Medidor online de temperatura y humedad de bajo consumo

Estudiante: Jesús Santiago Fernández Prieto
"Ingeniería Técnica de Informática de Sistemas"

Consultor: Jordi Bécares Ferrés

11 de Junio de 2013

When I read commentary about suggestions for where C should go, I often think back and give thanks that it wasn't developed under the advice of a worldwide crowd.
(D. Ritchie)

A mi familia, novia y amigos.

Resumen

Para este proyecto se ha diseñado un dispositivo con capacidad de conexión a Internet a través de un punto de acceso WIFI para el envío de datos y alertar en caso de posibles temperaturas y humedades críticas, definidas con un valor mínimo y un máximo. El diseño de este dispositivo se ha hecho teniendo en cuenta el reducir el consumo de energía para maximizar autonomía.

El dispositivo está basado en una mota LPC1769 que posee un procesador Cortex-M3 de ARM el cual ejecuta nuestro programa desarrollado. Este programa hace uso del sistema operativo FreeRTOS (Free Real Time Operating System) que facilitará y nos asegurará estabilidad.

La mota llevará conectada un chip WiFly que se usará para poder conectarnos por WiFi al punto de acceso.

Cuenta además con un sensor SHT15 que permite, a la mota, tomar las mediciones de temperatura y humedad relativa. Estos serán posteriormente enviados al servidor. En caso de que en alguno de esos valores esté fuera del rango de seguridad se avisará al usuario a través de dos leds conectados también a la mota. Se le puede configurar una dirección de correo electrónico donde se enviarán alertas para enviar un correo al usuario advirtiéndole de un posible problema. De este modo se permite la supervisión remota.

Para recopilar los datos y mostrarlos interactivamente se dispone de un servicio web en el cual se recibe los datos y se presentan, al usuario, gráficas con la evolución de los valores de temperatura y humedad relativa en el tiempo. Este servicio web puede supervisar varios dispositivos diferentes al mismo tiempo, e incluso diferentes redes. También desde el servicio web se puede configurar el intervalo de toma de datos (en minutos) y los límites de seguridad de temperatura y humedad, así como la dirección de email a la que se enviarán los avisos de alarma.

Palabras clave: LPC1769, mota, WiFly, SHT15, SHT1x, empotrados, ARM Cortex M3, C, FreeRTOS, JTAG, Google App Engine, Java

Índice

Tabla de Figuras

1. Introducción

Durante este Trabajo Fin de Carrera se han estudiado como desarrollar para dispositivos empotrados, para crear desde sensores hasta robots con los conocimientos adquiridos.

Hemos estudiado como se relacionan un microcontrolador con uno o varios dispositivos vía UART y cómo crear un driver complejo para facilitar el acceso.

Posteriormente se ha investigado el uso de los puertos GPIO (General Purpose Input/Output) para alimentar una patilla de la mota e iluminar los leds. También se ha usado GPIO para transmitir datos a un sensor que soporta el protocolo TWI (Two-Wire), teniendo que implementar una señal de CLOCK, para marcar el ritmo del flujo de datos, y otra de DATA, para transmitir y recibir datos.

Se ha usado un sistema operativo en tiempo real (FreeRTOS) para facilitar el desarrollo de tareas en paralelo y manejar sus esperas. También como activar un modo sleep cuando el microcontrolador se encuentra esperandopara cumplir el requisito de nuestro proyecto de reducir el consumo en lo posible.

Tanto el envío como la recepción de datos de Internet ha sido clave para el correcto funcionamiento del proyecto debido a que los datos deben guardarse en el servidor el cuál es usado también para leer la configuración actual de los sensores monitorizados.

Derivado del punto anterior ha sido necesario hacer modificaciones al proyecto Arp@Lab para adecuarlo a las características del proyecto. Se han añadido nuevas funcionalidades que lo han adecuado para el uso del prototipo que nos ocupa. A continuación veremos una imagen del prototipo creado.

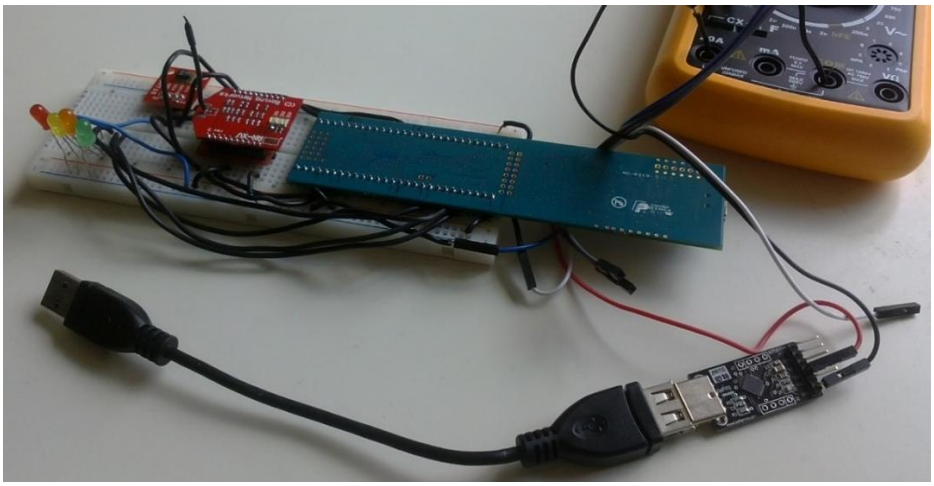


Figura 1 Medidor online de temperatura y humedad de bajo consumo

1.1 Justificación

Las condiciones climatológicas siempre fueron dadas a ser recopiladas y almacenadas ya sea por motivos estadísticos o para monitorizar el correcto estado de una sala o laboratorio, por poner sólo dos ejemplos.

Debido a esto sería muy interesante crear un dispositivo independiente que sea capaz de satisfacernos esas necesidades, ósea que sea capaz de conocer la temperatura y humedad de un lugar y hacernos lo saber sin necesidad de estar a su lado.

Aprovechando la flexibilidad que nos aporta Internet podríamos incluso no limitarnos a la monitorización de un solo lugar, sino que podríamos tener una red que se dedicase a monitorizar diferentes lugares, los cuales no tienen que ser necesariamente cercanos, y enviando los datos climatológicos actuales a un servidor común de Internet para que los usuarios puedan comprobarlos remotamente.

El usuario podría querer además configurar los parámetros de estos dispositivos, como intervalo de toma de datos, fijar unos valores máximos y mínimos o configurar una cuenta de correo donde se avise de que han superado ciertos valores “seguros”.

Con este dispositivo se nos permitiría:

- Tener almacenadas mediciones de temperatura/humedad dentro de una base de datos para poder realizar una estimación futura.
- Monitorizar si se ha encendido nuestra calefacción en nuestra casa.
- Alertar de posibles incendios, o valores críticos de humedad.
- Centralizar el control de temperatura y humedad de varias habitaciones o laboratorios en un único punto y configurar avisos si se produce algún problema.
- Tomar el servicio web e implementar una solución que enviase a nuestra cuenta de twitter la temperatura de nuestra ciudad.

Las propuestas anteriores son sólo unos poco ejemplos, se nos podrían ocurrir infinidad más.

La elección de crear un dispositivo empotrado para desarrollar estas tareas fue clara. Un dispositivo empotrado nos permite la movilidad y flexibilidad que necesitamos. Además de garantizar un bajo consumo y una mayor autonomía. Gracias a una batería podría garantizar la toma de datos durante bastante tiempo.

1.2 Descripción del proyecto

Este medidor de temperatura y humedad es un pequeño sistema empotrado que permite la toma de datos de temperatura y humedad de un modo periódico y enviarlos a un servidor de Internet.

Los requisitos básicos son que partiendo del hardware que se ha elegido se deberá programar la mota LPC1769 para realizar las tareas necesarias de toma de datos, activación de alarma en caso necesario, envío de datos a Internet y optimizarlo para que consuma la menor batería posible cuando no tenga que hacer operaciones.

1.3 Objetivos del TFC

El objetivo de este proyecto es obtener un sistema que permita aplicar y asentar los conocimientos adquiridos durante las asignaturas cursadas en la ingeniería.

Dentro de los objetivos que debe de satisfacer el sistema tenemos:

- Tomar datos de temperatura y humedad
- Repetir la medición periódicamente
- Activar leds para identificar que medición ha dado un valor crítico
- Enviar las mediciones de datos al servidor
- Ser capaz de configurarse dinámicamente a través del servidor
- Valores máximos y mínimos de temperatura
- Valores máximos y mínimos de humedad
- Intervalo de toma de datos
- Almacenamiento y muestra de los datos de temperatura y humedad recogidos
- Capacidad de usar múltiples dispositivos
- En caso de que el servidor reciba un valor fuera del rango de seguridad deberá avisar vía correo electrónico
- Tener un consumo bajo cuando no se estén realizando acciones

Resumiendo, deberemos de crear un dispositivo autónomo, de bajo consumo, capaz de comunicarse con un servidor para el envío de las mediciones que realice o actualizar su configuración en base a las decisiones de un usuario.

1.4 Enfoque y Método seguido

Para la consecución del proyecto se ha seguido un desarrollo interactivo y creciente basado en ciclo de vida en cascada (o clásico). Las etapas se han diferenciado y separado en Análisis, Diseño, Codificación y Pruebas.

El proyecto ha pasado por tres fases importantes.

1.4.1 Primera fase

Consistió en una fase de toma de contacto con el entorno y el hardware inicial provisto antes del curso. Esta consistió básicamente en los siguientes puntos:

- Instalación del IDE y familiarización con las opciones.
- Usar el WiFly conectándolo por UART al PC para conectar a Internet y probar algunas de las opciones que se implementarían posteriormente el driver WiFly para conectar la mota a Internet.
- Implementación del driver UART que permite enviar datos desde la mota al WiFly o a un ordenador conectado por USB.
- Implementación del driver WiFly para manejar el WiFly desde el LPC1769.
- Implementación de un programa productor-consumidor utilizando las herramientas que nos provee FreeRTOS.

1.4.2 Segunda fase

Después de la anterior fase de formación y toma de contacto con los elementos del proyecto continuamos con la segunda fase del proyecto. Durante esta fase se realizó la labor de estudio y implementación del proyecto.

Esta fase se dividió del siguiente modo:

- Estabilización del sistema
    - Ensamblado de componentes
        - Realizar las conexiones: Usando una breadboard se realizó las conexiones necesarias para dar más estabilidad al sistema.
        - Probar que funciona correctamente: Asegurarse de que todo sigue funcionando correctamente.
    - Mejorar drivers
        - Separar drivers WiFly y UART: Separar la funcionalidad de ambos drivers para dar coherencia al diseño.
        - Usar esperas pasivas: Usar esperas pasivas de las tareas para esperar un tiempo determinado y que el microcontrolador pueda procesar otras tareas.
        - Usar semáforos para acceder al UART: Controlar el acceso a los diferentes puertos UART para que no haya varias tareas accediendo al mismo puerto.
        - Comentar las nuevas funciones.
- Interacción con el sensor de temperatura y humedad
    - Instalar el sensor dentro del sistema
        - Estudiar la documentación: Comprender como funciona el sensor y que protocolo debemos usar para comunicarnos con él.
        - Realizar las conexiones necesarias: Añadir el sensor al breadboard realizando las conexiones necesarias.
        - Escribir funcionalidad para leer información del sensor: Escribir el código para que el LPC1769 pueda obtener los datos de temperatura y humedad.
        - Escribir documentación de las funciones añadidas.
    - Envío de datos a Internet
        - Escribir el código de envío de datos al servidor de Internet: Escribir el código para que dispositivo pueda enviar y recibir datos. Y el servidor pueda entender las nuevas funcionalidades que se piden.
        - Escribir documentación de las funciones añadidas.
- Finalización del proyecto

- Testeo del sistema: Probar que todo lo implementado funciona de un modo estable y no hay problemas
- Finalizar la documentación.

1.4.3 Documentación

Escritura de esta memoria durante el final de la fase anterior y tras la finalización de la anterior fase.

1.2 Planificación

A continuación, se presentarán dos diagramas de Gantt. El primero se corresponde a la planificación inicial que se esperaba y el segundo correspondiente a cómo han ido evolucionando las tareas.

1.5.1 Planificación inicial

Esta primera planificación muestra la evolución ideal de las tareas que hemos descrito anteriormente. Para lograr cumplir todas las estimaciones apenas debería de haber surgido ningún problema, posteriormente veremos cómo han evolucionado las diferentes tareas tras planteársenos los problemas que conlleva cualquier desarrollo.

ID	Task Name	Duration	Start	Finish
1	**Primera fase**	**44 days**	Wed 27/02/13	**Tue 23/04/13**
2	Instalación IDE	15 days	Wed 27/02/13	Sun 17/03/13
3	Conexión WiFly	15 days	Wed 27/02/13	Sun 17/03/13
4	Driver UART	10 days	Wed 27/02/13	Mon 11/03/13
5	Driver WiFly	9 days	Thu 21/03/13	Tue 02/04/13
6	Productor - Consumidor	9 days	Wed 03/04/13	Sat 13/04/13
7	**Segunda fase**	**38 days**	**Sun 14/04/13**	**Tue 04/06/13**
8	**Estabilización del sistema**	**11 days**	**Fri 03/05/13**	**Fri 17/05/13**
9	**Ensamblado de componentes**	**2 days**	**Fri 03/05/13**	**Mon 06/05/13**
10	Realizar las conexiones	1 day	Fri 03/05/13	Fri 03/05/13
11	Probar que funciona correctamente	1 day	Mon 06/05/13	Mon 06/05/13
12	**Mejorar drivers**	**9 days**	**Tue 07/05/13**	**Fri 17/05/13**
13	Separación Driver WiFly y UART	3 days	Tue 07/05/13	Thu 09/05/13
14	Usar esperas pasivas	2 days	Mon 13/05/13	Tue 14/05/13
15	Usar semáforos para acceder al UART	2 days	Tue 14/05/13	Wed 15/05/13
16	Comentar las funciones	2 days	Thu 16/05/13	Fri 17/05/13
17	**Interacción con el sensor de temperatura y humedad**	**6 days**	**Mon**	**Mon**
18	**Instalar el sensor dentro del sistema**	**4 days**	**Mon 20/05/13**	**Thu 23/05/13**
19	Estudiar documentación	1 day	Mon 20/05/13	Mon 20/05/13
20	Realizar las conexiones necesarias	1 day	Mon 20/05/13	Mon 20/05/13
21	Escribir funcionalidad para leer información del sensor	4 days	Mon	Thu 23/05/13
22	Escribir documentación de las funciones añadidas	1 day	Mon	Mon
23	**Envío de datos a Internet**	**2 days**	**Fri 24/05/13**	**Mon 27/05/13**
24	Escribir código de envío de datos al servidor de Internet	2 days	Fri 24/05/13	Mon
25	Escribir documentación de las funciones añadidas	1 day	Fri 24/05/13	Fri 24/05/13
26	**Finalizacación del proyecto**	**4 days**	**Tue 28/05/13**	**Fri 31/05/13**
27	Testeo del sistema	4 days	Tue 28/05/13	Fri 31/05/13
28	Finalizar la documentación	2 days	Tue 28/05/13	Wed 29/05/13
29	**Documentación**	**19 days**	**Thu 16/05/13**	**Tue 11/06/13**
30	Memoria	19 days	Thu 16/05/13	Tue 11/06/13

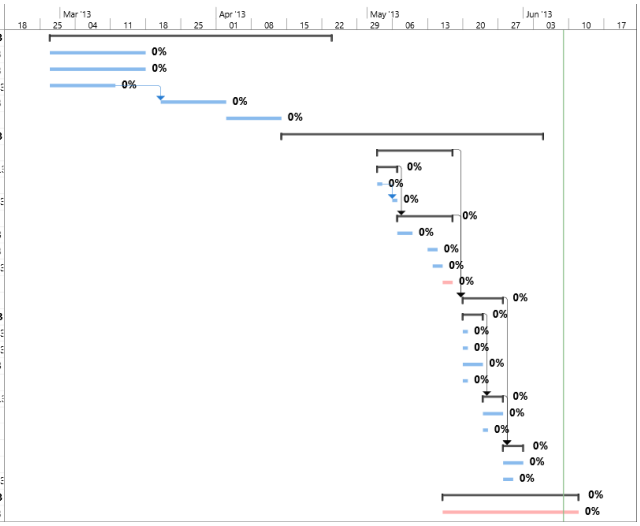


Figura 2 Diagrama de Gantt de la planificación inicial

1.5.2 Planificación final

En este segundo diagrama de Gant podemos ver cómo han ido evolucionando finalmente las tareas.

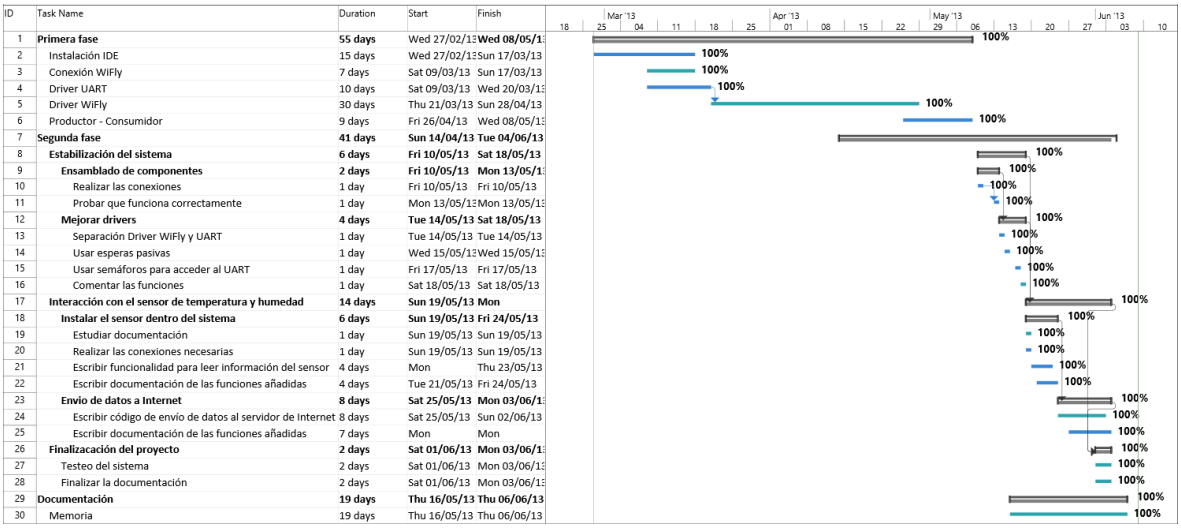
El primer problema que nos hemos encontrado ha sido un retraso de los materiales de algo más de una semana. Por lo tanto, al principio, sólo pude dedicar tiempo a la instalación de entorno a la espera de recibir los materiales.

Una vez recibido el material estaba incompleto y sólo se envió 5 cables hembra-hembra. Hubo que comprar más para poder hacer las conexiones necesarias, lo cual nos llevó a un pequeño retraso más.

Tras solucionar las primeras incidencias. Se puede apreciar que hubo un largo desfase en la tarea de desarrollo del driver WiFly debido a problemas para conectar el WiFly al LPC1769. Los pines del WiFly están demasiado juntos como para conectar los cables y tras varios intentos dejó de funcionar y hubo que reemplazarlo por uno nuevo. Esto obligó a comprar un nuevo WiFly.

Tras todos estos incidentes y una larga dilatación de los tiempos de las tareas de la primera fase, la segunda progresó correctamente, incluso reduciendo los tiempos planificados en un primer momento.

Se puede apreciar también un crecimiento de la tarea de "Envío de datos a Internet" ya que a esta se le añadieron nuevas funcionalidades, como el soporte para alarmas entre otras.



ID	Task Name	Duration	Start	Finish
1	**Primera fase**	**55 days**	Wed 27/02/13	**Wed 08/05/13**
2	Instalación IDE	15 days	Wed 27/02/13	Sun 17/03/13
3	Conexión WIFly	7 days	Sat 09/03/13	Sun 17/03/13
4	Driver UART	10 days	Sat 09/03/13	Wed 20/03/13
5	Driver WiFly	30 days	Thu 21/03/13	Sun 28/04/13
6	Productor - Consumidor	9 days	Fri 26/04/13	Wed 08/05/13
7	**Segunda fase**	**41 days**	**Sun 14/04/13**	**Tue 04/06/13**
8	**Estabilización del sistema**	**6 days**	**Fri 10/05/13**	**Sat 18/05/13**
9	**Ensamblado de componentes**	**2 days**	**Fri 10/05/13**	**Mon 13/05/13**
10	Realizar las conexiones	1 day	Fri 10/05/13	Fri 10/05/13
11	Probar que funciona correctamente	1 day	Mon 13/05/13	Mon 13/05/13
12	**Mejorar drivers**	**4 days**	**Tue 14/05/13**	**Sat 18/05/13**
13	Separación Driver WIFly y UART	1 day	Tue 14/05/13	Tue 14/05/13
14	Usar esperas pasivas	1 day	Wed 15/05/13	Wed 15/05/13
15	Usar semáforos para acceder al UART	1 day	Fri 17/05/13	Fri 17/05/13
16	Comentar las funciones	1 day	Sat 18/05/13	Sat 18/05/13
17	**Interacción con el sensor de temperatura y humedad**	**14 days**	**Sun 19/05/13**	**Mon**
18	**Instalar el sensor dentro del sistema**	**6 days**	**Sun 19/05/13**	**Fri 24/05/13**
19	Estudiar documentación	1 day	Sun 19/05/13	Sun 19/05/13
20	Realizar las conexiones necesarias	1 day	Sun 19/05/13	Sun 19/05/13
21	Escribir funcionalidad para leer información del sensor	4 days	Mon	Thu 23/05/13
22	Escribir documentación de las funciones añadidas	4 days	Tue 21/05/13	Fri 24/05/13
23	**Envio de datos a Internet**	**8 days**	**Sat 25/05/13**	**Mon 03/06/13**
24	Escribir código de envío de datos al servidor de Internet	8 days	Sat 25/05/13	Sun 02/06/13
25	Escribir documentación de las funciones añadidas	7 days	Mon	Mon
26	**Finalizacación del proyecto**	**2 days**	**Sat 01/06/13**	**Mon 03/06/13**
27	Testeo del sistema	2 days	Sat 01/06/13	Mon 03/06/13
28	Finalizar la documentación	2 days	Sat 01/06/13	Mon 03/06/13
29	**Documentación**	**19 days**	**Thu 16/05/13**	**Thu 06/06/13**
30	Memoria	19 days	Thu 16/05/13	Thu 06/06/13

Figura 3 Diagrama de Gantt de la planificación final

1.6 Recursos empleados

Para la creación de este dispositivo se han usado varios recursos, tanto de hardware como de software.

1.6.1 Hardware

En el caso del hardware usado para la realización de este proyecto podemos encontrar tanto dispositivos electrónicos, cables y diferentes accesorios.

También se han utilizado ciertos aparatos de medición y depuración para comprobar el correcto funcionamiento del sistema.

- **Breadboard**
  Usada para realizar las conexiones sobre ella.

Figura 4 Breadboard

- **LPC1769**
  Placa que contiene el microcontrolador ARM Cortex M3

Figura 5 LPC1769

- **WiFly RN-XV 802.11b/g**
  Módulo WiFi

Figura 6 WiFly RN-XV 802.11b/g

- Adaptador DIP para XBEE
  Sirve para conectar el WiFly a la placa breadboard

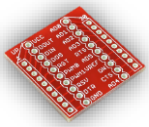


Figura 7 Adaptador DIP para XBee

- CP2102
  Convertidor UART a USB

Figura 8 CP2102

- SHT15

  Sensor de humedad y temperatura

Figura 9 SHT15

- 4xLED

  Para emitir señales

Figura 10 Diodo led

- Cables hembra-hembra

  Para realizar conexiones temporales

Figura 11 Cables hembra-hembra

- Zócalos de conexión

  Para soldar al adaptador y sensor

Figura 12 Zócalos de conexión

- Soldador JBC 14s

  Para soldar las conexiones

Figura 13 JBC 14s

- Multímetro Best DT9205A

  Para realizar las mediciones de consumo.

Figura 14 Multímetro Best DT9205A

- Analizador de señales Logic16
  Para la depuración de las señales emitidas por el driver GPIO para la comunicación con el sensor SHT15.

Figura 15 Analizador de señales Logic16

- PC con sistema operativo Windows 8 Enterprise, aunque podría haberse usado un ordenador con sistema operativo Linux ya que el IDE es compatible con él.

1.6.2 Software

- IDE LPCXpresso v5.1.2_2065 basado en Eclipse Juno 4.2.1
- Sistema operativo en tiempo real FreeRTOS V6.1.1
- Software Logic16
- Lenguaje de programación C.
- IDE LPCXpresso v5.1.2_2065 basado en Eclipse Juno 4.2.1
- Lenguaje de programación Java
- Eclipse Juno 4.2.2
- Google App Engine
- HTML

1.7 Productos obtenidos

Como se verá en los siguientes capítulos el resultado obtenido son dos productos.

Primero tenemos un sistema empotrado en forma de prototipo que usando la mota LPC1769 el cual se ha conectado al WiFly y el sensor de humedad y temperatura SHT15 para cumplir los objetivos marcados en el proyecto.

También tenemos el servidor web, que se encarga de almacenar y mostrar los datos recibidos además de usarse para configurar el dispositivo empotrado remotamente.

1.8 Descripción de los otros capítulos de la memoria

En el resto de capítulos se hará un análisis de las tecnologías, componentes, funcionalidades creadas, viabilidades, etc.

- **Capítulo 2**: Se hará una presentación del estado de la tecnología que hemos usado para realizar el proyecto y un estudio de mercado actual.
- **Capítulo 3**: Aquí se hará una descripción funcional del sistema donde trataremos el sistema completo, la parte del servidor y la mota por separado.
- **Capítulo 4**: En este capítulo se entrará a hablar en profundidad de forma técnica en cómo funciona el sistema.
- **Capítulo 5**: Se hará una viabilidad técnica del proyecto que nos hemos propuesto. Y resumiremos los puntos fuertes y débiles del mismo.
- **Capítulo 6**: Se comprobará la viabilidad económica del proyecto.
- **Capítulo 8**: Se comentarán las conclusiones sacadas tras la finalización del proyecto.
- **Capítulo 9**: Un glosario donde se podrán encontrar una relación de ciertos términos que pueda desconocer el lector.
- **Capítulo 10**: Aquí mostraremos la bibliografía usado durante el transcurso del proyecto para entender ciertos conceptos o resolver dudas.
- **Capítulo 11**: Dentro de este capítulo encontraremos los anexos del proyecto.

2. Antecedentes

2.1 Estado del arte

Un sistema empotrado es un sistema de computación diseñado para realizar un conjunto de funciones o tareas específicas en tiempo real. Las tareas que debe de realizar van desde control de motores, procesamiento de datos a pequeñas escala, entre otras.

El componente central de un sistema empotrado es la CPU. Y es la que controla todo el sistema. Gracias a él se pueden conectar otros dispositivos que realizan funciones de medición, control, etc.

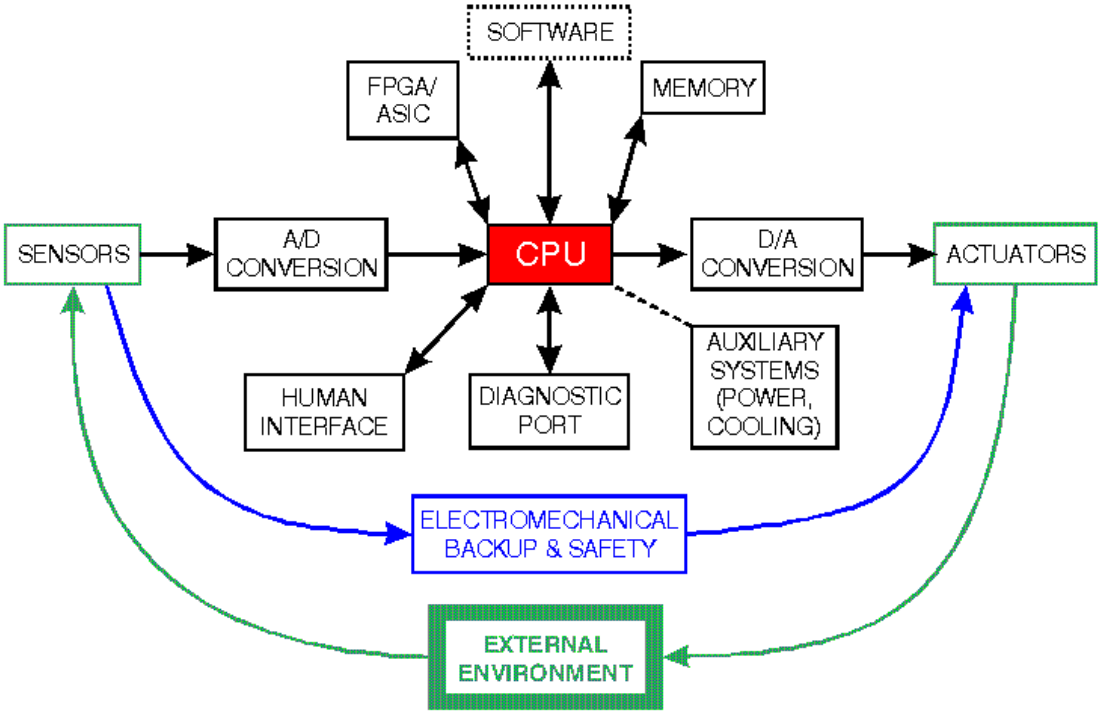


Figura 16 Esquema conceptual de un sistema empotrado

En la figura anterior podemos ver el diagrama de bloques de cómo funcionaría un sistema empotrado. Como podemos observar tenemos un bloque principal que se corresponde a la CPU y a la cual se le conectan diversos periféricos para realizar tareas diversas.

Ahora mostraremos una descripción más completa de los componentes que usamos en el sistema y gracias a los cuales este proyecto se ha podido completar satisfactoriamente.

2.1.1 LPC1769

Este es el principal componente del sistema. De él podríamos decir que es el cerebro de todo, ya que es el dispositivo que programaremos para que se encargue de controlar el resto de componentes del sistema.

Es un microcontrolador ARM Cortex-M3 de 32 bits, con un 512 KB de memoria flash y soporte Ethernet, USB2.0.

El LPC1769 que usamos puede operar hasta velocidades de 120 MHz.

En la parte de las conexiones, el LPC1769 cuenta con las siguientes funcionalidades:

- Ethernet
- USB
- Un controlador DMA con 8 canales.
- 4 puertos UART.
- interfaces de bus I²C
- Un reloj en tiempo real (RTC) de muy bajo consumo.
- Y muchas más…

Como podemos ver es un chip muy completo.

2.1.2 RN-XV WIFLY 802.11 B/G

Este dispositivo permite a la placa LPC1769 el envío y recepción de datos desde Internet para poder almacenar en un servidor remoto las mediciones tomadas por el sensor de temperatura y humedad.

Para la comunicación con LPC1769 se usa uno de los puertos UART, que es el principal modo que tenemos de comunicarnos con el WiFly. Lleva un servidor TCP integrado mediante el cual una vez que estemos a un punto de acceso un ordenador, que se encuentre en la misma red que el WiFly, podría conectarse utilizando el protocolo telnet al puerto 2000 y de este modo ver los diferentes mensajes que le llegan al WiFly.

Debido a que la separación de pines es inferior a la del resto de componentes se ha tenido que adquirir un adaptador XBee, compatible con el WiFly. De este modo se ha podido insertar el WiFly en la Breadboard sin ningún problema para darle un poco de solidez al sistema. Como comentamos

en la planificación la mínima separación de pines del WiFly fue problemática al principio. Por tanto recomiendo completamente la adquisición de este adaptador para evitar estos problemas.

2.1.3. CP2102

Este adaptador permite la comunicación bidireccional entre el ordenador y el sistema electrónico.

Gracias a él podemos tener un canal de log entre el LPC1769 y un programa de terminal dentro del ordenador.

Otro de los usos de este adaptador es conectar el WiFly para hacer pruebas de conexión contra servidores sin tener que escribir el código y cargárselo al LPC1769. Al principio del TFC se usó para comprender el funcionamiento del WiFly para luego tener las pautas a seguir para escribir el driver.

Para la comunicación con los otros componentes se usa UART y para la comunicación con el ordenador USB. Puede alimentar a los componentes gracias a que el USB permite que el CP2102 haga de fuente de alimentación.

2.1.4 SHT15

Este es el sensor de humedad y temperatura que he usado en el proyecto. Es muy rápido en realizar mediciones, sólo necesita aproximadamente 80 milisegundos para obtener cualquiera de los dos datos. Para comunicarse con el sensor hay que enviar comandos usando el protocolo TWI (Two Wire) para lo cual he tenido que dedicar 2 pines GPIO del LPC1769 para que pudiesen hablar.

Este sensor es capaz de medir una temperatura entre $(-40, 123.8)°\mathbb{C}$ con un posible error de $\pm 0.3°\mathbb{C}$. En el caso de la humedad relativa al tratarse de porcentaje los valores variarán entre el 0% y 100%, con un margen de error del $\pm 0.3\%$.

2.2 Estudio de mercado

Lo primero conoceremos un poco de la historia de los primeros sistemas empotrados.

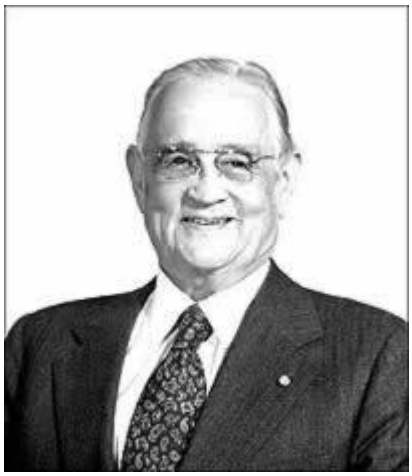


Figura 17 Charles Stark Draper

Charles Stark Draper (2 de octubre de 1901 – 25 de julio de 1987) fue un ingeniero y científico estadounidense, es llamado frecuentemente como “el padre de la navegación inercial”. Fue el fundador del Laboratorio de Instrumentación (MIT Instrumentation Laboratory).

Suyo fue el primer sistema empotrado reconocido. Este fue el ordenador de guía del Apollo en el Laboratorio de Instrumentación del MIT.

Este proyecto fue considerado el más delicado de todo el Proyecto Apollo, porque utilizaba por primera vez un circuito integrado desarrollado monolíticamente para reducir su peso y su tamaño.

Gracias a este dispositivo empotrado se hicieron posibles los alunizajes de la era Apollo.

Tras esta breve nota de la historia de los dispositivos empotrados, veremos qué posición ocupan en la actualidad.

En la actualidad existen multitud de dispositivos empotrados y debido a su bajo coste, cada día aumenta el número de aficionados a la electrónica que los usan para diseñar sus prototipos basados.

Tenemos diferentes compañías dedicadas al diseño de microcontroladores que pueden ser utilizados para nuestros sistemas empotrados. Algunos de ellos, con ejemplos de algunos de sus productos son los siguientes:

- AMCC: PowerPC 4xx
- Atmel: Serie AT89, AT90, ATtiny, ATMega (usado en Arduino), …
- Cypress MicroSystems: CY8C2xxxx, CY8C3xxxx, CY8C5xxxx
- Freescale Semiconductor
    - 8 bits: 68HC05, 68HC08, 68HC11
    - 16 bits: 68HC12, 68HC16, Freescale DSP56800
    - 32 bits: Freescale 683XX, MPC500, MPC 860, MPC 8240/8250, MPC 8540/8555/8560
- Intel
    - 8 bits: MCS-48, MCS-51, 8xC251

    - 16 bits: MCS-96, Intel MCS 296
- NEC: 17K, V25, 75X, 78K, V850
- STMicroelectrics: ST 62, ST 7, ST 10
- Texas Instruments: TMS370, MSP430
- Toshiba: TLCS-47, TLCS-870, TLCS-900, TX19A

Estos son sólo algunos de los fabricantes y modelos de microcontroladores que podemos encontrar actualmente en el mercado.

Para facilitar el trabajo al programador de sistemas empotrados tenemos la posibilidad de utilizar diferentes sistemas operativos para sacar el máximo partido al microcontrolador y sus recursos. Facilitando varias herramientas como la parelelización de tareas, controlar el acceso a determinado recurso, la comunicación entre las tareas que se estén ejecutando en paralelo.

Microsoft por ejemplo tiene su propio sistema operativo para dispositivos empotrados, este es el Windows Embedded Compact (Windows CE), este se puede encontrar desde smartphones (los últimos en los que se ha usado son en los Windows Phone 7.X), notebooks, pocket pcs y gps.

En el caso de Linux tenemos multitud de distribuciones para dispositivos empotrados como por ejemplo los famosos Androids. Pero además muchas distribuciones para PCs también tienen su versión empotrada.

En nuestro caso hemos utilizado FreeRTOS que es un sistema operativo en tiempo real que utiliza un planificador sencillo, pero suficiente para las tareas que hemos realizado.

Ya buscando más información sobre el proyecto que nos ocupa se ha encontrado un producto que podría asemejarse a nuestro medidor online, este es el GPRS Temperarature Logger S26x.

Figura 18 GPRS Temperature Logger S26x

Este producto permite medir y enviar a Internet la temperatura actual además de mostrarla en una pantalla simple como podemos ver en la figura anterior.

Este medidor tiene varias desventajas con respecto al nuestro, es más pesado, más grande y sólo mide temperatura.

Pero a favor tiene que en su pantalla puede mostrar temperatura actual y la fecha.

3. Descripción funcional

Antes de entrar en el detalle de cada una de las partes se realizará una exposición general del sistema.

3.1 Sistema total

El sistema se divide en dos partes independientes:

- Medidor online de temperatura y humedad de bajo consumo, que se corresponde con el dispositivo que realizará las mediciones y mostrará una alarma luminosa en caso de temperatura o humedad crítica. Á su vez puede obtener la configuración actual del servidor de Internet.
- Servidor web: Almacenará y representará los datos de temperatura y humedad emitidos por el medidor. Además almacena la configuración y en caso de recibir un dato fuera de los rangos de seguridad mandará un correo electrónico a un usuario para alertar de un valor crítico recibido.

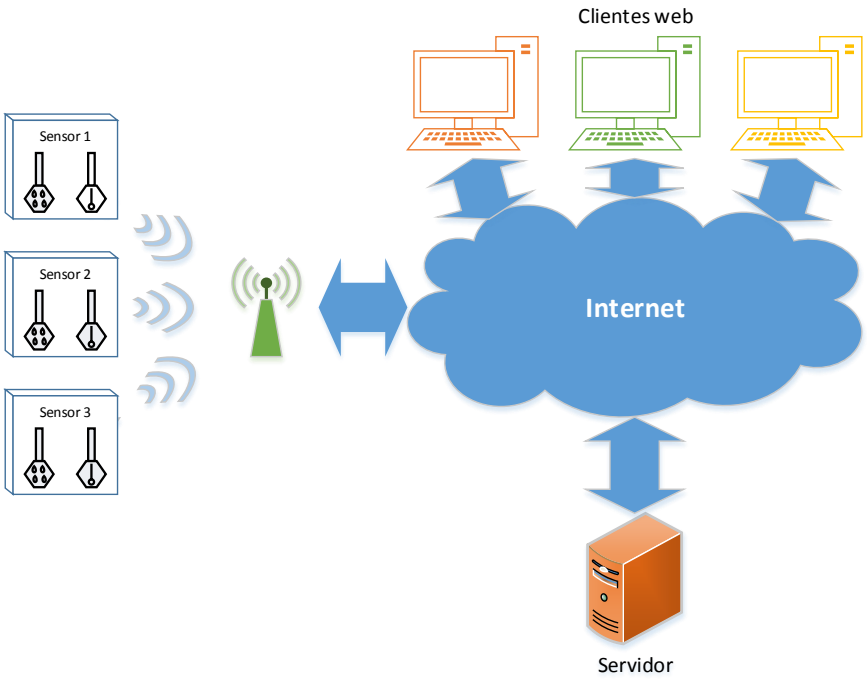


Figura 19 Diagrama de bloques del Sistema Total

Como podemos ver en la figura anterior, el servidor al ser una aplicación web puede ser accedida por varios usuarios con a la vez desde sus estaciones de trabajo, tablets, portátiles, móviles, etc. No se le pone ninguna restricción para mayor flexibilidad.

Los diferentes sensores que pueda haber instalados no pueden funcionar de forma autónoma ya que necesitan conocer la configuración que sólo el servidor puede proporcionar. Si usamos diferentes sensores estos se diferenciarán en el servidor por su dirección IP y al identificador de red a la que están conectados. De este modo podríamos tener varias redes de sensores funcionando al mismo tiempo y varios sensores dentro de cada red.

Para la comunicación entre el sensor y el servidor es necesario que haya un punto de acceso que permita la conexión a Internet de dispositivos Wi-Fi, este punto de acceso debe tener visibilidad del servidor de Internet, aunque el sistema funcionaría en red local sin problema. Para poder enviar los datos tiene que ser posible hacer peticiones HTTP desde los dispositivos conectados a ese punto de acceso ya que este es el modo en el que se comunican sensores y servidor.

3.2 PC

La aplicación de servidor ha sido desarrollada para mostrar la funcionalidad y potencial de este sistema. Los sensores podrían integrarse sin muchos problemas dentro de otro sistema con mínimas modificaciones.

En la figura de a continuación podremos ver el interfaz que da al usuario.

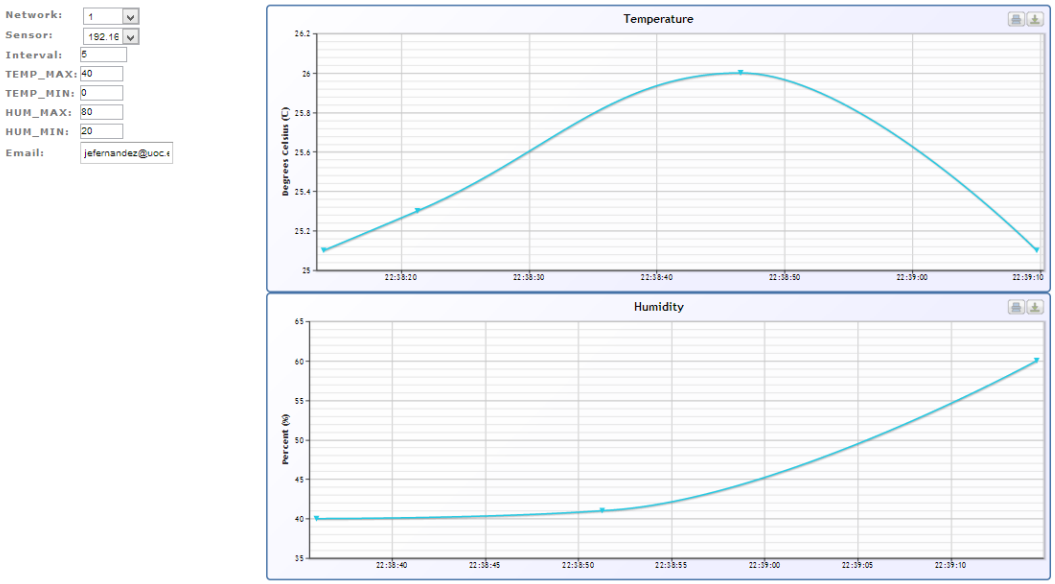


Figura 20 Aplicación web

El diagrama de bloques de cómo se relaciona el servidor se pude ver en la siguiente figura.

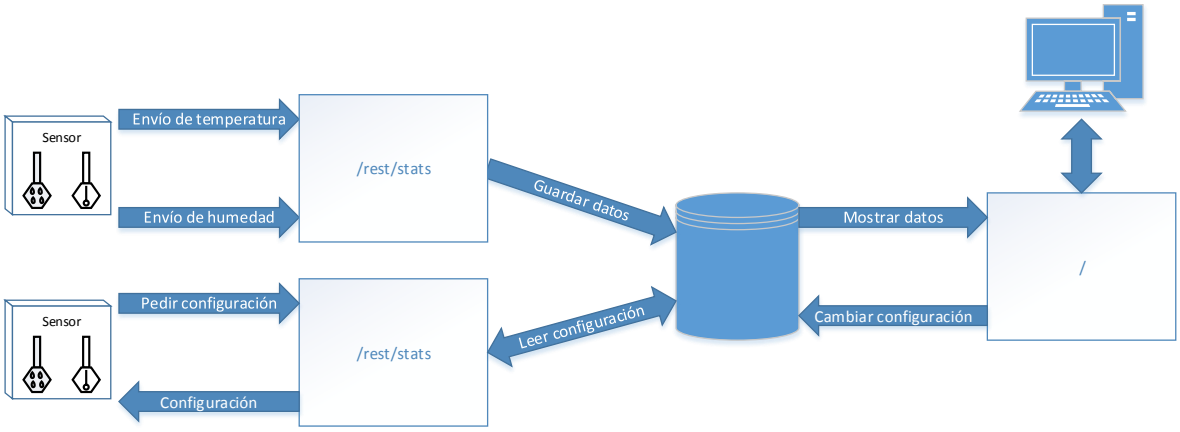


Figura 21 Diagrama de bloques Aplicación web

En ella vemos como los sensores hacen una petición a una URL del servidor para enviar el dato. El servidor guarda sus datos para su posterior consulta.

Un sensor también podría hacer una consulta de los datos de configuración actuales para sobrescribir los valores de configuración que estén en uso actualmente.

En caso de que un usuario quiera consultar los datos que han sido registrados este usuario debería acceder a la raíz del servidor donde tenemos la aplicación web que nos muestra en tiempo real las mediciones. El usuario no tiene ningún tipo de restricción respecto a los datos que puede ver.

El usuario conectado al interfaz podría cambiar los datos actuales para que en próximas actualizaciones de los sensores estos cojan la nueva configuración.

3.3 Mota

La mota es un dispositivo sensor de temperatura y humedad. Es decir se ocupa de medir la temperatura y humedad ambiental en lugar donde se encuentre instalado tratando de consumir la menor energía eléctrica posible para aumentar su autonomía al máximo.

El LPC1769 está programando para tener tres tareas siempre funcionando e interaccionando con los diferentes dispositivos instalados dentro de sistema empotrado.

Una de estas tareas se ocupará de la lectura de la configuración de Internet utilizando el módulo Wi-Fi conectado a la placa. Esta petición como las otras se hará enviando una petición HTTP contra el servidor. La respuesta del servidor será la configuración actual, que contiene:

- Intervalo de toma de datos (en minutos)
- Temperatura máxima y mínima
- Humedad máxima y mínima.

Una vez se lea por primera vez la configuración la siguiente tarea en ejecutarse será la de lectura de datos del sensor SHT15 conectado a la placa. Cuando tiene los datos los inserta dentro de una cola para ser mandados a la tercera tarea. También cuando tiene los datos los comprueba con los valores máximos y mínimos guardados en la configuración para ver si existe algún problema y en ese caso encender el led correspondiente.

La tercera y última tarea se queda a la espera de que haya datos de temperatura o humedad para ser enviados al servidor.

En la siguiente figura se aprecia el diagrama de bloques de cómo interactúan los diferentes módulos del dispositivo empotrado.

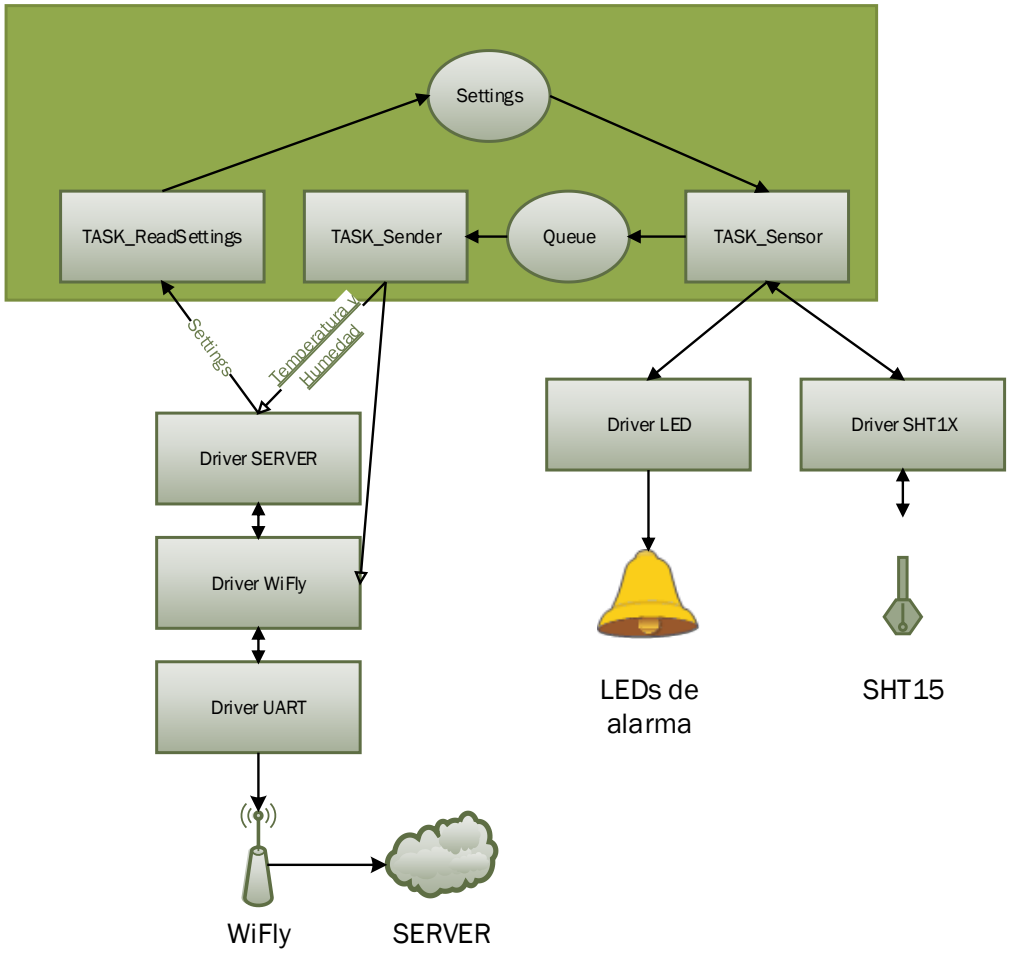


Figura 22 Diagrama de bloques del programa de la mota

4. Descripción detallada

Ya tenemos información general de cómo funciona el sistema, ahora nos centraremos en cómo funciona el código que hemos escrito para hacer funcionar todo el dispositivo.
Como ya hemos comentado en otros puntos programaremos el LPC1769 para poder realizar las acciones necesarias.
Empezaremos comentando el programa principal que se encarga de hacer las llamadas a los drivers correspondientes.
Posteriormente pasaremos a repasar uno por uno los diferentes drivers que se han escrito para facilitar la tarea y no repetir código.
La biblioteca creada para el proyecto contiene los siguientes módulos:

- UARTIO
- WIFLY
- SERVER
- SHT1X
- LED
- SLEEP

Cada módulo encapsula una funcionalidad que puede ser utilizada desde otros módulos o desde el programa principal.
A continuación podemos ver un diagrama de bloques que nos muestra cómo se relacionan los diferentes drivers entre sí.

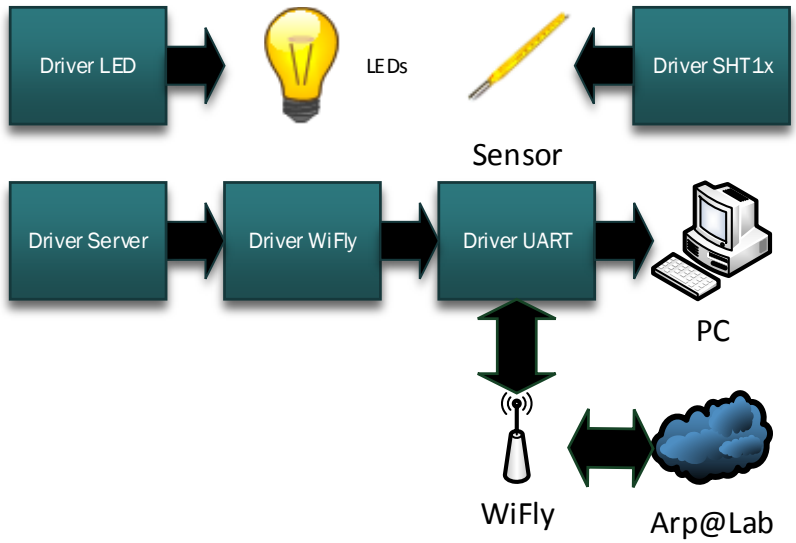


Figura 23 Diagrama de bloques de los drivers

Como hemos visto en la figura anterior tenemos tres controladores que funcionan de un modo independiente, estos son el driver SHT1x, el driver LED y el driver UART.
Por otro lado también vemos que el driver WiFly depende del driver UART. Y a su vez el driver Server depende del driver WiFly para conectar con el servidor.
El caso del UART es especial, ya que puede funcionar por sí solo para que el programa pueda enviar mensajes a un ordenador conectado a CP2102. Pero también puede ser usado desde el driver WiFly para comunicarse con el WiFly.

Aunque en los siguientes apartados veremos cómo se han realizado las conexiones entre el LPC1769 y los diferentes dispositivos con más detalle en la figura siguiente podremos ver una imagen global de las conexiones.

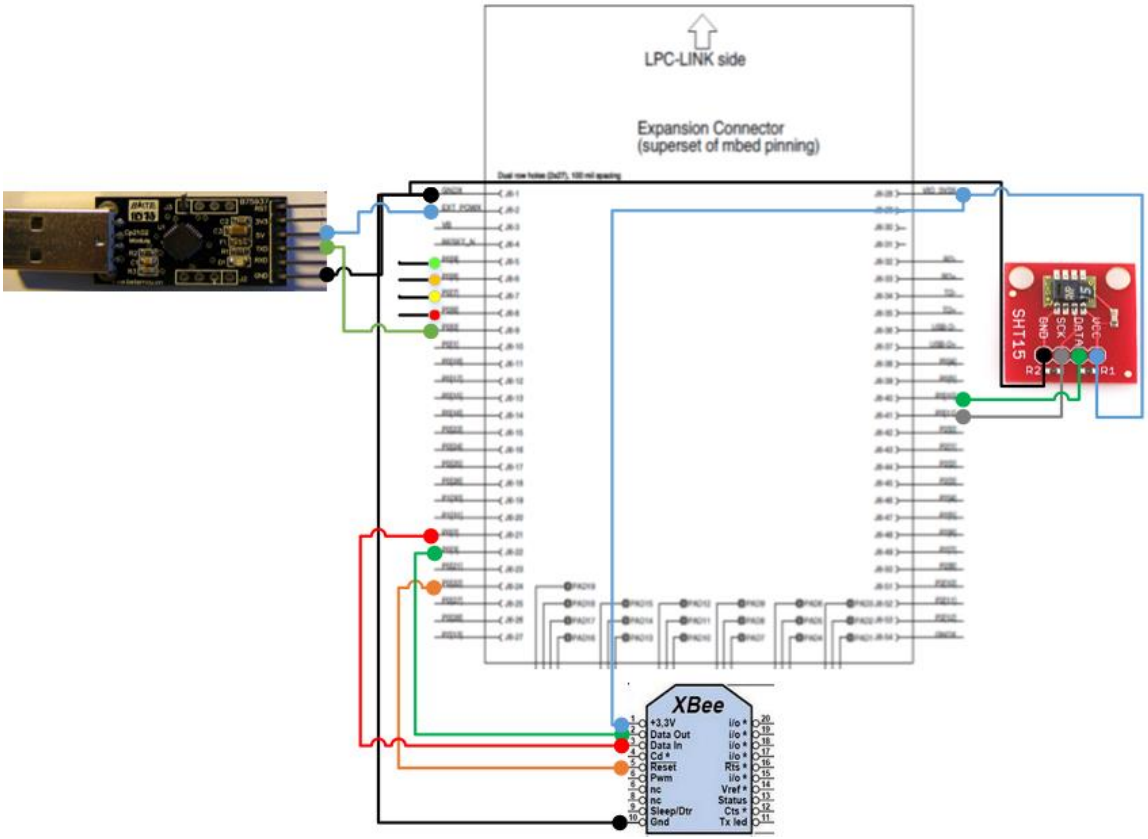


Figura 24 Conexiones entre los componentes y el LPC1769

4.1 Programa principal

El programa principal se encarga de canalizar el flujo de cómo funcionan las diferentes tareas que hemos identificado anteriormente.

Recordemos que tenemos tres tareas independientes ejecutándose paralelamente, a continuación procederemos a explicarlas.

4.1.1 Lectura de configuración

Esta tarea se corresponde con el siguiente diagrama de flujo donde podemos ver cómo se comporta.

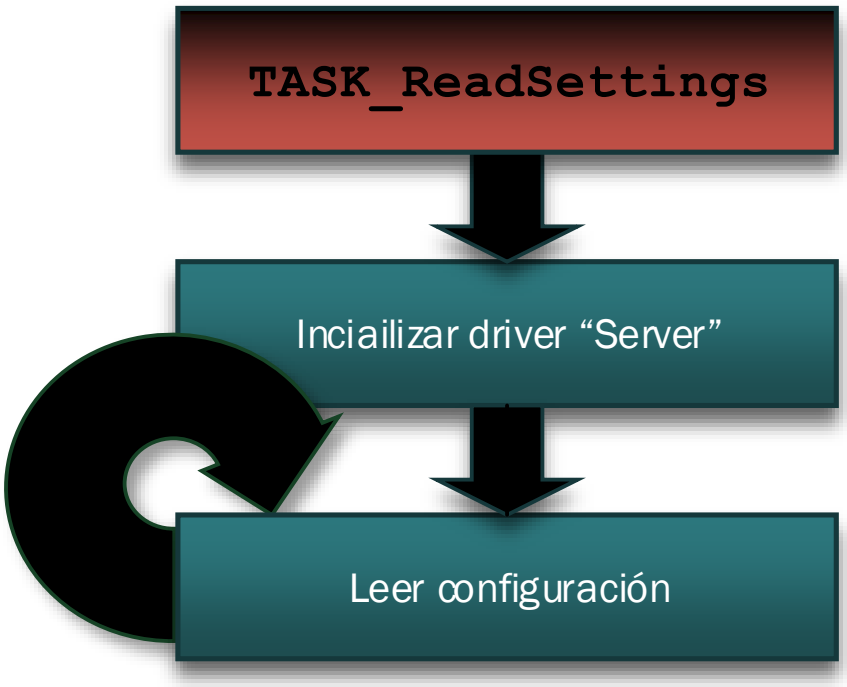


Figura 25 Diagrama de flujo ReadSettings

En ella vemos que esta tarea se encarga de acceder periódicamente a Internet en busca de nuevos datos que usar a modo de configuración. Se encarga de inicializar el driver Server ya que es la primera tarea que hace uso de ese driver.

4.1.2 Lectura de datos del sensor

Esta tarea se encarga de la lectura de los datos que nos suministra el sensor. Antes de nada se inicializará el driver que controla el sensor para tenerlo disponible.

Depende, como hemos dicho, de la tarea anterior ya que hasta que no tengamos la configuración disponible no procederemos a la lectura de datos. Esto es así para evitar realizar lecturas de datos que podrían estar activando una alarma y los estaríamos pasando desapercibidos. También se

podría leer los datos y esperar tras ello por la configuración, pero eso nos introduciría cierto desfase entre que leemos los datos y los enviamos al servidor.

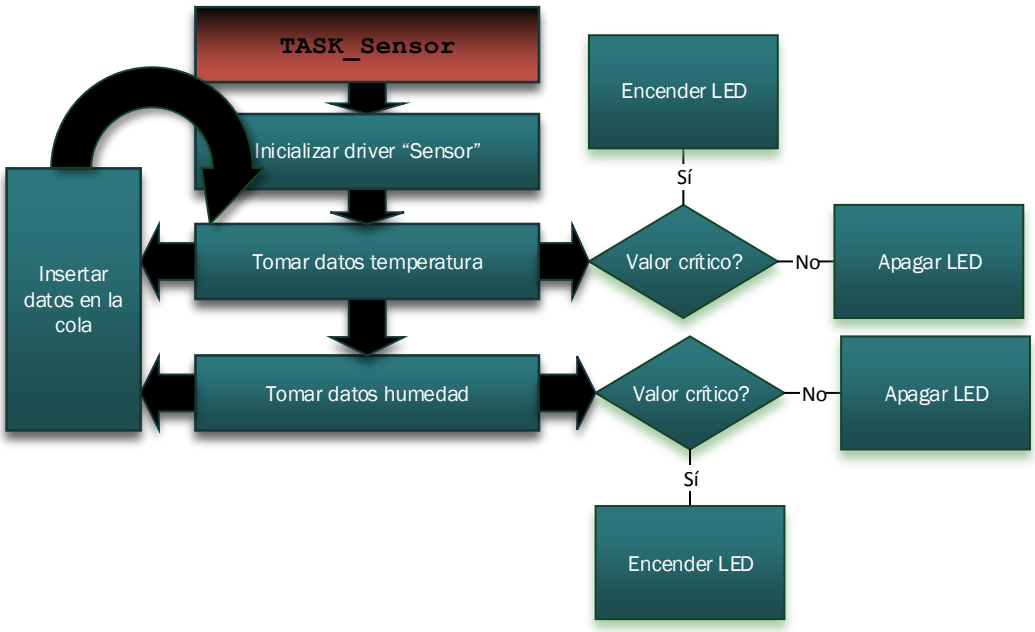


Figura 26 Diagrama de flujo Sensor

Pediremos al sensor los datos de temperatura y humedad y por cada uno comprobaremos con la configuración actual si es necesario encender uno de los leds de alarma.
Por cada dato que leamos este será metido dentro de la cola para ser consumido por la siguiente y última tarea. Como podemos ver esta tarea tiene el rol de productor, según una PEC anterior donde teníamos un modelo productor-consumidor.

4.1.3 Envío de datos a Internet

El diagrama de flujo de la tarea es el siguiente.

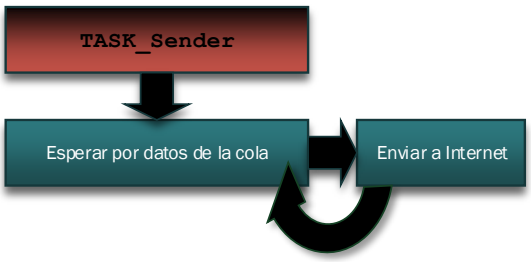


Figura 27 Diagrama de flujo Sender

Podemos observar que esta es la única de las tres tareas que no realiza ninguna inicialización de drivers, esto es así porque el único driver que usa (driver server) ya se encuentra inicializado, al ser usado por la primera tarea.

La tarea se dedica a esperar datos que deben de ser colocados en la cola compartida con la tarea que usa el Sensor, para poder después enviar estos datos a Internet.

Una vez hecho el envío de datos la tarea vuelve a ponerse a la espera de recibir más datos.

Esta tarea hace las veces de consumidor siguiendo el esquema productor-consumidor.

4.2 UARTIO

Esta fue la primera biblioteca implementada para el sistema y se utiliza para comunicar el LPC1769 con los diferentes dispositivos que hacen uso de UART para su comunicación a través de los pines que le indiquemos.

Gracias a este módulo podremos comunicarnos con un ordenador conectado por el adaptador CP2102 o al módulo WiFly, ya que ambos utilizan UART para su comunicación.

Las conexiones que tenemos que hacer entre el CP2102 y el LPC1769 se pueden ver en la siguiente figura. Las conexiones con el WiFly se estudiarán más adelante.

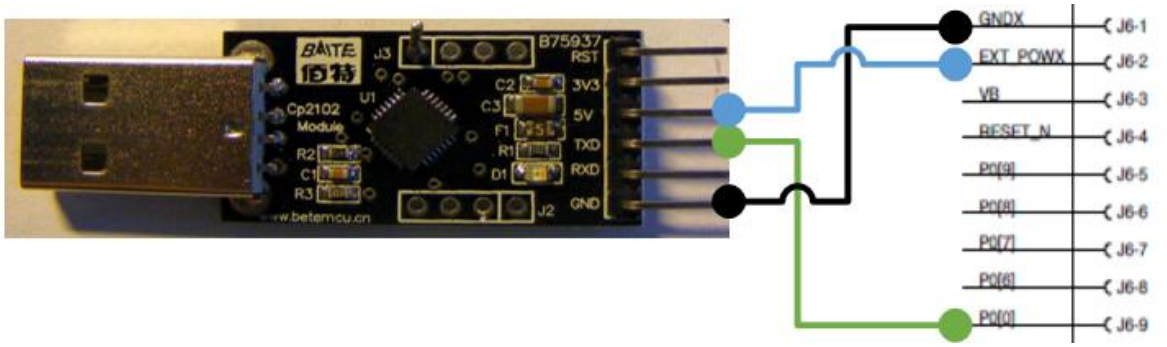


Figura 28 Conexiones CP2102 - LPC1769

Pin LPC1769	Pin CP2102
GNDX J6-1	GND
EXT_POWX J6-2	5V
P0[0] J6-9	TXD

Está conexión se corresponde al puerto UART3.

Se podría también conectar el pin RXD pero ya que no es necesario al no haber comunicación desde el ordenador al sistema, así que lo hemos omitido.

Comentar también que se está usando el CP2102 para alimentar el sistema, gracias a que usando un multímetro conectado en serie en medio de la conexión 5V – EXT_POWX podemos conocer el consumo del sistema en todo momento.

Para hacer uso de esta librería hay que incluir el archivo “uartio.h” que contiene las cabeceras con las funciones disponibles en la biblioteca.

Para hacer uso de un puerto UART con esta biblioteca es necesario inicializar el puerto antes de poder utilizarlo para de ese modo configurar las interrupciones necesarias y el mutex de acceso al puerto.

Mención especial hay que hacer a unas macros definidas como “LOG” que permiten el envío rápido al ordenador conectado por CP2102 de mensajes. Esta macro que facilita estos mensajes encapsula la función de envío de datos añadiéndole automáticamente la función en la que nos encontramos cuando aparece el mensaje, la línea. Esto es muy útil para encontrar errores con un simple vistazo a un terminal. En la figura siguiente podemos ver una imagen del “putty” mostrando los mensajes de depuración:

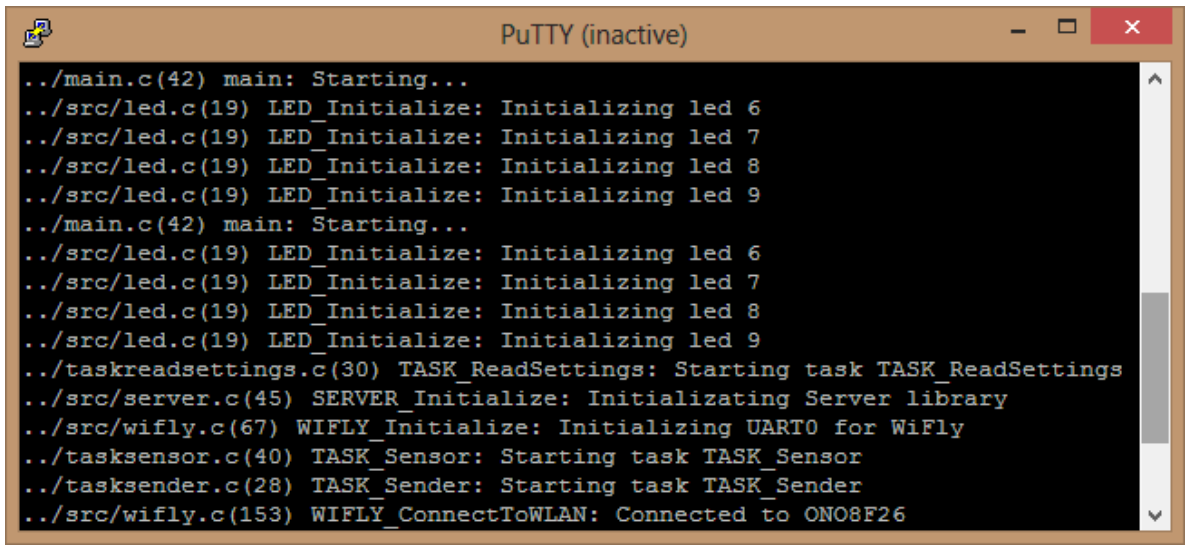


Figura 29 Putty mostrando mensajes de depuración

Comentar también que la macro LOG tiene también varios niveles de funcionamiento pudiendo mostrar más o menos mensajes según sea necesario. De este modo se permite que dependiendo de si hay problemas o no dentro de nuestro código podamos mostrar más información o sólo la básica para ir siguiendo el flujo de la aplicación.

4.3 WIFLY

Esta biblioteca ha sido implementada para operar con el módulo WiFly y de ese modo tener acceso a Internet en todo momento.

Del mismo modo que el CP2102 que vimos en la sección anterior este dispositivo se conecta también por UART y más concretamente al puerto UART0.

Las conexiones que hay que realizar para integrar el WiFly dentro del sistema se pueden apreciar en el detalle del diagrama de conexiones que podemos ver a continuación.

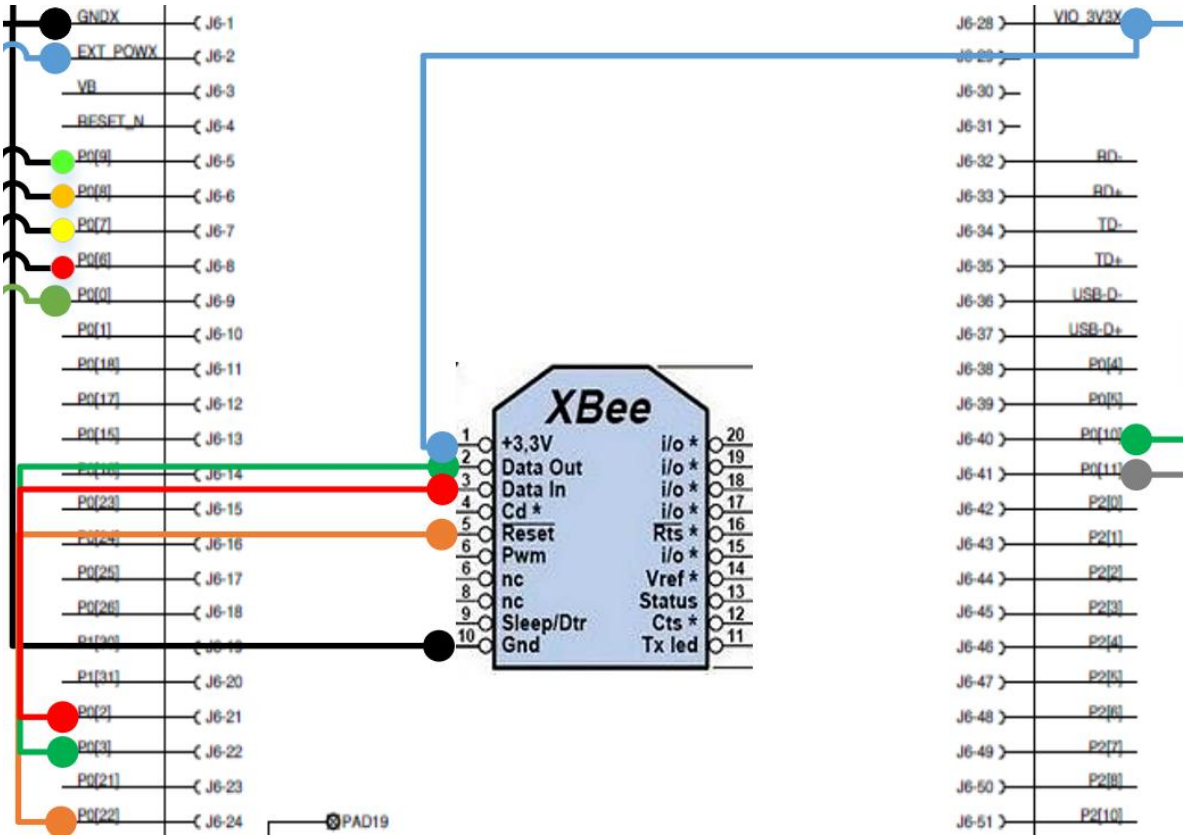


Figura 30 Conexiones WiFly - LPC1769

Pin LPC1769	Pin WiFly
VIO 3V3X J6-28	+3,3V
P0[3] J6-22	Data Out
P0[2] J6-21	Data In
P0[22] J6-24	Reset
GNDX J6-1	Gnd

Aparte de los pines para comunicarnos con el WiFly por el UART0, la alimentación y toma de tierra podemos ver un quinto pin conectado a la patilla RESET del WiFly desde una de las GPIO. Este pin es usado cuando vamos a realizar la inicialización del WiFly para reiniciarlo y así asegurarnos de que acaba de ser encendido.

Para hacer uso de esta librería tenemos que incluir el archivo “wifly.h” que nos dará acceso a las funciones de la biblioteca.

Esta biblioteca como la anterior es necesario inicializarla para poder configurar las estructuras que nos permiten bloquear el acceso al WiFly cuando ya está en uso.

También es necesario inicializarlo para que pueda inicializar a su vez el puerto UART que utilizará para comunicarse con el WiFly.

4.4 Server

Esta biblioteca encapsula los comandos para enviar o recibir datos del servidor. Hace uso internamente de la biblioteca WiFly. El diagrama de bloques de como se relaccionan los tres drivers para al final acceder a Internet viene representado por la siguiente figura.

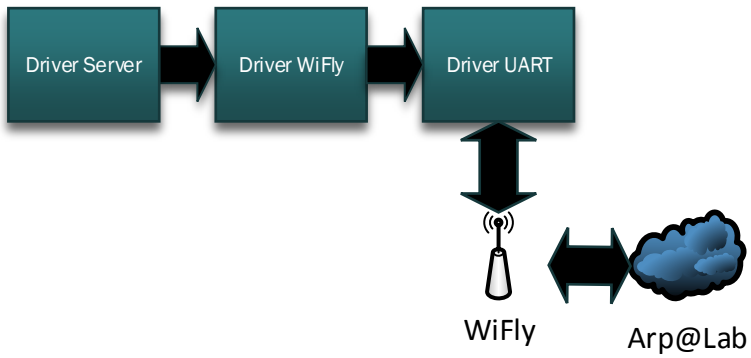


Figura 31 Diagrama de bloques del driver Server

Para usar esta biblioteca tenemos que incluir en nuestra aplicación el archivo de cabecera “server.h” y a partir de ese momento podrémos utilizar las siguientes funciones:

- **SERVER_Initialize**: Inicializa el módulo de control del envio y recepción de datos. El primer parámetro que hay que pasarle a esta función es el identificador de UART al que tenemos conectado el WiFly. Además como necesita conectarse a la red a través de un punto de acceso hay que pasarle los parámetros que vienen dados por: ssid, tipo de seguridad habilitada en el punto de acceso y contraseña del punto de acceso.

- **Server_SendStats**: Envía un dato al servidor para lo almacene en su base de datos. Los parámetros que recibe son los siguientes, nombre de aplicación y valor a enviar. Pensando en una posterior ampliación del proyecto no se limita sólo a las dos aplicaciones que contemplamos ahora (temperatura y humedad).
  Devuelve verdadero en caso de que el envío se haya realizado correctamente y falso si ha fallado.
- **Server_GetSettings**: Recibe la configuración del servidor y la almacena en la estructura pasada por parámetro.

En este caso hemos comentado las funciones ya que dos de ellas son muy importantes para entender el flujo del sistema correctamente. Se corresponden con las acciones que puede hacer el dispositivo contra el servidor. La otra es la función de inicialización, que es necesaria antes de empezar a trabajar con el servidor.

Las dos funciones que hemos comentado que interactúan con el servidor web comprueban siempre antes de tratar de realizar el envío si estamos conectados a la red, para que antes de nada intentar volver a conectarse. De este modo podrá sobreponerse el sistema a errores de conexión que se hayan podido dar o si el WiFly se ha dormido y acaba de ser despertado.

4.5 SHT1X

Esta biblioteca se encarga de interactuar con un sensor de temperatura y humedad SHT1x.

Este sensor se conecta al microcontrolador a través de un pin GPIO para la señal de DATA y otro pin GPIO que hará las veces de CLOCK y que estará conectada a la SCK en el sensor.

Además de estos pines habrá que alimentarlo y conectarlo a tierra como el resto de componentes, para ello usaremos las líneas comunes.

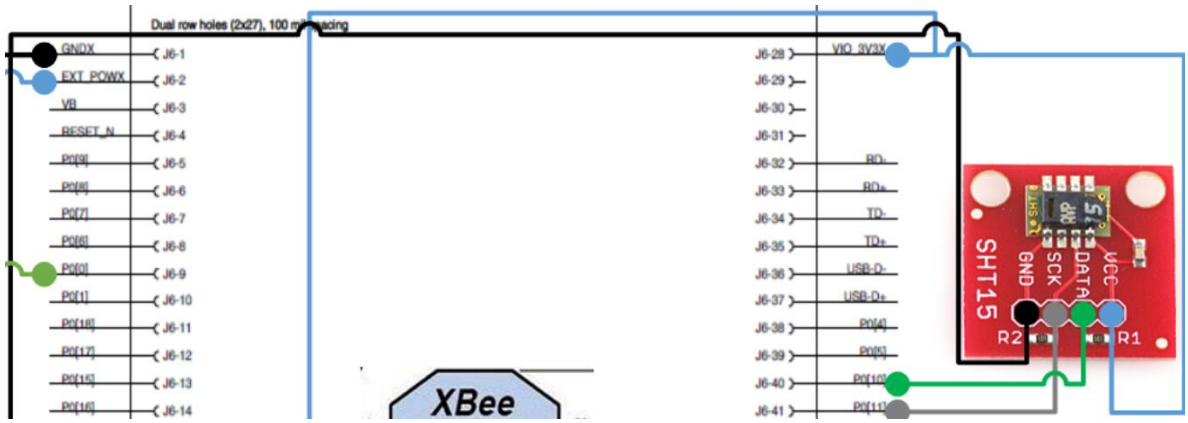


Figura 32 Conexiones SHT15 - LPC1769

Pin LPC1769	Pin SHT15
VIO 3V3X J6-28	+3,3V
P0[10] J6-40	DATA
P0[11] J6-41	SCK
GNDX J6-1	Gnd

Para enviar un comando al sensor tendremos que enviar una secuencia de datos con el pin DATA que nos marcará el comando que queremos ejecutar y con la señal CLK marcaremos el ritmo al que enviamos y recibimos datos para así sincronizar el sensor y el código que se está ejecutando en el LPC1769.

En la siguiente figura encontramos un ejemplo de cómo enviar el comando de leer humedad y cómo recibiremos la respuesta desde el sensor. Cada comando se corresponde a un código de 8 bits por lo que un comando de medir temperatura (por ejemplo) daría unas señales similares.

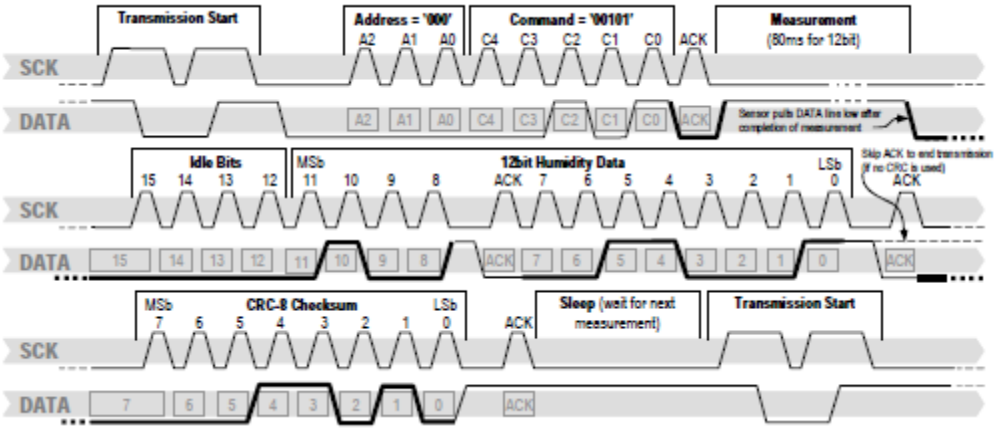


Figura 33 Señales de ejemplo para el comando Medir Humedad

Como se puede ver lo primero que debemos hacer es indicarle al sensor que vamos a iniciar una transmisión. Esto despertará al sensor, ya que se arranca en modo idle, y nos permitirá enviar seguidamente un comando que puede ser:

- Medir Temperatura
- Medir Humedad Relativa
- Leer el Registro de Estado
- Escribir el Registro de Estado
- Soft Reset

En nuestro proyecto sólo se utilizarán los dos primeros comandos que corresponden a medir la temperatura y humedad. Esto es así porque los valores por defecto del SHT15 nos sirven perfectamente y no es necesario hacer ninguna modificación de sus registros de estado.

El Soft Reset reinicia el interfaz y pone los registros de estado al valor por defecto. Este comando tampoco sería necesario.

Tras el envío del comando deberemos esperar a que el sensor realice la medición (entre 50 y 80 ms). Cuando la medición ha terminado el sensor nos avisará poniendo el pin que usamos para DATA a 0, previamente para aceptarnos el comando nos ha subido a 1 el valor de este pin. Tras el envío de la medición podremos pedirle al sensor, opcionalmente, que nos envíe además un CRC para comprobar que se ha leído el valor correctamente. En este proyecto no lo hemos utilizado por no añadir una complejidad innecesaria al alcance del proyecto.

La biblioteca antes de ser utilizada debe de ser inicializada para poder configurar los pines correctamente.

Para hacer uso de esta librería deberemos incluir el archivo “sht1x.h” que nos permitirá usar las tres funciones siguientes para poder comunicarnos con el SHT15:

- **SHT1X_Initialize**: Inicializa el soporte para el sensor en el LPC1769 configurando los pines que utilizaremos para conectarnos a él. Estos pines se los pasaremos por parámetro a esta función.
- **SHT1X_ReadTemperature**: Lee la temperatura del sensor y la devuelve en la unidad que le pasamos por parámetro. Esta función soporta devolver la temperatura en Celsius, Fahrenheit y Kelvin.
- **SHT1X_ReadRelativeHumidity**: Pide al sensor la humedad relativa y usando la temperatura pasada realiza un ajuste mejor de ella. Debido a este parámetro que hay que

pasarle a la función no podemos leer los datos en paralelo y tenemos que limitarnos a leerlos en serie, ya que la humedad relativa depende de la temperatura.

Para facilitar la implementación de los comandos se ha implementado una función común que envía el comando usando la señal DATA y SCK. Esta función es interna por lo tanto no está disponible al usuario.

4.6 LED

Esta biblioteca se encarga del control de los leds de la placa. Cada uno de estos estará conectado a un pin GPIO por el ánodo y a la toma de tierra común del breadboard por el cátodo.

Hemos usado 4 leds para dar diferente información.

- Led verde → Midiendo datos de temperatura y humedad. Se enciende cuando la tarea que interactua con el sensor está en funcionamiento. Este led tiene el ánodo conectado a P0[9] (J6-5).
- Led naranja → Alarma de temperatura. Este led se ilumina cuando la última medición de temperatura ha dado un valor que no se situa entre el máximo y mínimo de humedad indicados en la configuración obtenida desde el servidor. Este led se apagará cuando la última temperatura medida est dentro de los valores seguros. Este led tiene el ánodo conectado a P0[8] (J6-6).
- Led amarillo → Alarma de humedad. Este led se ilumina cuando la última medición de humedad relativa ha dado un valor fuera de los rangos de humedad relativa segura. Se apagará cuando la humedad relativa vuelva a estar en valores seguros. Este led tiene el ánodo conectado a P0[7] (J6-7).
- Led rojo → LPC1769 ocupado, se apaga cuando entra en estado idle, que además llevará al microcontrolador a sleep. Este led tiene el ánodo conectado a P0[6] (J6-8).

Estos leds son muy útiles para ver en un primer vistazo el estado del sistema.

A continuación podemos ver el fragmento del diagrama de conexiones que se corresponde a los leds.

Figura 34 Conexiones Leds - LPC1769

Pin LPC1769	Led
P0[9] J6-5	Verde
P0[8] J6-6	Naranja
P0[7] J6-7	Amarillo
P0[6] J6-8	Rojo

Para utilizar la librería LED debemos de incluir el archivo de cabecera “led.h”. Esta biblioteca se usa inicializándola antes de poder utilizarla, pasándole una máscara de los bits que tienen que ser inicializados.

Una vez la biblioteca ha sido inicializada podremos encender o apagar los leds cuando sea necesario.

La función que nos da la funcionalidad de modificar los leds permite también como la de inicialización pasar una máscara de leds sobre la que queremos operar para facilitar un poco el código o por si hay condiciones en las que todos los leds deben de estar apagados.

4.7 Bajo consumo

Mención especial merece uno de los requisitos más importantes que se puso al proyecto, este era estudiar el uso de cómo poder reducir en lo posible el consumo del sistema.

Para ello los microcontroladores LPC17xx tienen 4 modos para reducir el consumo, los comentaremos a continuación.

En la siguiente figura vemos un diagrama de flujo de cómo funciona el planificador de tareas cuando una tarea se queda a la espera durante un tiempo o porque está esperando que un mutex sea liberado.

Figura 35 Diagrama de flujo bajo consumo

Hay que hacer mención también a que cuando la tarea de envío de datos no recibe datos durante un tiempo determinado pone el WiFly en modo sleep. Y cuando lo vuelve a necesitar se despierta automáticamente al intentar entrar el modo comandos del WiFly.

El consumo medido gracias a un multímetro en modo sleep ronda la mitad del consumo del modo normal.

4.6.1 Sleep mode

En este modo, la ejecución de las instrucciones es suspendida (el reloj del controlador es suspendido) hasta que se recibe un Reset u ocurre una interrupción cualquiera. Los periféricos continúan funcionando durante este modo, con lo cual pueden generar interrupciones que reanudarían la ejecución

Este modo ha sido el que se ha utilizado para este proyecto, aunque no es el que más reduce el consumo ha servido para comprender el funcionamiento de los diferentes modos de consumo.

4.6.2 Deep-sleep mode

Cuando el chip entra en este modo, el oscilador principal es apagado, y prácticamente todos los relojes son parados. El IRC continúa corriendo y puede ser configurado para llevar al timer del Watchdog, permitiendo a este despertar a la CPU. El oscilador RTC de 32 kHz tampoco es parado y las interrupciones RTC también podrían ser usadas para despertar al chip. La memoría flash se deja en modo standby para permitir un despertar rápido.

Dentro de este modo de consumo es menor que en el *sleep* normal. Pero también despertar de este modo es más costoso debido a que hay que restaurar más periféricos.

4.6.3 Power-down mode

Este modo hace todo lo que hace el modo anterior, pero también apaga la memoria flash. Con lo cual se ahorra aún más energía. Pero en el momento de despertar requiere esperar más para restaurar la memoria flash antes de poder ejecutar el código o acceder a los datos. Cuando el chip entra en este modo el IRC, el oscilador principal y todos los relojes son parados. El RTC sigue funcionando si ha sido activado y las interrupciones RTC podrían ser usadas para despertar la CPU.

Este modo consume aún menos que el Deep-sleep a costa de un arranque más lento y de apagar más componentes.

4.6.4 Deep power-down mode

En este último modo el chip es apagado por completo con la excepción del RTC, el pin de Reset, el WIC y los registros de backup del RTC. Para optimizar la conservación de energía, el usuario tiene la opción adicional de apagar el oscilador de 32 kHz. También es posible usar circuitería externa para apagar el regulador a través de los pines $V_{DD(REG)(3V3)}$ después de entrar en este modo. La alimentación tiene que ser devuelta antes de poder reiniciar.

Despertarse de este modo puede ocurrir cuando una señal de reset externa es aplicada, o la interrupción del RTC está activa y esa interrupción ha sido generada.

5. Viabilidad técnica

Con este proyecto se ha demostrado lo polivalente de los dispositivos empotrados, con unos conocimientos medios de programación y unas nociones de electrónica básica se han podido crear un dispositivo de gran utilidad.

Aun así se han detectado al final del desarrollo varios puntos que podrían haberse mejorado y que hubiese creado un dispositivo más sólido.

Como conclusión podemos afirmar que aunque quizá tenga sus deficiencias el proyecto ha sido muy útil para aprender a desenvolverse en este entorno creando diferentes drivers para conectar a ciertos componentes del sistema además de entender cómo sacar partido a un sistema operativo en tiempo real como el FreeRTOS que se ha usado durante el desarrollo.

Ahora enumeraremos los puntos fuertes y débiles encontrados durante todo el diseño, desarrollo y prueba del prototipo.

5.1 Puntos fuertes del proyecto

- Flexibilidad del sistema, si fuese necesario podríamos añadir varios sensores más dando un dispositivo más versátil aún.
- Demuestra la flexibilidad que tienen los entornos webs. Aunque la aplicación web que se ha desarrollado es bastante limitada, las posibilidades son incontables.
- Es 100% Open Source. El código se puede redistribuir sin ningún problema, lo cual podría ayudar a otros desarrolladores a crear nuevas soluciones basadas en nuestro trabajo.
- Este tipo de dispositivos son muy usados a día de hoy lo que nos garantiza que podremos encontrar ayuda en Internet sin mayor problema.
- Es sencillo y potente. Pese a que hace uso de un sistema operativo en tiempo real la dificultad del código no es muy elevada, pero permite la realización de tareas complejas.

5.2 Puntos débiles del proyecto

- Necesita conexión a la red para funcionar. Sería interesante tener un display donde mostrar los valores actuales de temperatura y humedad relativa para en caso de no tener conexión a la red el dispositivo no sea completamente inútil. Además aportaría más información a la alarma de leds ya que actualmente no sabemos si la temperatura (o humedad) es demasiado alta o demasiado baja.
- El consumo de energía no es del todo óptimo. Este es un punto flaco importante del proyecto y que podría dificultar su viabilidad.
- Se podrían mejorar las alarmas poniendo algún tipo de alarma sonora, ya que con dos leds apenas llamaría la atención en una situación crítica.
- Baja seguridad de la aplicación web.

7. Valoración económica

Tras el estudio de la viabilidad técnica, ahora haremos un análisis aproximado de la viabilidad económica del desarrollo del prototipo.

A continuación se expondrá una tabla que resume el coste aproximado y orientativo del proyecto.

Nombre	Precio unitario	Cantidad	Total
LPC1769-LPCXpresso Board	38,55€	1	38,55€
RN-XV WIFLY 802.11 B/G	31,95€	1	31,95€
Adaptador DIP XBee	2,50€	1	2,50€
USB 2.0 to Serial TTL CP2102 Chipset	10,11 €	1	10,11 €
Sensor de humedad y temperatura SHT15	33,95€	1	33,95€
Cables de conexión hembra-hembra	0,33€	15	4,88€
Leds de colores	0,22€	4	0,88€
Multimetro BEST DT9205A	19,46€	1	19,46€
Logic16 USB Logic Analyzer	226,65€	1	226,65€
Horas de desarrollo	12€	250	3.000€
Instalación	50€	1	50€
Puesta en marcha	50€	1	50€
Mantenimiento anual	500€	1	500€
		Coste total	**3.968,93€**

Cabe destacar que el mayor coste viene dado por las horas de desarrollo, pero derivan sobre todo por el desconocimiento de las tecnologías a utilizar. De todos modos el coste en caso de trabajar en un nuevo proyecto de la misma área sería mucho menor al reducirse las horas de desarrollo.

Tampoco se ha añadido el WiFly de recambio que se ha tenido que comprar debido a que el proporcionado junto con el material se estropeó durante las pruebas.

En caso de que el producto se comercializase, habría que tener en cuenta que los costes se reducirían enormemente ya que los costes unitarios serían menores, además de que los costes de desarrollo sólo se aplicarían la primera vez. Si nos planteásemos la comercialización quizá podríamos usar elementos con un menor coste, como por ejemplo una versión inferior a la mota al no necesitar todos los periféricos que trae al que hemos usado.

8. Conclusiones

El prototipo cumple con los objetivos propuestos. Se ha logrado completar la funcionalidad necesaria, a nivel de software tras superar los problemas iniciales.

Con un poco más de información en la primera etapa del proyecto se hubiesen evitado muchos problemas. El WiFly al principio fue muy problemático. Se hubiesen podido haber evitado adquiriendo el adaptador DIP XBee desde el principio.

8.1 Propuesta de mejoras

La principal propuesta de mejora como se comentó en la viabilidad técnica es el poder usar el dispositivo sin requerir conexión a un servidor. Para esto sería necesario incluir en un futuro prototipo un display para mostrar los datos actuales.

Otra mejora importante sería la de añadir más sensores para crear un dispositivo de medición más completo.

Y como también he comentado anteriormente la posibilidad de conectarle alarmas sonoras además de las visuales.

La parte de servidor web también tiene mucho margen de mejora y aquí son prácticamente ilimitadas. Al principio del proyecto se planteó el objetivo de publicar las mediciones a través de una cuenta de twitter, al final fue desestimado por no poner en peligro la entrega a tiempo del proyecto. De cara al futuro podría ser buena idea planteárselo.

8. 2 Autoevaluación

He encontrado este proyecto muy interesante. Aunque al principio me costó el adaptarme a estas nuevas tecnologías, pero una vez fueron comprendidas y se consiguió cierta soltura el desarrollo fue mucho más rápido. Llegando a alcanzar una velocidad bastante notable. Hubiese sido interesante tener más tiempo para estudiar e investigar los diferentes elementos con más calma y así asentar mejor los conocimientos que se ha ido adquiriendo durante el trabajo fin de carrera.

Gracias a él he llenado cierto vacío que habían dejado las asignaturas cursadas anteriormente donde no se ha tenido muy en cuenta los dispositivos empotrados o incluso el lenguaje C. Que

personalmente opino que es uno de los lenguajes más importantes que existe. Nos da una flexibilidad como pocos lenguajes. Nos permite desarrollar para un amplio abanico de ámbitos, desde aplicaciones de escritorio, hasta programas para dispositivos empotrados, pasando por el desarrollo de utilidades de utilidades para el sistema operativo.

Volviendo al tema del proyecto, tengo planeado seguir mejorándolo e intentando aplicarle las posibles mejoras encontradas y comentadas en puntos anteriores. Esto será un trabajo personal en el que podré investigar nuevas tipos de sensores, como conectar una pantalla, etc.

Por último comentar que durante el transcurso de este proyecto he ido aprendiendo mucho y que independientemente del resultado encuentro este TFC como una experiencia muy satisfactoria. Después de este proyecto estoy deseando crear más dispositivos empotrados.

9. Glosario

802.11 b/g	Es un estándar de modulación para redes Wireless. Es la evolución del 802.11g.
ARM	Arquitectura RISC (Reduced Instruction Set Computer) de 32 bits desarollada por ARM Holdings.
Arp@Lab	Aplicación web desarrollada para enviar datos a Internet y poder comprobar el correcto comportamiento de los sistemas creados.
Baud Rate	Número de unidades de señal por segundo. Se mide en baudios y un baudio puede contener varios bits.
Breadboard	También llamada protoboard, es un tablero con orificios conectados eléctricamente entre sí siguiendo patrones. Se usa para facilitar las conexiones en los prototipos de dispositivos electrónicos.
C	Lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del lenguaje B.
CLOCK	En electrónica y especialmente en circuitos digitales asíncronos, la señal de reloj es un tipo particular de señal que oscila entre unos valores y es utilizada para marcar el ritmo de envío de datos.
Cortex-M3	Procesador de ARM de 32bits que es utilizado en numerosas aplicaciones en tiempo real.
DATA	Es la señal por la que se envían o reciben los datos, puede estar marcada por la señal Clock.
Diagrama de Gantt	Es una herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades.
DIP	Un tipo de encapsulado de circuitos integrados electrónicos.
FreeRTOS	Siglas de Free Real Time Operating System. Es un Sistema operativo en tiempo real gratuito que soporta multitud de arquitecturas.
GPIO	Siglas de General Purpose Input/Output, es un pin genérico cuyo comportamiento puede ser controlado por el usuario en tiempo de ejecución.
HTML	Siglas de HyperText Markup Language, es un lenguaje de marcado para desarrollar páginas web.

Java	Lenguaje de programación originalmente desarrollado por James Gosling para Sun Microsystems.
JTAG	Sistema que permite el testing y programación de sistemas empotrados.
Led	Diodo que emite luz.
Logic16	Analizador de señales USB para depurar las señales que se emiten entre dos dispositivos electrónicos.
LPC1769	Es un microcontrolador Cortex-M3 que puede ser usado para aplicaciones en dispositivos empotrados con un alto nivel de integración y un consumo de energía bajo.
Microcontrolador	Es un circuito integrado programable capaz de ejecutar las órdenes grabadas en su memoria.
Mota	Placa con procesador y transmisión/recepción vía radio.
Multímetro	También denominado polímetro, tester, o multitester es un instrumento electrónico portátil para medir directamente magnitudes eléctricas.
Putty	Programa de terminal para Windows que permite conectar a puertos COM o por SSH, entre otros.
RTC	Siglas de Real Time Clock, es un reloj de ordenador que permite programar alarmas.
SHT15	Sensor de temperatura y humedad relativa digital creado por la compañía Sensirion AG.
UART	Siglas de Universal Asynchronous Receiver/Transmitter, es un chip que se encarga de manejar las interrupciones de los dispositivos conectados al puerto serie, convertirlos a datos en formato paralelo y transimitirlos al bus del sistema.
USB	Siglas de Universal Serial Bus, es un estándar industrial desarrollado en los años 90 que define cables, conectores y protocolos usados en un bus.
WiFi	Mecanismo de conexión de dispositivos electrónicos de forma inalámbrica.
WiFly	Componente que capacita a sistemas empotrados a conectarse a redes Wireless compatibles con 802.11b/g.
Wireless	Es un tipo de transferencia de datos donde no se utilizan cables para conectar diferentes ordeandores.

XBee	Nombre de un módulo creado por la compañía Digi International para la conexión punto a punto de dispositivos.

10. Bibliografía

- Wiki de la asignatura Sistemas Empotrados de la UOC:
  http://cv.uoc.edu/app/mediawiki14/wiki/IniciCortexM3
- Arp@Lab:
  http://arpalab.appspot.com/
- FreeRTOS:
  http://www.freertos.org/
- Foro NXP:
  http://forums.nxp.com/
- Datasheet LPC1769/68/67/66/65/64/63:
  http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf
- WiFly User Manual and Command Reference:
  http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/WiFi/WiFly-RN-UM.pdf
- Datasheet SHT1x:
  https://www.sparkfun.com/datasheets/Sensors/SHT1x_datasheet.pdf

11. Anexos

11.1 Manual del usuario

11.1.1 Medidor online de temperatura y humedad de bajo consumo

Para instalar utilizar este medidor simplemente tenemos que enchufar el cable USB dado a un ordenador o a un adaptador para conectar a la corriente. También se podría conectar a una pila que diese la suficiente potencia para alimentar el circuito. Una vez conectado los cuatros leds que hay sobre la placa parpadearán durante dos segundos indicándonos que el dispositivo comenzará a funcionar.

11.1.2 Servidor web

Para acceder al servidor web tendremos que dirigirnos a la dirección:

http://tfcjefernandez.appspot.com/

Una vez dentro encontraremos el interfaz de usuario que podemos ver a continuación.

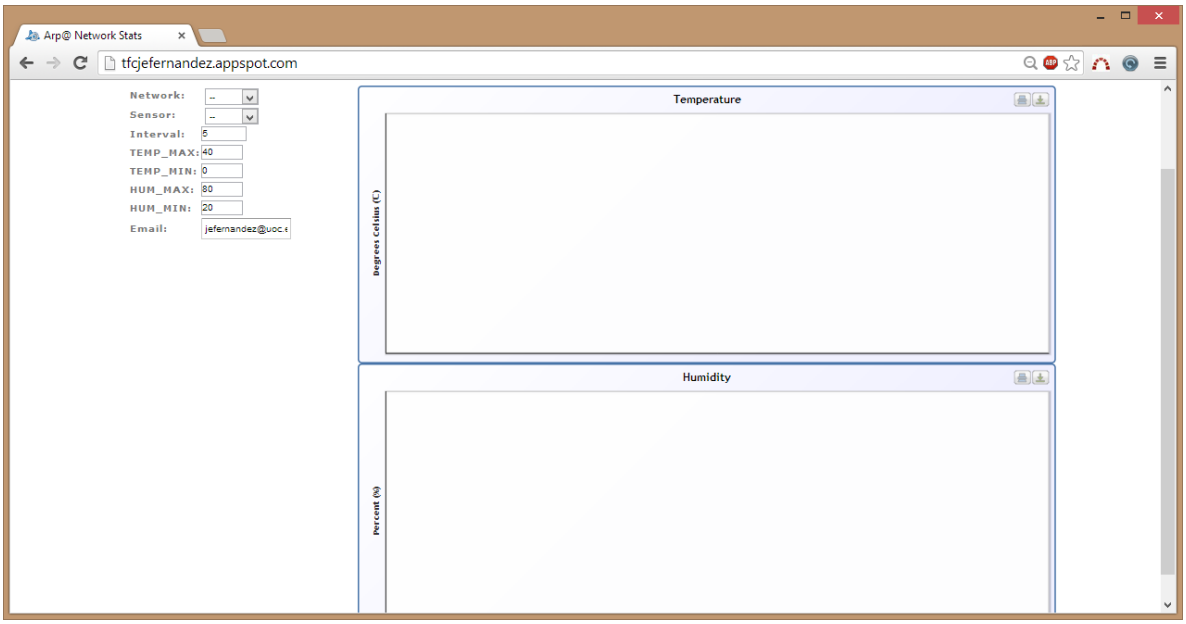


Figura 36 Interfaz web

Desde aquí podremos seleccionar la red a la que nos queremos conectar.

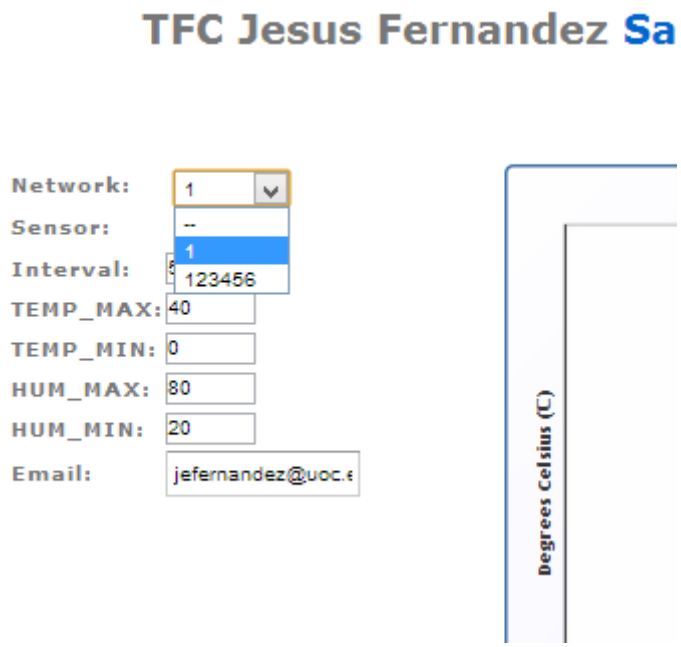


Figura 37 Selección de red

Con lo que nos saldrán las diferentes ips que están enviando datos al servidor.

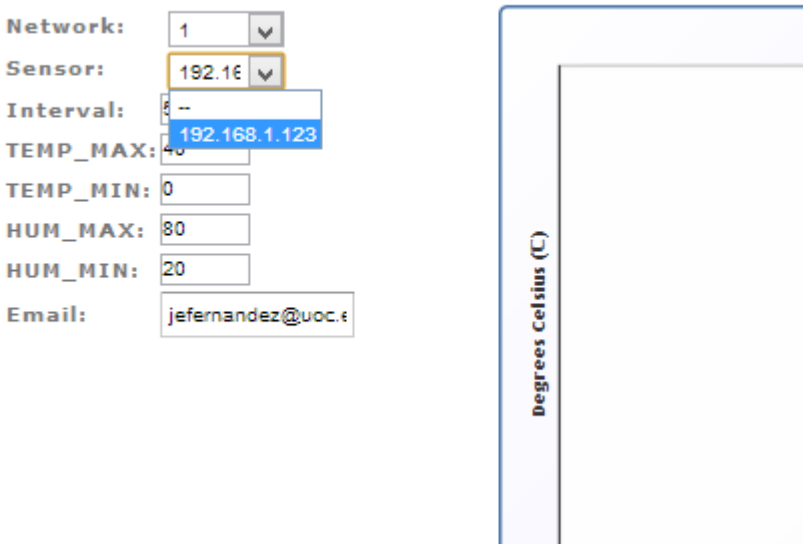


Figura 38 Selección de IP

Una vez seleccionada la red que buscamos y la IP del sensor que queremos monitorizar automáticamente se pintarán las gráficas como se ve a continuación.

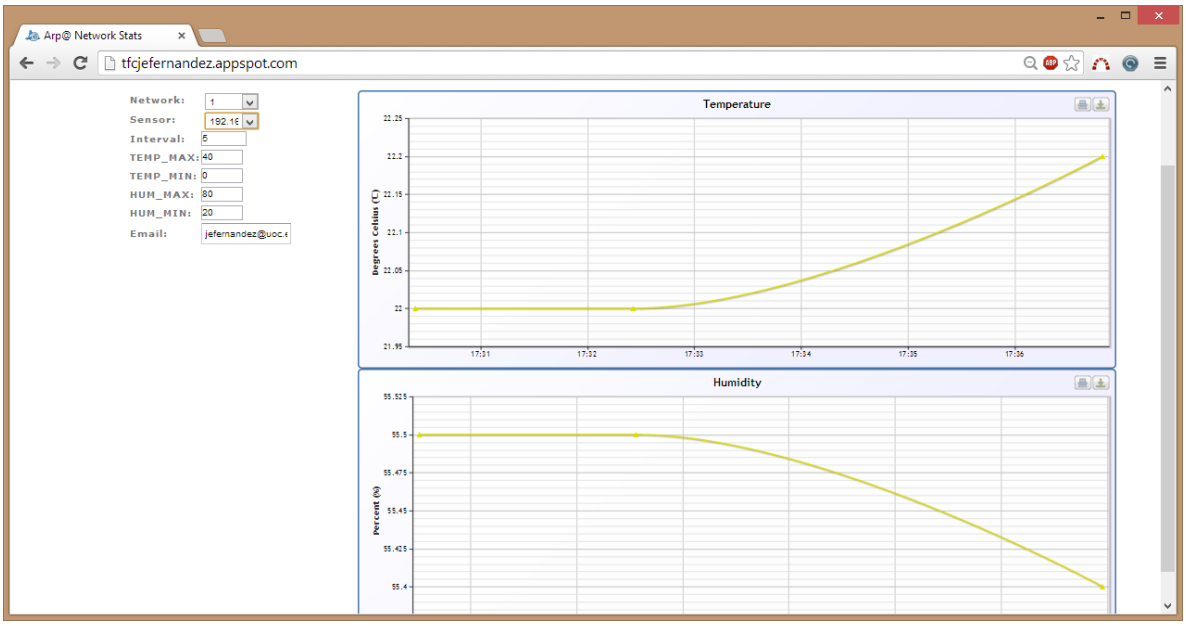


Figura 39 Monitorizando un sensor

A partir de ahora podremos dejar esa página cargada donde nos irán apareciendo actualizaciones de los datos enviados o podremos configurar las alarmas, intervalo de tiempo y dirección de correo electrónico donde se enviarán las notificaciones.

11.2 Cómo compilar el código con LPCXpresso

1. Para poder compilar el proyecto tendremos que tener el LPCXpresso que podremos descargar desde la url (previo registro): http://lpcxpresso.code-red-tech.com/LPCXpresso/
2. Una vez descargado, instalado, abierto y con un nuevo workspace seleccionado procederemos a importar el Zip con el código fuente descargado. Para ello desde el QuickStart Panel haremos click en la opción “Import archived projects (zip)” dentro de la

sección “Import and Export”.

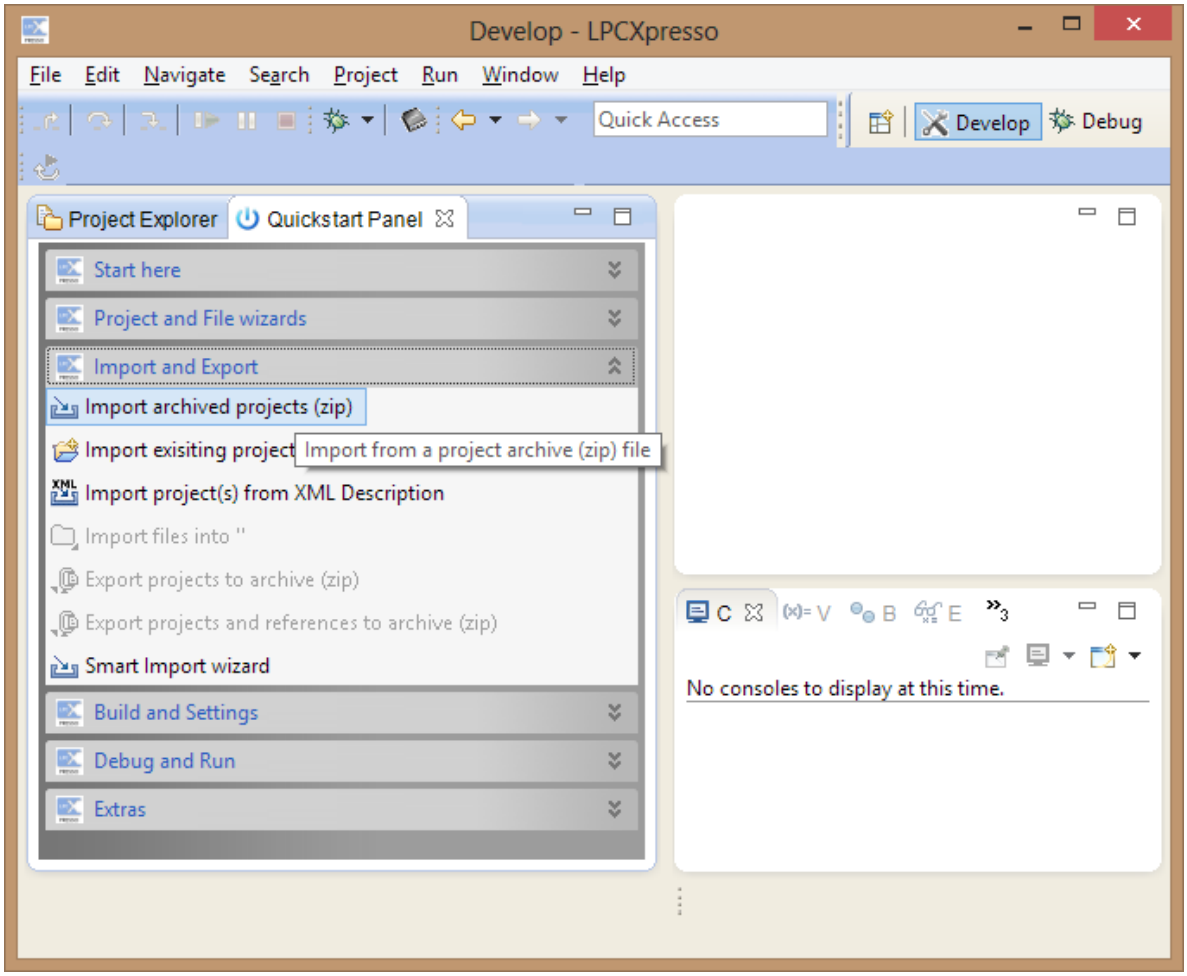


Figura 40 Seleccionando la opción de importar zip

3. Buscaremos el archivo Zip descargado:

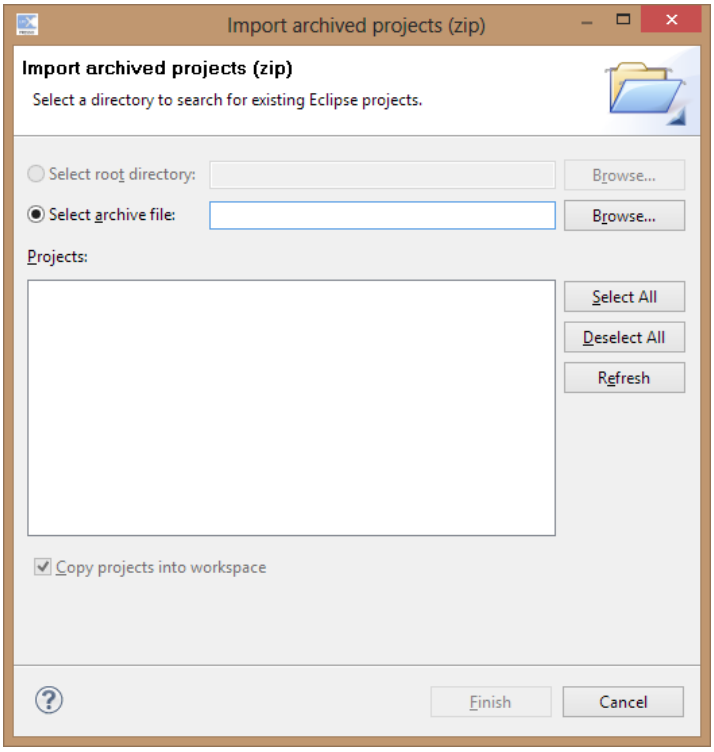


Figura 41 Diálogo de importar zip

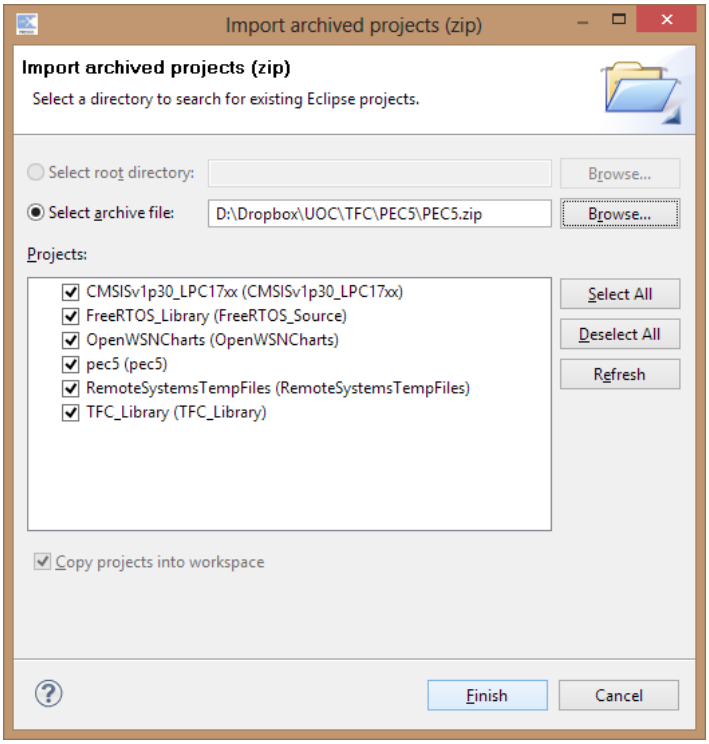


Figura 42 Zip seleccionado

4. Dejaremos seleccionados todos los proyectos excepto “OpenWSNChars” y serán los que importaremos a nuestro nuevo workspace. Finalmente pulsaremo el botón “Finish”.

5. Entonces ya tendremos los proyectos cargados y podremos ver el código de cada uno de los archivos.

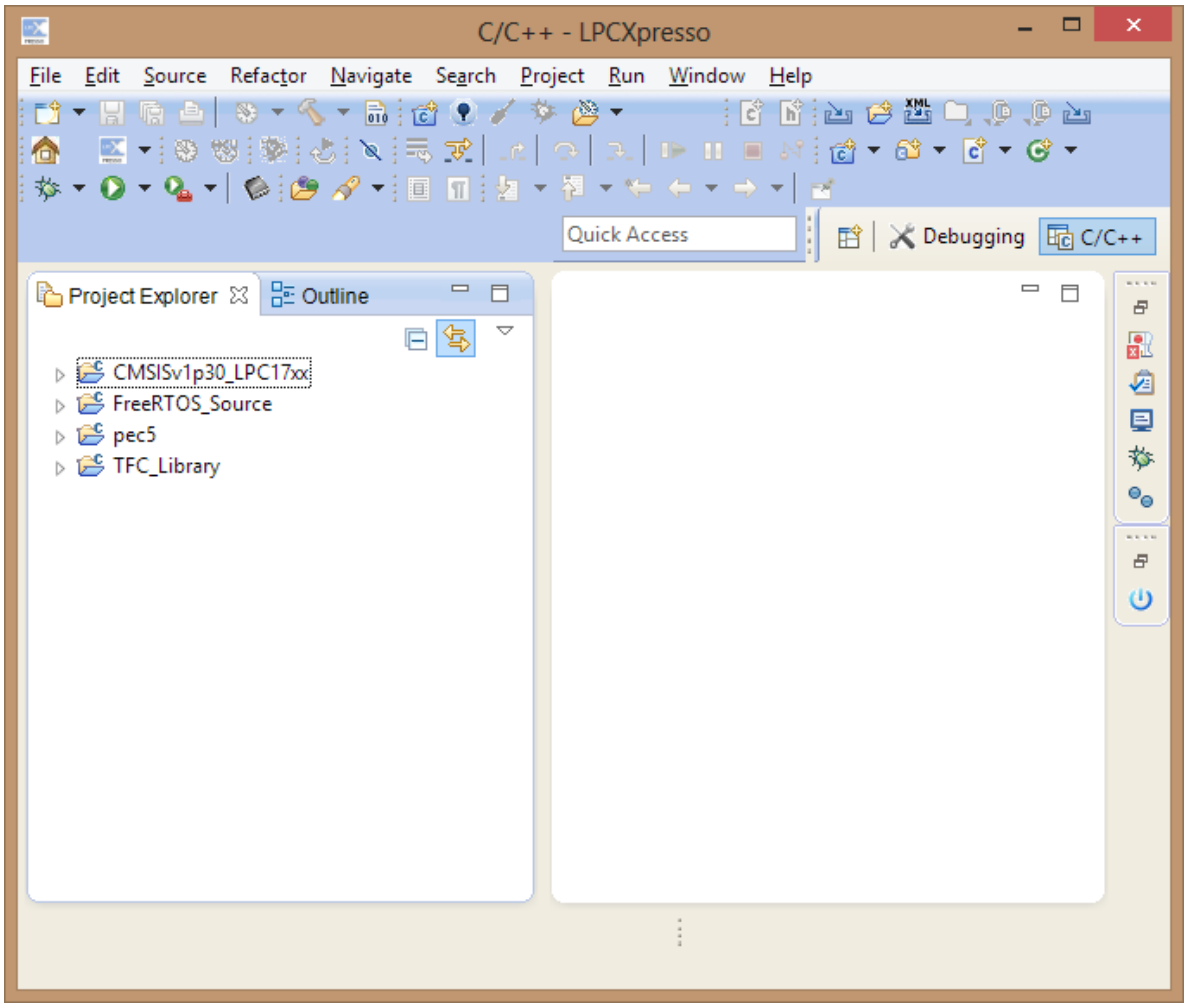


Figura 43 IDE con los proyectos cargados

6. Una vez llegado aquí seleccionaremos el proyecto pec5 y usando la opción del menú Project\Build Project el proyecto comenzará a compilarse.

11.3 Cargar el código en la memoría Flash de nuestro LPC1769

Una vez hayamos compilado el proyecto como se indicaba en el punto anterior podremos pasar a escribir nuestro código en la mota para probarlo.

Para ello:

1. Seleccionaremos el botón “Program Flash” de la barra de herramientas.

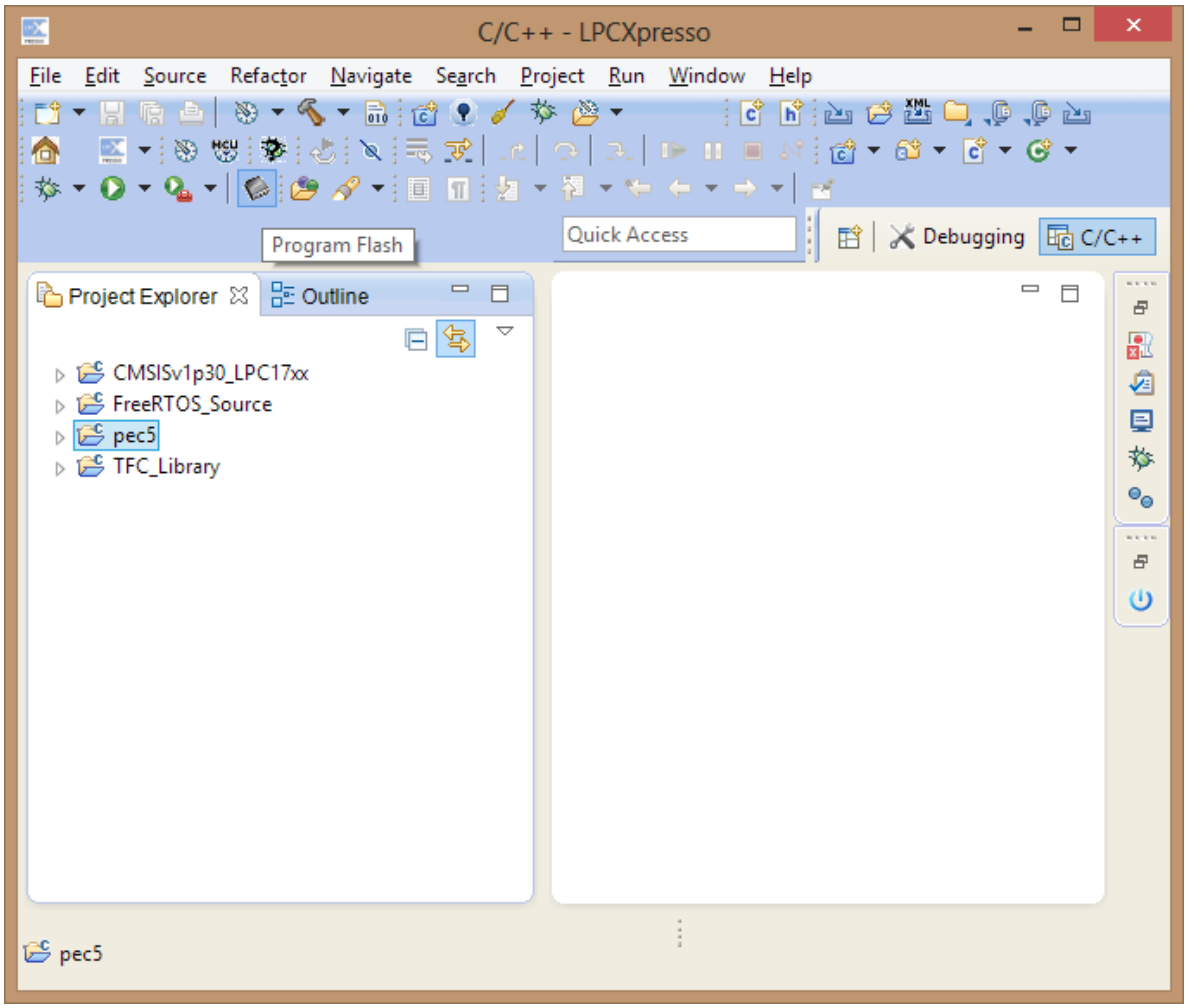


Figura 44 Program flash

2. Y nos saldrá una ventana informándonos de que está buscando el hardware, deberemos tener el LPC1769 conectado por JTAG al ordenador. Si pasa el tiempo veremos la

siguiente pantalla.

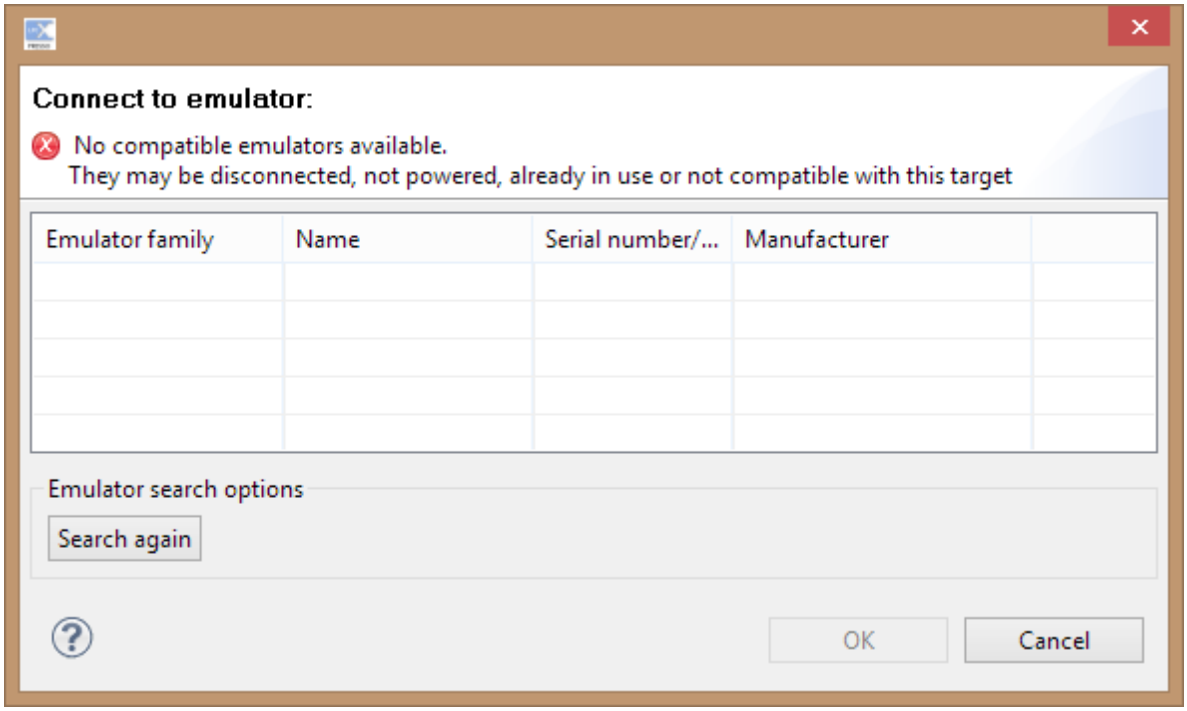


Figura 45 Buscando dispositivo

3. Donde cuando lo hayamos conectado podremos darle a “Search again” y nos encontrará el hardware y estaremos listos para cargar nuestro código en la memoria flash.

4. Buscaremos el archivo de extensión “axf” que queremos cargar, en nuestro caso el pec5.axf y estaremos listos para hacer flash al LPC1769.

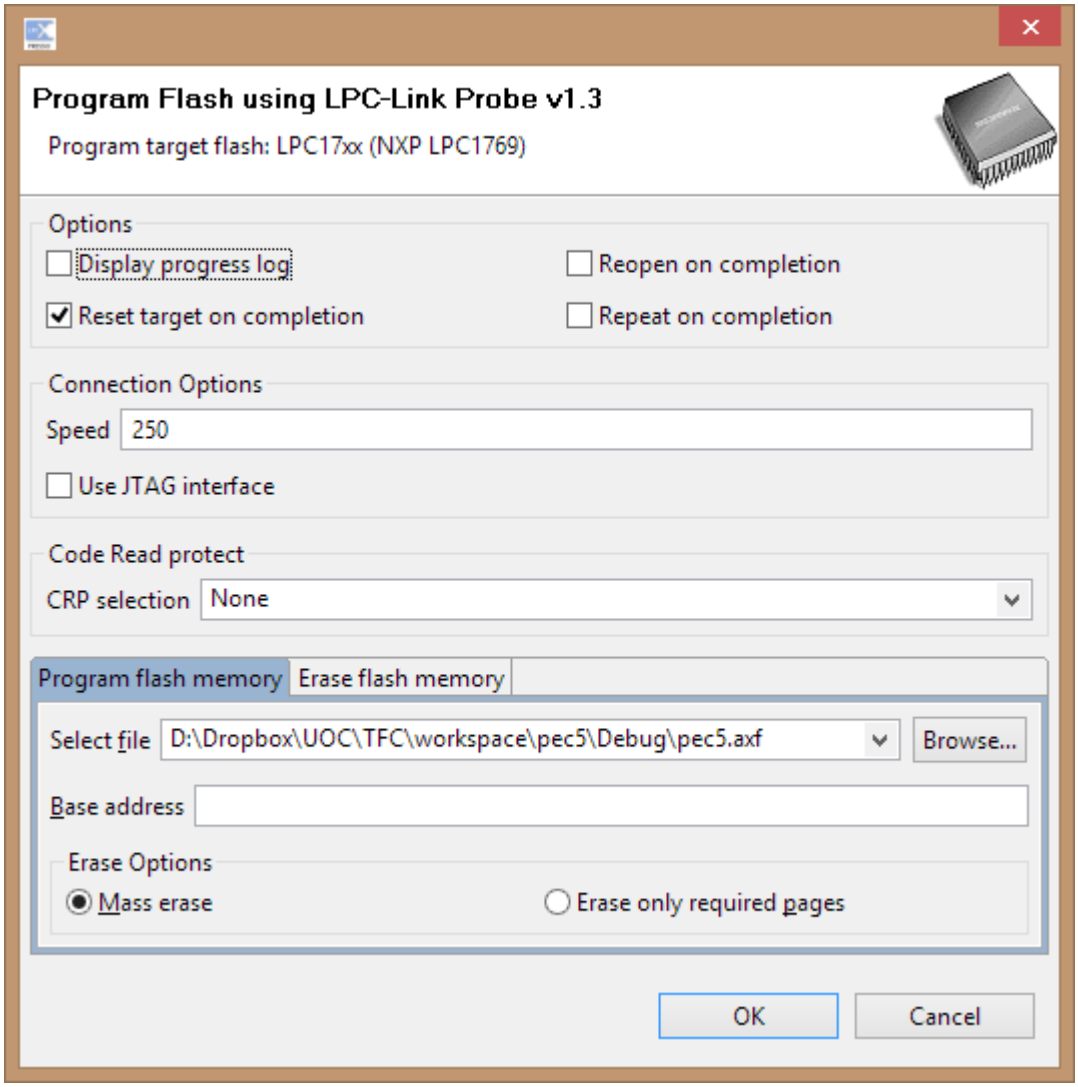


Figura 46 Diálogo de Program flash

5. Pulsaremos el botón OK y se pondrá a flashear la memoría. Tras unos instantes tendremos el dispositivo listo para ser probado.