INGENIERÍA DE SISTEMAS ELECTRÓNICOS

Curso 2019-2020
BLOQUE 1

Ignacio Blasco Hernández
Jorge Rosales Martín
GRUPO V1V2V3 - PUESTO 3

Departamento de Ingeniería Telemática y Electrónica
Universidad Politécnica de Madrid

Índice del documento

1 OBJETIVOS DEL BLOQUE

1.1 Resumen de los objetivos del bloque

En este primer bloque los objetivos generales que se han tratado están relacionados con la programación avanzada de microcontroladores en sistemas empotrados. Los objetivos que se han realizado son:
-Manejo de protocolos de comunicaciones como por ejemplo IP o HTTP, para la comunicación del microcontrolador con un servidor vía ethernet
-Uso de periféricos para la conversión de datos, como por ejemplo el uso del conversor analógico digital.
-Gestión avanzada de las interrupciones, como por ejemplo en la interrupción del joystick o del Watchdog
-Técnicas de temporización avanzadas con el uso de un servidor SNTP complementado con el reloj RTC parar tener actualizada la hora incluso cuando el microcontrolador no está conectado.
- Caracterización del rendimiento de la aplicación mediante el uso del Watchdog para poder conocer si se ha producido alguna excepción o fallo no deseado.
-Modificación de la memoria flash del micro a través de la IAP

1.2 Resumen de los objetivos de las prácticas realizadas

A continuación, describiremos qué objetivos se han alcanzado en cada practica y cómo se han alcanzado dichos objetivos.

1.2.1 Objetivos de la práctica 1

El objetivo de la práctica 1 es construir un servidor web embebido en el LPC1768, de manera que desde una página web se puedan controlar y monitorizar distintos recursos del microprocesador.

Primeramente, se ha comprobado el encendido y apagado de los Leds desde la página web, después se ha realizado la escritura de la pantalla del microprocesador desde la página web y por último se ha monitorizado el conversor A/D, pudiendo observar en la página web el estado éste.

Para todo lo descrito anteriormente se han alcanzado los objetivos de manejo de protocolos de comunicaciones. Ya que para poder comunicar el microcontrolador y sus recursos con la página web han utilizado los protocolos de comunicaciones IP, TCP, HTTP y ETHERNET de las distintas capas del modelo OSI.

También se ha cumplido con el objetivo del uso de periféricos para la conversión de datos como por ejemplo a la hora de convertir la tensión proporcionada por el potenciómetro

1.2.2 Objetivos de la práctica 2

El objetivo de la práctica 2 es añadir a los resultados de la práctica anterior un sistema horario controlado tanto por el servidor web como por el hardware del LPC1768.
Lo primero que hemo realizado en esta práctica ha sido configurar el RTC y mostrar tanto la hora como la fecha por pantalla. Además, añadiendo una batería de 3,3V conseguimos que la hora no dejase de contar cuando el microcontrolador no tuviera alimentación.
El segundo, ha sido obtener una referencia horaria mediante el uso del servicio SNTP para sincronizar la hora y la fecha del RTC. Además, se añadido la posibilidad de poder reiniciar la hora y la fecha mediante la pulsación de un botón.
Por último, se ha añadido la funcionalidad de poder visualizar el estado del RTC en el servidor y establecer una hora y fechas concretas desde el servidor web, actualizando por consiguiente el RTC.

Por lo tanto, los objetivos generales que se han cumplido en esta práctica han sido el uso de técnicas de temporización avanzadas con el uso de un servidor SNTP complementado con el reloj RTC parar tener actualizada la hora incluso cuando el microcontrolador no está conectado.

1.2.3 Objetivos de la práctica 3

El objetivo de la tercera practica ha sido comprender como funciona el watchdog y de que manera se puede leer y escribir la memoria flash del LPC1768.

En la primera parte de la práctica se ha comprendido cual es el modo de funcionamiento y la utilidad del watchdog y se ha implementado a los resultados de la práctica 2.
En la segunda parte, se ha analizado cómo funcionan las escrituras y lecturas en la memoria flash a través del IAP mediante un ejemplo. También se ha implementado en la práctica 2, consiguiendo almacenar el estado de los leds y demás información relevante de las prácticas anteriores en la memoria.
Además, en ambas partes se ha incluido el uso del Led RGB con distintos usos, en la primera práctica se ha utilizado para conocer cómo se reseteo la última vez el microprocesador. En el segunda se ha utilizado para mostrar el resultado de una comparación entre un valor del ADC y el resultado de lectura de una posición de la memoria flash.
Por ello, se han alcanzado los objetivos de caracterización del rendimiento de la aplicación mediante el uso del Watchdog, gestión avanzada de interrupciones a la hora de manejar el Watchdog y la modificación de la memoria flash del micro a través de la IAP.

1.3 Acrónimos utilizados

Identifique los acrónimos usados en su documento.

RTC	Real Time Clock
SNTP	Simple Network Time Protocol
IAP	In Application Program
IP	Internet Protocol
TCP	Transmission Control Protocol
HTTP	HiperText Transfer Protocol
OSI	Open Systems Interconnection
A/D	Conversor Analógico Digital

1.4 Tiempo empleado en la realización de cada una de las prácticas.

El tiempo aproximado que se ha utilizado para la realización de la práctica es aproximadamente de 60 horas por alumno, que corresponden tanto al trabajo individual como al trabajo grupal.
De las 60 horas, de media, 20 horas corresponden a cada práctica realizada. De estas 20 horas por práctica, al menos 12 horas han sido de trabajo individual y 8 de trabajo colaborativo.

El trabajo colaborativo ha consistido en la puesta en común del avance individual de cada uno, resolver las dudas que hayan podido surgir, plantear una pauta con la que avanzar de manera individual (sobre como avanzar) y avanzar en la práctica.
Para trabajo individual se ha utilizado la herramienta Teams de Microsoft para almacenar los ficheros de la práctica, e ir actualizando dichos ficheros según se avanza.
Se ha utilizado este método de trabajo por motivos de compatibilidad de horario y por los efectos causados por el Covid-19. Además, este método de trabajo ha sido posible gracias a que ambos disponemos del microprocesador y de la placa de expansión.

1.5 Bibliografía utilizada

[RD1] Manual del Lpc1768: https://www.nxp.com/docs/en/user-guide/UM10360.pdf

[RD2] Documentación de CMSIS: https://arm-software.github.io/CMSIS_5/Core/html/index.html

[RD3] Ejemplos de la placa MCB1000

[RD4] Tutoriales para el LPC1768:
https://www.exploreembedded.com/wiki/index.php?search=LPC1768&title=Special%3ASearch

[RD5] Esquemático del LPC1768:
https://os.mbed.com/media/uploads/chris/lpc1768-refdesign-schematic.pdf

[RD6] Esquemáticos de la placa de expansión:
https://os.mbed.com/media/uploads/chris/mbed-014.1_b.pdf

[RD7] Make-Unix-timestamp-c de Mark Solters (mktime.c y mktime.h)
https://github.com/msolters/make-unix-timestamp-c

1.6 Autoevaluación.

A continuación, procedemos a la autoevaluación personal de las competencias que hemos adquirido durante la realización de este primer bloque de la asignatura.

Autoevaluación estudiante A: Ignacio Blasco Hernández
Personalmente creo que he alcanzado algunos de los objetivos de aprendizaje que se indican en la guía, creo que soy capaz de realizar un proyecto cumpliendo con las especificaciones requeridas, y de elaborar documentación técnica. Además, he mejorado mis habilidades de aprendizaje autónomo y de filtrado y búsqueda de documentación. Por otro lado, soy capaz de emplear las herramientas de desarrollo y depuración (keil) sin ningún problema.
Por último, he tenido dificultades con la migración de RTOSv1 a RTOSv2, aunque los proyectos no lo requerían tengo dificultades en la compresión de la sincronización de hilos mediante el uso de señales (esperar a varias señales y evaluar cual ha llegado, por ejemplo).

Autoevaluación estudiante B: Jorge Rosales Martín

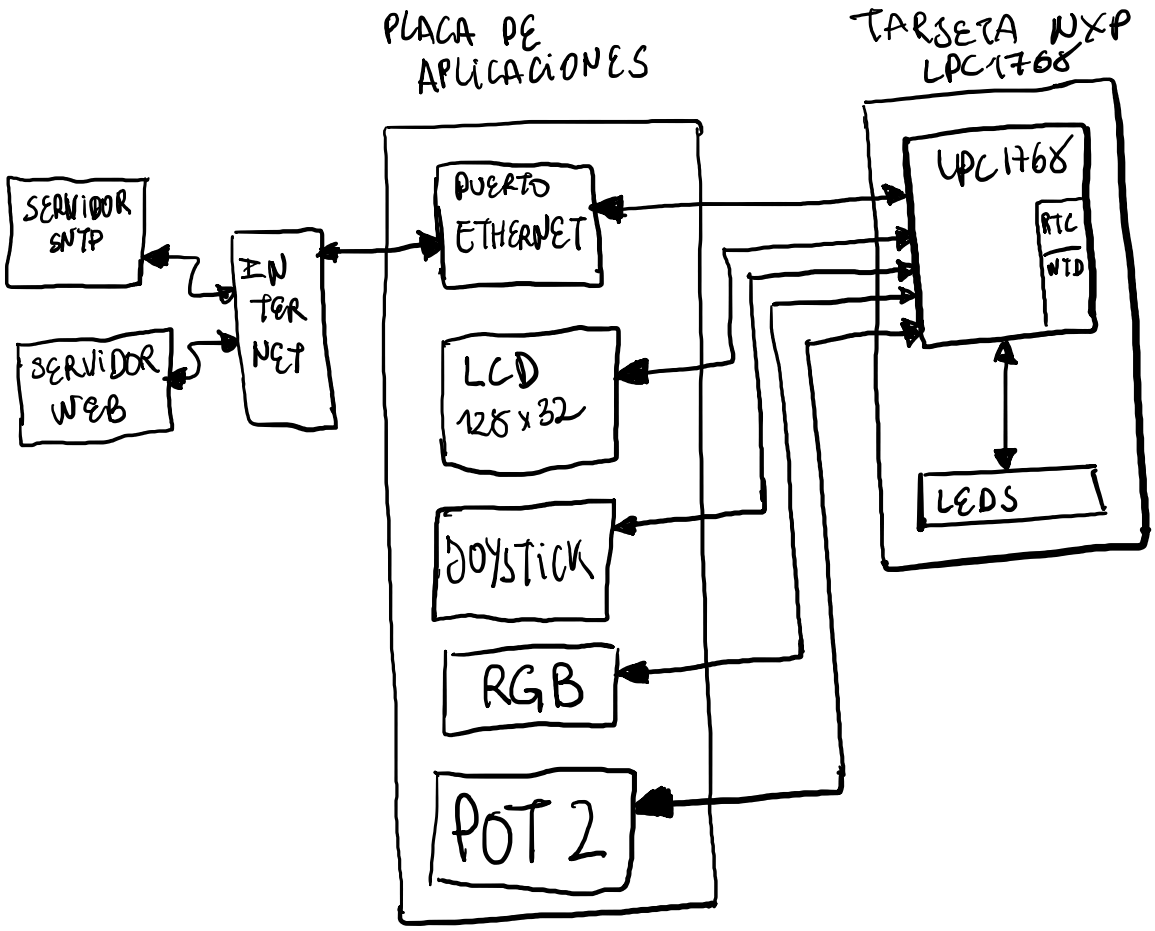
Durante la realización de este bloque, las competencias y resultados de aprendizaje propuestos en la guía que he adquirido son: la capacidad para buscar, razonar y seleccionar la información necesaria para el proyecto dentro de los documentos correspondientes de nuestra área, la habilidad de utilizar las tecnologías de la información y de las comunicaciones para poder realizar el proyecto junto a mi compañero de manera telemática debido a la situación de cuarentena en la que lo hemos tenido que realizar y por último la capacidad de desarrollar un sistema electrónico de mediana complejidad utilizando los diferentes conocimientos adquiridos a lo largo de la carrera.

Las mayores complejidades que he tenido a la hora de realizar este bloque han tenido que ver mas por la parte de problemas en el Keil y en la propia tarjeta, ya que me era imposible realizar el reseteo de la tarjeta y con ello probar el funcionamiento del Watchdog.

2 RECURSOS UTILIZADOS DEL MICROCONTROLADOR

2.1 Diagrama de bloques hardware del sistema completo (práctica 3).

Este apartado debe contener un diagrama de bloques donde se identifiquen claramente los elementos utilizados en el desarrollo del sistema completo. Debe elaborar una figura que muestre todos los elementos utilizados de la tarjeta NXP LPC1768 y su interconexión con los elementos externos (tarjeta de aplicaciones, sensores, etc.).



2.2 Justificación de las soluciones adoptadas.

De la tarjeta NCP LPC1768 se han utilizado los siguientes recursos:

- Para poder dotar de sistema operativo, conexión ethernet y pines de entrada salida es necesario activar las siguientes opciones en el menú Manage Run-Time Enviorenment.
  - CMSIS->CORE
  - CMSIS->RTOS2(API)-> Keil RTX5
  - CMSIS Driver->Ethernet MAC (API) -> Ethernet Mac
  - CMSIS Driver->Ethernet PHY (API) -> DP83838C (Driver Ethernet del LPC1768)
  - CMSIS Driver->SPI->SSP

- DEVICE->GPDMA
- DEVICE->GPIO
- DEVICE->PIN
- DEVICE->Startup
- NETWORK-> CORE
- NETWORK->SERVICE->SNTP
- NETWORK->SERVICE -> WEB SERVER COMPACT
- NETWORK->SOCKET-> TCP
- NETWORK->SOCKET->UDP

- Los 4 leds, configurados como pines de salida.
- El watchdog, configurado en modo interrupción para el hito 1 y como reset para el hito 2, se ha seleccionado el PCLK como fuente para el clk del watchdog. Para calcular los 5 segundos, se ha utilizado la siguiente fórmula

$$LPC_{WDT} \rightarrow WDTC = 5 * SystemCoreClock/16$$

  Se divide entre 16 dado que el watchdog tiene un prescaler preconfigurado de ¼ y el PLCK predeterminado de ¼.
- El RTC se ha configurado para generar una interrupción cada segundo, poniendo el registro LPC_RTC->CIIR =0x01, esto produce que cada vez que se incrementa en 1 la cuenta de los segundos, se genera una interrupción: Por otro lado, se ha configurado una alarma cada minuto, es decir, cada vez que los segundos pasan a 0, se activa la alarma, esto se consigue con la siguientes líneas; LPC_RTC-> AMR =0xFE; (Para que solo compare los segundos) y PC_RTC->ALSEC=0 (valor de los segundos a comparar).
- Es necesario aumentar el número de hilos que vienen por defecto ya que se utilizan más de los indicados, para esto, en el archivo RTX_config.h, se ha marcado el checkbox Thread Configuration->Object specific memory allocation y aumentar el número de Number of user thread lo que sea necesario, en nuestro caso, 11 (no se usan todos, se pone 11 por comodidad). También puede resultar necesario aumentar la memoria predeterminada.
- Por otro lado, para configurar la conexión a internet, es necesario configurar en el archivo RTE_Device.h seleccionando el checkbox ENET(Ethernet Media Interface), y desplegando el menú, seleccionar en MIIM (Management Data Interface)->ENET_MDC Pin el P1_16 y en MIIM (Management Data Interface)->ENET_MDIO Pin el P1_17. A continuación, en el archivo Net_Config_ETH_0.h indicar una dirección Mac, IPv4->IP address, IPv4->subnet mask válidos. Y deshabilitar la función Dynamic Host Configuration Protocol, para utilizar una dirección ip fija. Conviene destacar que a la hora de conectarlo al router puede ser de utilidad marcar esta opción, para que el router le asigne una dirección valida, y, con una aplicación (fing, por ejemplo) que escanee las direcciones ip que hay conectadas a la red, muestre la que asigna al Lpc1768.
- Sntp (No es un componente físico, pero por simplicidad lo tomamos como un periférico), es necesario configurar una dirección ip válida para que devuelva la petición. Esto se puede hacer o bien en el archivo Net_Config_SNTP_Client.h, escribiendo la dirección Ip, o escribiendo la siguiente sentencia:

  ```
  const NET_ADDR4 ntp_server = {NET_ADDR_IP4, 0, 130, 206, 3, 166};
  ```

- Memoria Flash, se ha definido un tamaño de buffer de 256 bytes, y se ha elegido el sector 29 para almacenar los datos necesario. Esta decisión viene de que es el último sector, y asegura que no se va a machacar ningún dato importante, dejando colgada la aplicación. Para ejecutar las operaciones de borrado y escritura se ha utilizado el fichero lpc_17xx_iap que contiene las funciones necesarias para ejecutar dichas operaciones. Por último, para leer datos, se ha utilizado la siguiente función:

  ```
  memcpy (read, (void *) FLASH_PROG_AREA_START, numeroBytes);
  ```

Guarda en un array read el resultado de la operación, especificada por la dirección desde la que se lee y el número de bytes a leer.

De la tarjeta de aplicaciones se han utilizado los siguientes recursos:

- LCD, ADC y Joystick, se ha reciclado de la asignatura SBM el código referente a éstos.
- RGB, se han configurado los 3 pines de salida, cabe destacar que son activos a nivel bajo, es decir, cuando se escribe un 1 en el pin verde, por ejemplo, se enciende, y si se escribe un 0, se apaga.

3 SOFTWARE

La estructura de la aplicación se podría dividir en cuatro partes:

- Main.c, encargado de la inicialización del de la aplicación, en concreto inicializa el kernel de SO, los leds y el adc, además de iniciar un hilo (app_main) que dará paso al resto del programa. Si bien es cierto que la inicialización de los leds y del adc se podría haber hecho más adelante, se ha puesto ahí por simplicidad.
- HTTP_Server.c, contiene hilos encargados con la comunicación de periféricos.
- HTTP_Server_CGI.c, fichero encargado de leer y escribir valores en la página web.
- El resto de los ficheros son de periféricos (lcd.c, lcd.h, adc.c, adc.h, led.c, led. h, …) que contienen funciones de inicialización, configuración, lectura y escritura, en resumen, funciones que permiten utilizar dichos periféricos.

Las tareas creadas se encuentran principalmente en el archivo HTTP_Server.c, de las cuales, la principal app_main() ya que inicializa el resto de las tareas. Este hilo se llama desde el archivo main.c una vez que se ha inicializado el core del sistema operativo. Además, actualiza el estado de los leds leyendo de la memoria flash, para que se inicien en el mismo estado que estaban antes del reset/apagado.
Por otro lado, en orden según el número de línea encontramos las siguientes tareas:

- ➔ Display(), encargado de actualizar lo que se muestra en el Lcd, tras la creación del hilo, se inicializa el lcd y se obtienen tanto la dirección IP y la Mac para almacénalas en la memoria Flash (Dirección 0x78000), después se escribe en la pantalla la dirección IP y se mantiene el hilo pausado a la espera de que el usuario actualice el valor desde la web.
- ➔ BlinkLed(), encargado de cambiar el valor de los leds cuando se encuentran en el estado Ledrun=true, este cambio de estado se produce cada 200ms.
- ➔ Rtc_setTime() y Rtc_setdate(), permiten cambiar la hora y la fecha cada vez que les llega una señal. Esta señal se envía desde HTTP_serverCGI cada vez que el usuario actualiza la hora desde el servidor.
- ➔ SW_Thread(), que la única función que realiza es eliminar los rebotes en el joystick. Dicho hilo espera a una señal enviada desde el Eint3_irqHandler cada vez que detecta una pulsación.
- ➔ Timer_SNTP_callback(), es un timer periódico de 3 minutos que envía una señal a SNTP_Thread para que actualice la hora.
- ➔ SNTP_thread, encargado de obtener la hora (llamando a get_time), espera hasta que el timer le envíe una señal, y ejecuta get_time(), y espera a que el callback del SNTP.c le avise de que ha terminado, pasado este si el estado de LEDrun=false enciende y apaga el led 3.
- ➔ Por último, en este fichero encontramos una serie de funciones auxiliares que permiten la lectura y escritura y borrado del sector 29 de la memoria flash. Y una función auxiliar que permite convertir un carácter numérico (char) en entero (int) utilizado para convertir la hora.

Toda la sincronización se hace el envío de señales a hilos, de los cuales:

- ➔ Señal enviada a Display(), se hace desde HTTP_ServerCGI (petición POST) cada vez que el usuario envía la orden desde el servidor de actualizar los valores de la pantalla.
- ➔ Señal enviada a Rtc_setTime(), se hace desde HTTP_ServerCGI (petición POST) cada vez que el usuario envía la orden desde el servidor de actualizar el valor de la hora.
- ➔ Señal enviada a Rtc_setDate(), se hace desde HTTP_ServerCGI (petición POST) cada vez que el usuario envía la orden desde el servidor de actualizar el valor de la fecha.
- ➔ Señal enviada a Rtc_setDate(), se hace desde HTTP_ServerCGI (petición POST) cada vez que el usuario envía la orden desde el servidor de actualizar el valor de la fecha.
- ➔ Señal enviada a SW_thread(), se hace desde switch.c->EINT3_IRQHandler(), cada vez que se recibe una interrupción del pulsador central del joystick.
- ➔ Señal enviada a SNTP_thread(), se envía una primera señal desde el timer periódico de 3 minutos y después se envía otra desde SNTP.c-> time_callback(), una vez ha actualizado el valor de fecha y hora.

➔ Por otro lado, se han utilizado sentencias como la siguiente para simular timeOuts que permitan continuar con la ejecución: osThreadFlagsWait (0x80U, osFlagsWaitAny, timpo_ms)
Lo que hace es que se espera a que se envíe la señal 0x80 (que nadie va a enviar) durante un periodo de tiempo, pasado este tiempo, continua le ejecución. Sin embargo, mientras espera, está permitida la ejecución de otros hilos, de modo que no es bloqueante.

Se ha utilizado, como se comentó anteriormente para actualizar la hora mediante el servicio SNTP cada 3 minutos.

En cuanto al uso de semáforos, somos conscientes de que la escritura en memoria es algo delicado, pero y debería de hacerse con sumo cuidado (empleando semáforos), sin embargo, no hemos detectado comportamientos anómalos (dado que solo se hace una escritura al comienzo) y hemos decidido no implementarlos.

Por otro lado, se ha habilitado las interrupciones del RTC para que generen una interrupción por segundo para que se actualice la hora, y se ha habilitado la interrupción EINT3_IRQhandler para poder interactuar con el joystick.

Por último, para esta última práctica se ha decidido organizar el proyecto en dos partes, código principal y periféricos.

➔ El código principal, engloba los ficheros main.c, HTTP_Server.c, cuya funcionalidad se explicó anteriormente y el fichero HTTP_ServerCGI.c, que es el intermediario entre el servidor web (pagina) y el código, analizando las peticiones recibidas mediante Post (actualizar los leds, la pantalla, la hora, …) y enviar peticiones (estado inicial de la pantalla, valor del adc, valor de la hora, …).

➔ Los periféricos, engloba los ficheros con funcionalidades para hacer uso del hardware entre otros.

Encontramos los siguientes ficheros:

- Lcd.c y Lcd.h, librería que permite que se escriban caracteres en la pantalla de la placa de aplicaciones. Se ha utilizado la librería hecha en la asignatura Sistemas Basados en Microprocesadores, y se ha incluido la función void actualizar (char lcd_text [2][21]), que escribe en las dos líneas de la pantalla los datos necesarios.
- Adc.c y Adc.h, librería que permite obtener el estado del potenciómetro de la placa de aplicaciones mediante una conversión analógico digital. Se ha utilizado la librería hecha en la asignatura Sistemas Basados en Microprocesadores.
- Led.c y Led.h, librería que permite actualizar el estado de los leds del LPC1768. Se ha utilizado la librería que ofrece keil para el MCB1000 (Manage Run-Time Enviorenment->Board Support->LED) dado que es simple e intuitiva, y recogía la funcionalidad requerida, además, se ha modificado para los pines del LPC1768.
- Rtc.c y Rtc.h, esta librería recoge las funciones necesarias de configuración, escritura y lectura del real RTC.
- Sntp.c y Sntp.h, librería que permite obtener los segundos de un servidor SNTP y actualizar el RTC según los datos recibidos.
- Switch.c y Switch.h, librería que gestiona las pulsaciones recibidas por el Joystick de la placa de aplicaciones, se ha utilizado la librería hecha en la asignatura Sistemas Basados en Microprocesadores.
- Mktime.c y Mktime.h, es una librería que permite obtener el total de segundos transcurridos desde el 1-1-1970 (Tiempo POSIX). Se utiliza para convertir los valores del RTC a segundos totales y pasarlos al servidor web en una sola transferencia (en vez de pasar minutos, segundos, horas, día, mes y año en varias). Esta librería ha sido escrita por Mark Solters y es se ha encontrado en GitHub, el enlace está disponible en la bibliografía.
- Wtdg.c y Wtdg.h, librería que contiene las funciones necesarias para la inicialización, configuración, feed, … para monitorizar la ejecución de programa y actuar en consecuencia. (Se utiliza en la primera parte de la práctica 3.)

- Rgb.c y Rgb.h, librería que permite actualizar el estado (color) del RGB situado en la placa de aplicaciones.
- Lpc17xx_iap_c y Lpc17xx_iap.h, librería obtenida del directorio: lpc175x_6x_cmsis_driver_library_0.zip, contiene las funciones necesarias para borrar y escribir en la memoria flash del LPC1768. Por otro lado, en el archivo HTTP_server.c se incluyen una serie de funciones que permiten escribir distintos valores en la memoria flash. Se podrían haber incluido perfectamente en el Lpc17xx_iap, pero se ha decidido, por simplicidad, dejarlas en el HTTP_server.c.

3.1 Descripción de los proyectos de Keil de cada una de las prácticas.

Proyecto	*Funcionalidad*	*Origen*
Práctica 1	Servidor WEB compacto adaptado al LPC1768 Permite gestionar remotamente el comportamiento de distintos elementos (LEDS, ADC; LCD)	Parte de los ejemplos de Keil para la placa MCB1000 (HTTP_server, etc)
Práctica 2	Añade a la funcionalidad de la práctica 1, un sistema horario controlado tanto por hardware como por software. Por hardware permite ponerlo a "0" y por software, permite actualizar la hora cada tres minutos mediante el servicio SNTP y además permite tanto mostrar como modificar la hora y la fecha en una nueva pestaña del servidor web.	Parte de la práctica 1.
Práctica 3 (H1, H2 y H3)	Añade a la funcionalidad de la práctica 2 la capacidad de detectar comportamientos anómalos y actuar en consecuencia reiniciando el microprocesador.	Parte de la práctica 2
Práctica 3 (H4 y H5)	Conjunto de vario programas sencillos donde cada uno tiene una funcionalidad. Se pueden resumir en las siguientes: Borrar un sector de memoria. Escribir las 16 primeras posiciones de un sector. Leer las 16 primeras posiciones de un sector. Leer las 16 primeras posiciones, cambiar una y escribirla.	Parte del directorio: lpc175x_6x_cmsis_driver_library_0\ lpc175x_6x_cmsis_driver_library\ Examples\IAP
Práctica 3 (H6 y H7)	Añade a la funcionalidad de la práctica 2 la capacidad de almacenar la dirección ip y mac en la memoria flash al inicio del programa, de almacenar el estado de los leds cada vez que se cambia en el servidor y de leer, el último de estado de los leds tras un reset y de leer un valor de la memoria concreto para encender o apagar el RGB.	Parte de la práctica 2 y del hito 5 de la práctica 3.

4 DEPURACION Y TEST

En este apartado debe indicar el mecanismo que ha utilizado para realizar las pruebas que demuestren el correcto funcionamiento de cada una de las prácticas.

4.1 Plan de pruebas del sistema.

A continuación, vamos a realizar una planificación de las pruebas que deberíamos realizar para poder validar todas las funcionalidades realizadas por el programa de cada práctica.

4.1.1 Plan de pruebas del sistema completo desarrollado en la Práctica 1.

En esta primera práctica hay que comprobar el funcionamiento de la conexión del servidor con el microcontrolador, el correcto funcionamiento de los leds en ambos modos de funcionamiento, la correcta escritura en la pantalla LCD y por último comprobar que la tensión del potenciómetro se muestra en la página web.

Para comprobar la correcta conexión del servidor con el microcontrolador, primero realizamos un ping a la dirección IP que hemos configurado en el microcontrolador. Con esto intentamos comprobar si la página web es accesible vía internet. El resultado esperado es un envío satisfactorio, lo que significa que la página web esta activa, y el microcontrolador está conectado con el servidor web.

En cuanto al funcionamiento de los leds, necesitamos comprobar que podemos manejar su funcionamiento a través de la página web. Para poder comprobarlo realizamos diversas pruebas, primero comprobamos que se encienden y se apagan los 4 leds individualmente, si desde la página web tenemos activada la opción browser y al marcar y desmarcar cada uno de los leds se encienden y apagan, incluso encendiendo dos leds juntos o todos ellos a la vez. También comprobamos que la opción de running lights funciona, para ello al activarlo comprobamos que comienzan a encenderse los leds en secuencia de izquierda a derecha y viceversa.

Para comprobar el funcionamiento de la escritura de la pantalla LCD desde la página web realizamos una escritura en la línea 1 y comprobamos que se escribe todo en la pantalla LCD incluyendo algún carácter especial. Después realizamos la misma prueba para la línea 2 y probamos a escribir en ambas líneas, al realizarlo comprobamos que se escribe perfectamente en ambas líneas todo aquello que escribimos y mandamos desde la página web.

Por último, comprobamos el funcionamiento del conversor A/D. Para ello vamos a realizar varias modificaciones del valor del potenciómetro y comprobamos que debería cambiar el valor en la barra de la página web, estando la barra rellena en un extremo y vacía en el otro.

4.1.2 Plan de pruebas del sistema completo desarrollado en la Práctica 2.

En la segunda práctica tenemos que comprobar que se muestra correctamente la hora y la fecha mediante el empleo del RTC (tanto alimentando el microcontrolador como si no se alimenta) y que se utiliza y funciona correctamente el servicio SNTP para poder referenciar la hora de la ubicación en la que nos encontramos.

Para probar primeramente el uso del RTC mostramos la hora en la pantalla para poder observar que el formato de la hora y la fecha se encuentran correctamente. Probamos también a modificar la hora y fecha y actualizarla a la hora actual. Dejamos el programa corriendo y comprobamos que la hora se mantiene actualizada.

Para comprobar el funcionamiento de la hora a través de RTC cuando el microcontrolador no se encuentra alimentado, probaremos a desconectar el microcontrolador (montando la batería de 3,3V) en distintas situaciones y comprobaremos que la hora al volver a conectar el microcontrolador continúa funcionando y se ha incrementado como debería.

En cuanto a la prueba del servidor SNTP realizaremos una prueba para comprobar que recibimos la hora de la franja horaria adecuada. Para ello mostramos por pantalla la hora y fecha. El resultado debería ser la hora y fecha actual en la que se ha mostrado por pantalla.
También tenemos que comprobar que sincronizamos la hora a través de SNTP al arrancar el microprocesador. Por ello comprobamos si al iniciar el microcontrolador obtenemos la hora y fecha adecuada mostrándola por. El resultado debería ser que al arrancar el programa nos mostrara la hora y fecha actualizada de la localización.
Por último, tenemos que comprobar que la hora se actualiza cada 3 minutos. Para comprobar esto modificamos la sincronización para que ocurra cada minuto, y dado que tenemos configurado el botón central para iniciar la hora, después de que se haya iniciado el programa y sincronizado por primera vez la hora, pulsamos el botón central para comprobar que se realiza la sincronización. Repetimos este proceso en varias ocasiones para comprobar que realiza la sincronización periódicamente cada minuto. El resultado debería ser que, después de pulsar el botón de reset de la hora, se mostrase por pantalla la fecha y hora actualizada.

4.1.3 Plan de pruebas del WatchDog desarrollado en la Práctica 3.

En esta primera parte de la práctica 3, el plan de pruebas a consistido en comprobar que el WatchDog funciona de la manera esperada, tanto cuando se configura para que genere una interrupción o un reset.
Ha sido necesario configurar distintas prioridades a la hora de atender las interrupciones para poder maximizar el efecto de los fallos. Además, se han añadido dos ejemplos para cada hito, uno que presenta un problema y otro en el que el problema ha sido solucionado. Plan de pruebas del sistema completo desarrollado en la Práctica 3. En el hito 1, en el problema encontramos que la interrupción generada por el pulsador no se limpia correctamente cuando se han pulsado más de 10 veces, provocando que no se alimente el watchdog y por tanto se produce una interrupción en este. Se observa que los leds 1 y 2 dejan de conmutar y se enciende de manera interrumpida el led 3. En la solución corregida, se limpia el flag correctamente y tras 10 pulsaciones, se continúa alimentando el watchdog y por tanto no se producen interrupciones.
En el hito 2, la diferencia está en que en vez de interrupción se produce un reset. Además, se ha configurado el led RGB para que se encienda durante tres segundos tras iniciar el programa. Se encenderá en rojo si se reseteó por el watchdog y en verde si se reseteó por el botón de la tarjeta o se acaba de alimentar. Dicho esto, se observa claramente que, en el problema, tras diez pulsaciones, el programa se reinicia y por tanto se prende el led rojo del RGB, sin embargo, si se interrumpe la alimentación o se pulsa el botón de reset se enciende en vede. En la solución corregida, solo se observa que el led se prende en verde tras interrumpir la alimentación o al pulsar el botón de reset.
Para esta parte de la entrega hemos detectado un comportamiento anómalo a la hora de depurar el programa. Es necesario pulsar el botón de run dos veces, puesto que se queda parado al inicio por un breakpoint, el cual desconocemos donde se encuentra. En cualquier caso, pulsando el botón run dos veces se soluciona. Si se descarga el programa en la flash, es necesario resetear usando el botón una vez para correr el programa tras cargarlo, hecho esto ya no será necesario pulsarlo más veces.

4.1.4 Plan de pruebas del sistema completo desarrollado en la Práctica 3.

En este apartado comprobamos que las escrituras en la memoria y las lecturas se hacen de manera correcta. Para ello se ha visualizado en la ventana memory 1, los valores que hay en la dirección 0x78000 (sector 29) cada vez que se escriben datos, éstos se cotejan con el array que se manda escribir (buffer) visualizando en la ventana watch 1. Tras cada lectura, se visualiza en la ventana watch 1 el array read y se comprueba con lo almacenado en la dirección que se indica, también en el memory 1. Por otro lado, se ha comprobado que el rgb se enciende según el valor almacenado en la memoria. El resto de las funcionalidades se comprobaron en la práctica 2 y por tanto resulta redundante volver a comprobarlo.

4.1.5 Pruebas realizadas.

Ejecutadas todas las pruebas comentadas en el plan de pruebas, podemos afirmar que el sistema funciona correctamente. A excepción del RTC que dadas las circunstancias causadas por el Covid-19 no hemos podido

encontrar un zócalo adecuado para la pila, para poder comprobar que se sigue contando cuando se quita la alimentación principal. Además, encontramos un comportamiento anómalo, y es que cada vez que se alimenta el circuito, el RTC toma un valor aleatorio, es decir, no mantiene el último dato almacenado tras interrumpir la alimentación.

5 OTROS ASPECTOS

5.1.1 Procedimiento para el envío y la recepción de datos entre el microprocesador y el servidor.

Cuando se desea modificar u obtener el valor de una variable, es necesario marcar en el archivo .cgi, (datetime.cgi por ejemplo) el inicio de la línea con una letra c, para diferenciarlo del resto de código. Después es necesario, identificar que se quiere escribir, para ello se coloca otra letra del abecedario, esta tiene que ser uncia, es decir, no puede haber otro archivo con la misma letra. En este ejemplo se ha puesto una n.

```
t  <th width
t </tr><tr>
t <p align=c
c n 1 <td><i
c n 2  <td><
```

Cuando se envían datos, desde la función netCGI_Script() en el archivo HTTP_ServerCGI.c, se utilizan sentencias como la siguiente:

```
case 'n':
switch (env[2]) {
  case '1':
    len = (uint32_t)sprintf (buf, &env[4], time_text[0]);
    break;
  case '2':
    len = (uint32_t)sprintf (buf, &env[4], time_text[1]);
    break;

}
break;
```

Esto permite actualizar datos en la página web cada vez que el servidor realice una petición, ya sea de manera periódica o por la acción asociada a un botón.

Cuando se reciben datos, desde la función netCGI_ProcessData() en el archivo HTTP_ServerCGI.c, se se utilizan sentencias como la siguiente:

```
else if (strncmp (var, "dset=", 5) == 0) {
  // LCD Module line 2 text
  strcpy (time_text[1], var+5);
  osThreadFlagsSet (TID_Rtc_setDate, 0x20);
}
```

Esto permite obtener el valor de las variables almacenadas en los ficheros .cgi cada vez que se recarga la página o se hace click en un botón de tipo submit. Una vez se reciben estos datos, se avisa al hilo pertinente para actuar en consecuencia. En este caso se actualiza la fecha del rtc con el valor enviado desde el servidor.

5.1.2 Conversión de los segundos en unix timeStamp a horas, minutos, segundos en javascript.

Como se comentó en apartados anteriores, se tomó la decisión de mandar al servidor (datetime.cgi) el valor de los segundos totales, para que, al actualizar el reloj de éste, solo fuese necesario hacer una transferencia por segundo. Para convertir los segundos de unix timestamp a una fecha legible se ha utilizado el siguiente código:

```
var d= new Date();
d.setTime((document.getElementById("time").value-3600)*1000);
var año=d.getFullYear();
var mes=d.getMonth();
document.getElementById("time1").value=""+d.getHours()+":"+d.getMinutes()·
document.getElementById("date").value =""+d.getDate()+"/"+mes+"/"+año;
```

Además, se ha establecido que la hora se refresque cada segundo tal y como se hacía en el ejemplo del adc, con la salvedad de que, en este caso, el checkbox de actualización periódica está marcado de manera predeterminada y además está escondido, para que no se pueda desmarcar. Esto se consigue modificando las propiedades de los botones como se muestra a continuación.

```
<input type=button style="display: none" value="Refresh" onclick="updateMultiple(formUpdate,plotADGraph)">
<input style="display: none" checked  type="checkbox" id="adChkBox" onload="periodicUpdateAd()">
```