UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Máster Universitario en Ingeniería Informática

TFM del Máster Universitario en Ingeniería Informática

Comunicación TCP/IP con sistemas empotrados

Presentado por RPC
en Universidad de Burgos — 10 de febrero de 2019
Tutor: AMG

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Máster Universitario en Ingeniería Informática

D. AMG, profesor del Departamento de Ingeniería Electromecánica, Área de Ingeniería de Sistemas y Automática.

Expone:

Que el alumno D. RPC, con DNI 12345678Z, ha realizado el Trabajo final de máster del Máster Universitario en Ingeniería Informática titulado *Comunicación TCP/IP con sistemas empotrados*.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 10 de febrero de 2019

Vº. Bº. del Tutor:

D. AMG

Resumen

Las placas de desarrollo facilitan el estudio y desarrollo de sistemas empotrados. Un sistema empotrado conectado a una red de comunicaciones de datos obtiene una nueva vía de interacción con otros sistemas, ya sean empotrados o convencionales. Un sistema empotrado conectado permite interactuar de forma remota con él, pudiendo realizar entre otras operaciones la consulta de sus sensores o la activación de sus actuadores.

En este proyecto se muestra como conectar una placa de desarrollo FRDM-K64F a una red de área local usando el conjunto de protocolos TCP/IP. Aprovechando tanto el *hardware* del que dispone la placa como del *hardware* conectado a ella, se ejemplifica la interacción con algunos de sus dispositivos a través de una aplicación web. Como dispositivos configurados para su uso remoto se encuentran el led integrado en la placa, una pantalla LCD y una placa de expansión.

Descriptores

Sistemas embebidos, placa desarrollo, familia de protocolos de internet, aplicación web.

Abstract

Development boards facilitate the study and development of embedded systems. An embedded systems connected to a data network obtains a new way of interaction with other systems, whether embedded or conventional. A connected embedded system allows to interact remotely with it, being able to carry out, among other operations, the query of its sensors or the activation of its actuators.

This project shows how to connect a FRDM-K64F development board to a local area network using the TCP / IP protocol suite. Taking advantage of both the hardware available on the board and the hardware connected to it, the interaction with some of its devices is exemplified through a web application. As devices configured for remote use are the LED integrated in the board, an LCD screen and an expansion board.

Keywords

Embedded systems, development board, Internet protocol suite, web application.

Índice general

Índice de figuras

Índice de tablas

Introducción

Los sistemas empotrados o embebidos (SE) son sistemas diseñados para realiza una función específica y, por tanto, solo realizan una o unas pocas tareas concretas. Este hecho les diferencia de otros sistemas de propósito general como ordenadores o teléfonos móviles, que son capaces de realizar multitud de tares de diferente naturaleza.

Como el propio nombre de los SE indica, este tipo de sistemas se suele encontrar integrado en otros sistemas eléctricos o mecánicos de mayor envergadura y que se encargan de controlar. Por esta razón, se requiere que los SE cuenten con ciertas características: tamaño reducido, bajo consumo o bajo coste. Requisitos que a su vez provocan nuevos atributos, uno de ellos es menor potencia de cálculo en comparación con otro tipo de sistemas. Además, en función de las condiciones ambientales donde se encuentre el SE es posible que necesite ser dotado de protección adicional ante situaciones especiales de temperatura, humedad, vibración, etc.

Ciertos SE controlan el funcionamiento de otros sistemas que requieren que sus operaciones se realicen de manera determinista. Es decir, una instrucción dada se tiene que realizar de manera inmediata o con un retardo mínimo y conocido de antemano. Para dar soporte a operaciones en tiempo real, los SE cuentan con Sistemas Operativos en Tiempo Real (RTOS) que se encargar de repartir el tiempo de ejecución de cada tarea y asignándolo en función de la prioridad de cada tarea.

A nivel de *hardware*, un microcontrolador es quien se encarga de ejecutar las instrucciones programadas. Cada microcontrolador cuenta con un microprocesador, con memoria y con determinados periféricos de entrada y salida. Para que el SE se pueda comunicar con sus periféricos o con otros dispositivos se emplean los buses serie de datos. Algunos de los buses más

utilizados son Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), o Universal Serial Bus (USB).

Otro medio que emplean los SE para comunicarse son las redes de comunicaciones de datos a las que se puede conectar tanto por cable como de manera inalámbrica. Existen varios protocolos para transmitir datos en red, un conjunto muy común y conocido es la familia de protocolos de internet, también conocido como pila o conjunto de protocolos TCP/IP (Transmission Control Protocol / Internet Protocol). Esta familia de protocolos es ampliamente utilizada para comunicar equipos en redes de área local (LAN), en redes de área local inalámbricas (WLAN) y en todo el mundo a través de Internet.

Con el uso de TCP/IP se bre un abanico de nuevas funcionalidades permitiendo realizar funciones, apoyadas en otros protocolos de la pila TCP/IP, que antes no eran posible. Por ejemplo, usando el Hypertext Transfer Protocol (HTTP) se pueden transferir páginas web, con File Transfer Protocol (FTP) transferir archivos, con Simple Mail Transfer Protocol (SMTP) enviar correos electrónicos o con Message Queuing Telemetry Transport (MQTT) se pueden enviar mensajes bajo el patrón de mensajería de publicar-suscribir.

En este trabajo se muestra como crear un SE conectado usando TCP/IP en una placa de desarrollo FRDM-K64F del fabricante NXP. Se parte de la configuración de los componentes *hardware* y *software* necesarios, para terminar demostrando la interacción, de manera remota, con el SE desde una aplicación web. Desde dicha aplicación es posible enviar comandos para realizar alguna de las funciones programadas en la placa.

1.1. Estructura de la memoria

La presente memoria se estructura de la siguiente manera:

- **Introducción:** descripción abreviada de los temas principales abordados en el proyecto. La introducción está acompañada de la estructura que toma la memoria y un listado con el contenido adjunto a la misma.
- **Objetivos del proyecto:** declaración de los objetivos generales, técnicos y personales que se pretenden conseguir con el desarrollo de este trabajo.
- **Conceptos teóricos:** explicación de los conceptos teóricos más relevantes en la realización del proyecto. Se tratan tanto los sistemas empotrados como la familia de protocolos TCP/IP.

- **Técnicas y herramientas:** descripción de las técnicas y las herramientas que han sido empleadas para el desarrollo del proyecto. También se muestran las alternativas valoradas y los motivos de escoger la opción seleccionada.
- **Aspectos relevantes del desarrollo:** presentación de los aspectos o facetas que han tomado mayor relevancia durante la ejecución de trabajo.
- **Trabajos relacionados:** estado de la técnica en la creación de sistemas que emplean sistemas empotrados conectados en red.
- **Conclusiones y líneas de trabajo futuras:** conclusiones extraídas tras la realización del proyecto, así como nuevas líneas de trabajo sobre las que mejorar o ampliar lo presentando en este proyecto.

1.2. Anexos a la memoria

La memoria se presenta acompañada de los siguientes anexos:

- **Plan del proyecto software:** exposición de la planificación temporal y del estudio de viabilidad del proyecto, tanto la económica como la legal.
- **Especificación de requisitos del software:** presentación del catálogo de requisitos, así como la descripción de cada uno de ellos.
- **Especificación de diseño:** descripción de la fase de diseño, el diseño de datos, el diseño procedimental y el diseño arquitectónico, del *software* del SE y de la aplicación web.
- **Manual del programador:** explicación en detalle de aquellos aspectos que un programador debe conocer para trabajar con el código fuente del proyecto.
- **Manual de usuario:** explicación para que un usuario interesado en el proyecto sea capaz de instalar, configurar y operar con el SE y con la aplicación web.

1.3. Contenido adjunto

Se adjunta el siguiente contenido a la memoria y los anexos:

- *Software* para la placa de desarrollo.
- Aplicación web para interacción remota.
- Documentación del *software*.
- Documentación de la aplicación web.
- Repositorio en línea con el código del *software*.
- Repositorio en línea con el código de la aplicación web.

Objetivos del proyecto

En este capítulo se detallan los objetivos que se pretenden conseguir con la ejecución del proyecto. Se diferencian tres tipos de objetivos, los generales que dan causa al proyecto, los técnicos inherentes al tipo de proyecto realizado y los personales que se desean conseguir *motu proprio*.

2.1. Objetivos generales

- Configurar un sistema empotrado que sea capaz de conectarse en red.
- Dotar al sistema empotrado de diversas funciones usando algunos de los periféricos de los que dispone.
- Demostrar la ejecución correcta de las funciones implementadas.
- Crear una interfaz web que permita interactuar con el sistema empotrado.

2.2. Objetivos técnicos

- Configurar el *hardware* de una placa de desarrollo FRDM-K64F para poder conectarla a través de su puerto Ethernet a una LAN.
- Utilizar componentes *hardware* de la placa como son sus LED de colores para mostrar que es capaz de comunicarse via TCP/IP.
- Utilizar el bus serie de datos I^2C presente en la placa para extender su funcionalidad y que pueda mostrar mensajes en una pantalla.

- Emplear modulación por ancho de pulsos para regular la intensidad del brillo de unos LED presentes en una placa de expansión.
- A nivel de comunicaciones, usar la implementación ligera de TCP/IP "lwIP" para manejar las transmisiones.
- Crear una aplicación web usando la tecnología JSF capaz de comunicarse con una placa conectada en red.

2.3. Objetivos personales

- Ampliar los conocimientos y la experiencia en el desarrollo de *software* para sistemas empotrados.
- Conocer el conjunto de herramientas de trabajo que proporciona el fabricante de la placa de desarrollo.
- Extender el entendimiento en la familia de protocolos TCP/IP.
- Revisar el proceso de desarrollo de una aplicación web.
- Emplear el sistema de control de versiones distribuido Git a través de la plataforma de desarrollo GitHub.
- Emplear la técnica de desarrollo ágil Scrum en las diferentes fases del desarrollo.
- Aprender a usar el sistema de composición de textos LaTeX y utilizarlo para realizar la documentación del proyecto.

Conceptos teóricos

En este capítulo se sintetizan los aspectos esenciales tratados en este proyecto para mejorar su compresión y entendimiento.

3.1. Sistemas empotrados

Parte importante del proyecto se centra en obtener un sistema empotrado (SE) capaz de comunicarse usando los protocolos TCP/IP. A continuación se describen los conceptos más relevantes en torno a los SE.

Descripción

Se puede considerar que un sistema empotrado es aquel cuyo *hardware* y *software* se encuentran estrechamente relacionados, está diseñado para cumplir con una función específica, se encuentra integrado en un sistema mayor, no se espera que el usuario lo modifique y puede trabajar sin interacción o con la mínima interacción humana necesaria [1].

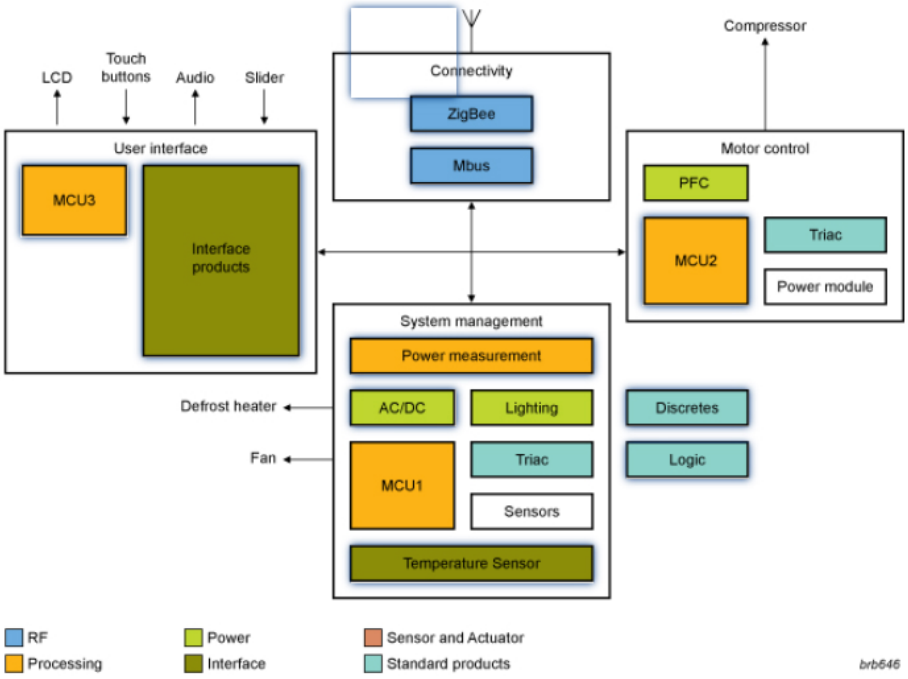

Figura 3.1: Ejemplo de integración de SE. [2]

En la figura 3.1 se muestra un ejemplo de aplicación de varios SE dentro de un sistema mayor, en este caso un frigorífico inteligente. El sistema cuenta con varios componentes: la interfaz de usuario, la gestión del sistema, el control del motor o el módulo de conectividad. Cada uno de los componentes se ayuda de un SE para realizar su función asignada.

Cada SE utiliza el *hardware* específico para realizar su función correspondiente.

- La interfaz de usuario cuenta con pantallas o botones.
- La gestión del sistema tiene acceso a sensores, el control de la corriente, la iluminación o los ventiladores.
- El control del motor presenta componentes eléctricos encargados de la regulación del compresor: un módulo de energía, un triodo para corriente alterna o un módulo de corrección del factor de potencia.

Del mismo modo, el *software* ejecutado por cada SE estará diseñado e implementado según la función a desempeñar realizando únicamente tareas específicas a su función.

Por otra parte, también se puede advertir que los SE no están pensados para que el usuario los modifique o programe, ni requieren que la interacción sea constante por parte del usuario para su correcto funcionamiento.

Los SE y los sistemas que los emplean se encuentran en todas partes. Se hallan en sistemas de movilidad y transporte, automatización industrial, sector sanitario, edificios inteligentes, redes de suministro inteligentes, investigación científica, seguridad pública, supervisión de salud estructural, recuperación de desastres, robótica, agricultura y ganadería, aplicaciones militares, telecomunicaciones y electrónica de consumo [3].

Características del *hardware*

Aunque el *hardware* de los SE se adapta a la función que desempeñan, todos ellos cuenta con un microcontrolador (MCU) encargado de controlar las operaciones del SE. Un MCU está compuesto por un procesador, memoria — RAM y ROM — y puertos de entrada y salida. El procesador se encarga de ejecutar las instrucciones de los programas que manejan las entradas y las salidas del sistema. Los programas ejecutados y los datos generados se almacenan en la memoria. Y los puertos de entrada y salida, envían y reciben la señales con las que trabaja el procesador[1].

También existen otros elementos que se encuentran a menudo en los SE:

- Puertos de comunicación serie o paralelo
- Dispositivos de interfaz humana
- Sensores
- Actuadores
- Conversores analógica-digital (ADC)
- Conversores digital-analógica (DAC)
- Componentes de diagnóstico y redundancia
- Componentes de apoyo al sistema
- Otros subsistemas:
    - Circuito integrado para aplicaciones específicas (ASIC)
    - Matriz de puertas programables (FPGA)

En la figura 3.2 se presenta un diagrama de bloques en el que se pueden observar los componentes que forman un SE de forma general y su dirección en el funcionamiento. Las entradas son procesadas por el microcontrolador que a su vez generará las salidas apropiadas.

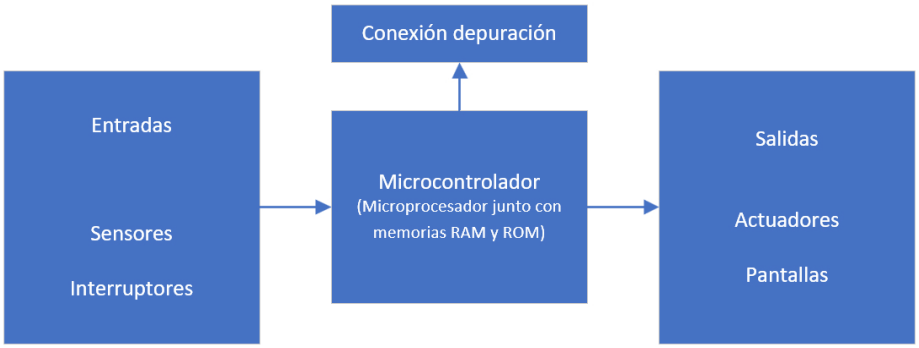

Figura 3.2: Diagrama de bloques del HW de un SE

Características del *software*

En cuanto al *software* presente en los sistemas embebidos, se pueden diferenciar varios grupos. El primero de ellos los *drivers* o controladores encargados de la interacción directa con el *hardware* del SE.

Luego se encuentra el *middleware*, que es aquel *software* que ocupa una posición entre el sistema operativo y los programas. El término se usa con frecuencia en librerías de rutinas de infraestructura que proporcionan servicios a los desarrolladores de los programas. [4]

Junto a lo anterior se puede hallar un Sistema operativo en tiempo real (RTOS). En un sistema operativo de propósito general varias tareas se ejecutan de forma aparentemente simultánea. De este modo se puede repartir el tiempo de ejecución de manera equitativa entre usuarios, por ejemplo. En cambio, en un RTOS se prima la ejecución de las tareas en un tiempo estrictamente limitado. Con un sistema de prioridades se determina la importancia de las tareas y cuales necesitan ser realizadas sin demora. [5]

Por último y funcionado sobre lo anterior se encuentra los programas necesarios para el funcionamiento del SE. En caso de contar con un RTOS será este el que se encargue de ejecutar una tarea u otra. Sino, de la forma conocida como *bare-metal* se ejecutan las tareas de forma secuencial. Secuencia que solo es alterada en caso surgir una interrupción.

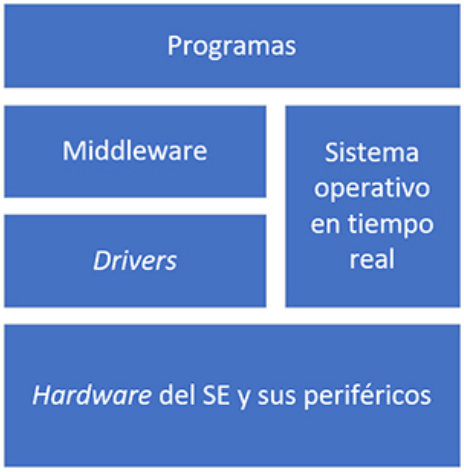

Figura 3.3: Digrama de bloques del SW de un SE

En el diagrama de bloques de la figura 3.3 se observan como se ubican unos componentes sobre otros. Los componentes se comunican con aquellos adyacentes. El RTOS es un componente opcional y que funciona de forma paralela a *drivers* y *middleware*.

3.2. Familia de protocolos TCP/IP

Otra parte fundamental del proyecto consiste en dotar de conectividad al SE usando TCP/IP. Para que dos dispositivos sean capaz de comunicarse necesitan intercambiar datos de una forma conocida y consensuada entre ambos. Con el objetivo de garantizar la interoperabilidad entre sistemas de diferentes modelos o fabricantes surgen protocolos estandarizados que garantizan dicha comunicación.

En *A dictionary of computing* [4] se define un protocolo como:

> "Un acuerdo que gobierna los procedimientos usados para intercambiar información entre entidades cooperantes. [...] La información es transferida a la ubicación remota usando el protocolo de más bajo nivel, después pasa de manera ascendente via interfaces hasta alcanzar el nivel correspondiente en el destino. En general, el protocolo regulará el formato de los mensajes, la generación de la información de comprobación, el control de flujo, además de las acciones a tomar ante el acontecimiento de errores. [...]"

Como se puede ver en la definición, los protocolos definen la manera en la que se transmite información en un mismo nivel. Dependiendo del modelo usado existen más o menos niveles intercambiando información. Por ejemplo, según el modelo de interconexión de sistemas abiertos [6], también conocido como Modelo OSI, existen 7 niveles o capas.

Descripción

En este proyecto se utiliza la familia de protocolos TCP/IP, la cual se compone de decenas de protocolos. Los protocolos se reparten en alguno de los siguientes niveles: enlace, internet, transporte, aplicación.

En comparación con el modelo OSI, la primera capa (física) queda delegada al *hardware* y las tres últimas (sesión, presentación y aplicación), se combinan en una sola.

Las capas se definieron en los *Request for Comments* (RFC) [7] y [8]. Siguiendo un orden lógico de menor a mayor nivel se describen de la siguiente manera [9]:

Enlace Esta capaz es la encargada de actuar como interfaz entre el resto de capas superiores y la red local subyacente. Algunas redes TCP/IP no usan protocolos en este nivel. Por ejemplo, las redes Ethernet se encargan de manejar las capas uno y dos. En otras redes sin implementación de este nivel se usan protocolos como Serial Line Internet Protocol (SLIP) [10] y Point-to-Point Protocol (PPP) [11].

Además, en esta capa se especifican otros protocolos comunes como el Address Resolution Protocol (ARP) [12] encargado del descubrimiento de las direcciones de enlace, o Neighbor Discovery Protocol (NDP) [13] que proporciona cierta funcionalidades más que ARP en redes que usan el Protocolo de Internet Version 6 (IPv6).

Internet Nivel correspondiente a la capa de red del modelo OSI. Se encarga del direccionamiento lógico de los dispositivos, del empaquetado, manipulación y envío de los datos, y del enrutamiento.

En esta capa de define el protocolo fundamental de todo el conjunto conocido como Internet Protocol (IP). Actualmente hay dos versiones en funcionamiento, la 4 y la 6. Para distinguir una de otra se usa la nomenclatura IPv4 [14] y IPv6 [15] respectivamente.

Además, en esta capa existen otros protocolos de soporte. Internet Control Message Protocol para IPv4 (ICMP) [16] y IPv6 (ICMPv6) [17]

son usados para el control y la notificación del errores. Internet Group Management Protocol (IGMP) [18] y Multicast Listener Discovery (MLD) [18] se usan para establecer grupos de difusión múltiple.

Transporte La principal labor realizada en esta capa es facilitar la comunicación de dispositivos aunque pertenezcan a redes de diferentes tipos. La comunicación se puede orientar de dos maneras, la primera con protocolos orientados a conexión y la segunda con protocolos no orientados a conexión.

Los protocolos fundamentales de esta capa son el Transmission Control Protocol (TCP) [19] y el User Datagram Protocol (UDP) [20]. TCP junto con IP son los protocolos más importantes de toda la familia como se puede deducir al ser usados en su nombre "TCP/IP".

TCP es un protocolo orientado a conexión, proporciona un conjunto completo de servicios para aquellas aplicaciones que lo necesiten y su uso se considera fiable. Su direccionamiento en la capa de transporte permite que varias aplicaciones puedan usar una misma dirección IP. El protocolo establece una conexión virtual entre dos dispositivos capaz de enviar información de manera bidireccional. Las transmisiones usan una ventana deslizante que permite detectar aquellas que no reconocidas para que sean retransmitidas.

UDP es un protocolo no orientado a conexión, más simple que TCP y sus transmisiones pueden resultar no fiables. Aunque también proporciona un medio de direccionamiento como TCP, UDP no agrega nada más, no se establece una conexión este dispositivos, no se realizan retransmisiones, por lo tanto, determinadas transmisiones pueden llegar a perderse. Su uso se justifica en aquellos escenarios donde la velocidad de la transmisión prima por encima de todo aunque se incurra ocasionalmente en la perdida de información.

Aplicación Esta es la cuarta y última capa del modelo TCP/IP, por ello es la capa de mayor nivel. Los protocolos de esta capa tienen objetivos de índole más diversa que los protocolos que se encuentran en capas de menor nivel.

Por ejemplo, con Dynamic Host Configuration Protocol (DHCP) [21] un servidor DHCP puede asignar direcciones IP y otros parámetros de configuración a los dispositivos que lo soliciten. Domain Name System (DNS) [22] [23] resuelve o traduce entre direcciones IP y nombres inteligibles por los usuarios.

Hay protocolos que sirven para la transferencia en la web como Hypertext Transfer Protocol (HTTP) [24], su segunda versión (HTTP/2) [25], o de forma segura Hypertext Transfer Protocol Secure (HTTPS) [26]. Protocolos para la recepción de correo electrónico como Internet Message Access Protocol (IMAP) [27] o Post Office Protocol version 3 (POP3) [28], también para el envío con Simple Mail Transfer Protocol (SMTP) [29].

Otros protocolos comunes son Network Time Protocol (NTP) [30] para la sincronización de relojes en línea. Session Initiation Protocol (SIP) [31] es usado para establecer sesiones en tiempo real típicas de aplicaciones de voz, vídeo o mensajería. Secure Shell (SSH) [32] establece una conexión segura sobre redes inseguras, muy utilizado para iniciar y ejecutar sesiones remotas.

Para terminar, dos protocolos usados habitualmente en sistemas empotrados o en dispositivos de Internet de las cosas (IoT) son Message Queuing Telemetry Transport (MQTT) [33] y Constrained Application Protocol (CoAP) [34]. Mientras que el primero se apoya en TCP y en un bróker para el intercambio de mensajes, el segundo emplea UDP y está caracterizado por su baja sobrecarga ideal para ser utilizado en dispositivos limitados.

Técnicas y herramientas

Este capítulo presenta las técnicas metodológicas y las herramientas de desarrollo usadas durante la realización del proyecto. En algunas de las técnicas y herramientas existen varias alternativas a utilizar. En este capítulo se comentan las características principales de cada alternativa y se expone el porqué de la elección tomada.

4.1. Técnicas metodológicas

Scrum

Scrum es definido por dos de sus mayores promotores Schwaber y Sutherland [35] como:

> "Un marco de trabajo en el que las personas pueden abordar problemas flexibles y complejos, mientras se entregan productos creativa y productivamente del mayor valor posible."

Desde el punto de vista del desarrollo del *software*, trabajar en un producto en toda su extensión puede atraer problemas debidos, por ejemplo, a cambios en los requisitos de los clientes. Para solventar este problema, Scrum enuncia que el trabajo se realice de forma ligera, iterativa e incremental, permitiendo que el desarrollo pueda responder mejor ante imprevistos.

Scrum propone grupos de trabajo donde sus integrantes tienen diferentes roles y responsabilidades. El grupo sigue un flujo de trabajo que gira entorno al concepto de *sprint*, definido como la unidad básica de desarrollo. Antes de cada *sprint* se planifica su duración temporal, se definen la tareas a realizar

en él y cuando ya está en marcha se revisa diariamente para analizar su evolución. Una vez terminado se comprueban los resultados obtenidos y se proporcionan los entregables generados.

Por tanto, Scrum se ha escogido con el propósito de realizar el desarrollo de forma ágil.

4.2. Herramientas *hardware* de desarrollo

Placa de desarrollo FRDM-K64F

El uso de placas de desarrollo sirve como toma de contacto al microprocesador y al resto del *hardware* que componen un SE. En este proyecto la placa utilizada es el modelo FRDM-K64F.

Figura 4.4: Placa de desarrollo FRDM-K64F [36]

El microcontrolador (MCU) de esta placa es el Kinetis® K64. Este MCU cuenta con el microprocesador Cortex®-M4 capaz de funcionar a 120 Mhz. Destacan la capacidad de funcionar en modos de energía ultra

bajos, su elevado rendimiento y los dispositivos integrados de conectividad y comunicación como los puertos General Purpose Input/Output (GPIO), la interfaz Ethernet o los buses de datos serie I^2C o SPI.

La placa K64F ofrece el *hardware* necesario para poder aprovechar las características que ofrece el MCU. Por ejemplo, cuenta con los pines suficientes para dar cabida a todos los GPIO. Además, la disposición de los conectores es compatible con la disposición utilizada en placas Arduino permitiendo el uso de multitud de placas de extensión ya existentes.

También presenta un puerto Ethernet que permite conectar la placa a una red local y, a la postre, a Internet. También cuenta con *hardware* extra que posibilita nuevas opciones a la hora de investigar y desarrollar con la placa. Por ejemplo, puertos USB, para alimentación, depuración y conexión; y diodos emisor de luz (LED) de colores rojo, verde y azul (RGB).

Placa de expansión Arduino Basic I/O

Como la placa K64F es compatible con las placas de expansión de Arduino se utiliza la placa Arduino Basic I/O para poder utilizar *hardware* adicional.

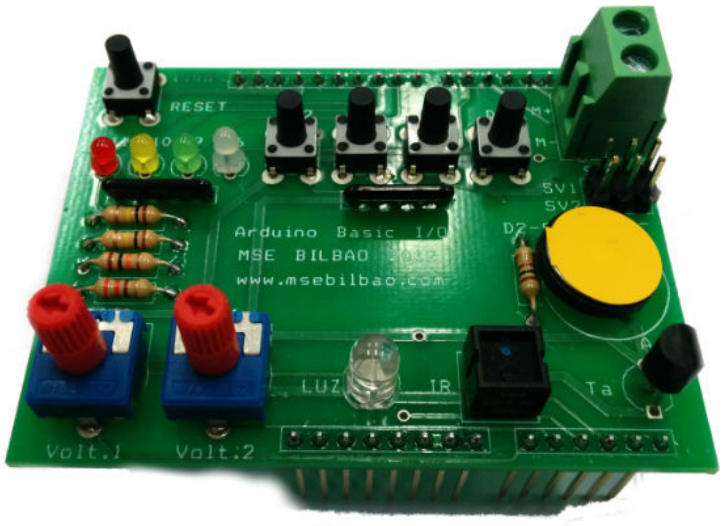

Figura 4.5: Placa de expansión Arduino Basic I/O [37]

En concreto, se utilizan sus 4 LED de colores en combinación con la técnica de modulación de señales pulse-width modulation (PWM). Con esta técnica se consigue regular la intensidad del brillo de cada uno de los LED.

Pantalla de cristal líquido (LCD)

El disponer de una pantalla permite mostrar información en forma de texto a un usuario del SE. En particular, la pantalla utilizada puede mostrar 2 líneas de texto de 16 caracteres cada una. Gracias a esta pantalla el usuario puede conocer ciertos datos acerca del estado de inicialización de la placa o puede usarla para mostrar sus propios mensajes.

Cabe destacar que la pantalla usa una conexión de tipo paralela, necesitando hasta 16 pines para la transmisión de datos, la alimentación, la iluminación y el control de la transmisión. Para simplificar su uso la pantalla incorpora un módulo de comunicación I^2C reduciendo las conexiones necesarias a solo de 4.

Accesorios extra

Para realizar el montaje de las placas y la pantalla se utilizan varios accesorios extra.

Las placas de pruebas [38] son unos tableros con orificios que permiten conectar otras placas, componentes electrónicos, conectores y cables. Su orificios están conectados eléctricamente, siguiendo un patrón de bloques y líneas, permitiendo así la conexión entre los elementos del sistema a montar. La rapidez y facilidad del montaje y desmontaje de los elementos facilita el prototipado de sistemas antes de su fabricación.

Con el fin de conectar los pines de los componentes se usan varios cables puente [39]. Algunos están conectados directamente a las placas, otros están conectados a las placas de pruebas. Para evitar confusiones los cables son de colores, de tal manera que es posible distinguir la función que realiza cada uno.

Por último, como el objetivo del proyecto es realizar un SE que utilice los protocolos TCP/IP, es necesario conectar la placa K64F con un cable Ethernet a un determinado equipamiento de acceso a la red, por ejemplo, un *switch*.

4.3. Herramientas *software* de desarrollo

Entorno de desarrollo integrado (IDE)

Opciones valoradas:

Kinetis Design Studio IDE: entorno multiplataforma basado en *software* libre como Eclipse IDE o GNU Compiler Collection (GCC). Incorpora Processor Expert, una utilidad que permite añadir y configurar los componentes necesarios para un proyecto.

MCUXpresso IDE: al igual que KDS es un IDE multiplataforma basado en Eclipse IDE. Dispone de unas herramientas de configuración que permiten habilitar y configurar los pines, los relojes y los periféricos de la placa de desarrollo en uso.

Eclipse IDE for Enterprise Java Developers: IDE adaptado para Java EE. Facilita el despliegue de aplicaciones en servidores de aplicaciones como GlassFish entre otros.

En cuanto a los IDE para el desarrollo del *software* de la placa K64F, se contemplan los dos primeros IDE. Ambos pertenecen a NXP, pero el propio fabricante indica que Kinetis Design Studio IDE ha dejado de ser desarrollado y que no se recomienda su uso en nuevos proyectos. Para nuevos proyectos se insta a usar su otro entorno MCUXpresso IDE. Por lo tanto, la opción seleccionada como entorno de desarrollo es **MCUXpresso IDE**.

Las herramientas de configuración que incorpora MCUXpresso se agrupan bajo el nombre de Config Tools [40]. La herramienta Pins Tools, sirve para configurar los pines del MCU. Clocks Tool permite configurar gráficamente de los relojes del sistema. Y Peripherals Tool asiste en la configuración de los periféricos disponibles en la placa de desarrollo.

Como tercer y último componente de la *suite* MCUXpresso queda citar el MCUXpresso Software Development Kit (SDK) [41]. El kit incluye un repertorio amplio de *drivers* y *middleware* listos para funcionar en la placa de desarrollo.

Por otra parte, para el desarrollo de la aplicación web la opción utilizada es **Eclipse IDE**. La versión especifica para Java EE permite manejar los servidores de aplicaciones, pudiendo arrancarlos y detenerlos desde el propio IDE, asi como pudiendo desplegar o quitar en ellos la aplicaciones en desarrollo directamente.

Otras herramientas utilizadas por el SE

FreeRTOS: RTOS encargado de planificar la ejecución de las tareas del SE. En función la configuración establecidas y de las prioridades asignadas, una tarea prioritaria será capaz de detener la ejecución del resto. El RTOS también se puede encarga del envío de mensajes entre tareas, mediante mensajes directos, colas o buzones.

lwIP: Implementación liviana de la pila de protocolos TCP/IP. Permite a la placa utilizar la mayoría de los protocolos habituales: IPv4, IPv6, TCP, UDP, ICMP, IGMP...

Estas herramientas se han utilizado por su conocimiento previo más que por un exhaustivo examen de todas las alternativas existente.

Repositorio del código fuente

Opciones valoradas:

GitHub: La versión gratuita permite la publicación de repositorios ilimitados, tanto públicos como privados. Ambos tipos de repositorios se encuentran almacenados en su nube. También permite un número ilimitado de colaboradores.

Bitbucket: De forma gratuita admite repositorios tanto privados como públicos, permitiendo 1 GB de almacenamiento en los privados. El número de colaboradores está limitado a 5. Y el almacenamiento de los repositorios se realiza en la nube.

GitLab: En la versión gratuita ofrece repositorios privados y públicos, disponiendo los públicos de 10 GB de almacenamiento. Respecto al almacenamiento cuenta con dos opciones, repositorios almacenados en la nube o de forma local. En cualquiera de los casos, permite un número ilimitado de colaboradores.

Las tres opciones basan su funcionamiento en Git por lo que cualquiera sería válida. Teniendo en cuenta la posibilidad de usar ZenHub para la gestión de proyectos, el repositorio escogido es **GitHub**.

En efecto, hay dos repositorios para el código fuente del proyecto alojados en GitHub, uno para el código usado por la K64F y otro para la aplicación web.

Herramientas de documentación

Opciones valoradas:

Word: Procesador de textos por excelencia perteneciente a la *suite* ofimática Office de Microsoft. Sus documentos son formato cerrado pero el programa también permite la apertura y edición de otro tipo de documentos.

Writer: Procesador de texto de código abierto que está incluido en la *suite* ofimática LibreOffice, también de código abierto. Sus documentos son formato abierto, pero también es posible editar documentos en otros formatos como los de Word.

LaTeX: Sistema de preparación de documentos para la producción de documentación técnica y científica. Usado de manera casi estándar para la creación y publicación de documentos científicos.

GitHub Wikis: Espacio integrado dentro de los repositorios de GitHub para documentar el contenido del proyecto, permitiendo añadir información relevante para otros usuarios.

Visual Studio Code: Editor de código fuente desarrollado por Microsoft. Compatible con infinidad de lenguajes y su funcionalidad puede ser ampliada con el uso de extensiones.

Texmaker: Editor de texto para LaTeX. Incluye las funcionalidades habituales y las herramientas necesarias para la edición y compilación de documentos. Incorpora un visor de archivos pdf que permite comprobar rápidamente el estado del documento en edición.

A causa de lo declarado en los objetivos personales del proyecto 2.3, para la documentación de memoria y anexos se escoge **LaTeX** como herramienta para la documentación. La decisión se debe a su calidad tipográficas y por su práctica estandarización en publicaciones técnicas y científicas.

Para la edición del texto que hay que proporcionar a LaTeX se emplea **Visual Studio Code**, usado junto a la extensión LaTeX Workshop, que añade características y utilidades relativas a LaTeX. Es necesario tener instalada una distribución de LaTeX para poder generar los archivos, en este caso se usa MikTeX.

Además, la funciones de autocompletado y detección de errores disponibles para los lenguajes HTML y CSS hacen apto al programa para la edición de la página web mostrada por la aplicación web.

Herramientas de documentación del código fuente

Opciones valoradas:

Doxygen: Herramienta considerada como el estándar de facto para la documentación de códigos fuente en lenguaje C++ aunque también soporta el lenguaje C. Esta herramienta automatiza la extracción de los comentarios y es capaz de generar documentación en HTML válida para referencia en línea o documentación en LaTeXpara referencia fuera de línea.

Javadoc: Utilidad que genera documentación en formato HTML a partir de los comentarios incluidos en el código fuente. Viene includa con el SDK de Java y puede ser ejecutada individualmente, directamente desde el IDE o desde otras herramientas como Maven.

Dada la propia finalidad de cada herramienta, para la documentación del código de la placa K64F se usa **Doxygen** y para la documentación de la aplicación web se emplea **Javadoc**.

Herramientas de comunicación

Opciones valoradas:

Correo electrónico: Solución para correos electrónicos de Microsoft. Disponible en el cliente web, y aplicaciones de escritorio y dispositivos móviles.

Skype Empresarial: Solución de comunicación perteneciente a la suite ofimática Office de Microsoft. Permite la comunicación mediante videollamadas, llamadas de voz o mensajería instantánea.

A parte de usar correo electrónico, el empleo de **Skype Empresarial** permite la comunicación directa entre los interesados del proyecto.

Herramientas de gestión de proyectos

Opciones valoradas:

Trello: *Software* de administración de proyectos que además de contar con una aplicación web cuenta con aplicaciones para dispositivos móviles. Funciona de manera similar a un tablón digital que permite organizar en tableros los diferentes proyectos del usuario. En cada uno de estos tableros se pueden colocar tarjetas representando las tareas a realizar por los participantes del proyecto.

ZenHub: Herramienta para la gestión ágil de proyectos integrada dentro de GitHub. Cuenta con tableros donde ubicar las tareas desde su concepción hasta su finalización. Es compatible con la metodología ágil Scrum y su concepto de *sprint*.

La herramienta de gestión de proyectos empleada es **ZenHub** por su integración con GitHub.

Resumen de las herramientas y tecnologías utilizadas

La siguiente tabla muestra las herramientas y tecnologías más relevantes en cada parte del proyecto. Como se puede ver algunas solo están involucradas en una parte, como el lenguaje C, mientras que otras se encuentran presentes a lo largo de todo el desarrollo, como el uso de GitHub.

Herramientas	firmware	web app	memoria
C	X		
Doxygen	X		
FreeRTOS	X		
lwIP	X		
Java		X	
JSF		X	
Javadoc		X	
Maven		X	
GlassFish		X	
XHTML		X	
HTML5		X	
CSS3		X	
LaTeX			X
MikTeX			X
VS Code		X	X
GitHub	X	X	X
ZenHub	X	X	X

Tabla 4.1: Herramientas y tecnologías utilizadas

Aspectos relevantes del desarrollo del proyecto

Durante el desarrollo del proyecto han surgido diversos hechos relevantes. En este capítulo se comentan los eventos ocurridos y la decisiones tomadas al respecto.

5.1. Aprendizaje

Una faceta fundamental del proyecto ha sido el aprendizaje. Conocer nuevas herramientas, técnicas y metodologías, profundizar en las ya conocidas y revisar aquellas que hubieran podido caer en el olvido.

Revisando los objetivos personales 2.3 se puede ver como el deseo de aprendizaje está presente al dejar patente que no solo se quiere cumplir con los objetivos generales 2.1 y técnicos 2.2.

Querer utilizar nuevas herramientas, técnicas, metodologías, etc. no está exento de problemas o contratiempos que no dudaron en dejarse ver desde el inicio.

5.2. Inicio

Para desarrollar el *software* que va a utilizar la placa K64F es necesario usar un IDE. En la sección sobre los IDE valorados 4.3 se comentan brevemente las diferencias entre Kinetis Design Studio (KDS) y MCUXpresso.

La decisión de usar MCUXpresso estuvo basada fundamentalmente en el abandono por parte de NXP de KDS. Esto no quiere decir que no se

pudiera haber usado KDS. Pero siguiendo la propia recomendación del fabricante y queriendo usar herramientas modernas se tomó la decisión de usar MCUXpresso.

La principal diferencia entre IDE es la presencia de Processor Expert (PE) en KDS y la respectiva ausencia en MCUXpresso. Esta herramienta fue la usada en su momento para aprender a desarrollar SE. Al perder PE fue necesario dedicar tiempo a aprender como usar las nuevas herramientas presentes en la *suite* de MCUXpresso, el IDE, el SDK y las Config Tools.

La primera toma de contacto estuvo apoyada en los ejemplos ofrecidos por el SDK. Son pequeñas piezas de código que realizan funciones simples que permiten experimentar con la placa y añadir modificaciones para ver como responde.

Familiarizado con el IDE, tocaba aprender a configurar los relojes del sistema. Usar como base la configuración de los ejemplos sirvió para conocer que relojes se tenían que activar, a que frecuencia y que salidas se tenían que habilitar.

Por último queda configurar el MCU y sus pines. Para poder usar un componente hay que decir al MCU que pines se van a utilizar y la función deseada. Que un dispositivo funcione correctamente requiere de la sección de los pines correctos y de la configuración eléctrica correcta. Por ejemplo, para poder usar el puerto Ethernet no sirve la configuración por defecto sino que hay que configurar el estado *pull-up* de alguno de sus pines.

Dominado MCUXpresso se pudo comenzar el desarrollo propiamente dicho.

5.3. Desarrollo con la placa K64F

Técnicas y metodologías

En esta fase se tomó la decisión junto con el tutor, de integrar metodologías y técnicas que podían resulta beneficiosas para el proyecto.

Una medida tomada fue usar GitHub como repositorio del código y sistema de control de versiones. Además, como los IDE y VS Code están preparados para usar Git, su uso resultó bastante asequible y cómodo.

Otra medida tomada fue usar Scrum y para ello el trabajo se planificó en sucesivos *sprints*. Scrum propone equipos de desarrollo de varias personas, reuniones diarias, reuniones al terminar el *sprint*, el rol de Scrum Master,

el Product Owner, etc. Como el proyecto se ha realizado individualmente, usar Scrum ha servido como forma de conocer la metodología, usando principalmente el *sprint* como herramienta para dividir el trabajo y planificar el proyecto.

Conexión a la red

Con la placa siendo capaz de usar el puerto Ethernet se presentaba el asunto de la obtención del dirección IP. Establecer una dirección fija presenta problemas si ya está usada o si se conecta en una subred con diferente rango de direcciones.

Por este motivo se tomó la decisión de usar DHCP. Esta decisión obliga a contar con un servidor DHCP en la misma red a la que se conecta la placa. De no obtener dirección la placa se quedaría a la espera indefinidamente. Para que el usuario sepa la dirección de la placa, en un primer momento se mostraba en la consola de depuración para más tarde mostrar también en el LCD tanto la dirección IP como el puerto TCP a la escucha.

Con la pila TCP/IP en funcionamiento era posible enviar paquetes de datos a la placa. Como lwIP implementa varios protocolos, ICMP entre ellos, era posible realizar *pings* para comprobar el estado de la placa.

Funciones del *hardware*

Para ejemplificar el uso de un SE conectado en red se decidió aprovechar el *hardware* incluido en la placa. Representando a unos actuadores y siendo los componentes más visibles de la placa se decidió utilizar los LED de colores RGB. Para poder activar o desactivar los LED se determinó enviar un comando con una instrucción y argumento específicos para el LED a alterar. El comando se transmitiría dentro un paquete TCP.

El parámetro de cada comando sirve para indicar el color a alterar. Los comandos para encender los tres colores básicos de los LED RGB son:

“led:r”
“led:g”
“led:b”

Como los LED se pueden encender simultáneamente se puede combinar su luz para crear nuevos colores. En la tabla siguiente se pueden ver los todos colores posibles.

Color	LED Rojo	LED Verde	LED Azul
Rojo	X		
Verde		X	
Azul			X
Amarillo	X	X	
Magenta	X		X
Cyan		X	X
Blanco	X	X	X
Ninguno			

Tabla 5.2: Colores producibles usando los LED RGB

El LCD se incluyó con la intención de mejorar la comunicación del usuario y ampliar la funcionalidad del SE. Esta decisión provocó inconveniente que volvía a involucrar el uso de MCUXPresso. Con anterioridad para poder utilizar el LCD se usaba una librería de funciones adaptada para usar los componentes generados por PE.

En origen la librería ya había sido adaptada desde código para Arduino a código para PE. Así que surgió la necesidad de adaptar de nuevo la librería para usar el código generado por MCUXpresso. Como la librería usaba funciones de espera también se tuvieron que portar estas funciones.

El comando a transmitir indica que la operación se trata de un mensaje para la pantalla, la línea en la que mostrar el mensaje y la cadena de caracteres a presentar. Ejemplos de comandos para mostrar una cadena de caracteres en cada línea:

“msg:0:Lorem ipsum”
“msg:1:dolor sit amet”

La última función añadida fue modificar la intensidad del brillo de los LED incorporados en la placa de expansión. La regulación de la intensidad se consigue usando los pines compatibles con PWM del MCU. Los pines PWM no están comunicados con los LED RGB de la propia placa K64F así que se conectaron a los pines de los LED de placa de expansión.

Al transmitir un comando y tras indicar que es una instrucción de tipo PWM, los argumentos indican el LED a regular y la intensidad (de 0 % a 100 %) deseada. Por ejemplo, los siguientes comandos sirven para regular la intensidad de cada LED con incrementos del 25 %.

“pwm:w:0”
“pwm:g:25”
“pwm:y:50”
“pwm:r:100”

Montaje *hardware*

Una vez que cada componente funcionaba de la manera esperada se realizó el montaje en las placas de pruebas para poder hacer funcionar el SE al completo.

Tareas del RTOS

Como se ha visto en la sección sobre *software* de los SE 3.1, es habitual contar con un RTOS encargado de la planificación de las tareas ejecutadas. El RTOS a usar fue FreeRTOS 4.3 por su conocimiento y experiencia previa. Para cada función disponible en la placa se creó una tarea concreta. Es decir, hay una tarea para usar el LCD, otra para los LED de la placa de expansión y otra para los LED RGB de la propia placa. Las tareas quedan a la espera de recibir un comando de la tarea, de mayor prioridad, que está escuchando en la dirección y el puerto asignados.

Desarrollo de la aplicación web

Con la placa operativa llegaba el turno de crear una aplicación web que permitiese actuar remotamente con la placa. La tecnología para crear la aplicación web es JSF, la parte encargada de comunicarse con la placa escrita en Java y la página visible usada por el usuario en XHTML y CSS.

Al desconocer CSS se presentaba una oportunidad de aprender sobre ese lenguaje. De nuevo requería dedicar cierto tiempo al estudio y aprendizaje hasta lograr una página con un aspecto aceptable.

Al usuario se le muestran cuatro apartados. El primero le permite introducir la dirección IP y el puerto TCP mostrados por el LCD. En caso de existir varios SE se pueden cambiar estos parámetros para escoger el SE con el que interactuar.

El siguiente bloque muestra un cuadrado por cada uno de los colores que se pueden generar con los LED RGB. Si lo que se desea es apagar los LED, pulsando el cuadrado negro se apagan todos.

El tercer apartado permite introducir una cadena de texto que será enviada al LCD. Como la pantalla solo puede mostrar 16 caracteres los cuadros de texto están limitados a esta longitud. Se muestran dos cuadros, uno para cada línea del LCD.

El último bloque está compuesto por cuatro controles deslizantes. Hay uno por LED de la placa de expansión. Usando estos controles se puede ajustar de forma visual la intensidad, yendo de 0 % a 100 % deslizando el control de izquierda a derecha.

Como todo el contenido se muestra en una página era necesario modificar el comportamiento realizado tras pulsar los botones. Por defecto, al pulsar un botón la página se desplazaba hasta arriba. El inconveniente se solucionó inyectando código HTML en los botones para modificar su comportamiento.

5.4. Documentación

En cuanto a la documentación, más allá de aprender a utilizar LaTeX no se presentaron mayores inconvenientes. Aparecieron las dudas habituales que surgen cuando se está aprendiendo a usar nuevas herramientas o técnicas pero se fueron solventando a medida que fueron surgiendo.

El uso de la plantilla facilitada por el tutor allanó la creación y edición de los documentos de la memoria y anexos. Se realizaron pequeñas modificaciones a la plantilla con la intención de mejorar el resultado obtenido.

Uno de los cambios fue usar el paquete *biblatex-iso690* para generar la bibliografía de acuerdo a la norma UNE-ISO 690:2013. Otra adición fue el paquete *url* que se encarga de mejorar la forma en que se tratan las URL en el texto.

Trabajos relacionados

En el campo de los SE se han realizado sistemas complejos y de gran valor en numerosos ámbitos. En este sentido, NXP muestra varias aplicaciones de los SE en las que se pueden emplear el mismo MCU, Kinetis ® K64, existente en la placa de desarrollo.

Los sistemas expuestos muestran características similares a las presentes en la placa K64F y a las usadas en este proyecto. El uso de conectividad en red via TCP/IP hace posible la comunicación con el SE a distancia. Y la utilización de GPIO, I^2C o PWM permite operar con los sensores, actuadores u otros componentes del SE.

A continuación se muestran cuatro sistemas pertenecientes a diferentes sectores de aplicación. Los ejemplos pertenecen al sector industrial, sanitario y comercial. En concreto, los sistemas operados por SE son los siguientes:

- Vehículo no tripulado
- Sistema de telesalud
- Cama de hospital eléctrica
- Impresora de terminal punto de venta

La funcionalidad, desempeño y ámbito de uso de las aplicaciones resultan muy diversas. Así pues, queda patente la versatilidad y el amplio espectro de entornos en los que se pueden encontrar SE.

Vehículo no tripulado

Un vehículo no tripulado es aquel vehículo capaz de funcionar sin un humano a bordo. De forma general los vehículos no tripulados se clasifican en función del medio en el que operan. Pueden ser terrestres, aéreos, acuáticos, submarinos, espaciales... Independientemente del tipo que sean y de que estén operados por control remoto o sean autónomos, los vehículos se sirven de SE para controlar sus funciones.

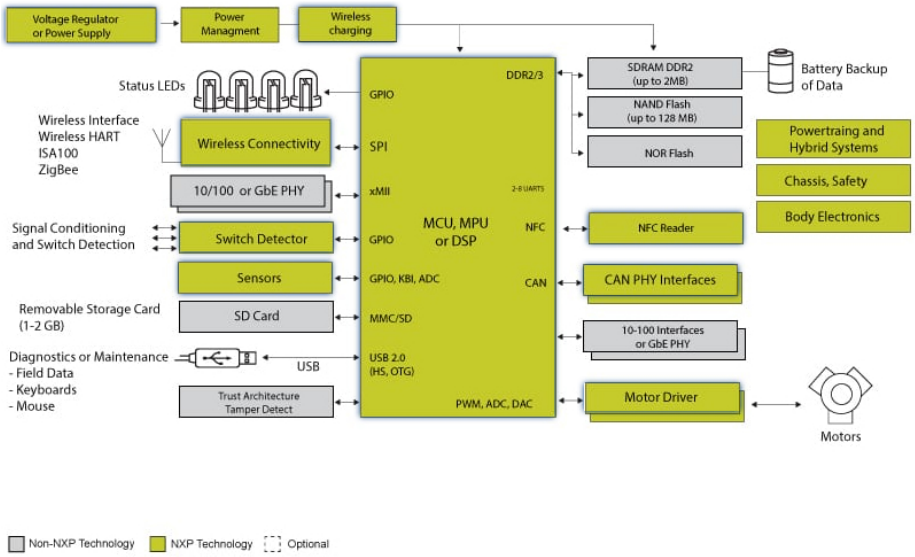

Figura 6.6: Diagrama de bloques de un vehículo no tripulado [42]

Con LED controlados por GPIO se puede conocer el estado del vehículo. Para dotar de conectividad al vehículo están disponibles interfaces cableadas e inalámbricas.

El uso de PWM es una característica compartida con el proyecto. Mientras que en el proyecto se ha usado para regular la intensidad del brillo de unas luces, en los vehículos no tripulados se emplea PWM para regular sus motores

Aunque no se ha incluido en el proyecto, la capacidad de usar tarjetas de memoria, de dispositivos conectados a un puerto USB o la comunicación con periféricos usando SPI está presente tanto en vehículos como en la placa de desarrollo.

Sistema de telesalud

El acceso a la asistencia sanitaria no siempre es factible o asequible. Vivir en entornos rurales alejados de centros médicos, la falta de medios de transporte, la pérdida de movilidad de los pacientes, o la falta de recursos económicos son causas que pueden impedir dicha asistencia. Con el avance de la tecnología y las telecomunicaciones ha surgido la posibilidad de abordar dicha problemática con el uso de sistemas de telesalud.

Con el uso de sensores de diverso tipo se recogen datos sobre el estado de salud del paciente. Su presión arterial, su ritmo cardíaco, su temperatura corporal, etc. Los datos son recogido por el sistema, procesados y retransmitidos al personal médico encargado de velar por la salud del paciente.

En la figura 6.7 se muestran los componentes que integran el sistema. Entorno al MCU se hallan presentes una interfaz infrarroja que recoge los datos de los sensores y una pantalla que muestra información al paciente. También se puede ver como el sistema usa PWM para generar sonidos.

Periódicamente el sistema envía los datos bien usando una conexión Ethernet o bien usando una conexión USB a un ordenador conectado a Internet. De manera opcional se puede conectar el sistema a Internet de forma inalámbrica. Este hecho serviría para aumentar la portabilidad del sistema de telesalud al completo.

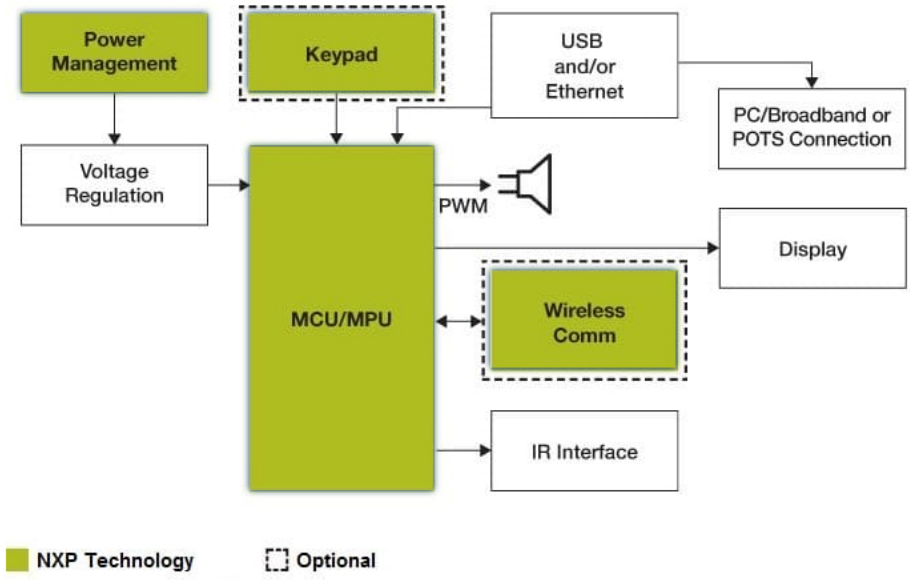

Figura 6.7: Diagrama de bloques de un sistema de telesalud [43]

Cama de hospital eléctrica

Siguiendo en el sector sanitario, se presenta otra muestra de aplicación de SE en las camas de hospital eléctricas. El uso de SE permite ofrecer el máximo confort posible al paciente y mejorar atención sanitaria recibida.

Con una serie de elementos de regulación se puede acomodar la cama a las necesidades del usuario. Para conseguir tal comodidad el SE cuenta con motores encargados de regular la inclinación o altura de varias partes de la cama.

Al integrar un monitor de constantes vitales o una bomba de infusión, dispostivos también creados con SE, se puede realizar una monitorización y atención completa del paciente. Y si además se conecta la cama a una red inalámbrica se abre la posibilidad de monitorizar y atender remotamente al paciente. Por ejemplo, desde un control de enfermería.

Para comunicarse telefónicamente con el enfermo se utiliza la conexión Ethernet en conjunción con la tecnología voz sobre protocolo de internet (VoIP).

Por último, que la cama disponga de una pantalla LCD permite que conocer *in situ* el estado de todo el sistema, paciente incluido.

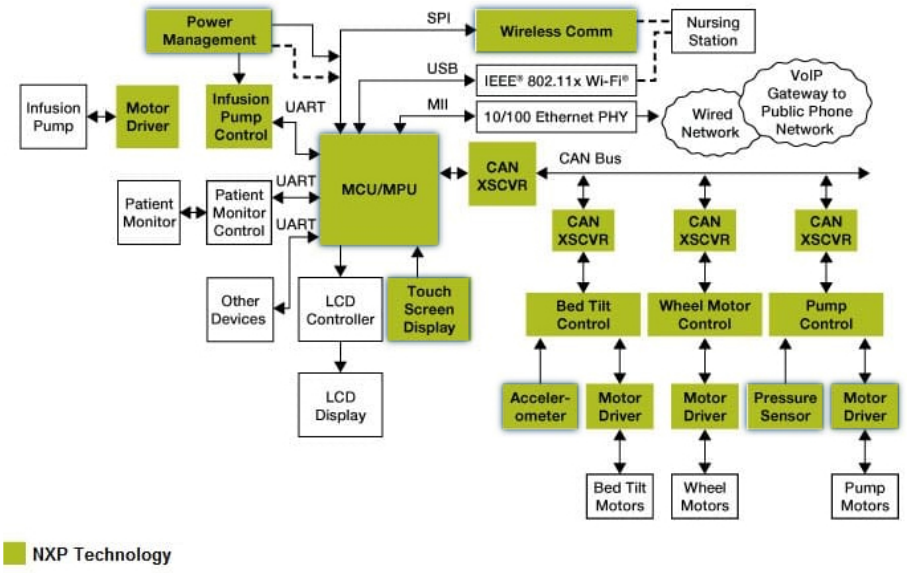

Figura 6.8: Diagrama de bloques de una cama de hospital eléctrica [44]

Impresora de terminal punto de venta (TPV)

Para terminar, se pasa al ámbito del sector minorista. Se muestra el sistema utilizado en una impresora de un TPV. La impresora está compuesta por dos SE. Mientras que uno de ellos se encarga de las entradas digitales, el otro se ocupa del control de los motores y dispositivos de impresión.

Las opciones de conectividad permiten comunicarse con la impresora de varias maneras. Con la conexión Ethernet se puede trabajar en red con la impresora. Igualmente, se pueden usar las conexión USB o RS-485.

Para controlar el *hardware* relativo al proceso de impresión se cuenta con la ayuda de un segundo SE. Como el SE auxiliar recibe señales analógicas requiere usar ADC, el resto de entradas y salidas se conectan directamente a los pines GPIO. Finalmente, la comunicación con el SE principal se realiza mediante I^2C o SPI.

Por otra parte, el SE principal cuenta con mayor número de conexiones al recibir las entradas del usuario, los controles digitales, gestionar la conectividad y manejar los sensores.

De nuevo están presentes características utilizadas a su vez en el proyecto: conectividad en red a través de un puerto Ethernet, conexión con periféricos usando el bus de datos serie I^2C y comunicaciones diversas via GPIO.

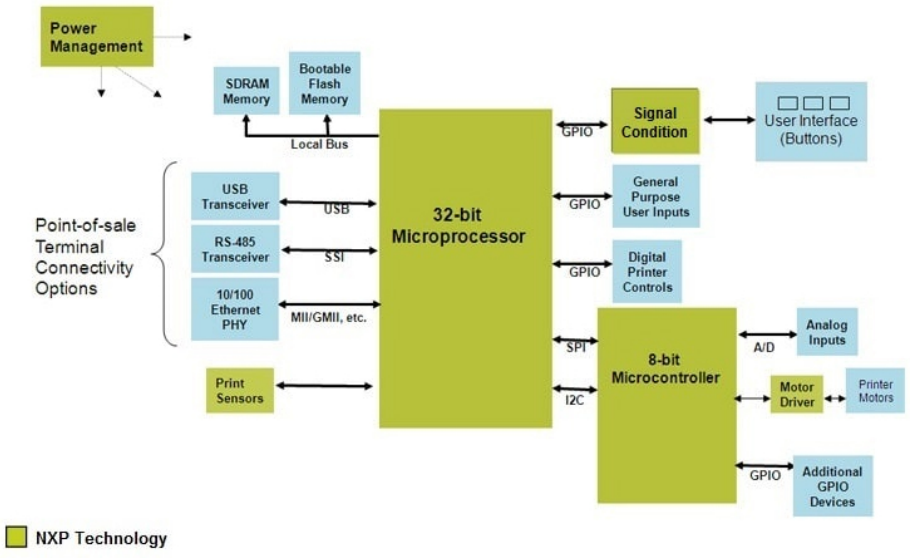

Figura 6.9: Diagrama de bloques de una impresora de TPV[45]

Conclusiones y Líneas de trabajo futuras

Conclusiones

Tras finalizar el proyecto han surgido varias conclusiones relativas al mismo.

- Al concebir el proyecto se establecían una serie de objetivos generales 2.1, técnicos 2.2 y personales 2.3 que se han completado satisfactoriamente. Se ha logrado crear un SE 3.1 conectado en red gracias al uso de la familia de protocolos TCP/IP 3.2. De manera más específica, se utilizado una placa de desarrollo K64F 4.2 junto con la implementación de la pila de protocolos TCP/IP lwIP 4.3 para conectar en red dicha placa. Además, para dar uso a la conectividad adquirida, la placa realiza varias funciones aprovechando sus interfaces, buses y dispositivos.

- Para su realización, no solo se han puesto en práctica los conocimientos aprendidos en el trascurso del Máster Universitario en Ingeniería Informática sino que también se ha aprovechado esta oportunidad para conocer otros. Como se indica en los aspectos relevantes del proyecto 5.1, uno de los principios esenciales del proyecto ha sido aprender nuevas herramientas, técnicas y metodologías; y revisar aquellas ya conocidas.

- Ha resultado un descubrimiento conocer la cantidad de contextos en los que puede aparecer un SE. Era conocida su presencia en entornos con fuerte presencia de la tecnología de la información, p. ej., electrónica

de consumo, automatización industrial, control de viviendas y edificios, interfaces hombre-máquina;pero su aparición en otros ámbitos como el sector de la salud donde su uso se extiende desde camas eléctricas 5.4 hasta bombas de insulina ha sido sorprendente.

- Desarrollar un SE se convierte en un desafío en comparación con el desarrollo para sistemas convencionales. Las restricciones del *hardware* se traducen en una necesidad constante de ahorrar espacio en memoria, limita la cantidad y el tamaño de librerías a usar, insta a usar implementaciones livianas como en el caso de lwIP y dado el caso que lo requiera, apremia el uso de soluciones que reduzcan el consumo de energía por parte del sistema.

- Aunque la pila de protocolos TCP/IP no sean los únicos tipos de protocolos usados por los SE su empleo ha facilitado el desarrollo con la placa y su posterior uso. Con inicializar la pila de protocolos era suficiente para poder enviar un *ping* y ver como la placa respondía satisfactoriamente.

- Los mecanismos de depuración incluidos en la placa de desarrollo y en el IDE 4.3 han demostrado su valía. En numerosas ocasiones mostrar en la consola de depuración un mensaje con el estado de la ejecución ha sido de gran utilidad para corregir errores. Sin ir más lejos, hasta la incorporación de la pantalla LCD, un mensaje de depuración era la única manera de saber la dirección IP asignada a la placa.

- Por lo que se refiere al desarrollo de la aplicación web, el aprendizaje y empleo de CSS ha servido para mostrar que se puede hacer en términos de presentación y diseño visual; y como se puede hacer. El desarrollo también ha servido para revisar las tecnologías usadas: Java EE, Enterprise Beans, Java Server Faces, GlassFish, Maven, etc.

- Con el uso de Scrum se consiguió repartir las tareas del desarrollo de forma práctica y efectiva. Siguiendo la planificación general, el desarrollo se efectuó en función del *sprint* del momento, concentrando los esfuerzos en la meta fijada en él.

- Por último, citar el empleo de Git y de GitHub 4.3. Su utilización ha evidenciado de nuevo su utilidad e idoneidad. En más de una ocasión ha sido valiosa la capacidad de poder recuperar el código fuente de una versión estable. Además, al estar disponible en línea ha permitido al tutor revisar y orientar sobre la evolución del desarrollo.

Líneas de trabajo futuras

El proyecto realizado puede ser mejorado o ampliado en diversos frentes. A continuación se describen aquellos aspectos de mayor relevancia.

- A nivel de *hardware* la placa de desarrollo tiene integrados más dispositivos de los utilizados en el proyecto con los que se podría tratar. Los botones físicos, el acelerómetro y magnetómetro, los canales de comunicación UART, SPI, USB, ADC; o el lector de tarjetas de memoria podrían proporcionar nuevas funcionalidades.
- En línea con lo anterior, la placa cuenta con la posibilidad de añadir módulos de conexión inalámbrica Wi-Fi y Bluetooth, de esta manera, resultaría interesante crear un SE conectado en red y móvil.
- El concepto de internet de las cosas (IoT) cada vez está más presente y la cantidad de dispositivos inteligentes conectados en red está aumentando rápidamente. Una mejora del proyecto vendría de la mano de la integración en la pila TCP/IP de alguno de los protocolos usados en IoT. Por ejemplo, MQTT o CoAP.
- En ese sentido, usando protocolos IoT se podría conectar el SE a la nube. Servicios como Azure IoT Hub permiten conectar, supervisar y administrar multitud de dispositivos inteligentes.
- Con esa orientación hacia IoT, también existen RTOS dedicado al asunto. Zephyr es compatible con la placa K64F y además integra opciones de seguridad y protección no siempre presentes en SE actuales.
- También es cierto que hay medidas de seguridad que se pueden integrar directamente. Aprovechando que lwIP implementa el protocolo criptográfico Transport Layer Security (TLS) las comunicaciones podrían ser cifradas para proporcionar seguridad en la transmisión de datos.
- Respecto a la aplicación web, aunque se ha tratado de que contase con un diseño web adaptable siempre se podría mejorar y reestructurar de acuerdo alguna de las metodologías usadas en el diseño de la experiencia del usuario.
- Además, se podrían incrementar las formas de interacción con la placa. Usando la interfaz de línea de comandos, un programa de ordenador con interfaz gráfica o mediante una aplicación para dispositivos móviles.

Bibliografía

1. JIMÉNEZ, Manuel; PALOMERA, Rogelio y COUVERTIER, Isidoro. *Introduction to Embedded Systems: Using Microcontrollers and the MSP430*. New York: Springer, 2014. ISBN 978-1-4614-3143-5. Disponible desde DOI: 10.1007/978-1-4614-3143-5.
2. NXP SEMICONDUCTORS. *Interactive Block Diagram: Refrigerator* [online]. 2016 [visitado 2019-01-25]. Disponible desde: https://www.nxp.com/applications/solutions/internet-of-things/smart-things/smart-home/refrigerator:REFRIGERATOR.
3. MARWEDEL, Peter. *Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things*. 3.ª ed. Cham: Springer, 2018. ISBN 978-3-319-56045-8. Disponible desde DOI: 10.1007/978-3-319-56045-8.
4. BUTTERFIELD, Andrew; EKEMBE NGONDI, Gerard y KERR, Anne. *A Dictionary of Computer Science*. 7.ª ed. Oxford: Oxford University Press, 2016. Oxford Quick Reference. ISBN 978-0-199-68897-5. Disponible desde DOI: 10.1093/acref/9780199688975.001.0001.
5. AMAZON.COM, INC. *Why RTOS and What is RTOS?* [online] [visitado 2019-01-25]. Disponible desde: https://www.freertos.org/about-RTOS.html.
6. INTERNATIONAL TELECOMMUNICATION UNION TELECOMMUNICATION STANDARDIZATION SECTOR. *Information technology - Open Systems Interconnection - Basic Reference Model: The basic model* [X.200]. ITU-T Publications, 1994 [visitado 2019-01-26]. Disponible desde: https://www.itu.int/rec/T-REC-X.200-199407-I/en. ITU-T Recommendation.

7. INTERNET ENGINEERING TASK FORCE. *Requirements for Internet Hosts - Communication Layers* [RFC 1122]. RFC Editor, 1989 [visitado 2019-01-26]. Disponible desde DOI: 10.17487/RFC1122. Internet Request for Comments.

8. INTERNET ENGINEERING TASK FORCE. *Requirements for Internet Hosts - Application and Support* [RFC 1123]. RFC Editor, 1989 [visitado 2019-01-26]. Disponible desde DOI: 10.17487/RFC1123. Internet Request for Comments.

9. KOZIEROK, Charles M. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. San Francisco: No Starch Press, 2005. ISBN 978-1-593-27047-6.

10. ROMKEY, J. *Nonstandard for transmission of IP datagrams over serial lines: SLIP* [RFC 1055]. RFC Editor, 1988 [visitado 2019-01-26]. Disponible desde DOI: 10.17487/RFC1055. Internet Request for Comments.

11. INTERNET ENGINEERING TASK FORCE. *The Point-to-Point Protocol (PPP)* [RFC 1661]. RFC Editor, 1994 [visitado 2019-01-26]. Disponible desde DOI: 10.17487/RFC1661. Internet Request for Comments.

12. PLUMMER, David C. *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware* [RFC 826]. RFC Editor, 1982 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC0826. Internet Request for Comments.

13. T., Narten y col. *Neighbor Discovery for IP version 6 (IPv6)* [RFC 4861]. RFC Editor, 2007 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC4861. Internet Request for Comments.

14. POSTEL, J. *Internet Protocol* [RFC 791]. RFC Editor, 1981 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC0791. Internet Request for Comments.

15. S., Deering y R., Hinden. *Internet Protocol, Version 6 (IPv6) Specification* [RFC 2460]. RFC Editor, 1998 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC2460. Internet Request for Comments.

16. J., Postel. *Internet Control Message Protocol* [RFC 792]. RFC Editor, 1981 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC0792. Internet Request for Comments.

17. A., Conta y S., Deering. *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification* [RFC 4443]. RFC Editor, 2006 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC4443. Internet Request for Comments.

18. HOLBROOK, H.; CAIN, B. y HABERMAN, B. *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast* [RFC 4604]. RFC Editor, 2006 [visitado 2019-01-27]. Disponible desde DOI: 10.17487/RFC4604. Internet Request for Comments.

19. POSTEL, J. *Transmission Control Protocol* [RFC 793]. RFC Editor, 1981 [visitado 2019-01-28]. Disponible desde DOI: 10.17487/RFC0793. Internet Request for Comments.

20. POSTEL, J. *User Datagram Protocol* [RFC 768]. RFC Editor, 1980 [visitado 2019-01-28]. Disponible desde DOI: 10.17487/RFC0768. Internet Request for Comments.

21. DROMS, R. *Dynamic Host Configuration Protocol* [RFC 2131]. RFC Editor, 1997 [visitado 2019-01-28]. Disponible desde DOI: 10.17487/RFC2131. Internet Request for Comments.

22. MOCKAPETRIS, P. *Domain names - concepts and facilities* [RFC 1034]. RFC Editor, 1987 [visitado 2019-01-28]. Disponible desde DOI: 10.17487/RFC1034. Internet Request for Comments.

23. MOCKAPETRIS, P. *Domain names - implementation and specification* [RFC 1035]. RFC Editor, 1987 [visitado 2019-01-28]. Disponible desde DOI: 10.17487/RFC1035. Internet Request for Comments.

24. FIELDING, R. y RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing* [RFC 7230]. RFC Editor, 2014 [visitado 2019-01-29]. ISSN 2070-1721. Disponible desde DOI: 10.17487/RFC7230. Internet Request for Comments.

25. BELSHE, M. y PEON, R. *Hypertext Transfer Protocol Version 2 (HTTP/2)* [RFC 7540]. RFC Editor, 2015 [visitado 2019-01-29]. ISSN 2070-1721. Disponible desde DOI: 10.17487/RFC7540. Internet Request for Comments.

26. RESCORLA, E. *HTTP Over TLS* [RFC 2818]. RFC Editor, 2000 [visitado 2019-01-29]. Disponible desde DOI: 10.17487/RFC2818. Internet Request for Comments.

27. CRISPIN, M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1* [RFC 3501]. RFC Editor, 2003 [visitado 2019-01-29]. Disponible desde DOI: 10.17487/RFC3501. Internet Request for Comments.

28. MYERS, J. y ROSE, M. *Post Office Protocol - Version 3* [RFC 1939]. RFC Editor, 1996 [visitado 2019-01-30]. Disponible desde DOI: 10.17487/RFC1939. Internet Request for Comments.

29. KLENSIN, J. *Simple Mail Transfer Protocol* [RFC 5321]. RFC Editor, 2008 [visitado 2019-01-30]. Disponible desde DOI: 10.17487/RFC5321. Internet Request for Comments.

30. MILLS, D.; J., Burbank y W., Kasch. *Network Time Protocol Version 4: Protocol and Algorithms Specification* [RFC 5905]. RFC Editor, 2010 [visitado 2019-01-30]. ISSN 2070-1721. Disponible desde DOI: 10.17487/RFC5905. Internet Request for Comments.

31. J., Rosenberg y col. *SIP: Session Initiation Protocol* [RFC 3261]. RFC Editor, 2002 [visitado 2019-01-30]. Disponible desde DOI: 10.17487/RFC3261. Internet Request for Comments.

32. T., Ylonen. *The Secure Shell (SSH) Transport Layer Protocol* [RFC 4253]. RFC Editor, 2006 [visitado 2019-01-30]. Disponible desde DOI: 10.17487/RFC4253. Internet Request for Comments.

33. BANKS, Andrew y GUPTAU, Rahul. *MQTT Version 3.1.1 Plus Errata 01* [MQTT Version 3.1.1 Plus Errata 01]. OASIS, 2015 [visitado 2019-01-31]. Disponible desde: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html. Internet Request for Comments.

34. SHELBY, Z.; HARTKE, K. y BORMANN, C. *The Constrained Application Protocol (CoAP)* [RFC 7252]. RFC Editor, 2014 [visitado 2019-01-31]. ISSN 2070-1721. Disponible desde DOI: 10.17487/RFC7252. Internet Request for Comments.

35. SCHWABER, Ken y SUTHERLAND, Jeff. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game* [online]. 2017 [visitado 2019-02-02]. Disponible desde: https://www.scrumguides.org/.

36. NXP SEMICONDUCTORS. *FRDM-K64F: Freedom Development Platform for Kinetis® K64, K63, and K24 MCUs* [online]. 2016 [visitado 2019-02-02]. Disponible desde: https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F.

37. CÓDIGO 21. *Tarjeta Arduino Basic I/O V2* [online] [visitado 2019-02-02]. Disponible desde: http://codigo21.educacion.navarra.es/autoaprendizaje/tarjeta-arduino-basic-io-v2/.

38. WIKIPEDIA. *Placa de pruebas* [online]. 2019 [visitado 2019-02-02]. Disponible desde: https://es.wikipedia.org/wiki/Placa_de_pruebas.

39. WIKIPEDIA. *Cable puente* [online]. 2018 [visitado 2019-02-02]. Disponible desde: https://es.wikipedia.org/wiki/Cable_puente.

40. NXP SEMICONDUCTORS. *MCUXpresso Config Tools - Pins, Clocks, Peripherals* [online]. 2016 [visitado 2019-02-03]. Disponible desde: https://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools/mcuxpresso-config-tools-pins-clocks-peripherals:MCUXpresso-Config-Tools.

41. NXP SEMICONDUCTORS. *MCUXpresso Software Development Kit (SDK)* [online]. 2016 [visitado 2019-02-03]. Disponible desde: https://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools/mcuxpresso-software-development-kit-sdk:MCUXpresso-SDK.

42. NXP SEMICONDUCTORS. *Unmanned Vehicles (Ground, Air, Water)* [online]. 2016 [visitado 2019-02-04]. Disponible desde: https://www.nxp.com/applications/solutions/internet-of-things/smart-things/unmanned-aerial-vehicles-uavs/flight-controller-and-sensors/unmanned-vehicles-ground-air-water:UNMANNED-VEHICLES.

43. NXP SEMICONDUCTORS. *Telehealth System* [online]. 2016 [visitado 2019-02-04]. Disponible desde: https://www.nxp.com/applications/solutions/internet-of-things/smart-things/healthcare/telehealth-system:TELEHEALTH-SYSTEM.

44. NXP SEMICONDUCTORS. *Powered Patient Beds* [online]. 2016 [visitado 2019-02-04]. Disponible desde: https://www.nxp.com/applications/solutions/internet-of-things/smart-things/healthcare/powered-patient-beds:POWERED-PATIENT-BEDS.

45. NXP SEMICONDUCTORS. *POS Printer* [online]. 2016 [visitado 2019-02-04]. Disponible desde: https://www.nxp.com/applications/solutions/internet-of-things/secure-things/secure-transactions/pos-printer:POS-PRINTER.