Reloj de palabras con sincronización en tiempo real online

Jorge Saiz Gil
Ingeniería Técnica de Informática de Sistemas
Sistemas Embebidos

Jordi Bécares Ferrés
Pere Tuset Peiró
Xavi Vilajosana Guillem

13 de Junio de 2016

Dedicatoria

A Elena, el amor de mi vida, sin su apoyo, ayuda y paciencia no podría haber llegado hasta el final de mis estudios.

A mi familia, por hacer que me haya convertido en la persona que soy, especialmente a mi madre, por el esfuerzo y sacrificio durante tantos años dedicados a su familia.

Gracias.

FICHA DEL TRABAJO FINAL

Título del trabajo:	Reloj de palabras con sincronización en tiempo real online
Nombre del autor:	*Jorge Saiz Gil*
Nombre del consultor/a:	**Jordi Bécares Ferrés**
Nombre del PRA:	**Pere Tuset Peiró** **Xavi Vilajosana Guillem**
Fecha de entrega:	06/2016
Titulación:	Ingeniería Técnica de Informática de Sistemas
Área del Trabajo Final:	*Sistemas embebidos*
Idioma del trabajo:	*Español*
Palabras clave:	*Sincronización en tiempo, reloj, embebido*

Resumen del Trabajo

El objeto de este trabajo es desarrollar un dispositivo embebido capaz de sincronizar la hora a través de Internet y mantenerla con un desfase máximo de medio segundo, además de poder mostrarla a través de un reloj con un formato novedoso, que se puede denominar “reloj de palabras”. Este producto podría ser utilizado en versiones más maduras como servidor de tiempo para cualquier otro dispositivo, además de ser un reloj original y con un gran atractivo visual.

Para obtener este dispositivo, se ha utilizado el sistema operativo FreeRTOS sobre la placa de LPC1769 a la que se le han conectado un módulo Wifly RN-XV para proveer el acceso a Internet, un módulo RTC DS3231 para almacenar la hora, un módulo CP2102 para poder tener acceso al registro de actividad del sistema y una panel de leds diseñado específicamente para mostrar la hora. Mediante estos componentes, la aplicación embebida que controla el dispositivo, realiza comunicaciones con servidores de tiempo NTP para obtener el valor de la hora actual y almacenarla en el RTC, para después poder mostrarla a través del reloj. Todo esto controlando los posibles errores y desfases de tiempo que puedan ocurrir durante el proceso.

Finalmente se ha obtenido un dispositivo embebido que cumple con los requisitos mínimos que se exigían y que puede funcionar de forma totalmente independiente, pero que necesitaría ser mejorado si se quisiese llevar a producción.

Abstract

The purpose of this work is to develop an embedded device that can synchronize time over the Internet and keep it with a maximum delay of half a second, in addition the device is being able to show this time through a watch with a new format, which can be referred as “Word Clock”. This product could be used to, in future versions, as a time server for any other device, besides being an original watch and great visual appearance.

To get this device, we use FreeRTOS operating system over the LPC1769 board, which will have connected a Wifly RN-XV module to provide access to Internet, an RTC DS3231 module to store the time, CP2102 module able to access the system activity log and a led panel specifically designed to display the time. Through these components, the embedded application that controls the device, performs communications with NTP time servers to get the value of the current time and stored it in the RTC, then it show through the clock. All of this, always controlling the possible errors and time lags that may occur during the process.

Finally we obtained an embedded device that meets the minimum requirements needed and it’s able to operate completely independently, but it need to be improved if we wanted to bring it into production.

Índice

Lista de figuras

1. Introducción.

En la actualidad, es cada vez más común que cualquier dispositivo del que disponemos pueda conectarse a Internet para enviar y recibir información de cualquier tipo. La evolución de la tecnología ha permitido que cada vez podamos tener dispositivos más pequeños, más potentes y de menor coste. Esto ha propiciado que este en auge lo que se conoce como el “Internet de las cosas“ (en inglés, Internet of Things, abreviado IoT)(1). A partir de este concepto surge la idea de desarrollar este proyecto.

1.1 Contexto y justificación del trabajo.

Desde el principio de la civilización, el hombre ha tenido la necesidad de poder medir el tiempo para regular sus hábitos y tareas. De esta necesidad surgieron los primeros calendarios y relojes, y a medida que fue evolucionando la humanidad, los relojes lo hicieron también. Hoy día tenemos relojes en cualquier parte, en los teléfonos móviles, en nuestros televisores, en el horno o colgados de la pared. Medir el tiempo se ha convertido en una obsesión humana.(2)

Centrándonos en como lo relojes nos dan la hora, estamos acostumbrados a utilizar relojes de agujas o relojes digitales, que nos muestran la hora de forma numérica, los minutos, los segundos e incluso pueden mostrarnos la fecha en la que nos encontramos. Uno de los objetivos principales de este proyecto es crear un reloj que nos muestre la hora de una forma diferente y peculiar, que llamaremos reloj de palabras. Aunque este no sea el objetivo principal del proyecto, sí que es la parte que resulta más atractiva o diferente, sobre todo, si lo comparamos con los relojes que estamos acostumbrados a ver.

Aunque el reloj de palabras es la parte más visible y estética del proyecto, el verdadero objetivo es poder obtener y servir la hora con el mínimo retraso posible con respecto a la hora real en que nos encontramos. Para ello, utilizara un sistema embebido que se conecte a servidores de tiempo de Internet de donde obtendrá la hora actual.

Combinando las facilidades que nos da la tecnología actual, dispositivos pequeños, potentes y baratos, con el diseño innovador del reloj de palabras, obtenemos un dispositivo que sirve la hora en tiempo real y la muestra de una forma novedosa.

1.2. Descripción del trabajo.

Este proyecto consiste en la creación de un dispositivo embebido que cuenta con dos funcionalidades principales.

Por una parte, realiza una sincronización de la hora con un error de precisión inferior a quinientos milisegundos con respecto al tiempo universal coordinado (UTC). Para ello el sistema es capaz de realizar y gestionar conexiones a Internet a través de redes Wi-Fi, y obtener la hora actual mediante peticiones a servidores de tiempo NTP. Una vez recibida la hora, realiza un ajuste del valor obtenido, teniendo en cuenta los retrasos producidos en la sincronización y proceso de almacenado, para finalmente proceder a almacenar el valor obtenido en un reloj interno e independiente que actuara como reloj de referencia para el resto de funcionalidades del dispositivo.

Figura 1: Ejemplo reloj de palabras.

Por otra parte, el dispositivo muestra la hora sincronizada a través de un panel con un formato muy especial. El panel contiene una sopa de letras, formada por diez filas y once columnas, donde se pueden encontrar todas las palabras necesarias para expresar de forma trascrita, y en rangos de cinco minutos, la hora en la que se encuentra. Mediante la iluminación de cada una de las palabras que forman la transcripción, se podrá leer en el panel la hora actual. Además, para dotar de precisión de minutos al panel, cada esquina cuenta con un indicador que sirve para especificar en qué minuto en concreto, se encuentra dentro del rango de cinco minutos. Por ejemplo, si el reloj tiene que mostrar que son las 19:33, en el panel se iluminara la frase “SON LAS SIETE Y MEDIA” y los indicadores de los minutos uno, dos y tres. (Figura 1).

Además de las funcionalidades indicadas, el dispositivo cuenta con un sistema de monitorización que registra en tiempo real la actividad del dispositivo, a la cual se puede acceder mediante un puerto serie.

1.3. Objetivos del TFC.

Los objetivos que se han planteado para este proyecto se pueden dividir en dos grupos, por un lado tenemos los objetivos principales, los cuales proporcionan al sistema las funcionalidades básicas para su correcto funcionamiento, y después tenemos los objetivos secundarios, que completan el sistema mejorando ciertos aspectos:

- **Objetivos principales:**
  - Sincronización horaria del dispositivo mediante el protocolo NTP.
  - Dotar al sistema de hora cuando el dispositivo no tenga conexión a Internet usando un reloj externo.
  - Notificación visual de la hora a través de un reloj de palabras.
- **Objetivos secundarios:**
  - Poder configurar el dispositivo a través de una página web o aplicación de móvil.
  - Mejorar el sistema de gestión de la iluminación del panel mediante el uso de un circuito integrado o driver led.
  - Protección ante caídas de tensión.

1.4 Enfoque y método seguido.

La metodología utilizada para desarrollar el proyecto ha sido el método Bottom-up(3). En este método se diseñan individualmente las funcionalidades más básicas para luego ir enlazándolas dentro de los métodos de más alto nivel. Por ello, el inicio del proyecto estuvo centrado en el desarrollo de las librerías de comunicación entre los diferentes módulos del dispositivo como el panel de leds, el módulo Wifly o el reloj externo. Una vez desarrollado el nivel más bajo del sistema, se procedió a analizar los siguientes requisitos del sistema e ir avanzando de forma jerárquica en el desarrollo global del dispositivo. De esta forma se fueron cubriendo

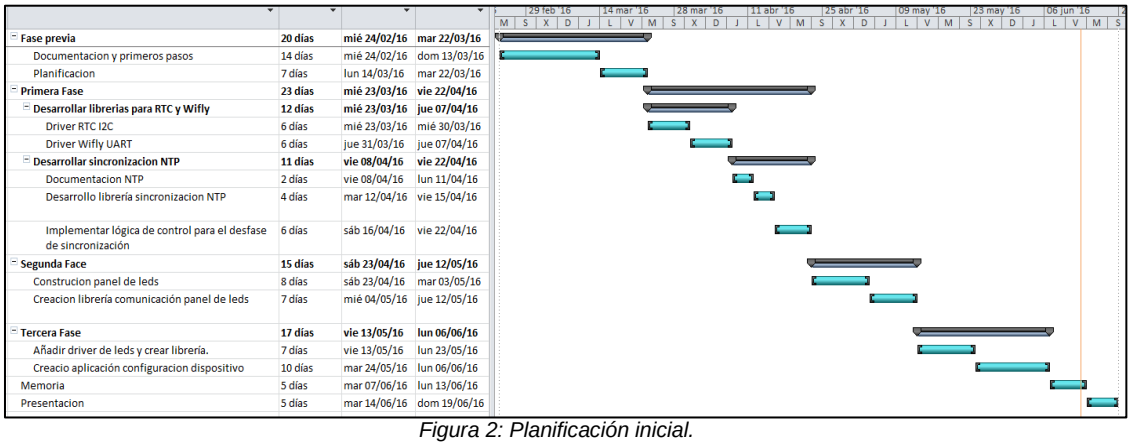
necesidades nivel a nivel, utilizando las librerías o métodos ya implementados, hasta conseguir un sistema funcional. Gracias al uso de este método, se ha conseguido una adaptación progresiva con el entorno de desarrollo, los componentes del dispositivo y el sistema operativo embebido utilizado.

1.5 Planificación del trabajo.

En este apartado se describe la planificación inicial que se realiza en el comienzo del proyecto, así como la planificación final que resulta después de tener en cuenta los problemas y retrasos producidos a lo largo del desarrollo del proyecto.

- **Planificación inicial:**

  La planificación inicial que se planteó, contaba con una fase previa en la que se hacía una toma de contacto con el sistema y los componentes a utilizar, más tres fases en las que se desarrollaba el proyecto hasta llegar a alcanzar los objetivos principales y secundarios marcados. Estas fases se hicieron coincidir con las entregas parciales programadas por la universidad (Figura 2).



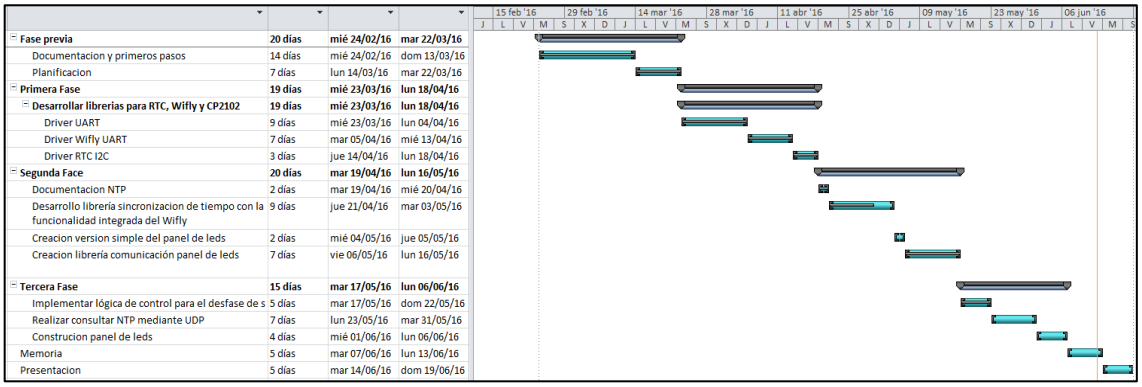
Fase previa	**20 días**	**mié 24/02/16**	**mar 22/03/16**
Documentacion y primeros pasos	14 días	mié 24/02/16	dom 13/03/16
Planificacion	7 días	lun 14/03/16	mar 22/03/16
Primera Fase	**23 días**	**mié 23/03/16**	**vie 22/04/16**
Desarrollar librerias para RTC y Wifly	**12 días**	**mié 23/03/16**	**jue 07/04/16**
Driver RTC I2C	6 días	mié 23/03/16	mié 30/03/16
Driver Wifly UART	6 días	jue 31/03/16	jue 07/04/16
Desarrollar sincronizacion NTP	**11 días**	**vie 08/04/16**	**vie 22/04/16**
Documentacion NTP	2 días	vie 08/04/16	lun 11/04/16
Desarrollo librería sincronizacion NTP	4 días	mar 12/04/16	vie 15/04/16
Implementar lógica de control para el desfase de sincronización	6 días	sáb 16/04/16	vie 22/04/16
Segunda Face	**15 días**	**sáb 23/04/16**	**jue 12/05/16**
Construcion panel de leds	8 días	sáb 23/04/16	mar 03/05/16
Creacion librería comunicación panel de leds	7 días	mié 04/05/16	jue 12/05/16
Tercera Fase	**17 días**	**vie 13/05/16**	**lun 06/06/16**
Añadir driver de leds y crear librería.	7 días	vie 13/05/16	lun 23/05/16
Creacio aplicación configuracion dispositivo	10 días	mar 24/05/16	lun 06/06/16
Memoria	5 días	mar 07/06/16	lun 13/06/16
Presentacion	5 días	mar 14/06/16	dom 19/06/16

Figura 2: Planificación inicial.

- **Planificación final:**

  Como era de esperar, durante el proceso de desarrollo del proyecto han surgido problemas y necesidades que no se habían contemplado en la planificación inicial. Estas incidencias han producido que el proceso de desarrollo del proyecto evolucionase de diferente forma (Figura 3).

En un principio, se consideró que durante la primera fase del proyecto se podría llegar a alcanzar el objetivo principal del mismo, tener un dispositivo capaz del sincronizar la hora a través de Internet con el mínimo desfase posible, por el contrario, una vez finalizada la primera fase, solo se había podido llegar a implementar el desarrollo de las librerías de control y comunicación entre los diferentes dispositivos, esto fue debido principalmente a problemas con el modulo inalámbrico Wifly.



Fase previa	**20 días**	**mié 24/02/16**	**mar 22/03/16**
Documentacion y primeros pasos	14 días	mié 24/02/16	dom 13/03/16
Planificacion	7 días	lun 14/03/16	mar 22/03/16
Primera Fase	**19 días**	**mié 23/03/16**	**lun 18/04/16**
Desarrollar librerias para RTC, Wifly y CP2102	**19 días**	**mié 23/03/16**	**lun 18/04/16**
Driver UART	9 días	mié 23/03/16	lun 04/04/16
Driver Wifly UART	7 días	mar 05/04/16	mié 13/04/16
Driver RTC I2C	3 días	jue 14/04/16	lun 18/04/16
Segunda Face	**20 días**	**mar 19/04/16**	**lun 16/05/16**
Documentacion NTP	2 días	mar 19/04/16	mié 20/04/16
Desarrollo librería sincronizacion de tiempo con la funcionalidad integrada del Wifly	9 días	jue 21/04/16	mar 03/05/16
Creacion version simple del panel de leds	2 días	mié 04/05/16	jue 05/05/16
Creacion librería comunicación panel de leds	7 días	vie 06/05/16	lun 16/05/16
Tercera Fase	**15 días**	**mar 17/05/16**	**lun 06/06/16**
Implementar lógica de control para el desfase de s	5 días	mar 17/05/16	dom 22/05/16
Realizar consultar NTP mediante UDP	7 días	lun 23/05/16	mar 31/05/16
Construcion panel de leds	4 días	mié 01/06/16	lun 06/06/16
Memoria	5 días	mar 07/06/16	lun 13/06/16
Presentacion	5 días	mar 14/06/16	dom 19/06/16

Figura 3: Planificación final.

Estos retrasos producidos en la primera fase, desencadenaron en que el resto de la planificación se adaptase, concentrándose en poder cumplir con los objetivos principales del proyecto. Además, en previsión de futuras incidencias, se decidió implementar una alternativa más sencilla del sistema de sincronización de la hora a través de Internet, así como, una versión simple del panel de leds.

Finalmente, no se volvió a producir ningún problema grave durante el resto del desarrollo, por lo que se ha obtenido un dispositivo que cumple con los objetivos principales del proyecto.

1.6 Recursos empleados

Para exponer los recursos que han sido necesarios para la elaboración del proyecto, se ha decidido separarlos en recursos de hardware, de software y otros recursos.

Indicar que, de los recursos de hardware, tanto la placa LPC1769, el módulo Wifly y el CP2102 han sido facilitados por la universidad como base de desarrollo para realizar el proyecto y el resto de elementos, se encuentran en el mercado a disposición de cualquiera y fácilmente adquiribles.

Sobre los recursos de software, exceptuando el sistema operativo de Microsoft, todos los elementos indicados son de licencia gratuita y están disponibles en las páginas web de sus correspondientes desarrolladores.

- **Recursos de hardware empleados.**

  - Placa de desarrollo LPCXpresso LPC1769 (Figura 4).

Figura 4: Placa LPC1769.

  - Módulo inalámbrico Wifly RN-XV junto con adaptador DIP Xbee (Figura 5).

Figura 5: Modulo Wifly y adaptador DIP Xbee

  - Reloj en tiempo real con comunicación I2C, DS3231 (Figura 6).

Figura 6: Reloj en tiempo real DS3231.

  - Convertidor USB/UART CP2102 (Figura 7).

Figura 7: Convertidor CP2102

- Fuente de alimentación de 5V/2A (Figura 8).

Figura 8: Fuente de alimentación.

- Diodos led blancos 3.2V - : 20mA y resistencias para trabajar con 5V (Figura 9).

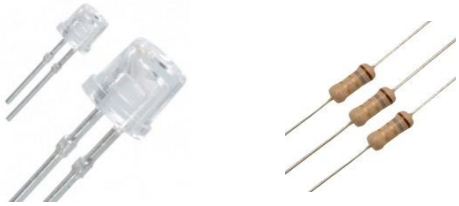


Figura 9: Diodos led y resistencias.

- Drivers de potencia ULN2803 (Figura 10).

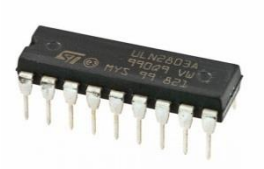


Figura 10: Driver de potencia ULN2803.

- Placas y cables para prototipos (Figura 11).

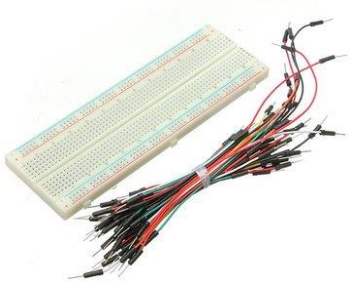


Figura 11: Placa de pruebas.

- Material necesario para realizar trabajos de electrónica: Soldador, estaño, cables,…
- Ordenador de sobremesa con procesador Intel I5, 8Gb de memoria RAM y disco duro SSD 256Gb.

- **Recursos de software empleados.**

  - Entorno de desarrollo LPCXpresso IDE v8.0.0_526.
  - Sistema operativo en tiempo real embebido FreeRTOS.
  - Software emulación de terminal serie. Tera Term v4.9.
  - Software de diseño vectorial Inkscape v0.91
  - Sistema operativo Microsoft Windows 7 64bits. Compatible con el entorno de desarrollo.

- **Otros recursos empleados:**

  Además de los recursos ya mencionados, para la construcción del reloj de palabras han sido necesarios los siguientes materiales:

  - Vinilo adhesivo negro.
  - Panel de metacrilato de 40 cm x 40 cm.
  - Caja de madera de 40 cm x 40 cm x 3 cm.
  - Cuadricula de contrachapado de 10 filas x 11 columnas.

1.7 Productos obtenidos

Una vez finalizado el proyecto, los productos obtenidos son, en primer lugar, un dispositivo embebido capaz de sincronizar la hora a través de Internet con un desfase máximo de quinientos milisegundos con respecto a la hora y que es capaz de servir ese valor, en este caso al reloj de palabras (Figura 12Figura 13). Para sincronizar la hora, cada cierto intervalo de tiempo el dispositivo realiza peticiones a servidores de tiempo, ajustando los posibles retrasos que se producen y almacenando el valor en un reloj interno e independiente, el cual es capaz de mantener la hora aunque el dispositivo se quede sin alimentación. Además, es capaz de gestionar el estado de la conexión a Internet.

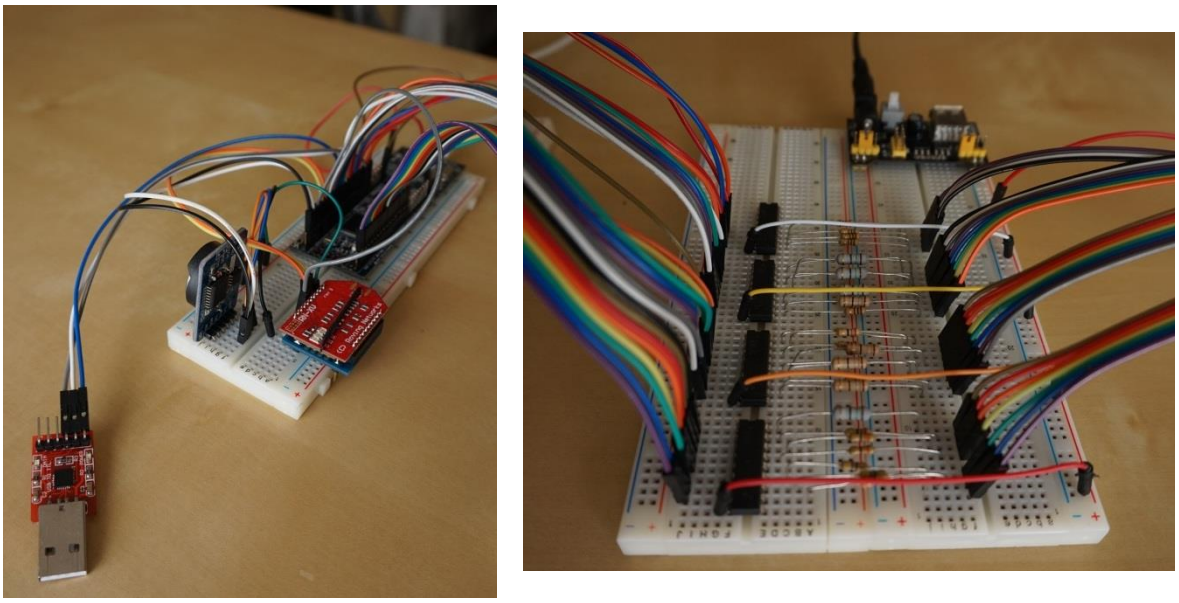


Figura 12: Dispositivo embebido y circuito de conexión con reloj

En segundo lugar, se ha obtenido un panel de leds, que llamaremos reloj de palabras, en el que el dispositivo ilumina en una sopa de letras las palabras necesarias para que se pueda leer la hora en el panel (Figura 13).

Figura 13: Reloj de palabras..

Además de los productos mencionados, también se ha obtenido una versión de pruebas del panel. Construido sobre una placa de prototipos, cuenta con veintiocho leds que representan cada una de las palabras e indicadores necesarios para mostrar la hora. De esta forma, cuando un led se ilumina significa que en el panel definitivo se iluminaria la palabra correspondiente.

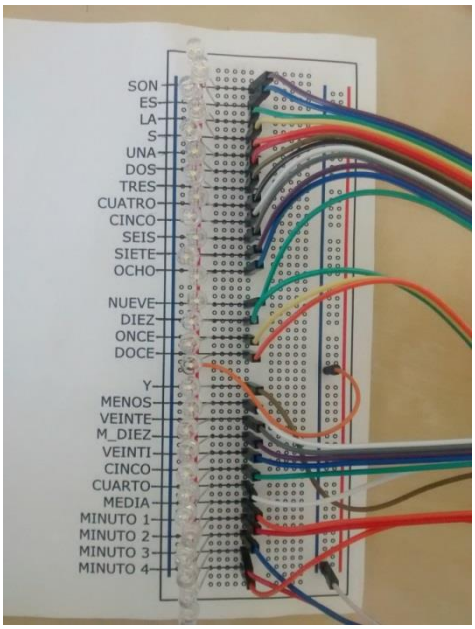


Figura 14: Panel de leds simple.

1.8 Breve descripción de los otros capítulos de la memoria.

Una vez descrito de forma general el proyecto objeto de esta memoria, se presenta un resumen de cada uno de los capítulos siguientes.

En el capítulo dos, se analiza el estado actual de las tecnologías usadas para diseñar el dispositivo, además se hace un estudio de mercado donde se muestran las alternativas existentes al sistema propuesto. En el capítulo tres, descripción funcional, se enumeran y explican las funcionalidades con las que cuenta el dispositivo y que le permiten cumplir con los objetivos que se han marcado. Una vez descritas las funcionalidades del dispositivo, en el capítulo cuatro se procede a detallar como se ha conseguido implementarlas mediante el desarrollo de una aplicación embebida y el uso de los componentes de hardware necesarios. En el capítulo cinco se realiza un estudio sobre viabilidad técnica del dispositivo, para continuar en el capítulo seis, con un análisis del coste asociado a la realización del proyecto y para finalizar, en el capítulo siete, se exponen las conclusiones que se han alcanzado, un análisis reflexivo sobre el proyecto y para acabar se realizan algunas propuestas de mejora para el dispositivo.

2. Antecedentes.

Los sistemas embebidos son sistemas formados por componentes electrónicos que mediante microcontroladores pueden realizar procesamiento de datos, pero que a diferencia de los ordenadores comunes están diseñados para realizar una o algunas pocas tareas muy específicas. En la actualidad existen miles de ejemplos de sistemas embebidos, como pueden ser teléfonos móviles, televisores, reproductores multimedia, routers, electrodomésticos o relojes. Estos sistemas están diseñados para optimizar su función a la vez que el tamaño, el coste y el consumo, además de poder realizar procesos en tiempo real.

El continuo avance de la tecnología ha provocado que los sistemas embebidos cada vez sean más económicos, potentes y compatibles con un gran número de sistemas y periféricos.

Junto con esta evolución de la tecnología, han surgido una gran variedad de plataformas de desarrollo de código abierto, que han hecho que este tipo de sistemas puedan llegar al gran público y casi cualquiera pueda crear su propio sistema embebido.

Todo esto ha provocado que se cree un nuevo movimiento cultural, autodenominado “Cultura Maker”, que gracias a las grandes posibilidades de los sistemas embebidos promueve el que cada uno pueda ser capaz de fabricarse sus propios dispositivos. Todo esto nos da a entender que los sistemas embebidos están en auge, y que formaran parte de nuestro futuro.

2.1 Estado del arte

La mencionada evolución de los sistemas embebidos hace que existan una gran variedad de plataformas de desarrollo con las que realizar el proyecto objeto de esta memoria, ante la gran cantidad de plataformas existentes se ha decido enumerar tres de las más extendidas:

- **Arduino.**

  Plataforma de hardware de código abierto, que cuenta con su propio lenguaje de programación y su propio entorno de desarrollo basado en Wiring. Existen una gran variedad de modelos disponibles, la mayoría se apoyan en microcontroladores de la familia Atmel AVR, de entre ellas la más extendida es la placa Arduino UNO (Figura 15). Una de las ventajas de este sistema es que podemos comprar cualquiera de sus versiones ya construidas o comprar los componentes y construirla nosotros mismos, permitiendo un ahorro en costes. Gracias a esta flexibilidad y al bajo coste, este sistema se ha extendido tanto en el ámbito académico como a nivel de aficionados, en consecuencia, la documentación e información sobre el desarrollo de esta plataforma abunda en Internet.

Figura 15: Placa Arduino UNO

- **Raspberry.**

  Este sistema está considerado como un ordenador de placa simple de bajo coste y fue creado para usarse como herramienta de aprendizaje (4). Cuenta con un SoC que integra CPU, GPU, SDRAM, que le dota de una potencia superior a las alternativas mencionadas aquí, integra conexiones de audio y video además de contar con un lector para tarjeta SD y conexiones USB. Aunque esta alternativa sea más potente, está enfocada para la ejecución de sistemas operativos completos y realizar tareas más relacionadas con las que puede hacer un ordenador común. La versión del dispositivo que se distribuye actualmente es la Raspberry Pi 3 Model B (Figura 16).

Figura 16: Raspberry Pi 3 Model B

- **Plataforma LPCXpresso**

  Esta plataforma es una herramienta de desarrollo de bajo coste, fabricada por NXP y basada en la familia de microcontroladores de bajo coste energético LPC que utilizan la tecnología ARM. También cuenta con un IDE propio de licencia gratuita y al contrario que ocurre con Raspberry, esta plataforma si está más enfocada al desarrollo de sistemas embebidos. Para este proyecto se utilizara la placa LPCXpresso LPC1769 (Figura 4).

Una vez analizadas las principales plataformas de desarrollo de sistemas embebidos, entramos a analizar el estado del arte en lo referente a los posibles métodos de sincronización horaria que existen en la actualidad:

- **Sincronización por radio.**

  Alrededor del mundo existen diferentes emisoras de radio que emiten señales de radio de onda larga que transmiten la información horaria, esas señales pueden ser recibidas por módulos receptores que la interpretan (5). Las señales puede llegar a recibirse a 2000 Kilómetros de su fuente, pero, uno de los principales problemas de esta tecnología es que no existe un estándar unificado para la emisión de las señales horarias, por lo que, dependiendo de en qué parte del mundo nos encontremos, deberemos adaptar nuestro sistema de recepción a las señales de esa zona. Además, este tipo de señales se puede ver afectadas por fenómenos ajenos al sistema, que pueden producir problemas de sincronización. En la Figura 17 podemos ver algunos de los tipos de señal horaria que existen.

Protocolo de la señal	País de emisión	Frecuencia
DCF77	Alemania	77.5 kHz
JJY	Japón	40 kHz
wwvb	Estados Unidos	60 kHZ
TDF	Francia	162 kHz

Figura 17: Tipos de señales horarias de radio.

- **Sincronización GPS.**

  Además de la longitud, latitud y altitud, el Sistema de Posicionamiento Global (GPS) proporciona una cuarta dimensión esencial: la cronometría (6). Cada satélite de la constelación GPS contiene múltiples relojes atómicos que contribuyen con datos horarios muy precisos a las señales del GPS. Estas señales pueden ser recibidas por receptores GPS que sincronizan su hora a través de ellas. El problema de este sistema de sincronización, es que necesitamos estar al aire libre para poder recibir la señal correctamente, por lo que su uso en interiores, al igual que para el posicionamiento GPS, no es recomendado.

- **Sincronización a través de internet (NTP).**

  NTP es un protocolo de internet estandarizado que sirve para sincronizar los relojes de los sistemas informáticos con una gran precisión, utiliza el protocolo de la capa de transporte

UDP a través del puerto 123 para enviar paquetes que contienen la información horaria. Este protocolo está extendido globalmente y existen cientos de servidores de los que poder obtener la hora utilizando cualquier interfaz de red (Ethernet, Wi-Fi,..)

2.2 Estudio de mercado

A la hora de realizar el estudio de mercado sobre el dispositivo desarrollado, hay que tener en cuenta que podemos enfocar el estudio desde diferentes puntos. Por un lado, se puede analizar el mercado considerando el dispositivo como solo un sistema de sincronización horaria en tiempo real de alta precisión, sin tener en cuenta el reloj de palabras y por otro lado, podemos realizar el estudio considerando el dispositivo como un reloj de palabras que cuenta con sincronización horaria en tiempo real.

Si realizamos el estudio considerando el dispositivo solo como un sistema de sincronización horaria en tiempo real de alta precisión, por un lado, se observa que no existen o no se han encontrado dispositivos embebidos que realicen únicamente esa función, al parecer el uso de la sincronización horaria a través de internet está más enfocado a aplicaciones de software que se ejecutan sobre sistemas operativos completos (Linux, Windows,…). Por otro lado, si consideramos la sincronización horaria en tiempo real como una funcionalidad complementaria, si existe una gran cantidad de dispositivos embebidos que la implementan, y lo hacen utilizando cualquiera de los métodos de sincronización mencionados en el estudio del arte. Como por ejemplo, la estación meteorológica Oregon Scientific BAR 386 (Figura 18), que además de realizar sus funciones específicas de predicciones meteorológicas, cuenta con un sistema de sincronización horaria mediante radio que utiliza la señal DCF-77.

Figura 18: Estación meteorológica con sincronización horaria

Otro ejemplo de sistema que utiliza la sincronización horaria, en este caso a través de GPS, es el dispositivo de navegación asistida TomTom GO 61 (Figura 19) que sincroniza la hora automáticamente utilizando las señales GPS que recibe.

Figura 19: Navegador GPS

Como ejemplo de sistema embebido que cuenta con la sincronización horaria a través de Internet como función compleméntenla, en este mismo proyecto se está utilizando uno. El modulo inalámbrico Wifly RN-XV cuenta con una vía para poder sincronizar la hora a través de servidores de tiempo NTP, durante el desarrollo del proyecto se han implementado funciones para utilizar este método como método de obtención de la hora a través de Internet.

Una de las posibles salidas de mercado que tendría el dispositivo considerándolo solo como un sistema de sincronización en tiempo real, seria, ofrecer el dispositivo embebido como servidor de tiempo de bajo coste, tanto energético como económico.

Ahora, realicemos el estudio considerando el dispositivo como un reloj de palabras con sincronización de la hora en tiempo real a través de internet. En este caso si existe en el mercado una alternativa clara a nuestro dispositivo, es un reloj de palabras que muestra la hora con el mismo formato que el construido en este proyecto, y sincroniza la hora mediante señales de radio.

Figura 20: QLOCKTWO CLASSIC

Este dispositivo es el QLOCKTWO CLASSIC del fabricante alemán BIEGERT&FUNK (7) (Figura 20). Realiza la sincronización horaria mediante señales de radio y cuenta con un sistema multilenguaje que permite traducir el reloj a diferentes idiomas simplemente cambiando el panel frontal. El precio de venta de este reloj parte de 985€.

3. Descripción funcional.

Una vez introducido el proyecto y analizados los antecedentes, se pasa a explicar de manera sencilla como está compuesto el dispositivo embebido y que funcionalidades implementa para conseguir cumplir con los obtenidos que se le exigen.

3.1 Reloj de palabras con sincronización en tiempo real online.

Los objetivos principales que se han marcado para este proyecto son, obtener un dispositivo capaz de sincronizar la hora a través de Internet de la forma más precisa posible y poder mostrar ese valor obtenido en un reloj de palabras. Se ha diseñado un sistema embebido basado en la placa de desarrollo LPCXpresso LPC1769 que utiliza el sistema operativo en tiempo real FreeRTOS y que cuenta con un módulo inalámbrico Wifly para dotarlo de acceso a Internet, un reloj en tiempo real DS3231 para poder mantener la hora, un reloj de palabras para mostrar la hora y un convertidor CP2102 para enviar los registros de actividad (Figura 21).

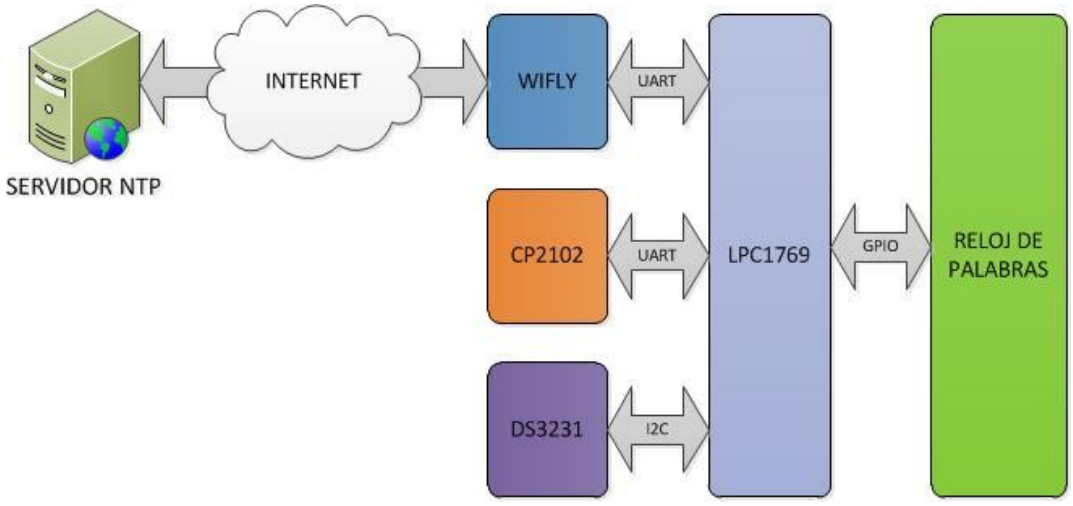


Figura 21: Diagrama de bloques básico del dispositivo.

Para lograr estos objetivos mediante el uso de los componentes mencionados, el sistema cuenta con las siguientes funcionalidades:

- **Gestión de la conexión a Internet.**

  Para poder realizar las comunicaciones con los servidores de tiempo, el sistema cuenta con un módulo inalámbrico Wifly, gestionado a través de un puerto de comunicación serie (UART) del LPC1769. Gracias a esto, el dispositivo embebido puede realizar conexiones a Internet a través de redes Wi-Fi y puede comprobar el estado de la conexión, de forma que si pierde la conexión a Internet, automáticamente, realiza reintentos de conexión hasta que vuelve a tener acceso a la red.

  Predeterminadamente, el dispositivo se conecta a Internet cada vez que se inicia, pero, siempre que vaya a intentar sincronizar el tiempo con un servidor de tiempo de Internet, hará una comprobación de la conexión. Así se asegura de realizar las peticiones solo cuando exista conexión a Internet.

  Además, cuando no es necesario sincronizar el tiempo online, el dispositivo pone el módulo Wifly en modo reposo (Figura 22). Así, el consumo de potencia del dispositivo, relacionado con la conexión a Internet será siempre el mínimo posible.

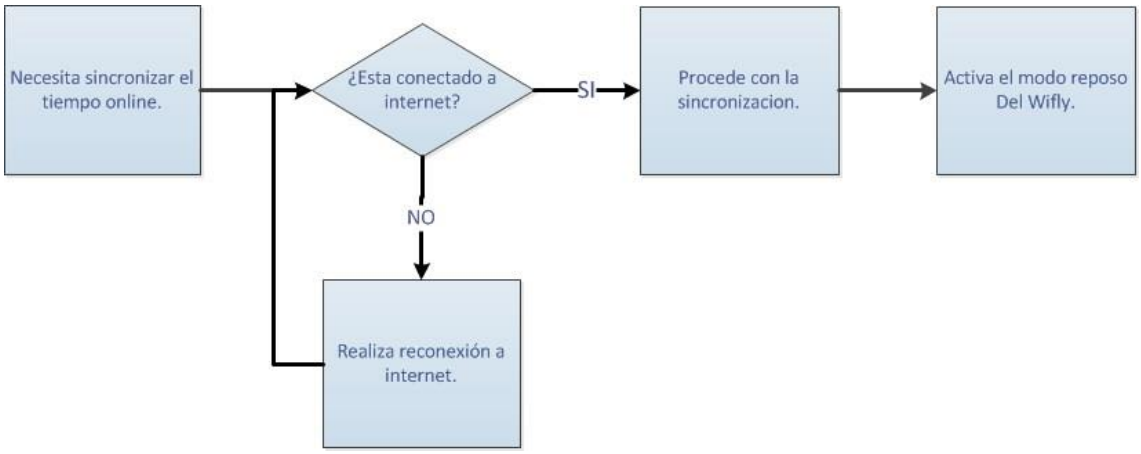


Figura 22: Gestión de la conexión a Internet, diagrama simple.

- **Obtención de la hora a través de servidores de tiempo NTP y almacenamiento en el DS3231**

  Para obtener el valor de la hora actual de un servidor de tiempo conectado a Internet, el dispositivo debe utilizar el protocolo Network Time Protocol (NTP) que trabaja sobre el protocolo de transporte UDP, para ello crea un paquete NTP utilizando la estructura de

datos definida por el protocolo (Figura 23) y lo envía al servidor de tiempo definido en la configuración del dispositivo. Una vez realizada la petición, el servidor de tiempo contestara con otro paquete NTP, que tendrá la misma estructura que el paquete enviado, y que contendrá el valor de la hora actual en el estándar de tiempo Coordinated Universal Time (UTC)

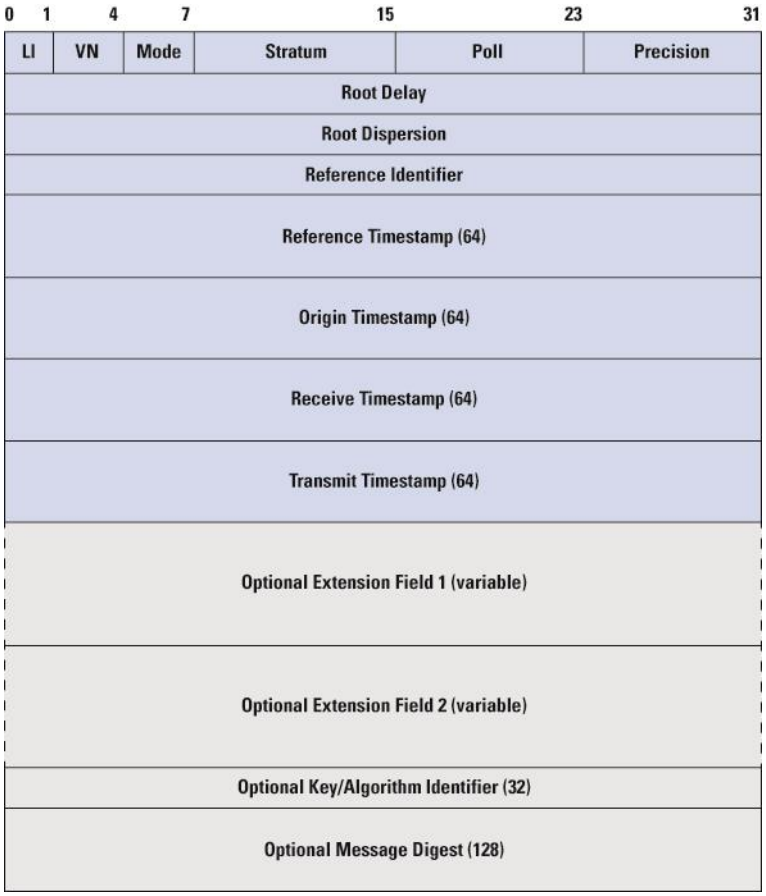


Figura 23: Estructura paquete NTP

Como alternativa a este procedimiento, y usada en fase de desarrollo, el sistema tiene implementada otra forma de obtener el tiempo a través de Internet. En ella, el dispositivo utiliza una funcionalidad integrada en el módulo Wifly que permite obtener la hora actual de una forma más sencilla y abstraída para él.

Utilizando cualquiera de las dos formas descritas para obtener el tiempo desde un servidor NTP, siempre hay que tener en cuenta que el valor de la hora devuelto por el servidor, estará en formato UTC, por lo que habrá que ajustar la hora a la zona horaria en la que se encuentra el dispositivo. En el caso de este proyecto se ha predefinido la zona horaria UTC/GMT + 1 horas, que es la correspondiente a España. El sistema también tiene en cuenta el horario de verano o Daylight Saving Time (DST) que se aplica en España,

actualmente empieza a las dos de la madrugada del último domingo del mes de Marzo, y acaba a las dos de la madrugada del último domingo del mes de Octubre.

Por último, intentando conseguir el mayor ahorro de recursos y de energía posible, el sistema controla que solo se realice el proceso de almacenamiento de un valor nuevo de la hora en el DS3231, cuando ese valor ya ha sido obtenido y procesado mediante los procedimientos anteriores (Figura 24). Antes de almacenarlo, solo le quedara por hacer el reajuste del desfase que se describe en la funcionalidad de control de desfase de la hora.
Todo este proceso de sincronización de la hora a través de Internet está configurado para que se ejecute cada cierto intervalo de tiempo, en este caso, y para facilitar la etapa de desarrollo se ha decidido que el dispositivo realizara intentos de sincronización cada treinta segundos.

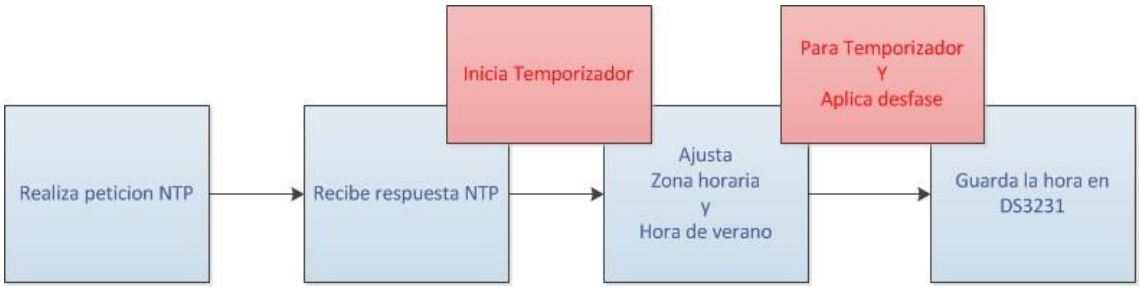


Figura 24: Sincronización de la hora, diagrama simple.

- **Control del desfase de sincronización de la hora.**

Uno de los procesos más importantes del sistema es el control del posible desfase de tiempo que se produce desde que obtiene la hora del servidor de tiempo, hasta que la almacena en el reloj en tiempo real, DS3231 (Figura 25). El sistema, por muy eficiente que sea, siempre va a tardar unas milésimas de segundo en obtener el paquete NTP desde el puerto UART del Wifly, extraer la hora, ajustar la zona UTC/GMT, ajustar el horario de verano, convertir el formato de la hora al requerido por el DS3231 y enviar el valor a través del puerto I2C que utiliza el RTC para que lo almacene. Si no se tuviese en cuenta ese desfase, la hora que almacena el sistema estaría retrasada con respecto a la hora real.

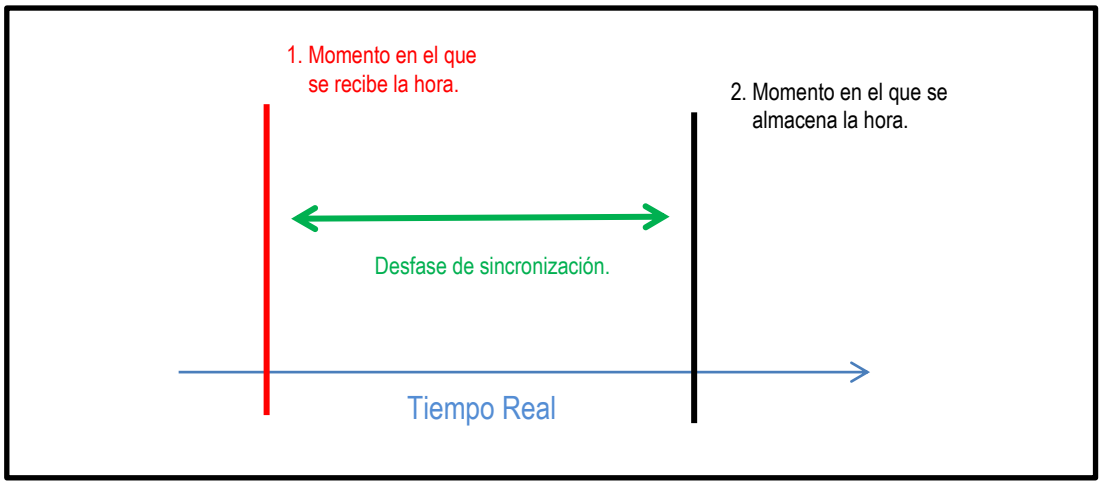


Figura 25: Ejemplo desfase de sincronización.

Para solucionarlo el sistema inicializa un temporizador justo en el momento en el que recibe el paquete con la hora actual desde el servidor NTP y lo detiene justo antes de almacenarlo en el DS3231. Una vez detenido, el sistema reajusta el valor de la hora a guardar, sumándole el tiempo que ha estado en marcha el temporizador (Figura 24). Aquí surge otro pequeño problema, la precisión que ofrece el temporizador del LPC1769 es superior a la precisión del módulo DS3231, el primero devuelve valores de millonésimas de segundo y DS3231 acepta segundos como unidad de tiempo más pequeña. Para solucionarlo, el sistema redondea el tiempo de desfase a la alza, cuando las décimas de segundo son iguales o superiores a cinco, y a la baja si son inferiores. Con este ajuste conseguimos que el desfase máximo de la hora de nuestro sistema, con respecto a la hora real, sea menor o igual a medio segundo.

- **Mostrar la hora en el reloj de palabras.**

El panel de leds construido para mostrar la hora en forma de reloj de palabras, está diseñado de tal manera que se pueden controlar todas las palabras e indicadores mediante los puertos GPIO del LPC1769, en concreto son necesarios veintiocho puertos de la placa. Cuando el sistema quiere mostrar la hora a través del panel de leds, obtiene el valor de la hora directamente del módulo DS3231, lo analiza y decide que palabras e indicadores tiene que iluminar para que se pueda leer la hora en el panel de leds y finalmente procede a activar los puertos GPIO seleccionados (Figura 26). Este proceso es totalmente independiente del sistema de sincronización de la hora online. El valor almacenado en el RTC siempre será el de referencia para mostrar la hora, ya que el sistema siempre considerara el tiempo del DS3231 como correcto.

Figura 26: Mostrar la hora, diagrama simple.

- **Registro de la actividad vía puerto serie.**

  Para cerciorarse del correcto funcionamiento del dispositivo, es necesario contar con un registro donde el usuario pueda ver el funcionamiento y el estado de la ejecución del sistema, algo de gran utilidad en el proceso de desarrollo. El problema es que el único interfaz de usuario del que dispone el dispositivo, es el reloj de palabras, y este tiene una funcionalidad muy definida como para poder mostrar toda la información del registro. Esto hace que el sistema necesite de otro interfaz en el que se pueda ver de forma cómoda el registro de actividad.

  Para solucionar esto, el sistema utiliza un puerto de comunicaciones serie (UART) del LPC1769, junto con el conversor USB/UART CP2102, para enviar la información que quiere registrar. El usuario, mediante cualquier software de emulación de terminal serie, podrá acceder al registro de actividad del dispositivo.

4. Descripción detallada

4.1 Introducción.

Una vez expuesta la descripción funcional del dispositivo, pasamos a realizar una descripción más detallada y técnica de los componentes que lo forman, así como de la aplicación embebida que lleva a cabo cada una de las funcionalidades descritas en el aparatado anterior. Para ello, primero se analizan en detalle los componentes y el reloj de palabras, después se detallan las librerías que ha sido necesario implementar para que finalmente se expongan las tareas que se llevan a cabo durante la ejecución de la aplicación embebida.

4.2 Descripción de los componentes.

Antes de analizar las funcionalidades del dispositivo a nivel técnico, es necesario conocer en detalle los diferentes componentes de hardware utilizados, así como sus características.

- **LPCXpresso LPC1769.**

  La placa de desarrollo LPC1769 del fabricante Embedded Artists, se basa en la familia de procesadores ARM Cortex-M3 fabricados por NXP y está enfocada al desarrollo de aplicaciones embebidas. El procesador ARM Cortex-M3 de 32 bits, cuenta con 64 Kb de memoria SRAM y 512 Kb de memoria flash y puede trabajar hasta velocidades de 120 MHz. Además, ofrece un sistema mejorado de depuración, compatible con CMSIS-DAP, y un mayor nivel de integración de bloques.

  El diagrama de bloques de la arquitectura de la placa se puede ver en la Figura 27.

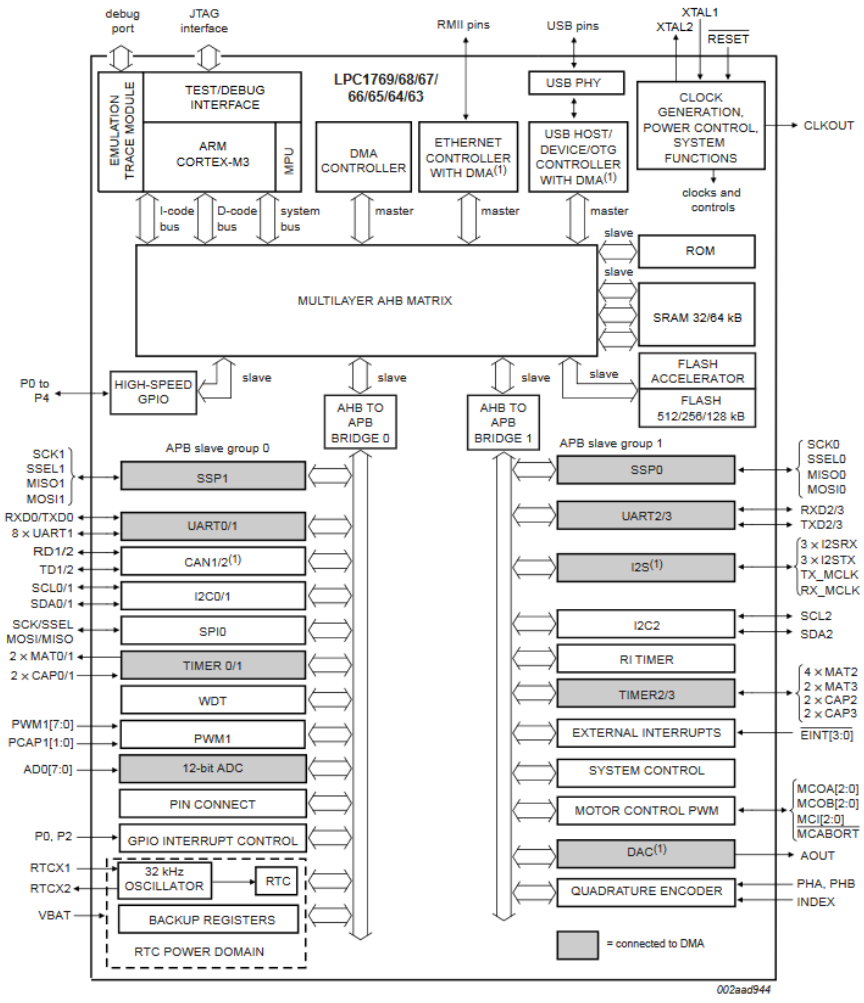


Figura 27: Diagrama de bloques LPC1769

Algunas de las características más relevantes de la placa en relación con las necesidades del proyecto son:

- Alimentación 3,3V (2.4V a 3.6V) para los periféricos.
- Conexión USB 2.0 con controlador DMA dedicado.
- Controlador DMA de propósito general de ocho canales.
- 4 puertos UART.
- 3 interfaces mejorados de bus I2C.
- 4 temporizadores o contadores de propósito general.
- 52 pines de propósito general de entrada y salida o GPIO.

Para conocer el resto de características de la placa se puede consultar su hoja de datos en la web del fabricante (8).

Los pines de la placa LPC1769 utilizados en este proyecto para conectar los diferentes módulos, se detallan en la siguiente tabla (Figura 28). No se han reflejado los pines utilizados para el reloj de palabras, se ha preferido detallarlos en el apartado correspondiente.

Pin	Puerto	Función	Conexión
J6-9	P0.0	UART3 TX	TX CP2102
J6-10	P0.1	UART3 RX	RX CP2102
J6-21	P0.2	UART0 TX	TX Wilfy
J6-22	P0.3	UART0 RX	TX Wifly
J6-25	P0.27	I2C0 SDA	SDA DS3231
J6-26	P0.28	I2C0 SCL	SCL DS3231

Figura 28: Tabla de pines del LPC1769 utilizados

- **Módulo inalámbrico Wifly RN-XV**

Este dispositivo es capaz de gestionar conexiones Wi-Fi que utilicen el protocolo 802.11 b/g. Distribuido por Sparkfun, es una solución de conexión inalámbrica basada en el módulo Wi-Fi RN-171, y estas son algunas de sus principales características:

- Alimentado con 3.3V.
- Procesador de 32 bits.
- Ultra bajo consumo, 4uA en reposo, 39uA en funcionamiento.
- Integración del protocolo TCP/IP.
- Interfaz de comunicación UART.

- 8 GPIOs.
- Reloj en tiempo real integrado.
- Antena integrada.

Podemos ver su diagrama de bloques en la Figura 29.

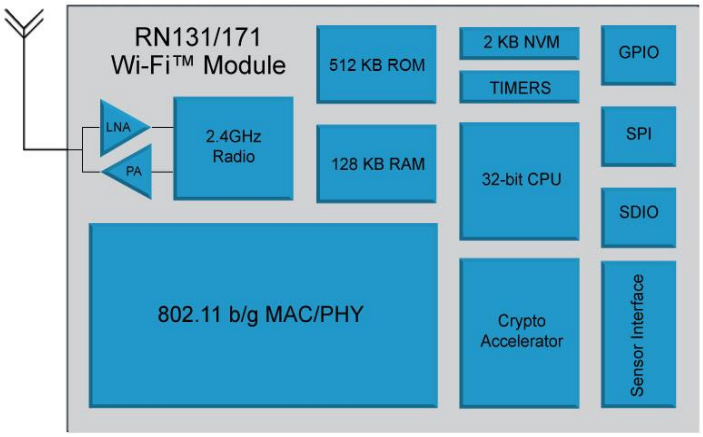


Figura 29: Diagrama de bloques del Wifly RN-XV

Además, el modulo cuenta con un firmware (última versión disponible v4.41) que integra un API que permite, de una forma muy sencilla y a través de UART, gestionar las comunicaciones, así como configurar las diferentes funcionalidades integradas del dispositivo. Para más información podemos ver su manual de usuario en la web del fabricante (9).

- **Reloj en tiempo real con comunicación I2C, DS3231**

El módulo DS3231 de Maxim Inegrated, es un reloj en tiempo real de gran precisión que utiliza el bus I2C. Cuenta con una batería de tres voltios, tipo CR2032, que le permite mantener el tiempo con precisión si el dispositivo principal se queda sin alimentación. El modulo mantiene horas, minutos y segundos de la hora, y de la fecha, día de la semana, día del mes, mes y año. Otras de sus principales características son:

- Corrección automática del último día de mes para meses con menos de 31 días.
- Corrección de año bisiesto.
- Precisión de ±2ppm entre 0ºC y 40ºC
- Sensor de temperatura con precisión de ±3ºC
- Modo de bajo consumo con batería.

El diagrama de bloques que ilustra el diseño del módulo es el siguiente (Figura 30):

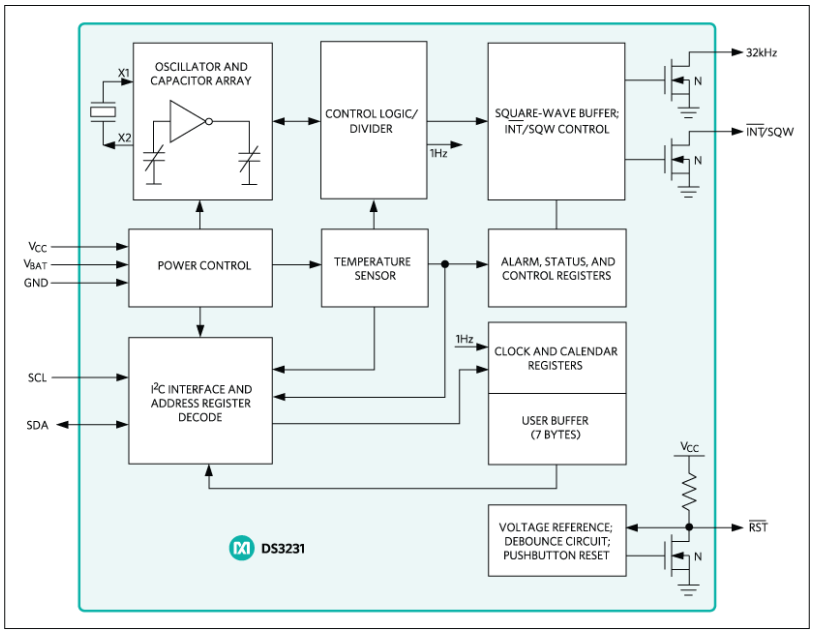


Figura 30: Diagrama de bloques DS3231

La estructura de los registros del DS3231 donde se almacenan los valores de la fecha y la hora se puede ver en la Figura 31.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Figura 31: Estructura registros DS3231

Para conocer el resto de sus especificaciones podemos consultar su hoja de datos (10)

- **Convertidor USB a UART CP2102**

  Este módulo hace de controlador puente entre una conexión USB y una conexión UART, permitiendo enviar y recibir datos fácilmente, desde ambos extremos. Mediante el uso de un controlador gratuito disponible para casi todos los sistemas operativos actuales, crea un puerto serie virtual en el ordenador, que nos permite utilizar cualquiera de las aplicaciones de comunicación serie que existen. Para conocer en más detalle el módulo CP2102 podemos ver su hoja de datos en la página web del fabricante (11), su diagrama de bloques se expone a continuación (Figura 32).

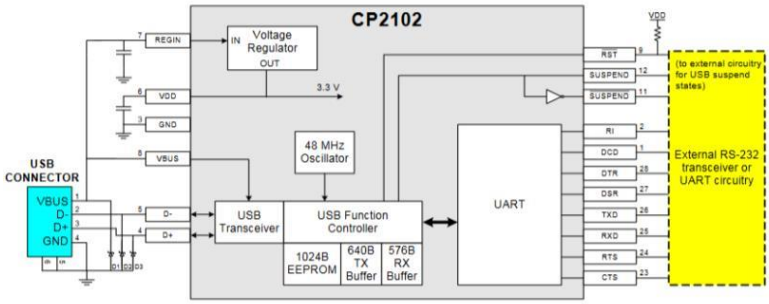


Figura 32: Diagrama de bloques CP2102

4.3 Descripción del reloj de palabras.

Aunque el reloj de palabras puede considerarse como un componente más del dispositivo, su estructura, diseño y funcionamiento hacen que merezca tenerlo en consideración a parte del resto de componentes.

El reloj de palabras que se ha construido para mostrar la hora del dispositivo embebido, está formado por un panel frontal, que consiste en una sopa de letras recortada en negativo sobre un vinilo adhesivo, pegado a una placa de metacrilato y, una caja trasera que contiene una rejilla de contrachapado con ciento diez orificios de cinco milímetros, centrados cada uno, detrás de cada letra de la sopa de letras, en estos orificios están los leds que iluminaran las letras (Figura 33, derecha). La plantilla de sopa de letras que contiene todas las palabras necesarias para poder mostrar la hora de forma transcrita se ha diseñado mediante un software de diseño vectorial (Figura 33, izquierda). Teniendo en cuenta que la construcción del reloj de palabras no entra dentro del ámbito de este proyecto y una vez ya se ha descrito a grandes rasgos, pasamos a la descripción detallada de los circuitos lógicos que contiene el reloj de palabras y que permiten que se ilumine.

Figura 33: Frontal e interior del reloj de palabras.

Para poder gestionar los leds que iluminan las letras que forman las palabras, se ha decido conectarlos de forma agrupada y en cantidades correspondientes a las letras que forman cada palabra, por ejemplo para la palabra “CUATRO” se agrupan seis leds y para la palabra “MEDIA”, cinco leds. De esta forma, reducimos el número de señales que el dispositivo tiene que controlar. Los grupos de leds, están conectados entre sí en paralelo, haciendo que la tensión necesaria para hacerlos funcionar, independientemente del número que lo formen, sea siempre la misma. En contraposición a esto, la intensidad de corriente que necesitan los grupos de leds es mayor de la que facilitan los pines del LPC1769, asunto que se puede solucionar gracias a los driver de potencia ULN2803, más adelante se aborda este problema en detalle.

El panel de leds está alimentado por una fuente de 5V y 2A, como los leds trabajan con 3.2V es necesario ajustar la tensión, para ello utilizaremos resistencias de diferentes valores, teniendo en cuenta que los leds están conectados en grupo y en paralelo (Figura 34).

	Leds y Resistencia		**GPIO**	
Palabra	**Nº de leds**	**Resistencia**	**Puerto**	**Pin**
ES	2	100 Ohm	J6.52	P2.11
SON	3	33 Ohm	J6.53	P2.12
LA	2	47 Ohm	J5.51	P2.10
S	1	100 Ohm	J6.50	P2.8
UNA	3	33 Ohm	J6.49	P2.7
DOS	3	33 Ohm	J6.48	P2.6
TRES	4	27 Ohm	J6.47	P2.5
CUATRO	6	15 Ohm	J6.46	P2.4
CINCO (hora)	5	18 Ohm	J6.45	P2.3

SEIS	4	27 Ohm	J6.44	P2.2
SIETE	5	18 Ohm	J6.43	P2.1
OCHO	4	27 Ohm	J6.42	P2.0
NUEVE	5	18 Ohm	J6.41	P0.11
DIEZ (hora)	4	27 Ohm	J6.40	P0.10
ONCE	4	27 Ohm	J6.39	P0.5
DOCE	4	27 Ohm	J6.38	P0.4
Y	1	100 Ohm	J6.7	P0.7
MENOS	5	18 Ohm	J6.20	P1.31
VEINTE	6	15 Ohm	J6.19	P1.30
DIEZ (minutos)	4	27 Ohm	J6.18	P0.26
VEINTI	6	15 Ohm	J6.17	P0.25
CINCO (minutos)	5	18 Ohm	J6.16	P0.24
MEDIA	5	18 Ohm	J6.15	P0.23
CUARTO	6	15 Ohm	J6.14	P0.16
Indicador minuto 1	1	100 Ohm	J6.13	P0.15
Indicador minuto 2	1	100 Ohm	J6.12	P0.17
Indicador minuto 3	1	100 Ohm	J6.11	P0.18
Indicador minuto 4	1	100 Ohm	J6.8	P0.6

Figura 34: Tabla de palabras en el reloj de palabras.

Se puede observar en la Figura 34, que se ha considerado la letra “S” y la palabra “VEINTI” como palabras independientes aunque por si solas no tengan sentido, esto es debido a que, en el caso de la letra “S” la podemos usar conjuntamente con “LA” para las frases que necesiten el plural del articulo (“Son la una”, ”Son la-s dos”), el caso de la palabra “VEINTE” es parecido, aquí, se usa para complementar la palabra “CINCO” que indica minutos (“Es la una y cinco”, ”Es la una y veinte-cinco”). En la Figura 35 podemos ver la cantidad de resistencias y el valor de las mismas, que serán necesarias para todo el panel.

Valor resistencia	Resistencias necesarias
100 Ohm	6
47 Ohm	2
33 Ohm	3
27 Ohm	7
18 Ohm	6
15 Ohm	4
Total:	**28**

Figura 35: Tabla de resistencias necesarias por valor.

Como se ha mencionado, la intensidad de corriente necesaria para iluminar los grupos de leds, es mayor de la obtenida a través de los pines de salida de la placa LPC1769. Para solucionarlo, se utiliza el driver de potencia ULN2803, este IC es capaz de acondicionar señales digitales de baja intensidad de tal manera que puede alimentar componentes que necesitan corrientes superiores a las proporcionadas por la señal, en concreto se obtiene una intensidad de 500 mA por salida y cuenta con ocho, pudiendo controlar con un solo ULN2803 ocho señales (Figura 36). Por lo que con cuatro unidades (8 x 4 = 32 señales) se podrán iluminar, con la suficiente intensidad, todos los grupos de leds que forman el panel.

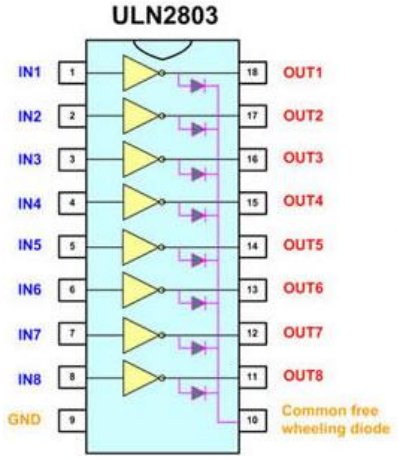


Figura 36: Diagrama ULN2803A

Finalmente, para que el LPC1769 pueda controlar el panel de leds, se han utilizado los puertos GPIO del microcontrolador, siendo necesarios tantos como palabras tiene el reloj, veintiocho. De esta forma cuando el dispositivo necesite iluminar una palabra activara el puerto GPIO que le corresponda. La señal llegara a un ULN2803 que la dejara pasar, aumentando la intensidad, y llegara al grupo de leds que iluminaran la palabra. En la Figura 34 podemos ver la correspondencia entre los puertos GPIO y cada palabra del reloj.

Aunque el reloj de palabras es una forma muy atractiva de mostrar la hora, durante el proceso de desarrollo se consideró necesario crear una versión más simple del mismo para trabajar con más comodidad. Esta, está construida sobre una placa de prototipos que contiene veintiocho leds colocados en línea, cada led está conectado directamente a un puerto GPIO, además sobre la placa hay una plantilla donde se pueden ver la palabra que representa cada led (Figura 14).

4.4 Esquema de conexiones del dispositivo.

Una vez descritos tanto los componentes como el reloj de palabras, para realizar un resumen, que mejor que ilustrar mediante un diagrama de bloques como están interconectados todos los componentes (Figura 37).

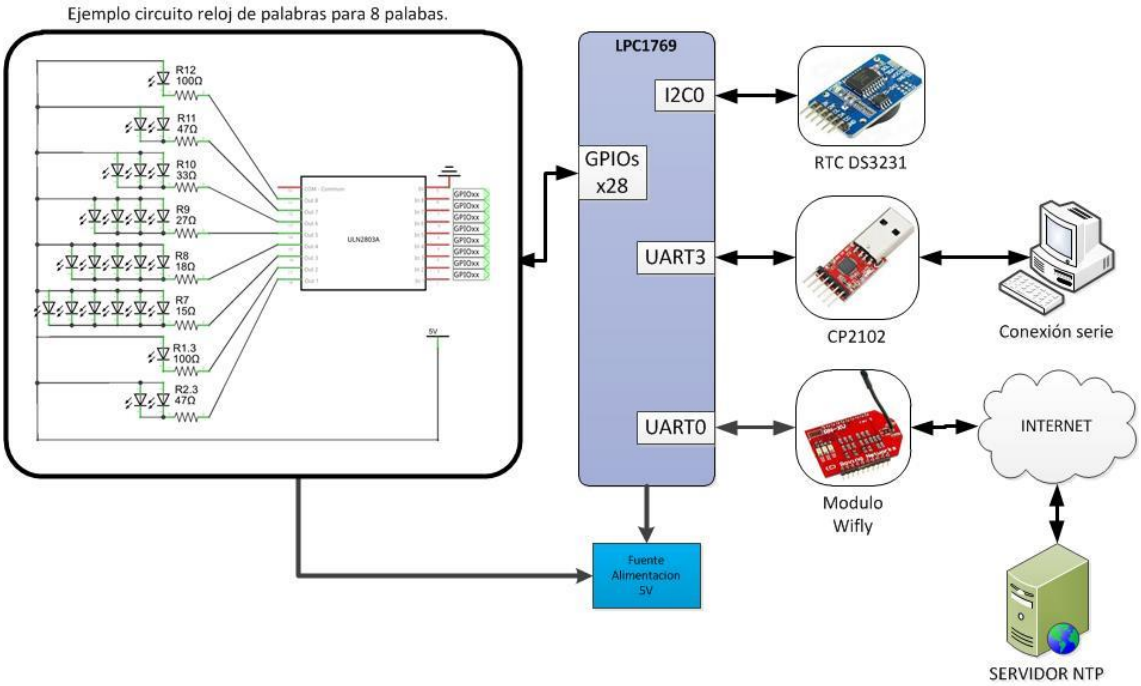


Figura 37: Diagrama de bloques del dispositivo.

4.5 Descripción de la aplicación embebida

En este apartado se realiza una descripción detallada de las librerías y tareas que contiene la aplicación, que le permiten llevar a cabo y controlar la ejecución de las funcionalidades del sistema.

Antes de empezar es necesario explicar en qué consiste el sistema operativo en tiempo real FreeRTOS y algunas de sus funcionalidades, ya que han sido necesarios para el desarrollo del proyecto. Recalcar la ayuda obtenida gracias al libro “Using the FreeRTOS Real Time Kernel” (12) que ha sido imprescindible para el desarrollo de la aplicación embebida.

FreeRTOS es un sistema operativo en tiempo real de código libre. Se ha hecho muy popular en el desarrollo de sistemas embebidos ya que es compatible con un gran número de microcontroladores y plataformas de desarrollo. Este sistema operativo está enfocado a la computación orientada a tareas, para lo que cuenta con un planificador de tareas que se encarga de gestionar la ejecución eficiente de las tareas que se implementan, esto le permite que aunque realmente solo ejecute una tarea en cada momento, parezca que el sistema es

multitarea. Para ello el FreeRTOS integra un sistema de estados y transiciones de las tareas (Figura 38).

Como se ha indicado el S.O. utiliza tareas para gestionar la ejecución de la aplicación, estas, son en sí mismas pequeños programas que ejecutan diferentes funciones, normalmente se diseñan para que se ejecuten continuamente mediante un bucle infinito. Cuando son creadas, se les puede definir, además de otras características, la prioridad que tendrán. Esto, permite al S.O. tener un orden de prioridades a la hora de ejecutar una u otra tarea.

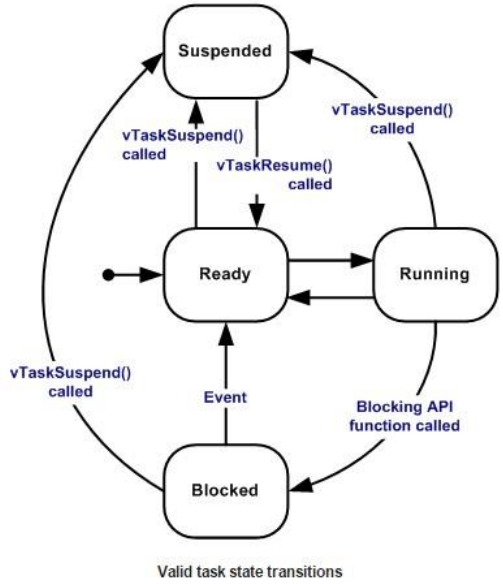


Figura 38: Diagrama de estados FreeRTOS.

Para la implementación de la aplicación, también ha sido útil la posibilidad de utilizar las colas de FreeRTOS. Mediante ellas podemos enviar y recibir datos de una tarea a otra, todo gestionado por el S.O.

Por último, también ha sido muy importante en la programación de las librerías que forman la aplicación, el uso de los semáforos de FreeRTOS, ya que estos permiten controlar el acceso a funciones desde las tareas. Lo que hacen es bloquear el uso de una función por parte de una tarea hasta que esta ha finalizado, evitando el uso concurrente por parte de varias tareas. En este proyecto se han utilizado los semaforos para controlar que la comunicación entre los diferentes módulos del dispositivo solo se realice a través de una tarea a la vez.

Una vez introducido el sistema operativo, base para el sistema, se procede a hacer la descripción detallada de las librerías implementadas para realizar el control de los puertos de la placa LPC1769:

- **Controladores de los puertos UART:**

  - **“uart”:** Este driver es el encargado de controlar y manejar la comunicación a través de los puertos UART de la placa LPC1769. Cuenta con las funciones de control de las interrupciones para los puertos UART0, UART1, UART3. Además de las de funciones de inicialización de los puertos, envío de datos y limpieza de los buffers.

  - **“UARTprint”:** Este controlador es una abstracción del anterior driver que permite que, tanto la escritura como la lectura a través de los puertos UART sean más sencillas a la vez que completas. Cabe descartar las siguientes funciones:

    - UARTPrint_Text. Esta función permite enviar cadenas a través de puertos UART de forma simple, además permite el uso de variables en las cadenas, esto hace que sea muy útil para enviar cualquier tipo de valor.

    - UARTPrint_Text_LF. Realiza lo mismo que esta función pero añadiendo un retorno de carro y un fin de línea al final de cada envio.

    - UART_CheckRead. Esta función puede realizar lecturas de datos a través de UART, comprobando si en los datos obtenidos existe un valor pasado como parámetro. Es muy útil para comprobar que lo que se obtiene es lo que se estaba esperando.

    - UART_Read_NTP_Packet. Implementada específicamente para realizar las lecturas de paquetes NTP a través de UART. Trata los datos obtenidos como enteros en lugar de como caracteres para obtener correctamente todos los bytes de los paquetes NTP recibidos del servidor de tiempo.

    UARTprint es una librería muy importante para este proyecto, ya que la mayoría de datos del sistema fluyen por ella.

- **Controlador de los puertos I2C.**

  - **“I2C”:** Se encarga de manejar la comunicación de los puertos I2C, en este caso solo se ha implementado el uso del puerto I2C0. Como no han sido necesarias para el proyecto, no cuenta con el manejo de las interrupciones, pero si es posible enviar y recibir datos a través de este bus de comunicaciones.

- **Controlador de los puertos GPIO.**

    - **“GPIO”:** Librería que permite la configuración los puertos GPIO. Implementa la forma en que hay que configurar un pin de la placa LPC1769 para que pueda funcionar como GPIO de entrada o de salida. Además, cuenta con una función para la escritura sobre los puertos GPIO, muy útil para manejar con facilidad los pines GPIO de la placa y básica para la comunicación con el reloj de palabras.

Una vez expuestos los controlados de los puertos de la placa LPC1769, seguidamente se describen las librerías implementadas para el manejo de los módulos utilizados para este proyecto, aquí también se incluye la descripción del controlador de los temporizadores que la placa tiene integrados.

- **Controlador del módulo RTC DS3231.**

    - **“DS3231”:** Cuenta con las funciones que permiten el control del reloj en tiempo real DS3231, puede obtener el valor de fecha y hora almacenadas en el RTC, así como, escribir un nuevo valor en él, para ello escribe y lee mediante el driver I2C los registros específicos del DS3231 (Figura 31). Para gestionar los valores de fecha y hora, utiliza el tipo de dato Timedate, que ha sido creado expresamente para ello y se explicara en detalle más adelante.

- **Controlador del módulo Wifly.**

    - **“WIFLY”:** Para implementar el control sobre el modulo inalámbrico, se ha querido abstraer la comunicación (lectura y escritura) con el modulo, de las funciones que podríamos denominar genéricas, es decir, la librería cuenta con unas funciones públicas que realizan trabajos que serían necesarios para la correcta sincronización horaria fuera cual fuera el modulo que se utilizase para dar conexión a internet al dispositivo, y cuenta con funciones privadas que se encargan de realizar las comunicaciones adaptándose a los requisitos del módulo Wifly.

      La comunicación con este módulo se realiza mediante el envio de comandos a través de un puerto UART, y la recepción de datos mediante el mismo puerto. Para el desarrollo de las funciones privadas, se han considerado por un lado, los comandos que necesitan parámetros que puede ser variables, como la definición del nombre de la red Wi-Fi (SSID) a la hay que conectarse, y por otro lado, los comandos que no necesitan de parámetros variables o son especiales, como el comando “$$$”, que tiene que ser enviado sin los caracteres de fin de línea. De esta forma se ha conseguido disminuir el número de funciones necesarias para la comunicación con el Wifly. Indicar

también que todas las funciones que envían comandos, están implementadas para comprobar la respuesta del módulo, comparándola con el valor esperado que ha sido pasado como parámetro. De estas funciones privadas cabe destacar las siguientes:

- _WIFLY_Send_Command. Esta función es la que se utiliza para los comandos que no necesitan de parámetros variables. Al pasar a la función el comando y la respuesta esperada, la función devolverá si el comando se ha procesado correctamente o no.

- _WIFLY_reset. Se ha conectado el pin RESET del módulo Wifly a un puerto GPIO de la placa LPC1769, esto permite que esta función, mediante el uso del controlador GPIO, resetee el dispositivo. Es muy útil porque permite evitar estados no controlados al inicializar del módulo.

De las funciones públicas que realizan tareas genéricas sobre el módulo Wifly, hay que destacar las siguientes:

- WIFLY_CheckConnection. Es la encargada de comprobar si el módulo Wifly está conectado correctamente a internet. Para ello mediante el envío del comando "show connection" y la lectura de la respuesta correspondiente, se recibe el valor del registro que contiene el estado de conexión actual del módulo (Figura 39), todo esto comprobando previamente si el modulo se encuentra en reposo, y en tal caso, reactivandolo. Para simplificar la lectura de los valores del registro, se ha comprobado que si el valor del registro es igual a "8730" (valor en decimal) el modulo se encuentra conectado a la red, por lo que si la respuesta al comando es diferente a ese valor, se considera que no está conectado a Internet.

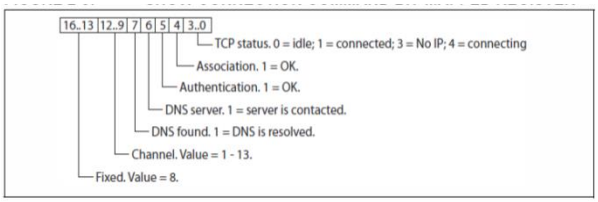


Figura 39: Registro de estado de conexión del módulo Wifly.

- WIFLY_Connect_WLAN. Esta función es la encargada de que el módulo Wifly se conecte a internet a través de una la red Wi-Fi. Pasándole como parámetro el nombre de la red Wi-Fi, la contraseña y un valor que define el tipo de seguridad que utiliza la red (solo se han configurado dos posible tipos, WPA y WEP), la función, restablece el módulo Wifly a los parámetros de fábrica, lo reinicia y le envía todos los comandos necesarios para que se conecte a la red indicada y este configurado correctamente para el envío y recepción de paquetes UDP.

- WIFLY_ConfigNTP. Encargada de configurar y guardar en el módulo la dirección IP y el puerto del servidor NTP al que queremos realizar peticiones.

- Wifly_NTP_request. Esta función es la encargada de realizar una tarea crucial para el funcionamiento del dispositivo embebido. Prepara el paquete de solicitud siguiendo la estructura definida por el protocolo NTP (Figura 23), lo envía al servidor a través del módulo Wifly utilizando el protocolo UDP y recibe la respuesta del servidor. Si este proceso ha sido correcto, del paquete recibido se extrae el valor de la hora que el servidor ha devuelto, que está en formato Timestamp y contiene los segundos que han pasado desde el 1 de Enero de 1900, como el resto de la aplicación utiliza como valor de referencia para los Timestamp el 1 de Enero de 1970, la función ajusta el valor del Timestamp obtenido a esa referencia. Finalmente si toda la ejecución ha sido correcta devuelve el Timestamp con la hora del servidor y si no ha sido correcta, la función devuelve cero. Justo en el momento anterior de enviar la petición al servidor, la función inicia un temporizador que después servirá para calcular el desfase de sincronización.

- **Controlador de los temporizadores (TIMERS).**

  - **"Timer":** Esta librería, como en el caso del driver I2C, solo implementa el uso de un temporizador de los cuatro disponibles en la placa LPC1769. Es capaz de inicializarlo, y una vez hecho, cuenta con funciones para arrancarlo, pararlo y como no, para obtener el valor del temporizador en cualquier momento. Para evitar problemas, cada vez que se inicia el temporizador, el valor del contador es puesto a cero.

- **Controlador del módulo CP2102.**

  Para manejar el CP2102 no ha sido necesario el desarrollo de una librería específica, ya que con el uso de los controladores de los puertos UART es suficiente, además, el modulo se utiliza solo para enviar cadenas de texto a través de un puerto UART mediante la librería "Debug" que se detalla más adelante.

Ahora que se han descrito en detalle todas las librerías que controlan las comunicaciones del dispositivo embebido, pasamos a detallar las librerías que realizan funciones específicas o cuentan con funciones útiles para el resto de librerías.

- **Herramientas para tratar valores de tiempo.**

  - "**TimeTools**": Contiene herramientas para tratar los diferentes tipos de datos que almacenan valores de tiempo y que se utilizan en la aplicación. Como se indica más arriba, para leer y escribir en los registros del DS3231 se ha creado un tipo de dato personalizado, Timedate, cuyo objetivo es facilitar la lectura y escritura de los valores de la fecha y hora en el DS3231, este tipo de dato se adapta a la estructura de los registros para hacer más sencillo y rápido el proceso. La estructura de Timedate está definida en el fichero de cabeceras "TimeTools.h".

    De las funciones implementadas en esta librería, merecen ser comentadas las siguientes:

    - Time_TimestamptoTimedate. Función que convierte un valor en formato Timestamp a formato Timedate, necesaria para ajustar los valores que se obtiene de los servidores NTP a la estructura de los registros del DS3231.

    - Time_adjustTimeZoneDST. Ajusta sobre la hora obtenida, la zona horaria y el horario de verano de España. Trabaja con valores en formato Timestamp.

    - Time_adjunstSyncDelay. Devuelve el valor obtenido de redondear el valor del desfase de sincronización de picosegundos a segundos y sumárselo al valor Timestamp, ambos pasados como parámetros. Función imprescindible para ajustar la precisión del sistema.

- **Herramienta para convertir un valor de tiempo en señales digitales.**

  - **"LedDisplay":** Esta librería es la encargada de convertir un valor de tiempo en formato Timedate en las señales digitales que se envían a través de los puertos GPIO de la placa LPC1769 que están conectados al circuito del reloj de palabras. En concreto, analiza el valor de la hora y escribe un uno en los puertos GPIO que corresponden a las palabras que transcriben ese valor. Este proceso lo realiza la función LedDisplay_showTime.

- **Herramientas para el registro de actividad.**

  - "**Debug**": Dispone de una función que permite enviar texto con formato a través del puerto UART al que está conectado el CP2102, funciona tratando los parámetros que se le pasan como variables, para después concatenarlos y enviar el resultado a través del puerto UART, utilizando el controlador "UARTprint".

Una vez descritos los controladores y las librerías que forman parte de la aplicación embebida, solo nos queda detallar las tres tareas que gestionan estos recursos para ejecutar las funcionalidades descritas en el capítulo tres.

- **Tarea "Task_SyncTimeOnline".**

  Esta tarea es la más compleja de todas, ya que se encarga de realizar los procesos de gestión de conexión a internet, obtención del tiempo desde un servidor de tiempo y ajuste del desfase de sincronización. La primera vez que se ejecuta esta tarea, realiza una conexión a Internet y configura el servidor de tiempo mediante las funciones de la librería "Wifly", una vez hecho, se inicia un bucle en el que lo primero que hace es comprobar el estado de la conexión a Internet. Si el dispositivo está conectado correctamente, procede a realizar la comunicación con el servidor NTP, seguidamente, si este último paso ha resultado correcto, la tarea obtiene el valor del temporizador que se ha iniciado durante el proceso de comunicación (funcion "Wifly_NTP_request"), y ajusta el valor de la hora obtenido con la función "Time_adjunstSyncDelay", para después, poner el valor de la hora definitivo en la cola de tiempo y activar el modo reposo del Wifly. La cola de tiempo, es una cola que se ha creado específicamente para gestionar el traspaso entre tareas del valor del tiempo actualizado. Si el proceso de comunicación con el servidor NTP no ha funcionado correctamente, se disminuye el tiempo en el que se volverá a reintentar la sincronización. Si el dispositivo no está conectado a Internet, se vuelve a intentar la conexión a Internet y se reduce también el tiempo de reintento de sincronización.

  Una vez realizado todo este proceso la tarea pasa ha estado bloqueada durante un tiempo, que dependerá del resultado de la comprobación de conexión y la sincronización, para después volver al principio del bucle. Podemos ver el diagrama de flujo de la tarea en la Figura 41.

- **Tarea "Task_UpdateTimeRTC".**

  Esta tarea es la encargada de almacenar en el reloj en tiempo real DS3231, el valor de la hora obtenido a través de Internet. Para ello, la tarea entra en un bucle infinito, en el que en cada ciclo comprueba si la cola de tiempo contiene algún valor o no. Si contiene algún valor, quiere decir que la tarea "Task_SyncTimeOnline" ha sincronizado, obtenido y ajustado el tiempo correctamente, por lo que procede a almacenarlo en el DS3231, ademas de registrar el proceso en el Debug. Si no existe ningún valor, simplemente, sigue dando vueltas al bucle infinito hasta que llegue un valor a la cola de tiempo.

Para que el tiempo que pasa desde que el valor de la hora es puesto en la cola y recogido por esta tarea, sea el mínimo posible, la prioridad de esta tarea es la mayor de todas, así siempre será la primera en ejecutarse. Para ilustrar el funcionamiento tenemos la Figura 42.

- **Tarea "Task_ShowTimeOnDisplay".**

  Esta tarea es la encargada de mostrar el tiempo que hay almacenado en el DS3231 a través del reloj de palabras. Al igual que las otras tareas, cuando se inicia entra en un bucle infinito, que este caso, antes de volver a empezar un ciclo, deja la tarea en estado bloqueado durante un segundo. En cada vuelta del bucle obtiene la hora almacenada en el RTC y mediante la función "LedDisplay_showTime" ilumina en el panel de leds las palabras que correspondan. Como el reloj de palabras tiene precisión de minutos, no es necesario actualizar el panel cada segundo, por lo que, en cada ejecución comprueba si realmente es necesario actualizar el tiempo o no, revisando si el minuto ha cambiado desde la última vez que se comprobó. El diagrama de flujo de esta tarea se puede ver en la Figura 43.

Para finalizar la descripción detalla, solo queda comentar que lo primero que realiza la aplicación embebida cuando se arranca es ejecutar la función "main.c" en la que se inicializan todos los componentes del sistema y se crean tanto las tareas anteriores descritas como la cola donde se gestiona el tiempo obtenido (Figura 40). Además, se ha creado un fichero de cabeceras ("Config.h") que incluye las definiciones de los valores más importantes para la ejecución de la aplicación, como por ejemplo:

- Activar o desactivar el nivel de detalle del registro de actividad.
- Definir el puerto UART y el Baudrate de los módulos que utilizan estos puertos.
- Definir el intervalo de tiempo en el que el dispositivo realiza sincronizaciones de tiempo.
- Definir la dirección IP y puerto del servidor NTP.

A continuación se exponen los diagramas de flujo de las tres tareas mencionadas anteriormente y del proceso de inicio de la aplicación:

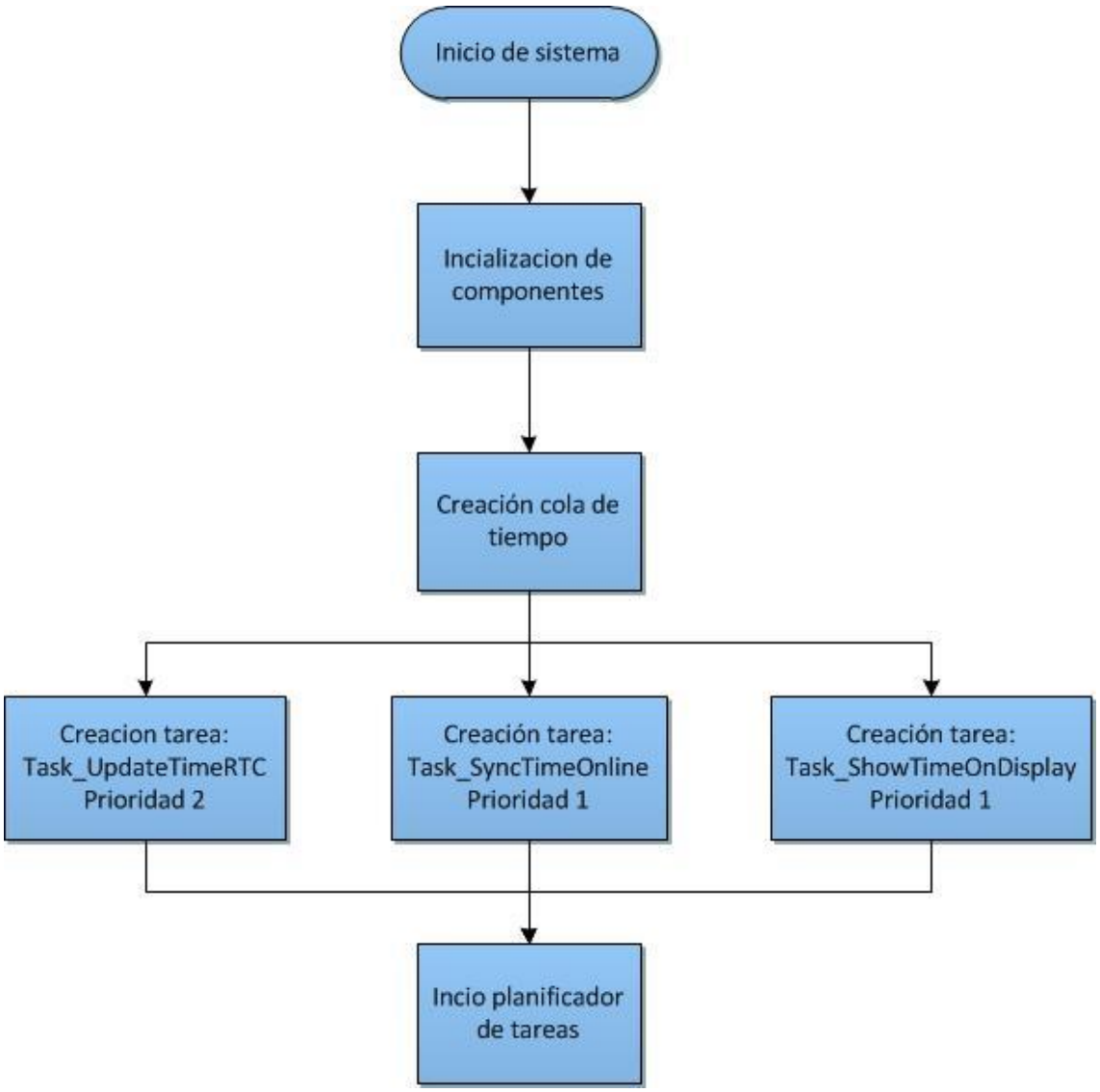


Figura 40: Diagrama de flujo del inicio del sistema.

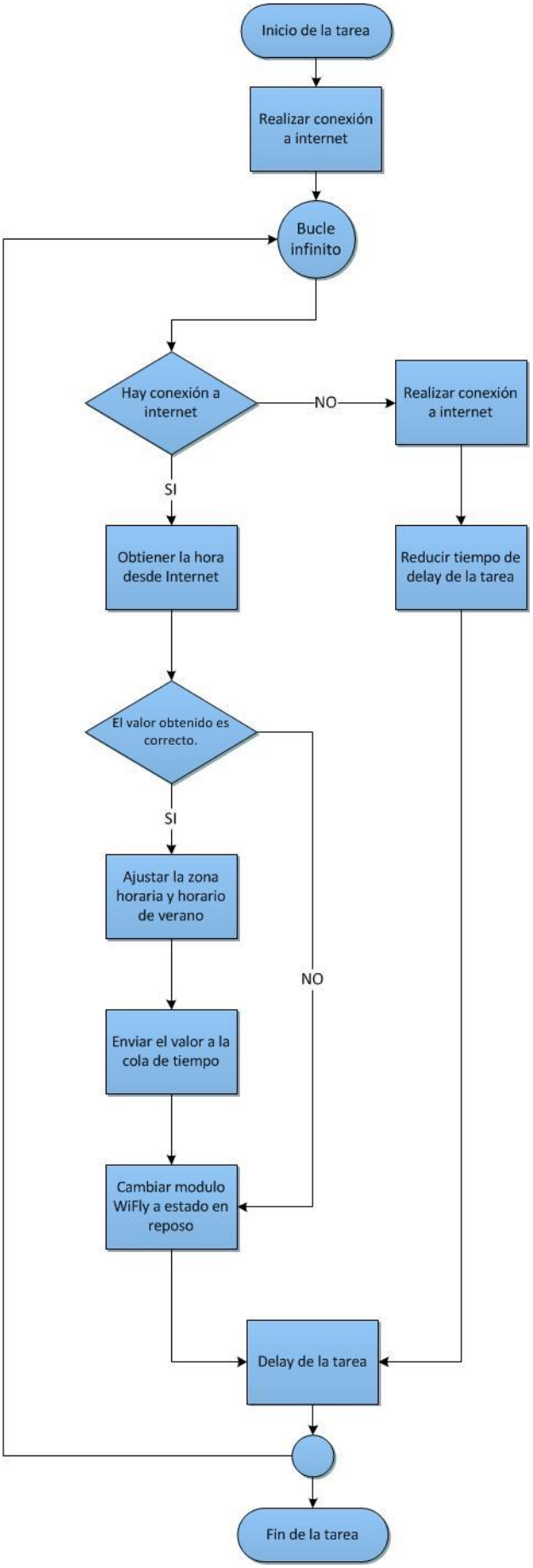


Figura 41: Diagrama de flujo Task_SyncTimeOnline

Figura 42: diagrama de flujo Task_UpdateTimeRTC

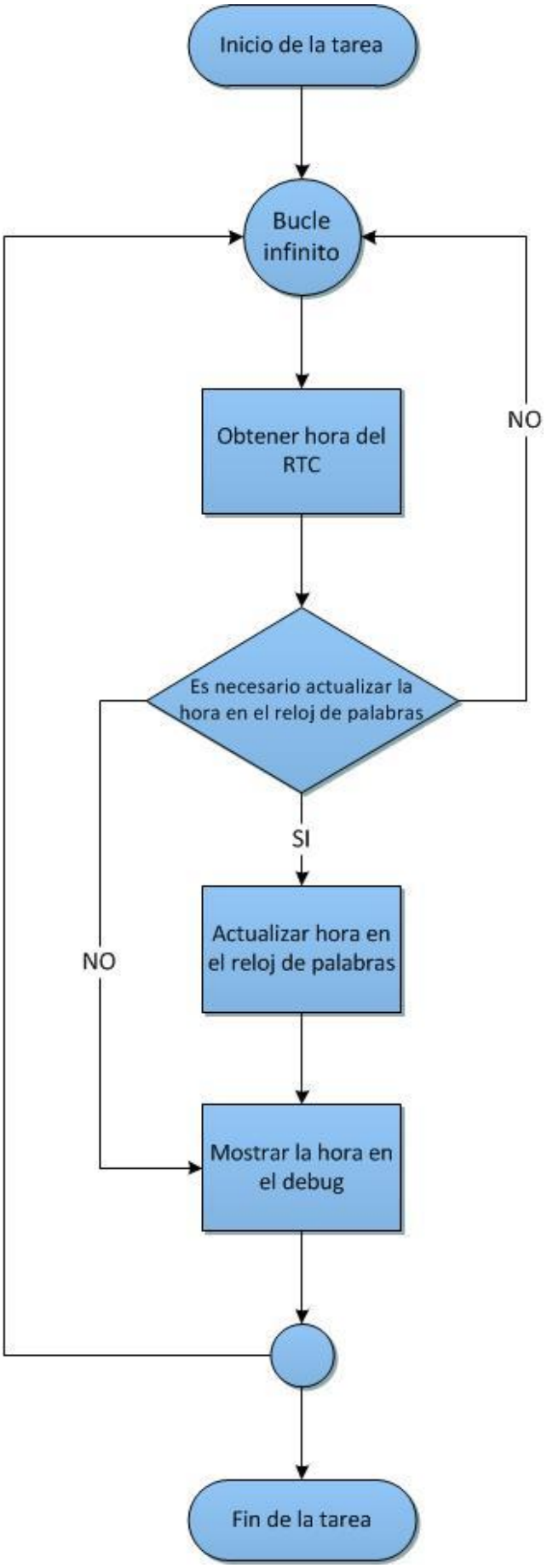


Figura 43: Diagrama de flujo Task_ShowTimeOnDisplay

4.6 Diagramas de la aplicación embebida.

Al igual que con la descripción de los componentes, la mejor forma de resumir todas las librerías que forman parte de la aplicación embebida y como se relacionan, es mediante un diagrama de bloques, el cual podemos ver en la Figura 44.

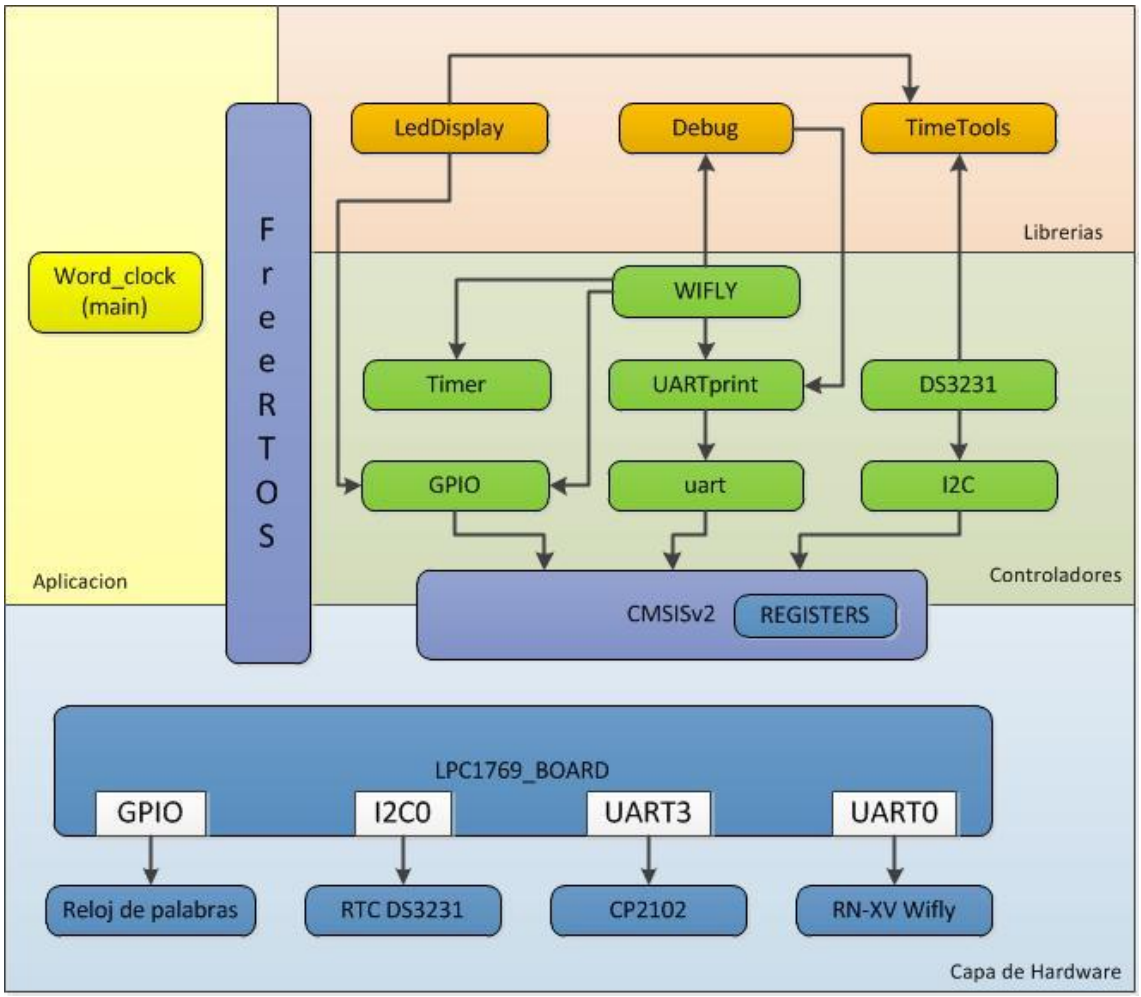


Figura 44: Diagrama de bloques de la aplicación embebida.

5. Viabilidad técnica.

Conceptualmente, está claro que el sistema es viable ya que se ha podido llegar a construir un sistema embebido que plasma los requisitos del proyecto, puede que el sistema lo haga de mejor o peor forma, pero materializa el concepto del proyecto.

Funcionalmente, el dispositivo obtenido es capaz de obtener la hora a través de Internet con el mínimo desfase posible y mostrar la hora a través del reloj de palabras, todo esto mediante el uso de los componentes mencionados en el capítulo cuatro. El diseño modular del sistema hace que todos los elementos que lo forman puedan ser sustituidos o reparados sin causar

demasiados problemas, además, también permitiría suplirlos por otros componentes, si por ejemplo dejasen de fabricarse o se encontrase otro que pudiera mejorar o abaratar el sistema, sin tener que realizar mucho trabajo. Quizá el diseño del reloj de palabras hecho para el prototipo pudiera generar algún problema por lo precario de la estructura de los leds, pero al tratarse de eso mismo, un prototipo, es una buena base de diseño para la creación de un modelo de producción.

Técnicamente, se han necesitado recursos de hardware que, o han sido facilitados por la universidad, o han sido obtenidos de una forma sencilla, y recursos de software, que no solo son gratuitos (exceptuando la licencia de Windows), sino que son tecnologías muy extendidas y globalizadas (lenguaje de programación C, sistema operativo FreeRTOS) que nos permiten tener una gran información sobre ellos.

En resumen, se puede decir que el proyecto es totalmente viable, pero siempre teniendo en cuenta que se trata de un prototipo y que necesitaría de muchas mejoras para llegar a ser un dispositivo de producción.

6. Valoración económica

En este apartado se efectúa un análisis de los costes que han supuesto la creación y el desarrollo del sistema embebido creado en este proyecto.

Para realzar este análisis, no se han tenido en cuenta el material informático (Ordenador, software,..) que se ha utilizado, así como los materiales necesarios para realizar trabajos de electrónica, ya que en este caso ya se disponía de ellos, y en caso contrario, se suponen básicos para el desarrollo de cualquier proyecto de estas características.

Primero se realiza un presupuesto del material físico que se ha necesitado para la creación del prototipo (Figura 45).

Concepto	Unidades	Precio unidad	Total
Placa de desarrollo LPCXpresso LPC1769	1	22,33 €	22,33 €
Módulo inalámbrico Wifly RN-XV	1	38,66 €	38,66 €
adaptador DIP Xbee	1	3,03 €	3,03 €
RTC DS3231	1	2,50 €	2,50 €
Convertidor USB/UART CP2102	1	5,99 €	5,99 €
Fuente de alimentación de 5V/2A	1	4,99 €	4,99 €

Diodos led blancos 5mm 3.2V - 20mA (pack indivisible de 200 unidades)	1	11,04 €	11,04 €
Resistencias	60	0,08 €	4,80 €
Drivers de potencia ULN2803	5	1,38 €	6,90 €
Panel de metacrilato 40x40 cm	1	4,90 €	4,90 €
Vinilo adhesivo sopa de letras	1	9,50 €	9,50 €
Caja de madera 40x40x2.5 cm	1	30,00 €	30,00 €
		Total:	**144,64 €**

Figura 45: Presupuesto del material para el prototipo.

Anadir que durante el desarrollo del proyecto el módulo Wifly original se estropeo y fue necesario comprar otro igual, por lo que si tuviésemos en cuenta este coste, el total del presupuesto aumentaría a 183.3 €

Una vez expuesto el coste del material, se presenta un presupuesto del coste de desarrollo del proyecto, donde se cuantifican las horas de trabajo dedicadas al mismo (Figura 46). Solo se ha tenido en cuenta el tiempo empleado en el desarrollo de la aplicación embebida y de la construcción del reloj de palabras, no se ha incluido las horas dedicadas a la documentación y aprendizaje sobre las tecnologías utilizadas, ya que se considera que a la hora de realizar un presupuesto de desarrollo, se da por hecho que se cuenta con los conocimientos necesarios.

Concepto	Horas	Precio hora	Total
Desarrollo de la aplicación embebida	170	15 €	2550 €
Construcción del reloj de palabras	50	15 €	750 €
		Total:	**3300 €**

Figura 46: Presupuesto del desarrollo del sistema.

El precio de la hora se ha decidido teniendo en cuenta el que podría ser un sueldo para un Ingeniero Técnico de Sistemas recién titulado.

Sumando todos los costes de desarrollo y construcción del sistema embebido el coste total del proyecto sería de 3444.64 Euros. Este coste parece muy elevado, pero hay que considerar que crear el primer prototipo siempre va a ser más costoso. A partir del momento que se cuente con un prototipo que pueda pasar a producción, los costes bajaran. Además al tratarse de un producto de consumo, una vez industrializado, los costes de desarrollo pasarían a ser mininos y solo necesarios para mejorar el sistema o producir otras versiones. También hay que tener en cuenta que la industrialización del producto, afectaría a los costes de los componentes haciendo que fuesen más asequibles.

7. Conclusiones

Cuando empecé con el proyecto, nunca me podría haber imaginado el trabajo y dedicación que serían necesarios para llevarlo a cabo, pero tampoco podría haber imaginado la cantidad de conocimientos nuevos que iba a adquirir.

Durante el desarrollo del proyecto se fueron produciendo problemas que se convirtieron en grandes desesperaciones, pero justo esos problemas hicieron que tuviese que esforzarme más para conseguir superarlos. Primero fueron problemas con el IDE LPCXpresso, luego no entendía el funcionamiento de la placa LPC1769, después el módulo Wifly no funcionaba, más adelante no recibía bien los paquetes UDP del servidor NTP, etc. Todos estos problemas que me llevaron de cabeza ahora son conocimientos que he asumido y afianzado gracias al trabajo.

Ahora que ya he terminado y echando la vista atrás, he de decir que ha sido duro llegar hasta aquí pero ha valido la pena. Me siento mucho más preparado a nivel profesional para afrontar cualquier reto, ya que no solo he aprendido sobre sistemas embebidos, también he mejorado mis conocimientos sobre metodologías de trabajo, planificaciones, programación y documentación.

7.1 Autoevaluación.

Una vez finalizado el proyecto, como se ha expuesto y comprobado en esta memoria, se puede afirmar que los objetivos principales marcados para este han sido cumplidos, exceptuando, la protección ante caída de tensión, que quizá, en la planificación inicial no debió marcarse como un objetivo principal y definirse como un objetivo secundario, pues este, no ha sido imprescindible para la obtención de un prototipo funcional del proyecto. La decisión de convertir la protección ante caídas de tensión en un objetivo secundario, se tomó al poco tiempo de iniciar el proceso de desarrollo, y se considera que fue una elección correcta viendo la evolución del proyecto.

Valorando los objetivos secundarios, incluida la protección ante caídas de tensión, hay que decir que no se han podido completar. A posteriori, y con el conocimiento de causa sobre el trabajo que ha llevado realizar el proyecto, quizás estos objetivos secundarios estaban fuera del alcance del proyecto, mas por falta de tiempo, que por falta de conocimientos o ambición. Pero hay que aceptar que no se han cumplido.

A continuación, y en forma de resumen se exponen tantos los objetivos cumplidos como los no cumplidos:

- **Objetivos cumplidos.**

    - Sincronización horaria en el LPC mediante el protocolo NTP.
    - Dotar al sistema de hora cuando el dispositivo no tenga conexión a internet usando el RTC
    - Protección a caídas de tensión.
    - Notificación visual de la hora a través de un reloj de palabras.

- **Objetivos no cumplidos.**

    - Poder configurar el dispositivo a través de una página web o aplicación de móvil.
    - Mejorar el sistema de gestión de la iluminación del panel mediante el uso de un circuito integrado o driver led.
    - Protección antes caída de tensión.

7.2 Propuestas de mejora

Antes de enumerar las propuesta de mejora que le se podrían aplicar al proyecto, hay que incluir dentro de ellas la consecución de los objetivos no cumplidos, pues formaban parte de la planificación inicial y se consideran necesarios para obtener un producto más completo y perfeccionado. A parte de estos, durante el proceso de desarrollo se han ido detectado posibles mejoras para el dispositivo.

Las mejoras que se podrían aplicar al sistema, se puede dividir entre mejoras de software que solo afectan al código de la aplicación embebida, y mejoras de hardware que afectan a los componentes del dispositivo embebido.

- **Mejoras de software**:

    - Mejorar el sistema de sincronización de la hora:

        - Utilizar un temporizador del LPC1769 para complementar la precisión del DS3231 y conseguir que la precisión del sistema sea de milisegundos.

        - Mejorar el uso de paquete NTP, el paquete que prepara esta primera versión, solo contiene los datos necesarios para que el servidor lo de por bueno y conteste con su hora, pero se puede mejorar para enviarle al servidor la hora del dispositivo y así mejorar la precisión del valor devuelto por el servidor NTP.

- Mejoras sobre la gestión del módulo Wifly.

    - El módulo Wifly integra una funcionalidad muy interesante que le permite configurarse en modo punto de acceso y activar un mini servidor web al que podemos acceder desde cualquier navegador para realizar las configuraciones básicas del módulo. Mediante la gestión de esta funcionalidad se podría conseguir de una forma muy sencilla, el poder configura a que red Wi-Fi queremos que se conecte y sus parámetros.

- **Mejoras de hardware.**

    - Mejoras sobre el DS3231.

        - Como alternativa a la mejora software que da más precisión al sistema, se podría sustituir el módulo DS3231 por otro módulo RTC que directamente contase con precisión de milisegundos o incluso, se podría prescindir de este módulo y utilizar el RTC que tiene integrado la placa LPC1769.

    - Mejoras sobre el reloj de palabras.

        - Además de poder incluir un driver de leds para gestionar mejor los leds del reloj de palabras, si se cambiase la forma en que esta interconectados los leds a una forma en que cada led representase una posición de una matriz de dos dimensiones, podríamos conseguir utilizar el panel no solo para iluminar las palabras que forman la hora y ya están representadas en la sopa de letras, sino también para ,mediante técnicas de dimming, representar texto y caracteres sobre el panel como se puede hacer con las matrices de leds que se encuentran en el mercado.

8. Glosario

ARM (del inglés Advanced RISC Machine)
Arquitectura de procesadores RISC de 32 y 64 bits utilizada en la gran mayoría de sistemas embebidos.

CMSIS-DAP
Estandarización para el control del puerto de acceso a la depuración (DAP, del inglés Debug Access Port) de un microcontrolador ARM Cortex a través de USB.

Cortex-M3
Procesador de tipo ARM de 32 bits.

Dimming
Sistema de regulación y control de iluminación que permite un gran ahorro energético.

DMA (del inglés, Direct Memory Access)
Característica permite a los componentes que lo integran acceder a la memoria del sistema de forma independiente.

DST (del inglés, Daylight Saving Time)
Horario que sigue la convención por la cual se adelantan los relojes para usar más la luz diurna, generalmente conocido como horario de verano.

FreeRTOS
Sistema operativo en tiempo real diseñado especialmente para dispositivos embebidos.

GMT (del inglés, Greenwich Mean Time)
Zona horaria utilizada oficialmente en varios países de Europa y África.

GPIO (del inglés, General Purpose Input/Output)
Pin genérico en un chip, cuyo comportamiento (entrada o salida) se puede controlar en tiempo de ejecución.

GPS (del infles, Global Positioning System)
Sistema de navegación basado en 24 satélites en órbita sobre el planeta tierra que envía información sobre la posición de una persona u objeto.

I2C o **I^2C** (del inglés, Inter-Integrated Circuit)
Bus de datos serial que permite la conexión de varios chips en el mismo bus.

IDE (del inglés Integrated Development Environment)
Aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

NTP (del inglés Network Time Protocol)
Protocolo de Internet para sincronizar los relojes de los sistemas informáticos a través del enrutamiento de paquetes en redes con latencia variable.

ppm (partes por millón)
Unidad de medida con la que se mide la concentración, concepto pareció al de porcentaje.

RTC (del inglés Real-Time Clock)
Reloj en tiempo real embebido en un circuito integrado capaz de mantener la hora.

SoC (del inglés, System-on-Chip)
Circuito integrado que integra todos o gran parte de los módulos que componen un computador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.

SSID (del inglés Sevice Set IDentifier)
Nombre que identifica a una red inalámbrica.

UART (del inglés Universal Asynchronous Receiver-Transmitter)
Dispositivo que controla los puertos y dispositivos serie de un microcontrolador.

UDP (del inglés User Datagram Protocol)
Protocolo de la capa de transporte basado en el intercambio de datagramas.

UTC (del inglés, cordinated Universal Time)
Principal estándar de tiempo utilizado mundialmente para regular los relojes y el tiempo.

Wiring
Plataforma de prototipado electrónico compuesto de un lenguaje de programación, un entorno de desarrollo integrado (IDE), y un microcontrolador.

9. Bibliografía

1. Definición de Internet de las Cosas (IoT) | ThingsCity [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.thingscity.com/definicion-de-internet-de-las-cosas-iot/

2. Historia del reloj: Origen y datos que hay que conocer | Guioteca.com [Internet]. Guioteca.com | Educación para Niños. 2015 [citado 25 de mayo de 2016]. Recuperado a partir de: http://www.guioteca.com/educacion-para-ninos/historia-del-reloj-origen-y-datos-que-hay-que-conocer/

3. Tecnicas de diseño [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.desarrolloweb.com/articulos/2183.php

4. Raspberry Pi - Teach, Learn, and Make with Raspberry Pi [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: https://www.raspberrypi.org/

5. Synchronizing time with DCF77 and MSF60 [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.compuphase.com/mp3/h0420_timecode.htm

6. GPS.gov: Cronometría [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.gps.gov/applications/timing/spanish.php

7. QLOCKTWO BY BIEGERT & FUNK [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.qlocktwo.com/info.php?lang=en

8. LPCXpresso board for LPC1769 with CMSIS DAP probe|NXP [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus/lpc-cortex-m3/lpc1700-cortex-m3/lpcxpresso-board-for-lpc1769-with-cmsis-dap-probe:OM13085

9. RN-XV WiFly Module - Wire Antenna - WRL-10822 - SparkFun Electronics [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: https://www.sparkfun.com/products/10822

10. DS3231.pdf [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

11. CP2102/9 Data Sheet - CP2102-9.pdf [Internet]. [citado 11 de junio de 2016]. Recuperado a partir de: https://www.silabs.com/Support%20Documents/TechnicalDocs/CP2102-9.pdf

12. Barry R. Using the FreeRTOS Real Time Kernel. Cortex-M3 Edition [Internet]. Recuperado a partir de: http://shop.freertos.org/FreeRTOS_Tutorial_Book_Generic_Cortex_M3_Edition_p/pdf_cortex-m3_tutorial_book.htm