTRE CURVAS
        }
    }

    Memoria_Fin();

    return;

void Memoria_Dir_Esc(long dir){

    RE2=0; // Se selecciona el dispositivo del que se va a leer
    // Se inicializan los registros
    SSPSTAT=0xC0; // Se debe reconfigurar el periférico

```

```

    SSPEN=1; // Habilitación del módulo SSP

    // ESCRITURA EN LA MEMORIA
    // Se envían las instrucciones previas
    SSPBUF=WREN; // Habilitación de escritura
    while(BF==0); // Se espera al buffer lleno
    RE2=1;        // Se selecciona el dispositivo

    RE2=0;
    SSPBUF=PP;    // Programación de página
    while(BF==0); // Se espera a que el buffer este lleno

    // Se envía la dirección de escritura
    SSPBUF=((dir&0x00FF0000)>>16);
    while(BF==0);
    SSPBUF=((dir&0x0000FF00)>>8);
    while(BF==0);
    SSPBUF=(dir&0x000000FF);
    while(BF==0);

    return;
}

void Memoria_Dat_Esc(int muestra, char tipo){
    if(tipo==0){
        // Si tipo es 0 significa que estamos
        // guardando los puntos de una curva
        SSPBUF=((muestra & 0xFF00)>>8);
        while(BF==0);
        SSPBUF=(muestra & 0x00FF);
        while(BF==0);
    }
    // Si tipo es distinto de 0 significa que
    // estamos en el proceso de guardar muestras
    else{
        SSPBUF=(muestra&0x00FF);
        while(BF==0);
    }
}
/*
void Borrado(){
    int i;

    RE2=0; // Se selecciona el dispositivo
    SSPSTAT=0xC0; // Se debe reconfigurar el periférico
    SSPEN=1;        // Habilitación del módulo SSP

    // BORRADO DE LA MEMORIA
    // Se envían las instrucciones previas
    SSPBUF=WREN; // Habilitación de escritura
    while(BF==0); // Se espera a que el buffer este lleno
    RE2=1;        // Se selecciona el dispositivo

    RE2=0;
    SSPBUF=BE;    // Se envía la instrucción de borrado
    while(BF==0); // Se espera a que el buffer este lleno

    SSPEN=0;
    RE2=1; // Se deshabilita el dispositivo

    // Espera de 20 segundos para el borrado de la memoria

```



```

    Espera(800);
    return;
}
*/
void Borrar_Sector(long dir){
    // Se inicializan los registros
    SSPSTAT=0xC0; // Se debe reconfigurar el periférico
    SSPEN=1;      // Habilitación del módulo SSP

    RE2=0; // Se selecciona el dispositivo

    // BORRADO DEL UN SECTOR
    // Se envían las instrucciones previas
    SSPBUF=WREN; // Habilitación de escritura
    while(BF==0); // Se espera a que el buffer este lleno
    RE2=1;      // Se selecciona el dispositivo

    RE2=0;
    SSPBUF=SE; // Se envía la instrucción de borrado
    while(BF==0); // Se espera a que el buffer este lleno

    // Se envía la dirección de borrado
    SSPBUF=((dir&0x00FF0000)>>16);
    while(BF==0);
    SSPBUF=((dir&0x0000FF00)>>8);
    while(BF==0);
    SSPBUF=(dir&0x000000FF);
    while(BF==0);

    SSPEN=0;
    RE2=1; // Se deshabilita el dispositivo

    // Realizamos una espera de 2 segundos para el
    // borrado del sector pero fuera.
    //Espera(80);

return;
}

void Espera(int N){
    // N = multiplica a 25 ms para conseguir un tiempo de espera
    int i;

    for(i=0;i<N;i++){ // Repetimos tantas veces como se indique
        TMR1H=(unsigned char) (((65536-ESPERA)& 0xFF00)>>8);
        TMR1L=(unsigned char) ((65536-ESPERA)& 0x00FF);
        while(TMR1IF==0);
        TMR1IF=0;
    }
    return;
}

void Pitido(int N){
    // N= marcará el tiempo del pitido

    // Realizamos un pitido para indicar que se ha finalizado el proceso
    for(i=0; i<N; i++){
        TMR2IF=0;
        while(TMR2IF==0)

```

```
        CCPR1L=((DutC & 0xFFC)>>2);
    }
    CCPR1L=0x00;
    return;
}

void interrupt interrupts(void){

    TMROIF=0;

    // Iniciamos la temporización
    TMROH =(unsigned char) (((65536-SEGUNDO)& 0xFF00)>>8);
    TMROL =(unsigned char)  ((65536-SEGUNDO)& 0x00FF);

    unsigned int sensor1, sensor2;
    unsigned char data=0;
    // N puntero a la siguiente posición libre del espacio de muestras
    unsigned int n=0;

    // COMPROBACIÓN DE LA TEMPERATURA
    sensor1=Temperatura1();
    sensor2=Temperatura2();
    // FIN DE LECTURA DE TEMPERATURA
    GIE=0;
    Memoria_Dir_Lec(MUESTRA);
    data=Memoria_Dat_Lec();
    while(data!=0xFF){
        n++;
        data=Memoria_Dat_Lec();
    }
    Memoria_Fin();

    if(limite<BUF_T){
        Memoria_Dir_Esc(MUESTRA+n);
        Memoria_Dat_Esc(((sensor1+sensor2)/2),1);
        Memoria_Fin();
        GIE=1;
        limite++;

    }
    else
        interrup=0;

    return;
}
```

---

# Apéndice B

## Archivo cabecera: main.h

```
/** PROGRAMA PARA HORNO DE SOLDADURA POR OLA */
/** PROYECTO FIN DE CARRERA DIRIGIDO POR JESÚS HERNÁNDEZ MANGAS */
/** AUTOR: ALEJANDRO FERNÁNDEZ BLANCO */
/** FECHA: 22-10-2008 */

/** ESTA CABECERA INCLUYE LA DEFINICIÓN DE TODAS LAS VARIABLES */
/** UTILIZADAS EN EL PROGRAMA */

#ifndef _MAIN_H_
#define _MAIN_H_

#define CUENTA 40000 // Cuenta para el teclado de TMR1, 40000=20ms.
#define ESPERA 50000 // Tiempo de espera de TMR1, 50000=25ms.
#define SEGUNDO 62500 // Tiempo de muestreo de 1 segundo para TMR0

#define OFFSET 50 // Temperatura alcanzada al precalentar
#define BUZZ 207 // Valor que proporciona la frecuencia deseada
// BUZZ=[Fosc/4·Fpwm·(pre-escalaTMR2)] -1;
// Fosc=Frecuencia del reloj.
// Fpwm=Frecuencia para la señal PWM
// La preescala es 4 pero puede ser: 1; 4; 16

#define DutC 416 // Tiempo que estará activo el ciclo de PWM
// DutC=(Toc·Fosc)/(pre-escalaTMR2);
// Toc=tiempo de ocupación -> 50% = 0.5/Fpwm

#define Baud 51 // Valor del registro SPBGR.
// Baud=[Fosc/(16·BaudRate)]-1;
// BaudRate=9600
// El error: 0.16%

#define Baud1 25 // Valor del registro SPBGR.
// Baud=[Fosc/(16·BaudRate)]-1;
// BaudRate=19200
// El error: 0.15%

#define Baud2 12 // Valor del registro SPBGR.
// Baud=[Fosc/(16·BaudRate)]-1;
// BaudRate=38400
// El error: 0.15%

#define CS0 0x08 // CS1 y CS2 Deshabilitados
#define CS1 0x28 // Habilita CS1 (RA5)
#define CS2 0x18 // Habilita CS2 (RA4)
#define CS3 0x38 // CS1 y CS2 Habilitados
```

```

#define RST      RA3      // Reset del LCD
#define LCDEN    RA2      // Habilitación del LCD
#define RW       RA1      // Escribimos (0) o leemos (1)
#define DI       RA0      // Dato (1) o Instrucción (0)

// El LCD muestra en pantalla el contenido de la RAM
#define ON       0b00111111
// El LCD no muestra en pantalla el contenido de la RAM
#define OFF      0b00111110

// Indica la fila que se evalúa en el teclado
#define FILA1    0b00000111
#define FILA2    0b00001011
#define FILA3    0b00001101
#define FILA4    0b00001110

// Número máximo de curvas que se tendrán
#define NCURVA   32
// Número máximo de puntos por curva ( tiempo, temperatura)
#define NPUNTOS  8

#define TEC_0    64      // Columna del botón Aceptar
#define TEC_C    80      // Columna del botón Cancelar
#define TEC_D    48      // Columna del botón Derecha
#define TEC_I    32      // Columna del botón Izquierda
#define TEC_A    16      // Columna del botón Arriba
#define TEC_B    0       // Columna Del botón Abajo
#define CMD      56      // Página de los comandos
#define EDI      48      // Página de la edición de puntos

// Posición EEPROM de la variable de resistencia
#define RESIS    0
// Posición EEPROM para la variable de velocidad
#define VELOC    1

// Tamaño de las muestras extras que se mostrarán por el LCD
#define BUF_T    300
// Máxima temperatura que se puede alcanzar en el horno
#define MAXIMO   260
// Número máximo de capturas (10000 seg.)
#define CAPTURAS 10000
// 8puntos*4bytes*32curvas=1024bytes leídos de dos en dos -> 512;
#define CURVAS   512
// Define el valor que se devuelve si hay un error
#define ERROR    55
// Temperatura extra representada en la gráfica
#define EX_TEMP  20
// Tiempo extra representado en la gráfica
#define EX_TIEM  100

// Instrucciones de la memoria M25P16
#define PP       0x02    // Programar Pagina
#define READ     0x03    // Lectura
#define RDRS     0x05    // Lectura Registro Estado
#define WREN     0x06    // Habilitar escritura
#define SE       0xD8    // Borrar Sector
#define BE       0xC7    // Borrado Completo

// Posición de memoria para guardar las muestras
#define MUESTRA  0x00020000
// Posición de memoria para el índice de la última curva ejecutada.
#define INDICE    0x0002FFFF

```

```

// Posición de memoria donde se hará una copia de las curvas 108
#define COPIACV 0x00010000
// Posición de memoria inicial para guardar las curvas
#define REFEREN 0x00000000

/** TEXTOS DE LOS MENÚS DE PANTALLA */ 113

// Menú de Entrada o bienvenida
const unsigned char bvn1 []={ ">_HORNO_SMD_<";
const unsigned char bvn2 []={ "_Octubre_2008_";
const unsigned char bvn3 []={ "_Autor_PFC:_"; 118
const unsigned char bvn4 []={ "Alejandro_Fdez.";
const unsigned char bvn5 []={ "_Tutor:_";
const unsigned char bvn6 []={ "Jesus_M._Hdez._";

// Menú principal del horno 123
const unsigned char cab0 []={ "HORNO_SMD_2008";
const unsigned char conf []={ "Configurar_Horno";
const unsigned char curv []={ "Seleccion_Curva_";
const unsigned char trns []={ "Transferencia_PC";
const unsigned char proc []={ "Grafica_guardada"; 128

//      -      -> Botón de dirección hacia abajo
//      ^      -> Botón de dirección hacia arriba
//      {      -> Botón de dirección hacia la izquierda
//      }      -> Botón de dirección hacia la derecha 133
//      @      -> Boton de Aceptación
//      X      -> Botón de Cancelación
// Existen 5 tipos diferentes de lineas de comandos
// En preguntas antes de realizar algo se tiene el cmd2
// En el menú en el que se edita la curva se tiene cmd3
// En el menú de edición de los puntos se tiene cmd4 138
const unsigned char cmd0 []={ "_^_{}_@_X_";
const unsigned char cmd1 []={ "_^_{}_@_X_";
const unsigned char cmd2 []={ "_____@_X_";
const unsigned char cmd3 []={ "_^_{}_@_X_"; 143
const unsigned char cmd4 []={ "_____{}_@_X_";

// Menú 1 de Configuración del horno
// El número del menú principal aparece entre el título
const unsigned char cab1 []={ "HORNO_SMD_1_2008"; 148
const unsigned char rstr []={ "Tipo_Resistencia";
const unsigned char port []={ "Velocidad_Puerto";

// Menú 2 de Selección de curva
const unsigned char cab2 []={ "HORNO_SMD_2_2008"; 153
const unsigned char tabl []={ "id_{}_tmax_{}_Tmax_";
const unsigned char t1 []={ "t=";
const unsigned char T1 []={ "T=";

// Submenú 2 de Borrado de curva 158
const unsigned char cab21 []={ "BORRADO_2_2008";
const unsigned char borr1 []={ "_Se_va_a_borrar_";
const unsigned char borr2 []={ "_la_curva_";
const unsigned char borr3 []={ "_Borrando..._"; 163

// Submenú 2 de Edición de curva
const unsigned char cab22 []={ "EDITANDO_2_2008";
const unsigned char prgn2 []={ "_Ejecutar_curva_";

// Menú 3 de Transferencia al PC [ = -> (equivale a una flecha) 168
const unsigned char cab3 []={ "HORNO_SMD_3_2008";

```

```

const unsigned char prpc []={"_Proceso_ _PC_"};
const unsigned char hrpc []={"Curva_ Horno_ _PC_"};
const unsigned char pchr []={"Curva_ PC_ _Horno_"};

// Submenú 1 de Tipo de resistencia
const unsigned char cab11 []={"TIPO_ RES. _1_2008"};
const unsigned char drstr []={"Dos_ Resistencias"};
const unsigned char rsupr []={"Solo_ Superior_"};
const unsigned char rinfr []={"Solo_ Inferior_"};

// Submenú 1 de Velocidad del puerto serie
const unsigned char cab12 []={"VELOCIDAD_1_2008"};
const unsigned char vlmax []={"RS-232_ _38.400"};
const unsigned char vlmed []={"RS-232_ _19.200"};
const unsigned char vldef []={"RS-232_ _9.600"};

// Submenú 3 de Confirmación de transferencia // & == ?
const unsigned char cab31 []={"PUERTO_ PC_3_2008"};
const unsigned char velc1 []={"_V_ _9.600_"};
const unsigned char velc2 []={"_V_ _19.200_"};
const unsigned char velc3 []={"_V_ _38.400_"};
const unsigned char prgn3 []={"_&_ Iniciar_?_"};

// Pantalla de proceso
const unsigned char proces []={"PROCESO_ _T=_"};
const unsigned char precal []={"_Precalentando_"};
const unsigned char finpre []={"_"};
const unsigned char cmdpro []={"_X_ _T= _/ _"};

// Mensajes de transición
const unsigned char vacio []={"_"};

// La memoria de curvas esta llena
const unsigned char full1 []={"La_ memoria_ _esta_"};
const unsigned char full2 []={"llena, _borre_ una"};
const unsigned char full3 []={"curva_ para_ poder"};
const unsigned char full4 []={"continuar..."};

// Los datos que se han pasado por el puerto serie no son correctos
const unsigned char err1 []={"Error_ en_ proceso"};
const unsigned char err2 []={"_ Los_ datos_ de _"};
const unsigned char err3 []={"la_ curva_ no_ son_"};
const unsigned char err4 []={"correctos..."};

const unsigned char esp1 []={"_ En_ espera..."};
const unsigned char esp3 []={"_ Tiempo_ de _"};
const unsigned char esp4 []={"_ espera_ agotado_"};

/** FUNCIONES EXISTENTES EN EL PROGRAMA **/

// FUNCIONES DE CONFIGURACIÓN Y PERIFÉRICOS DEL SISTEMA
// Rutina que configura todos los puertos y perifericos del micro
extern void Configuracion(void);
// Rutina que escanea el teclado en busca de una pulsación
extern char Teclado(void);
// Rutina de lectura de la temperatura que da el integrado MAX6675
extern unsigned int Temperatura1(void);
// Rutina de lectura de la temperatura que da el integrado MAX6675
extern unsigned int Temperatura2(void);

// FUNCIONES DE TRANSMISIÓN PUERTO SERIE

```

```

// Envia los datos de tempertarua y tiempo por el puerto serie
extern void RS232_Envio(unsigned int, char);
// Recibe las curvas por el puerto serie
extern void RS232_Recibo();

// FUNCIONES QUE SE UTILIZAN PARA VISUALIZAR EN EL LCD
// Función que hace que el LCD muestre lo que hay en la RAM
extern void LCD_ON(void);
// Función que hace que el LCD no muestre lo que hay en la RAM
extern void LCD_OFF(void);
// Función que situa la pagina de la memoria RAM
extern void LCD_Pagina(unsigned char);
// Función que situa la columna de la memoria RAM
extern void LCD_Columna(unsigned char);
// Rutina que inicializa el LCD, borrando toda la RAM
extern void LCD_Inicializacion(void);
// Función que muestra el menú de bienvenida del horno
extern void LCD_Bienvenida(void);
// Funcion que pinta la pantalla inicial del primer menu
extern void LCD_Menu0(void);
// Funcion que pinta la pantalla inicial del menu de la opción 1
extern void LCD_Menu1(void);
// Funcion que pinta la pantalla del submenú de la opción 1
extern void LCD_Menu11(void);
// Funcion que pinta la pantalla del submenú de la opción 1
extern void LCD_Menu12(void);
// Funcion que pinta la pantalla inicial del menu de la opción 2
extern void LCD_Menu2(void);
// Funcion que pinta la pantalla de confirmación de borrar
extern void LCD_Menu21(void);
// Funcion que pinta la pantalla final del submenu de la opción 2
extern void LCD_Menu211(void);
// Funcion que pinta la pantalla de editar
extern void LCD_Menu22(void);
// Funcion que pinta la pantalla inicial del menu de la opción 3
extern void LCD_Menu3(void);
// Funcion que pinta la pantalla inicial del submenu de la opción 3.1
extern void LCD_Menu31(char, char);
// Funcion que pinta la pantalla del menú de la opción 5
extern void LCD_Menu5(void);
// Funcion que pinta la pantalla del menú de la opción 6
extern void LCD_Menu6(void);
// Función que advierte de un error en los datos
extern void LCD_Error(void);
// Pantalla de advertencia de memoria llena
extern void LCD_Mem_Full(void);
// Aviso que nos indica que curva se va a borrar
extern void LCD_Aviso(void);
// Aviso que indica que se esta borrando la curva seleccionada
extern void LCD_Aviso2(void);
// Funcion que pinta la pantalla con los ejes de la gráfica
extern void LCD_Grafica(char);
// Función que pasa al LCD el byte correspondiente a pintar
extern void LCD_Dato(unsigned char);
// Función que pinta el pixel indicado por la fila y columna
extern void LCD_Pixel(unsigned char, unsigned char, char);
// Función que pinta los caracteres en el LCD
extern void LCD_Caracter(unsigned char, unsigned char,
                        unsigned char, char);
// Función que pinta una cadena de caracteres
extern void LCD_Escribir(const unsigned char*, unsigned char,
                        unsigned char, char);

```

---

```

// Función que muestra números en el LCD (tiempo, temperatura...)
extern void LCD_Numeros(unsigned int, unsigned char,
                        unsigned char, char);
// Función que realiza una linea en el LCD según las coordenadas
extern void LCD_Linea(signed char, signed char, signed char,
                     signed char);
// Función que pinta la curva teorica que se quiere realizar
extern void LCD_Escalado(unsigned int*, unsigned char,
                        unsigned char, char, char);
// Función que invierte el caracter que ya hay en el LCD (crea un cursor)
extern void LCD_Selec(unsigned char, unsigned char);
//Función que muestra la temperatura en el LCD
extern void LCD_Temp(unsigned char, unsigned char);
// Funcion que pinta la curva que se está realizando
extern void LCD_Dibujar(int, int, int, unsigned int*);
// Función que muestra en una lista las curvas con sus valores máximos
extern void LCD_Curvas(char, unsigned int*);
// Función que muestra el punto de la curva que se quiere modificar
extern void LCD_Edicion(char, unsigned int*);
// Funcion que pinta los datos almacenados de la última ejecución
extern void LCD_Redibujar(unsigned int*);

// FUNCIONES QUE SE UTILIZAN PARA TRABAJAR CON LA MEMORIA

// Función que borra la memoria
extern void Borrado();
// Función que borra un sector de memoria
extern void Borrar_Sector(long);
// Función que borra o modifica una curva
extern void Borrar_Curva(unsigned int*, char, char);
// Función que prepara la memoria para leer
extern void Memoria_Dir_Lec(long);
// Función que recibe el dato leído de memoria
extern int Memoria_Dat_Lec(void);
// Función que prepara la memoria para escribir
extern void Memoria_Dir_Esc(long);
// Función que graba los datos en memoria
extern void Memoria_Dat_Esc(int, char);
// Función que cierra la comunicación con la memoria
extern void Memoria_Fin(void);
// Función que lee datos y los envia al puerto serie
extern void Lectura_Puerto(long, char);

// FUNCIONES DE CONTROL DEL HORNO

// Funcion que precalienta el horno hasta 50 °C
extern char Precalentar();
// Funcion que controla la temperatura de la curva a realizar
extern char Control(unsigned int*, char);
// Función que realiza esperas para el programa
extern void Espera(int);
// Función que realiza un aviso sonoro
extern void Pitido(int);
//Función que realiza el retardo para el LCD
extern void Retardo(void);

#endif
/* _MAIN_H_ */

```

---



# Apéndice C

## Código para el LCD: lcd.c

---

```
/** PROGRAMA PARA HORNO DE SOLDADURA POR OLA **/
/** PROYECTO FIN DE CARRERA DIRIGIDO POR JESÚS HERNÁNDEZ MANGAS **/
/** AUTOR: ALEJANDRO FERNÁNDEZ BLANCO **/
/** FECHA: 22-10-2008 */
3

/** FUNCIONES QUE SE REALIZAN CON EL LCD **/

void LCD_ON(void){
8
    // Seleccionamos las dos áreas para activarlas
    DI=0;

    LATA=CS3;      // CS1 y CS2 habilitados
13

    LATD=ON;       // Estado del LCD mostrando la RAM
    asm("nop");
    LCDEN=1;
    asm("nop");
    LCDEN=0;
    asm("nop");
18

    LATA=CS0;      // CS1 y CS2 deshabilitados
    return;
23
}

void LCD_OFF(void){
    // Seleccionamos las dos áreas para activarlas
    DI=0;
28

    LATA=CS3;      // CS1 y CS2 habilitados

    LATD=OFF;      // Estado del LCD mostrando la RAM
    asm("nop");
    LCDEN=1;
    asm("nop");
    LCDEN=0;
    asm("nop");
    LATA=CS0;      // CS1 y CS2 deshabilitados
    return;
33
38
}

void LCD_Pagina(unsigned char pag){
43

    // La palabra de selección de página es: b10111XXX
    // La página se debe sumar: 0~7
    LATD=184+pag; // Seleccionamos la página;
```

```

    LCDEN=1;
    Retardo();
    LCDEN=0;
    return;
}
48

void LCD_Columna(unsigned char col){
53
    // La palabra de selección de columna es: b01XXXXXX
    // La columna se debe sumar: 0~127
    LATD=64+col; // Seleccionamos la columna
    LCDEN=1;
    Retardo();
    LCDEN=0;
    return;
}
58

void LCD_Inicializacion(void){
63
    /* INICIALIZAMOS EL LCD PONIENDO TODOS LOS PUNTOS A CERO */

    int i,j;
68
    LCD_OFF();

    PORTA=0b111000;
    // CS1      1      Seleccionamos el área 1 (RA5)
    // CS2      1      Seleccionamos el área 2
    // RST      1      Señal de reset activa en baja
    // LCDEN 0      Deshabilitamos el lcd
    // RW  0      Indicamos que es una escritura
    // DI  0      Indicamos que es una instrucción
73
    // Para elegir el pixel de comienzo se utiliza: b0000XXXX
    LATD=192+0; // Seleccionamos línea de comienzo;
    LCDEN=1;
    Retardo();
    LCDEN=0;
    Retardo();
83

    LCD_Columna(0); // Repasamos todas las columnas

    for(j=0;j<8;j++){
88
        LCD_Pagina(j);

        // Inicializamos
        DI=1;
93

        for(i=0;i<64;i++){
            LCD_Dato(0);
        }
98

        DI=0;
    }
    return;
}

void LCD_Bienvenida(void){
103
    //LCD_OFF(); // No es necesario

```

```

    LCD_Escribir(bvn1,0,4,0);
    LCD_Escribir(bvn2,9,4,0);
    LCD_Escribir(bvn3,20,4,0);
    LCD_Escribir(bvn4,32,4,0);
    LCD_Escribir(bvn5,44,4,0);
    LCD_Escribir(bvn6,56,4,0);

    LCD_Linea(12,35,84,35);
    LCD_Linea(12,11,52,11);

    LCD_ON();          // Se muestra la RAM

    // Pitido en la entrada al programa
    Pitido(200);

    return;
}

// Menu principal donde se tienen las cuatro opciones principales
void LCD_Menu0(void){

    //LCD_OFF(); // Se inicializa el LCD antes de pintar este menú

    // La cabecera irá siempre en la fila superior (fila=0)
    LCD_Escribir(cab0, 0,0,0);
    LCD_Escribir(conf,16,0,0);
    LCD_Escribir(curv,24,0,0);
    LCD_Escribir(trns,32,0,0);
    LCD_Escribir(proc,40,0,0);
    LCD_Escribir(cmd1,CMD,0,0);
    // La línea de comandos irá siempre en la fila inferior (fila=56)

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;
}

// Menu de configuración del horno, tiene dos selecciones
void LCD_Menu1(){

    //LCD_OFF(); // Se inicializa el LCD antes

    LCD_Escribir(cab1, 0,0,0);
    LCD_Escribir(rstr,20,0,0);
    LCD_Escribir(port,36,0,0);
    LCD_Escribir(cmd1,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;
}

// Selección de las resistencias que se van a emplear
void LCD_Menu11(){

    //LCD_OFF();

```

```

    LCD_Escribir(cab11, 0,0,0);
    LCD_Escribir(drstr,16,0,0);
    LCD_Escribir(rsupr,28,0,0);
    LCD_Escribir(rinfr,40,0,0);
    LCD_Escribir(cmd1,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;
}

// Selección de la velocidad que se va a emplear con el puerto serie
void LCD_Menu12(){

    //LCD_OFF();

    LCD_Escribir(cab12, 0,0,0);
    LCD_Escribir(vldef,16,0,0);
    LCD_Escribir(vlmed,28,0,0);
    LCD_Escribir(vlmax,40,0,0);
    LCD_Escribir(cmd1,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);
    LCD_ON();

    return;
}

// Selección de la curva que se quiere hacer o editar
void LCD_Menu2(void) {

    //LCD_OFF();

    LCD_Escribir(cab2, 0,0,0);
    LCD_Escribir(tab1,10,0,0);
    LCD_Escribir(cmd0,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,45,127,45);
    LCD_Linea(0,9,127,9);
    LCD_Linea(18,9,18,55);
    LCD_Linea(75,9,75,55);

    LCD_ON();

    return;
}

// Menú que indica que la curva va a ser borrada
void LCD_Menu21(void) {

    //LCD_OFF();

    LCD_Escribir(cab21, 0,0,0);
    LCD_Escribir(borr1,24,0,0);
    LCD_Escribir(borr2,32,0,0);
    LCD_Escribir(cmd2,CMD,0,0);

```

```

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;
}

// Menú de confirmación de borrado
void LCD_Menu211(void){

    LCD_Escribir(vacio,24,0,0);
    LCD_Escribir(borr3,32,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);
    return;
}

// Menú de Editar
void LCD_Menu22(void){

    //LCD_OFF();

    LCD_Escribir(cab22, 0,0,0);
    LCD_Escribir(cmd3,CMD,0,0);

    LCD_ON();

    return;
}

// Menú de transferencia de datos con el pc
void LCD_Menu3(void){

    //LCD_OFF();

    LCD_Escribir(cab3, 0,0,0);
    LCD_Escribir(prpc,16,0,0);
    LCD_Escribir(hrpc,28,0,0);
    LCD_Escribir(pchr,40,0,0);
    LCD_Escribir(cmd1,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);
    LCD_ON();

    return;
}

// Menu que pregunta antes de iniciar la transferencia
void LCD_Menu31(char submenu3, char veloc){

    const unsigned char *texto;

    //LCD_OFF();

    LCD_Escribir(cab31,0,0,0);

    switch(submenu3){
        case 0:

```

```

        texto=prpc;
        break;

        case 1:
            texto=hrpc;
            break;

        case 2:
            texto=pchr;
            break;
    }
    LCD_Escribir(texto,16,0,0);

    switch(veloc){
        case 0:
            texto=velc1;
            break;

        case 1:
            texto=velc2;
            break;

        case 2:
            texto=velc3;
            break;
    }

    LCD_Escribir(texto,24,0,0);
    LCD_Escribir(prgn3,40,0,0);
    LCD_Escribir(cmd2,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;
}

// Menú de confirmación de ejecución
void LCD_Menu5(void){

    //LCD_OFF();

    LCD_Escribir(cab22, 0,0,0);
    LCD_Escribir(prgn2,16,0,0);
    LCD_Escribir(prgn3,40,0,0);
    LCD_Escribir(cmd2,CMD,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    LCD_ON();

    return;

}

// Menú de Proceso
void LCD_Menu6(void){

    //LCD_OFF();

```

```

    LCD_Escribir(proces, 0,0,0);
    LCD_Escribir(cmdpro,CMD,0,0);
    LCD_ON();
    return;
}

// Menu que avisa de error producido
void LCD_Error(void){

    //LCD_OFF();

    LCD_Escribir(err1,16,0,0);
    LCD_Escribir(err2,24,0,0);
    LCD_Escribir(err3,32,0,0);
    LCD_Escribir(err4,40,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    //LCD_ON();

    return;
}

void LCD_Mem_Full(void){

    //LCD_OFF();

    LCD_Escribir(full1,16,0,0);
    LCD_Escribir(full2,24,0,0);
    LCD_Escribir(full3,32,0,0);
    LCD_Escribir(full4,40,0,0);

    LCD_Linea(0,55,127,55);
    LCD_Linea(0,9,127,9);

    //LCD_ON();

    return;
}

void LCD_Aviso(void){

    //LCD_OFF();

    LCD_Escribir(vacio,16,0,0);
    LCD_Escribir(esp1,24,0,0);
    LCD_Escribir(vacio,32,0,0);
    LCD_Escribir(vacio,40,0,0);

    //LCD_ON();

    return;
}

void LCD_Aviso2(void){

    //LCD_OFF();

    LCD_Escribir(esp3,24,0,0);

```

```

    LCD_Escribir(esp4,40,0,0);
    //LCD_ON();

    return;
}

// Pinta los ejes de las gráficas
void LCD_Grafica(char tipo){

    char i;

    if(tipo==0){
        for(i=0;i<5;i++){
            LCD_Escribir(vacio,(8*i)+8,0,0);

            LCD_Linea(0,55,0,17);
            LCD_Linea(0,17,127,17);
            LCD_Caracter('t',40,120,0);
        }
    }
    else{
        LCD_Linea(0,55,0,9);
        LCD_Linea(0,9,127,9);

        //Lineas que crean la figura del horno
        LCD_Linea(116,49,127,49);
        LCD_Linea(127,49,127,31);
        LCD_Linea(116,49,116,31);
        LCD_Linea(116,31,127,31);

        LCD_Caracter('t',48,120,0);
    }
    LCD_Caracter('T',8,2,0);

    return;
}

void LCD_Dato(unsigned char pixel){

    LATD=pixel;
    asm("nop");
    LCDEN=1;
    asm("nop");
    LCDEN=0;
    asm("nop");

    return;
}

void LCD_Pixel(unsigned char pixel_x,
               unsigned char pixel_y,char tipo){

    char pag;    // Selecciona la página donde hay que dibujar
    char col;    // Selecciona la columna donde se debe dibujar
    char pixel;  // Indica el pixel que dbe pintarse
    unsigned char antes; // Valor anterior

    pixel=7-((pixel_y)%8);
    // Obtenemos la parte entera de la división (pixel a pintar)
    pag=(7-((pixel_y)/8));
    // El inicio de pantalla está en la página 7, en el pixel 7

```



```

    if(pixel_x>63){
        col=pixel_x-64;
        LATA=CS2;    // Se habilita solo CS2
    }
    else{
        col=pixel_x;
        LATA=CS1;    // Se habilita solo CS1
    }

    // Pintamos el contenido en la celda correspondiente
    DI=0;

    LCD_Pagina(pag);

    LCD_Columna(col);

    LATD=0x00;    // Limpiamos los latch del puerto D
    CMCON=0x07;    // Deshabilitamos el módulo comparador
    TRISD=0xFF;    // RD0-RD7 son ENTRADAS de la palabra del LCD

    RW=1;
    DI=1;
    LCDEN=1;    // Se realiza una lectura tonta,
    Retardo();
    LCDEN=0;
    Retardo();
    LCDEN=1;    // Se realiza la lectura del registro de salida
    Retardo();
    LCDEN=0;
    antes=PORTD;
    Retardo();
    RW=0;
    DI=0;

    TRISD=0x00;    // RD0-RD7 son salidas de la palabra del LCD
    LATD=0x00;    // Limpiamos los latch del puerto D
    PORTD=0x00;
    // Finalizada la lectura reposicionamos el puntero la columna

    if(pixel_x>63){
        LATA=CS2;
    }
    else{
        LATA=CS1;
    }

    LCD_Pagina(pag);

    LCD_Columna(col);

    DI=1;

    if(tipo==0)
        LCD_Dato(antes |(1<<pixel));
    else
        LCD_Dato(255-antes);

    DI=0;

    return;
}

```

```

void LCD_Character(unsigned char character,unsigned char fila,
                  unsigned char columna, char inv){
    543

    /* El caracter intruducido tiene que estar definido en la cabecera
    de fuentes. La fila corresponderá a la coordenada Y que equivale
    a un pixel que irá de 0 a 63 que se divide en páginas (0~7)
    La columna corresponderá a la coordenada X que equivaldrá a un
    548 pixel de 0 a 127.

        00      |
        08      |
        16      |
        24      |
        32      |
        40      |
        48      |
        56      |0-----127
    553
    */

    unsigned char i,n;      // Variable para el bucle for
    unsigned char pag;      // Variable que nos indica la página
    unsigned char col;      // Variable que nos indica la columna
    563 unsigned char pixel;
    unsigned char texto;
    unsigned char div;
    unsigned char antes;
    568

    // Primero: página de la fila
    pag=fila/8;
    // La parte entera de la división nos indicará la página
    pixel=fila%8;
    // El resto indica los pixeles desplazados
    573

    if((pixel!=0)&&(pag<7))
        div=2;
    else
        div=1;
    578

    for(n=0;n<div;n++){
        // Segundo: columna de inicio
        if(columna>63){
            col=columna-64;
            583 LATA=CS2;
        }
        else{
            col=columna;
            588 LATA=CS1;
        }
        }

    DI=0;

    // Tercero: se coloca el puntero de fila y columna
    593 LCD_Pagina(pag+n);

    LCD_Columna(col);

    // Cuarto: se realiza un bucle
    598 for(i=0;i<8;i++){

        texto=(Fuente[(character-32)*8+i]);

        if(inv!=0)
    603

```

```

texto=(255-texto);

if(pixel!=0){
    LATD=0x00; // Limpiamos los latch del puerto D
    CMCON=0x07; // Deshabilitamos el módulo comparador
    TRISD=0xFF; // RD0-RD7 son ENTRADAS del LCD

    RW=1;
    DI=1;
    LCDEN=1; // Se realiza una lectura tonta,
    Retardo();
    LCDEN=0;
    Retardo();
    LCDEN=1; // Lectura del registro de salida
    Retardo();
    LCDEN=0;
    antes=PORTD;
    Retardo();
    RW=0;
    DI=0;

    TRISD=0x00; // RD0-RD7 son salidas de la palabra del LCD
    LATD=0x00; // Limpiamos los latch del puerto D
    PORTD=0x00;
    // Finalizada la lectura volvemos a la columna

    LCD_Pagina(pag+n);

    LCD_Columna(col);

    if(n==0)
        texto=((texto<<pixel)|(antes & (0xFF>>(8-pixel))));
    else
        texto=((texto>>(8-pixel))|(antes & (0xFF<<pixel)));
}

DI=1;

LCD_Dato(texto);

DI=0;

col++;
// Se comprueba la columna
if((col>63)&&(RA5==1)){ // RA5 = CS1
    LATA=CS2;

    col=0;

    LCD_Pagina(pag+n);

    LCD_Columna(col);
}
else{
    if((col>63)&&(RA4==1)){ // RA4=CS2
        LATA=CS1;

        // pag++;
        // if(pag>7)
        //     pag=0;

        col=0;

```

```

        // LCD_Pagina(pag);
        LCD_Columna(col);
    }
    } // FIN else
    } // Fin for de i
} // Fin del for de n
LATA=CS0;    // CS1 y CS2 deshabilitados

return;
}

void LCD_Escribir(const unsigned char *texto, unsigned char fila,
                 unsigned char columna, char tipo){

    int i=0; // Variable para el bucle

    if(tipo==0){ // Si tipo == 0 significa que se hace la letra normal
        while(texto[i]){
            LCD_Caracter(texto[i++],fila,columna,0);
            columna+=8;
        }
    }
    else{ // Si tipo == 1 significa que se hace la letra en video inverso
        while(texto[i]){
            LCD_Caracter(texto[i++],fila,columna,1);
            columna+=8;
        }
    }
    return;
}

void LCD_Numeros(unsigned int numero, unsigned char fila,
                 unsigned char columna, char tipo){

    unsigned char millar=0;
    unsigned char centenas=0;
    unsigned char decenas=0;
    unsigned char unidades=0;

    // SEPARAMOS "MILLARES-CENTENAS-DECENAS-UNIDADES"
    millar =(numero)/1000;
    centenas=(numero-(millar*1000))/100;
    decenas =(numero-((millar*1000)+(centenas*100)))/10;
    unidades=(numero-((millar*1000)+(centenas*100)+(decenas*10)));

    // AHORA PINTAMOS EN EL LCD, EN LA COLUMNA Y LA PÁGINA QUE SE INDIQUE
    // Cero en ASCII es 48

    // Si TIPO = 0 --> tiempo de máximo 4 dígitos y unidad de medida
    // Si TIPO = 1 --> temperatura de 3 dígitos y unidad de medida
    // Si TIPO = 2 --> índice de la curva
    // Si TIPO = 3 --> índice de la curva seleccionado
    // Si TIPO = 4 --> tiempo de máximo 4 dígitos sin unidad de medida

    if((tipo==0)|| (tipo==4)){
        if(millar!=0) // Si millar es distinto de cero se debe pintar
            LCD_Caracter((millar+48), fila, columna,0);
        else
            LCD_Caracter('_', fila, columna,0);
    }
}

```

```

        columna+=8;
    }

    if((tipo!=2)&&(tipo!=3)){
        if((centenas!=0)||(millar!=0))
            LCD_Caracter((centenas+48), fila, columna,0);
        else
            LCD_Caracter('_', fila, columna,0);

        columna+=8;
    }

    if(tipo!=3){
        if((decenas!=0)||(centenas!=0)||(millar!=0)||(tipo==2))
            LCD_Caracter((decenas+48), fila, columna,0);
        else
            LCD_Caracter('_', fila, columna,0);

        columna+=8;
        // Las unidades siempre se deben pintar
        LCD_Caracter((unidades+48), fila, columna,0);
    }
    else{
        LCD_Caracter((decenas+48), fila, columna,1);
        columna+=8;
        // Las unidades siempre se deben pintar
        LCD_Caracter((unidades+48), fila, columna,1);
    }

    if(tipo==0){
        columna+=8;
        LCD_Caracter('s',fila,columna,0);
    }
    else{
        if(tipo==1){
            columna+=8;
            LCD_Caracter('#',fila,columna,0);
        }
    }

    return;
}

void LCD_Linea(signed char X1,signed char Y1,
               signed char X2, signed char Y2){
    /* Bersenham's algorithm
    El caracter intruducido tiene que estar definido en la cabecera
    de fuentes. La fila corresponderá a la coordenada Y que
    equivale a un pixel que irá de 0 a 63 que se divide en páginas
    (0~7) La columna corresponderá a la coordenada X que
    equivaldrá a un pixel de 0 a 127.
    */
    56      |
    48      |
    40      |
    32      |
    24      |
    16      |
    8       |
    0       | 0-----127

```

```

int Dx,Dy,incx,incy,xerr,yerr;
unsigned char distance,t;

Dx = X2 - X1 ;
Dy = Y2 - Y1 ;

if ( Dx > 0 ) incx = 1;
else
if (Dx == 0 ) incx = 0;
else incx = -1;

if ( Dy > 0 ) incy = 1;
else
if (Dy == 0) incy = 0;
else incy = -1;

if(Dy<0) Dy = - Dy;
if(Dx<0) Dx = - Dx;

if ( Dx > Dy ) distance = Dx ;
else distance = Dy ;

yerr = 0;
xerr = 0;

for(t=0;t<=distance+1;t++){
    LCD_Pixel(X1,Y1,0);
    xerr += Dx ;
    yerr += Dy ;

    if ( xerr > distance ){
        xerr -= distance ;
        X1 += incx ;
        if ( X1 < 0 ) X1 = 0 ;
        // if ( X1 > XMAX ) X1 = XMAX ;
    }
    if(yerr > distance ){
        yerr -= distance ;
        Y1 += incy;
        if ( Y1 < 0 ) Y1 = 0 ;
        // if ( Y1 > YMAX ) Y1 = YMAX ;
    }
}
return;
}

void LCD_Escalado(unsigned int *axis, unsigned char tam_x,
                 unsigned char tam_y, char inic_x, char inic_y){

    unsigned int i;
    unsigned int esc[2];
    unsigned int max[2]={0,0};
    unsigned int curva[((NPUNTOS*2)+2)];

    LCD_Grafica(0);

    for(i=0; i<(((NPUNTOS*2)+2)/2); i++){
        if(axis[i*2]>max[0]) // Máxmio de tiempo
            max[0]=axis[i*2];

        if(axis[1+(i*2)]>max[1]) // Máximo de Temperatura
            max[1]=axis[1+(i*2)];

```

```

    }

    // Para el tiempo se disponen de 128 pixeles
    if((max[0]+EX_TIEM)>tam_x)
        esc[0]=((max[0]+EX_TIEM)*10)/tam_x;
    else
        esc[0]=10;

    // Se añaden 20 grados por si hay sobrepico
    if((max[1]!=0)&&(((max[1]-OFFSET)+EX_TEMP)>tam_y))
        esc[1]=(((max[1]-OFFSET)+20)*10)/tam_y;
    else
        esc[1]=10;

    for(i=0;i<(NPUNTOS+1);i++){
        // Calculamos la columna en la que estará el pixel
        curva[(i*2)]=((axis[(i*2)]*10)/esc[0])+inic_x;

        if(axis[(i*2)+1]<OFFSET) // Calculamos la fila que se debe pintar
            curva[(i*2)+1]=inic_y;
        else
            curva[(i*2)+1]=(((axis[(i*2)+1]-OFFSET)*10)/esc[1])+inic_y;

    }

    for(i=0;i<NPUNTOS;i++){
        if(axis[(i*2)+2]!=0)
            LCD_Linea(curva[(i*2)], curva[(i*2)+1],
                      curva[(i*2)+2], curva[(i*2)+3]);
    }

    /*
    for(i=0;i<(NPUNTOS+1);i++){
        RS232_Envio(curva[(i*2)],1);
        RS232_Envio(curva[(i*2)+1],0);
    }
    */
    return;

}

void LCD_Selec(unsigned char fila, unsigned char columna){

    // El cursor que se tiene es de video inverso,
    // del caracter que se selecciona

    unsigned char i; // Variable para el bucle for
    unsigned char pag;

    pag=(7-(fila/8)); //Se ajusta la fila

    for(i=0;i<8;i++){
        LCD_Pixel((columna+i), (pag*8),1);
    }

    return;
}

void LCD_Temp(unsigned char fila, unsigned char columna){

    unsigned int sensor1, sensor2;

    // COMPROBACIÓN DE LA TEMPERATURA
    sensor1=Temperatura1();

```

```

    sensor2=Temperatura2();
    LCD_Numeros(((sensor1+sensor2)/2),fila,columna,1);
    // FIN DE LA COMPROBACIÓN

    return;
}

void LCD_Dibujar(int x0, int y, int j, unsigned int *pix){

    unsigned char pixel_tiempo; // Columna del pixel a pintar
    unsigned char pixel_Temp;   // Fila del pixel a pintar

    // Calculamos la columna en la que estará el pixel
    pixel_tiempo=((x0+j)*10)/pix[0];
    if(pixel_tiempo>115) // limite de tiempo
        return;

    // Calculamos la fila que se debe pintar
    // Se suma 9 para que empiece desde el pixel 9
    pixel_Temp=((y-OFFSET)*10)/pix[1]+9;
    if(pixel_Temp>54) //límite de temperatura
        pixel_Temp=54;

    LCD_Pixel(pixel_tiempo, pixel_Temp,0);

    return;
}

void LCD_Curvas(char avan, unsigned int *axis){

    int i,n;
    unsigned int aux;
    // Máximos de tiempo y Temperatura respectivamente
    unsigned int max[2];

    // Como se tienen cuatro posiciones en el LCD,
    //se recorren las cuatro primeras curvas y se recogen
    //los valores máximos de tiempo y temperatura
    //que serán los representativos de las curvas
    for(n=0;n<4;n++){
        Memoria_Dir_Lec(REFEREN+((n+avan)*32));

        for(i=0;i<(NPUNTOS*2);i++){
            aux=Memoria_Dat_Lec();
            axis[i+2]=((aux<<8)|Memoria_Dat_Lec());
        }
        Memoria_Fin();

        max[0]=0;
        max[1]=0;

        if(axis[2]!=0xFFFF){
            // Se recorre para extraer los máximos
            for(i=0; i<(((NPUNTOS*2)+2)/2); i++){
                if(axis[i*2]>max[0]) // Máxmio de tiempo
                    max[0]=axis[i*2];

                if(axis[1+(i*2)]>max[1]) // Máximo de Temperatura
                    max[1]=axis[1+(i*2)];
            }
        }
    }
}

```



```

    // Una vez se tiene los máxmimos, se pasan al LCD
    LCD_Numeros((n+1+avan), (8*n)+21, 0, 2);
    LCD_Escribir(t1,(8*n)+21, 19,0);
    LCD_Numeros(max[0], (8*n)+21, 35, 0);
    LCD_Escribir(T1,(8*n)+21, 78,0);
    LCD_Numeros(max[1], (8*n)+21, 94, 1);

}

return;
}

void LCD_Edicion(char punto, unsigned int *axis){

    // Se pasan los datos a la pagina 6 que será la indicada (fila 48)
    LCD_Numeros((1+punto), EDI, 0, 2);
    LCD_Caracter('|',EDI, 16,0);
    LCD_Escribir(t1,EDI, 19,0);
    LCD_Numeros(axis[(punto*2)+2], EDI, 35, 0);
    LCD_Caracter('|',EDI, 75,0);
    LCD_Escribir(T1,EDI, 78,0);
    LCD_Numeros(axis[(punto*2)+3], EDI, 94, 1);

    return;
}

void LCD_Redibujar(unsigned int *axis){

    unsigned int n; // Variables para los bucles
    unsigned int max[2]={0,0};
    unsigned int pix[2];
    unsigned char dato, pos;
    unsigned int i=0;

    // Valor máximo de tiempo y temperatura para la relación gráfica
    for(n=0; n<(((NPUNTOS*2)+2)/2); n++){
        if(axis[n*2]>max[0]) // Máxmio de tiempo
            max[0]=axis[n*2];

        if(axis[1+(n*2)]>max[1]) // Máximo de Temperatura
            max[1]=axis[1+(n*2)];
    }

    LCD_Numeros(max[0],56,88,0); // Muestra Tiempo máx.
    LCD_Numeros(max[1],0,(127-32),1);

    // Número de muestras existentes
    // Se vuelve a acceder a la memoria pero esta vez para pintar
    Memoria_Dir_Lec(MUESTRA);

    for(n=0;n<(max[0]+BUF_T);n++){
        dato=Memoria_Dat_Lec();
        if(dato!=0xFF) // Solo pinta valores correctos
            i++;
    }
    Memoria_Fin();

    max[0]=i; // Numero de puntos

    // Obtenemos la relación de escalado de los píxeles
    if((max[0]+EX_TIEM)>115)
        pix[0]=((max[0]+EX_TIEM)*10)/115;

```

```

else
    pix[0]=10;
    // Para la temperatura sólo se disponen de 46 pixeles,
    if(((max[1]-OFFSET)+EX_TEMP)>46)
        pix[1]=(((max[1]-OFFSET)+EX_TEMP)*10)/46;
    else
        pix[1]=10;
    // Con la relación se pasa a pintar

//Representamos los puntos de la curva teórica
for(n=0;n<NPUNTOS;n++){
    if(axis[(n*2)+2]!=0){
        // Punto
        LCD_Pixel(((axis[(n*2)+2]*10)/pix[0]),
            (((axis[(n*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto de la derecha
        LCD_Pixel(((axis[(n*2)+2]*10)/pix[0])+1,
            (((axis[(n*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto de la izquierda
        LCD_Pixel(((axis[(n*2)+2]*10)/pix[0])-1,
            (((axis[(n*2)+3]-OFFSET)*10)/pix[1])+9),0);
        // Punto superior
        LCD_Pixel(((axis[(n*2)+2]*10)/pix[0]),
            (((axis[(n*2)+3]-OFFSET)*10)/pix[1])+9)+1,0);
        // Punto inferior
        LCD_Pixel(((axis[(n*2)+2]*10)/pix[0]),
            (((axis[(n*2)+3]-OFFSET)*10)/pix[1])+9)-1,0);
    }
    else
        n=NPUNTOS;
}
// Una vez estan los puntos de la curva teórica,
//se pasa a realizar la curva practica

Memoria_Dir_Lec(MUESTRA);
dato=Memoria_Dat_Lec();

n=0;
while(dato!=0xFF){
    LCD_Dibujar(0,dato,n++,pix);
    dato=Memoria_Dat_Lec();
    if(dato<OFFSET)
        dato=OFFSET;
}

Memoria_Fin();

return;
}

void Retardo(void){

    asm("nop");
    asm("nop");

    return;
}

```

# Apéndice D

## Cabecera para la fuente del LCD: font.h

---

```
/** PROGRAMA PARA HORNO DE SOLDADURA POR OLA **/
/** PROYECTO FIN DE CARRERA DIRIGIDO POR JESÚS HERNÁNDEZ MANGAS **/
/** AUTOR: ALEJANDRO FERNÁNDEZ BLANCO **/
/** FECHA: 22-10-2008 */
5

/** CARACTERES QUE SE MUESTRAN EN EL LCD **/

#ifndef __FONT_H
#define __FONT_H
10

const bank2 unsigned char Fuente[]=
{
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
20
// Car cter 32 " "

0b00000000,
0b00011000,
0b00011000,
0b00011000,
0b00011000,
0b00011000,
0b00011000,
0b00011000,
0b00000000,
30
//33 !

0b00000000,
0b00000111,
0b00000111,
0b00000000,
0b00000111,
0b00000111,
0b00000000,
0b00000000,
40
//34 "

0b00000010, //0b00010100,
```

```
0b00000101, //0b01111111,
0b00000010, //0b01111111,
0b00000000, //0b00010100,
0b00111110, //0b01111111,
0b01000001, //0b01111111,
0b01000001, //0b00010100,
0b00100010, //0b00000000,
//35 #
45

0b00100100,
0b00101110,
0b01101011,
0b01101011,
0b00111010,
0b00010010,
0b00000000,
0b00000000,
//36 $
55

0b01000110,
0b01100110,
0b00110000,
0b00011000,
0b00001100,
0b01100110,
0b01100010,
0b00000000,
//37 %
65

0b00110000, //0b00110000,
0b01111000, //0b01111010,
0b01001101, //0b01001111,
0b01000101, //0b01011101,
0b01100000, //0b00110111,
0b00100000, //0b01111010,
0b00000000, //0b01001000,
0b00000000, //0b00000000,
//38 &
75

0b00000100,
0b00000111,
0b00000011,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
//39 '
85

0b00000000,
0b00011100,
0b00111110,
0b01100011,
0b01000001,
0000000000,
0b00000000,
0b00000000,
//40 (
95

0b00000000,
0b01000001,
0b01100011,
100

0b00000000,
0b01000001,
0b01100011,
105
```

```

0b00111110,
0b00011100,
0b00000000,
0b00000000,
0b00000000,
//41 )
110

0b00001000,
0b00101010,
0b00111110,
0b00011100,
0b00011100,
0b00111110,
0b00101010,
0b00001000,
//42 *
115

0b00001000,
0b00001000,
0b00111110,
0b00111110,
0b00001000,
0b00001000,
0b00000000,
0b00000000,
//43 +
125

0b00000000, //0b00000000,
0b00000000, //0b10100000,
0b00000000, //0b11100000,
0b00000000, //0b01100000,
0b11111111, //0b00000000,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
//44 ,
135

0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00001000,
0b00000000,
0b00000000,
//45 -
145

0b00000000,
0b00000000,
0b01100000,
0b01100000,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
//46 .
150

0b01100000,
0b00110000,
0b00011000,
0b00001100,
0b00000110,
155

0b01100000,
0b00110000,
0b00011000,
0b00001100,
0b00000110,
160

0b01100000,
0b00110000,
0b00011000,
0b00001100,
0b00000110,
165

```

```
0b000000011,
0b000000001,
0b000000000,
//47 /
```

170

```
0b001111110,
0b011111111,
0b01011001,
0b01001101,
0b011111111,
0b001111110,
0b000000000,
0b000000000,
//48 0
```

175

180

```
0b01000010,
0b01000010,
0b011111111,
0b011111111,
0b01000000,
0b01000000,
0b00000000,
0b00000000,
//49 1
```

185

190

```
0b01100010,
0b01110011,
0b01011001,
0b01001001,
0b01101111,
0b01100110,
0b00000000,
0b00000000,
//50 2
```

195

200

```
0b00100010,
0b01100011,
0b01001001,
0b01001001,
0b01111111,
0b00110110,
0b00000000,
0b00000000,
//51 3
```

205

210

```
0b00011000,
0b00011100,
0b00010110,
0b00010011,
0b01111111,
0b01111111,
0b00010000,
0b00000000,
//52 4
```

215

220

```
0b00100111,
0b01100111,
0b01000101,
0b01000101,
0b01111101,
0b00111001,
0b00000000,
```

225

```

0b00000000 , 230
//53 5

0b00111100 ,
0b01111110 ,
0b01001011 , 235
0b01001001 ,
0b01001001 ,
0b01111001 ,
0b00110000 ,
0b00000000 , 240
//54 6

0b00000011 ,
0b01100011 ,
0b01110001 , 245
0b00011001 ,
0b00001111 ,
0b00000111 ,
0b00000000 ,
0b00000000 , 250
//55 7

0b00110110 ,
0b01111111 ,
0b01001001 , 255
0b01001001 ,
0b01111111 ,
0b00110110 ,
0b00000000 ,
0b00000000 , 260
//56 8

0b00000110 ,
0b01001111 ,
0b01001001 , 265
0b01101001 ,
0b00111111 ,
0b00011110 ,
0b00000000 ,
0b00000000 , 270
//57 9

0b00000000 ,
0b00000000 ,
0b01101100 , 275
0b01101100 ,
0b00000000 ,
0b00000000 ,
0b00000000 ,
0b00000000 , 280
//58 :

0b00000000 ,
0b10100000 ,
0b11101100 , 285
0b01101100 ,
0b00000000 ,
0b00000000 ,
0b00000000 ,
0b00000000 , 290
//59 ;

```

```
0b00001000 ,
0b00011100 ,
0b00110110 ,
0b01100011 ,
0b01000001 ,
0b00000000 ,
0b00000000 ,
0b00000000 ,
//60 <
295

0b00010100 ,
0b00010100 ,
0b00010100 ,
0b00010100 ,
0b00010100 ,
0b00010100 ,
0b00000000 ,
0b00000000 ,
//61 =
300

0b00000000 ,
0b01000001 ,
0b01100011 ,
0b00110110 ,
0b00011100 ,
0b00001000 ,
0b00000000 ,
0b00000000 ,
//62 >
305

0b00000010 ,
0b00000011 ,
0b01010001 ,
0b01011001 ,
0b00001111 ,
0b00000110 ,
0b00000000 ,
0b00000000 ,
//63 ?
310

0b00011000 , //0b00111110 ,
0b00110000 , //0b01111111 ,
0b01100000 , //0b01000001 ,
0b00110000 , //0b01011101 ,
0b00011000 , //0b01011101 ,
0b00001100 , //0b00011111 ,
0b00000110 , //0b00011110 ,
0b00000011 , //0b00000000 ,
//64 @
315

0b01111100 ,
0b01111110 ,
0b00010011 ,
0b00010011 ,
0b01111110 ,
0b01111100 ,
0b00000000 ,
0b00000000 ,
//65 A
320

0b01000001 ,
```



```
0b01111111 ,
0b01111111 ,
0b01001001 ,
0b01001001 ,
0b01111111 ,
0b00110110 ,
0b00000000 ,
//66 B
355
360

0b00011100 ,
0b00111110 ,
0b01100011 ,
0b01000001 ,
0b01000001 ,
0b01100011 ,
0b00100010 ,
0b00000000 ,
//67 C
365
370

0b01000001 ,
0b01111111 ,
0b01111111 ,
0b01000001 ,
0b01100011 ,
0b01111111 ,
0b00011100 ,
0b00000000 ,
//68 D
375
380

0b01000001 ,
0b01111111 ,
0b01111111 ,
0b01001001 ,
0b01011101 ,
0b01000001 ,
0b01100011 ,
0b00000000 ,
//69 E
385
390

0b01000001 ,
0b01111111 ,
0b01111111 ,
0b01001001 ,
0b00011101 ,
0b00000001 ,
0b00000011 ,
0b00000000 ,
//70 F
395
400

0b00011100 ,
0b00111110 ,
0b01100011 ,
0b01000001 ,
0b01010001 ,
0b01110011 ,
0b01110010 ,
0b00000000 ,
//71 G
405
410

0b01111111 ,
0b01111111 ,
0b00001000 ,
415
```

```
0b00001000,
0b01111111,
0b01111111,
0b00000000,
0b00000000,
//72 H
```

420

```
0b00000000,
0b01000001,
0b01111111,
0b01111111,
0b01000001,
0b00000000,
0b00000000,
0b00000000,
//73 I
```

425

430

```
0b00110000,
0b01110000,
0b01000000,
0b01000001,
0b01111111,
0b00111111,
0b00000001,
0b00000000,
//74 J
```

435

440

```
0b01000001,
0b01111111,
0b01111111,
0b00001000,
0b00011100,
0b01110111,
0b01100011,
0b00000000,
//75 K
```

445

450

```
0b01000001,
0b01111111,
0b01111111,
0b01000001,
0b01000000,
0b01100000,
0b01110000,
0b00000000,
//76 L
```

455

460

```
0b01111111,
0b01111111,
0b00000110,
0b00001100,
0b00000110,
0b01111111,
0b01111111,
0b00000000,
//77 M
```

465

470

```
0b01111111,
0b01111111,
0b00000110,
0b00001100,
0b00011000,
```

475

```

0b01111111,
0b01111111,
0b00000000,
//78 N 480

0b00011100,
0b00111110,
0b01100011,
0b01000001,
0b01100011,
0b00111110,
0b00011100,
0b00000000,
//79 O 485

0b01000001,
0b01111111,
0b01111111,
0b01001001,
0b00001001,
0b00001111,
0b00000110,
0b00000000,
//80 P 490

0b00011110,
0b00111111,
0b00100001,
0b01110001,
0b01111111,
0b01011110,
0b00000000,
0b00000000,
//81 Q 505

0b01000001,
0b01111111,
0b01111111,
0b00011001,
0b00111001,
0b01101111,
0b01000110,
0b00000000,
//82 R 510

0b00100110,
0b01100111,
0b01001101,
0b01011001,
0b01111011,
0b00110010,
0b00000000,
0b00000000,
//83 S 515

0b00000011,
0b01000001,
0b01111111,
0b01111111,
0b01000001,
0b00000011,
0b00000000,
535

```

0b00000000 , //84 T	540
0b01111111 , 0b01111111 , 0b01000000 , 0b01000000 , 0b01111111 , 0b01111111 , 0b00000000 , 0b00000000 , //85 U	545
0b00011111 , 0b00111111 , 0b01100000 , 0b01100000 , 0b00111111 , 0b00011111 , 0b00000000 , 0b00000000 , //86 V	555
0b01111111 , 0b01111111 , 0b00110000 , 0b00011000 , 0b00110000 , 0b01111111 , 0b01111111 , 0b00000000 , //87 W	565
0b01100011 , 0b01110111 , 0b00011100 , 0b00001000 , 0b00011100 , 0b01110111 , 0b01100011 , 0b00000000 , //88 X	575
0b00000111 , 0b01001111 , 0b01111000 , 0b01111000 , 0b01001111 , 0b00000111 , 0b00000000 , 0b00000000 , //89 Y	585
0b01100111 , 0b01110011 , 0b01011001 , 0b01001101 , 0b01000111 , 0b01100011 , 0b01110001 , 0b00000000 , //90 Z	595
	600

```

0b00001000, //0b00000000,
0b00001000, //0b01111111,
0b01001001, //0b01111111,
0b01101011, //0b01000001,
0b00111110, //0b01000001,
0b00011100, //0b00000000,
0b00001000, //0b00000000,
0b00000000, //0b00000000,
//91 [
605

0b00000001,
0b00000011,
0b00000110,
0b00001100,
0b00011000,
0b00110000,
0b01100000,
0b00000000,
//92 \
610

0b00000000,
0b01000001,
0b01000001,
0b01111111,
0b01111111,
0b00000000,
0b00000000,
0b00000000,
//93 ]
615

0b00100000, //0b00001000,
0b00110000, //0b00001100,
0b00111000, //0b00000110,
0b00111100, //0b00000011,
0b00111000, //0b00000110,
0b00110000, //0b00001100,
0b00100000, //0b00001000,
0b00000000, //0b00000000,
//94 ^
620

0b00000100, //0b10000000,
0b00001100, //0b10000000,
0b00011100, //0b10000000,
0b00111100, //0b10000000,
0b00011100, //0b10000000,
0b00001100, //0b10000000,
0b00000100, //0b10000000,
0b00000000, //0b10000000,
//95 _
625

0b00000000,
0b00000000,
0b00000011,
0b00000111,
0b00000100,
0b00000000,
0b00000000,
0b00000000,
//96 `
630

0b00100000,
635

640

645

650

655

660

```

```
0b01110100,
0b01010100,
0b01010100,
0b00111100,
0b01111000,
0b01000000,
0b00000000,
//97 a
665

0b01000001,
0b00111111,
0b01111111,
0b01000100,
0b01000100,
0b01111100,
0b00111000,
0b00000000,
//98 b
675
680

0b00111000,
0b01111100,
0b01000100,
0b01000100,
0b01101100,
0b00101000,
0b00000000,
0b00000000,
//99 c
685
690

0b00110000,
0b01111000,
0b01001000,
0b01001001,
0b00111111,
0b01111111,
0b01000000,
0b00000000,
//100 d
695
700

0b00111000,
0b01111100,
0b01010100,
0b01010100,
0b01011100,
0b00011000,
0b00000000,
0b00000000,
//101 e
705
710

0b01001000,
0b01111110,
0b01111111,
0b01001001,
0b00000011,
0b00000010,
0b00000000,
0b00000000,
//102 f
715
720

0b10011000,
0b10111100,
0b10100100,
725
```

```

0b10100100,
0b11111000,
0b01111100,
0b00000000,
0b00000000,
//103 g
730

0b01000001,
0b01111111,
0b01111111,
0b00001000,
0b00000100,
0b01111100,
0b01111000,
0b00000000,
//104 h
735

0b00000000,
0b01000100,
0b01111101,
0b01111101,
0b01000000,
0b00000000,
0b00000000,
0b00000000,
//105 i
740

0b01000000,
0b11000100,
0b10000100,
0b11111101,
0b01111101,
0b00000000,
0b00000000,
0b00000000,
//106 j
745

0b01000001,
0b01111111,
0b01111111,
0b00010000,
0b00111000,
0b01101100,
0b01000100,
0b00000000,
//107 k
750

0b00000000,
0b01000001,
0b01111111,
0b01111111,
0b01000000,
0b00000000,
0b00000000,
0b00000000,
//108 l
755

0b01111100,
0b01111100,
0b00001100,
0b00011000,
0b00001100,
760
765
770
775
780
785

```

```
0b01111100,
0b01111100,
0b00000000,
//109 m
```

790

```
0b01111100,
0b01111100,
0b00000100,
0b00000100,
0b01111100,
0b01111000,
0b00000000,
0b00000000,
//110 n
```

795

800

```
0b00111000,
0b01111100,
0b01000100,
0b01000100,
0b01111100,
0b00111000,
0b00000000,
0b00000000,
//111 o
```

805

810

```
0b10000100,
0b11111100,
0b11111000,
0b10100100,
0b00100100,
0b00111100,
0b00011000,
0b00000000,
//112 p
```

815

820

```
0b00011000,
0b00111100,
0b00100100,
0b10100100,
0b11111000,
0b11111100,
0b10000100,
0b00000000,
//113 q
```

825

830

```
0b01000100,
0b01111100,
0b01111000,
0b01000100,
0b00011100,
0b00011000,
0b00000000,
0b00000000,
//114 r
```

835

840

```
0b01001000,
0b01011100,
0b01010100,
0b01010100,
0b01110100,
0b00100100,
0b00000000,
```

845



```

0b00000000, 850
//115 s

0b00000000,
0b00000100,
0b00111110, 855
0b01111111,
0b01000100,
0b00100100,
0b00000000,
0b00000000, 860
//116 t

0b00111100,
0b01111100,
0b01000000, 865
0b01000000,
0b00111100,
0b01111100,
0b01000000,
0b00000000, 870
//117 u

0b00011100,
0b00111100,
0b01100000, 875
0b01100000,
0b00111100,
0b00011100,
0b00000000,
0b00000000, 880
//118 v

0b00111100,
0b01111100,
0b01100000, 885
0b00110000,
0b01100000,
0b01111100,
0b00111100,
0b00000000, 890
//119 w

0b01000100,
0b01101100,
0b00111000, 895
0b00010000,
0b00111000,
0b01101100,
0b01000100,
0b00000000, 900
//120 x

0b10011100,
0b10111100,
0b10100000, 905
0b10100000,
0b11111100,
0b01111100,
0b00000000,
0b00000000, 910
//121 y

```

```
0b01001100,
0b01100100,
0b01110100,
0b01011100,
0b01001100,
0b01100100,
0b00000000,
0b00000000,
//122 z
915

0b00000000, //0b00001000,
0b00000000, //0b00001000,
0b00001000, //0b00111110,
0b00011100, //0b01110111,
0b00111110, //0b01000001,
0b01111111, //0b01000001,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
//123 {
920

0b00000000, //0b00000000,
0b11111111, //0b00000000,
0b00000000, //0b00000000,
0b00000000, //0b01111111,
0b00000000, //0b01111111,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
//124 |
930

0b00000000, //0b01000001,
0b00000000, //0b01000001,
0b01111111, //0b01110111,
0b00111110, //0b00111110,
0b00011100, //0b00001000,
0b00001000, //0b00001000,
0b00000000, //0b00000000,
0b00000000, //0b00000000,
//125 }
/*
0b00000010,
0b00000011,
0b00000001,
0b00000011,
0b00000010,
0b00000011,
0b00000001,
0b00000000,
//126 ~*/
940

};
#endif
945
950
955
960
```

---

# Bibliografía

- [1] Revista Elektor, enero 2006, artículo "SMD Reflow Soldering Ovenz Revista Elektor, diciembre 2007, artículo Reflow Techniques".
- [2] Datasheet transformador DAGNALL ELECTRONICS LIMITED 7.5VA short circuit proof transformer.
- [3] Datasheet regulador de tensión L7805 (POSITIVE VOLTAGE REGULATORS).
- [4] Datasheet de los triacs AVS10.
- [5] Datasheet optotriacas MOC 3041 (Zero-Cross Optoisolators Triac Driver Output).
- [6] Datasheet LCD MG12064B-SERIES de EVERBOUQUET INTERNATIONAL.
- [7] Datasheet PIC18FXX8 de Microsystems.
- [8] Datasheet MAX6675 [Cold-Junction-Compensated K-Thermocouple-to-Digital Converter (0°C to +1024°C)].
- [9] Datasheet M25P16 (16 Mbit, serial Flash memory, 75 MHz SPI bus interface).
- [10] Datasheet LD1086 SERIES (1.5A low drop positive voltage regulator adjustable and fixed).
- [11] Datasheet MAX232 (+5V-Powered, Multichannel RS-232 Drivers/Receivers).
- [12] Eduardo García Breijo,"Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC".*Marcombo*, 2008.
- [13] [http://es.wikipedia.org/wiki/Proporcional\\_integral\\_derivativo](http://es.wikipedia.org/wiki/Proporcional_integral_derivativo)
- [14] Datasheet 96SCR15AGS84 25G - MULTICORE/LOCTITE.
- [15] Datasheet SN62RA10BAS86-25G SYRINGE.