Sistema autónomo de temperatura

Mateo Sorroche Montellano
Ingeniería Técnica Telecomunicaciones Telemáticas
Sistemas empotrados

Jordi Bécares Ferrés
Pere Tuset Peiró

Junio 2016

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Sistema autónomo de temperatura*
Nombre del autor:	*Mateo Sorroche Montellano*
Nombre del consultor/a:	*Jordi Bécares Ferrés*
Nombre del PRA:	*Pere Tuset Peiró* *Xavi Vilajosana Guillem*
Fecha de entrega (mm/aaaa):	06/2016
Titulación::	*Ingeniería Técnica Telecomunicaciones Telemáticas*
Área del Trabajo Final:	*Sistemas empotrados*
Idioma del trabajo:	*Castellano*
Palabras clave	*Empotrado, control, temperatura*

Resumen del Trabajo:

El presente proyecto se desarrolla en el ámbito de los sistemas empotrados (embebidos). Un sistema empotrado se define como un sistema de computación diseñado para realizar funciones específicas encapsuladas en un dispositivo electrónico. En la actualidad los avances tecnológicos han permitido reducir el tamaño de estos dispositivos y aumentar sus prestaciones, a la vez que se han reducido sus costes de fabricación.

El proyecto nace de la necesidad de optimizar recursos y costes destinados al control y vigilancia en los procesos de control de temperatura. En cualquier proceso industrial o en el ámbito doméstico existen situaciones en la que se requiere un control preciso sobre alguna variable física (temperatura, humedad, iluminación, etc.) de forma autónoma y desatendida. Con él pretendemos sentar una base o punto de partida para resolver este y otros aspectos relacionados, ya que es fácilmente adaptable, configurable y ampliable.

El sistema desarrollado mantiene de forma autónoma y controlada la temperatura de un entorno (planta) en el nivel establecido, de tal forma que puede aplicar calor o disiparlo en función de la necesidad puntual existente para conseguir sus objetivos.

La planta o sistema controlado consta de dos elementos actuadores, el calefactor (con el que aportamos calor al sistema) y el ventilador (con el que disipamos calor del sistema), además de dos sondas de temperatura que monitorizan continuamente la temperatura del bloque calefactor y la temperatura de planta, siendo esta última nuestro referente.

El sistema, a partir de un valor de consigna programable de temperatura llevara a ese punto la temperatura real de planta, con la tolerancia adecuada previamente establecida, reaccionando ante posibles perturbaciones o cambios de consigna y haciendo uso de diferentes algoritmos de control.

El resultado del proyecto es un sistema autónomo de temperatura, capaz de mantener de forma autónoma y precisa la temperatura entre los límites establecidos.

Abstract:

The present project is developed within the field of embedded computing systems. An embedded computing system is defined as a system designed to carry out specific functions encapsulated in an electronic device. Nowadays, the technological advances have reduced the size of these devices and have increased their benefits, as well as they have reduced their manufacturing costs.

This project is a result of the need to optimize resources and costs for the control and monitoring temperature control processes. In any industrial process, even in a domestic environment, there are situations where a precise, autonomous and unattended control over any physical variable (temperature, humidity, lighting, etc.) is required. With this project we intend to provide a basis or a starting point to solve this and other related aspects, as it is easily adaptable, configurable and expandable.

The system developed maintains, in an independent and controlled way, the temperature of an environment (floor) at the level set, so it can be heated or dispelled depending on the specific need to achieve its goals.

The controlled system consists of two actuators: the heater (which provides heat to the system) and the fan (which dissipates heat from the system.) Also, two temperature sensors that continuously monitor the temperature of the heating block and floor temperature, being this our referent.

The system, from a programmable temperature set value will lead to that point the actual floor temperature with adequate tolerance previously established reacting to possible disturbances or set value changes and using different control algorithms.

The result of the project is an autonomous temperature system capable of maintaining autonomously and accurately the temperature within the established limits.

Dedicatoria y agradecimientos

Quiero dedicar este trabajo a la memoria de mi padre, recientemente fallecido (25/04/2016). Que en paz descanse. No hay día que no notemos su ausencia.

Quiero agradecer a todas las personas que me han acompañado durante estos años a lo largo de este camino, pues gracias a ellos nunca he andado solo. Sus ánimos y apoyo me han dado fuerzas suficientes, aun al límite, para concluir este viaje.

Quiero agradecer a todos los profesores, tutores y compañeros que a lo largo de estos años hemos compartido buenos momentos académicos y una gran pasión con el conocimiento.

Quiero agradecer especialmente a mi consultor Jordi Bécares Ferrés, cuya paciencia he puesto a prueba en numerosas ocasiones, y que siempre ha respondido con ánimos e indicaciones constructivas. Gran parte del mérito de concluir esta etapa de lo debo a él.

Quiero agradecer a toda mi familia, especialmente a mis padres y a mis hijos a los que quiero, y a los que no he podido dar toda la atención que se merecen. Su comprensión, paciencia y apoyo me ha dado el coraje suficiente, sobre todo en los momentos más difíciles, para continuar adelante.

Finalmente quiero agradecer a mi pareja Bienve, cómplice, confidente, por la confianza, soporte, ánimos. Por estar ahí. Por reconstruirme.

Gracias a todos

Sinceramente.

Índice

Lista de figuras

1. Introducción.

El presente proyecto se desarrolla en el ámbito de los sistemas empotrados (embebidos). Un sistema empotrado se define como un sistema de computación diseñado para realizar funciones específicas encapsuladas en un dispositivo electrónico. En la actualidad los avances tecnológicos han permitido reducir el tamaño de estos dispositivos y aumentar sus prestaciones, a la vez que se han reducido sus costes de fabricación.

El proyecto consiste en la definición y creación de un sistema autónomo de temperatura controlado a través de un microcontrolador Cortex-M3, integrado en una placa NXP LPC1769.

Nace de la necesidad de optimizar recursos y costes destinados al control y vigilancia en los procesos de control de temperatura. En cualquier proceso industrial o en el ámbito doméstico existen situaciones en la que se requiere un control preciso sobre alguna variable física (temperatura, humedad, iluminación, etc.) de forma autónoma y desatendida. Con él pretendemos sentar una base o punto de partida para resolver este y otros aspectos relacionados, ya que es fácilmente adaptable, configurable y ampliable

El sistema mantiene de forma autónoma y controlada la temperatura de la planta en el nivel establecido, de tal forma que puede aplicar calor o disiparlo en función de la necesidad puntual existente para conseguir sus objetivos.

La planta o sistema controlado consta de dos elementos actuadores, el calefactor (con el que aportamos calor al sistema) y el ventilador (con el que disipamos calor del sistema), además de dos sondas de temperatura que monitorizan continuamente la temperatura del bloque calefactor y la temperatura de planta (Tp), siendo esta última nuestro referente.

El sistema, a partir de un valor de consigna programable de temperatura (Tc) llevara a ese punto la temperatura real de planta (Tp), con la tolerancia adecuada previamente establecida, reaccionando ante posibles perturbaciones o cambios de consigna y haciendo uso de diferentes algoritmos de control.

El resultado es un sistema autónomo de control de temperatura, capaz de mantener de forma autónoma y precisa la temperatura entre los límites establecidos

El sistema es un útil recurso educativo con gran potencial didáctico.

1.1. Contexto y justificación del trabajo

Los sistemas embebidos han experimentado en estos últimos años grandes avances, pasando de ser limitados, costosos y con escasas aplicaciones prácticas a sistemas verdaderamente versátiles y económicos, donde prácticamente aparecen destacados para todo tipo de aplicaciones y en cualquier ámbito.

En este contexto pretendemos demostrar con este trabajo, que el uso de estas tecnologías actuales basadas en sistemas embebidos son idóneas para abordar todo tipo de aplicaciones que hasta hace unos años solo estaban al alcance de sistemas más tradicionales.

Por otro lado, el presente proyecto nace de la necesidad de optimizar recursos y costes destinados al control y vigilancia en los procesos de control de temperatura.

En cualquier proceso industrial o en el ámbito doméstico existen situaciones en la que se requiere un control preciso sobre alguna variable física (temperatura, humedad, iluminación, etc.) de forma autónoma y desatendida. El presente proyecto pretende sentar una base o punto de partida para resolver este y otros aspectos relacionados, ya que es fácilmente adaptable, configurable y ampliable.

El control autónomo y preciso de temperatura (o cualquier otra magnitud física) es muy importante porque permite optimizar costes y recursos (no usamos más recursos de los necesarios, luego el coste es óptimo), que en muchos casos implican incluso la viabilidad económica del proyecto. El ahorro de recursos, en cualquier ámbito, es un aspecto de gran relevancia económica y debe ser considerado prioritariamente en el diseño y desarrollo de proyectos.

En la actualidad este problema se resuelve con equipos muy específicos para aplicaciones muy concretas, limitados y sin posibilidades de ampliación o mejora. Equipos de mucho mayor coste pueden ser algo más versátiles y ofrecer configuración en algunos aspectos, aunque difícilmente ampliación de ningún tipo.

La realización de este trabajo proporciona como punto de partida una aplicación y un prototipo para experimentar sobre diferentes sistemas de control de temperatura, basándonos en el hardware disponible en la propia tarjeta controladora. El resultado obtenido es un sistema que permite de forma autónoma y precisa el control de temperatura.

1.2. Descripción del trabajo

El proyecto consiste en el desarrollo de un prototipo que simulara una planta sobre la que controlar su temperatura y un software de aplicación que realizara dicho control.

La planta o sistema controlado consta de dos elementos actuadores, el calefactor (con el que aportamos calor al sistema) y el ventilador (con el que disipamos calor del sistema), además de dos sondas de temperatura que monitorizan continuamente la temperatura del bloque calefactor y la temperatura de planta (Tp), siendo esta última nuestro referente.

El sistema, a partir de un valor de consigna programable de temperatura (Tc) llevara a ese punto la temperatura real de planta (Tp), con la tolerancia adecuada previamente establecida, reaccionando ante posibles perturbaciones o cambios de consigna y haciendo uso de diferentes algoritmos de control.

La disposición geométrica de la planta sitúa en uno de sus extremos un bloque regulador, con transductor de temperatura y en el extremo opuesto el transductor de temperatura de planta.

El bloque regulador se haya compuesto de los dos sistemas actuadores anteriores:

- Un sistema calefactor de pequeña potencia (alrededor de 50W) basado en la disposición geométrica de varias resistencias cerámicas que permita el paso del aire entre ellas y pueda disipar la potencia (transportar el calor) que generan.
- Un sistema ventilador de pequeña potencia basado en un ventilador de velocidad regulable, que proyecte el aire que impulsa a través de las resistencias calefactoras.

El bloque regulador incorpora además un transductor de temperatura a la salida de las resistencias para monitorizar su temperatura.

Para conseguir sus objetivos, el sistema actuara y activara los diferentes servicios (calefacción y refrigeración) de una manera lógica y proporcional (si se quiere más calor, se activara ventilador y calefacción, si se quiere más frio, solo ventilador), en función de los valores reales obtenidos de las sondas de temperatura y la consigna establecida.

La regulación de la potencia calefactora suministrada a la planta lo conseguimos a través de una señal PWM con la que atacamos la puerta de un transistor mosfet en el circuito de resistencias.

La regulación del caudal de ventilación suministrado a la planta lo conseguimos a través de una señal analógica proporcional a la velocidad de giro con la que atacamos un circuito acondicionador del ventilador.

Como hemos expuesto anteriormente, el sistema permite controlar de forma autónoma el valor de la temperatura de la planta. Para ello una vez inicializado el sistema, el menú principal nos permite selecciones una de las cuatro opciones disponibles en la versión inicial, donde estableceremos el método de control (manual, on/off, bandas o PID). Dentro de su correspondiente menú, la aplicación permite establecer algunos parámetros e iniciar el lazo de control. El sistema regulara la planta para obtener el valor de consigna de temperatura. Durante todo el proceso el sistema envía su estado y los datos adquiridos a una aplicación que permite su grafiado.

1.3. Objetivos del TFC.

El objetivo principal del sistema es controlar la temperatura de una planta basada en un calefactor y un ventilador de forma autónoma a través de un sistema empotrado.

Los objetivos del sistema desarrollado son:

- **Monitorización de las temperaturas del sistema.**

  Ha de estar monitorizada en todo momento la temperatura de las dos sondas que incorpora, la sonda de bloque y la sonda de planta.

- **Monitorización del estado del sistema.**

  Ha de estar monitorizado en todo momento esta estado del sistema, el valor de las salidas y entradas digitales.

- **Control de potencia disipada por el calefactor.**

  Ha de ser controlado en todo momento la potencia suministrada al calefactor.

- **Control de velocidad del ventilador.**

  Ha de ser controlado en todo momento la velocidad de giro del ventilador.

- **Interficie de usuario.**

  La interficie de usuario ha de permitir interactuar con el sistema, monitorizando datos o consignas.

- **Sistema autónomo.**

  La temperatura ha de ser regulada de forma autónoma.

- **Sistema retroalimentado.**

  El sistema optimiza recursos gracias a la retroalimentación.

- **Conectividad inalámbrica, web control**

  El sistema he de permitir su gestión a través de una red inalámbrica o web.

1.4. Enfoque y método seguido.

Para la realización del proyecto y consecución de los objetivos existen diferentes estrategias a seguir, todas ellas validas:

- **Desarrollar un producto nuevo.** Partimos de cero y en función de los objetivos desarrollamos todos cada uno de los componentes adaptados totalmente a nuestras necesidades..
- **Adaptar un producto existente.** Localizamos un producto en el mercado y adaptamos aquellas partes necesarias, dentro de las posibilidades del sistema, para alcanzar nuestros objetivos.
- **Desarrollar un producto nuevo en base a elementos existentes.** Partimos de elementos y bloques funcionales genéricos que adaptamos alcanzar nuestros objetivos y desarrollamos aquellos que sean necesario para completarlo.

En nuestro caso hemos optado por esta última estrategia. Hemos desarrollado un producto nuevo en base a algunos elementos del mercado y desarrollado algunas partes necesarias. Esta estrategia, hibrida entre las dos anteriores es la más adecuada a nuestras necesidades,

en cuanto a que nos permite centrarnos en aquellos aspectos del proyecto en el que realmente damos valor añadido, con un considerable ahorro de esfuerzo y tiempo (al no tener que desarrollar todas las partes íntegramente). Por otro lado, al no ser una adaptación de un producto existente, tenemos más margen de maniobra al no estar tan limitados a la funcionalidad original del sistema. Muchas adaptaciones de productos existentes son situaciones de compromiso entre la funcionalidad original y la funcionalidad proyectada, que pocas veces se suelen resolver de forma exitosa.

En la realización del proyecto, lo abordamos de manera gradual, distinguiendo dos grandes etapas que exponemos a continuación:

- **Documentación y análisis**

En esta primera etapa nos centramos en el conocimiento del sistema sobre el que vamos a desarrollar el proyecto. Recolectamos y estudiamos toda la información posible, vemos aplicaciones similares, investigamos nuevos productos, nos familiarizamos con el sistema y creamos las primeras pruebas y aplicaciones sobre el mismo. Toda la experiencia y el aprendizaje realizado será imprescindible para poder abordar la segunda etapa del proyecto.

En paralelo realizamos una investigación de los materiales necesarios para su realización. En esta ocasión hemos intentado reciclar para su uso todos aquellos que teníamos a nuestro alcance y adquirir tan solo los que no podíamos conseguir por otros medio. Hemos intentado seleccionar siempre los más adecuados para el proyecto, y con total compatibilidad con el sistema embebido.

- **Desarrollo del producto**

En esta segunda etapa entramos de lleno en el ámbito del proyecto.

Desarrollamos los módulos y librería (msorroche_LPC1769_Lib) necesarios para la aplicación (pwm, dac, iodig, etc.) y reciclamos alguno de los existentes (uart, printf, log, etc.), creados en la anterior etapa de documentación.

Desarrollamos la aplicación para el sistema embebido basada en FreeRTOS haciendo uso de los módulos creados e incorporado en la librería, para el control del sistema.

Configuramos la aplicación instalada en el PC con la que nos comunicamos para grafiar los datos.

Configuramos las aplicaciones de terminal para poder general la HMI y poder interactuar con el usuario.

Realizamos el montaje final del prototipo, cableado, conexionado e iniciamos las pruebas finales del sistema.

Paralelamente a todas las fases de esta última etapa iniciamos la redacción de la memoria, que vamos completando, añadiendo o modificando según avanza el proyecto.

1.5. Planificación del trabajo.

Las tareas previstas a realizar inicialmente son las siguientes:

Tarea	Descripción de la tarea	Días
Inicio		
01	Primeros pasos	9
02	Propuesta proyecto	9
03	Arp@Lab y WiFly	7
04	**PAC1 - Plan de trabajo / Planificación**	8
Primera fase		
05	Driver sensor temperatura	7
06	Activación ventilador	7
07	Activación calefactor	7
08	Driver control PID	7
09	Configuración Consignas fijas en código	7
10	HMI vía serie notificación y log	7
11	Leds de estado	7
Segunda fase		
12	Sistema regulación calefactor	7
13	Sistema regulación ventilador	7
14	HMI vía serie programación valores	7
15	Persistencia de datos	7
16	Recuperación valores de fabrica	7
Extras		
17	Conectividad inalámbrica	7
18	Notificación vía mail	7
19	Web control	7
Final		
20	**PAC2 - Seguimiento**	23
21	**PAC3 - Seguimiento código + previa memoria**	28
22	**Código final**	23
23	**Memoria**	44
24	**Presentación**	19

Figura 1.1: Previsión de inicial de tareas.

El tiempo destinado al proyecto será de unas 250 horas, lo que da un promedio de aproximadamente unas 3 horas diarias pasada la etapa de documentación i análisis.

A continuación presentamos el diagrama de Gantt de la previsión de tareas iniciales.

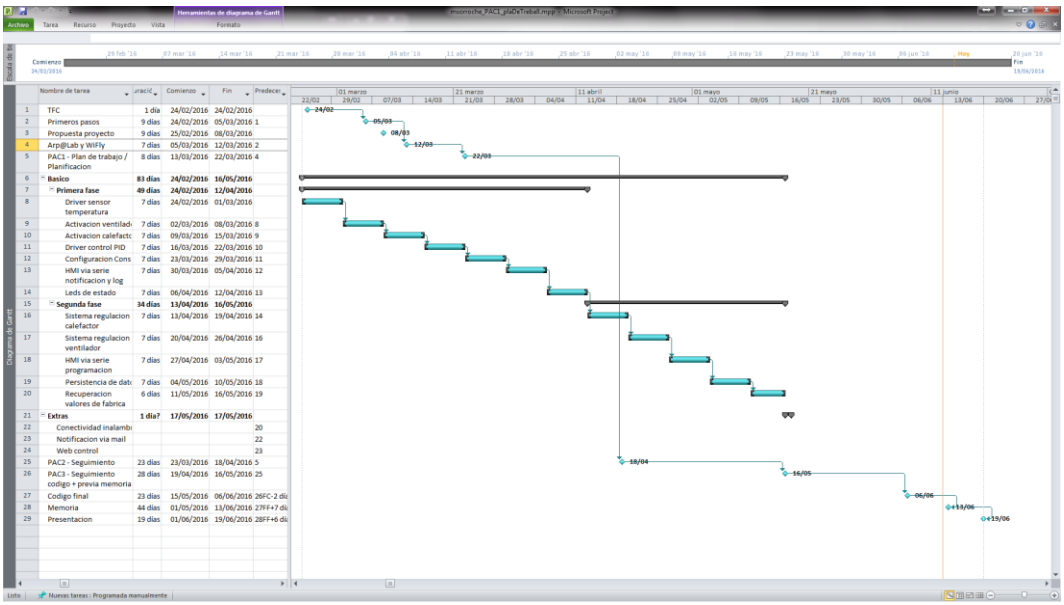


Figura 1.2: Cronograma inicial.

Como veremos a continuación en la relación de tareas y en el cronograma final, durante la realización del proyecto ha habido un retraso importante, debido fundamentalmente varias causas:

- Falta de tiempo por motivos laborales, sobrecarga puntual de trabajo que no me han permitido pese a mis esfuerzos mantener el ritmo de trabajo previsto.
- Motivos personales, lamentablemente mi padre falleció a finales de Abril tras algunas semanas ingresado en el hospital. Algunas semanas antes y después del suceso mi rendimiento y capacidad de trabajo estaba a unos niveles muy inferiores a los normales, lo que dificultaba el seguimiento del ritmo de trabajo previsto.
- Retraso en la entrega de ciertos materiales, que aunque se ha sido previsor y se han comprado con tiempo suficiente, estos han tardado en llegar. Algunos se ha tenido que compras a diferentes suministradores, confiando que alguno de ellos cumplirá lo términos de entrega de mercancía.
- Como tarea colateral del proyecto se ha construido un prototipo donde poder llevar a la práctica y desarrollar los algoritmos de control diseñados: Este prototipo o maqueta ha llegado a tener tal envergadura que prácticamente tiene entidad de trabajo por si sola, y ha reportado una ingente carga de trabajo fuera del ámbito de este proyecto.

A consecuencia de lo expuesto anteriormente fue necesario realizar algunos ajustes en el alcance extra (opcional) del proyecto y en la planificación requerida para poder llevar el proyecto a bien fin.

Paralelamente, parte del retraso se ha compensado por el trabajo extra realizado en las últimas semanas en las que se ha triplicado o cuadriplicado el tiempo destinado inicialmente al desarrollo del proyecto.

En la siguiente figura presentamos las tareas realizadas:

Tarea	Descripción de la tarea	Días
Inicio		
01	Primeros pasos	9
02	Propuesta proyecto	9
03	Arp@Lab y WiFly	7
04	**PAC1 - Plan de trabajo / Planificación**	8
Primera fase		
05	Driver sensor temperatura	7
06	Activación ventilador	7
07	Activación calefactor	7
08	Driver control PID	7
09	Configuración Consignas fijas en código	7
10	HMI vía serie notificación y log	7
11	Leds de estado	7
Segunda fase		
12	Sistema regulación calefactor	7
13	Sistema regulación ventilador	7
14	HMI vía serie programación valores	7
15	Recuperación valores de fabrica	7
Final		
16	**PAC2 - Seguimiento**	23
17	**PAC3 - Seguimiento código + previa memoria**	28
18	**Código final**	16
19	**Memoria**	37
20	**Presentación**	12

Figura 1.3: Tareas realizadas.

A continuación mostramos el cronograma final, adaptado a la realidad del proyecto.

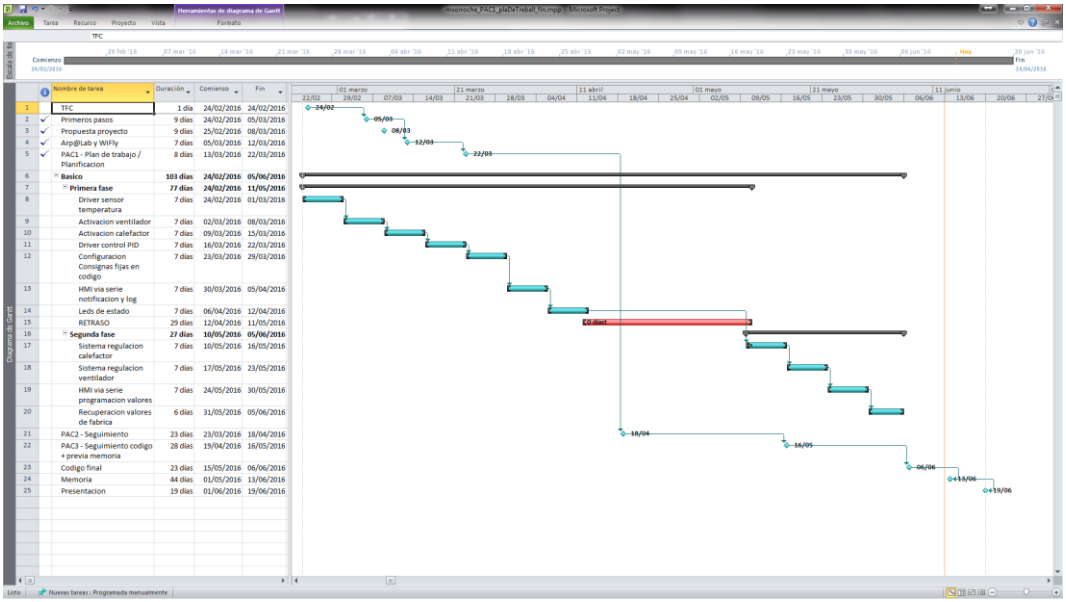


Figura 1.4: Cronograma final.

En el cronograma final podemos observar una tarea grafiada de color rojo que representa el retraso sufrido en el proyecto.

Como hemos comentado anteriormente, el retraso se ha podido recuperar en gran parte a un aumento espectacular de las horas dedicadas finalmente al trabajo en el proyecto.

1.6. Recursos empleados

En la realización de este proyecto se utilizó un ordenador portátil Acer Aspire 5741G con las siguientes características:

- Procesador Intel Core I5-450M (2,4GHz, 3Mb L3 cache)
- Placa grafica NVIDEA GeForce GT 320M
- Pantalla HD LED LCD 15,6”
- 4Gb Memoria RAM
- 640 GB HDD
- DVD Súper Multi DL Driver
- Monitor auxiliar AOC E2752P

El equipo portátil dispone del siguiente software instalado:

- Sistema operativo Windows 7 Home Premium 64 bits, SP1
- Paquete ofimático Microsoft Office Professional Plus 2010

- Entorno IDE LPCXpresso v7.6.2_326
- FreeRTOS
- CMSISv2p00
- Eclipse Plataform v4.4.0.v20140925
- Tera Term v 4.86 de Tera Term Project en OSDN
- Stamp Plot Lite V1.7 de Selmaware Solutions

Se han utilizado los siguientes recursos documentales:

- Sitios Web y folletos de los fabricantes de los diferentes componentes.
- Wiki de la UOC, WIKI: Embedded Systems Lab
- Fuentes documentales presentes en la web, de ámbito universitario.

En la realización del proyecto ha sido necesario el siguiente material

- NXP LPC1769 board (uC Cortex-M3)
- CP2102 (conversor UART-USB)
- Conector de mini-USB a USB
- HUB USB de 4 puertos.
- Sondas de temperatura DS18B20
- Ventilador 80x80
- Resistencias calefactoras dh BC15/15
- Cables de interconexión de pines
- Placa de relés CEBEK T1.
- Leds de señalización de cuatro colores (rojo, azul, amarillo y verde)
- Resistencias de montaje.
- Transistor mosfet irfp450
- Fuente alimentación PC-AT modificada
- Placa de montaje, cajas, armazón, conectores, interruptor, componentes varios para la construcción de la maqueta de prototipo.

1.7. Productos obtenidos

Se han obtenido los siguientes productos:

- Una API que sirve para controlar algunas de las características usadas en el proyecto y otras presentes en la placa LPC1769, tales como conversor DA, PWM, entradas y salidas digitales, control de led, comunicaciones asíncronas UART, protocolo Onewire, control PID, etc. Se ha creado una librería estática **msorroche_LPC1769_Lib** que implementa estas y otras funciones para acceso a periféricos y apoyo a la aplicación, explicadas con más detalle en próximos apartados. Están desarrolladas en lenguaje C sobre el sistema operativo en tiempo real FreeRTOS y hace uso de las librerías **FreeRTOS_library** y ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) **CMSISv2p00_LPC17xx**.
- Un proyecto ejecutable **msorroche_PFC**, que constituye el código principal de la aplicación que controla la temperatura de la planta. Están desarrollado en lenguaje C sobre el sistema operativo en tiempo real FreeRTOS y hace uso de las librerías **FreeRTOS_library** y ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) **CMSISv2p00_LPC17xx**, así como de la librería **msorroche_LPC1769_Lib**.
- Una aplicación freeware configurada para PC (Stamp Plot Lite V1.7 de Selmaware Solutions), que adquiere y representa gráficamente el valor de temperatura de la planta. Los datos son adquiridos a través de un puerto serie del pc a 19200 bps.

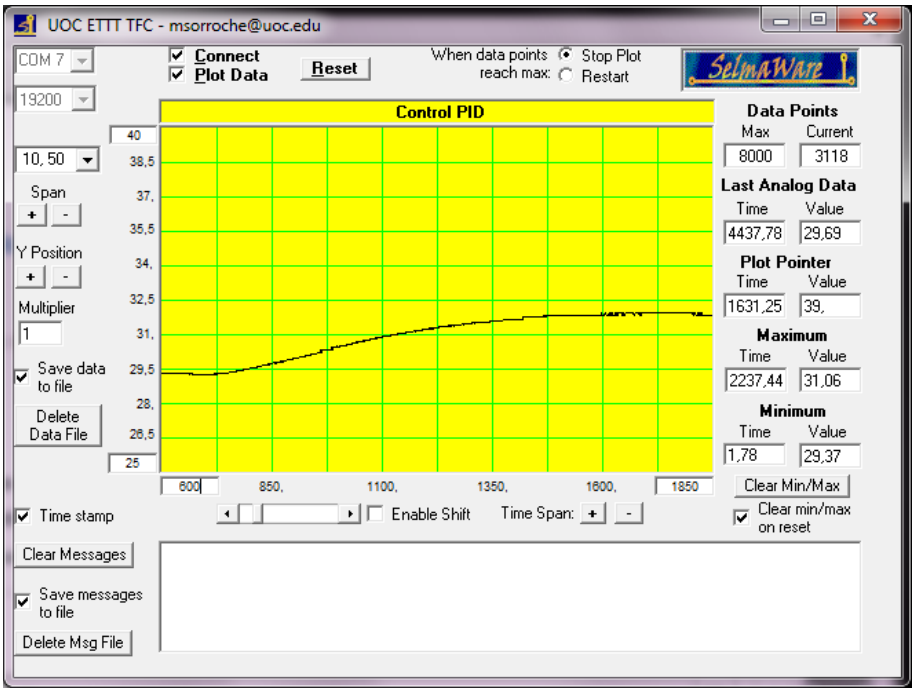


Figura 1.5: Aplicación Stamp Plot Lite V1.7, para adquisición de datos.

- Una aplicación emuladora de terminal freeware open source (Tera Term v 4.86 de Tera Term Project en OSDN) configurada para PC, que en dos instancias contiene la HMI de la aplicación y un monitor de log del sistema.

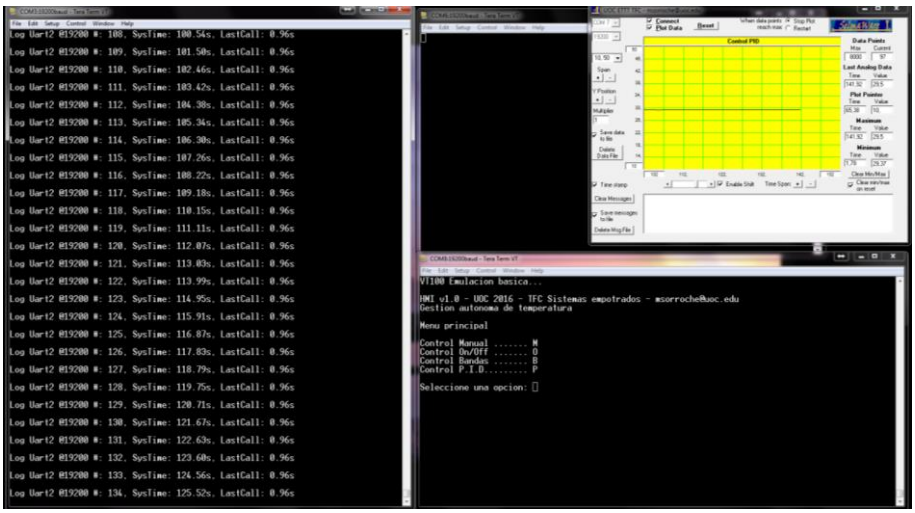


Figura 1.6: Aplicación Tera Term V4.86, para menú de usuario y log del sistema.

- Prototipo de planta con todas las partes de hardware ensambladas y testadas (sensores de temperatura, placa de relés, leds, alimentación, calefacción, ventilación, etc.).

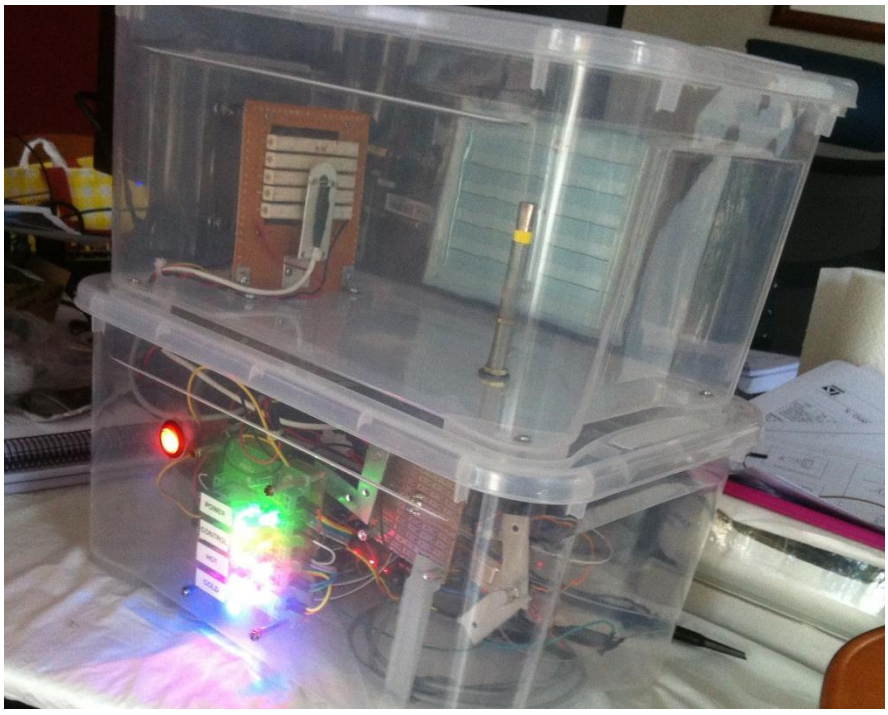


Figura 1.7: Prototipo de la planta totalmente ensamblado y testado.

- Un sistema empotrado formado por la placa LPC1769 conectada al prototipo y ejecutando la aplicación de gestión autónoma de temperatura, objetivo del proyecto.

1.8. Breve descripción de los otros capítulos de la memoria.

Presentamos a continuación una breve descripción de los capítulos restantes.

En el **capítulo 2** se realiza una aproximación a los sistemas embebidos, el estado del arte en la actualidad ya sea en sistemas empotrados, sistemas operativos (en tiempo real) o protocolos de comunicación. Finalizamos realizando un pequeño estudio de mercado actual.

En el **capítulo 3** se hace una descripción funcional del sistema desarrollado, justificando las decisiones adoptadas en su construcción. Explicamos el funcionamiento y diseño del sistema de forma global, la aplicación utilizada en el PC para adquisición de datos y el diseño realizado en el sistema embebido.

En el **capítulo 4** explicamos los diferentes aparatados del capítulo anterior de forma más teórica y detallada de los aspectos más importantes de cada uno de los elementos del sistema desarrollado.

En el **capítulo 5** no se detalla información alguna.

En el **capítulo 6** se hace un estudio de la viabilidad técnica del proyecto, exponiendo puntos fuertes y débiles.

En el **capítulo 7** se presenta una pequeña valoración económica del desarrollo del proyecto, de los costes de mantenimiento y de la industrialización del proyecto,

En el **capítulo 8** se detallan las principales conclusiones del proyecto, objetivos finalizados y pendientes, autoevaluación y líneas de trabajo futuro.

En el **capítulo 9** se presenta un glosario de palabras usadas en el contexto.

En el **capítulo 10** se publica toda la bibliografía y las referencias a las fuentes de información consultadas.

En el **capítulo 11** se adjuntan apartados autocontenidos no incluidos en la memoria por su extensión.

2. Antecedentes

Los sistemas empotrados o embebidos son sistemas de computación diseñados para realizar una o varias funciones dedicadas, frecuentemente en tiempo real. A diferencia de los sistemas de propósito general (computadores personales, servidores, etc.) los sistemas embebidos se diseñan para cubrir necesidades específicas, de tal forma que la mayoría de los componentes necesarios en una aplicación ya se encuentran disponibles en la propia placa base (entradas/ salidas analógicas, entradas/salidas digitales, temporizadores, etc.) siendo autosuficientes en gran medida.

A continuación mostramos una ilustrativa figura donde se presenta de forma simplificada un esquema de bloques genérico de un sistema embebido.

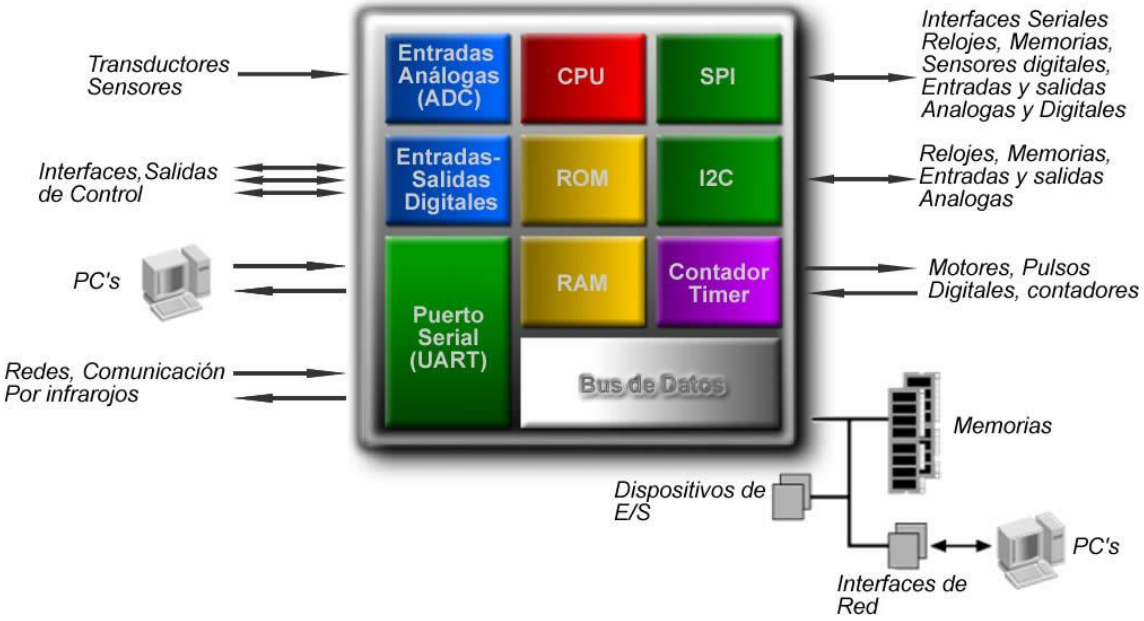


Figura 2.8: Diagrama de bloques genérico de un sistema embebido

Podemos observar en la figura, la posición destacada de la CPU en la parte central superior, en contacto con la memoria ROM y RAM, que pueden adoptar múltiples configuraciones.

En el lateral izquierdo encontramos las entradas y salidas analógicas y digitales, que nos permiten interconectar con transductores, sensores e interficies de control.

En la parte inferior encontramos los puertos serie (UART) siempre en número suficiente, que nos permite interconectar con redes, comunicación por infrarrojos y sistemas informáticos. También encontramos bus de datos que nos permiten acceder a memoria externa, dispositivos de entrada y salida, e interfaces de red para la interconexión con otros sistemas informáticos.

En el lateral derecho encontramos buses SPI e I2C, que nos permiten interconectar con un gran conjunto de interficies serie, relojes, memorias, sensores digitales, entradas y salidas analógicas y digitales. También encontramos los contadores y timers, que nos permiten controlar motores paso a paso o bien leer encoders.

2.1. Estado del arte

En la actualidad, el sector de los sistemas empotrados esta en creciente auge y difícilmente se ve alcanzar su fin ya que cada día aparecen más y más aplicaciones en todos los sectores inimaginables, desde el control de la orografía marina o el espacio aéreo hasta el internet de las cosas, en el ámbito doméstico o industrial.

No solo aparecen avances en cuanto a microprocesadores, plataformas y prestaciones, las herramientas de programación y desarrollo están a la altura a fin de permitir exprimir al máximo las prestaciones de los productos con la mayor facilidad y en el menor coste posible.

Aunque los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado en el mismo, lo más habitual es utilizar los compiladores específicos, en lenguaje C o C++ entre otros. Estos compiladores suelen estar integrados en entornos de desarrollo de alto nivel, donde se incorporan herramientas para la depuración y análisis del código, incluso en tiempo real, así como ayudas e información complementaria.

Los entornos de desarrollo facilitan y reducen drásticamente el tiempo de desarrollo lo cual ha influido notablemente en la popularización de estos sistemas, al igual que su reducido coste. Así pues podemos encontrar productos de múltiples fabricantes a un precio muy económico junto con una gran variedad de accesorios, sensores, actuadores y módulos específicos que cubren prácticamente cualquier necesidad.

2.1.1. Sistemas empotrados

Tal y como hemos descrito con anterioridad, los sistemas empotrados están compuestos por una unidad central de proceso CPU (o MCU), memoria (ROM, RAM, Flash, etc.), buses de direcciones y datos. Además incorporan en el encapsulado dispositivos conversores analógicos-digital (ADC), digitales-analógicos (DAC), moduladores de anchura de pulso (PWM), contadores y timers, temporizadores, entradas y salidas digitales de propósito general (GPIO) y diversas interficies de comunicaciones de distinto nivel tales como Ethernet, UART, I2C, SSP, CAN, etc. que permite la interconexión con el mundo exterior e interacción con multitud de dispositivos como sensores (temperatura, presión, humedad, posición, luminosidad, radiación, etc.), actuadores (motores, levas, relés, servo válvulas, etc.) u otros sistemas de computación de mayor o menor entidad.

En la siguiente tabla mostramos algunos de los fabricantes y productos ofrecidos en la industria, entre los cuales se encuentra NXP LPC1769.

Fabricante	Modelo	CPU	Bits	RAM KB	ROM KB	Freq. Mhz
Atmel	AT32UC3L064	AVR32UC	32	16	64	50
Freescale	MK11DX256VMC5	ARM Cortex M4	32	32	256	50
Microchip	PIC32MZ2048EFM144	PIC32MZ	32	512	2048	200
NXP	LPC1769	ARM Cortex M3	32	64	512	120
STM	STM32F767VG	ARM Cortex M7	32	512	1024	216
Texas Instr.	MSP432P401RIPZ	ARM Cortex M4F	32	64	256	48
Zilog	Z32F38412ALS	ARM Cortex M3	32	16	384	75

Figura 2.9: Tabla comparativa entre microcontroladores

La placa NXP LPC1769 es el producto que hemos utilizado en el presente proyecto (ARM Cortex M3). Dentro de las distintas opciones disponibles en el mercado es una de la más económicas y con una excelente relación calidad/precio. Es actualmente la placa de referencia en la UOC para sistemas embebidos.

2.1.2. Sistemas operativos

En este apartado nos centraremos en exclusiva en los denominados sistemas operativos de tiempo real como el que utilizamos en el desarrollo del proyecto. Los sistemas operativos sin esta prestación están destinados a sistemas embebidos más simples, más limitados y de menor alcance.

Un sistema operativo en tiempo real es aquel que tiene un control preciso sobre el tiempo, habitualmente a través de un planificador de tareas y control de interrupciones, y que permite que sus funciones y actividades estén acotadas en el tiempo.

Los sistemas operativos en tiempo real tienen un funcionamiento correcto en un tiempo determinado, que no significa corto sino acotado. Es un sistema determinista en lo que no se deja nada al azar o fuera del control del sistema, con entradas, salidas y restricciones temporales perfectamente conocidas.

Algunas de sus principales características son:

- Uso en sistemas embebidos
- Cambios rápidos de contexto con bajo intercambio entre el almacenamiento secundario y memoria.
- Gestión de archivos orientada a la velocidad de acceso más que la eficiencia.
- No son eficientes procesando información. Se usan procesadores predecibles.

- No existe paginación para evitar el factor aleatorio en la búsqueda.
- Tiempo de respuesta acotado.
- Objetivos muy limitados.
- Diseñados para soportar numerosas arquitecturas y montarse sobre muchos dispositivos.
- Máquinas de características baja (bajo procesamiento y poca memoria).

A continuación presentamos algunas de las características más representativas de una pequeña muestra de los sistemas operativos en tiempo real más importantes:

FreeRTOS

Es un sistema operativo desarrollado en lenguaje C y ensamblador.

Esta ampliamente extendido en el mundo académico.

Permite el control de tareas, prioridades, semáforos y colas.

Se puede integrar perfectamente con diferentes IDE como Eclipse o MPLAB para la edición y depuración de código.

Es el sistema operativo que usamos para la realización de este proyecto.

El kernel de tiempo real es totalmente gratuito.

Ofrece el código abierto por ser software libre.

Es el líder del mercado de RTOS, soportado por 31 arquitecturas de sistemas embebidos y con 77500 descarga anuales. Es el sistema operativo de tiempo real más ampliamente utilizado en el mundo.

VxWorks

Es un sistema operativo para sistemas empotrados, basado en Unix, vendido y fabricado por Wind River Systems. Como la mayoría de los sistemas operativos en tiempo real, vxWorks incluye kernel multitarea con planificador preemptive (los procesos pueden tomar la CPU arbitrariamente), respuesta rápida a las interrupciones, comunicación entre procesos, sincronización y sistema de archivos

Las características distintivas de VxWorks son:

- Compatibilidad POSIX
- Tratamiento de memoria
- Características de multiprocesador

- Shell de interfaz de usuario
- Monitor de rendimiento y depuración de código fuente y simbólico

Al contrario que en sistemas nativos como Unix, el desarrollo de vxWorks se realiza en un "host" que ejecuta Unix o Windows.

En la actualidad, vxWorks puede ejecutarse en prácticamente todas las CPU modernas del mercado de sistemas embebidos. Esto incluye la familia de CPUs x86, MIPS, PowerPC, SH-4, ARM, StrongARM y xScale

RTLinux

Es un sistema operativo de tiempo real que ejecuta Linux como un thread (hilo de ejecución) de menor prioridad que las tareas de tiempo real. Con este diseño, las tareas de tiempo real y los gestores de interrupciones nunca se ven retrasados por operaciones que no son de tiempo real.

La primera versión de RTLinux estaba diseñada para ejecutarse en la plataforma x86 y proporcionaba una pequeña API y un pequeño entorno de programación. La versión 2, que fue totalmente reescrita, fue diseñada para el soporte de multiprocesamiento simétrico (SMP) y para ser ejecutada en una amplia variedad de arquitecturas.

RTLinux proporciona la capacidad de ejecutar tareas de tiempo real y gestores de interrupciones en la misma máquina que el Linux estándar. Estas tareas y los gestores se ejecutan cuando se necesitan en detrimento de lo que estuviera ejecutando Linux.

La empresa Wind River es actualmente la propietaria de RTLinux.

Algunas de sus características son:

- Sistema operativo de tiempo real estricto.
- Extensiones para entorno multiprocesador SMP (x86).
- API similar al de POSIX threads. Planificador expulsivo por prioridades fijas, señales, sistema de archivos POSIX (open, close, etc.) semáforos y condición de variable.
- Depuración de código mediante GDB (GNU Debugger).
- Soporte para arquitecturas x86 y PPC.
- Acceso directo al hardware (puertos e interrupciones).
- Comunicación con procesos linux mediante memoria compartida y pipes
- Estructura modular para crear sistemas pequeños.
- Eficiente gestión de tiempos. En el peor caso se dispone de una resolución próxima al microsegundo (para un i486).

- Facilidades para incorporar nuevos componentes: relojes, dispositivos de E/S y planificadores.

QNX

Es un sistema operativo de tiempo real de tipo Unix que cumple con la norma POSIX. Es desarrollado principalmente para su uso en sistemas embebidos y está disponible para arquitecturas x86, MIPS, PowerPC, SH4, ARM, StrongARM, xScale, etc.

QNX está basado en una arquitectura de kernel micronúcleo que proporciona características de estabilidad avanzadas de memoria protegida frente a fallos de dispositivos, aplicaciones, etc.

El microkernel de QNX, llamado Neutrino, está implementado en 4 variantes que desarrolla y comercializa la compañía:

- QNX Neutrino RTOS. Ésta versión es la más completa y robusta pensada para cumplir los requerimientos de sistemas embebidos. Es un microkernel real de arquitectura modular.6
- QNX OS Safety. Está diseñada para cumplir con las normas ISO 26262 en ASIL D y las normas IEC 61508 en SIL3. Provee de un sistema diseñado sobre una base segura, para implementar en sistemas críticos como automóviles, trenes y automatización industrial.6
- QNX OS Medical. Cumple las normas IEC 62304 y está diseñado para reducir el esfuerzo en el desarrollo de dispositivos médicos que requieren de aprobaciones regulativas.6
- QNX OS Security. Es un RTOS de características completas, certificado en norma ISO/IEC 15408 EAL 4

ChorusOS

ChorusOS es un sistema operativo para aplicaciones empotradas y en tiempo real, desarrollado por la empresa Sun Microsystems. Actualmente está liberado bajo código abierto.

Es altamente escalable y estable, se utiliza para sistemas distribuidos, en red, empotrados o en tiempo real y se ha establecido como un sistema operativo muy utilizado embebido en hardware para comunicaciones, desde móviles hasta switches. También se encuentra en otras aplicaciones empotradas, tales como impresoras, autómatas, etc.

Está basado en arquitectura de componentes (módulos), lo que le dota de una alta capacidad de configuración y escalabilidad.

Este sistema operativo pertenece a la quinta generación de los sistemas operativos.

En el desarrollo del proyecto hemos utilizado el sistema operativo FreeRTOS, ya que se integra perfectamente con el hardware de la LPC1769, es open source y se ofrece el código fuente por

ser software libre. Es el sistema operativo en tiempo real de referencia elegido por la UOC para el desarrollo de aplicaciones empotradas.

2.1.3. Protocolos de comunicación

En este apartado expondremos de forma genérica alguno de los más importantes protocolos de comunicación presentes en la mayoría de los dispositivos citados anteriormente.

SPI

Serial Peripheral Interface es un estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos y es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj y es la opción más flexible cuando se tiene diferentes tipos de periféricos serie minimizando el número de conectores, pines y tamaño del circuito.

El SPI presenta las siguientes ventajas:

- Comunicación Full Duplex.
- Mayor velocidad de transmisión que con I²C o SMBus.
- Protocolo flexible en que se puede tener un control absoluto sobre los bits transmitidos.
- No está limitado a la transferencia de bloques de 8 bits.
- Elección del tamaño de la trama de bits, de su significado y propósito.
- Su implementación en hardware es extremadamente simple.
- Consume menos energía que I²C o que SMBus debido que posee menos circuitos (incluyendo las resistencias pull-up) y estos son más simples.
- No es necesario arbitraje o mecanismo de respuesta ante fallos.
- Los dispositivos clientes usan el reloj que envía el servidor, no necesitan por tanto su propio reloj.
- No es obligatorio implementar un transceptor (emisor y receptor), un dispositivo conectado puede configurarse para que solo envíe, sólo reciba o ambas cosas a la vez.
- Usa mucho menos terminales en cada chip/conector que una interfaz paralelo equivalente.

- Como mucho una única señal específica para cada cliente (señal SS), las demás señales pueden ser compartidas.

Y las siguientes desventajas:

- Consume más pines de cada chip que I²C, incluso en la variante de 3 hilos.
- El direccionamiento se hace mediante líneas específicas (señalización fuera de banda) a diferencia de lo que ocurre en I²C que se selecciona cada chip mediante una dirección de 7 bits que se envía por las mismas líneas del bus.
- No hay control de flujo por hardware.
- No hay señal de asentimiento. El servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada.
- No permite fácilmente tener varios servidores conectados al bus.
- Sólo funciona en las distancias cortas a diferencia de, por ejemplo, RS-232, RS-485, o Bus.

CAN

Controller Area Network es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología bus para la transmisión de mensajes en entornos distribuidos. El protocolo ofrece una solución a la gestión de la comunicación entre múltiples CPUs (unidades centrales de proceso), alta inmunidad a las interferencias, habilidad para el autodiagnóstico y la reparación de errores de datos.

Es un protocolo de comunicaciones normalizado, con lo que se simplifica y economiza la tarea de comunicar subsistemas de diferentes fabricantes sobre una red común o bus. El procesador anfitrión (host) delega la carga de comunicaciones a un periférico inteligente, por lo tanto el procesador anfitrión dispone de mayor tiempo para ejecutar sus propias tareas.

Es una red multiplexada, reduce considerablemente el cableado y elimina las conexiones punto a punto, excepto en los enganches.

CAN se basa en el modelo productor/consumidor y está orientado a mensajes, es decir la información que se va a intercambiar se descompone en mensajes, a los cuales se les asigna un identificador y se encapsulan en tramas para su transmisión. Cada mensaje tiene un identificador único dentro de la red, con el cual los nodos deciden aceptar o no dicho mensaje. Dentro de sus principales características se encuentran:

- Prioridad de mensajes.
- Garantía de tiempos de latencia.
- Flexibilidad en la configuración.

- Recepción por multidifusión (multicast) con sincronización de tiempos.
- Sistema robusto en cuanto a consistencia de datos.
- Sistema multimaestro.
- Detección y señalización de errores.
- Retransmisión automática de tramas erróneas
- Distinción entre errores temporales y fallas permanentes de los nodos de la red, y desconexión autónoma de nodos defectuosos.

I^2C

Inter-Integrated Circuit es un bus de datos serie desarrollado en 1982 por Philips Semiconductors (hoy NXP Semiconductors). Se utiliza principalmente para la comunicación interna entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.

El I²C está diseñado como un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro; el esclavo reacciona. Es posible tener varios maestros (Multimaster-Mode). En el modo multimaestro se pueden comunicar dos maestros entre ellos, de modo que uno de ellos trabaja como esclavo. El arbitraje (control de acceso en el bus) se rige por las especificaciones, de este modo los maestros pueden ir turnándose.

Una de las propiedades del I²C es el hecho de que un microcontrolador puede controlar toda una red de circuitos integrados con sólo dos I/O-Pins (Input/Output) y un software muy simple.

Aunque es más lento que los sistemas de bus más nuevos, I²C es beneficioso (debido al bajo coste) para los sistemas periféricos que no necesitan ser rápidos.

Es posible añadir o retirar microcontroladores al bus durante su funcionamiento (Hot-Plugin).

USB

Universal Serial Bus es un bus estándar industrial que define y unifica los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos. Ha sufrido a lo largo de los años una evolución mejorando prestaciones y características, estando en la actualidad en la versión 3.1 del estándar.

Las señales del USB se transmiten en un cable de par trenzado y utilizan señalización diferencial en half dúplex excepto el USB 3.0 que utiliza un segundo par de hilos para realizar una comunicación en full dúplex.

Se puede crear redes de hasta 127 elementos, con la facilidad de conexión "Hot Plug - en caliente", es decir, que se pueden conectar y desconectar los periféricos sin necesidad de reiniciar la red.

Los dispositivos USB se pueden clasificar en cuanto a su velocidad en cuatro tipos:

- Baja velocidad (1.0). Tasa de transferencia de hasta 1,5 Mbps (192 KB/s).
- Velocidad completa (1.1). Tasa de transferencia de hasta 12 Mbps (1,5 MB/s).
- Alta velocidad (2.0). Tasa de transferencia de hasta 480 Mbps (60 MB/s).
- Súper alta velocidad (3.0). Tasa de transferencia de hasta 4.8 Gbps (600 MB/s).

El estándar USB (USB 1.1/2.0) utiliza una arquitectura maestro/esclavo: un concentrador (HUB) USB actúa como USB maestro, mientras un dispositivo USB actúa como esclavo. Sólo los concentradores USB pueden gestionar la configuración y la transferencia de datos durante la conexión.

A partir de la norma USB-OTG cambia esta situación. Los dispositivos compatibles con USB-OTG son capaces de abrir una sesión, controlar la conexión e intercambiar las funciones maestro/dispositivo, introduciendo para ello dos nuevos protocolos:

- Protocolo SRP (Session Request Protocol: protocolo de solicitud de sesión).
- Protocolo HNP (Host Negotiation Protocol: protocolo de negociación de host).

Los dispositivos USB OTG son compatibles con USB 1.1/2.0 y se comportan como un dispositivo USB estándar cuando se conectan con dispositivos USB tradicionales (no OTG).

I^2S

Inter-IC Sound, Integrated Interchip Sound, o IIS, es un estándar eléctrico de bus serie usado para interconectar circuitos de audio digital. El bus I^2S separa las señales de datos y de reloj, lo que resulta una mejora en la cantidad de fluctuación (jitter) de la señal.

Este estándar fue publicado por primera vez en 1986 y su última revisión es de 1996.

El bus consiste de al menos tres líneas:

- La línea de reloj de bit.
- La línea de reloj de palabra (llamada también selección de palabra, WS, o reloj izquierda-derecha (LRCLK).
- Por lo menos una línea de datos multiplexados.

Opcionalmente, puede también incluir las siguientes líneas:

- Reloj maestro (típicamente 256 x LRCLK)
- Una línea multiplexada de entrada para obtener comunicación bidireccional.

JTAG

Joint Test Action Group, es el nombre común utilizado para la norma IEEE 1149.1-1990 denominada Standard Test Access Port and Boundary-Scan Architecture para test access ports utilizada para testear PCBs utilizando escaneo de límites. En 1994 se agregó un suplemento que contiene una descripción del boundary scan description language (BSDL). Desde entonces, esta norma fue adoptada por las compañías electrónicas de todo el mundo. Actualmente, Boundary-Scan y JTAG son sinónimos.

Diseñado originalmente para circuitos impresos, actualmente es utilizado para la prueba de submódulos de circuitos integrados, y es muy útil también como mecanismo para depuración de aplicaciones empotradas, puesto que provee una puerta trasera en el interior del sistema. Cuando se utiliza como herramienta de depuración, un emulador en circuito que usa JTAG como mecanismo de transporte permite al programador acceder al módulo de depuración que se encuentra integrado dentro de la CPU. El módulo de depuración permite al programador corregir sus errores de código o lógica de sus sistemas.

Una interfaz JTAG es una interfaz especial de cuatro o cinco pines agregadas a un chip, diseñada de tal manera que varios chips en una tarjeta puedan tener sus líneas JTAG conectadas en daisy chain, de manera tal que una sonda de testeo JTAG necesita conectarse a un solo "puerto JTAG" para acceder a todos los chips en un circuito impreso. Los pines del conector son:

- TDI (Entrada de Datos de Testeo)
- TDO (Salida de Datos de Testeo)
- TCK (Reloj de Testeo)
- TMS (Selector de Modo de Testeo)
- TRST (Reset de Testeo) es opcional.

Casi cualquier sistema embebido tiene un puerto JTAG, e incluso el conector de bus PCI tiene pines JTAG.

La interfaz JTAG es accesible por medio de aplicaciones de JTAG.

RS232

Recommended Standard 232, también conocido como EIA/TIA RS-232C, es una interfaz que designa una norma para el intercambio de una serie de datos binarios entre un DTE (Data Terminal Equipment, Equipo Terminal de Datos) y un DCE (Data Communication Equipment, Equipo de Comunicación de Datos), aunque existen otras en las que también se utiliza la interfaz RS-232. Una definición equivalente publicada por la UIT se denomina V.24.

Por medio de este protocolo se estandarizan las velocidades de transferencia de datos, la forma de control que utiliza dicha transferencia, los niveles de voltajes utilizados, el tipo de cable permitido, las distancias entre equipos, los conectores, etc.

Además de las líneas de transmisión (Tx) y recepción (Rx), las comunicaciones seriales poseen otras líneas de control de flujo, donde su uso es opcional dependiendo del dispositivo a conectar, como las siguientes:

- Request To Send (RTS) Esta señal se envía del dispositivo DTE al dispositivo DCE para indicar que se quieren transmitir datos. Si el DCE decide que esta OK, asiente por la línea CTS.
- Clear To Send (CTS) Afirmado por el DCE después de recibir la señal de RTS indica que la DTE puede transmitir.
- Data Terminal Ready (DTR) Esta línea de señal es afirmada por el DTE, e informa al DCE que el DTE está listo para recibir datos.
- Data Set Ready (DSR) Esta línea de señal es afirmada por el DCE en respuesta a una señal de DTR del DTE. El DTE supervisa el estado de esta línea después de afirmar DTR para descubrir si el DCE está encendido.
- Receive Signal Line Detect (RSLD) Esta línea de control es afirmada por el DCE e informa al DTE que se ha establecido una conexión física con otro DCE. A veces se conoce como detector de portadora (CD). Sería un error que una computadora transmita información a un DCE si esta línea no está activa, es decir si la conexión física no funciona.
- Transmit Data (TD) es la línea por donde el dato se transmite de un bit a la vez.
- Receive Data (RD) es la línea por donde el dato se recibe de un bit a la vez.

RS485

Recommended Standard 485 o también conocido como EIA-485 es desde 1983 un estándar de comunicaciones en bus de la capa física del Modelo OSI.

Está definido como un sistema de bus diferencial multipunto, es ideal para transmitir a altas velocidades sobre largas distancias (35 Mbit/s hasta 10 metros y 100 kbit/s en 1200 metros) y a través de canales ruidosos, ya que el par trenzado reduce los ruidos que se inducen en la línea de transmisión. El medio físico de transmisión es un par trenzado que admite 32, 128 o 254 estaciones en 1 solo par, con una longitud máxima de 1200 metros operando entre 300 y 19 200 bit/s y la comunicación half-duplex (semiduplex) dependiendo del consumo de cada driver. La transmisión diferencial permite alcanzar mayor distancia con una notable inmunidad al ruido, siempre que el bus de comunicación conserve las características de bus balanceado dando la posibilidad de una configuración multipunto.

Entre sus principales características tenemos:

- Interfaz diferencial
- Conexión multipunto
- Alimentación única de +5V
- Hasta 32 estaciones (ya existen interfaces que permiten conectar 256 estaciones)
- Velocidad máxima de 10 Mbit/s (a 12 metros)
- Longitud máxima de alcance de 1200 metros (a 100 kbit/s)
- Rango de bus de -7V a +12V

Onewire (1-Wire)

Es un protocolo de comunicaciones en serie diseñado por Dallas Semiconductor. Está basado en un bus, un maestro y varios esclavos de una sola línea de datos en la que se alimentan (con una resistencia pull-up) y una referencia a tierra común a todos los dispositivos.

1-Wire es el único sistema digital basado en voltaje que funciona con solo dos contactos, datos y común, para la comunicación bidireccional half-duplex. En contraste con otros sistemas de comunicación en serie, tales como I²C o SPI, los dispositivos 1-Wire están diseñados para su uso en un entorno de contacto. Desconectar del bus 1-Wire o una pérdida de contacto coloca a los esclavos 1-Wire en un estado de reposición definido. Cuando la tensión vuelve, los esclavos se despiertan y señalar su presencia. Con dos contactos, los dispositivos 1-Wire son la forma más económica para agregar funcionalidad a los equipos electrónicos.

ETHERNET

Es un estándar de redes de área local para computadores con acceso al medio por detección de la onda portadora y con detección de colisiones (CSMA/CD). Su nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI.

Ethernet se tomó como base para la redacción del estándar internacional IEEE 802.3, siendo usualmente tomados como sinónimos. Se diferencian en uno de los campos de la trama de datos, sin embargo las tramas Ethernet e IEEE 802.3 pueden coexistir en la misma red.

2.2. Estudio de mercado

En el mercado existen multitud de sistemas o dispositivos para el control de temperatura. De hecho son sistemas maduros tecnológicamente que han alcanzado un alto grado de especialización, fabricándose en muchos formatos y a precios muy asequibles.

Fruto de esa especialización y madurez han derivado esos equipos en sistemas muy cerrados y limitados a su función principal (única función) de control de temperatura. Se fabrican con costes muy optimizados y eso menoscaba su versatilidad, porque deja pocas opciones de personalización o funciones adicionales fuera de la estándar.

Por otro lado, los diseños y modelo actuales repiten patrones de modelos que hacen muchos años están en el mercado y dan prueba de que poca o nula innovación ha llegado recientemente hasta ellos, excepto en los modelos más sofisticados que incorporan avances en los sistemas de comunicación e integración en redes.

La incorporación de sistemas embebidos para la realización de este tipo de proyectos puede dar un impulso a estos sistemas, ya que a un precio muy reducido se incorpora la potencia de un sistema informático completo, con todas las ventajas que ello representa.

Los sistemas empotrados permiten fácilmente la personalización de las variables a controlar (temperatura, presión, etc.) que incluso puede que no estén físicamente cerca del equipo y se controlen remotamente, incorporación de nuevos lazos de control o algoritmos, posibilidad de adquisición de datos y análisis simultaneo, etc.

El sistema desarrollado, gracias a su versatilidad, va dirigido a múltiples sectores como por ejemplo:

- Sector industrial. Aplicaciones para el control de variables físicas como temperatura, presión, humedad, iluminación, fuerza, etc. en plantas industriales.
- Sector doméstico. Aplicaciones en el hogar, como control de temperatura vivienda, precalentado de agua en lavavajilla, lavadoras, etc. Control de iluminación variable en estancias, control de persianas, etc. Aplicaciones domóticas.
- Sector educativo: Plataformas educativas donde experimentar y desarrollar los conceptos estudiados.
- Sector agrario: Control y dosificación de pienso, etc.

3. Descripción funcional

En el presente capitulo expondremos el diseño y funcionalidad del sistema desarrollado, así como las decisiones que se han tomado y su fundamento.
Aun así se pretende crear tan solo una pincelada de los diversos elementos que componen el sistema y de la relación e interacción entre los mismos, para profundizar con más detalle en apartados posteriores.

3.1. Sistema autónomo de temperatura

El sistema se compone de dos partes principales:

- **Sistema de control**. Corresponde al sistema embebido y el programa de aplicación que permite realizar el control autónomo de temperatura.
- **Planta**: Corresponde con la planta a controlar. Incorpora los elementos actuadores (calefactor y ventilador, así como las sondas de temperatura y toda la electrónica necesaria para realizar la maniobra de estos elementos a través de líneas de control.

3.1.1. Diagrama de bloques de la aplicación

El diagrama de bloques del sistema es el mostrado a continuación. El sistema de control es una caja negra donde se introducen (INPUTS) la consigna de temperatura y el valor de las sondas de temperatura de la planta. En función de los algoritmos de control, genera dos salidas de acción (OUTPUT) correspondientes a los actuadores calefactor y ventilador, que desplazan la temperatura real de la planta (Tp) hacia la consigna establecida (Tc).

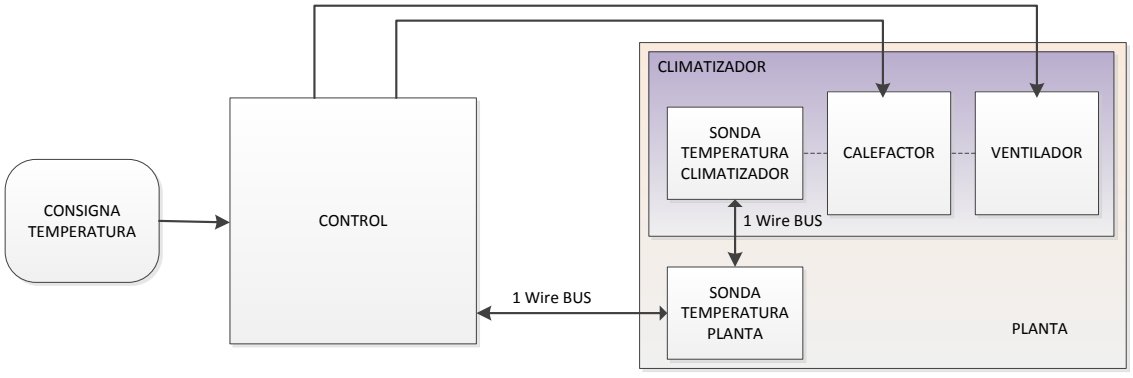


Figura 3.10: Diagrama de bloques del sistema.

El sistema se haya en equilibrio cuando la temperatura de la sonda coincide (dentro de la tolerancia aceptada) con la consigna programada.

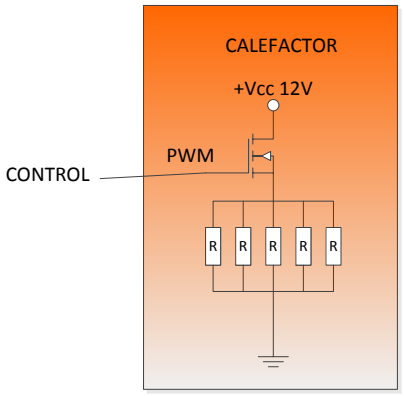


Figura 3.11: Esquema del bloque calefactor.

Dentro del bloque calefactor el esquema es simple. Disponemos de una alimentación en continua sobre un grupo de resistencias cerámicas conectadas en paralelo y reguladas a través de un transistor mosfet por una señal PWM. Variando la señal PWM conseguimos mayor o menor paso de corriente y por tanto potencia calorífica.

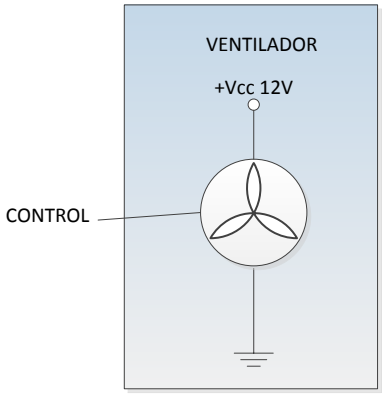


Figura 3.12: Esquema del bloque ventilador.

El diagrama del bloque ventilador consta de un ventilador alimentado en continua y controlado a través de una señal analógica. Variando la señal analógica conseguimos mayor o menor velocidad de giro y por tanto impulsión de aire.

3.1.2. Como es la red (posición y comunicación a través de la red)

El sistema no hace uso en estos momentos de ninguna conexión a red de datos. Esa funcionalidad esta prevista dentro de las líneas de trabajo futuras y está pendiente de ser desarrollada.

Por otro lado la aplicación se comunica con una red de sensores de los cuales obtiene el valor de temperatura del sistema.

La aplicación también se comunica con varios dispositivos (opcionalmente el mismo) a través de 3 conexiones uart.

3.1.3. Cómo interactúan los diferentes objetos del sistema

Para realizar las funcionalidades descritas anteriormente se ha implementado una librería que contiene los módulos que las resuelven. En el siguiente diagrama podemos ver el grado de interrelación y cohesión entre ellas.

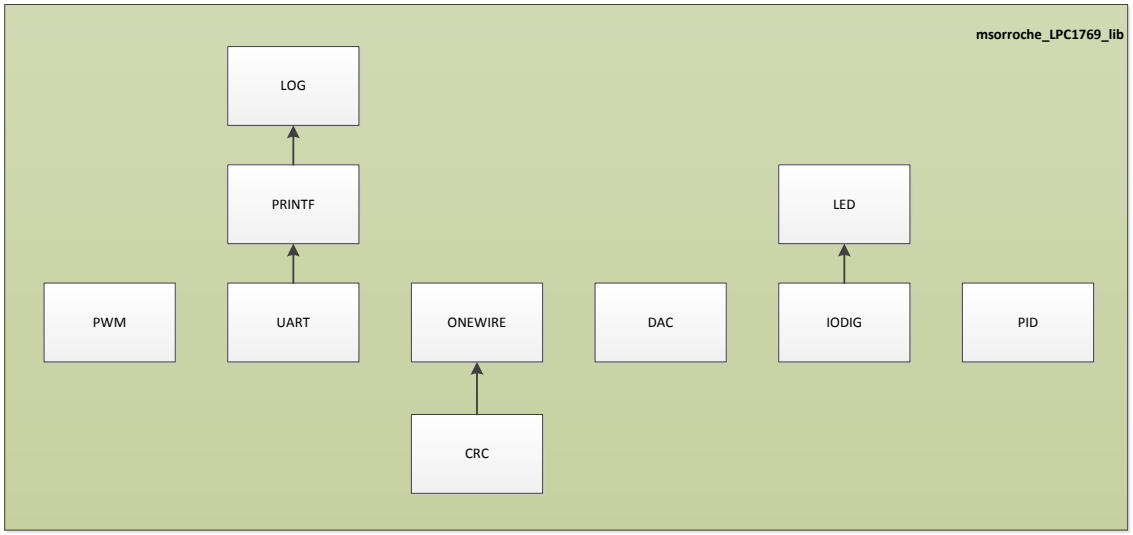


Figura 3.13: Diagrama de módulos del sistema.

Así pues vemos que el modulo Log se basa en el módulo Printf y este a su vez en el módulo Uart, o que el módulo Onewires se apoya en el módulo Crc8.

3.2. Aplicaciones PC

No se ha desarrollado aplicaciones PC propiamente, si no que se ha utilizado algunas aplicaciones freeware, configurándolas adecuadamente para cubrir las necesidades especificas de la aplicación embebida msorroche_PFC.

Efectivamente, la configuración de la aplicación Stamp Plot Lite v1.7 (ver figura 1,3) nos permite su uso como registrador de datos, ya que visualiza y grafía los datos de temperatura enviados por la aplicación embebida.

Por otro lado el uso de la aplicación Tera Term v4.86 (ver figura 1.4) nos permite la creación de una HMI para el control de la aplicación embebida desde un PC

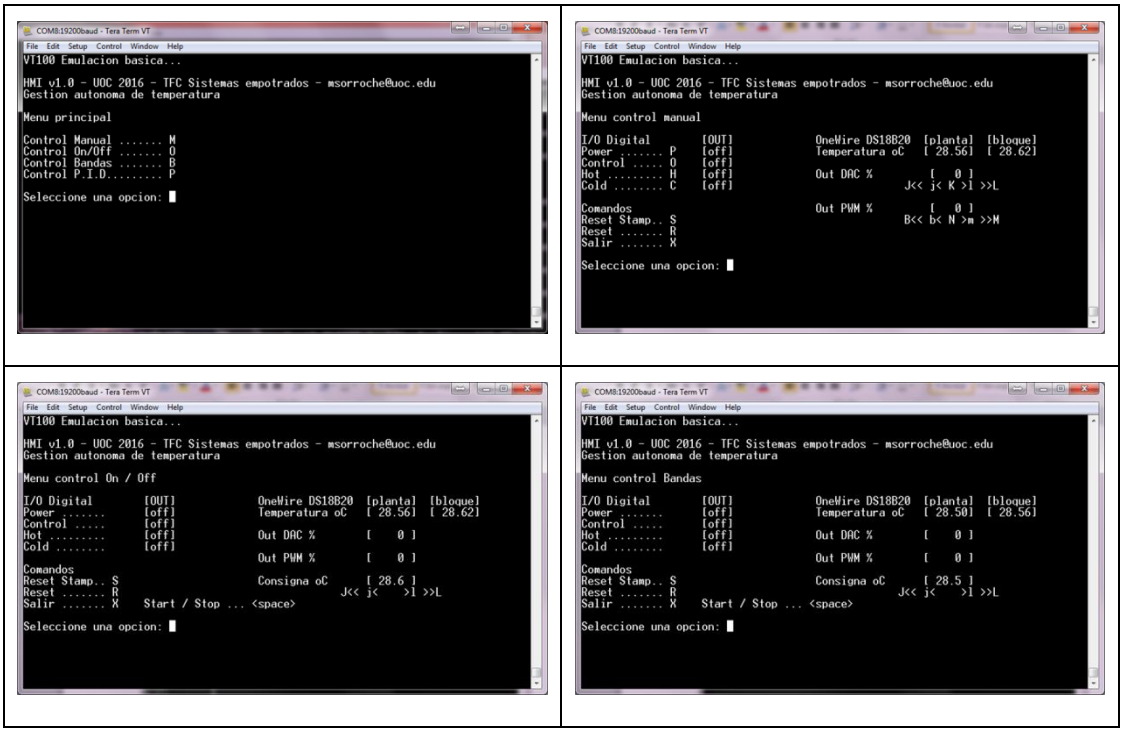


Figura 3.14: Interficie de usuario en PC.

En cualquier caso el PC cumple funciones de terminal o de registrador, no forma parte activa del sistema de control autónomo de temperatura.

3.3. Aplicación embebida msorroche_PFC

En el siguiente diagrama vemos las diferentes tareas y bucles principales de la aplicación.

La aplicación principal main inicializa los subsistemas y lanza 3 tareas concretas, quedándose en un bucle infinito.

Una de las tareas vTskAdquireSensor tiene como única misión la adquisición de los datos provenientes de los sensores de temperatura, darles formato y enviarlos al programa de traficación en el PC. Estos datos también quedan disponibles para las distintas funciones de control.

La tarea vTskLog tiene como misión el mostrar en una interficie HMI los mensajes de log que envía el sistema.

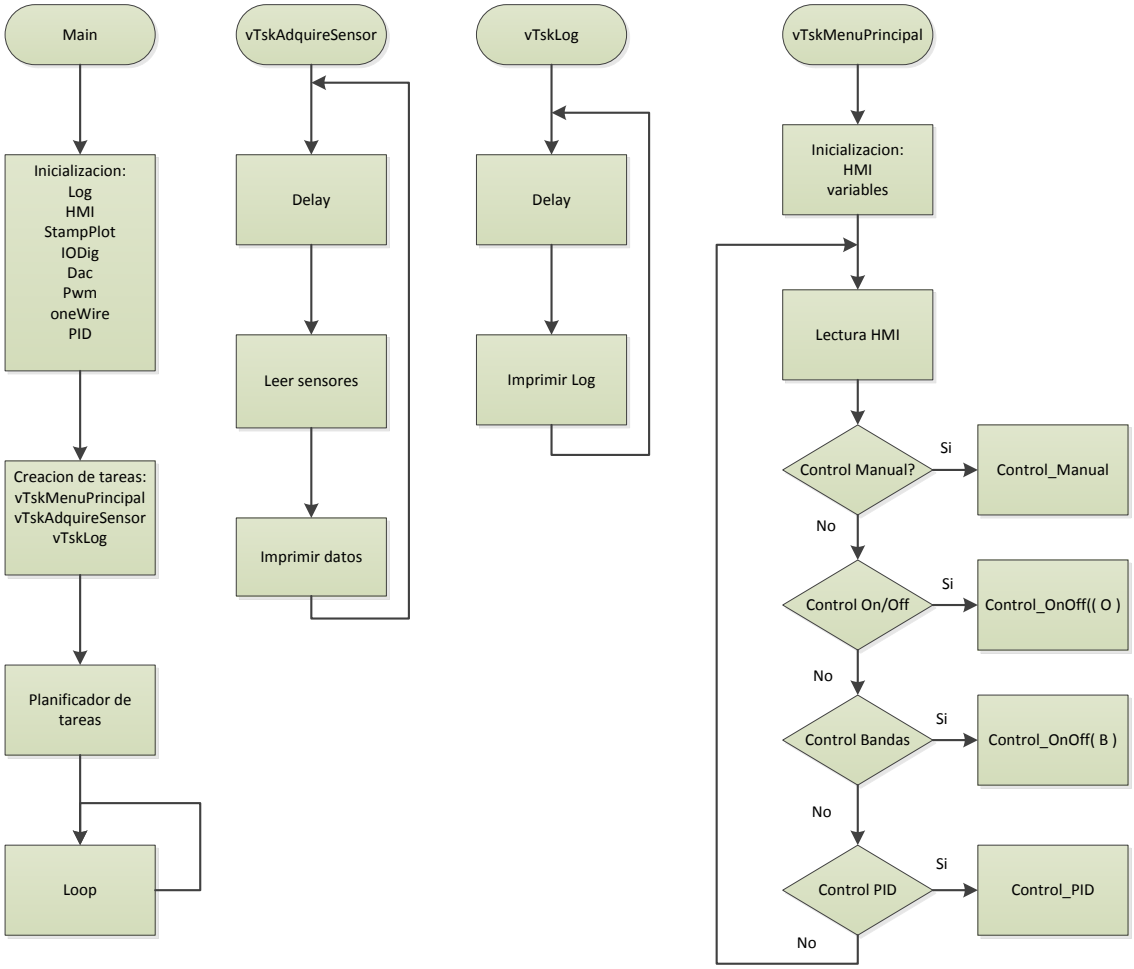


Figura 3.15: Diagrama de bloques de la aplicación embebida msorroche_PFC.

La tarea vTskMenuPrincipal inicializa la interficie HMI y queda a la espera de obtener algún comando que procesar. En función de estos comandos lanza diferente métodos para resolver los diferentes tipos de control posibles.

4. Descripción detallada

A continuación detallamos la descripción técnica de los componentes del sistema.

4.1. Lazos de control

El sistema proporciona varios tipos de algoritmos de control:

El primero es control manual, donde el sistema proporciona control manual de todas las funciones a través de la interficie HMI. No es propiamente un sistema de control puesto que carece de lógica programada y simplemente transfiere al sistema las instrucciones del operador.

El segundo es un control on/off, donde el sistema monitoriza la temperatura de planta Tp(t) y la compara con la temperatura de consigna TC(t). Si esta última es mayor, aplica calefacción al sistema. Si es menor aplica ventilación. El punto de equilibrio es inestable y el sistema regula permanentemente.

El tercer algoritmo es de control de bandas. El sistema aplica diferentes grados de control por bandas de proximidad a la consigna.

El último algoritmo es el principal de control y se basa en un sistema PID. El sistema monitoriza la temperatura de planta Tp(t) y la compara con la temperatura de consigna Tc(t).

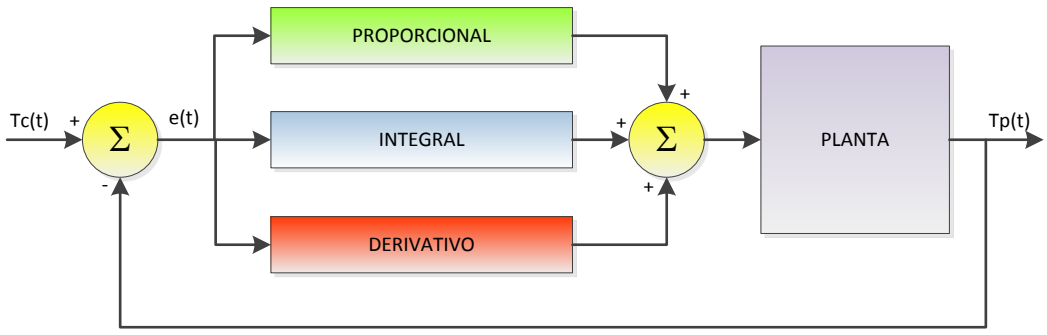


Figura 4.16: Diagrama de bloques del sistema PID.

En función de la señal de error e(t) (diferencia entre Tc(t)-Tp(t)) aplica de forma independiente un algoritmo de control proporcional, integral y derivativo (o cualquier combinación de estos), sumando sus efectos y aplicándolos a la planta, la cual reacciona desplazando la temperatura a Tp. El sistema monitoriza la temperatura de planta y vuelve a realizar un lazo de control de forma continua.

4.2. Conexionado PCB – LPC1769

A continuación mostramos el conexionado entre la placa PCB y la LPC1769

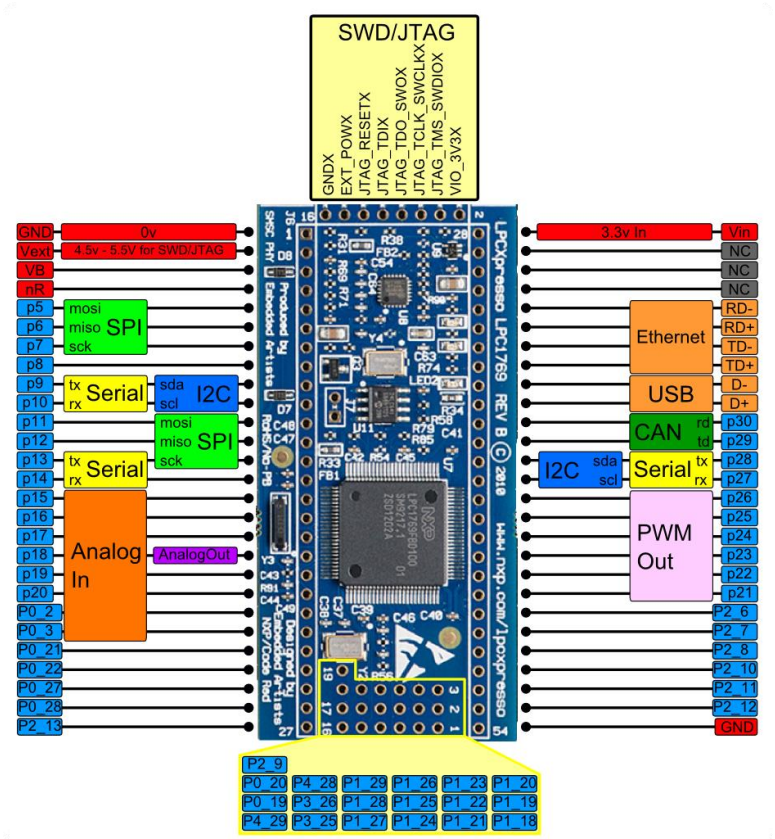


Figura 4.17: Diagrama de conexión LPC1769.

UART0	TX	P0.2	J6	21
UART0	RX	P0.3	J6	22
UART1	TX	P0.15	J6	13
UART1	RX	P0.16	J6	14
UART2	TX	P0.10	J6	40
UART2	RX	P0.11	J6	41
UART3	TX	P0.0	J6	9
UART3	RX	P0.1	J6	10
DAC		P0.26	J6	18
1WIRE		P0.21	J6	23
PWM1		P2.0	J6	42
POWER		P0.22	J6	24
CONTROL		P2.11	J6	52
HOT		P2.12	J6	53
COLD		P2.13	J6	27

Figura 4.18: Diagrama de conexión PCB – LPC1769.

5. Otros apartados que creéis convenientes

No se valorado la posibilidad de añadir ningún capitulo adicional a la obra, aun así mantenemos el apartado por coherencia con el índice.

6. Viabilidad técnica

El presente proyecto es viable técnicamente y económicamente, ya que se basa en desarrollo de aplicaciones y estructuras de control software sobre hardware existente, de fácil acceso, bajo coste y ampliamente documentado, basados en estándares de la industria y de código abierto. Los montajes y circuitos adicionales necesarios son de fácil realización u obtención.

Así, como puntos fuertes a potenciar tenemos:

- La alimentación de los distintos bloques de la planta se ha previsto a través de una fuente de alimentación conmutada PC-AT, de fácil adquisición, suficiente potencia y bajo coste. Modificamos los conectores para tener un fácil acceso a los diferentes voltajes de funcionamiento que podamos necesitar (12V y 5V). De ser necesario podríamos modificar la fuente a través de un potenciómetro o divisor de tensión para obtener voltajes intermedios requeridos.
- Para el sistema calefactor se ha dispuesto de 5 resistencias cerámicas dh BC15/15 que alimentadas a 12V consumen una corriente de 0,8A y disipan una potencia de 9,6W cada una aproximadamente. Esto nos da una elevada temperatura en superficie suficiente para la aplicación. Son elementos fáciles de encontrar y de sustituir por otros equivalentes.
- La etapa de control de potencia del calefactor se ha previsto con un transistor mosfet irfp450, capaz de soportar hasta 500V y 14A, muy lejos de los valores del proyecto, por lo que la temperatura de trabajo del transistor no requerirá un excesivo disipador de temperatura.
- Para el sistema ventilador se ha considerado utilizar uno de la casa StartTech, de 80x80, que proporciona suficiente caudal para disipar la potencia producida por el sistema calefactor. No requiere etapa de control adicional ya que la salida DAC de la LPC1769 debería ser suficiente para controlar la velocidad de giro.
- La conexión y desconexión del ventilador y calefactor a la alimentación se ha previsto a través de dos relés (HOT y COLD), que junto a los relés POWER y CONTROL son controlados por 4 salidas digitales de la LPC1769. La placa de relés que controlan estas salidas es una CEBEK T1, donde hemos sustituido los led de estado por otros de color determinado (VERDE, AMARILLO, ROJO Y AZUL) y que reutilizamos como indicadores del estado del sistema.
- Se han previsto 2 sondas digitales de temperatura modelo DS18B20 en conexión 1-wire con la LPC1769 para monitorizar la temperatura de planta y de calefactor. Se ha considerado la alimentación externa a 5V de estas sondas proveniente de la fuente de alimentación.
- Se ha diseñado y construido una protoboard que reúne, unifica y distribuye las diferentes conexiones a circuitos y buses, así como alimentaciones, facilitando su implementación. Es

fácil reconfigurar las distintas conexiones entre la protoboard y la LPC1769 y ampliarlas de ser necesario, como añadir nuevas sondas de temperatura o entradas salidas digitales

Así, como puntos débiles a mejorar tenemos:

- Las tensiones ofrecidas por la LPC1769 tanto para la señal PWM o la salida DAC están como máximo, cercanas a 3,20V. En algunos casos, depende del tipo de mosfet a excitar, resulta insuficiente para hacer que la puerta conduzca, aun estando dentro totalmente de las especificaciones del integrado. Esto es debido a que las tolerancias de los integrados suelen ser altas y dependiendo del lote de fabricación puede oscilar mucho. Se puede buscar otros tipos de transistores cuyas características estén más acorde con las características de la placa base. Otra solución es diseñar un pequeño circuito electrónico con un par Darlington, con lo que se resolvería el problema.
- Mejorar los algoritmos de control permitiendo configurar algunos parámetros internos.
- Realizar una función de auto sintonizado para el control PID, ya que si los parámetros no son adecuados la regulación es muy deficiente.

Así pues, a la vista de lo expuesto podemos determinar que el proyecto es viable técnica y económicamente, y se puede implementar y usar.

7. Valoración económica

A continuación presentamos el presupuesto de desarrollo en materiales y mano de obra. En el hemos intentado reflejar todos y cada uno de los componentes utilizados, tanto los que son directamente de adquisición en el mercado como los que han sido elaborados, adaptados o modificados para el proyecto.

Pos	Descripción	Cantidad	Unidad	Total €
01	NPX LPCXpresso LPC1769	1	20,00	20.00
02	UART-USB CP2102	4	5,50	22.00
03	Sensor temperatura DS18B20	2	6,30	12,60
04	Set cables M/H	40	0,15	6,00
05	Set cables H/H	40	0,15	6,00
06	Placa CEBEK T1	1	25,20	25,20
07	Resistencias dh 15R	5	2,40	12,00
08	Placa montaje resistencias	1	1,80	1,85
09	Cables, tornillería placa montaje resistencias	1	2,90	2,90
10	Ventilador StartTech 80x80	1	10,00	10,00
11	Rejilla protectora ventilador	1	1,75	1,75
12	Soporte separador ventilador	1	1,25	1,25
13	Soporte sonda temperatura calefactor	1	2,00	2,00
14	Soporte sonda temperatura planta	2	1,30	2,60
15	Cajas de plástico T BOX XS 38x26,5x19	2	8,20	16,40
16	Interruptor bipolar	1	5,75	5,75
17	Fuente alimentación PC-AT Media Magic 400W	1	27,00	27,00
18	Rejilla ventilación habitáculo equipo	1	2,20	2,20
19	Rejilla ventilación habitáculo planta	1	3,10	3,10
20	Hub USB 4 puertos	1	7,35	7,35
21	Cable extensor 1 m USB M/H	1	3,25	3,55
22	Cable USB – micro USB M/H	1	3,50	3,50
23	Protoboard pruebas, montaje preliminar	1	30,00	30,00
24	Protoboard prototipo, diseño y construcción	1	250,00	250,00
25	Protoboard montaje y soldadura componentes	1	60,00	60,00
26	Cables, conectores, resistencias y material	1	20,00	20,00
27	Documentación, desarrollo programación	250	10,00	2500,00
28	Montaje y puesta en marcha	40	10,00	400,00
Precio Total €				***3455,00***

Figura 7.19: Presupuesto de desarrollo

El presupuesto de mantenimiento es difícil de cuantificar a priori, ya que dependerá del alcance de la actuación y de los materiales empleados. Así pues se ha valorado el coste de un informe de reparación, de un informe de viabilidad técnica de desarrollo, el coste hora de servicio de asistencia técnica reparación y el coste de servicio asistencia técnica desarrollo. El número de horas vendrá determinado por los informes de reparación y viabilidad de desarrollo. El coste de los materiales se estima el coste de desarrollo más un 10%.

Pos	Descripción	Cantidad	Unidad	Total
01	Informe técnico reparación y presupuesto	1	25,00	25,00
02	Informe de viabilidad adaptación y presupuesto	4	25,00	100,00
03	Precio hora SAT reparación	variable	20,00	20,00
04	Precio hora SAT desarrollo	variable	25,00	25,00
05	Materiales empleados (precio desarrollo * 1,10)	variable	variable	variable

Figura 7.20: Presupuesto de mantenimiento

A continuación presentamos el presupuesto de industrialización.

Una vez realizado el estudio, desarrollo y construcción del prototipo planteamos llevar el producto a fabricación industrial. En este caso tenemos en cuenta diferentes aspectos, entre los que destacamos:

- El coste por unidad se reduce por economía de escala. La gestión eficaz de las compras de componentes en cantidades industriales reduce considerablemente su coste unitario. Aunque este principio no puede aplicarse a la totalidad de los componentes de forma lineal hemos estimado en conjunto que la fabricación de cantidades superiores a 1000 unidades reduciría el coste de los componentes a un 15% de su valor de desarrollo.
- El coste de montaje y puesta en marcha se reduce considerablemente al aplicar procedimientos estandarizados de la industria, cadenas de montajes y pruebas serializadas. Aun así, el margen de reducción no es tan elevado como en la gestión de compras y consideramos para 1000 unidades una reducción al 35% de su valor de desarrollo.
- El coste de desarrollo se divide entre las unidades fabricadas, repercutiendo una cantidad proporcional a cada una de ellas. Así pues, para 1000 unidades el coste de desarrollo será el 0,1% del valor de desarrollo por unidad.
- Aparecen costes adicionales como certificaciones, logística, distribución, mantenimiento, etc.
- Costes de marketing, publicidad, varios e imprevistos.

Pos	Descripción	Cantidad	Unidad	Total
01	Gestión de compras (15% de 555€)	1000	83.25	83250,00
02	Montaje y P.M. (35% de 400€)	1000	140,00	140000,00
03	Coste de desarrollo (0,1% de 2500€)	1000	2.50	2500,00
04	Certificación, logística y distribución	1000	5,00	5000,00
05	Imprevistos, varios, marketing, publicidad	1000	3,25	3250,00
			Precio Total € (para 1000 unidades)	**234000,00**

Figura 7.21: Presupuesto de industrialización

Así pues, con la previsión anterior tenemos un coste de industrialización de 234€ por unidad, para una fabricación mínima de 1000 unidades.

Si consideramos aplicar al producto un beneficio bruto de un 17,5%, el precio de venta resultante resulta de 274,95€ unidad (275€ unidad como precio objetivo).

Así pues, en base a las premisas anteriores estimamos **que el precio objetivo del equipo es de 275€ por unidad para una producción mínima de 1000 unidades** (impuestos no incluidos).

8. Conclusiones

8.1. Objetivos

Se han cubierto todos los objetivos principales del proyecto según la lista de tareas final y cronograma final. Bien es cierto que hay objetivos planteado en la lista de tareas iniciales que no han podidos realizarse, bien por falta de tiempo, bien por motivos personales.

A continuación analizamos todos los objetivos propuestos y valoramos el grado de consecución de los mismos.

8.1.1. Objetivos conseguidos

- **Monitorización de las temperaturas del sistema.**

El sistema dispone de dos sensores digitales de temperatura DS18B20 de resolución programable. Se comunican sobre el bus 1-wire y se han implantado de forma que resulte sencillo añadir nuevos sensores al sistema.

Conseguimos la monitorización de las temperaturas del sistema adaptando la librería de ejemplo suministrada por el fabricante, añadiendo alguna particularización a nuestra implementación. Se ha implementado también la tarea vTskAdquireSensor en el módulo principal cuya función se centra únicamente en la adquisición e impresión de los resultados adquiridos sobre el HMI. Para la realización de este fin nos basamos en los módulos onewire.c y crc8.c

- **Monitorización del estado del sistema.**

Conseguimos monitorizar permanentemente el estado del sistema a través de unos leds indicadores que proporcionan información sobre el mismo. Así pues tenemos los indicadores de POWER (alimentación del sistema), CONTROL (el sistema está en regulación), HOT (el sistema está aplicando calor a la planta) y COLD (el sistema está aplicando ventilador a la planta). Estos indicadores corresponden al estado de los relés de control del sistema, de los cuales obtienen su valor. El valor de estos indicadores también es reflejado en el HMI. Para la realización de este fin nos basamos en el módulo iodig.c.

- **Control de potencia disipada por el calefactor.**

Conseguimos el control de la potencia disipada por el calefactor de dos formas distintas y excluyentes, en función del tipo de control seleccionado.

La primera se realiza a través de la conmutación on/off del relé de alimentación del calefactor sin limitación alguna por la señal de control en la base del mosfet. Este tipo de control lo realizamos en las opciones de control manual, on/off y bandas.

La segunda se realiza a través de la salida de control pwm que ataca la base del transistor mosfet del circuito, estando el relé de alimentación activado. Este tipo de control se realiza en la opción de control PID.

En ambos tipos de control nos basamos en el módulo pwm.c y iodig.c desarrollados para este fin.

- **Control de velocidad del ventilador.**

Conseguimos el control de la velocidad de giro del ventilador a través de la aplicación de una tensión en la base del ventilador, proporcional a la velocidad de giro deseada. Para la realización de este fin nos basamos en el módulo dac.c desarrollado para este efecto.

Para la regulación del ventilador también usamos otro método que consiste en la conmutación on/off del relé de alimentación del ventilador sin limitación alguna en la tensión en su base. La activación/desactivación del ventilador proporciona ondas de caudal que ayudan a transportar el calor generado por las resistencias. Para este fin usamos el módulo iodig.c.

- **Sistema autónomo.**

Conseguimos un sistema autónomo gracias a los diferentes algoritmos de control (on/off, bandas, PID) que permiten con diferentes grados de precisión realizar sin supervisión y de forma autónoma la gestión de temperatura de la planta, en base a los parámetros definidos.

- **Sistema retroalimentado.**

Conseguimos optimizar recursos gracias a la retroalimentación del sistema, que aplica de forma proporcional los necesarios para conseguir los objetivos.

8.1.2. Objetivos pendientes

Los siguientes objetivos no han sido realizados por diferentes motivos, y en su momento se decidió apartar de la lista de tareas y objetivos finales. No obstante se reflejan en este apartado porque aparecieren en la planificación inicial.

- **Conectividad inalámbrica (wifi para configurar y reportar datos)**

Dotar al sistema de conectividad inalámbrica a través de un dispositivo WiFly para la configuración o la recuperación de datos.

- **Notificaciones vía mail**

Dotar al sistema de un mecanismo de notificación de incidencias a través de correo electrónico.

- **Web control**

Dotar al sistema de un pequeño servidor web que permita la configuración de los parámetros de la aplicación y el control de la planta, así como la visualización de los datos obtenidos.

8.2. Conclusiones

Un proyecto en lo que prácticamente todo lo que realizas es nuevo conlleva mucho esfuerzo de aprendizaje y paciencia. Las dudas asaltan a cada paso y hasta que no adquieres la habilidad suficiente como para empezar a sentirte cómodo no empiezas a realizar avances significativos. Cada paso debe ser afianzado antes de atreverte a seguir con el siguiente.

Y el reto es triple.

Efectivamente por un lado el hardware representa una novedad. Hasta el momento aunque había oído hablar de sistemas embebidos, nunca me había enfrentado a esta tecnología de una forma seria. He quedado gratamente sorprendido de la cantidad de sistemas, de las diferentes tecnologías, de los "sabores" existente. De la variedad, calidad y economía de periféricos asociados. Hay una floreciente industria en marcha.

Por otro lado el software, el entorno de desarrollo aunque amigable, no deja de tener cierta curva de aprendizaje que se ha de realizar. La configuración de la plataforma y la selección de las librerías adecuadas también puede ser una tarea compleja, sobre todo porque a veces he encontrado en falta indicaciones claras y precisas de cómo realizar el proceso.

Y finalmente el propio proyecto en sí. El tema central del proyecto requiere de un cierto aprendizaje, de una fase de estudio y de implementación, con los inevitables problemas a resolver.

Personalmente creo que he aprendido mucho, y de muchos temas que no sé cómo empezar a enumerar. Y que queda muchísimo por aprender, soy consciente. Y aunque empiezo a sentirme cómodo con esta tecnología sé que apenas estoy arañando la superficie y de que estoy al inicio del viaje.

Personalmente estoy muy satisfecho. He encontrado muchísimas dificultades que he ido resolviendo, con paciencia y perseverancia. He disfrutado del proceso, aunque a veces he estado bajo mucho estrés. He aprendido muchísimo y he empezado a manejar estas tecnología con cierta soltura.

8.3. Autoevaluación

En la realización del proyecto ha habido algunos puntos que han representado una cierta complejidad, en especial los relativos a la lectura de datos de temperatura.

También se han vivido momentos complicados, en el que aparentemente código sin cambios dejaba de funcionar. Afortunadamente se han ido resolviendo encontrando (no sin esfuerzo) la causa de error (tamaño de pila, por ejemplo).

Otro de los problemas encontrado es el uso del modificador volátiles en las variables que pueden ser modificadas por diferentes partes del programa. Esto también ha resuelto muchos problemas.

No soy consciente de la presencia de bugs y errores en el sistema desarrollado. Durante el desarrollo he protegido cada una de las funciones realizadas y he testeado a través del control manual toda la operativa realizada.

No he detectado warnings en el workspace. Aquellos que han ido apareciendo los he ido resolviendo inmediatamente.

El tiempo total dedicado es de unas 300 horas aproximadamente, repartidas entre recolección de información, documentación, desarrollo de código, prototipo, creación de maqueta, pruebas, validación de código, etc.

8.3.1. Reflexión critica

Aunque tengo un elevado grado de satisfacción por el desarrollo del proyecto a la vista de los acontecimientos ocurridos en este tiempo (con todas las dificultades que han acaecido), también es cierto que tengo cierto grado de desilusión por no haber completado todos los objetivos planteados inicialmente.

Tenemos tendencia a subestimar el alcance del trabajo que realizamos, aún más teniendo en cuenta que somos (o eso creo yo) neófitos en la materia. Así nos dejamos llevar por el entusiasmos y abordamos tareas que quedan aún algo lejos de nuestro alcance, por muchos motivos: conocimiento mínimo de la materia, error en el cálculo del tiempo o esfuerzo necesario para completarlas, complejidad oculta, etc.

Lógicamente la suma de los factores anteriores propicia a que solo en raras ocasiones y a través de grandes esfuerzo logren concluirse con alto grado de satisfacción, todos los objetivos propuestos.

En mi caso no he logrado culminar todos los objetivos propuestos inicialmente aunque si todos los principales, y tan solo los extras u opcionales han quedado fuera del desarrollo.

En motivo de este hecho es que durante el semestre académico han ocurrido acontecimientos que me han impedido progresar adecuadamente al ritmo de la planificación. Entre estos acontecimiento el único que merece una reseña es el fallecimiento de mi padre.

8.3.2. Análisis critico

Pese a mis esfuerzos he de reconocer que me ha resultado muy difícil seguir la planificación y la metodología a lo largo del producto, no porque no fuera adecuada, si no por motivos personales que ya he comentado en anterioridad en la obra.

Aun así, siempre que me ha sido posible he seguido la planificación prevista en todas las fases del proyecto.

8.4. Líneas de trabajo futuro

Muchas son las posibles líneas de trabajo futuro dentro del ámbito del proyecto. En primer lugar recuperamos algunos de los objetivos iniciales no realizados, suficientes conocidos y que pasamos solo a enumerar:

- Conectividad inalámbrica (wifi para configurar y reportar datos)
- Notificaciones vía mail
- Web control

Otras nuevas líneas de trabajo futuro:

- Crear una función de auto sintonizado para el controlador PID que inmunice el sistema en caso de programas unos coeficientes de controlador no apropiados.
- Crear nuevos controladores o variantes, por ejemplo el Fuzzi.
- Crear controladores multilazos.
- Poder programar curvas de temperatura. Proporcionar al sistema la habilidad de seguir una curva de temperatura de forma o no periódica (por ejemplo en una planta llevar la temperatura de 26º a las 13:00 a 18º a la 1:00, y viceversa, de forma gradual y periódica).

9. Glosario

A

ADC: Conversión analógica digital

ARM: Arquitectura RISC de microcontroladores desarrollada por ARM Holdings.

B

Baudrate: Velocidad de transmisión en señales segundo.

C

CAN: Controller Area Network. Controlador de red de area.

CMSIS: ARM® Cortex™ Microcontroller Software Interface Standard

Cortex-M3: Microcontrolador de 32-bit ARM.

CPU: Unidad central de proceso

D

DAC: Conversión digital analógica

E

Eclipse: plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones.

E/S: Sistema de comunicación digital entre la CPU y el exterior.

F

Firmware: Aplicación encapsulada en un dispositivo.

FreeRTOS: Sistema operativo de tiempo real para sistemas embebidos.

FW: Siglas de Firmware.

G

GPIO: Entradas y salidas de propósito general.

GUI: HMI. Interfaz Gráfica de Usuario, parte gráfica de una aplicación.

H

HMI: GUI. Interficie hombre máquina.

I

I^2C: Bus de comunicaciones serie unidireccional, con jerarquía maestro-esclavo con dos líneas de datos y una de reloj.

I^2S: Estándar eléctrico de bus serial usado para interconectar circuitos de audio digital.

Interrupción: Señal eléctrica asíncrona enviado por un periférica al procesador.

IP: Dirección que identifica a un elemento en la red.

ISP: Interrupt service process. Rutina de servicio de interrupción.

ISR: Rutina de servicio de interrupción. Pequeño programa que se ejecuta en respuesta a una interrupción determinada.

J

JTAG: Arquitectura usada para testear PCBs utilizando escaneo de límites.

L

LPC: Modelo de placas con arquitectura ARM y microcontrolador NXP CORTEX.

LPC1769: Microcontrolador Cortex-M3 usado en aplicaciones embebidas, con alto nivel de integración y bajo consumo de energía.

LPCXpresso: Entorno de desarrollo bajo Eclipse proporcionado por NXP para desarrollar aplicaciones para sus productos.

M

MCU: Abreviatura de microcontrolador.

Microcontrolador: Circuito integrado capaz de ejecutar las órdenes grabadas en su memoria.

Microprocesador: Circuito central de un sistema informático.

P

Periférico: Elemento hardware que funciona con entrada y salida de información.

PWM: Modulación por ancho de pulsos.

R

RISC: Reduced Instruction Set Code. Diseño de CPU con reducido conjunto de instrucciones.

RS-232: Protocolo de comunicaciones por puerto serie.

RTOS: Real Time Operating System. Sistema operativo en tiempo real.

RTS: Real Time System. Sistema en tiempo real.

S

Sistema embebido: Sistema de computación diseñado para realizar un conjunto limitado de tareas dedicadas, frecuentemente en tiempo real.

Sistema empotrado: Sistema embebido.

SPI: Serial Peripheral Interface. Interficie serie para control de periféricos.

SSP: Synchronous Serial Port. El puerto síncrono serie es un módulo que proporciona una interficie para la comunicación con otros dispositivos pudiendo operar en modo SPI o en modo I^2C.

Stamp Plot Lite V1.7: Es una aplicación freeware para grafiar datos analógicos o digitales de la empresa Parallax BASIC Stamp

U

UART: Hardware que traduce datos entre serie y paralelo.

USB: Universal Serial Bus. Bus estándar de comunicaciones entre computadores y dispositivos periféricos.

W

Wi-Fi: Tecnología de transmisión de redes de área local sin cables.

10. Bibliografía

Sistemes encastats: José María Gómez Cama, Francisco Hernández Ramírez, José López Vicario, Antoni Morell Pérez, Juan Daniel Prades García, Ignasi Vilajosana Guillén, Xavier Vilajosana Guillén, B-19.859-2011, © FUOC • CC-BY-SA • PID_00158468, Barcelona, 2011

Presentació de documents i elaboració de presentacions: Roser Beneito Montagut © FUOC • P08/19018/00446

Redacció de textos cientificotècnics: Nita Sáenz Higueras, Rut Vidal Oltra © FUOC • P08/19018/00445

Treball final de carrera: Antoni Pérez Navarro, Alfons Bataller Diaz, Roser Beneito Montagut, Nita Sáens Higueras, Rut Vidal Oltra © FUOC • P08/19018/00443

Using the FreeRTOS™ Real Time Kernel, © Real Time Engineers Ltd. 2010

The FreeRTOS™ Reference Manual version 8.2.0, © Real Time Engineers Ltd. 2015

Wiki de la asignatura. Página oficial con guías y ejemplos para iniciarse:

http://cv.uoc.edu/webapps/xwiki/wiki/matembeddedsystemslabhome/view/Material/IniciCortexM3?, consultado el10/06/2016

Informacion general:

https://www.google.es/, consultado el 10/02/2016

https://es.wikipedia.org/wiki/Wikipedia:Portada, consultado el 10/06/2016

https://es.wikipedia.org/wiki/Sistema_embebido, consultado el 10/06/2016

ARM

http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php, consultado el 10/06/2016

TERA TERM

http://ttssh2.osdn.jp/, consultado el 10/06/2016

Sistemas empotrados:

Atmel

https://es.wikipedia.org/wiki/Atmel, consultado el 10/06/2016

http://www.atmel.com/, consultado el 10/06/2016

Freescale

https://es.wikipedia.org/wiki/Freescale, consultado el 10/06/2016

http://www.nxp.com/, consultado el 10/06/2016

Microchip

https://es.wikipedia.org/wiki/Microchip, consultado el 10/06/2016

http://www.microchip.com/, consultado el 10/06/2016

NXP

https://es.wikipedia.org/wiki/NXP_Semiconductors, consultado el 10/06/2016

http://www.nxp.com/, consultado el 10/06/2016

STM

https://es.wikipedia.org/wiki/STMicroelectronics, consultado el 10/06/2016

http://www.st.com/content/st_com/en.html, consultado el 10/06/2016

Texas Instruments

https://es.wikipedia.org/wiki/Texas_Instruments, consultado el 10/06/2016

http://www.ti.com/, consultado el 10/06/2016

Zilog

https://es.wikipedia.org/wiki/ZiLOG, consultado el 10/06/2016

https://www.zilog.com/, consultado el 10/06/2016

Sistemas operativos (en tiempo real):

FreeRTOS

https://ca.wikipedia.org/wiki/FreeRTOS, consultado el 10/06/2016

http://www.freertos.org/, consultado el 10/06/2016

https://sourceforge.net/projects/freertos/, consultado el 10/06/2016

http://www.aosabook.org/en/freertos.html, consultado el 10/06/2016

VxWorks

https://es.wikipedia.org/wiki/VxWorks, consultado el 10/06/2016

http://www.windriver.com/products/vxworks/, consultado el 10/06/2016

http://www.embeddedstar.com/weblog/2011/02/28/intel-vxworks/, consultado el 10/06/2016

http://www.robotics-technology-simplified.com/honda-robot.html, consultado el 10/06/2016

RTLinux

https://es.wikipedia.org/wiki/RTLinux, consultado el 10/06/2016

http://web.archive.org/web/20100110062801/http://www.windriver.com/products/platforms/real-time_core/, consultado el 10/06/2016

http://www.rtlinux-gpl.org/, consultado el 10/06/2016

http://www.fsmlabs.com/, consultado el 10/06/2016

QNX

https://es.wikipedia.org/wiki/QNX, consultado el 10/06/2016

http://www.openqnx.com/, consultado el 10/06/2016

http://www.qnx.com/content/qnx/en/products/neutrino-rtos/index.html, consultado el 10/06/2016

ChorusOS

https://es.wikipedia.org/wiki/ChorusOS, consultado el 10/06/2016

http://www.redbend.com/en, consultado el 10/06/2016

http://docs.oracle.com/cd/E19048-01/chorus5/index.html, consultado el 10/06/2016

http://www.oracle.com/technetwork/documentation/legacy-op-sys-193044.html#os, consultado el 10/06/2016

Protocolos de comunicación:

SPI

https://es.wikipedia.org/wiki/Serial_Peripheral_Interface, consultado el 10/06/2016

http://www.epanorama.net/links/serialbus.html, consultado el 10/06/2016

http://www.mct.net/faq/spi.html, consultado el 10/06/2016

CAN

https://es.wikipedia.org/wiki/Bus_CAN, consultado el 10/06/2016

http://www.can-cia.org/, consultado el 10/06/2016

https://www.kvaser.com/about-can/the-can-protocol/, consultado el 10/06/2016

I^2C

https://es.wikipedia.org/wiki/I%C2%B2C, consultado el 10/06/2016

http://www.i2c-bus.org/i2c-bus/, consultado el 10/06/2016

http://www.interfacebus.com/Design_Connector_I2C.html, consultado el 10/06/2016

USB

https://es.wikipedia.org/wiki/Universal_Serial_Bus, consultado el 10/06/2016

http://www.usb.org/home, consultado el 10/06/2016

https://en.wikipedia.org/wiki/USB_On-The-Go, consultado el 10/06/2016

http://www.luisandradehd.com/2016/03/usb-otg/, consultado el 10/06/2016

http://www.usb.org/developers/onthego, consultado el 10/06/2016

I^2S

https://es.wikipedia.org/wiki/I%C2%B2S, consultado el 10/06/2016

http://web.archive.org/web/20080706121949/http://www.nxp.com/acrobat_download/various/I2SBUS.pdf, consultado el 10/06/2016

http://web.archive.org/web/20070928163203/http://www.anthemav.com/OldSitev1/pdf/i2Srev1.pdf, consultado el 10/06/2016

http://www.edn.com/design/consumer/4390981/Common-inter-IC-digital-interfaces-for-audio-data-transfer-, consultado el 10/06/2016

JTAG

https://es.wikipedia.org/wiki/JTAG, consultado el 10/06/2016

http://www.boundary-scan.co.uk/, consultado el 10/06/2016

http://hri.sourceforge.net/tools/jtag_faq_org.html, consultado el 10/06/2016

http://web.archive.org/web/20110713022446/http://www.inaccessnetworks.com/ian/projects/ianjtag/jtag-intro/jtag-intro.html, consultado el 10/06/2016

https://www.xjtag.com/about-jtag/jtag-high-level-guide/, consultado el 10/06/2016

http://www.jtagprogrammer.com/, consultado el 10/06/2016

RS232

https://es.wikipedia.org/wiki/RS-232, consultado el 10/06/2016

EIA standard RS-232-C: Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange. Washington: Electronic Industries Association. Engineering Dept. 1969. OCLC 38637094

RS485

https://es.wikipedia.org/wiki/RS-485, consultado el 10/06/2016

http://web.archive.org/web/20070626040348/http://www.wut.de/e-6wwww-11-apes-000.php3, consultado el 10/06/2016

https://www.maximintegrated.com/en/app-notes/index.mvp/id/763, consultado el 10/06/2016

ONEWIRE (1-WIRE)

https://es.wikipedia.org/wiki/1-Wire, consultado el 10/06/2016

http://www.1wire.info/, consultado el 10/06/2016

https://www.maximintegrated.com/en/products/digital/one-wire.html, consultado el 10/06/2016

https://www.maximintegrated.com/en/app-notes/index.mvp/id/1796, consultado el 10/06/2016

ETHERNET

https://es.wikipedia.org/wiki/Ethernet, consultado el 10/06/2016

https://www.rfc-es.org/rfc/rfc0894-es.txt, consultado el 10/06/2016

http://es.ccm.net/contents/672-ethernet, consultado el 10/06/2016

http://www.ethernetalliance.org/, consultado el 10/06/2016

11. Anexos

11.1. Esquemas de montaje

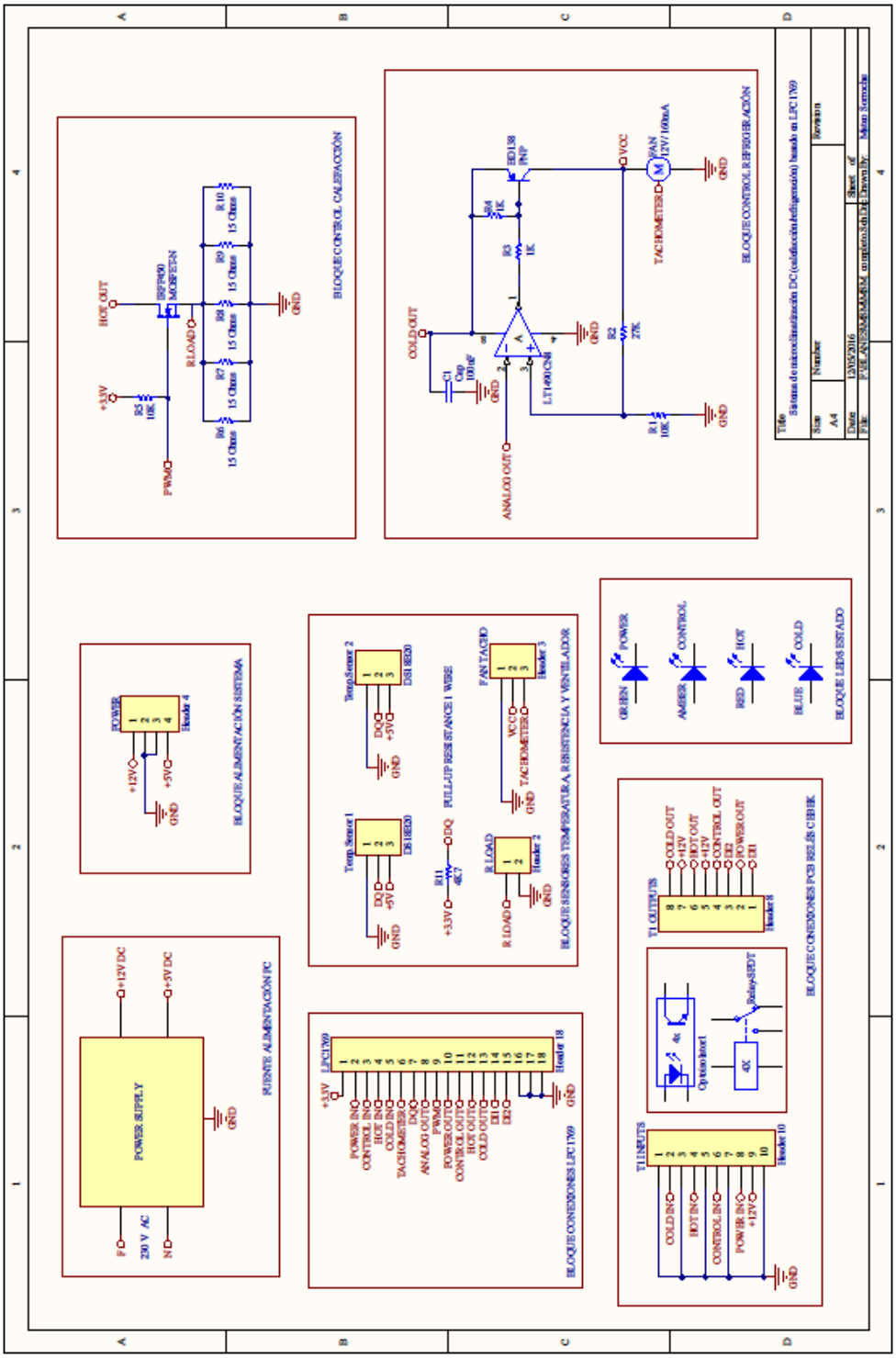


Figura 0.22: Esquema de montaje.

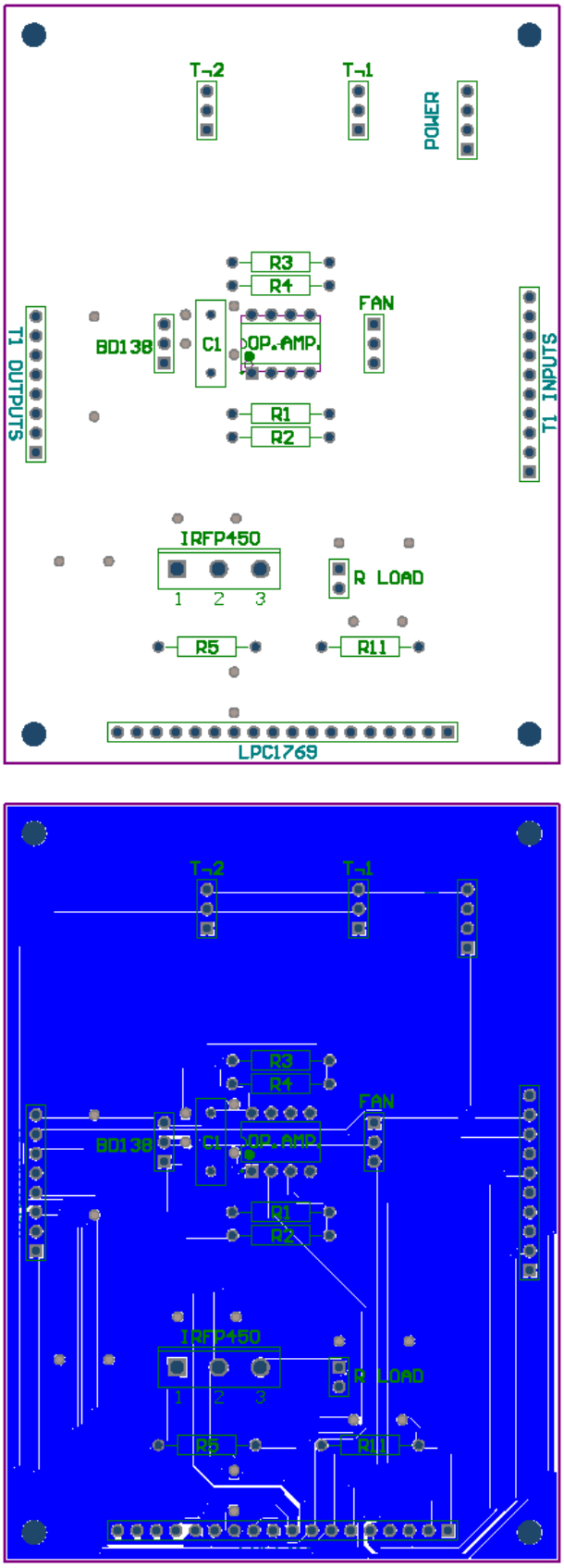
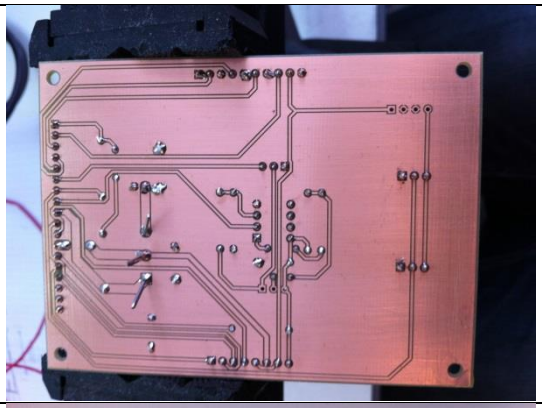
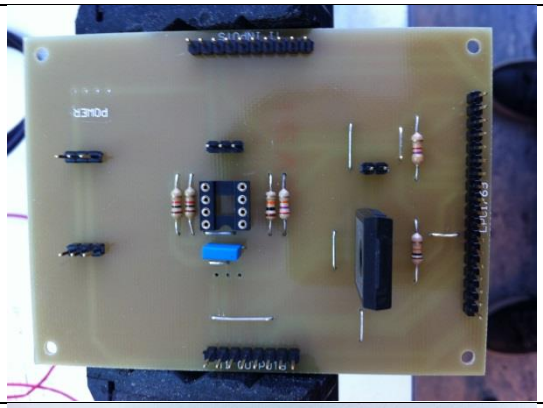
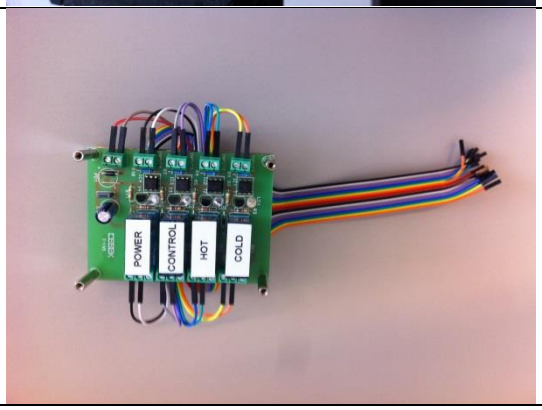


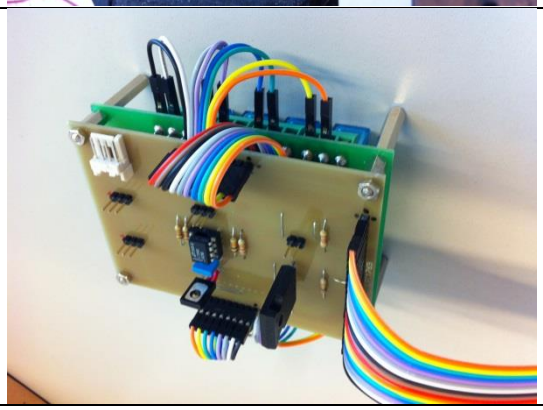
Figura 0.23: Placas PCB.

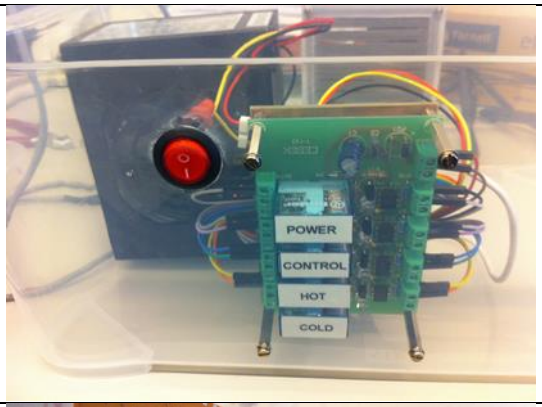
11.2. Fotografias prototipo

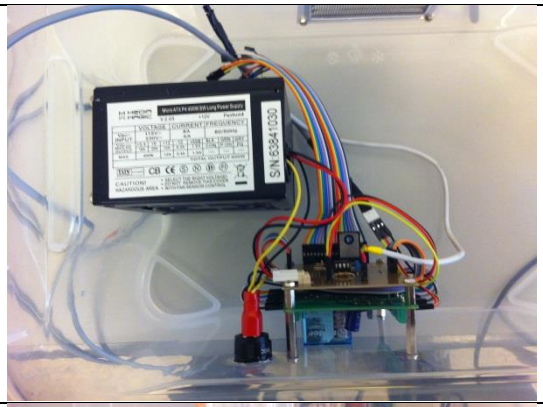


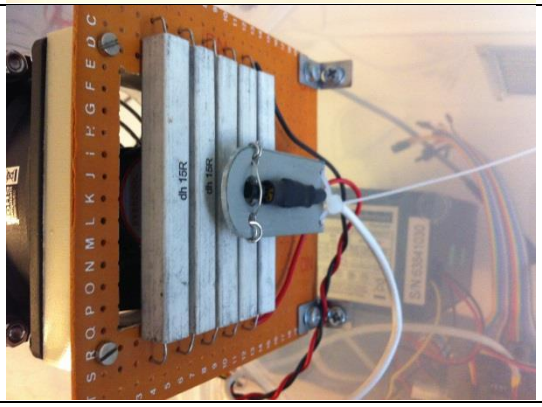












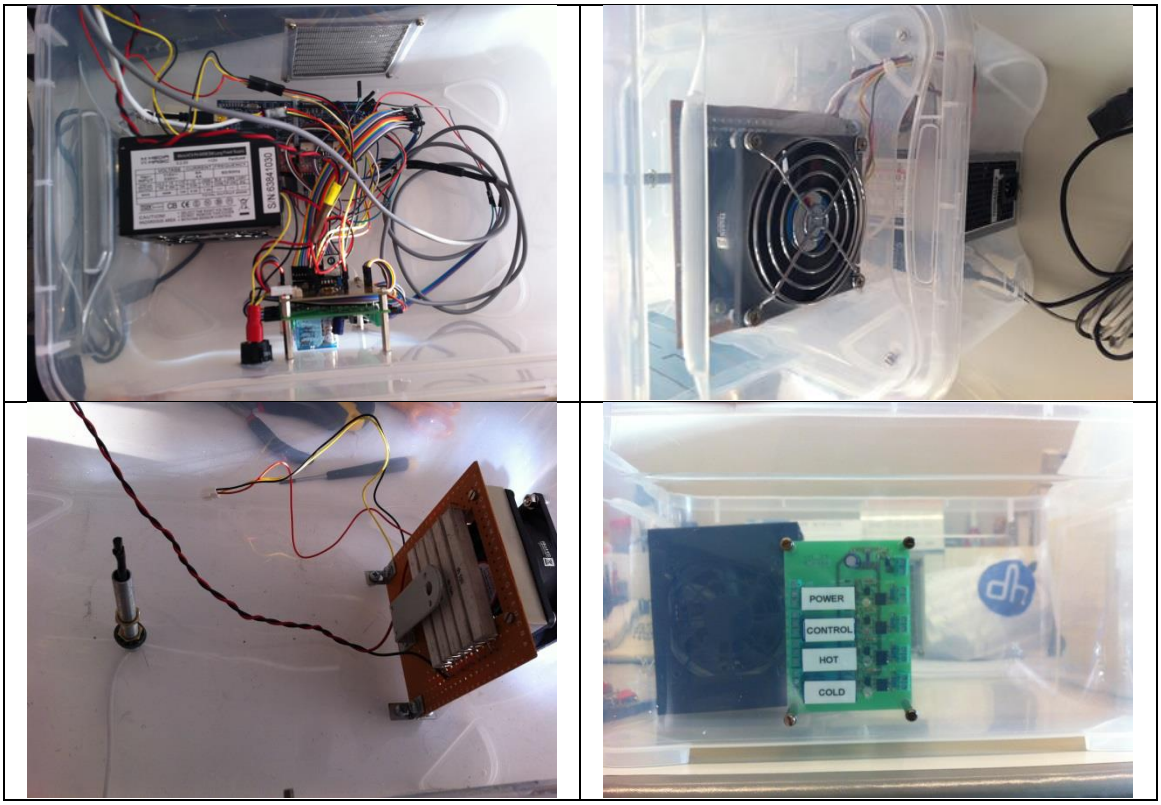


Figura 0.24: Fotografías.