|  |  |
| --- | --- |
|  |  |
| **INGENIERÍA DE SISTEMAS ELECTRÓNICOS** |  |
|  |  |
|  | *Curso 2021-2022*  *BLOQUE 2* |
|  | *Martin Sotil Marticorena*  *Diego Rosado Quiros*  M08M09M10 |

Departamento de Ingeniería Telemática y Electrónica

Universidad Politécnica de Madrid

Índice del documento

[1 Objetivos del bloque 2](#_Toc36117465)

[1.1 Resumen de los objetivos del bloque 2](#_Toc36117466)

[1.2 Resumen de los objetivos de las prácticas realizadas 2](#_Toc36117467)

[1.2.1 Objetivos de la práctica 1 2](#_Toc36117468)

[1.2.2 Objetivos de la práctica 2 2](#_Toc36117469)

[1.2.3 Objetivos de la práctica 3 2](#_Toc36117470)

[1.3 Acrónimos utilizados 2](#_Toc36117471)

[1.4 Tiempo empleado en la realización de cada una de las prácticas. 2](#_Toc36117472)

[1.5 Bibliografía utilizada 3](#_Toc36117473)

[1.6 Autoevaluación. 3](#_Toc36117474)

[2 RECURSOS UTILIZADOS DEL MICROCONTROLADOR 4](#_Toc36117475)

[2.1 Diagrama de bloques hardware del sistema completo (práctica 3). 4](#_Toc36117476)

[2.2 Justificación de las soluciones adoptadas. 4](#_Toc36117477)

[3 SOFTWARE 5](#_Toc36117478)

[3.1 Descripción del funcionamiento de la aplicación completa (Práctica 3). 5](#_Toc36117479)

[3.2 Descripción de los proyectos de Keil de cada una de las prácticas. 5](#_Toc36117480)

[4 DEPURACION Y TEST 6](#_Toc36117481)

[4.1 Plan de pruebas del sistema. 6](#_Toc36117482)

[4.1.1 Plan de pruebas del sistema completo desarrollado en la Práctica 1. 6](#_Toc36117483)

[4.1.2 Plan de pruebas del sistema completo desarrollado en la Práctica 2. 6](#_Toc36117484)

[4.1.3 Plan de pruebas del WatchDog desarrollado en la Práctica 3. 6](#_Toc36117485)

[4.1.4 Plan de pruebas del sistema completo desarrollado en la Práctica 3. 6](#_Toc36117486)

[4.2 Pruebas realizadas. 6](#_Toc36117487)

[5 OTROS ASPECTOS 7](#_Toc36117488)

# Objetivos del bloque

T

## Resumen de los objetivos del bloque

En este bloque se busca profundizar y mejorar los conocimientos adquiridos en la asignatura Sistemas Basados en Microprocesador ampliándolos. Entre otros hemos aprendido a manejar la memoria interna del micro (FLASH), el contador descendente Watchdog, Conocimientos de Diseño web con HTML y CGI, así como el funcionamiento de una página web dentro de un microprocesador entre otras cosas.

Para ello hemos contado con el código creado en SBM de control de Botones, leds, pantalla led, ADC…

## Resumen de los objetivos de la práctica realizada

Se deben enumerar los objetivos que ha alcanzado al realizar cada una de las tres prácticas que componen este bloque. Enumérelos de forma precisa y sencilla.

### - Fase 1: conexión mediante el protocolo I2C entre las dos tarjetas

- **Fase 2:** Diseño conceptual del bloque AGP

**- Fase 3:** Desarrollo del servidor Web y software de Unidad de Control (UC)

**- Fase 4:** Implementación del hardware del AGP

**- Fase 5:** Integración final

## Acrónimos utilizados

Identifique los acrónimos usados en su documento.

|  |  |
| --- | --- |
| RTC | Real Time Clock |
| SNTP | Simple Network Time Protocol |
| SBM | Sistemas Basados en Microprocesador |
| ADC | Analogical digital converter |
| RTOS | Sistema Operacional de Tiempo Real |
| CMSIS | Cortex Microcontroller Software Interface Standard |

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

Para realizar las practicas hemos necesitado unas 9 horas semanales fuera de clase para poder finalizarlas.

En total hemos calculado unas 36 horas en total desde el 20 de abril al 15 de mayo.

## Bibliografía utilizada

1. The Designer’s Guide to the Cortex-M Processor Family .pdf
2. [URL]<https://www.keil.com/support/man/docs/rlarm/rlarm_sntp_get_time.htm>
3. Amplificador operacional LM311[URL]

<https://www.alldatasheet.com/datasheetpdf/pdf/197819/SEOUL/LM311.html>

1. Amplificador operacional LM311[URL]

<https://pdf1.alldatasheet.com/datasheet-pdf/view/96584/TI/UA741.html>

1. Multiplexor Digital-analógico 74HC4051 [URL]

<https://pdf1.alldatasheet.com/datasheet-pdf/view/26984/TI/74HC4051.html>

1. Datasheet Lpc-1768 [URL]

<https://www.alldatasheet.com/datasheet-pdf/pdf/275723/PHILIPS/LPC1768.html>

## Autoevaluación.

Hemos tenido dificultades en la creación de la página web, y en el rediseño del hardware para adaptarlo a los inconvenientes que nos íbamos encontrando.

Por lo demás, hemos completado todos los objetivos satisfactoriamente.

# RECURSOS UTILIZADOS DEL MICROCONTROLADOR

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

Diagrama

Descripción generada automáticamente

## Justificación de las soluciones adoptadas.

Los recursos del microcontrolador que se han usado son el RTOS, el bus de comunicación SPI para la pantalla led y del RTC.

El proyecto se va a crear con distintos procesos en tiempo real gracias al RTOS que se configura en el fichero RTX\_conf\_CM.c.

Tanto el bus de comunicación SPI como la interfaz Ethernet se configuran en el fichero de configuración RTE\_device.h además de en el fichero Net\_Config\_ETH\_0.h para el caso de Ethernet.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Por otra parte, usaremos el protocolo I2C para unir los dos microcontroladores bajo una relación maestro-esclavo. Para ello configuraremos los pines en el fichero RTE\_device.h

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Además, para poder comunicar la interrupción que avisa del esclavo al maestro el desbordamiento del overload hemos usado los pines digitales DIP 15 en los dos casos. Para poder controlar el multiplexor desde el esclavo hemos usado los pines DIP 14, DIP 17 y DIP19. Por último, para poder captar las señales del hardware que avisan del desbordamiento del overload al esclavo hemos usado los pines digitales DIP 9 y DIP 10.

1. **Hardware**

## Diagrama de bloques

Diagrama de bloques realizado con la herramienta OrCAD Pspice.

Diagrama, Esquemático

Descripción generada automáticamente

## 3.2. Amplificador de instrumentación

Para la creación del amplificador de instrumentación (A.I) usaremos tres amplificadores operacionales ua741. Para configurar adecuadamente el A.I tenemos que elegir las resistencias R1 y R2 que están involucradas en el rechazo común del A.I. Nuestro objetivo es conseguir un buen rechazo al modo común, y para ello elegimos un valor de 10k ohm que nos aseguran una ganancia de 1 en la segunda etapa del A.I.

CMRR(A:I)=CMRR1+CMRR2

El CMRR de la primera etapa es igual a (G diferencial/ G común)

Para calcular el CMRR de la segunda etapa tenemos que calcular el CMRR de las Resistencias y hacer el paralelo con el del Amplificador Operacional que viene en su datasheet:

CMRR2=(CMRR(R2) // CMRR(A.O)) = (1 + R2/R2)/4\*TOL(resistencia) // CMRR(A.O)

Dado que la fórmula de la ganancia del A.I es G = 1 + 2R1/Rg hemos escogido por ejemplo 25k ohm para R1.

Diagrama, Esquemático

Descripción generada automáticamente

## Multiplexor de resistencia Rg

Para poder elegir las diferentes ganancias necesitamos variar la Rg del amplificador de instrumentación. Para poder hacerlo hemos desarrollado un circuito compuesto por un multiplexor 74HC4051 con 3 entradas de selección digital dando lugar a 8 entradas analógicas entre las que poder elegir.

Diagrama

Descripción generada automáticamente

Las entradas digitales de selección se gobernarán desde el microprocesador permitiéndonos elegir entre diferentes resistencias que incorporaremos a las entradas del multiplexor. Los niveles lógicos del multiplexor dependerán de la alimentación que le demos como se ve en esta tabla sacada del datasheet del multiplexor:

Tabla

Descripción generada automáticamente con confianza media



Si alimentamos con Vcc a 4.5V el nivel de tensión mínimo que acepta el multiplexor como nivel alto es de 3.15 V que es muy justo para el nivel alto de 3.3 V TTL que sale del microcontrolador. Por lo tanto, hemos decidido alimentar el multiplexor desde el microcontrolador a 3.3 V para así tener un nivel de voltaje mínimo para el nivel alto de aproximadamente 2.5 V.

En cuanto a las resistencias que se le van a añadir serán valores que permitan tener ganancias de 1, 5, 10, 50 y 100. Estos valores serán aproximadamente de ∞, 12,5KΩ, 5,55KΩ, 1,02KΩ, 505.5KΩ respectivamente.

* 1. Voltaje de comparación V comp + y V comp -

Para que el microcontrolador pueda fijar un nivel de overload hemos usado el DAC, pero este solo podrá aportar valores entre 0 y 3,3 V por lo tanto necesitaremos un amplificador no inversor de ganancia 3 para poder establecer niveles de comparación de 0 a 9,9 V

Diagrama

Descripción generada automáticamente

Para conseguir ganancia 3 usaremos una resistencia R2 de 2k Ω y una R1 de 1k Ω para cumplir:

G=1+R2/R1

Como también queremos un valor de comparación negativo necesitamos un amplificador inversor de ganancia 1 alimentado con Vcomp+. Usaremos dos resistencias de 1k Ω para cumplir: G= -R2/R1

Imagen que contiene Gráfico

Descripción generada automáticamente

* 1. Comparador Overload

Para comparar el valor amplificado por el A.I con Vcomp+- necesitamos dos comparadores LM311 que usaremos para marcar un nivel lógico alto y bajo en el microcontrolador a través de dos pines digitales dependiendo de si se ha excedido el overload.

Diagrama

Descripción generada automáticamente

Este comparador LM311 funciona de la siguiente manera, si Vp <Vn el transistor interno se pone en modo saturación y si Vp >Vn se pone a corte. Este transistor esta conectado como se ve a continuación:



De esta manera si el transistor está en corte, el voltaje que cae en Vo es Vo= Vcc (R1/R1+R2). Si ajustamos las resistencias por ejemplo a 86,6Ω y 174.5 Ω tenemos un valor Vo muy aproximado a 3,3 V que es el nivel alto que queremos enviarle al microcontrolador.

Sin embargo, cuando el transistor está en saturación obligamos a que Vo sea 0V adjudicándole un nivel bajo que transmitiremos al micro.

# SOFTWARE

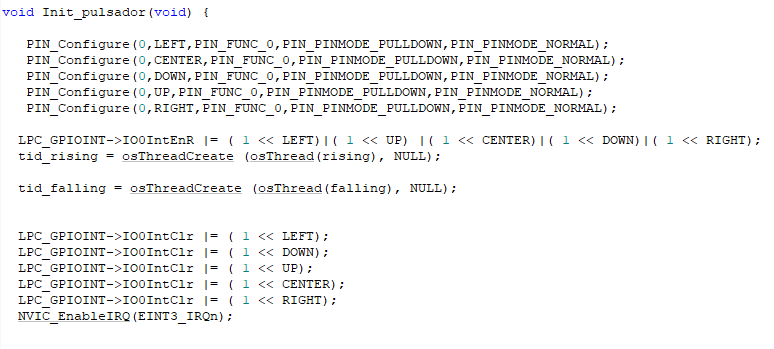
## Descripción del funcionamiento de la aplicación completa (Práctica 3).

La mayoría de partes del programa tienen un archivo.c y su librería.h. en el que se desarrolla una funcionalidad unica

**Gestor\_pulsador.c**

Dentro de este archivo esta la función llamada Init\_pulsador que inicializa los botones y los dos hilos que se

usan para gestionar los rebotes.



Además de esta función y las de los dos hilos falling y rising

también encontramos el manejador de interrupciones EINT3\_IRQHandler que nos permite bajar los flags de

todos los botones. Usamos el botón center y sus interrupciones para para reiniciar el RTC a 0:0:0

1/1/2000 como se ve en esta captura mandando una señal al hilo correspondiente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**LCD.c**

en este archivo se encuentran todas las funciones que se usaban el año pasado para hacer funcionar la

pantalla. Se instancia el SPI y a través de la función init\_LCD se inicializan todos los parámetros.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

además hemos incorporado respecto al año pasado un par de funciones llamadas pintarHoraPrimeraLinea()

y pintarFechaLinea() para que pasándoles solo los dígitos se pinten en pantalla.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**SNTP.c**

Contiene la función que inicializa el hilo del SNTP.

Imagen que contiene Logotipo

Descripción generada automáticamente

También encontramos la función del hilo Servidor\_SNTP que gestiona las señales que le llegan para sincronizar la fecha y hora.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Como el dato que devuelve el servidor SNTP en un valor entero con la cantidad de segundos desde 1970 tenemos funciones que convierten ese entero en los diferentes valores para asignárselos al RTC.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**RTC.c**

Dentro de este archivo lo primero que hay que destacar es la función de inicialización init\_RTC que instancia el RTC y los dos hilos que usamos llamados alarma y pantalla.

Texto, Carta

Descripción generada automáticamente

También encontramos un manejador de interrupciones RTC\_IRQHandler() que gestiona la alarma que hemos programado y que alza el flag cada minuto. En este manejador se activa una señal para el hilo alarma.

Texto

Descripción generada automáticamente

En este hilo llamado alarma gestionamos con un contador de modulo 3 que cada tres minutos se sincronice el RTC con el SNTP y se encienda un led durante 5 segundos parpadeando. Todo esto gracias a la señal que se manda desde el manejador de interrupciones RTC\_IRQHandler()

Texto

Descripción generada automáticamente con confianza media

Con El otro hilo llamado pantalla gestionamos las señales que se mandan para escribir por pantalla tanto las dos frases que se pueden pasar desde la página web como la hora y la fecha del reloj.

Imagen que contiene Texto

Descripción generada automáticamente

Además, en este archivo también se pueden encontrar las funciones que nos permiten activar o desactivar el RTC, configurar el tipo de interrupción (segundos, minutos, horas…).

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

**HTTP\_Server\_CGI.c**

Este archivo es parte de la aplicación HTTP\_server desde la que comenzamos a construir las prácticas. Debido a que se ha incorporado funcionalidades como el control de los leds, la escritura por pantalla, la lectura del ADC entre otras cosas, ha sido necesario ampliar el código de las funciones cgi\_process\_data() que procesa los mensajes POST que se mandan desde la parte del cliente. También hemos añadido código a la función que genera las paginas web para poder incorporar los datos que se mandan como la hora y la fecha en el nuevo apartado de Time que se ve en la página.

**RTC.cgi y RTC.cgx**

Basándonos en los códigos de adc.cgi y adc.cgx hemos creado una nueva pestaña en index.htm llamada Time en la que se muestra la hora y la fecha del RTC cada vez que se actualiza o periódicamente dos veces por segundo.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

También hemos incorporado una pestaña que muestra dos IP de dos servicios SNTP diferentes pero que no hemos podido acabar de hacer funcionales por falta de tiempo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Adc.c**

Dentro de este archivo lo primero que encontramos es una función de inicialización llamada init\_adc() que lo activa e inicializa las interrupciones 4 y 5 para poder usar los dos adc.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

En las funciones conversión\_ADC\_P0T1 y conversión\_ADC\_P0T2 se activa una interrupción para el AD0.4 apagando la AD0.5 en el registro AD0CR. Para después iniciar la conversión y esperar al resultado. El único ADC que hemos programado y que se pide usar es el AD0.5.

Texto

Descripción generada automáticamente

Ademas hemos creado una función para hacer un retardo y que la medición del SDC se pueda hacer correctamente

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

**HTTP\_Server.c**

Es la función principal (main ) donde se usan todas las funciones de inicialización, se habilita el RTC y la alarma de un minuto del RTC.

Texto, Carta

Descripción generada automáticamente

Para la fase 7 almacenamos un valor para compararlo después con el ADC

Texto

Descripción generada automáticamente

También encontramos un hilo que permite usar la funcionalidad de Runnig leds desde la pagina web

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Además es aquí donde se ejecuta el código que se ha programado para la practica 7 para guardar el estado de los leds y iniciarlos igual que estaban en el último arranque además de guardar la dirección IP del servidor y la MAC.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Gestion\_FLASH.c**

En este archivo hemos codificado todas las funciones que usaremos para interactuar con la memoria FLASH.

Para borrar un sector entero usamos la función borrar\_sector(numero\_sector)

Texto, Carta

Descripción generada automáticamente

Usamos la función escribirMemoria() para guardar datos sin importarnos que en ese sector ya haya datos y se sobrescriban

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Si nuestra intención es solo leer de la Flash usamos la función leerFlash()

Texto

Descripción generada automáticamente

Pero si lo que queremos es escribir sin sobrescribir los datos que hay guardados en un sector sin pisar esos datos tenemos que primero leer esos datos y volver a copiarlos junto con las modificaciones que hayamos hecho.

Para eso usamos estas funciones modificarXposicion() y leerFlashxPosicionesyRetornarArray

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**pinoutConfigure.c**

en este archivo inicializamos los leds en la función pinConfiguration para que puedan ser usados posteriormente.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para inicializar los leds usamos una estructura llamada led con el puerto y el pin

Texto

Descripción generada automáticamente

Para poder seleccionar desde la pagina web el led que podemos encender desde la función browser hemos creado una serie de funciones para encender y apagar el adecuado en cada momento

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Para poder usar la funcionalidad de runnig leds desde la pagina web hemos creado esta otra función que los enciende y apaga cíclicamente.

Texto, Carta

Descripción generada automáticamente

**Led\_RGB.c**

En este archivo encontramos una serie de funciones auxiliares para usar el led rgb cómodamente en sus tres colores.

Texto

Descripción generada automáticamente

# 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.

## Plan de pruebas del sistema.

Por cada práctica (excepto para la práctica 3 que tiene dos partes diferenciadas) debe proponer un plan de pruebas que permita validar alguna de las funcionalidades indicadas en la especificación. El objeto de este plan es permitir que un tercero que no conoce el código pueda validar si el comportamiento del sistema es el correcto. Para cada prueba que proponga debe incluir la siguiente información:

1. Objetivo de la prueba: se debe detallar qué aspecto del sistema se quiere probar (por ejemplo, funcionamiento correcto del RTC en ausencia de alimentación, envío del valor del ADC al cliente Web, escritura y lectura correcta en la memoria flash, integración de todo el sistema, etc.)
2. Método de prueba: se indicará detalladamente los pasos que se deben seguir para realizar la prueba
3. Resultados esperados: debe indicarse cuáles son los resultados que se esperan al realizar los pasos anteriores.

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

La primera prueba que hicimos fue comprobar la conexión de nuestro servidor a la red local asignándole la correspondiente IP y configurando las opciones de Ethernet.

Para ello el método usado fue usar la consola de comandos de Windows y hacer un ping a la dirección ip esperando obtener respuesta.

La segunda prueba fue realizada cuando codificamos las funciones de los leds y las integramos con las del servidor. Usando un buscador y utilizando la página web vimos si el código implementado era correcto. Para ello como en la mayoría de las pruebas hicimos uso del debugger y los puntos de ruptura.

Para poder integrar el LCD y el ADC seguimos un procedimiento igual al anteriormente descrito.

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

Las pruebas de la segunda práctica que se realizaron fueron para ver si el RTC había sido bien instanciado al igual que la alarma de un minuto empleada, ver si el servidor SNTP respondia, y ver si la nueva pagina web creada respondia adecuadamente. Para ello se usaron las minsmas herramientas que anteriormente.

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

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

El objetivo de las pruebas realizadas en este apartado se enfoca sobre todo a ver como funciona la memoria Fash. Usando el debugger y la ventana de Memory para ver directamente el comportamiento de la memoria y las funciones.

## Pruebas realizadas.

Todas las pruebas anteriormente descritas han sido realizadas y una vez comprobados sus resultados estos han sido satisfactorios.