UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

**TFM del Máster Universitario en Ingeniería Informática**

# Comunicación TCP/IP con sistemas empotrados

Presentado por RPC
en Universidad de Burgos — 3 de febrero de 2019
Tutor: AMG

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. tutor, 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 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, 3 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 son los 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 como son un tamaño reducido, un bajo consumo o un bajo coste. Requisitos que a su vez provocan nuevas características como la 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 de temperatura, humedad, vibración, etc. no habituales.

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*, es un microcontrolador quien se encarga de ejecutar las instrucciones programadas. Un microcontrolador cuenta con un microprocesador, con la memoria y con los periféricos de entrada y salida. Periféricos habituales y que permiten la interacción entre el SE y otros dispositivos son aquellos que usan buses serie de datos. Algunos de los buses

más frecuentes 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 comunicarse en red, siendo unos de los más comunes los pertenecientes a la familia 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 abre un abanico de nuevas funcionalidades permitiendo realizar funciones 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 o recibir correo electrónico 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 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) de un 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 tanto los sistemas embebidos como las comunicaciones realizadas mediante la familia de protocolos TCP/IP. 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 que se pretenden conseguir con el desarrollo de este trabajo.
- **Conceptos teóricos:** explicación de los conceptos teóricos más relevantes en la realización del proyecto.

- **Técnicas y herramientas:** descripción de las técnicas y las herramientas que han sido empleadas para el desarrollo del proyecto.
- **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 entorno a los sistemas empotrados y los protocolos TCP/IP.
- **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 *firmware* 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:

- *Firmware* para la placa de desarrollo.
- Aplicación web para interacción remota.
- Documentación del *firmware*.
- Documentación de la aplicación web.
- Repositorio en línea con el código del *firmware*.
- 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 alguno de los buses serie de datos con los que cuenta la placa para crear una funcionalidad que pueda ser ejecutada de manera remota.
- Emplear alguno de los puertos de entrada y salida (digital o analógica), utilizando una alguna de las técnicas de modulación.

- A nivel de *software*, usar la implementación ligera de TCP/IP lwIP para manejar las comunicaciones.
- 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 algunos de los aspectos 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 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 haya 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]

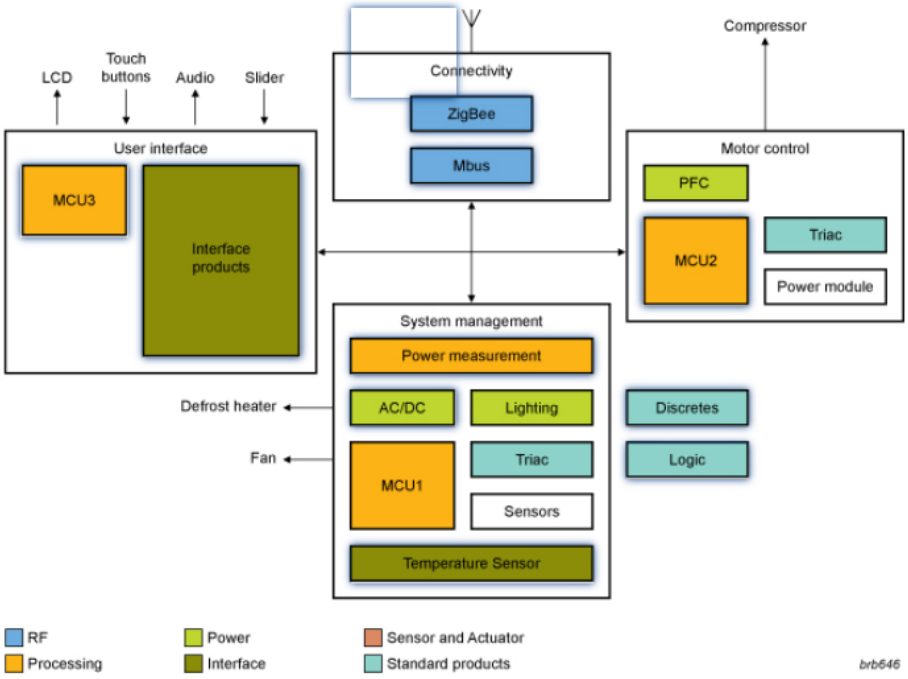


Figura 3.1: Ejemplo de integración de SE. [2]

En la figura 3.1 se muestra un ejemplo de uso 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 la conectividad. Cada uno de los componentes se ayuda de un SE para realizar su función asignada.

Los SE cuentan con el *hardware* específico para la tarea a realizar. La interfaz de usuario puede contar 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 para la regulación del compresor. Un módulo de energía, un triodo para corriente alterna o un módulo de corrección del factor de potencia son componentes que pueden estar presentes en el control del motor. Asimismo, el *software* ejecutado en cada uno de estos SE varía según la función a desempeñar realizando únicamente las tareas necesarias.

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 fácilmente. 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*

Los SE disponen de componentes *hardware* que siendo específicos para la tarea a la que están destinados se pueden generalizar en procesador, memoria 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 puede observar de forma general los componentes que forman un SE y su funcionamiento. Las entradas son procesadas por el microcontrolador que a su vez generará las salidas apropiadas.

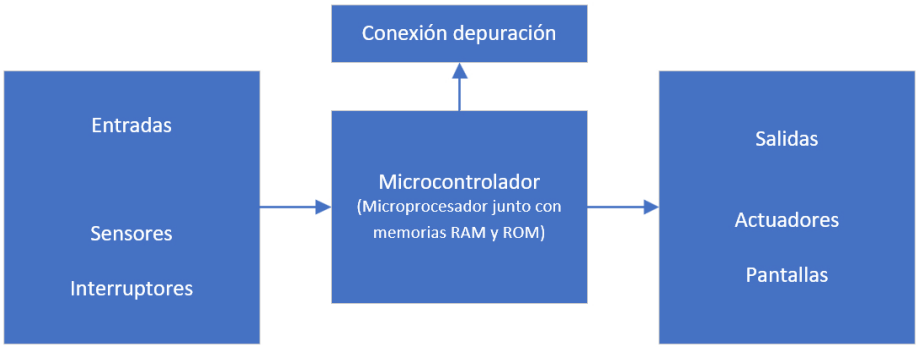


Figura 3.2: Diagrama de bloques del HW de un SE

## 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 forma conocida como *bare-metal* se ejecutan las tareas de forma secuencial. Secuencia que solo es alterada en caso surgir una interrupción.

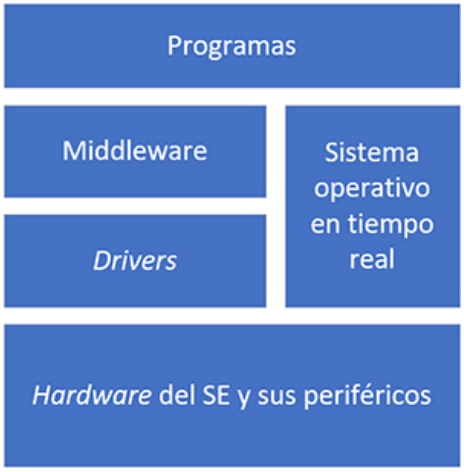


Figura 3.3: Digrama de bloques del SW de un SE

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. 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 envía 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 en [9] se describen de la siguiente manera:

**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 la misma y más funcionalidades que ARP en redes que usan el Protocolo de Internet Version 6 (IPv6).

**Internet** Capa 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 primordial de la familia de protocolos 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.

Asimismo existen otros protocolos de soporte como el Internet Control Message Protocol para IPv4 (ICMP) [16] y IPv6 (ICMPv6) [17] usado

para el control y la notificación del errores, o Internet Group Management Protocol (IGMP) [18] y Multicast Listener Discovery (MLD) [18] usados 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 resulta 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 no agrega nada más, no se establece una conexión este dispositivos, no se realizan retransmisiones, por lo tanto, los datos pueden 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 el Dynamic Host Configuration Protocol (DHCP) [21] un servidor DHCP puede asignar direcciones IP y otros parámetros de configuración a un dispositivo que lo solicite. El 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 el Hypertext Transfer Protocol (HTTP) [24] , su segunda versión (HTTP/2) [25], o el Hypertext Transfer Protocol Secure (HTTPS) [26]. Protocolos para la recepción de correo electrónico como el Internet Message Access Protocol (IMAP) [27] o el Post Office Protocol version 3 (POP3) [28], también para el envío con el Simple Mail Transfer Protocol (SMTP) [29].

Otros protocolos comunes son el Network Time Protocol (NTP) [30] para la sincronización de relojes en línea. El Session Initiation Protocol (SIP) [31] 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 usado 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 su uso 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 la realización del trabajo 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.

## 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. Destaca la capacidad de funcionar en modos de energía ultra bajos, su prominente 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. Cuenta con los pines suficientes para dar cabida a los GPIO. La disposición de los conectores es compatible con la 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.

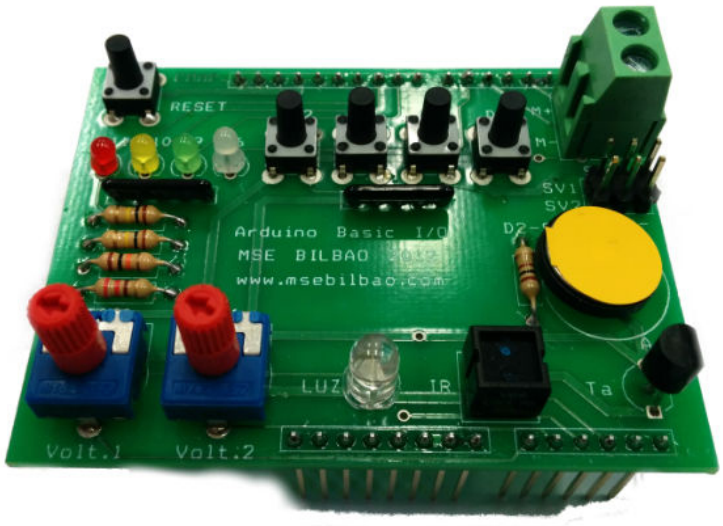


Figura 4.5: Placa de expansión Arduino Basic I/O [37]

En concreto, se utilizan sus 4 LED de colores que usados en combinación de la técnica de modulación de señales pulse-width modulation (PWM) se puede 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, esta pantalla 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. Como forma de simplificar su uso la pantalla incorpora un módulo de comunicación $I^2C$, de tal manera que solo es necesario el empleo de 4 pines.

### 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 conectados directamente a las placas, otros 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, ambos IDE pertenecen a NXP y 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 ello 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, asigna la función de los pines del MCU, establece sus propiedades eléctricas y genera automáticamente el código fuente que actúa sobre los pines. Otra herramienta es Clocks Tool la cual posibilita la configuración gráfica de los relojes del sistema y sus correspondientes frecuencias. La última herramienta es Peripherals Tool y asiste a la hora de habilitar y configurar 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. Como ejemplo de componentes del SDK utilizados en el proyecto se encuentran el RTOS FreeRTOS o la implementación liviana de TCP/IP lwIP.

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.

## Repositorio del código fuente

Opciones valoradas:

**GitHub:** Su versión gratuita permite la publicación de repositorios ilimitados, tanto públicos como privados. Ambos tipos de repositorios se encuentran almacenados en su nube. Por último, 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. 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:** 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 o 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

Opción valorada:

**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.

# Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo.

Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales.

Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW...

Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

# Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

# Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

# 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.