



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 Enrique del Olmo Domínguez
en Universidad de Burgos — 7 de julio de 2022
Tutor: Alejandro Merino Gómez y Daniel
Sarabia Ortiz



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Alejandro Merino Gómez y D. Daniel Sarabia Ortiz, profesores del departamento de Ingeniería Electromecánica (Área de Ingeniería de Sistemas y Automática).

Exponen:

Que el alumno D. Enrique del Olmo Domínguez, con DNI 71309191Z, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Comunicación con Sistemas Embebidos.

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

En Burgos, 7 de julio de 2022

Vº. Bº. del Tutor:

D. Alejandro Merino Gómez

Vº. Bº. del co-tutor:

D. Daniel Sarabia Ortiz

Resumen

Los sistemas embebidos (SE) son pequeños controladores capaces de realizar unas tareas determinadas programadas con anterioridad. Este tipo de sistemas son muy utilizados en la actualidad ya que, su hardware, bajo consumo, facilidad de uso y su escalabilidad los hacen propicios para solucionar un multitud de tareas. Estas tareas pueden ser desde la inclinación de una cama de hospital a la gestión de unas cintas transportadoras de elementos en una industria. En este proyecto podremos ver como se pueden programar sistemas embebidos para comunicarse entre sí mediante Internet. También se mostrará una simulación de para qué se podrían utilizar este tipo de sistemas en un entorno real. Para ello dispondremos de un laboratorio de pruebas con varias placas FRDM-K64F, dos servomotores, una pantalla LCD y un ‘shield’ de expansión.

Descriptores

Sistemas embebidos, Sistemas empotrados, conexión ethernet, comunicación entre sistemas, Internet de las cosas (IoT), NXP, UART, I2C, ADC, FRDM K64F.

Abstract

Embedded systems (ES) are small controllers capable of performing specific tasks programmed in advance. This type of systems are widely used nowadays because their hardware, low power consumption, ease of use and scalability make them suitable to solve a lot of tasks. These tasks can be from the inclination of a hospital bed to the management of conveyor belts of elements in an industry. In this project we will see how embedded systems can be programmed to communicate with each other over the Internet. We will also see a simulation of what these types of systems could be used for in a real environment. For this we will have a test laboratory with several FRDM-K64F boards, two servomotors, an LCD screen and an expansion shield.

Keywords

Embedded systems, ethernet connection, inter-system communication, Internet of Things (IoT), NXP, UART, I2C, ADC, FRDM K64F.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
Introducción	1
1.1. Descripción del trabajo realizado	2
1.2. Estructura de la memoria	4
1.3. Anexos	5
1.4. Contenido adjunto	5
Objetivos del proyecto	7
2.1. Objetivos generales	7
2.2. Objetivos técnicos	7
2.3. Objetivos personales	8
Conceptos teóricos	9
3.1. Sistemas embebidos	9
3.2. Tecnologías de comunicación para SE	13
3.3. Comunicaciones para los periféricos	17
3.4. Periféricos	20
Técnicas y herramientas	23
4.1. Técnicas: metodologías ágiles	23
4.2. Herramientas hardware	25
4.3. Herramientas software	34

4.4. FreeRTOS	34
4.5. Herramientas de documentación	36
4.6. Herramientas de comunicación	37
4.7. Herramientas de gestión de proyectos	37
Aspectos relevantes del desarrollo del proyecto	39
5.1. IDE: Kinetis vs MCUXpresso	39
5.2. FRDM K64F vs Arduino	40
5.3. RTOS	40
5.4. Metodologías Ágiles	41
5.5. Dificultades y Problemas	42
5.6. Caso de uso Real	43
Trabajos relacionados	45
6.1. SE en equipos médicos	45
6.2. SE en gestión de la energía	46
6.3. Conexión desde otros dispositivos	47
Conclusiones y Líneas de trabajo futuras	49
7.1. Conclusiones	49
7.2. Líneas de trabajo futuras	50
Bibliografía	53

Índice de figuras

1.1.	Esquema de conexiones del laboratorio.	3
1.2.	Planta piloto del proyecto.	4
3.3.	Diagrama de bloques placa FRDM K64F.	10
3.4.	Diagrama de bloques de OpenSDA.	11
3.5.	Capas del modelo TCP/IP.	14
3.6.	Trama UART.	18
4.7.	Diagrama resumen de la Metodología SCRUM. [22]	24
4.8.	Placa FRDM K64F.	25
4.9.	Placa de expansión Arduino basic I/O.	26
4.10.	Pantalla LCD con modulo IIC/I2C.	27
4.11.	Motor EMG30 utilizado en la planta piloto.	28
4.12.	Placa controladora de los motores.	29
4.13.	Sensor de temperatura, LM35.	30
4.14.	Pines del módulo wifi 8266.	31
5.15.	Caso de uso Kronospan.	43
5.16.	Placa ‘Maestro’ en el CPD.	44
6.17.	Esquema arquitectónico de un SE de medición de sangre.	45
6.18.	Sistema de Calefaccion con SE.	46

Índice de tablas

3.1. Diferencias USART y UART	18
4.2. Comandos Motores EMG30.	30
4.3. Comandos AT.	33

Introducción

Los sistemas embebidos o empotrados (SE) están muy presentes en nuestra vida cotidiana [11]. Algunos ejemplos de sistemas empotrados que podemos encontrar en nuestro día a día serían los electrodomésticos, relojes, coches, semáforos, entre otros. Todos estos aparatos, junto con robots o máquinas industriales, componen un campo importante para nuestra sociedad, ya que estos sistemas facilitan enormemente tareas pesadas o repetitivas en la vida de las personas.

Podemos referirnos a estos sistemas como un microcontrolador. Es importante conocer las diferencias entre un microcontrolador y un microcomputador [17]. Como diferencias a nivel técnico tenemos:

- En lo referido al software, encontramos que los microcontroladores tienen como objetivo la realización de pequeñas tareas programadas por un desarrollador para un fin concreto. Es por ello que cuentan con algunas características a nivel general de todos los microcontroladores como puede ser, bajo consumo, tamaño reducido y bajo coste. En cambio, un microcomputador se suele utilizar para entornos complejos y tareas extensas que a su vez requieren de otras tareas en cascada.
- A nivel de hardware, encontramos que un microcontrolador engloba tanto la unidad de procesamiento como una pequeña unidad de memoria (ROM, RAM, etc), además de algunos puertos para periféricos, un temporizador, etc. Podemos pensar en un microcontrolador como una minicomputadora.

En cuanto a los periféricos, se les pueden añadir sensores dotándoles de nuevas funcionalidades como por ejemplo, medición de humedad y calor,

envío y recibo de ultrasonidos y comunicaciones bluetooth o infrarroja y un largo etc.

Para que todos estos dispositivos puedan funcionar adecuadamente, se necesitan o al menos se prefiere tener a estos sistemas conectados entre sí. Mediante la transferencia de datos se pueden implementar aún más procesos, siempre con el objetivo de ser más eficiente a la hora de solucionar un problema.

Ahora que ya hemos visto su importancia y uso en la actualidad, veamos cómo funcionan los sistemas empotrados en tiempo real. En la mayoría de los casos en los que se utilizan sistemas embebidos, se requiere que realicen las operaciones rápidamente. Una instrucción debe ser ejecutada de manera inmediata o con un retardo mínimo. Para conseguirlo se suelen utilizar Sistemas Operativos en Tiempo Real (RTOS) para la gestión del tiempo de ejecución de cada tarea.

Los SE son parte central de este nuevo mundo interconectado, en el cual todos nuestros aparatos electrónicos se comunican entre ellos mediante un hardware específico y un software que cumple con tareas en tiempo real. Esta tecnología cada vez está más presente en nuestros hogares y sin duda es un gran avance hacia el bienestar de las personas.

1.1. Descripción del trabajo realizado

La idea principal de este proyecto es disponer de varios sistemas embebidos conectados en red, que se comuniquen mediante cable ethernet. Esta conexión será de tipo punto a punto. Dispondremos de tres placas FRDM-K64F. A una de ellas se le añadirá además un *shield* de expansión que la dotará de más periféricos, de las cuales usaremos los 4 botones, las 4 luces led de cara a la interacción con el usuario, además de los dos potenciómetros y el sensor de temperatura. En el caso de las otras dos placas, disponen de una pantalla LCD y además están conectadas a una placa controladora de dos motores EMG30.

La Figura 1.1 muestra el diagrama de conexiones para ver cómo está construida, la planta piloto.

Como podemos observar, la planta piloto consta de una red local cableada de tipo ethernet. La red está compuesta por tres sistemas empotrados conectados entre sí mediante un switch a través de cable ethernet, siguiendo una topología de estrella. La comunicación se realiza terminal a terminal, pasando por un switch, que reenvía los paquetes enviados de forma que

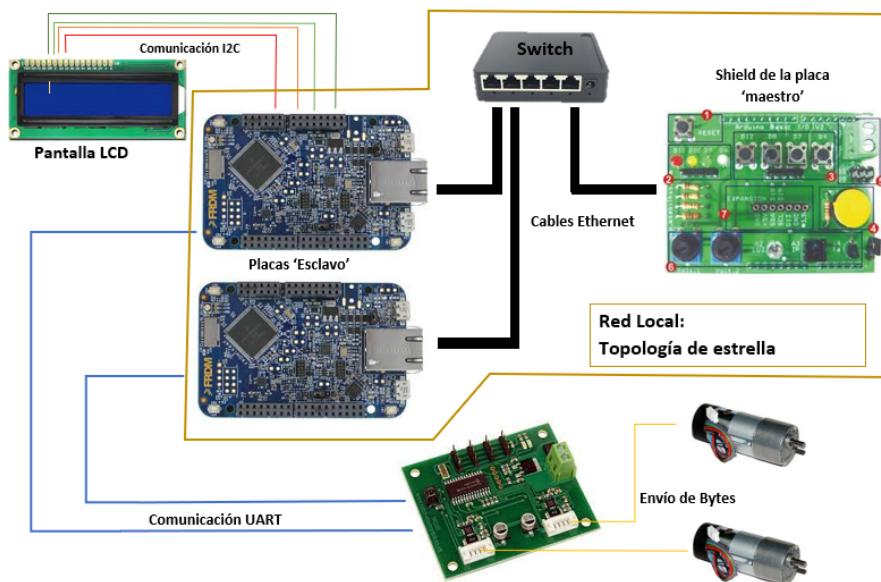


Figura 1.1: Esquema de conexiones del laboratorio.

lleguen correctamente a su destinatario. La placa de color verde que vemos en la imagen anterior es el shield de expansión que está colocado encima de la placa maestro. El shield contiene los dos potenciómetros, 4 botones, 4 leds y el sensor de temperatura con los que se interactuará con el resto de los sistemas embebidos. Por último, la Figura 1.2 muestra la planta piloto.

El funcionamiento consta de los siguientes casos de uso:

- Mediante el potenciómetro 1 fijaremos la velocidad y sentido del giro del motor A. Una vez regulado enviaremos la petición pulsando el botón 1.
- El potenciómetro 2 y el botón 2 se utilizan de la misma manera para fijar la velocidad del motor B.
- El botón 3 nos sirve como parada de emergencia para los dos motores.
- Al pulsar el botón 4 la placa maestro reporta la temperatura captada por el sensor de temperatura y se muestra por pantalla.
- El botón 5 y 6 realizan una petición a la placa controladora de los motores para saber a qué velocidad giran el motor A y el motor B respectivamente.

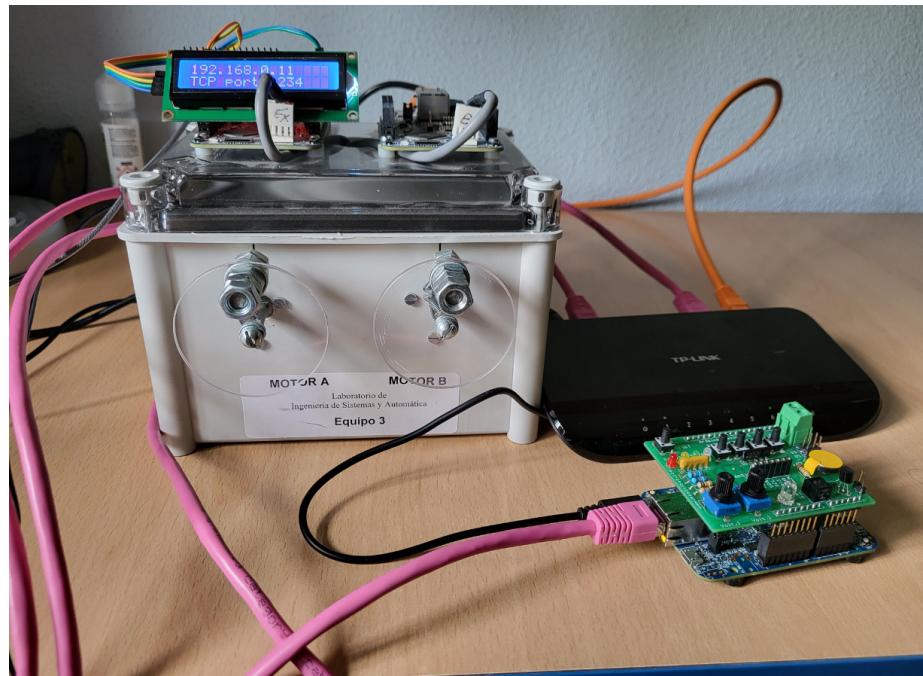


Figura 1.2: Planta piloto del proyecto.

1.2. Estructura de la memoria

La memoria de este proyecto está estructurada de la siguiente manera.

Introducción. Se explican los temas principales en los que se basa tanto la idea como el procedimiento y funcionalidad final del proyecto.

Objetivos del proyecto. Se explican los objetivos del proyecto tanto generales, como técnicos y personales, que se esperan cumplir durante la realización del proyecto.

Conceptos teóricos. Se tratan los conceptos teóricos más detalladamente. Conceptos como el protocolo TCP/IP, las conexiones en red y comunicaciones con otros periféricos.

Técnicas y herramientas. En este apartado se verán las herramientas utilizadas para llevar a cabo este proyecto, así como las técnicas para la correcta organización y gestión del trabajo.

Aspectos Relevantes del desarrollo. En este capítulo se mostrarán algunos conceptos e hitos importantes durante el desarrollo del trabajo.

Trabajos relacionados. Se mostrarán algunos ejemplos de uso real en la actualidad, que utilizan sistemas empotrados para su desarrollo

Conclusiones y líneas de trabajo futuras. Se mencionarán algunas propuestas sobre hacia donde podría evolucionar el trabajo expuesto en esta memoria y a que conclusiones se ha llegado tras haber completado el objetivo del trabajo.

1.3. Anexos

Los anexos consisten en:

Plan del proyecto software. Se muestra la planificación temporal y la viabilidad del proyecto dividida en dos partes, legal y económica.

Especificación de requisitos del software. Se expone un catálogo de requisitos para la utilización de este sistema en un entorno real. Este catálogo viene con la definición de cada una de sus partes.

Especificación de diseño. Exposición de la fase de diseño, el plan procedimental y el diseño arquitectónico de las placas y sensores que componen el sistema.

Manual del programador. Explicación de aquellos conocimientos e ideas que un programador debería conocer para poder entender y seguir desarrollando el código fuente.

Manual de usuario. Contiene una explicación sobre el debido uso y pasos a seguir para que un usuario pueda utilizar adecuadamente el software.

1.4. Contenido adjunto

1. Software para las placas ‘Shield’ y ‘Placa motor A’ y ‘Placa motor B’.
2. Video del funcionamiento de la planta piloto compuesta por dos motores y su placa controladora, una pantalla LCD, tres placas k64f y una placa de expansión.
3. Repositorio de GitHub [7] con todo el contenido de desarrollado.
4. Software documentado.

Objetivos del proyecto

En este apartado de la memoria se detallan de forma concisa los objetivos de la realización de este proyecto. Por un lado tenemos los objetivos generales de este proyecto, por otro los objetivos técnicos y por último los objetivos que yo mismo me marco y por lo que realizo este TFG.

2.1. Objetivos generales

- Configuración de una red local LAN basada en ethernet de sistemas empotrados que sean capaces de conectarse y transmitir información entre ellos mediante los protocolos TCP/IP. Para ello se creará una relación punto a punto con sistemas embebidos.
- Uso de sistemas embebidos y exploración de todo su ecosistema, sensores, tipos de placas, configuración de periféricos usando otros protocolos, configuración de pines y todas sus funcionalidades.
- Demostración de las posibles utilidades que se le pueden dar a estos sistemas en distintos entornos mediante la utilización de distintos periféricos.
- Demostración de la correcta ejecución y uso de la red de sistemas empotrados.

2.2. Objetivos técnicos

- Programación de software en sistemas empotrados. Comprender el funcionamiento de este tipo de sistemas y la gestión de los procesos mediante sistemas operativos en tiempo real (RTOS).

- Configuración y programación de una comunicación I2C para la representación de caracteres por la pantalla LCD.
- Configuración y programación de una comunicación UART para el envío de comandos a los motores y conseguir así su correcto funcionamiento
- Manejo y configuración de conversores analógico-digitales (ADC).
- Programación de aplicaciones de red en un sistema empotrado mediante la implementación del modelo TCP/IP.
- Comprender el funcionamiento de otros protocolos de red como DHCP o la configuración de un router y un switch.

2.3. Objetivos personales

- Comprender el funcionamiento de los sistemas embebidos y sus utilidades en la vida real.
- Conocer el funcionamiento de envío y recepción de paquetes del protocolo TCP/IP.
- Aumentar mi conocimiento en el ámbito de redes. Establecimiento de una conexión, envío de paquetes, etc.
- Usar en un proyecto real las técnicas adquiridas durante la carrera en ámbitos como redes, organización de proyectos y programación.
- Utilizar la metodología Scrum vista en diferentes asignaturas del Grado.
- Emplear el sistema de control de versiones distribuido Git a través de la plataforma de desarrollo GitHub.
- Aprender a utilizar la herramienta LATEX para el desarrollo de documentación profesional.

Conceptos teóricos

En esta sección se detallarán los conceptos teóricos necesarios para comprender el desarrollo del proyecto.

3.1. Sistemas embebidos

En la introducción se mostraban algunas especificaciones y datos sobre que es un sistema empotrado, en este apartado profundizaremos más sobre ello. Veremos las funcionalidades de estos dispositivos y su uso en un entorno real. También se detallarán los tipos de comunicación elegidos para este proyecto y el motivo de su elección en relación a otros pero, primero, ¿Qué es exactamente un sistema empotrado?

Los sistemas embebidos o empotrados son herramientas de computación programadas con una, o varios objetivos concretos. Las grandes ventajas de estos sistemas son que trabajan de forma autónoma, ininterrumpida y sin necesidad de mantenimiento. Estas características hacen que su uso sea muy interesante para el sector industrial y doméstico.

Estos sistemas permiten hacer prácticamente cualquier tipo de tarea ya que, además del hardware mínimo para que se ejecute un programa, se le pueden añadir infinidad de periféricos que aumentan las utilidades de estas placas. Una de las características más importantes en los sistemas embebidos es su capacidad de conexión con otros SE. La comunicación entre sistemas embebidos hace que un SE pueda conocer datos de otro que se encuentra a distancia y actuar en consecuencia. Un ejemplo sencillo sería la conexión entre dos sistemas empotrados, de los cuales uno está conectado a una electroválvula en una habitación y otro cuenta con un sensor de humedad introducido en una maceta que está en otra habitación. Cuando el sensor de

humedad reporta que la humedad es excesiva, el SE se puede comunicar con el otro SE, mediante distintas tecnologías de comunicación, para que cierre la electroválvula.

Hardware

En los sistemas embebidos, prácticamente todos los componentes están integrados en el microcontrolador. En este caso el microcontrolador viene instalado en una placa de demostración que provee la opción de conectar varios periféricos mediante pines o entradas específicas para un periférico en concreto. En la figura 3.3 se muestra el diagrama de bloques de un sistema embebido, concretamente de la placa que se ha utilizado en este proyecto, FRDM K64F.

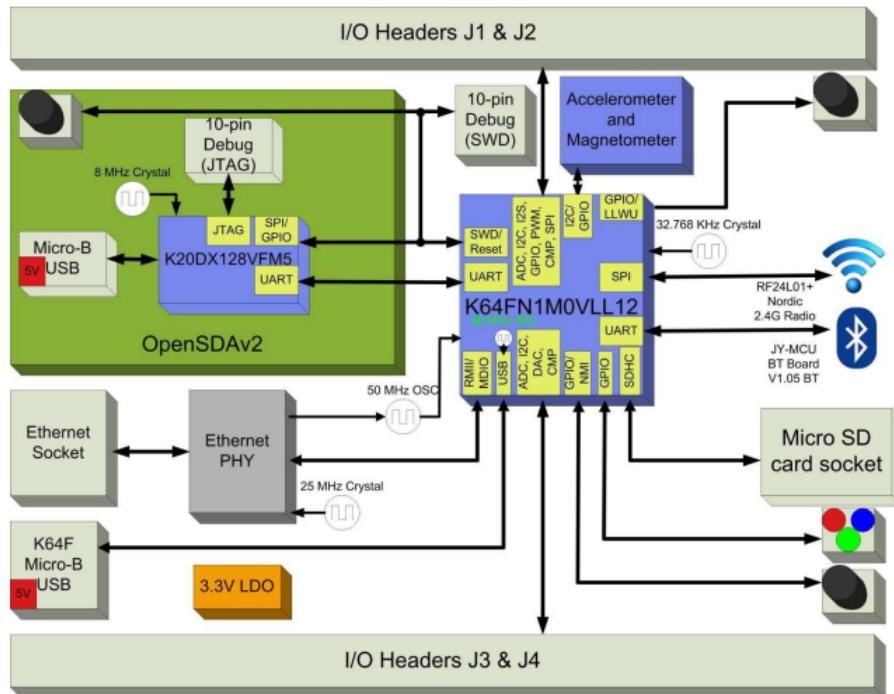


Figura 3.3: Diagrama de bloques placa FRDM K64F.

En el caso de los sistemas embebidos, cada uno de ellos se construye según el propósito específico que se va a realizar con ello, es decir, dependiendo de su objetivo tendrá unas características hardware u otras. Sin embargo, sí que hay algunos componentes mínimos presentes en todas las placas como pueden ser por ejemplo, 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 MCU se encarga de ejecutar las instrucciones del programa cargado en memoria, en otras palabras, gestiona las entradas y salidas de datos. Además de este componente, necesitaríamos de más periféricos para obtener algunas funcionalidades más específicas, algunos de los periféricos que podríamos utilizar son:

- Sensores: humedad, temperatura, ultrasonidos, etc.
- Actuadores: motores, leds, altavoces, etc.
- Dispositivos de interfaz humana.

Software

El software embebido o empotrado reside en memoria de sólo lectura. Con relación al software y hardware utilizados en este proyecto existen 2 posibilidades de cara a cargar el programa en el microcontrolador [3]:

- MBED. Este es el modo en el que vienen las placas por defecto. En este modo, al conectar la placa al ordenador aparecerá como un medio extraíble y deberemos arrastrar los ficheros ‘.bin’ en el que estaría el desarrollo de nuestro programa.
- OpenSDA. (Open Serial and Debug Adapter) o adaptador para depuración serie y comunicación serial en castellano. OpenSDA es la interfaz de bajo costo que ofrece NXP para la depuración y programación de sus microcontroladores. En la Figura 3.4 encontramos su diagrama de bloques.

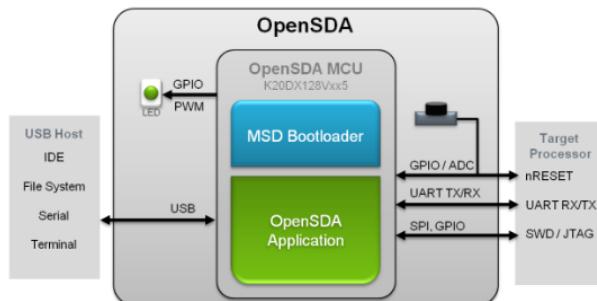


Figura 3.4: Diagrama de bloques de OpenSDA.

En el caso de este proyecto es necesario diferenciar algunos conceptos en lo relativo al software:

SO en tiempo real. [37] Es un sistema operativo que se utiliza para facilitar la gestión de multitareas en dispositivos con recursos y tiempos limitados [5]. Además, las tareas deben ser deterministas en el tiempo de ejecución. Para poder comprender adecuadamente la utilidad de un RTOS debemos conocer los siguientes conceptos:

Tarea. Las tareas, a las cuales también podemos referirnos como procesos o hilos, se ejecutan de manera independiente, esto quiere decir que tienen su propio espacio de memoria. El aislamiento del espacio de memoria se garantiza mediante protección por hardware (MPU) restringiendo su acceso. Las tareas pueden tener diferentes estados según lo determine el RTOS:

Bloqueado. La tarea está esperando un evento que puede ser un bloqueo de tipo mutex o semáforo, o una liberación de espacio en memoria .

Listo. La tarea está lista para ejecutarse en la CPU, pero se mantiene a la espera porque la CPU ya está siendo utilizada.

En ejecución. La tarea se está llevando a cabo.

Existen distintas técnicas para la programación de estos SO:

Expropiativo. Se ejecuta la tarea con mayor prioridad. Incluso se puede interrumpir una tarea en curso si hay otra lista con prioridad mayor. Esto tiene una mayor carga en el microcontrolador ya que tiene que gestionar el cambio de tarea.

No Expropiativo La ejecución de la tarea es ininterrumpida y solo se detiene su ejecución si se detiene o cede el control voluntariamente.

Comunicación entre tareas. Como ha ocurrido en el software realizado es común que algunas variables locales de alguna tarea deban ser utilizadas en otras tareas. Para ello existen dos opciones. La primera y más simple es la utilización de variables globales. La segunda opción es la utilización de colas y buzones que nos modifiquen o lean el valor de esa variable.

Middleware. Este software se encarga de ‘comunicar’ el sistema operativo con los programas. Es un conjunto de librerías que sirven como rutinas para crear una infraestructura que ofrece servicios a los desarrolladores.

Un ejemplo de *middleware* seria la librería lwIP que se ha utilizado en el software del proyecto.

Drivers. Los *drivers* [29] o controladores de dispositivos proveen las instrucciones necesarias para que un dispositivo externo, por ejemplo un periférico, pueda ser controlado por el sistema operativo que el sistema embebido utilice. Dos de los *drivers* más utilizados en este proyecto han sido, enet para configurar el correcto funcionamiento de la conexión en red vía cable ethernet y el *driver* ADC para poder utilizar los potenciómetros.

3.2. Tecnologías de comunicación para SE

Los sistemas empotrados son capaces de comunicarse de varias formas, tanto con periféricos como entre dos o más SE. Algunas tecnologías para comunicarse entre SE serían *bluetooth*, infrarrojos o wifi, entre otros. De esta manera conseguimos que podamos enviar y recibir información de otros sistemas.

Para la comunicación con los periféricos existen otras tecnologías acorde a sus necesidades durante el intercambio de datos. En los próximos dos apartados hablaremos sobre este tipo de tecnologías de comunicación y sus utilidades específicas.

Comunicación mediante TCP/IP

En este apartado vamos a hablar sobre la comunicación mediante TCP/IP. Se explicará cómo se lleva a cabo la comunicación en red vía cable ethernet, que es la que se ha usado en el resultado final del proyecto. Se profundizará en los protocolos que componen el modelo TCP/IP [1] que ha sido utilizado para el establecimiento de conexión en red, envío de paquetes y cierre de la conexión en el software del proyecto.

Modelo TCP/IP

IBM [13] define un protocolo como “Los protocolos son conjuntos de normas para formatos de mensaje y procedimientos que permiten a las máquinas y los programas de aplicación intercambiar información. Cada máquina implicada en la comunicación debe seguir estas normas para que el sistema principal de recepción pueda interpretar el mensaje.”

Podríamos compararlo con las normas sintácticas que seguimos los humanos

para hablar y entendernos los unos a los otros. En este apartado, vamos a centrarnos en el conjunto de protocolos TCP/IP que se dividen en capas o niveles.

La figura 3.5 muestra la disposición de las capas del modelo TCP/IP.

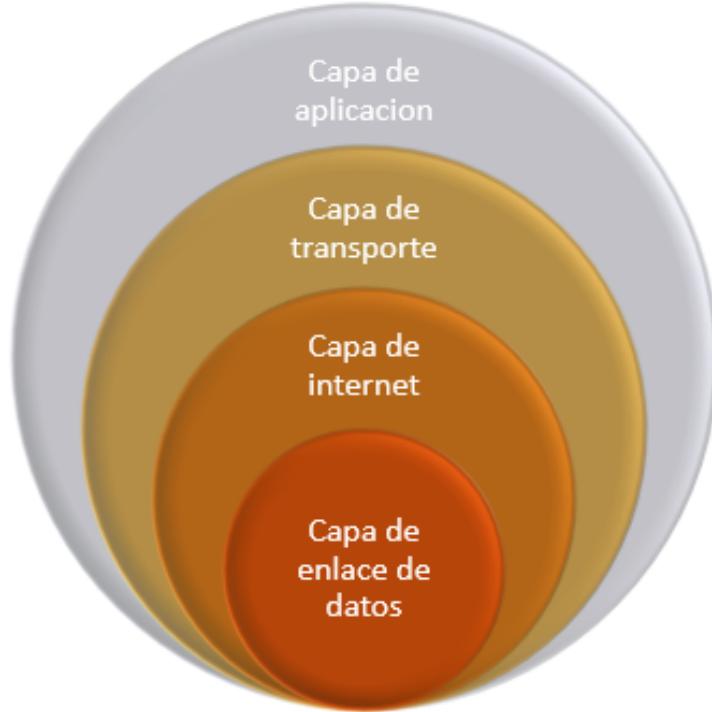


Figura 3.5: Capas del modelo TCP/IP.

Capa de enlace de datos [6] Esta capa se encarga del intercambio de datos entre un *host* y la red a la que se está conectado. Su objetivo es establecer una correcta comunicación entre las capas superiores (Red, Transporte y Aplicación) y el medio físico de transporte de datos. Esta comunicación debe ser segura entre los dos nodos, implementa un sistema de notificación de errores de la topología de la red y el control en la transmisión de las tramas. Esta capa se encarga de la transmisión y direccionamiento de datos sin errores y con seguridad entre nodos que pertenezcan a la misma red (comunicación punto a punto), a través del medio físico . Por otro lado, la capa de Red que es la encargada de la transmisión y direccionamiento de datos entre

host que se encuentran en redes diferentes. Ambas capas trabajan conjuntamente, puesto que la capa de Enlace de datos proporciona sus servicios a la capa de Red, ofreciendo una transmisión de datos confiable a través de un enlace físico. Como resumen de sus principales funciones tenemos:

- Proporciona los servicios y medios necesarios para establecer una comunicación confiable y eficiente entre dos *host*. Para ello agrega una secuencia de bits al inicio y final de los paquetes bajo un formato predefinido formando tramas.
- Añade bits de paridad para ejercer un control de errores en el envío de las tramas. Estos bits son CRC (Códigos Cílicos Redundantes)
- Envía los paquetes de nodo a nodo.
- Regula la congestión de la red.
- Gestiona la velocidad del tráfico de datos.

Capa de Internet La capa de Internet [24] (también denominada capa de red) controla el movimiento de los paquetes alrededor de la red. Se encarga del direccionamiento de los dispositivos y del empaquetado y manipulación de los datos para su correcto envío, entre *host* que pueden estar ubicados en redes geográficamente distintas. Su misión es conseguir que los datos lleguen a su destino aunque no tengan conexión directa. Para conseguir su objetivo, ofrece servicios al nivel superior (capa de transporte) y se apoya en la capa de enlace de datos para encaminar los paquetes, mantener un control de la congestión de la red y el control de errores. Esta capa utiliza las versiones IPV4 e IPV6 para el encaminamiento, control y notificación de errores, existe además, IGMP (Internet Group Management Protocol) y MLD (Multicast Listener Discovery) que se usan para establecer grupos de difusión múltiple.

El protocolo IP determina el destinatario del mensaje mediante 3 elementos:

- Dirección IP: Dirección del equipo.
- Máscara de subred: una máscara de subred le permite al protocolo IP establecer la parte de la dirección IP que se relaciona con la red.

- **Gateway(Puerta de enlace):** le permite al protocolo de Internet saber a qué equipo enviar un datagrama, si el equipo de destino no se encuentra en la red de área local.

Capa de Transporte La capa de transporte [27] se encarga de la segmentación de datos y ensamblaje de las partes dentro de los canales de comunicación. Esta capa se asegura de conseguir que la conexión de datos sea fiable entre dos dispositivos. Para ello, envía los datos en paquetes y se asegura de que el otro equipo indique que ha recibido los paquetes correctamente. En otras palabras, se encarga de facilitar la comunicación lógica entre dispositivos. Esta comunicación puede utilizar dos protocolos:

TCP es un protocolo orientado a la conexión, proporciona un conjunto completo de servicios para aquellas aplicaciones que lo necesiten y su uso se considera fiable. Con el uso de puertos consigue 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.

UDP A diferencia de TCP, UDP no utiliza ningún mecanismo de establecimiento de la conexión, no dispone de mecanismo que aseguren, la transferencia fiable de los segmentos, por lo tanto, determinados segmentos 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 pérdida de información.

Capa de aplicación Esta capa nos ofrece la posibilidad de acceder a otras capas para usar sus servicios [16]. Además a esta capa, pertenecen protocolos como:

- Sistema de nombres de dominios (DNS): este protocolo relaciona nombres de Internet en direcciones IP.
- Telnet: Proporciona acceso remoto a servidores y dispositivos de red.
- Protocolo simple de transferencia de correo (SMTP): Se usa para enviar mensajes y archivos adjuntos de correo electrónico.
- Protocolo de configuración dinámica de host (DHCP): Este protocolo asigna una dirección IP y direcciones de máscara de subred, de gateway predeterminado y de servidor DNS a un host.

- Protocolo de transferencia de hipertexto (HTTP): Se encarga de transferir los archivos que conforman las páginas Web de la World Wide Web.
- Protocolo de oficina de correos (POP): Es utilizado por los clientes de correo electrónico para recuperar el correo electrónico de un servidor remoto.
- Protocolo de acceso a mensajes de Internet (IMAP): Se utiliza para recuperar correo electrónico.

Los protocolos nombrados, son usados tanto por los dispositivos de origen como de destino durante la comunicación.

3.3. Comunicaciones para los periféricos

Como ya mencionamos anteriormente, la mayor funcionalidad que podemos dar a estas placas viene con la condición de poder comunicarnos entre ellas y con otros periféricos. Veamos algunos tipos de conexión, además de vía ethernet o wifi, que hemos utilizado en este trabajo.

UART

UART, 'Universal Asynchronous Receiver-Transmitter' o en castellano Receptor-transmisor asíncrono universal. Su funcionamiento es sencillo. Utiliza solamente dos hilos para transmitir los datos. Las conexiones son cruzadas entre los dos dispositivos, es decir el pin de transmisión (TX) del dispositivo que emite el mensaje, estará conectado al pin receptor (RX) del dispositivo que recibe la comunicación y viceversa. Esta comunicación es asíncrona, por lo que no utiliza relojes para el envío y recepción de mensajes. Para que ambos dispositivos sepan cuando tienen que empezar y dejar leer bits se añaden a cada envío bits de inicio y parada. Es importante que ambos equipos tengan la misma tasa de Baudios configurada para que los bits se lean adecuadamente, de no ser así, la comunicación fallaría ya que un dispositivo enviaría bits a una velocidad diferente de la que se leen, ocasionando lecturas erróneas. La velocidad predeterminada suele ser de 9600 baudios. En la Figura 3.6 podemos ver los bits que forman una trama UART [26]:

Como podemos observar los bits que forman la trama son:

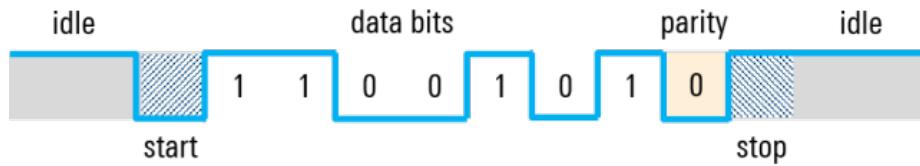


Figura 3.6: Trama UART.

- Bits de inicio y parada. Puesto que UART es asíncrono, son necesarios para que el receptor sepa desde dónde tiene que empezar a leer y cuando parar.
- Bits de datos. Son la carga útil, puede haber de 5 a 9 bits por trama.
- Bit de paridad. Este bit se utiliza para la detección de errores en las tramas. Existen dos tipos de paridad, paridad par o impar.

Por otro lado tenemos la comunicación USART ‘Universal Synchronous and Asynchronous Receiver-Transmitter’ que puede realizar procesos de comunicación con relojes. En la Tabla 3.1 podremos ver sus diferencias [2] principales:

Características	USART	UART
Modo	Semidúplex	Dúplex
Velocidad	USART es mayor	UART es menor.
Funcionamiento	Señales de datos y de reloj	Señales de datos
Datos	Bloques	Bytes
Complejidad	Más complejo	Más simple de utilizar

Tabla 3.1: Diferencias USART y UART

El principal motivo de hacerlo con UART en vez de con USART, es su sencillez. Además los comandos de los motores se envían en bytes y no en bloques, por lo que es más directo hacerlo con UART.

I2C

El protocolo I2C, ‘Inter-Integrated Circuit’ usa dos líneas para comunicarse con otros dispositivos, además de las líneas de tierra y voltaje. Por un lado tenemos SCL (línea de reloj en serie) y por otro lado tenemos SDA

(línea de datos), esta última es la encargada de transmitir la información. Mediante estas dos líneas el protocolo I2C es capaz de enviar datos de forma segura. Veamos algunos de los bits importantes que, tanto el maestro y esclavo [12] pueden utilizar para formar y leer las tramas:

- Inicio ó Start – S
- Parada – P
- Confirmación – ACK
- NoConfirmación – NACK
- Lectura-/Escritura – L/W
- 7 bits para la dirección del dispositivo esclavo/maestro
- 8 bits de dirección (para algunos sensores pueden ser 16 bits)
- 8 bits de datos

Existen distintos modos de comunicación, dependiendo de esos modos se utilizarán distinto orden y número de bits para formar las tramas. Los modos principales de comunicación son:

- Maestro-Transmisor y Esclavo-Receptor. Este modo se usa cuando se desea configurar un registro del esclavo I2C.
- Maestro-Receptor y Esclavo-Transmisor. Se usa cuando queremos leer información del sensor I2C.

Este tipo de comunicación suele estar destinada al intercambio de datos con módulos o sensores y se usa en arquitecturas maestro-esclavo. Varios esclavos pueden estar conectados a un maestro al mismo tiempo pero, como es lógico, cuantos más esclavos estén conectados al maestro mayor latencia habrá en el envío y recepción de datos. Esto se debe a que solo se puede usar el bus I2C para comunicar un esclavo al mismo tiempo. También hay que tener en cuenta que dos maestros no pueden utilizar el mismo puerto. Veamos las características del maestro y el esclavo:

- Maestro: Dispositivo que proporciona un reloj para la comunicación. Sus funciones principales son:

1. Iniciar la comunicación – S
 2. Enviar 7 bits de dirección – ADDR
 3. Generar 1 bit de Lectura o Escritura – R/W
 4. Enviar 8 bits de dirección de memoria
 5. Transmitir 8 bits de datos –
 6. Confirmar la recepción de datos – ACK – ACKnowledged
 7. Generar confirmación de No-recepción, NACK – No-ACKnowledged
 8. Finalizar la comunicación
- Esclavo: Dispositivo que utiliza el reloj del maestro. Sus funciones principales son:
 1. Enviar información en paquetes de 8 bits.
 2. Enviar confirmaciones de recepción, llamadas ACK.

En nuestro caso, utilizamos este tipo de comunicación para el uso de la pantalla LCD.

3.4. Periféricos

En esta parte de la memoria, vamos a ver las especificaciones de los periféricos utilizados en este proyecto

Potenciómetro y sensor de temperatura

El potenciómetro [25] es un elemento hardware que busca determinar un potencial eléctrico en una conexión. Generalmente, se consigue mediante la comparación de la potencia de entrada y la de salida. Esta diferencia de potencial es lo que se conoce como voltaje. Su funcionamiento es sencillo, cuenta con tres resistencias, dos en cada uno de los extremos y una tercera resistencia con la cual podemos interactuar, permitiéndonos aumentar o disminuir su valor.

Un sensor de temperatura [4] es un dispositivo que transforma los cambios de temperatura (magnitud física) en señales eléctricas(voltaje). Por lo general, está formado por un sensor encapsulado en una cubierta protectora. Entre el sensor y la cápsula encontramos un material que es conductor térmico y que permite traspasar rápidamente estos cambios de temperatura. Existen tres tipo de sensores de temperatura descritos perfectamente en [28]:

- Termopares. Funcionan mediante un principio de generación de una corriente entre dos metales diferentes unidos que tienen diferente comportamiento eléctrico en función de la temperatura. La señal generada se procesa y da lugar a una medición de temperatura. Son equipos sencillos, baratos y con una precisión suficiente para su uso en edificación. Sin embargo, tienen una respuesta lenta.
- Termorresistencias. Están constituidas por resistencias cuya conductividad varía en función de la temperatura, lo cual genera una señal que, una vez procesada, permite obtener la medición de temperatura. Su velocidad de respuesta depende de la masa de la resistencia.
- Sensores electrónicos. Funcionan mediante dispositivos electrónicos que generan una corriente o señal en función de la temperatura. Son equipos con una respuesta mucho más rápida, pero más caros.

Técnicas y herramientas

En este capítulo se exponen todas las herramientas que se han utilizado y las técnicas que se han seguido para poder desarrollar el proyecto de una manera sencilla y organizada.

4.1. Técnicas: metodologías ágiles

Como metodología ágil [14] para el desarrollo y organización del proyecto, se ha utilizado la metodología SCRUM, ya que además de ser la más usada en el entorno laboral real, ha sido una de las que hemos trabajado durante las asignaturas del grado. Pero para poder entender su potencial veamos que es SCRUM.

SCRUM es un proceso en el que se aplican un conjunto de buenas prácticas que permiten trabajar de forma colaborativa y organizada mediante el uso de Sprint y tareas. El desarrollo de todo el proyecto se separa en diferentes Sprint, que contienen una serie de tareas necesarias para conseguir un software final. Cada uno de los Sprint debe ser completado en unas 3 semanas por lo que en la reunión inicial, se deben discutir qué tareas estarán en el sprint actual. El equipo deberá valorar la dificultad y el tiempo de cada una de las tareas y tratar de realizar la mayoría de las tareas posibles en ese sprint. En la finalización de cada sprint se deberá entregar un resultado válido. En la metodología SCRUM, es de gran importancia la comunicación con el resto del equipo, por lo que al final de cada día se realiza una reunión de unos 15 minutos en la que se exponen problemas, avances y dificultades. En la Figura 4.7 encontramos un diagrama que sirve como resumen del funcionamiento de esta técnica.

Las figuras más importantes de esta metodología son:

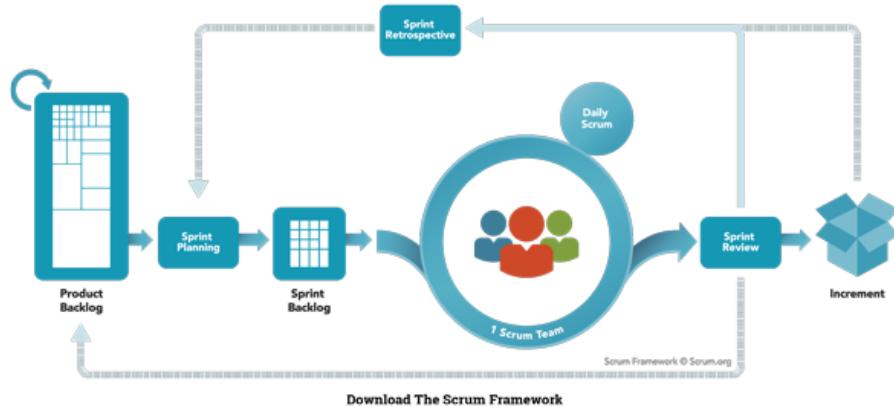


Figura 4.7: Diagrama resumen de la Metodología SCRUM. [22]

El Product Owner Es el encargado de hablar con el cliente y exponer sus peticiones al SCRUM master y al equipo de desarrollo. En otras palabras, es quien define las tareas, condiciones y prioridades centrándose en el retorno sobre la inversión, en inglés Return on Investment (ROI), del proyecto.

El SCRUM Master Es quien dirige y lidera al SCRUM team. Es el encargado de guiar al equipo y hacer que se cumplan las normas de la metodología SCRUM. Esta figura está muy ligada al Product Owner, ayudándole a maximizar el ROI.

El SCRUM team Es el equipo desarrollador del software del proyecto. Suele estar formado por grupos de entre 3 y 9 personas que deben tener buenas capacidades referidas a la organización y gestión de tareas y procesos.

Partiendo de los actores anteriores podemos definir que, tanto el cargo de Product Owner y SCRUM Team han sido interpretados por mí, a diferencia de la figura de SCRUM Master la habrían representado mis tutores.

Todo esto nos ayuda a poder lograr un resultado final óptimo en el desarrollo de proyectos complejos, dónde se necesita obtener resultados en un corto margen de tiempo y donde las tareas pueden variar. Se usa en entornos complejos donde prima la competitividad, la flexibilidad y la productividad.

4.2. Herramientas hardware

Veamos con más detalle los componentes hardware de este proyecto.

Placa FRDM K64F

Esta placa es un conjunto hardware que, además de contener el microcontrolador, nos ofrece las características necesarias para poder conectar varios periféricos a este controlador.

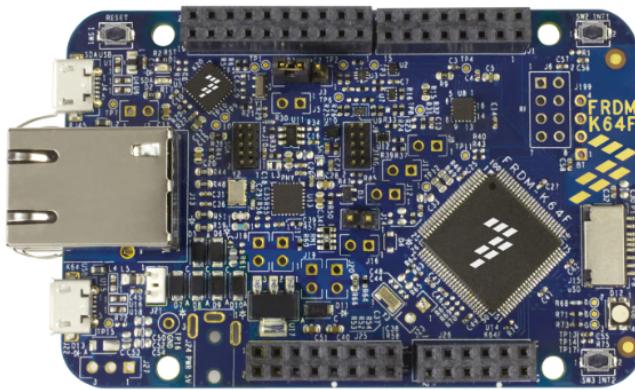


Figura 4.8: Placa FRDM K64F.

Sus características son:

- Microcontrolador MK64FN1M0VLL12 MCU: (120 MHz, 1 MB de memoria flash, 256 KB RAM, USB de bajo consumo, sin cristal, y 100 Low profile Quad Flat (LQFP))
- Núcleo ARM Cortex M4F. [36]
- Interfaz USB de doble función con conector USB micro-B
- LED RGB
- Acelerómetro y magnetómetro FXOS8700CQ
- Dos botones de usuario
- Opción de alimentación flexible - OpenSDAv2 USB, Kinetis K64 USB y fuente externa

- Fácil acceso a la entrada/salida del MCU a través de Arduino™ R3 compatible Conectores de E/S
- Circuito de depuración programable OpenSDAv2 compatible con el software CMSISDAP Interface que proporciona:
 - Interfaz de programación flash del dispositivo de almacenamiento masivo (MSD)
 - Interfaz de depuración CMSIS-DAP a través de una conexión USB HID sin controlador proporcionando depuración de control de ejecución y compatibilidad con herramientas IDE
 - Interfaz de puerto serie virtual
 - Proyecto de software CMSIS-DAP de código abierto
 - Ethernet
 - SDHC
 - Módulo RF adicional: nRF24L01+ Nordic 2.4GHz Radio
 - Módulo Bluetooth adicional: JY-MCU BT board V1.05 BT

Se puede encontrar más información acerca de los microcontroladores en el libro: [9].

Placa de expansión Arduino Basic I/O



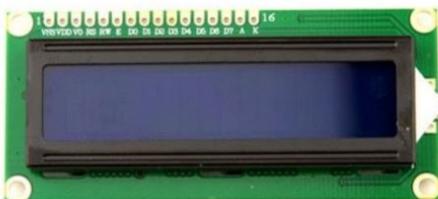
Figura 4.9: Placa de expansión Arduino basic I/O.

Esta placa se sitúa sobre la placa K64F conectándola directamente sobre sus pines y dotándola de varias características, como por ejemplo: Altavoz, 4

leds de diferentes colores, 4 botones, sensor de infrarrojos, 2 potenciómetros, entre otros. De esta manera se consigue dotar a la placa de más funciones.

LCD

Una pantalla LCD (*liquid crystal display*) o pantalla de cristal líquido, es una pantalla delgada y plana. Puede estar formada por píxeles monocromos o en color, colocados sobre una fuente de luz. Estos píxeles son una capa de moléculas situada entre dos electrodos transparentes, dos filtros polarizados. Sin cristal líquido entre el filtro polarizante, la luz quedaría bloqueada al tratar de pasar el segundo filtro.



(a) Pantalla LCD



(b) Modulo IIC/I2C

Figura 4.10: Pantalla LCD con modulo IIC/I2C.

Tanto para el desarrollador como sobre todo para el cliente final que recibe el proyecto, es interesante el uso de una pantalla que muestre información sobre las operaciones que está realizando el sistema empotrado. En este proyecto, se ha utilizado una pantalla de 2 líneas de 16 caracteres cada línea. Por lo general, estas pantallas necesitan de la conexión de más de 15 pines para la transmisión de datos, la alimentación, la iluminación y el control de la transmisión. Sin embargo, en este caso la pantalla utilizada incorpora un módulo I2C que deja la conexión en tan solo 4 pines que serían:

SCL (*System Clock*) es la línea de los pulsos de reloj que sincronizan el sistema.

SDA (*System Data*) es la línea por la que se mueven los datos entre los dispositivos.

VCC Es el pin por el que reciben energía. En este caso serán 5 voltios.

GND Es el pin de tierra o masa que sirve para cerrar el circuito.

La comunicación utilizada en este tipo de elementos es I2C.

Motores

El motor recibe el nombre de EMG30 (codificador, motor, reductor 30:1). Es un motor de corriente continua de 12v, totalmente equipado con codificadores y un reductor 30:1.



Figura 4.11: Motor EMG30 utilizado en la planta piloto.

Es ideal para aplicaciones robóticas pequeñas o medianas. También incluye un condensador de supresión de ruido estándar en el motor. Las conexiones del motor son:

1. Purple (1) Hall Sensor B Vout
2. Blue (2) Hall sensor A Vout
3. Green (3) Hall sensor ground
4. Brown (4) Hall sensor Vcc
5. Red (5) + Motor
6. Black (6) - Motor

Estas conexiones van conectadas a la placa que se muestra en la Figura 4.12 y será la encargada de controlar los bytes recibidos y enviados.

La placa K64F envía a través de conexión UART o I2C los bytes con las instrucciones para los motores a la placa controladora y posteriormente esta placa enviará la información al propio motor que realizará las acciones correspondientes a ese comando. Por otro lado, los bytes enviados consisten, en la mayoría de las instrucciones, en un byte de sincronización, un byte para elegir el motor, puesto que se pueden conectar hasta dos motores, y

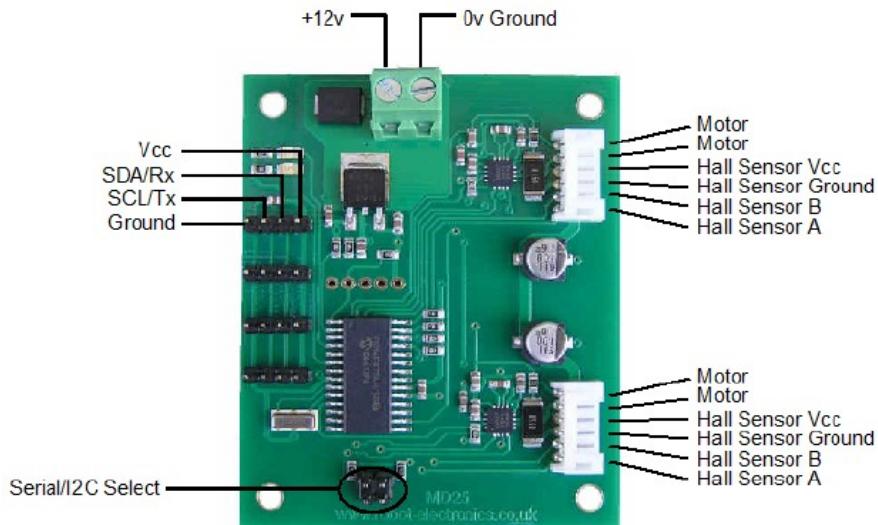


Figura 4.12: Placa controladora de los motores.

el byte con la instrucción. Estos bytes se pueden enviar tanto en formato hexadecimal como decimal u octal. Vamos a ver los modos y comandos que se pueden utilizar con estos motores. En cuanto a los modos disponemos de 4 modos:

- Modo 0. Si se elegimos usar el modo 0 entonces los registros de velocidad son velocidades literales en el rango de 0 (retroceso total) 128 (parada) 255 (avance total).
- Modo 1. El modo 1 es similar al modo 0, excepto que los valores de velocidad se interpretan como valores con signo. El rango es -128 (retroceso total) 0 (parada) 127 (avance total).
- Modo 2. En el modo 2 la velocidad 1 controle la velocidad de ambos motores, y la velocidad2 se convierte en el valor de giro. Los datos están en el rango de 0 (retroceso total) 128 (parada) 255 (avance total).
- Modo 3. El modo 3 es similar al modo 2, excepto que los valores de velocidad se interpretan como valores con signo. Los datos están en el rango de -128 (retroceso total) 0 (parada) 127 (avance total).

En la Tabla 4.2 se muestran los comandos (CMD) más importantes y su descripción.

CMD	Nombre	Bytes Env-Rec	Descripción
0x21	Get Speed 1	2 - 1	Obtener la velocidad del motor A
0x22	Get Speed 1	2 - 1	Obtener la velocidad del motor B
0x2A	Get Acceleration	2 - 1	Devuelve la aceleración
0x31	Set Speed 1	3 - 0	Fija la velocidad del motor A
0x32	Set Speed 1	3 - 0	Fija la velocidad del motor B
0x33	Set Acceleration	3 - 0	Fija la aceleración
0x34	Set Mode	3 - 0	Fija el modo
0x38	TimeOut OFF	2 - 0	No apagar en 2s sin comunicación
0x39	TimeOut ON	2 - 0	Apagar tras 2s sin comunicación

Tabla 4.2: Comandos Motores EMG30.

Sensor de Temperatura: LM35



Figura 4.13: Sensor de temperatura, LM35.

El sensor de temperatura LM35, es un circuito electrónico cuyas características se describieron en el capítulo anterior [3.4](#), de salida analógica, que permite medir temperaturas. Este sensor proporciona un voltaje proporcional a la temperatura en la que se encuentra. Tiene un rango de medición desde -55 grados centígrados, hasta un máximo de 150 grados. Sus características principales son:

- Resolución: 10mV por grado centígrado.
- Voltaje de alimentación. Por ejemplo, este sensor se puede alimentar desde 4Vdc hasta 20Vdc.

- Tipo de medición: Salida analógica.
- Numero de pines: 3 pines, GND, VCC y VSalida.

Wifi: módulo ESP8266

El módulo ESP8266 [8] se trata de un chip integrado con conexión Wifi y compatible con el protocolo TCP/IP. El objetivo principal es dar acceso a cualquier microcontrolador a una red. La gran ventaja del ESP8266 es su bajo consumo. Soporta IPv4 y los protocolos TCP/UDP/HTTP/FTP. La Figura 4.14 muestra los pines del módulo ESP8266.

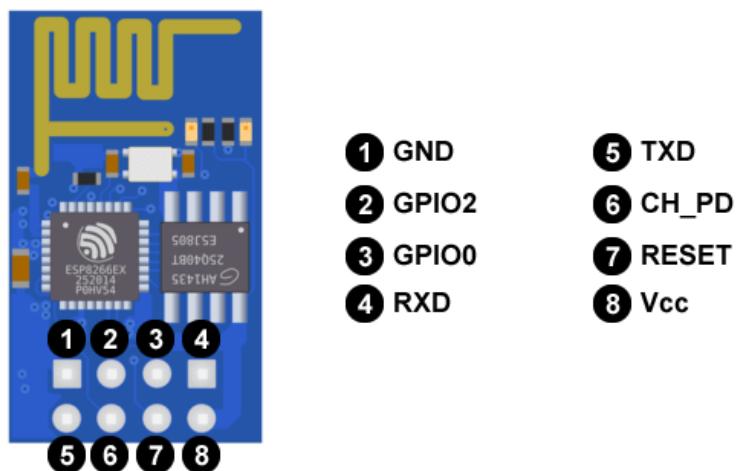


Figura 4.14: Pines del módulo wifi 8266.

ESP32 soporta las siguientes características:

- Soporta los principales buses de comunicación (SPI, I2C, UART).
- Comunicación unicast encriptada y sin encriptar.
- Se pueden mezclar clientes con encriptación y sin encriptación.
- Permite enviar hasta 250 bytes de carga útil.
- Se pueden configurar *callbacks* para informar a la aplicación si la transmisión fue correcta.
- Largo alcance, pudiendo superar los 200 metros en campo abierto.

Pero también tiene sus limitaciones:

- El número de clientes con encriptación está limitado. Esta limitación es de 10 clientes para el modo Estación, 6 como mucho en modo punto de acceso o modo mixto.
- El número total de clientes con y sin encriptación es de 20.
- Sólo se pueden enviar hasta 250 bytes.

Uso del módulo ESP8266

El módulo ESP8266 [20] es un dispositivo TTL “Serial to Wireless Internet” funciona mediante el envío de comandos ‘AT’. En la Tabla 4.3 se explican los comandos más importantes. Los comandos están ordenados según los pasos que se deberían seguir para establecer comunicación entre un módulo y un servidor.

Comando	Descripción
AT+CWMODE='X'	<ul style="list-style-type: none"> ■ 1 = Modo estación (cliente) ■ 2 = Modo AP (huésped) ■ 3 = Modo AP + Estación (modo dual)
AT+CWLAP	Lista APs disponibles
AT+CWJAP='ssid','Pswd'	El módulo se conecta a la red con el nombre ssid indicado y la contraseña pwd suministrada.
AT+CIPMUX='X'	<ul style="list-style-type: none"> ■ 0 = Conexión única ■ 1 = Múltiples conexiones, hasta 4
AT+CIPSERVER=Mode,puerto	<p>Configura el módulo como servidor donde el modo:</p> <ul style="list-style-type: none"> ■ 0 = Borrar servidor ■ 1 = Crear servidor <p>puerto: número del puerto, por defecto es el 333</p>
AT+CIFSR	Retorna la dirección IP local del módulo como cliente.
AT+CIPSEND	Envía datos sin adornos cada 20ms. El módulo retorna »"después ejecutar el comando, si se recibe el comando -++"se regresa al modo comando.

Tabla 4.3: Comandos AT.

4.3. Herramientas software

Veamos que herramientas software he utilizado a lo largo del desarrollo del proyecto.

KDS IDE. Kinetis ® Design Studio (KDS) es un entorno de desarrollo integrado complementario para los MCU Kinetis que permite una edición, compilación y depuración sólidas de sus diseños. Kinetis está basado en software gratuito de código abierto que incluye Eclipse, GNU Compiler Collection (GCC), GNU Debugger (GDB) y otros, Kinetis Design Studio IDE ofrece a los diseñadores una herramienta de desarrollo simple sin limitaciones de tamaño de código. Además, el software Processor Expert ® habilita su diseño con su base de conocimiento y ayuda a crear aplicaciones potentes con unos pocos clics.

Este fue el IDE sobre el que empecé el proyecto y del que tras un par de semanas terminaría migrando a MCUXPRESSO 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. Este IDE está basado en eclipse y sobre el que se ha desarrollado el proyecto. El IDE de MCUXpresso [21] ofrece funciones avanzadas de edición, compilación y depuración. Añade también vistas de depuración específicas de MCU, rastreo y creación de perfiles de código, además de depuración multinúcleo y herramientas de configuración integradas. Es un IDE muy completo que presenta una interfaz simple para el usuario pese a gran número de opciones y configuraciones que ofrece.

4.4. FreeRTOS

Algunos apartados más atrás ya hablamos sobre que era un *RTOS* [3.1], en esta sección veremos cual ha sido el sistema operativo que he utilizado en este proyecto.

FreeRTOS [30] es robusto, tiene un tamaño reducido y una compatibilidad del 100 % con el microcontrolador. Es el sistema operativo en tiempo real más utilizado en microcontroladores pequeños en el mundo.

Además cuenta con varias demostraciones preconfiguradas y detalladas referentes al Internet de las cosas (IoT). AWS se encarga de su mantenimiento y soporte a largo plazo. Como hemos podido observar es un software completísimo, referente mundial, fácil de usar y con un soporte que saca actualizaciones constantemente.

LwIP

La librería light weight IP pretende dar un servicio basado en el protocolo TCP/IP. Este software fue desarrollado por Adam Dunkels en Computer and Laboratory de Arquitecturas de Redes (CNA) en el Instituto Sueco de Ciencias de la Computación (SICS).

El enfoque de la implementación de lwIP TCP/IP es reducir el uso de RAM sin dejar de tener un TCP a escala completa. Esto hace que lwIP sea adecuado para su uso en sistemas embebidos con decenas de kilobytes de RAM libre y espacio para alrededor de 40 kilobytes de código ROM.

Protocolos implementados

- IP (Protocolo de Internet, IPv4 e IPv6), incluido el reenvío de paquetes múltiples interfaces de red.
- ICMP (Protocolo de mensajes de control de Internet) para mantenimiento y depuración de redes.
- IGMP (Protocolo de gestión de grupos de Internet) para la gestión del tráfico de multidifusión.
- MLD (descubrimiento de oyentes de multidifusión para IPv6). Tiene como objetivo cumplir con RFC 2710. Sin soporte para MLDv2.
- ND (descubrimiento de vecinos y configuración automática de direcciones sin estado para IPv6). Tiene como objetivo cumplir con RFC 4861 (descubrimiento de vecinos) y RFC 4862 (Auto-configuración de direcciones).
- UDP (Protocolo de datagramas de usuario) que incluye extensiones UDP-lite experimentales.
- TCP (Protocolo de control de transmisión) con control de congestión, estimación de RTT y recuperación rápida/retransmisión rápida.
- API nativa/sin formato para un rendimiento mejorado.

- API de socket similar a Berkeley opcional.
- DNS (resolución de nombres de dominio).

Docklight. Docklight [18] es una herramienta de prueba, análisis y simulación de protocolos de comunicación en serie. Este programa se utiliza para captar las comunicaciones serie y Uart. En este proyecto se ha utilizado para poder comunicarnos con la placa de una manera más sencilla a la hora de tener que introducir comandos. Para la captura de estos mensajes es necesario conocer el puerto de salida 'comm' y la velocidad en baudios a la que se transmiten los datos, además del número de bits, paridad, etc. Por otro lado, este software cuenta con algunas características añadidas como poder guardar comandos que usamos de forma continua o poder ver la información en ascii, binario o hexadecimal que en algunas ocasiones puede ser necesario.

Termite. Este software es muy parecido a Docklight pero en este caso cuenta con una interfaz más simple. En este caso el programa se configura con los parámetros del otro dispositivo con el que nos vamos a comunicar y se reciben o envían datos.

Packet Sender. Este programa ha sido de gran ayuda puesto que se utilizó para realizar las pruebas de recepción y envío de paquetes a las tres placas. Packet Sender es una utilidad de código abierto que permite enviar y recibir paquetes TCP y UDP. También admite conexiones TCP mediante SSL, generación de tráfico intenso, solicitudes HTTP GET/POST y generación de paneles.

4.5. Herramientas de documentación

Veamos las herramientas utilizadas para documentar y trabajar sobre el proyecto.

Textmaker. [31] Es una herramienta gratuita que nos ayuda a escribir documentos de texto integrando las funciones necesarias para poder realizar documentos con Latex. Además este software es multiplataforma.

Latex. [32] Latex es un compositor de textos destinado a la creación de documentos profesionales que requieran una alta calidad tipográfica. Se utiliza, por lo general, en la realización de artículos y libros científicos que incluyen elementos y expresiones matemáticas.

Bibtex. [33] Es una herramienta utilizada para la creación de referencias bibliográficas. Genera un formato para cada una de las referencias con los datos aportados por el usuario y generalmente se utiliza en la realización de documentos con LaTeX.

4.6. Herramientas de comunicación

Para la comunicación con mis tutores para aclarar dudas y resolver fallos se han utilizado las siguientes herramientas.

Microsoft Outlook. [34] Es un gestor de correo electrónico desarrollado por Microsoft y que podemos encontrar en la suite de Microsoft office.

Microsoft Teams. [35] De nuevo es un programa desarrollado por Microsoft. En este caso se trata de una plataforma para realizar reuniones virtuales, cuenta también con salas de chat y la posibilidad de generar documentos compartidos.

4.7. Herramientas de gestión de proyectos

En el primer apartado hablábamos de la técnica de organización y desarrollo de proyectos mediante metodologías ágiles: SCRUM, en este apartado veremos cuales son las herramientas con las que conseguimos facilitar estas tareas.

GitHub. [10] GitHub es una plataforma pensada para que los desarrolladores puedan alojar su repositorios de código de forma segura en la nube. Además incluye un sistema de control de versiones conocido como Git.

Por otro lado, también permite el desarrollo colaborativo entre distintos desarrolladores y ofrece todas las herramientas para poder trabajar con SCRUM.

GitKraken. [19] GitKraken es una aplicación que nos permite manejar Git y por tanto nuestros archivos de GitHub de forma más sencilla. Esta herramienta se encuentra disponible para todas las plataformas.

Aspectos relevantes del desarrollo del proyecto

A medida que avanzaba con el proyecto han ido surgiendo algunos hitos importantes que voy a remarcar en este capítulo.

5.1. IDE: Kinetis vs MCUXpresso

Como ya vimos en el apartado de herramientas se han utilizado dos *IDEs*, ambos dos creados por la misma empresa, NXP. En un primer momento se comenzó utilizando Kinetis Design Studio puesto que mis tutores me facilitaron algunas prácticas realizadas con ese programa de cara a familiarizarme con ello. Tras la realización de las prácticas enseguida me di cuenta de que estaba bastante anticuado y que las nuevas versiones de *drivers* y *middleware* no funcionaban correctamente. Es por ello que decidí migrar hacia MCUXpresso. Este cambio hizo que tuviera que volver a familiarizarme con el nuevo IDE y la manera de programar el hardware a bajo nivel con la herramienta *Config Tools*. Aunque KDS estuviera anticuado, en lo referente al uso de esta herramienta era algo más sencillo. A favor de la MCUX diré que incluye decenas de programas de ejemplo con los que poder probar y comprender su funcionamiento, además de un montón de *drivers* que habilitan la instalación de un gran número de sensores. También se puede observar que MCUXpresso cuida más la limpieza y simplicidad de las interfaces a la hora de interactuar con ella. Una vez comprendes como funciona la programación de periféricos, relojes y pines, lo cual no es sencillo en un principio, programar un sistema embebido se convierte en algo sencillo.

5.2. FRDM K64F vs Arduino

Este proyecto también podría haberse desarrollado con las placas Arduino UNO, las cuales son muy parecidas en cuanto a su funcionamiento y uso de periféricos. Además, utilizar Arduino hubiera sido más sencillo debido a que los pines vienen ya configurados, al igual que los relojes y tan solo hubiera sido enchufar y programar las funciones de las tareas que quisiéramos. Arduino también cuenta con una comunidad mayor por lo que tendríamos librerías más simples y avanzadas y mayor información en Internet de cara a resolver fallos. Entonces, ¿Por qué decidí utilizar las FRDM K64F?

La respuesta es simple, el objetivo de la realización de este trabajo no era simplemente desarrollar una planta piloto que realizara una tarea útil. El objetivo principal no es que los motores funcionen adecuadamente, o la pantalla, o el sensor de temperatura, etc. El objetivo principal era aprender sobre el funcionamiento de los sistemas embebidos. Es por ello que se decidieron utilizar estas placas, ‘puesto que ofrece al usuario mayor control sobre ella por la necesidad de ser programada a bajo nivel. Desde un principio se ha pretendido centrar los esfuerzos en comprender como se realiza la configuración de sus pines y relojes, etc. En un entorno real, se deben conocer el funcionamiento de estos sistemas a bajo nivel puesto que, cada proyecto desarrollado en el entorno laboral necesitará que el SE cumpla con unos requisitos específicos y no exceda de ellos para disminuir el coste y tamaño del sistema. En cada proyecto se desarrollan unas placas específicas para esos requisitos y es conveniente saber su funcionamiento interno para poder diseñarlas correctamente.

Por otro lado, en el caso del homólogo a la placa FRDM K64F, que sería el Arduino Uno, no hubieramos podido utilizar ni FreeRTOS puesto que no dispone de reloj de tiempo real, ni lwIP debido a que su potencia es menor a la de las placas K64F. Además, con Arduino tampoco hubieramos podido *debuguear* puesto que no usa OpenSDA para cargar el software. Debido a todo esto se optó por la utilización de las placas FRDM-K64F.

5.3. RTOS

En el caso de los sistemas operativos en tiempo real encontramos, algunas alternativas a FreeRTOS. Las características y conceptos más importantes de FreeRTOS ya los vimos en el apartado 4.4. Como alternativa a este sistema operativo encontramos:

Embedded Operating System (embOS), es un sistema operativo en tiempo

real, desarrollado por la empresa SEGGER Microcontroller. Está diseñado para ser utilizado como base para el desarrollo de aplicaciones integradas en tiempo real para una amplia gama de microcontroladores. El funcionamiento es prácticamente el mismo.

MQX es otra opción de sistema operativo en tiempo real propuesto por NXP. Es un SO que ofrece una API sencilla y una arquitectura modular que hace que este software sea escalable.

El motivo de haber elegido la utilización de FreeRTOS es su presencia en un mayor número de proyectos que sus competidores. Esto hace que existan comunidades en Internet, que nos brindan mayor información y soluciones para su correcta utilización. Además de que su propio manual ya nos ofrece los pasos a seguir, es realmente sencillo de utilizar, una vez has aprendido los conocimientos básicos de su funcionamiento.

5.4. Metodologías Ágiles

Como alternativas a SCRUM teníamos dos metodologías ágiles muy utilizadas y quizás más sencillas de implementar:

Extreme Programming XP Esta metodología es muy utilizada en pequeñas empresas durante sus comienzos y consolidación. Se basa en centrar sus esfuerzos en la comunicación entre clientes y empleados, potenciando las relaciones personales mediante el trabajo en equipo y promoviendo la comunicación y la eliminación de tiempos muertos. Sus principales fases son:

1. Diseño y planificación del proyecto con el cliente.
2. Programación por parejas dentro del equipo de forma que ambos puedan intercambiar conocimientos consiguiendo mejores resultados.
3. Realización continua de pruebas de código.

Kanban La metodología Kanban se basa en el uso de un *layout* con columnas, generalmente tres, en las que se muestran las tareas que quedan por hacer: 'Pendientes', las que están en curso: 'En proceso' y las que ya se terminaron: 'Terminadas'. Cada usuario o equipo tiene la posibilidad de aumentar el número de columnas según les sea de mayor utilidad. De esta manera se tiene conocimiento sobre el estado del

proyecto en tiempo real, mejorando la productividad y eficiencia del desarrollo del trabajo. Las principales ventajas de esta metodología son:

1. Facilidad para la planificación de tareas.
2. Mejora en el rendimiento de trabajo del equipo.
3. Visión global de un solo vistazo.
4. Los plazos de entregas son continuos.

Pese a estas dos alternativas se decidió usar SCRUM ya que es la metodología más usada y estudiada durante el grado. Además, el uso de la herramienta GitHub hace que sea fácil de usar y también consigue que la generación de código quede bien expuesta y organizada.

5.5. Dificultades y Problemas

Durante el desarrollo de todo el proyecto he tenido algunos inconvenientes tanto personales como técnicos. Veamos algunos de ellos:

Ya desde un principio perdí tiempo por el cambio de IDE y tener que familiarizarme de nuevo con las interfaces del programa.

Por otro lado, la parte de configuración de pines, periféricos y relojes es algo complicada, sobre todo al principio del proyecto, ya que durante el grado no se ha visto nada parecido. A la hora de buscar información sobre como realizar algún procedimiento, tanto en código, como en configuración de pines, no encontraba demasiada información. El uso de estas placas es algo específico ya que están pensadas para usuarios con ciertos conocimientos en el ámbito de los SE.

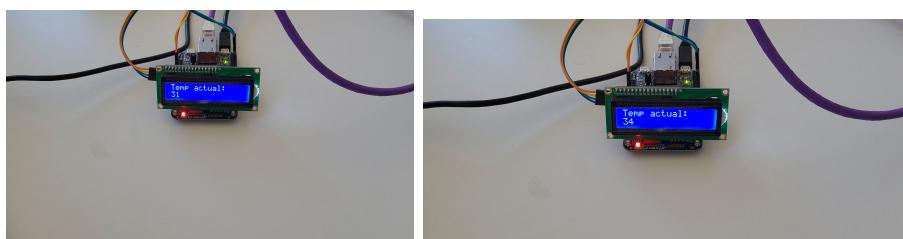
Relacionado con la organización y planificación fue complicado puesto que mi situación personal fue cambiando a lo largo del proyecto. Al principio de su realización no tenía demasiadas obligaciones, lo que me permitía tener un cierto control en cuanto a horas y horarios diarios. A mitad del desarrollo comencé a realizar las prácticas curriculares y el tiempo libre disminuye complicando tener una estabilidad diaria para realizar este proyecto.

Volviendo al desarrollo, la adaptación del tipo de comunicaciones, sobre todo la UART, al envío de los comandos de los motores, fue algo

compleja. Estas dificultades vinieron dadas en gran parte por los cambios de tipos.

5.6. Caso de uso Real

El ultimo aspecto relevante de este proyecto que quiero destacar es el uso de este proyecto en un entorno real. Durante el tiempo que he estado realizando el TFG también he estado cursando las prácticas curriculares en la empresa Kronospan S.L. Esta empresa me propuso poder usar parte del proyecto en sus instalaciones, aunque solo fuera a modo de representación de un caso de uso real. Por ello decidimos que una gran utilidad de estos sistemas para la empresa y departamento en el que estaba, podía ser la medición de la temperatura de su CPD en tiempo real. Para desempeñar esta tarea se puso una placa con el sensor de temperatura dentro del CPD conectada en red por cable y por otro lado se puso otro SE con una pantalla LCD y un led rojo en la mesa en la que trabajo. El sensor de temperatura enviaba la temperatura actual si pasaba de un máximo determinado y el SE de mi mesa reportaba la temperatura con el led rojo y la pantalla que mostraba la temperatura en ese momento. Esto se utilizó durante una semana a forma de demostración, por supuesto al tratarse de una multinacional ellos ya disponían de sus propios sensores en cajas de Rittal con sistemas anti incendios, varios sistemas de aire acondicionados, etc. A continuación, se muestran algunas imágenes de la ejecución de ese proyecto.



(a) Temp: 31 grados

(b) Temp: 34 grados

Figura 5.15: Caso de uso Kronospan.

En la Figura 5.16 vemos como la placa maestro está introducida en el CPD para la captación de la temperatura.



Figura 5.16: Placa ‘Maestro’ en el CPD.

Estos han sido los hitos que más dificultades me han causado durante la realización del TFG.

Trabajos relacionados

Este capítulo mostrara algunos trabajos que se han hecho con sistemas empotrados. De esta manera podremos hacernos a la idea de algunas de las aplicaciones reales de estos sistemas. Veamos los distintos sectores en los que se utilizan los SE:

6.1. SE en equipos médicos

Muchos de los aparatos que se utilizan de forma periódica en la sanidad, usan un microcontrolador [15]. La Figura 6.17 muestra un ejemplo de un medidor de presión sanguínea:

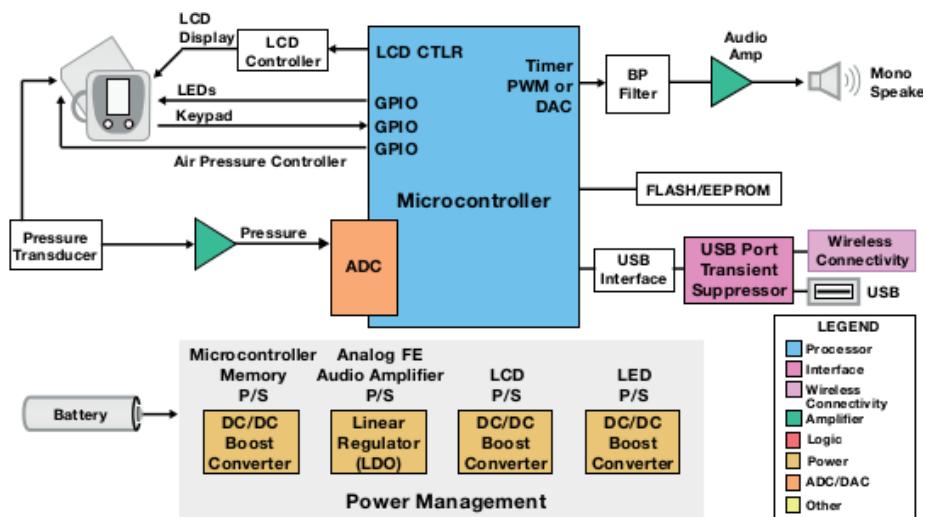


Figura 6.17: Esquema arquitectónico de un SE de medición de sangre.

En la imagen podemos apreciar como existen dos partes importantes, por un lado tenemos todos los elementos que necesitan corriente y cómo se conectan a una fuente de alimentación y por otro lado está el microcontrolador al que se conectan todos los periféricos que nos ayudan a llevar a cabo la tarea que queremos realizar. Algunos de esos periféricos son un altavoz o una pantalla para poder interactuar con el usuario que use el sistema. También tenemos el sensor de presión sanguínea que utilizará un valor analógico y lo convertirá en uno digital (ADC) para poder saber la presión sanguínea del cliente. Otro ejemplo sería la utilización de un SE para controlar un desfibrilador (DESA). Un desfibrilador sirve para recuperar el ritmo cardíaco. En la actualidad se pueden encontrar este tipo de sistema de primeros auxilios, en algunos establecimientos, generalmente en ambientes deportivos o donde el flujo de gente es de avanzada edad. Estos aparatos cuentan con un microcontrolador que mediante un altavoz explica como debes usarlo y aplica de forma cíclica una serie de descargas dependiendo del estado del paciente y con poca interacción humana. En este caso el esquema arquitectónico sería muy parecido a la figura que presentaba arriba cambiando el flujo de aire por un sensor de ritmo cardíaco.

6.2. SE en gestión de la energía

Todos en nuestra casa disponemos de una caldera y un termostato con el que controlamos la temperatura. Este termostato podría ser perfectamente un sistema empotrado. En este ejemplo además vamos a poder observar la importancia y utilidad de que los SE puedan comunicarse entre ellos y no solo con otros periféricos. Veamos la siguiente Figura 6.18:

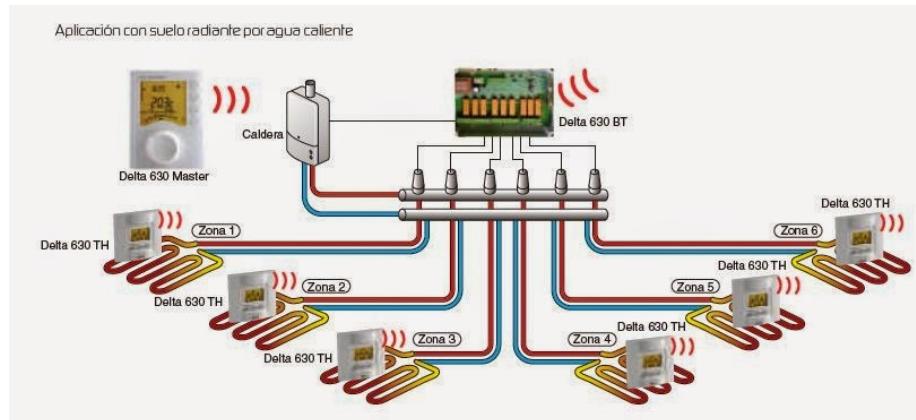


Figura 6.18: Sistema de Calefaccion con SE.

Como podemos apreciar en la Figura 6.18 se muestran 2 sistemas embebidos, además de uno por cada zona a calentar. El primer sistema embebido, ‘SE1’, es el que utilizará el usuario para encender la caldera y configurar la temperatura de cada zona. El SE1 se comunica con la caldera y con el ‘SE2’. El SE2 es el encargado de abrir o cerrar las electroválvulas que dejarán pasar el agua caliente a cada una de las zonas calentándolas. El SE2 se comunicará con cada uno de los sistemas empotrados que encontramos en cada zona para conocer la temperatura a la que están. Como hemos visto de esta manera podremos conseguir distintas temperaturas en cada zona dependiendo de nuestras necesidades sin necesidad de tener que subir la calefacción de todas las habitaciones de la casa o de todas las oficinas de la empresa en caso de que alguna, por ejemplo, no se esté usando en ese momento.

6.3. Conexión desde otros dispositivos

En este apartado me gustaría hacer mención del trabajo de fin de máster que desarrolló un compañero, [23] , de esta misma universidad en el año 2018. Su TFG estaba relacionado con los sistemas empotrados. En este caso, él realizó la conexión a Internet de una sola placa que se controlaba a través de un servidor que él gestionaba. Utilizando las interfaces propuestas en el servidor podía gestionar las luces led de la placa, tanto su color como intensidad. Me parece muy interesante la idea de poder controlar estas placas ya no solo mediante sus botones, potenciómetros, etc, sino el hecho de poder hacerlo desde un ordenador o dispositivo móvil. Esto es una gran idea por varias razones:

- En primer lugar no necesitamos de un SE que nos haga de ‘puente’ para poder configurar las acciones de otros puesto que utilizamos un móvil u ordenador, elementos que siempre tenemos a mano.
- Además de esta manera conseguimos poder comunicarnos con las otras placas desde cualquier sitio y no desde donde esté físicamente ese sistema.

Roberto lo hizo, en este caso, mediante una conexión a Internet, pensando en poder comunicarnos con él aunque estemos a grandes distancias pero como se comentó en los primeros capítulos existen más formas en caso de que lo tengamos relativamente cerca, como serían bluetooth o incluso radiofrecuencias.

Conclusiones y Líneas de trabajo futuras

En este capítulo veremos las conclusiones llegadas tras haber terminado el proyecto. Además se expondrán algunas ideas de cómo podríamos continuar con este mismo trabajo o realizar algunos distintos también basados en los sistemas embebidos y las comunicaciones entre ellos.

7.1. Conclusiones

En este proyecto se han trabajado conocimientos adquiridos en varias asignaturas entre las cuales destacarían:

- Programación concurrente y de tiempo real.
- Administración de redes y sistemas.
- Programación.
- Gestión de proyectos.

Por supuesto, este trabajo también me ha hecho darme cuenta de muchos conocimientos que no tengo y he tenido que aprender. Otra de las partes más importantes de la realización del TFG es la gestión emocional. El hecho de saber organizarte para un trabajo de tantas horas con una fecha de entrega tan lejana, junto con la realización de las demás actividades de tu vida, hacen que tengas que mejorar tus habilidades en materias como la responsabilidad, organización, fuerza de voluntad, perseverancia, entre

otras, y por supuesto al mismo tiempo tienes que luchar contra otras muchos vicios contraproducente tales como la pereza y tener que sobreponerte a las adversidades. En cuanto a la organización y planificación del proyecto ha sido muy enriquecedor el hecho de utilizar GitHub y la metodología Scrum. De esta manera he podido ver cómo se realiza un proyecto de este tipo en un ejemplo real y aprender cómo funciona esta herramienta. Además usarla ha permitido que en caso de fallos en la programación del sistema embebido dispusiera siempre de una copia anterior que me servía como *backUp*.

Dicho esto es importante hacer hincapié en que, habiendo terminado el trabajo puedo dar por completados los objetivos técnicos 2.2, generales 2.1 y personales 2.3.

Personalmente ha sido de gran interés el uso de sistemas embebidos en mi proyecto, puesto que a medida que me adentraba más en ese campo y comprendía su utilización cada vez surgían más y más usos que se le pueden dar, tal y como veremos en el próximo apartado, 'líneas de trabajo futuras'. Además su estructura permite ampliar sus funcionalidades fácilmente por lo que cada vez se utilizan más en distintos ámbitos como, industrial, electrónico, informático o incluso en la salud. Otra de las grandes ventajas de usar estos sistemas en la actualidad es el hecho de que se puedan comunicar entre ellas estando a pocos metros o incluso a kilómetros de distancia.

Es de justicia decir también, que al igual que estos sistemas permiten un gran número de usos, también tienen puntos débiles. Debemos poner de manifiesto que su programación a bajo nivel puede resultar bastante complicada al principio y en muchas ocasiones se necesitan placas específicamente creadas para algunos periféricos concretos, debido a su voltaje, número de pines o funcionamiento. Esto genera que cada vez que se hace un proyecto, dependiendo de la dificultad y dimensiones de este, se deba crear un sistema específico. Además por lo general disponen de poca memoria y por lo tanto requieren que las librerías y el propio código y variables utilizadas no sean demasiado extensos, ni requieran del almacenamiento de muchos datos.

7.2. Líneas de trabajo futuras

Este proyecto, al tratarse de un sistema embebido, puede ser continuado en varios puntos. Las opciones son prácticamente infinitas y solo se debe imaginar un objetivo para poder continuar. Algunos puntos importantes sobre los que se podría trabajar serían:

La integración de la conexión vía wifi con el módulo ESP8266 o semejantes.

De este modo podríamos replicar el mismo proyecto u otro diferente sin depender de la utilización de cables de red.

Continuando con más conexiones, también podríamos utilizar la conexión bluetooth tanto para conectar la placa o bien a un móvil o a un ordenador de modo que pudiéramos utilizar un hardware que siempre tenemos a mano para gestionar los parámetros necesarios.

Otra línea de futuro sería implementar algún sistema de domótica, pudiendo llegar al punto de utilizarlo en tu propia casa. Algunos ejemplos serían, continuando con la idea del sensor de temperatura conectar o bien por infrarrojos o por bluetooth un SE al aire acondicionado y que este variara la temperatura automáticamente, incluso la implementación de un sistema de reconocimiento de voz.

Estas placas mediante protocolos IoT y servicios como por ejemplo Azure IoT Hub permiten conectar, supervisar y administrarlos. Además se pueden conectar a la nube para subir y descargarse datos de manera que pueden crear informes sobre el funcionamiento de un sensor.

Relacionado con la seguridad se podría tratar de cifrar todas las comunicaciones que haya entre dispositivos mediante bluetooth o Internet. En el caso de la conexión a Internet la propia pila que se ha utilizado en este proyecto, lwIP, dispone de un tipo cifrado conocido como *Transport Layer Security* (TLS) de manera que las comunicaciones entre placas estuvieran cifradas.

En otros ámbitos con estas placas se podría realizar proyectos para realizar una acción, como por ejemplo abrir una puerta, mediante reconocimiento biométrico, bien facial o bien dactilar.

Bibliografía

- [1] Avast Academy. ¿Qué es TCP/IP y cómo funciona? URL: <https://www.avast.com/es-es/c-what-is-tcp-ip>.
- [2] acervolima. Diferencias entre UART y USART. URL: <https://es.acervolima.com/diferencia-entre-usarty-uart/>.
- [3] altapps. EMBOSS. URL: <https://es.altapps.net/soft/embos>.
- [4] Proyecto arduino. ¿Que es un sensor de Temperatura? URL: <https://proyectoarduino.com/sensor-de-temperatura/>.
- [5] Wellings A.. Burns, A. *Sistemas de tiempo real y lenguajes de programación*. Addison Wesley., 2003.
- [6] ComDatosGrupo4. Capítulo 2. La Capa de Enlace de Datos. URL: https://sites.google.com/site/comdatosgrupo4/contenidos/cap2_capa-enlace-datos.
- [7] Enrique del Olmo. Comunicación TCP/IP con sistemas empotrados. URL: https://github.com/eod1001/TFG_CONEXION_FRDMK64F, 2022. Mi repositorio de GitHub.
- [8] Luis del Valle Hernández. ESP8266 todo lo que necesitas saber del módulo WiFi para Arduino. URL: <https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/>, 2018.
- [9] Antonio. Díaz Estrella. *Teoría y diseño con microcontroladores de Freescale: familia Flexis de 32 bits MCF51QE*. McGraw-Hill Interamericana de España S.L., 2008.

- [10] Yúbal Fernández. Qué es Github y qué es lo que le ofrece a los desarrolladores. URL: <https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>, 2019.
- [11] Michael Gonzalez Harbour. Los sistemas embebidos en tiempo real están presentes en nuestra vida cotidiana y lo estarán cada vez más. URL: https://web.unican.es/noticias/Paginas/2021/septiembre_2021/Harbour-digitalizacion-sistemas-embebidos.aspx, 2021.
- [12] HetPro. I2C – Puerto, Introducción, trama y protocolo. URL: <https://hetpro-store.com/TUTORIALES/i2c/>, 2018.
- [13] IBM. Protocolos TCP/IP. URL: <https://www.ibm.com/docs/es/aix/7.1?topic=protocol-tcip-protocols>, 2021.
- [14] iebschool. Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. URL: <https://alexmarket.medium.com/gitkraken-ea27eb8e8301>, 2014.
- [15] Texas Instrument. Sistemas empotrados en equipos médicos. URL: <https://cempotrado.blogs.upv.es/2014/03/20/ti-healthtech-sistemas-empotrados-en-equipos-medicos/>, 2013.
- [16] Itroque. Protocolos de la capa de aplicacion. URL: <http://itroque.edu.mx/cisco/cisco1/course/module10/10.1.1.4/10.1.1.4.html>.
- [17] Acervo Lima. ¿cuál es la diferencia entre microcontrolador y microcomputador? URL: <https://es.acervolima.com/cual-es-la-diferencia-entre-microcontrolador-uc-y-microprocesador-up/>.
- [18] Marco Flachmann Oliver Hegelbacher. Docklight. URL: https://github.com/eod1001/TFG_CONEXION_FRDMK64F. [Internet; descargado 28 marzo 2021].
- [19] Alex Market. GitKraken. URL: <https://alexmarket.medium.com/gitkraken-ea27eb8e8301>, 2019.
- [20] Naylamp Mechatronics. Tutorial ESP8266 Parte I. URL: https://naylampmechatronics.com/blog/21_tutorial-esp8266-parte-i.html.

- [21] NXP. MCUXpresso Integrated Development Environment (IDE). URL: <https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE>.
- [22] Scrum ORG. The home of Scrum. URL: <https://www.scrum.org/>, 2022.
- [23] Roberto Peña. Comunicación TCP/IP con sistemas empotrados. URL: <https://github.com/rpc0027>, 2018. Repositorio de compañero de master Roberto Peña.
- [24] Lopez Quesada. Estructura TCP/IP. URL: http://dis.um.es/~lopezquesada/documentos/IES_1314/LMSGI/curso/xhtml/xhtml11/html/pag3.html.
- [25] José Luis R. Como funciona un potenciómetro. URL: <https://como-funciona.co/un-potenciometro/>.
- [26] Rohde Schwarz. ¿Que es UART? URL: https://www.rohde-schwarz.com/es/productos/test-y-medida/osciloscopios/educational-content/que-es-uart_254524.html.
- [27] Google Sites. Capa de Transporte. URL: <https://sites.google.com/site/informaticaredesdecomputadoras/unidad-2-capas-superiores-del-modelo-osi-y-tcp-ip/2-2-capa-de-transporte>.
- [28] SP solerpalau. Tipos de sensores de Temperatura. URL: <https://www.solerpalau.com/es-es/blog/sensor-temperatura/>, 2017.
- [29] Guille Ven. Que es un Driver o Controlador? URL: https://www.tecnologia-informatica.com/que-son-drivers-controladores/#%C2%BFQue_son_los_drivers_o_controladores_de_dispositivos?, 2021.
- [30] FreeRTOS Web. FreeRTOS. URL: <https://www.freertos.org/>, 2015.
- [31] Wikipedia. TextMaker — Wikipedia La enciclopedia libre. URL: https://github.com/eod1001/TFG_CONEXION_FRDMK64F.
- [32] Wikipedia. LaTeX — Wikipedia, La enciclopedia libre. URL: <https://es.wikipedia.org/w/index.php?title=LaTeX&oldid=84209252>, 2015.

- [33] wikipedia. Bibtex. URL: <https://es.wikipedia.org/wiki/BibTeX>, 2021.
- [34] wikipedia. Microsoft Outlook. URL: https://es.wikipedia.org/wiki/Microsoft_Outlook, 2022.
- [35] wikipedia. Microsoft Teams. URL: https://es.wikipedia.org/wiki/Microsoft_Teams, 2022.
- [36] Joseph Yiu. *The Definitive Guide to the ARM Cortex-M3 and Cortex-M4 Processors*. Elservier Inc, 2014.
- [37] Lim Jia Zhi. Sistemas operativos en tiempo real (rtos) y sus aplicaciones. URL: <https://www.digikey.es/es/articles/real-time-operating-systems-and-their-applications>, 2021.



Este obra está bajo una [licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional](#).