# PFC-Sistemas empotrados

## Temporizador de tiempo de rearme caldera y control temperatura sala de control

**Autor**
Antonio Barquero Rodríguez
"Enginyería d'Informàtica "

**Memoria**
PFC-Sistemas empotrados

**Consultor**
Sebastiá Cortes Herms

**Fecha de entrega**
5 de enero de 2014

# *Agradecimientos*

A mi familia por el tiempo que les he quitado y la paciencia que han tenido conmigo.

Al consultor por el apoyo recibido y los ánimos que me ha dado.

*"If a first you don't succeed, try try again."*
*(Proverb)*

# *Resumen*

Este proyecto corresponde a una de las asignaturas de pràcticum de la UOC, la cual está pensada para aplicar los conocimientos teóricos adquiridos a lo largo de las diferentes asignaturas cursadas. Está elección fue motivada por el puesto de trabajo que desempeño en una empresa privada del ramo textil.

En esta empresa trabajo como operador de calderas, en que mi función es de controlar y supervisar el funcionamiento de la caldera. Los controles de la caldera son llevados a cabo por mecanismos electrónicos que envían los datos a un sistema informático para su control. Al descubrir este área entre las asignaturas de pràticum, me animé a realizar la para ver como funciona este tipo de dispositivos y que lenguaje usan para su programación.

El proyecto desarrollado consiste en el control de tiempo que queda para el rearme de la caldera, que es un máximo de dos horas, que es lo marcado por la legislación vigente sobre seguridad en las calderas industriales de vapor.

Y también en el control de la temperatura de la Sala de Control que es donde se encuentran todos los sistemas electrónicos y de control, los cuales han de permanecer a una temperatura estable por debajo de los 30 grados. Así, nos aseguramos de su correcto funcionamiento ya que las temperaturas altas influyen sobre el rendimiento de los aparatos electrónicos, y pueden ocasionar daños imprevisibles.

El lenguaje utilizado es el C (el MinGW para Windows), java para la aplicación web y el LPCXpresso basado en el Eclipse. Este ultimo está basado en el *Cortex-M3* de ARM y como sistema operativo el FreeRTOS (real Time Kermel).

La arquitectura utilizada es el **LPCX1769** Board, el **Wifly** module RN171 y el UART to USB converter based on **CP2102.**

El funcionamiento consiste en el control del tiempo y temperatura por parte del LPC1769. La aplicación insertada en el LPC1769 recibe los datos del sensor de temperatura y controla el tiempo transcurrido. Por otro lado, envía los datos a la web para su consulta en tiempo real y envío de SMS de alarma, mediante el Wifly.

# *Indice de contenidos*

1. Introducción..... 6
1.1. Justificación ..... 6
1.2. Descripción del proyecto ..... 6
1.3. Objetivos del proyecto ..... 6
1.4. Metodología ..... 7
1.5. Planificación ..... 8
1.6. Recursos empleados ..... 8
1.7. Productos obtenidos ..... 12

2. Antecedentes ..... 13
2.1. Estado del arte ..... 13
2.2. Estudio de mercado ..... 13

3. Descripción funcional ..... 16

4. Descripción detallada..... 17
4.1. Hardware ..... 17
4.2. Software ..... 18
4.2.1. El programa principal ..... 18
4.2.2. Aplicación web ..... 19
4.2.3. Anolog to Digital Converter (ADC) ..... 24
4.2.4. General Purpose Inpu/Ouput (GPIO) ..... 25
4.4.5. UART ..... 27
4.2.6. Driver de comunicación con WiflyRN171 ..... 27

5. Viabilidad técnica ..... 28

6. Valoración económica ..... 29

7. Conclusiones ..... 30
7.1. Propuesta de mejoras ..... 30
7.2. Autoevaluación ..... 30

8. Glosario ..... 32

9. Bibliografía ..... 34

10. Anexos ..... 36
10.1. Clase AddStats.java de Arp@Lab ..... 36
10.2. Guía del usuario ..... 37
10.2.1. Descargar, construir y debug ..... 37
10.2.2. Crear librería estática ..... 41
10.2.3. Control temperatura y rearme caldera ..... 44

# *Tabla de figuras*

Figura 1. LPC1769 ..... 9
Figura 2. Wifly RN-XV 802 11 b/g ..... 9
Figura 3. UART CP2102 ..... 10
Figura 4. Led tricolor ..... 10
Figura 5. Led difuso ..... 10
Figura 6. Pulsador ..... 10
Figura 7. Led verde ..... 11
Figura 8. TMP36 ..... 11
Figura 9. Breadboard ..... 11
Figura 10. Temporizador XTD 212 ..... 14
Figura 11. Panel control con temporizador ..... 14
Figura 12. Caldera industrial ..... 15
Figura 13. Conexiones del hardware ..... 17
Figura 14. Diagrama bloques ..... 18
Figura 15. Web de envío SMS ..... 20
Figura 16. Formulario ..... 21
Figura 17. Respuesta formulario ..... 21
Figura 18. Datos temperatura ..... 22
Figura 19. Visualización minutos paro caldera ..... 22
Figura 20. Captura teléfono móvil ..... 23
Figura 21. Mensajes enviados ..... 24
Figura 22. Quickstart panel ..... 38
Figura 23. Import archived projects (zip) ..... 38
Figura 24. Build controlTemperature ..... 39
Figura 25. Buscar tarjeta LPC1769 ..... 39
Figura 26. Erase Program Flash ..... 40
Figura 27. Program flash memory ..... 40
Figura 28. LPCXpresso C Project ..... 41
Figura 29. New LPCXpresso C Project ..... 42
Figura 30. Project name ..... 43
Figura 31. Select target MCU ..... 43
Figura 32. Select options for C project ..... 44
Figura 33. Cliente web ..... 45
Figura 34. Selección sensor ..... 46

# 1. Introducción

## 1.1 Justificación

El presente proyecto final de carrera nace de la necesidad de optimizar costes en el control y vigilancia de la caldera industrial, y la "Sala de Control" donde se ubican los equipos de control.

Por este motivo, este proyecto nos permitiría realizar otras tareas o desplazarnos un poco mas lejos del centro de control visual de la caldera. Y si por algún motivo nos descuidamos del rearme se nos avisará con un *mensaje de texto al móvil (SMS)* indicándonos que nos queda poco tiempo para el rearme de la caldera, que son 120 minutos. También, vigilará la temperatura ambiente de la *Sala de Control* donde están ubicados los ordenadores y demás sistemas eléctricos de control. Una elevada temperatura puede ocasionar mal funcionamiento o avería de los equipos electrónicos.

## 1.2 Descripción del proyecto

El sistema nos permite realizar un seguimiento de la temperatura de la Sala de Control y del tiempo que nos queda para rearmar la caldera.

El tiempo de rearme sera controlado por una pre-alarma que encenderá un led de color azul, a los 90 minutos, y enviará una señal a la web http://tonibarquerodri.appspot.com. Desde la web se enviará un mensaje corto de texto (SMS) al móvil del operario para informar sobre el tiempo de rearme. Además, cuando se llega a los 120 minutos se para la caldera, por lo que se encenderá un led de color rojo y se enviará una señal a la web indicando esta situación.

En la Sala de Control colocaremos otro dispositivo para que nos controle la temperatura. Enviara a la web la temperatura de la sala cada 20 segundos. Por otro lado, si la temperatura supera los 30 grados enviara una señal de alarma, y encenderá un led de color rojo.

El servidor donde tenemos la aplicación web se encargará de mostrar la temperatura enviada, y las señales de alarma. Además, enviara un SMS de alarma al móvil del responsable de la vigilancia y control, para que quede avisado y realice las acciones mas pertinentes.

El dispositivo de control del tiempo consta de un botón para que el lpc1769 vuelva a contar desde cero, y realiza un reset sobre las alarmas surgidas.

## 1.3 Objetivos del proyecto

El objetivo principal es disponer de un dispositivo de control del rearme de la caldera, y control de la temperatura de la Sala de Control. Los objetivos que debe satisfacer el sistema son los siguientes:

- Contar el tiempo que queda para el paro de la caldera
- Enviar señal de pre-alarma al servidor y encender un led de color azul
- Enviar señal de alarma de tiempo consumido y encender led de color rojo
- Adquirir datos del sensor de temperatura y enviarlos al servidor
- Enviar señal de alarma si la temperatura es superior a 30 grados y encender led de color rojo
- Al pulsar el botón se vuelve a contar desde cero el tiempo que queda para el paro caldera, y si existe alarma de temperatura también la resetea
- La aplicación instalada en el servidor mostrara los datos y enviara un SMS si existe alguna alarma tanto de tiempo como de temperatura

## 1.4 Metodología

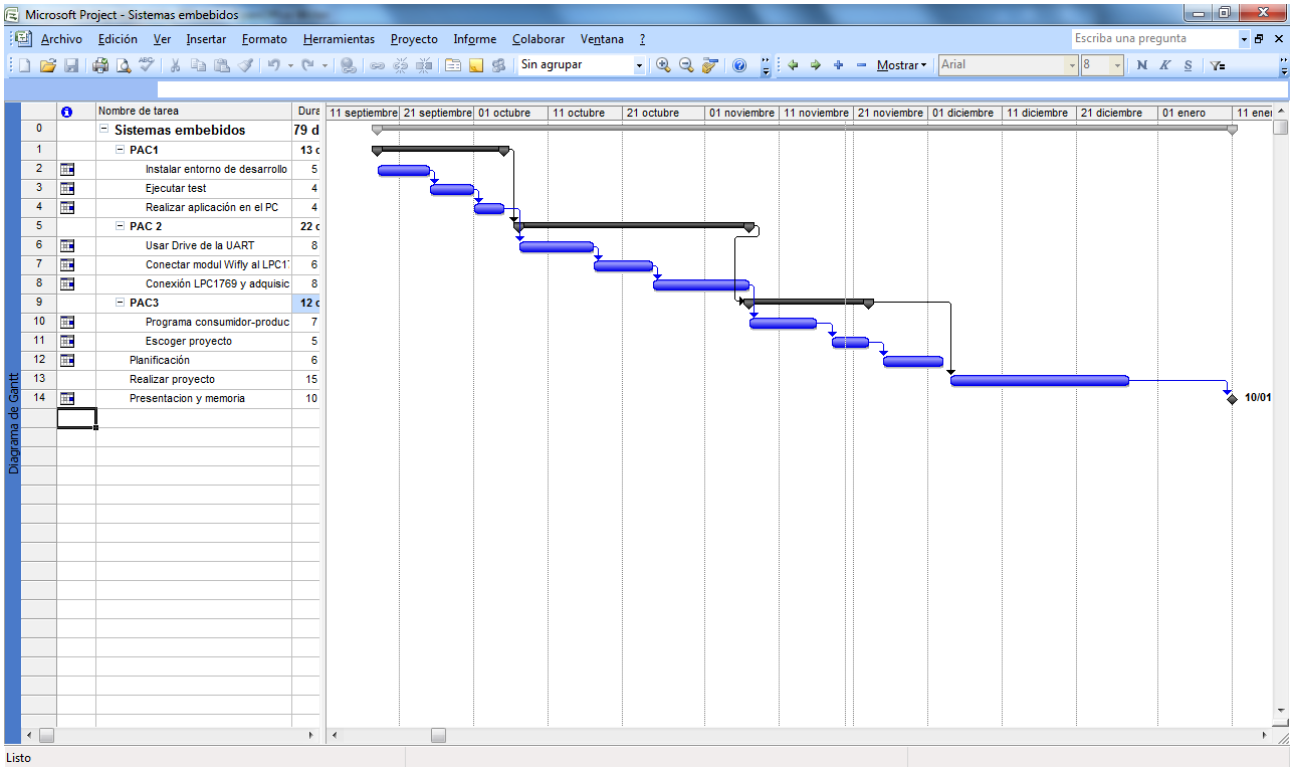
El método de trabajo seguido ha sido la realización de los ocho puntos marcados en la planificación del semestre, distribuidos en tres PACs. Se han ido trabajando punto por punto y no se ha pasado al siguiente hasta que no se ha conseguido realizar el punto anterior. Las PACs realizadas son las siguientes:

| Nombre del subsistema | Descripción de tareas |
|---|---|
| PAC1 | Instalar el entorno de desarrollo y familiarizarse con el SW y el HW.<br>Ejecutar el test de USB_UART_WIFI.<br>Realizar aplicación en el PC que envíe datos a Internet usando la API http://openarpalab.appspot.com/ y el modulo WIFI_UART. |
| PAC2 | Realizar aplicación en el LPC1769 y que el printf llegue al PC.<br>Conectar el modulo WIFLY al LPC1769 y realizar un driver de comunicación para conectarse a Internet.<br>Conexión del sensor MMA7361 Accelerometer al LPC1769 y adquisición de datos. |
| PAC3 | Realizar un programa que siga el ejemplo clásico de productor-consumidor.<br>Escoger el proyecto a realizar.<br>Planificación del proyecto. |
| Realizar proyecto | Programar, montar elementos y realizar pruebas. |
| Memoria | Redactar la memoria del proyecto. |
| Presentación | Realizar presentación en PowerPoint y video de funcionamiento. |

## 1.5 Planificación

La jornada de trabajo suele ser de 8 horas, si es una dedicación exclusiva, pero como he tenido que compaginar el trabajo y la vida familiar , le he dedicado una media de dos o tres horas.

Por suerte he podido cumplir con la planificación propuesta, y acabar en las fechas señaladas.




## 1.6 Recursos empleados

### Hardware

- **LPC1769**
  La placa está compuesta de dos partes, la placa del micro-controlador y la electrónica necesaria para su funcionamiento o target board, y la placa del programador y depurador o JTAG.

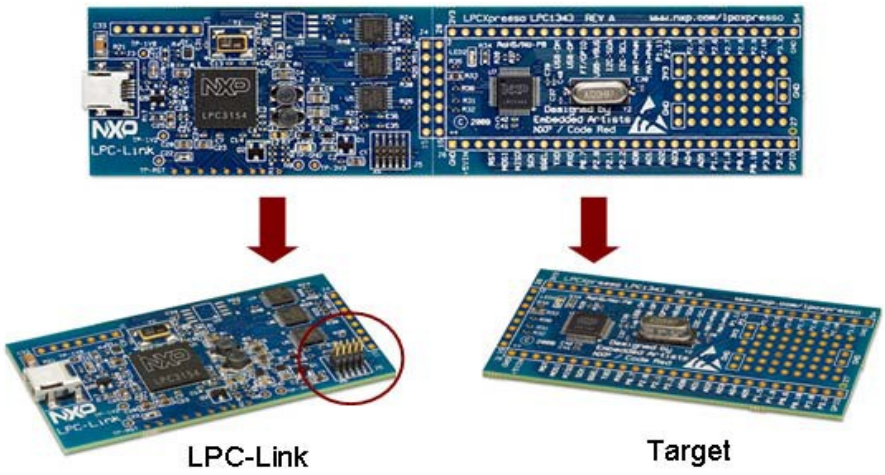



*Figura 1 . LPC1769*

- **Wifly RN-XV 802 11b/g**
  El modulo Wifly dispone de antena para la conexión wifi, con lo que nos permite conectarnos con redes wifi TCP/IP estándar mediante la interface serie RS-232. Debido a su pequeño tamaño y su poco consumo de energía, es perfecto para aplicaciones móviles inalambrícas, tales como seguimientos de activos, sensores y dispositivos portátiles.


*Figura 2. Wifly RN-XV 802 11b/g*

- **Xbee pin adapter**
  Es un adaptador para el módulo Wifly, ya que la distancia entre los pines es mas pequeña en el Wifly que en los demás dispositivos.

- **UART to USB converter based on CP2102**
  El CP2102 incluye un puerto USB 2.0 de alta velocidad con función controlador, transmisor-receptor USB, oscilador, EEPROM, y el bus de datos asíncrono (UART), con señales de control de módem. La EEPROM se puede utilizar para personalizar el USB Vendor ID, ID del producto, Product Description String.

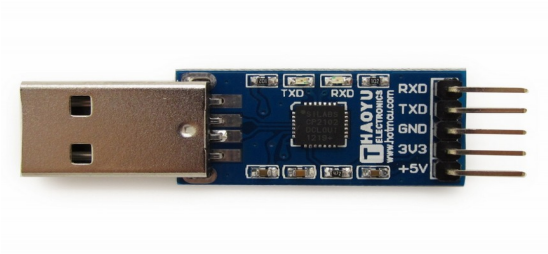



*Figura 3. UART CP2102*

- **Diodo LED Tricolor RGB**
  La patilla más larga es el cátodo y las tres otras son ánodos, uno por cada color.

*Figura 4. Led tricolor.*

- **Diodo LED Tricolor RGB Difuso**
  Este led es opaco, es decir difuso, para difuminar mejor la luz emitida. La distribución de las patillas es igual que el anterior.

*Figura 5. Led difuso.*

- **Pulsador switch 12mm**
  Pulsador normalmente abierto que cuando se pulsa se cierra el circuito.

*Figura 6. Pulsador.*

- **LED de color verde**

*Figura 7. Led verde.*

- **Sensor de temperatura TMP36**
  Es un sensor de temperatura analógico con un voltaje de corte proporcional a la temperatura absoluta con un rango de 10mv/ºk.

*Figura 8. TMP36.*

- **Breadboard**
  Es una placa perforada con conexiones internas en la que podemos conectar las patas de nuestros componentes electrónicos o cables sin tener que realizar ninguna soldadura.

*Figura 9. Breadboard.*

**Software**

- LPCXpresso IDE v5.1.2_2065 basado en Eclipse
- Sistema operativo en tiempo real FreeRTOS
- Lenguaje de programación C, el "mingw32" version 4.6.2 (GCC)
- Eclipse INDIGO con java
- Google App Engine
- Windows 7
- Portátil HP Pavilion

## 1.7 Productos obtenidos

Se han obtenido los siguientes productos:

- Un sistema empotrado formado por la placa LPC1769, sensor de temperatura, leds, botón, modulo Wifly, y manejado por el sistema operativo en tiempo real FreeRTOS.
- Aplicación web que guarda los valores enviados por el sistema empotrado, y según el valor recibido envía un SMS.
- Aplicación escrita en C en el PC, que se utiliza para configurar el Wifly y realizar pruebas de conexión con Internet.

# 2. Antecedentes

## 2.1 Estado del arte

Un sistema embebido es una plataforma con recursos muy limitados si lo comparamos con un PC. Por este motivo, no tienen un sistema operativo completo, sino un subconjunto de funciones que se ejecutan solo en momentos determinados del programa.

El componente central del sistema es la CPU, la cual controla todo el sistema. Por otro lado, incorporan otros dispositivos como conversores ADC, DAC, controladores Ethernet, PWN y otros protocolos de comunicación como UART, I2C, SSP, CAN.

Nuestro sistema embebido utiliza el OS FreeRTOS ( sistema operativo en tiempo real). Que esta formado por un conjunto de aplicaciones que nos ayudan a gestionar los recursos de hardware disponibles, como el tiempo del procesador y la memoria.

Los componentes utilizados en nuestro sistema son los siguientes:

**LPC1769**
Es un micro-controlador ARM Cortex_M3 de 32 bits, con 512 KB de memoria flash. La placa está compuesta de dos partes, la placa del microcontrolador y la electrónica necesaria para su funcionamiento o target board, y la placa del programador y depurador o JTAG.
La placa se puede alimentar mediante una fuente externa de entre 3.15V y 3.3V o desde el conector usb del JTAG de 5V.

**Wifly RN-XV 802 11b/g**
El modulo Wifly dispone de antena para la conexión wifi, con lo que nos permite conectarnos con redes wifi TCP/IP estándar mediante la interface serie RS-232. Esto nos permite conectarnos con el LPC1769 para el envío de datos al servidor.

**CP2102**
Este conector nos permite realizar comunicaciones bidireccionales entre el PC y el LPC1769 o el Wifly. También lo hemos usado como fuente de alimentación para alimentar el Wifly o el LPC1769. También nos ha servido para realizar pruebas del código realizado en el LPC1769 con el PC.

## 2.2 Estudio de mercado

En el mercado no encontrado un temporizador que controle el tiempo de rearme de la caldera y envíe los datos a Internet, y de paso un SMS de alarma. He encontrado la empresa que se dedica a realizar automatismos industriales y tiene el temporizador que lleva incorporado la caldera industrial donde trabajo.

No disponía de la documentación sobre el temporizador, lo he buscado en la web, y tampoco esta

disponible, a no ser que compres el producto, y su precio es de 114 €. El temporizador que utiliza la caldera es el siguiente:


Figura 10. Temporizador XTD 212

Es un temporizador especial calderas de un intervalo fijo de 120 minutos de cuatro cifras, con pulsador de rearme y reset. Además, dispone de un reset exterior, con pre-alarma ajustable antes de llegar al limite de los 120 minutos. Puede ser alimentado con corriente continua o alterna, los rangos van de 12 voltios a 220 v.


*Figura 11. Panel control con temporizador*

La idea principal era que este temporizador suministrara al LPC1769 las señales de pre-alarma y alarma, y este enviara una señal de alarma mediante un SMS. Pero, dada la imposibilidad de realizar este tipo de pruebas decidí que el LPC1769 controlara el tiempo de rearme y nos enviara señales de pre-alarma y alarma mediante la iluminación de leds, y de paso enviar un SMS.

Los temporizadores están limitados a 120 minutos por la normativa referente a la seguridad de las calderas industriales, y paran la caldera sino se persona presencialmente una persona donde esta ubicada la caldera, y readrme dicho temporizador. Se sigue este procedimiento para obligar al operario que haga una vigilancia visual de los parámetros de funcionamiento de la caldera, ya que los sistema de control a distancia pueden producir errores de lectura y no avisar de los posibles peligros que existan.

*Figura 12. Caldera industrial.*

# 3. Descripción funcional

El sistema esta pensado para que se instale un conjunto LPC1769, Wifly y sensor de temperatura en la Sala de Control, que es donde están situados los sistemas de control. Para que controle la temperatura de la sala y envíe un mensaje de alarma por alta temperatura.

Por otro lado, instalar un conjunto LPC1769, Wifly y temporizador en la sala de calderas para el control del rearme y el envío de señales de alarma mediante un SMS.

Luego está el servidor de Google App que es donde se ha instalado la aplicación Arp@Lab, y se ha modificado para enviar mensajes cortos de texto (SMS). Es decir, se ha incluido un par de funciones para enviar los SMS.

Al final, hemos realizado todas las conexiones con una sola target LPC1769 y un modulo Wifly con una placa Breadboard.

# 4. Descripción detallada

## 4.1. Hardware

Ya hemos explicado el hardware utilizado con lo que las conexiones utilizadas entre los distintos dispositivos son las siguientes:

| WiflyRN171 | LPC1769 |
|---|---|
| TX Pin 2 | J6.10 (RXD3) |
| RX Pin 3 | J6.9 (TXD3) |

| WiflyRN171 | CP2012 |
|---|---|
| VDD 3V3 Pin 1 | 3V3 |
| GND Pin 10 | GND |

Así es como queda montado el hardware:

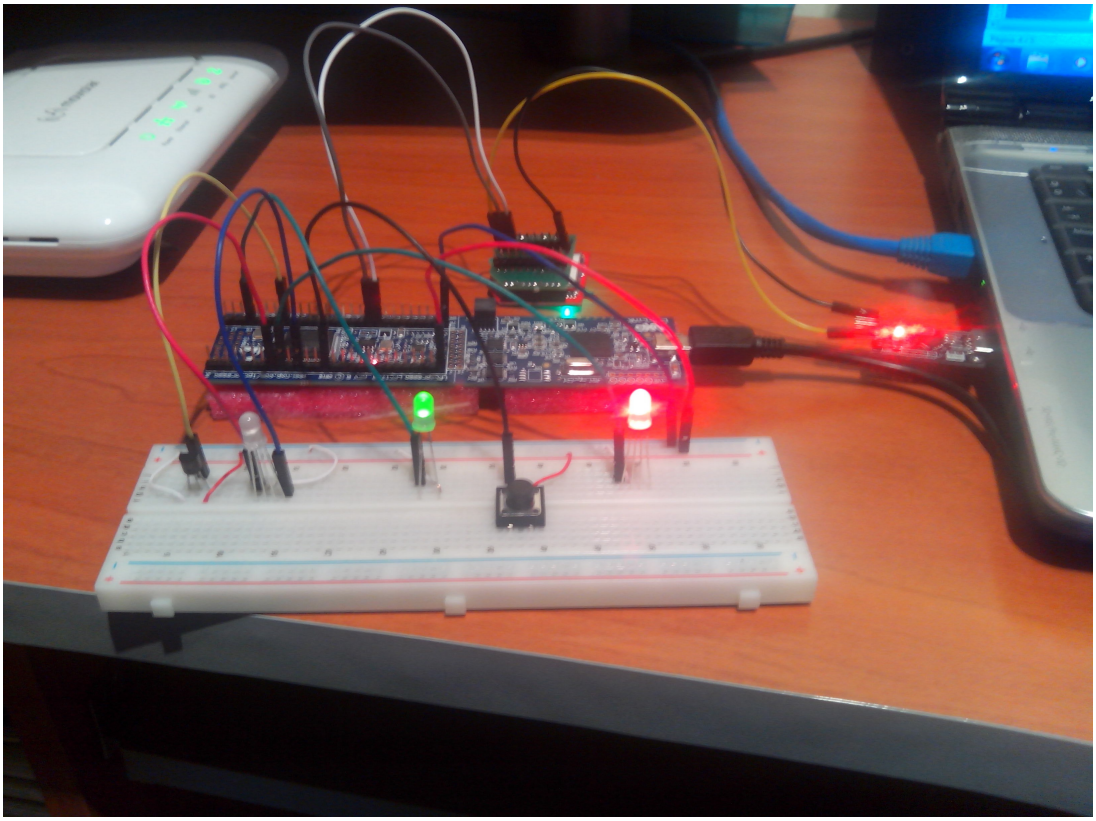


*Figura 13. Conexiones del hardware*

A continuación podemos ver un diagrama de como se relacionan las diferentes partes del proyecto:

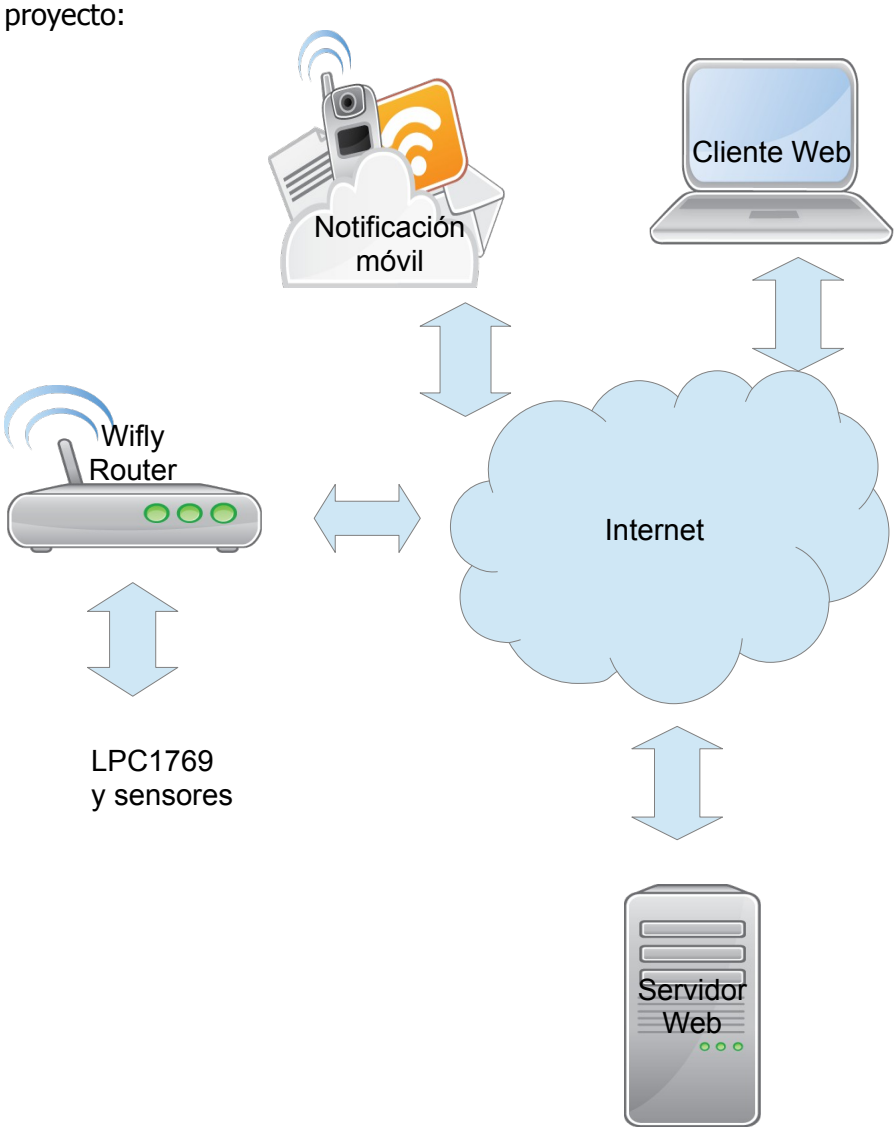



*Figura 14. Diagrama de bloques*

## 4.2. Software

### 4.2.1 El programa principal

Está compuesto de tres tareas, dos productoras de datos y una consumidora. La función *main()* es la primera función que se ejecuta, aquí es donde se inicializa el sistema, creamos las diferentes tareas y se lanza el gestor de tareas o Scheduler.

Una vez lanzado el scheduler, ya no podemos decidir que tarea ejecutar, lo decide el propio gestor de tareas. Por lo tanto, por este motivo debemos poner prioridades a las tareas para saber cual ejecutar en cada momento. Por este motivo, he puesto a la receptora de datos la prioridad mas baja para que se ejecute cuando tenga un dato en la cola.

La prioridad mas baja es el 0 la cual tiene una función especial y es utilizada por una tarea llamada "idle" u ociosa. Cuando ninguna de las tareas requiere el procesador, el sistema ejecuta esta tarea. Esto permite fácilmente contabilizar el nivel de ocupación del CPU, poner el mismo en modo bajo consumo o correr cualquier tarea que pudiera ser de utilidad para el sistema cuando no debe atender ninguno de sus eventos.

La tarea que mas utiliza la UART es la que controla la temperatura, ya que enviamos la lectura del valor a la web. Le he puesto un retraso de 20 segundos para que tome los valores de temperatura cada 20 segundos. Y a la tarea que controla el tiempo un retraso de 10 segundos. Así, nos aseguramos que están bien sincronizadas y la que envía los datos no sera interferida en su funcionamiento hasta que haya enviado un dato a la red.

La función utilizada es *"vTaskDelayUntil()",* la cual pone en estado **Blocked** la tarea hasta que espira la cantidad de ticks indicados. Cada vez que expira un periodo de ejecución, el scheduler revisa si alguna tarea de mayor prioridad a la actual sale del estado blocked, y paso al estado ready.

Por otro lado, la tarea receptora solo recibirá un valor a enviar, y estará bloqueada indefinidamente hasta que reciba un valor en la cola. Esto se consigue con la función *portMAX_DELAY*, y poniendo a 1 el *INCLUDE_vTaskSuspend* que está incluido en el *FreeRTOSConfig.h.*

La tarea que controla el tiempo de rearme caldera, se realiza mediante la función *xTaskGetTickCount,* y almacena el valor en una variable de tipo portTickType. El tiempo se mide en ticks y esta variable esta definida como un tipo de 32 bits sin signo, ya que el *configUSE_16_BIT_TICKS* está a cero.

Con esta variable de 32 bits podemos contar el tiempo hasta 1193 horas:

2^32 = 4294967296 / 1000Hz = 4294967,296 segundos / 120 = 1193 horas.

Si nuestro sistema ha de estar funcionando mas de ese tiempo, debemos tener en cuenta el *overflow,* por lo tanto tendremos que añadir una rutina que lo tenga en cuenta.

### 4.2.2 Aplicación web

He utilizado la aplicación Arp@Lab de la UOC y la he instalado en el *Google App Engine.* La utilizo para enviar la lectura del sensor de temperatura y los avisos de alarma. Una vez instalada he buscado por Internet una web que me permita enviar SMS, y me he dado de alta, como se puede ver en la figura siguiente:


*Figura 15. Web de envío SMS*

He tenido que comprar créditos para poder enviar mensaje. Cuanto mas dinero te gastes mas económico sale el crédito. Cada crédito equivale a un mensaje.

Una vez hecho esto , he intentado enviar una petición con el Wifly y me daba error. El problema reside en que el comando *"set com remote"* solo acepta un string de 32 bytes. Este tamaño es insuficiente para poder enviar la petición correctamente. Por lo tanto, he tenido que incluir un par de funciones en código java en la aplicación web, para cuando reciba una señal de alarma envíe un SMS. Las funciones incluidas son las siguientes:

- **private String sendSMS(String value):** Recibe el mensaje a enviar y lo envía.
- **private String readStream(InputStream in):** Este método es para recibir el streaming de la respuesta del servidor (o response).

Estas funciones las he incluido en la clase ***AddStats.java.*** Para realizar pruebas sin tener que utilizar el Wifly, he creado un formulario que envía una petición GET, y nos visualiza la respuesta.

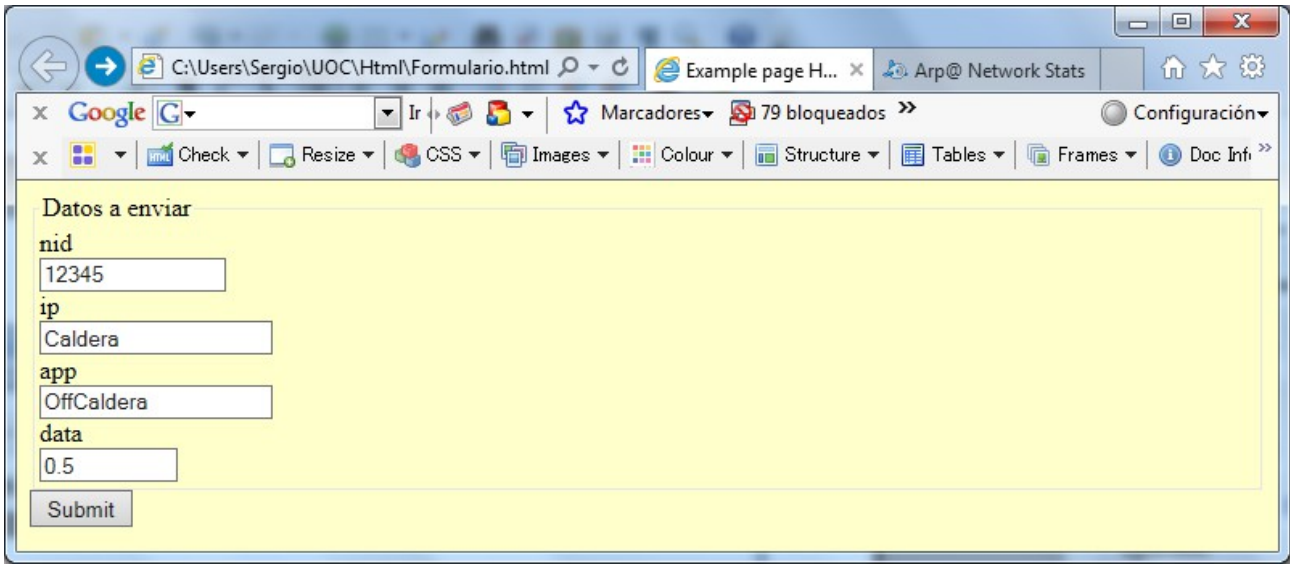



*Figura 16. Formulario*

Y la respuesta que nos da es la siguiente:

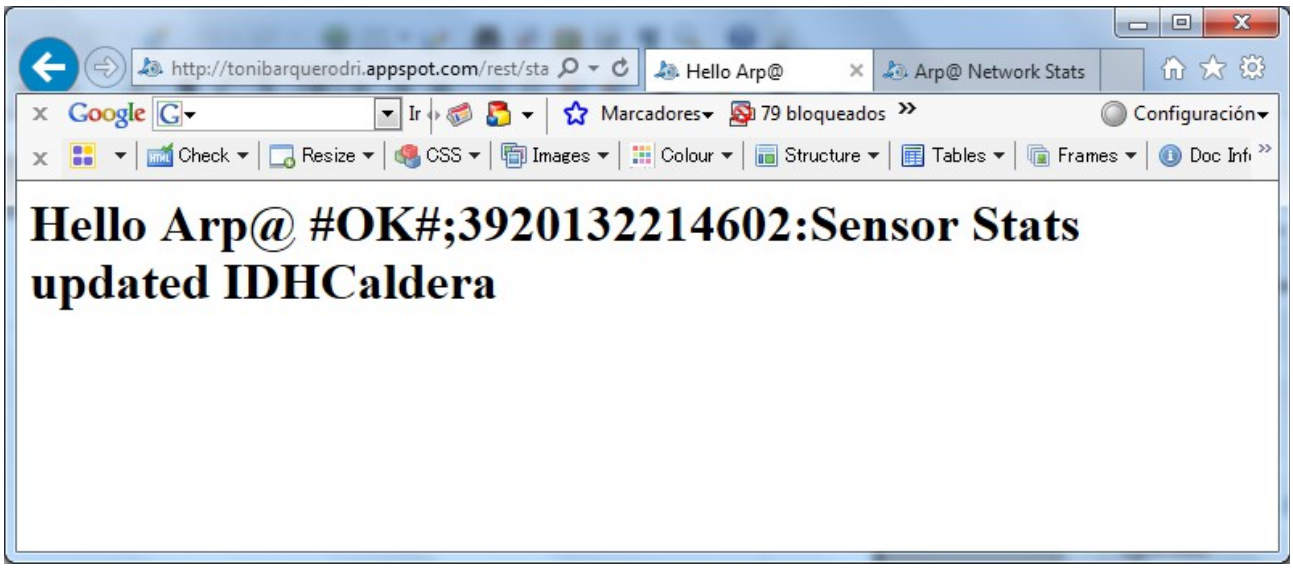



*Figura 17. Respuesta formulario.*

La respuesta del envío a la petición realizada es la siguiente:

- ✔ **#OK#:** Indica que se ha enviado correctamente al móvil.
- ✔ **3920132214602:** Es el código de identificación del sms para logs y consultas de estado.

Hay que tener en cuenta que la primera vez que se envía el valor de un sensor y no esta dado de alta, la aplicación lo da de alta. Ya que el código incluido solo se ejecuta cuando el sensor está dado de alta, es decir ya existe en el servidor.

La siguiente imagen muestra los datos de temperatura enviados a la aplicación web. Como se puede apreciar nuestro sensor de temperatura está dañado y no marca bien la temperatura:

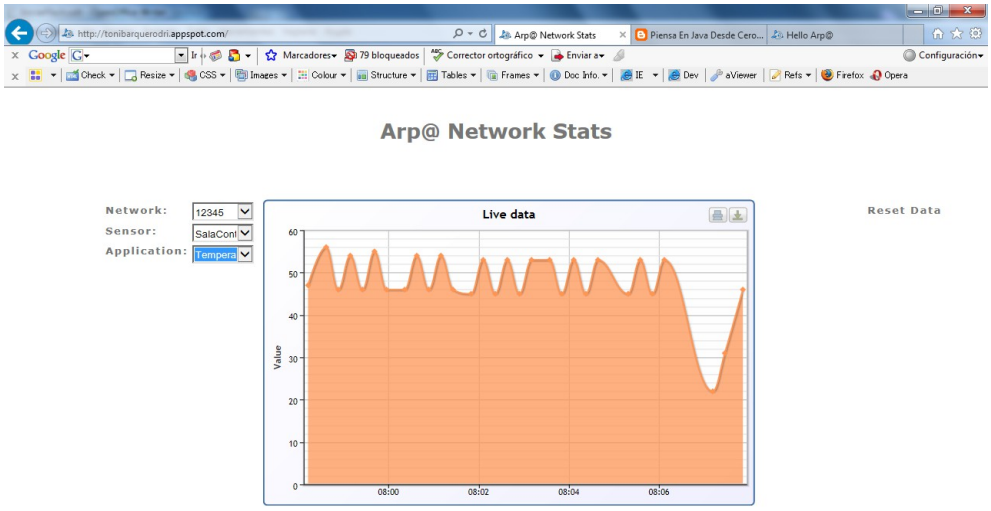



*Figura 18. Datos de temperatura.*

En esta otra figura nos muestra los minutos que quedan para el paro de la caldera que va de 15 a cero. Aquí se puede observar que cuando quedaban pocos minutos se ha realizado un reset, ya que ha vuelto a enviar los minutos que quedan cuando se ha cumplido el tiempo.

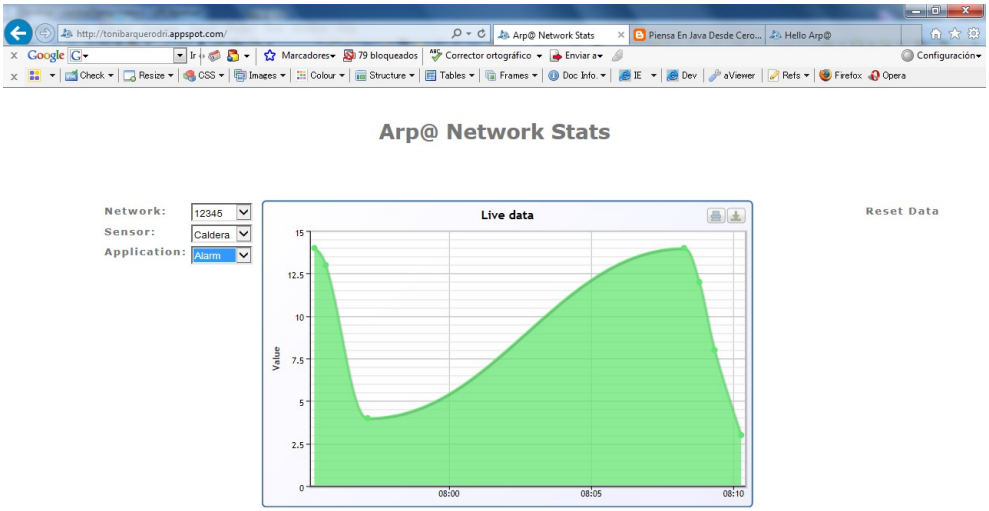



*Figura 19. Visualización minutos paro caldera.*

En esta otra figura podemos observar los mensajes que se envían al teléfono.

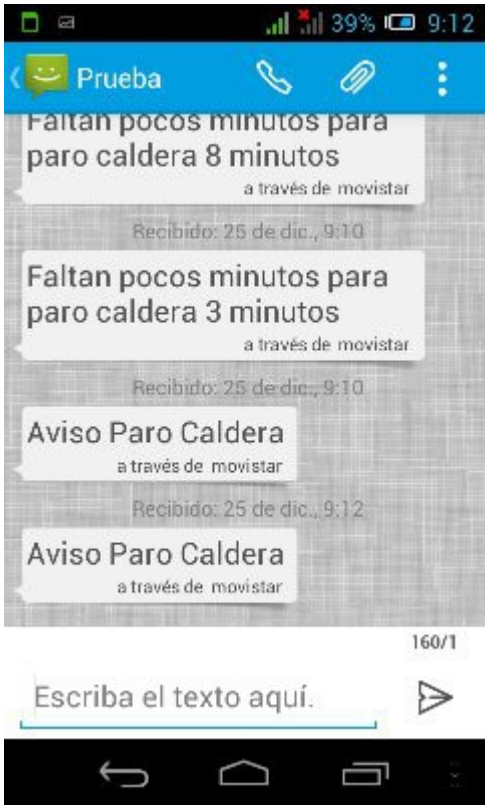



*Figura 20. Captura teléfono móvil.*

Por ultimo, la web que nos suministra el servicio de envío de mensajes de texto nos permite configurar los teléfonos y las palabras clave que se puede enviar, que en este caso he puesto "Prueba". También los mensajes enviados y si se ha podido enviar o no como se puede apreciar en la siguiente figura:

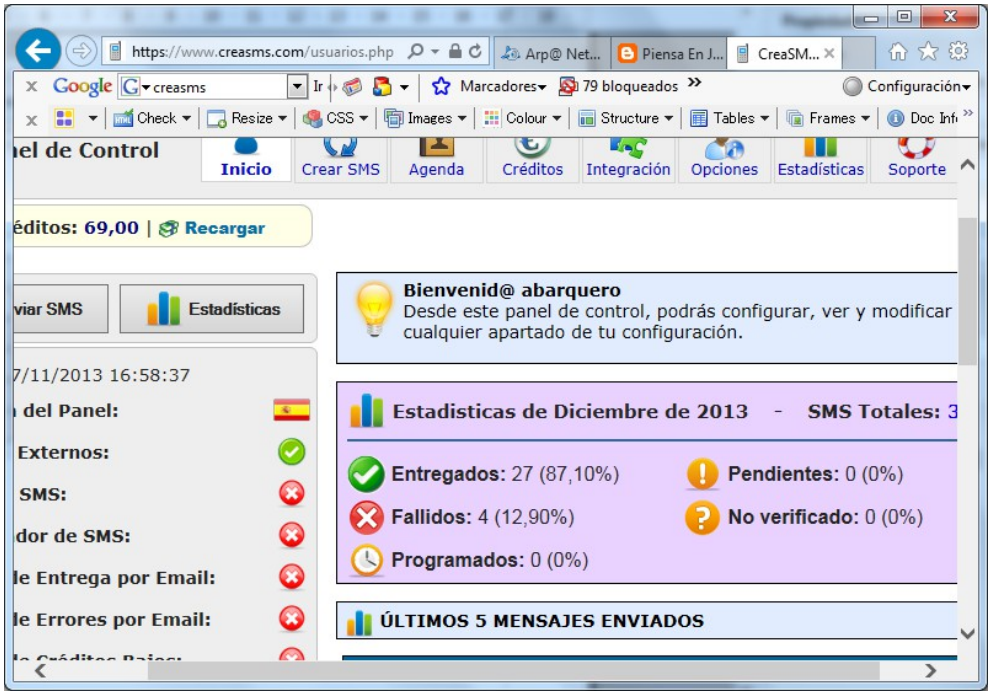



*Figura 21. Mensajes enviados.*

### 4.2.3 Anolog to Digital Converter(ADC)

Un conversor o convertidor de señal analógica a digital es un dispositivo electrónico capaz de convertir una señal analógica de voltaje en una señal digital con un valor binario. La señal analógica que varia de forma continua en el tiempo, se conecta a la entrada del dispositivo y se somete a un muestreo a una velocidad fija, obteniendo así una señal digital la salida del mismo.

Poseen dos señales de entrada llamadas Vref+ y Vref- y determinan el rango en el cual se convertirá una señal de entrada. El dispositivo establece entre su entrada (señal analógica) y su salida (digital) dependiendo de su resolución.

En el LPC1769 el calculo de este valor utiliza los siguientes valores:

- La tasa de conversión es 12 bits: a 200Khz
- El rango de medida es VREFN a VREFP
- la entrada puede ser de 3,3 V o 5 V.

Así tenemos lo siguiente:

- Resolución = 3,3 V / 2^12 → 3.300 / 4096
- 1LSB = VREFN – VREFP / 4096 = 0.8

Esto nos indica que por cada 0.8 mili-voltios que aumente el nivel de tensión entre las entradas al conversor, está aumentará en una unidad su salida, sumando en forma binaria.

El LPC1769 dispone de 8 puertos o pines que están distribuidos de la siguiente manera:

- AD0.0 – AD0.3 entre los puertos P0.23 – P0.26
- AD0.4 – AD0.5 entre los puertos P1.30 y P1.31
- AD0.6 y AD0.7 entre los puertos P0.3 y P0.2

Por otra parte, si queremos saber la función que desempeñan los distintos registros, como PINMODEx ( controla el modo de entrada de todos los puertos), PINSELx, ADDRx, etc., podemos consultar el "user.manual.lpc17xx".

Librerías utilizadas:

| | |
|---|---|
| **ADCInit** (uint32_t ADC_CLOK) | Inicializa los diferentes canales, habilita el reloj y su frecuencia. |
| **ADCRead**(uint8_T channelNum) | Lee la entrada y realiza la conversión devolviendo el resultado obtenido. |

### 4.2.4 General Purpose Input/Output (GPIO)

Estos registros especiales nos permiten configurar los periféricos para el uso que nos interese. Los GPIOs se pueden acceder por el uso general del controlador DMA. Solo los puertos 0 y 2 se pueden utilizar para generar una interrupción.

Para poder acceder a estos registros, NXP nos provee de archivos y estructuras en las que se incluyen todos los registros de cada periférico. La estructura LPC_GPIO0 está formada por los siguientes registros:

- FIODIR: Permite establecer la dirección del pin 1 para salida, 0 para entrada
- FIOSET: Podemos poner un 1 lógico en un pin
- FIOCLEAR: Permite poner un 0 lógico en un pin
- FIOPIN: Podemos leer el estado de los pines
- FIOMASK: Establece una máscara para las acciones de pin, SET o CLEAR

Para el manejo de un led debemos configurar el GPIO como output. Al encenderlo se pone en su nivel alto 3.3V, y al apagar lo se pone en su nivel de GND.

Por otro lado, para el manejo del botón debemos configurar el GPIO como input. Debemos de informar al sistema del tipo de activación "pull-up" o "pull-down". Estas son resistencias que llevan ese nombre por la función que cumplen: sirven para asumir un valor por defecto de la señal recibida en una entrada del circuito cuando por ella no se detecta ningún valor concreto, ni ALTO ni BAJO.

La diferencia entre ellas está en su ubicación en el circuito: las resistencias "pull-up" se conectan directamente a la fuente de 3.3V y las "pull-down" directamente a tierra.

En nuestro proyecto hemos utilizado la opción pull-down. Cuando pulsamos el interruptor la

entrada del circuito estará conectada a una señal de entrada válida. Y cuando dejemos de pulsar, la entrada del circuito estará conectada a la resistencia "pull-down", la cual tira a tierra.

| Conexiones GPIO | | |
|---|---|---|
| **Puerto** | **Pin** | **Función** |
| PINSEL4 P2.5 | J6.47 | Enciende la luz roja del led. Indica paro de caldera, tiempo excedido. |
| PINSEL4 P2.2 | J6.44 | Enciende la luz azul del led tricolor. Indica pre-alarma faltan 15 minutos para el paro. |
| PINSEL4 P2.4 | J6.46 | Enciende led rojo. Indica alta o baja temperatura de la Sala de Control. |
| PINSEL1 P0.22 | J6.24 | Enciende led de color verde indicando funcionamiento del LPC1769. |
| PINSEL0 P0.11 | J6.41 | Conectado al botón en la forma pull-down ressitor. Al pulsar el botón se provoca una interrupción. |

Librerías utilizadas:

| | |
|---|---|
| **GPIOSetDirp**(ortNum, bitposi, dir) | Selecciona la dirección del pin.<br>portNum → es el número del puerto 0-4.<br>bitposi → es la posición del número del puerto y puede estar entre 0 – 30<br>dir → 0 para input y 1 para output |
| **GPIOSetValue**(portNum, bitposi, value) | value → determina si un IO está o no encendido, 0 para off y 1 para on. |
| **GPIOSetPull**(portNum, bitposi, dir) | dir → seleccionar la resistencia no Pull 0, Pull up 1 y Pull down 2 (conectar a Vcc). |
| **GPIOISetInterrupt**(portNum, bitPosi, dir) | Indica el puerto donde se habilita la interrupción. |
| **GPIOCheckInterrupts**(portNum, dir) | Para comprobar si hay algún levantamiento de la interrupción. |
| **GPIOClearInterrupt**() | Se ha de ejecutar cada vez que se realice una interrupción |

Cada vez que se produce una interrupción, al pulsar el botón, se ejecuta el código de **EINT3_IRQHandler()**. Esta función esta definida el el "cr_startup_plc17.c".

Las interrupciones son señales del hardware que provoca que se ejecute por separado un código adicional. Como por ejemplo, en nuestro caso para borrar los contadores, apagar el led e inicializar el contador.

### 4.2.5 UART

El LPC1769 dispone de tres puertos UART. Hemos utilizado la UART3, el RxD3 que es el puerto P0.1 y pin J6.10, y el TxD3 con el puerto P0.0 que corresponde con el pin J6.9.

Librerías utilizadas

| | |
|---|---|
| **Serial_begin**(PortNum, baudrate) | Inicializa el puerto, selecciona el pin, el reloj, la paridad, FIFO, etc.<br>PorNum → Es el puerto UART 0 , 1 o 3.<br>baudrate → Velocidad en baudios en nuestro caso 9600. |
| **Serial_printString**(poerNum, BufferPtr) | Envía string a la UART byte a byte.<br>BufferPtr → string a enviar. |
| **Serial_available**(portNum) | Devuelve los bytes que hay a la entrada en el RX FIFO. |
| **UART3_IRQHandler**() | Interrupción handler cada vez que se utiliza el puerto. |
| **Serial_Read**(porNum) | Lee los datos del puerto serie. |

### 4.2.6 Driver de comunicación con el WiflyRN171

Se han programado aquellos comandos que se han tenido que utilizar para las pruebas y el proyecto. Si se quiere se pueden ir añadiendo mas comandos. Las funciones mas utilizadas son las siguientes:

| | |
|---|---|
| **bool setCMD**() | Entra en modo comando. |
| **bool setJoinSSID**(ssid) | Asocia el Wifly con la red wifi disponible. |
| **bool setComRemote**(get) | Guarda el GET a enviar a la web. |
| **bool setOpen**(web, port) | Abre la conexión con la web y puerto especificado. |
| **bool setSave**() | Guarda los cambios realizados en el Wifly. |

# 5. Viabilidad técnica

El proyecto es viable y se podría conectar al control sobre la caldera, pero deberíamos conseguir una alimentación de 3,3 voltios. Por un lado, lo podemos utilizar como temporizador para controlar el tiempo de rearme, y el envío de notificaciones. Por otro lado, también podemos utilizarlo para recibir la señal del temporizador que incluye la caldera para el envío de SMS de aviso. Sabemos que consta de dos salidas (pre-alarma y alarma) y una entrada de reset (que pone a 120 el contador y va restando hasta llegar a cero).

El inconveniente que le veo es que el LPC1769 tendría de poder conectarse a una fuente de alimentación de 12 voltios. Y también, debería tener una entrada para conectarlo a un adaptador de corriente y poder alimentarlo sin tener que utilizar la conexión USB. Y dejar esta conexión USB para programar y cargar el programa. Además, debemos realizar la instalación de la conexión de Internet en la sala de la caldera.

La conexión en la Sala de Control no hay ningún problema ya que existe conexión a Internet y solo nos haría falta conectar un modulo wifi para que reciba señal nuestro modulo Wi-Fly.

# 6. Valoración económica

A continuación exponemos un coste aproximado y orientativo del proyecto. Debemos tener en cuenta que las horas de desarrollo son orientativas, ya que el aprendizaje no se debería contar y una vez cogida experiencia es mas fácil de programar y realizar el montaje.

| Descripción | Precio unitario € | Cantidad | Total € |
|---|---|---|---|
| LPC1769 LPCXpresso Board | 22,56 | 1 | 22,56 € |
| RN-XV WIFLY 802.11 b/g | 28,01 | 1 | 28,01 € |
| Adaptador DIP Xbee | 2,50 | 1 | 2,50 € |
| Sensor temperatura TMP36 | 2,30 | 1 | 2,30 € |
| Set de cables M/H | 2,78 | 1 | 2,78 € |
| Led tricolor | 2,30 | 2 | 4,60 € |
| Placa prototipo | 10,90 | 1 | 10,90 € |
| Creasms(servidor de SMS) | 10 | 1 | 10,00 € |
| Horas desarrollo y programación | 22,00 | 200 | 4.410,00 € |
| | | **Coste total** | **4.473,65 €** |

# 7. Conclusiones

## 7.1 Mejoras

Las posibles mejoras podrían ser las siguientes:

1. Se podría incluir una pantalla para poder visualizar el tiempo que queda para que el temporizador llegue a cero.

2. Construir una aplicación web en exclusiva para controlar y guardar los datos suministrados por el LPC1769, y de los notificaciones enviados.

3. Conectar un botón al pin de RESET del LPC1769 para realizar un reset cada mes para evitar el overflow, por ejemplo, o para cualquier otro motivo.

4. Indicar en las notificaciones de donde provienen las alarmas, si de la Caldera o la Sala de Control.

5. Enviar el número de móvil en el mensaje.

6. Crear aplicación nativa en el móvil que nos sirva para entrar parámetros de alerta, ver históricos, etc.

7. Mirar de controlar mas de una caldera.

## 7.2 Autoevaluación

Este proyecto me ha servido para aprender lo que son y como funcionan los sistemas embebidos. Que hasta la fecha no sabía que existían, y eso que estamos rodeados de ellos. Además, me ha servido para aprender lo que es la corriente alterna y la corriente continua, y todo lo relacionado con ella como son el voltaje y la potencia.

Para mi ha sido bastante gratificante los conocimientos adquiridos, pero por otro lado he de reconocer que me ha costado bastante conseguir aprender el funcionamiento de estos nuevos sistemas. Estoy contento por ver que se pueden aplicar a la vida real y que es muy utilizado en la industria, y queda mucho recorrido para estos sistemas.

También me ha ayudado a reencontrarme con el lenguaje C que lo tenía olvidado, y me he dado cuenta que es muy útil, y que ocupa poco espacio no como otros lenguajes como el java. Creía que el C estaba anticuado y que no se utilizaba, y este proyecto me ha demostrado todo lo contrario.

También me he dado cuenta que otros sistemas operativos como Linus, Knoppix o el Unbutu llevan

incorporados el C. Y hecho de menos no haber instalado uno de estos sistemas operativos y haberme familiarizado con ellos. Pienso que deberíamos perder el miedo que tenemos a cambiar de sistema operativo y pasarnos a uno de estos. Esto me fuese ayudado a realizar el proyecto con mas facilidad, ya que a lo largo de las pruebas tenía fallos en el IDE de LPCXpresso y era debido a que me fallaba el compilador gcc. Como ya he explicado he tenido que instalar una versión para Windows el MinGW.

En definitiva, valoro positivamente esta experiencia que me será muy útil tanto en la vida privada como en la industria, si me dejan trabajar sobre este tipo de dispositivos. Y como dice un proverbio:

"You'are never too old to learn"

# 8. Glosario

**Acuotubulares** Son calderas en las que el fluido de trabajo se desplaza por tubos durante su calentamiento. Son las más utilizadas en las centrales termoeléctricas, ya que permiten altas presiones a su salida y tienen gran capacidad de generación.

**ARM** Arquitectura RISC (Reduced Instruction Set Computer) de 32 bits desarrollado por ARM Holdings.

**Caldera industrial** Es una máquina o dispositivo de ingeniería diseñado para generar vapor. Este vapor se genera a través de una transferencia de calor a presión constante, en el cual el fluido, originalmente en estado líquido, se calienta y cambia su fase. Además, es un recipiente a presión, por lo cual es construida en parte con acero laminado a semejanza de muchos contenedores de gas. Existen dos tipos de caldera: **Acuotubulares** y **Pirotubulares.**

**GPIO** General Purpose Input/Output, es un pin genérico de entrada/salida de propósito general.

**FreeRTOS** Free Real Time Operating System, es un sistema operativo en tiempo real para dispositivos embebidos.

**HTML** HypertText Markup Language, es un lenguaje de marcado para el desarrollo de páginas web.

**Interrupción** Señal eléctrica asíncrona enviada por un periférico al procesador. Cada vez que se envía este tipo de señal, se interrumpe la ejecución normal de la tarea que este activa en el procesador en aquel momento. Cuando esto sucede, se deja en contexto actual (puntero de instrucciones, estados del registro, etc.) y se ejecuta una rutina del servicio de interrupción. Cuando finaliza la rutina de servicio de interrupción, el procesador continua con la ejecución normal de la tarea que había quedado interrumpida.

**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. Es equivalente al termino *gestor de interrupciones.*

**LPC1769** Micro-controlador Cortex-M3 que puede ser usado para aplicaciones en dispositivos empotrados con un alto nivel de integración y un consumo de energía bajo.

**Periférico** Elemento de maquinari (excluyendo el procesador) que funciona de entrada y salida de información.

**Pila** Se trata de una lista abierta que permite la adición y eliminación de elementos de manera que el ultimo en entrar siempre es el primero accesible para salir. En sistemas informáticos, estas estructuras de datos permiten guardar información que se ha de recorrer de manera secuencial. Este funcionamiento particular la hace especialmente adecuada para hacer el seguimiento de tareas muy jerarquizadas.

**Pirotubulares** Es un tipo de caldera en que el fluido en estado líquido se encuentra en un recipiente atravesado por tubos, por los cuales circulan gases a alta temperatura, producto de un proceso de combustión. El agua se evapora al contacto con los tubos calientes productos a la circulación de los gases de escape.

**Sistema operativo en tiempo real** Sistema operativo diseñado especialmente para sistemas en tiempo real. Se utilizan en los sistemas que hay que cumplir unas especificaciones temporales determinadas.
Sigla **RTOS**

**Scheduller o planificador** Elemento de programario integrado en el núcleo del sistema operativo responsable de decidir que tarea ha de utilizar el procesador en un momento determinado.

**UART** Universal Asynchonous Receiver/Transmitter, es un chip que se encarga de manejar las interrupciones de los dispositivos conectados al puerto serie, convertidos a datos en formato paralelo y transmitirlos al bus del sistema.

**Tarea** Cualquier calculo individual, conjunto de cálculos, la lógica de la toma de decisiones, intercambio de información o combinación de ellas que ha de llevar a termino en tiempo de ejecución el programario. Es la abstracción de los RTOS. Las tareas comparten un espacio de memoria común.

**Temporizador** Periférico que cuenta hechos externos o ciclos del procesador.

**Wi-Fi** Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.

**WiFlyRN171** Componente que capacita a sistemas empotrados a conectarse a redes Wireless compatibles con 802.11b/g.

**Wireless** Transferencia de datos donde no se utilizan cables para conectar diferentes ordenadores.

# 9. Bibliografía

**NXP**. *LPCWARE (Software & Support for NXP MCUs).* Submitted by lpcxpresso-suppport on Mon, 2013-10-21. Disponible en : http://www.lpcware.com/lpcxpresso

**BRC.Electronics.** *SimpleCortex.* Joomia 1.7 templates – Joomia template marker, 2011. Disponible en: http://www.brc-electronics.nl/

**Code_red.** *Innovative tools for rapid 32-bit embedbed microcontroller development.* CrSupportAb, 2013-02-21. Disponible en: http://support.code-red-tech.com/

**UOC.** *IniciCortexM3.* Administrator, XWiki.org, 2013-09-07. Disponible en: http://cv.uoc.edu/webapps/xwiki/wiki/matembeddedsystemslabhome/view/Material/IniciCortexM3

**Fundación Wikipedia.** *Wikipedia, la enciclopedia libre.* GNU Free Documentation License, 2013. Disponible en: http://es.wikipedia.org/

**CreaSMS.** *Envia tus SMS a cualquier país.* HGM Network S.L., 2013. Disponible en: https://www.creasms.com/

**BricoGeek.** *Tienda!.* Tienda BricoGeeek.com, 2005-2013. Disponible en: http://www.bricogeek.com/shop/

**Real Time Engineers Ltd.** *FreeRTOS.* Richard Barry, 2010-2013. Disponible en: http://www.freertos.org/

**Embedded Artists.** *The art of Embebedded Systems Development.* Embedded Artists AB, 2013. Disponible en: http://www.embeddedartists.com/products/lpcxpresso/lpc1769_xpr.php

**Sparkfun.** SparfFun Electronics, 2013. Disponible en: https://www.sparkfun.com/products/11049

**Teunis van Beelen.** *RS-232 for Linux and Windows.* Teunis van Beelen, electronics engineer, 2010. Disponible en: http://www.teuniz.net/RS-232/index.html

**MinGW.** *Minimalist GNU for Windows.* MinGW.org, 2013. Disponible en: http://www.mingw.org/

**NXP Semiconductors.** *LPC1769/68/67/66/65/64/63.* NXP B.V. 2013. Disponible en: http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf

**Roving networks.** *User Manual and Command Reference WiFly GSX/EZX.* WiFly-RN-UM 11/9/2011. Disponible en: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/WiFi/WiFly-RN-UM.pdf

**Analog devices.** *Low Voltage Temperature Sensonrs (TMP35/TMP36/TMP37).* 1996 – 2010. Disponible en: www.analog.com

**Freescale semiconductor.** *Micromachined Accelerometer (MMA7361L).* 2008. Disponible en: www.freescale.com

**China young sun led tecchnology co., LTD.** *Red/green/blue triple color led white diffused lens.* 2009. Disponible en: www.100LED.com

**China young sun led tecchnology co., LTD.** *Red/green/blue triple color LED.* 2009. Disponible en: www.100LED.com

# 10. Anexos

## 10.1 Clase AddStats.java de Arp@Lab

A esta clase se ha añadido dos funciones:

- **private String senSMS(Stringvalue):** Recibe el String con en el mensaje a enviar, y abre la conexión con el servidor de SMS y envía la petición.
- **private String readStream(InputStream in):** Este metodo recibe la respuesta del servidor de SMS y devuelve la respuesta.

```
.....
            SensorStats se=net.getSensorStats(ip);
                              Date date=new Date();

                              //rssi
                            TupleDateValue tup= new TupleDateValue(date, value);
                              List<TupleDateValue> li=new
ArrayList<TupleDateValue>();
                              li.add(tup);
                              se.addValues(app, li);
                              if(app.equals("Alarm")){

      response=sendSMS("Faltan+pocos+minutos+para+paro+caldera+"+minutos+"+minutos");
//añadido

                              }else if(app.equals("OffCaldera")){
                                    response=sendSMS("Aviso+Paro+Caldera");
//añadido
                              }else if(app.equals("AlarmTemp")){

      response=sendSMS("Temperatura+fuera+de+limites");
                              }
                              text=response +":Sensor Stats updated IDH"+
se.getSensorId();
                        }
                        return text;
      …..........................



                  private String  sendSMS(String value){
                        String body = " ";
                        try{

                              //Construir datos
                              String sms = "http://api.creasms.com/web/enviar.php?
usuario=33571&encripta=1&clave=dG9uaXRvNjM=&pais=34&movil=676816231&remitente=Prueba&me
nsaje=" + value;
```

```
                                //Enviar datos
                                URL url = new URL(sms);
                                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();

                                //Obtener respuesta
                                body = readStream(urlConnection.getInputStream());
                                urlConnection.disconnect();

                        } catch (Exception e){
                                return "ERROR SMS " + e;
                        }
                        return body;
                }
                /**
                 * El método para recibir el streaming de la respuesta del servidor
(o response)
                 * @param in
                 * @return string
                 * @throws IOException
                 */
                        private String readStream(InputStream in) throws
IOException{
                                BufferedReader r = null;
                                r = new BufferedReader(new InputStreamReader(in));
                                StringBuilder total = new StringBuilder();
                                String line;
                                while ((line = r.readLine()) != null) {
                                        total.append(line);
                                        } if(r != null){
                                                r.close();
                                                } in.close();
                                                return total.toString();
                                                }
```

## 10.2 Guía del usuario

### 10.2.1 Descargar, construir y debug

Primero debemos tener instalado el LPCXpresso, el cual se puede descargar desde la url http://www.lpcware.com/lpcxpresso/download. Una vez instalado y creado nuestro espacio de trabajo iniciamos el programa.

Primero , seleccionamos "import projects" desde la Quickstart panel:

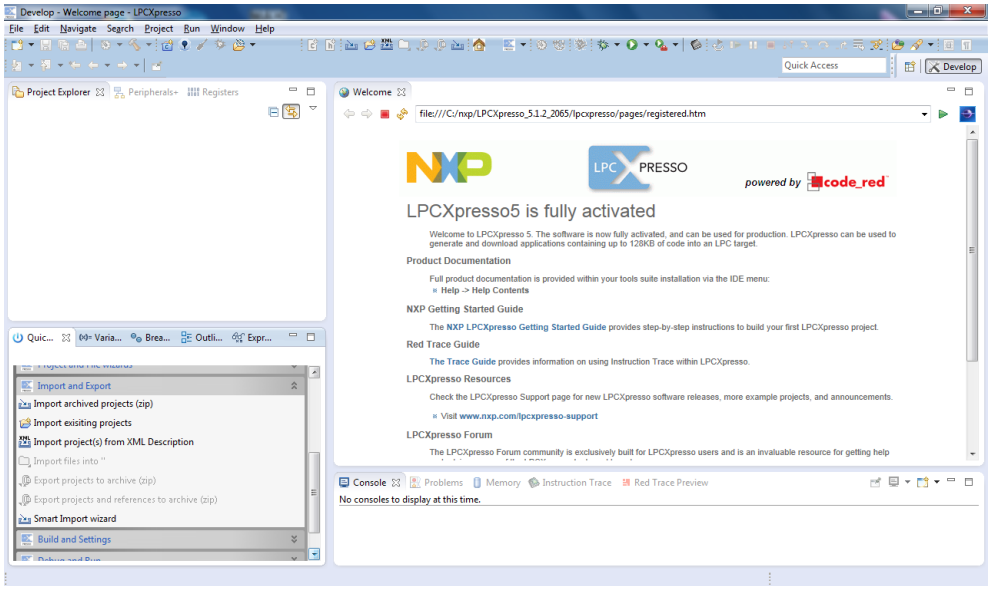



*Figura 22. Quickstart panel*

Buscamos el archivo zip que contiene los programas:

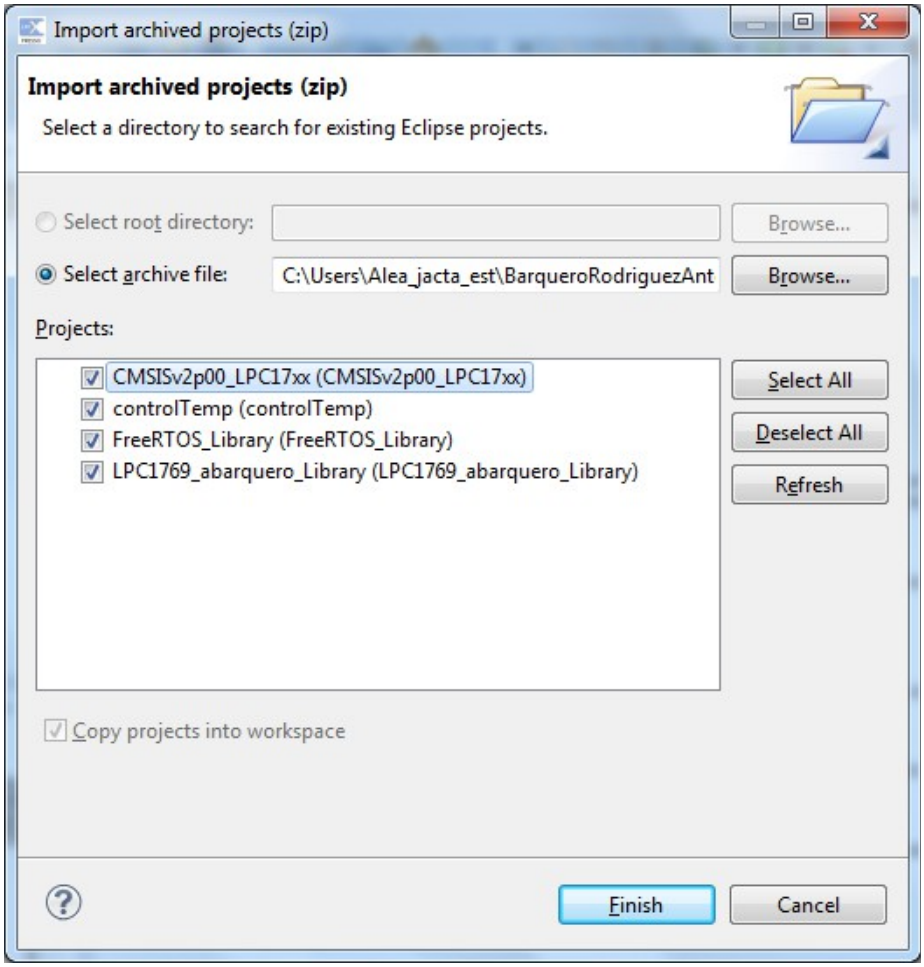



*Figura 23. Import archived projects (zip)*

Ahora compilaremos el código seleccionando "Build controlTemperatue":

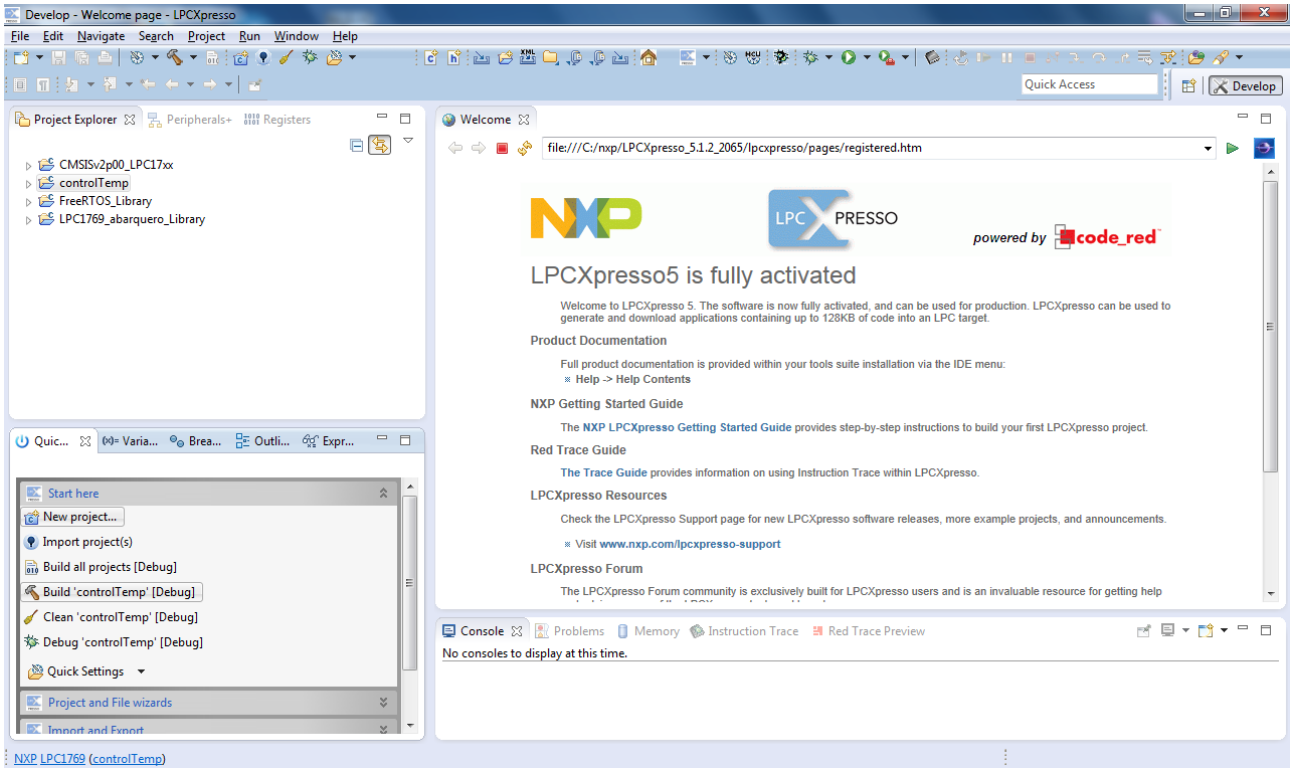



*Figura 24. Build controlTemperature*

Una vez compilado y solucionado los errores que puedan surgir, cargaremos el código en le memoria flash. Seleccionamos el icono "Program Flash":

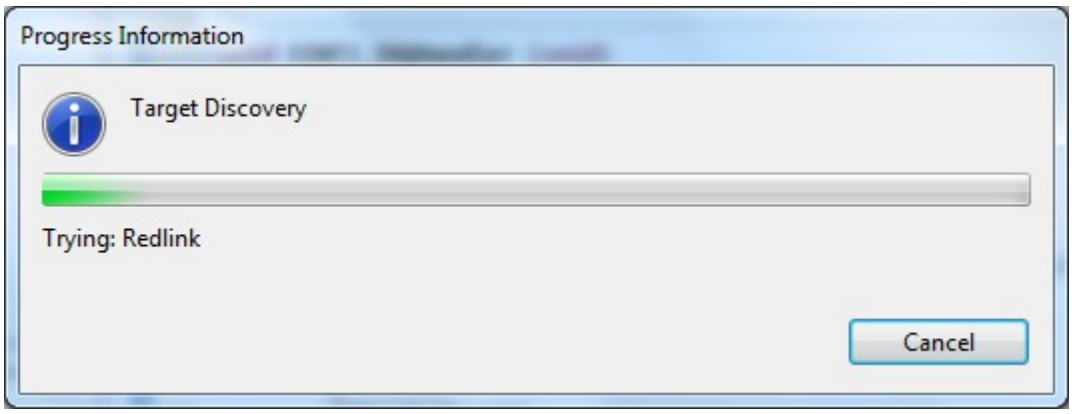



*Figura 25. Buscando tarjeta LPC1769*

Seleccionamos borrar el contenido de la memoria para poder luego cargar el código:

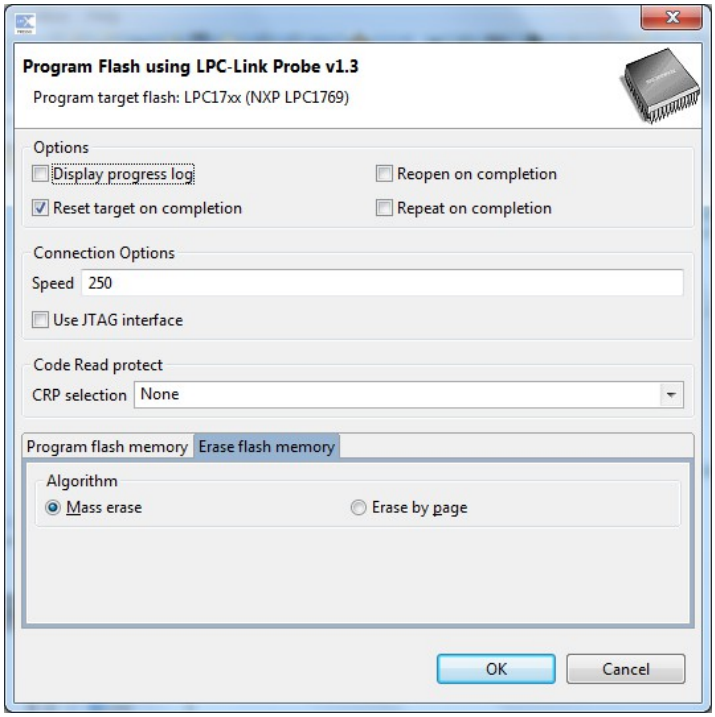



*Figura 26. Erase Program Flash*

Una vez borrado volvemos a seleccionar el icono de la memoria flash, y seleccionamos "Program flash memory" y pulsamos el botón OK. Y nuestro código se carga en la memoria flash.

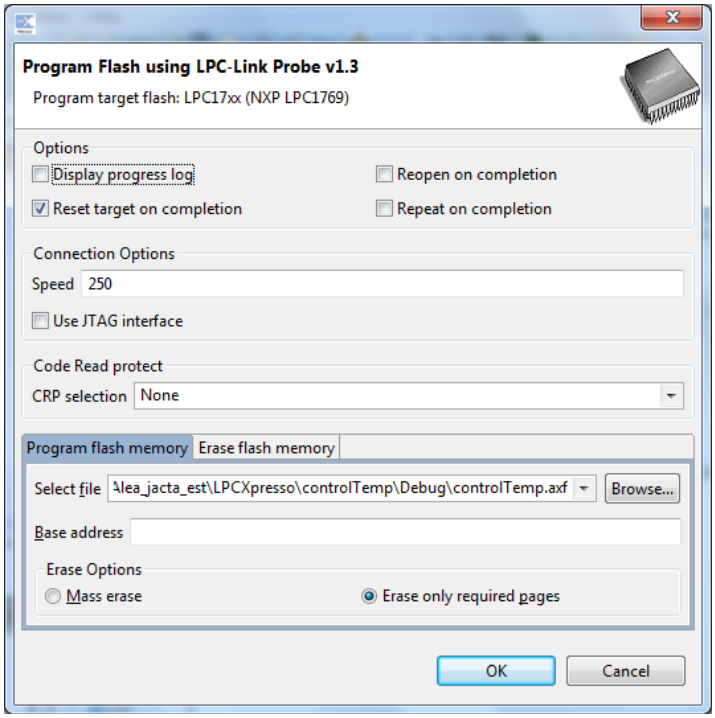



*Figura 27. Program flash memory*

Finalmente nuestro programa arrancará automáticamente con solo suministrar corriente al LPC1769.

### 10.2.2 Crear librería estática

Seleccionamos "New Project" → "LPCXpresso C Project"

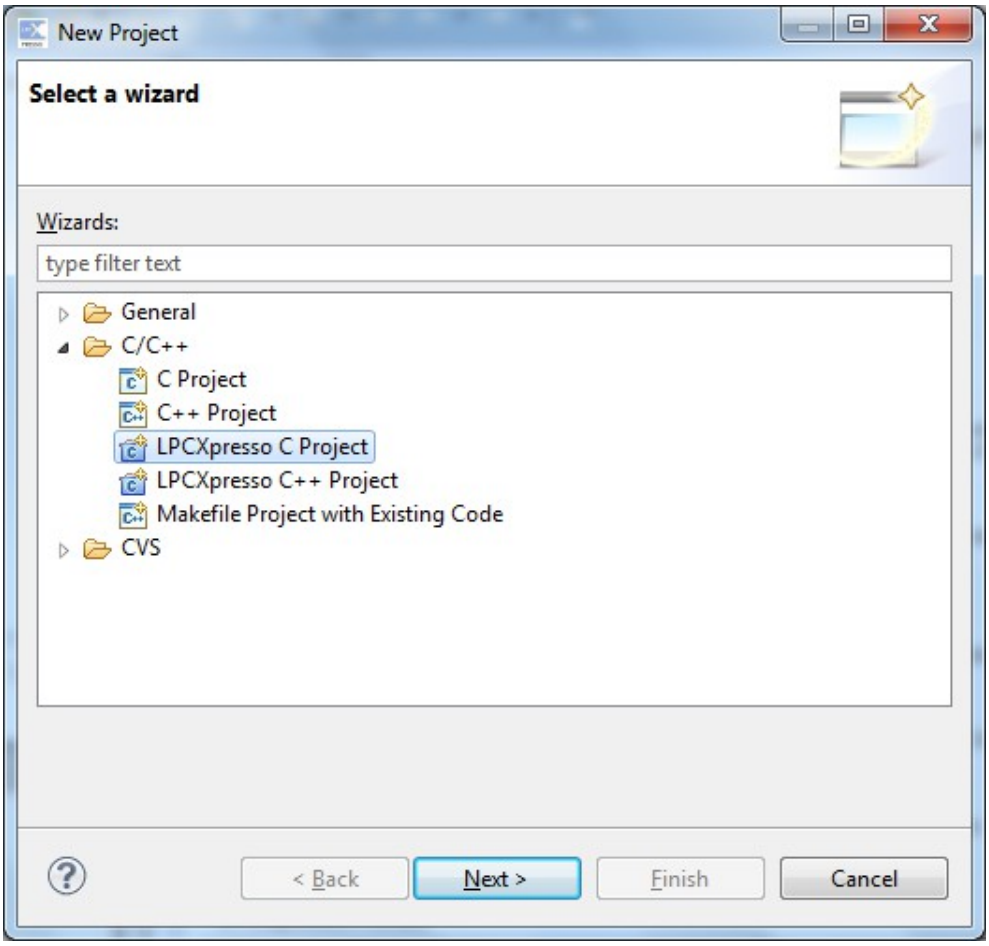



*Figura 28. LPCXpresso C Project*

Después seleccionamos la opción "C Static Library Project":

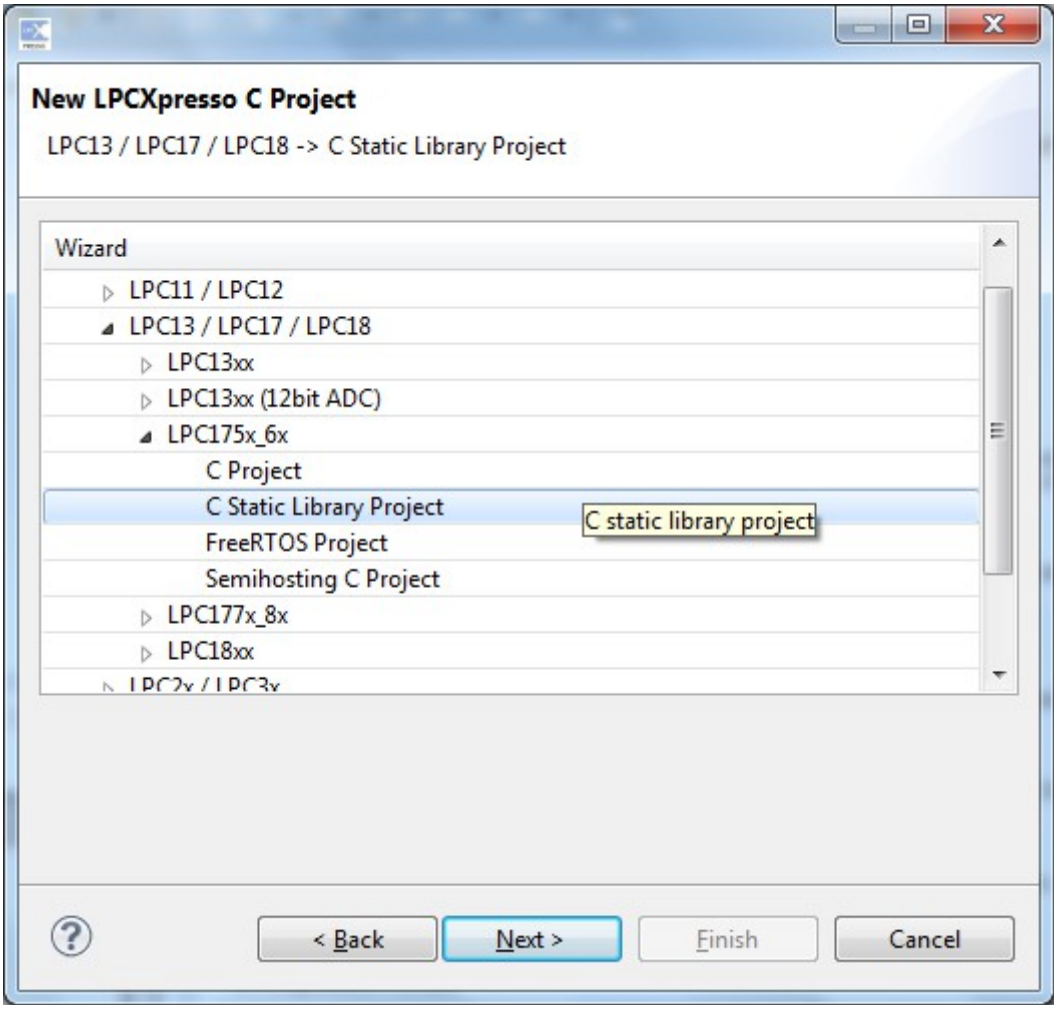



*Figura 29. New LPCXpresso C Project*

En la siguiente ventana introducimos el nombre que queramos darle a nuestro proyecto:

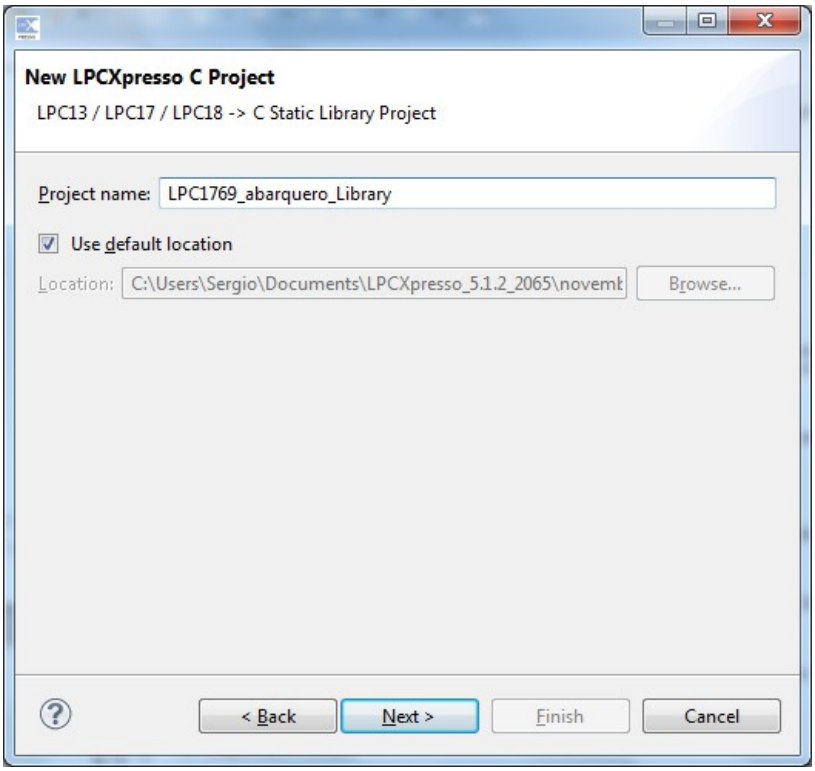



*Figura 30. Project name*

Ahora seleccionamos nuestra target de la lista.

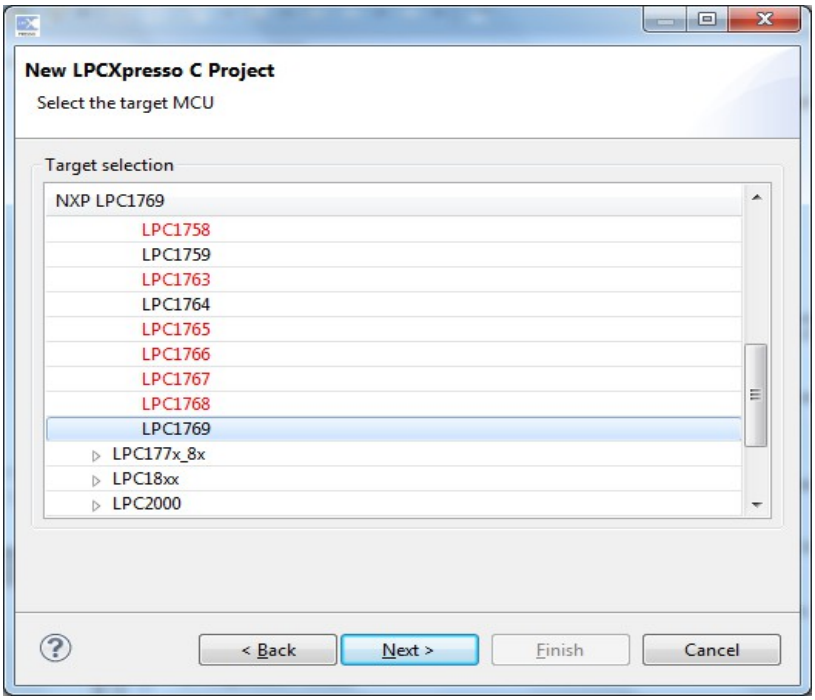



*Figura 31. Select the target MCU*

Y finamente seleccionamos la librería para este proyecto, en este caso CMSISv2p00_LPC17xx:

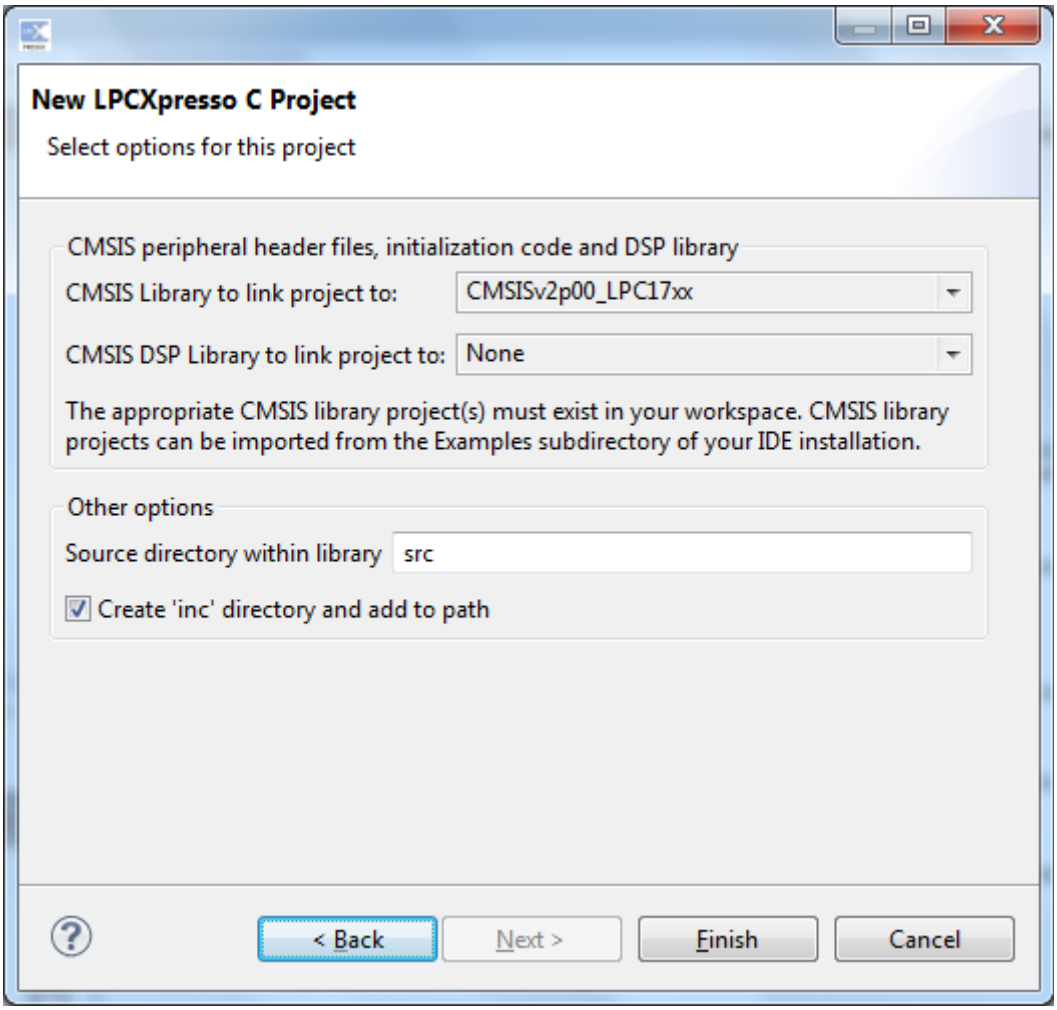



*Figura 32. Select options for C project*

### 10.2.3 Control temperatura y rearme caldera

Una vez instalada la aplicación en el LPC1769 debemos alimentar de corriente el dispositivo, mediante un cable USB conectado al ordenador. También podíamos conectarlo a una batería que nos suministrara una corriente continua de 3,3 voltios. Una vez hecho esto el dispositivo comenzaría a funcionar.

Para acceder al servidor web tendremos de teclear la siguiente dirección en nuestro navegador:

http://tonibarquerodri.appspot.com/

Y nos encontraríamos con la siguiente interfaz.

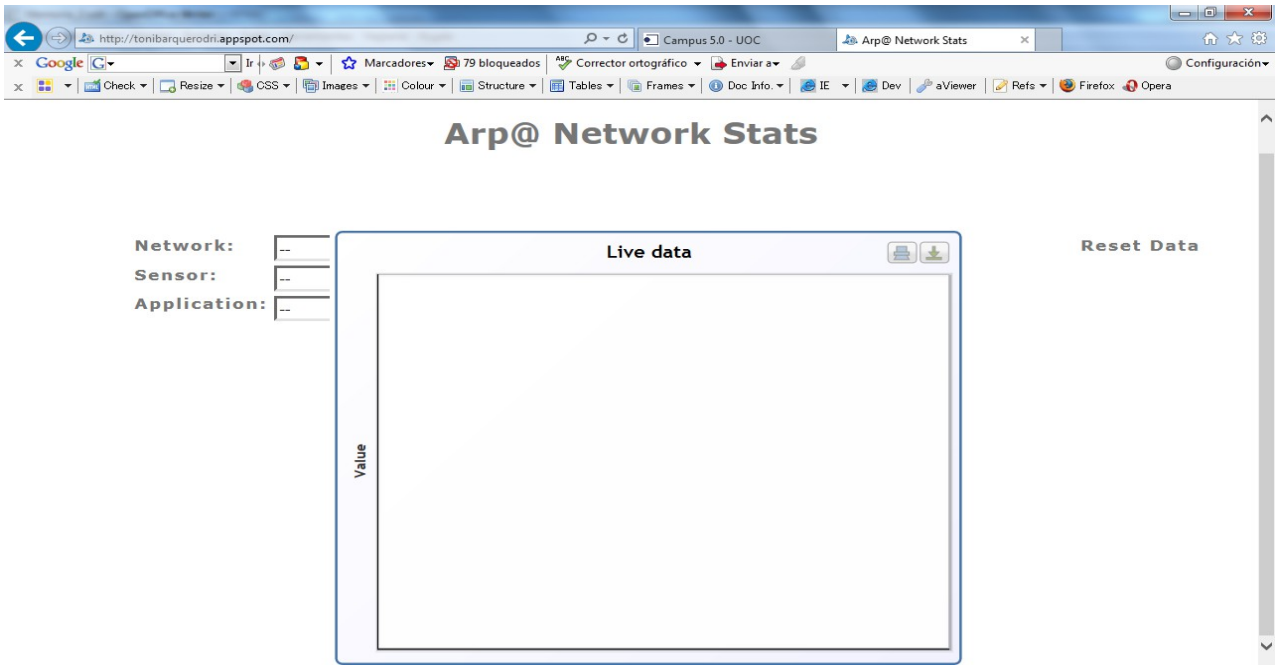



*Figura 33. Cliente web*

Desde aquí seleccionamos la red (Network) que queremos consultar, el sensor (Sensor) que sería:

- **SalaControl:** que consta de dos aplicaciones, *AlarmTemperatura*, que nos muestra la temperatura cuando sobrepasa los 30 grados, y *Temperatur*a que muestra la temperatura de la sala dentro de los parámetros normales de funcionamiento.
- **Caldera:** que también dispone de dos aplicaciones, *Alarm* que nos muestra el tiempo que queda para el paro de la caldera que va de 15 a cero; y *OffCaldera* que nos muestra cuando se ha parado la caldera.

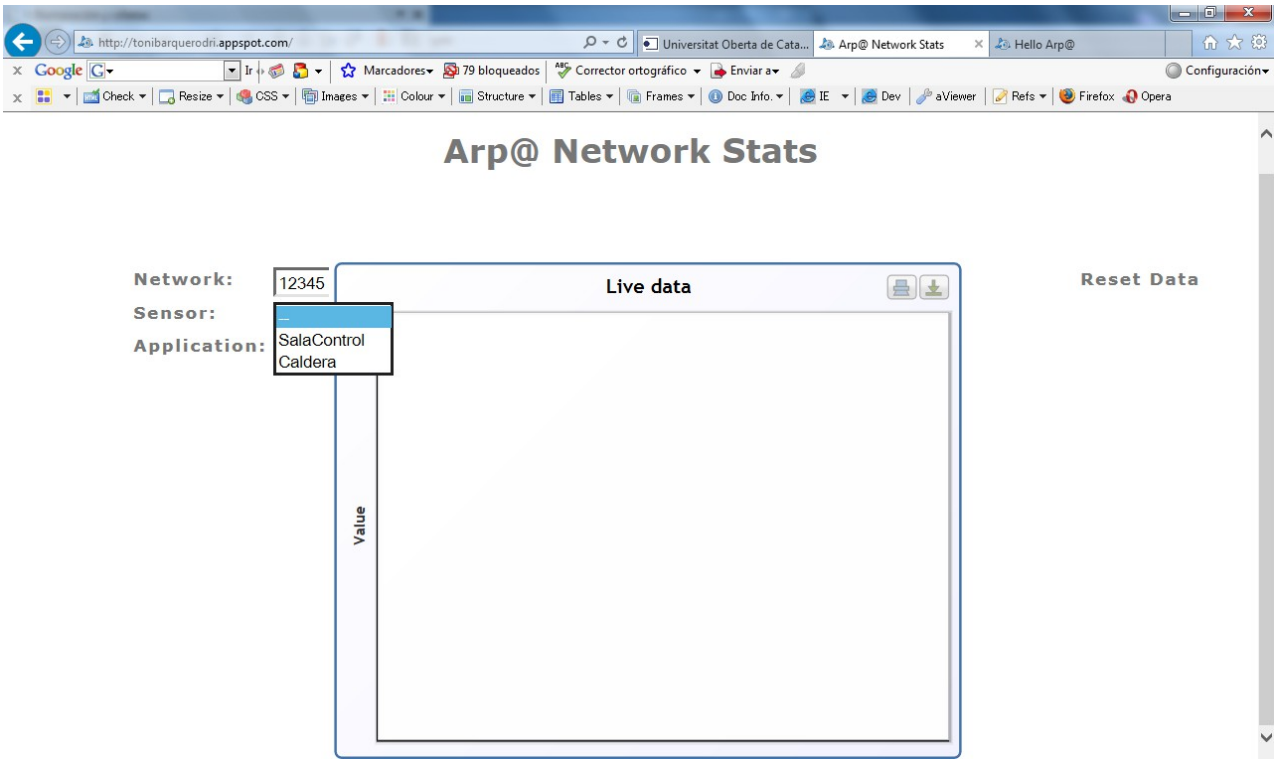



*Figura 34. Selección de sensor*

Desde la aplicación web y según los datos recibidos se nos envían las notificaciones de alarma a nuestro dispositivo móvil.

Por ultimo, cuando veamos en nuestro sistema embebido que se ha encendido el led de color azul nos indica que faltan 15 minutos para el paro de la caldera. Por lo que deberíamos pulsar el botón para que se apague el led y vuelva a contar desde cero. Del color azul pasa al rojo, esto nos indica que se ha mandado una señal de paro, por lo que deberíamos poner en marcha la caldera y realizar un reset de las alarmas pulsando el botón.

Y para el sensor de temperatura solo se encenderá el color rojo si se sobrepasa la temperatura de referencia, que en nuestro caso es de 30, pero que podríamos poner también una temperatura mínima. Y deberíamos comprobar si existe alta temperatura en la Sala de Control, y sino es el caso deberíamos sustituir el sensor.