UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Comunicacion con Sistemas Embebidos

Presentado por Enrique del Olmo Dominguez
en Universidad de Burgos — 13 de junio
de 2022
Tutor: Alejandro Merino Gomez y Daniel
Sarabia Ortiz

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Alejandro Merino Gomez y D. Daniel Sarabia Ortiz, profesores del departamento de Ingeniería Electromecánica y área de Ingeniería de Sistemas y Automática.

Expone:

Que el alumno D. Enrique del Olmo Dominguez, 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 del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de junio de 2022

Vº. Bº. del Tutor:

D. Alejandro Merino Gomez

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 montón de tareas. Estas tareas pueden ser desde la inclinación de una cama de hospital a la gestión de de unas cintas transportadoras de elementos en una industria. En este proyecto podremos ver como se pueden programar sistemas embebidos para comunicarse entre si mediante Internet. También veremos una simulación de para que se podrían utilizar este tipo de sistemas en un ámbito real. Para ello dispondremos de un laboratorio de pruebas con varias placas FRDM-K64F, dos servomotores, una pantalla LCD y un shield de expansion.

Descriptores

Sistemas embebidos, Sistemas empotrados, conexion wifi, conexion ethernet, comunicacion 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, wifi connection, ethernet connection, communication between systems, Internet of things (IoT), NXP, uart, I2C, ADC, FRDM K64F.

Índice general

Índice de figuras

Índice de tablas

Introducción

Los sistemas embebidos o empotrados (SE) están muy presentes en nuestra vida cotidiana. Podemos referirnos a estos sistemas como un microcontrolador, el cual se diferencia de los microcomputadores en el modo de realizar las diferentes tareas, como veremos en los siguientes párrafos. Teniendo en cuenta que el cometido de los SE es la realización de pequeñas tareas programadas por un desarrollador para un fin concreto encontramos que, por lo general, todos cuentan con las mismas características, tamaño reducido, bajo coste y bajo consumo. Además se les pueden añadir más componentes tales como sensores de humedad, calor, ultrasonidos y un largo etc, que dotarán de mayores funcionalidades a las placas. 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.

Para que todos estos dispositivos puedan funcionar adecuadamente se necesitan o al menos se prefiere tener a estos sistemas interconectados entre si ya que a partir de la transferencia de datos 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 que son y cómo funcionan los sistemas empotrados en tiempo real. Para ello vamos a ver la diferencia entre un ordenador y un microcontrolador. La gran diferencia es una cuestión de escala. Un ordenador se suele utilizar para entornos complejos y tareas extensas que a su vez requieren de otras tareas en cascada. En cambio un microcontrolador se programa para llevar a cabo algunas tareas específicas, en un entorno controlado y generalmente pequeño.

Un ejemplo para terminar de diferenciarlos seria, en el caso de querer medir la humedad y temperatura en una maceta usaríamos un microcontrolador programando esas dos tareas para que se ejecuten en tiempo real de forma cíclica, sería un desperdicio utilizar un microprocesador para esto.

Cabe destacar como este tipo de sistemas brilla en una de las ultimas ramas de la informática como es el Internet de las cosas (IOT). Los SE son parte central de este nuevo mundo interconectado, en el cual todos nuestros aparatos electrónicos se comunican entre ellos mediante un hardware especifico 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

Antes de continuar, vamos a explicar en qué consiste este proyecto. La idea principal es disponer de una red de sistemas embebidos comunicados entre ellos mediante cable de red. 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 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 unos motores. La idea principal del ejemplo de uso es que mediante los potenciómetros de la placa de expansión regulemos el giro de los motores. Esto lo podemos ver más detallado en los requisitos funcionales explicados en los anexos. Dispondremos también de un switch para poder enviar paquetes mediante el cable ethernet utilizando el protocolo TCP/IP.

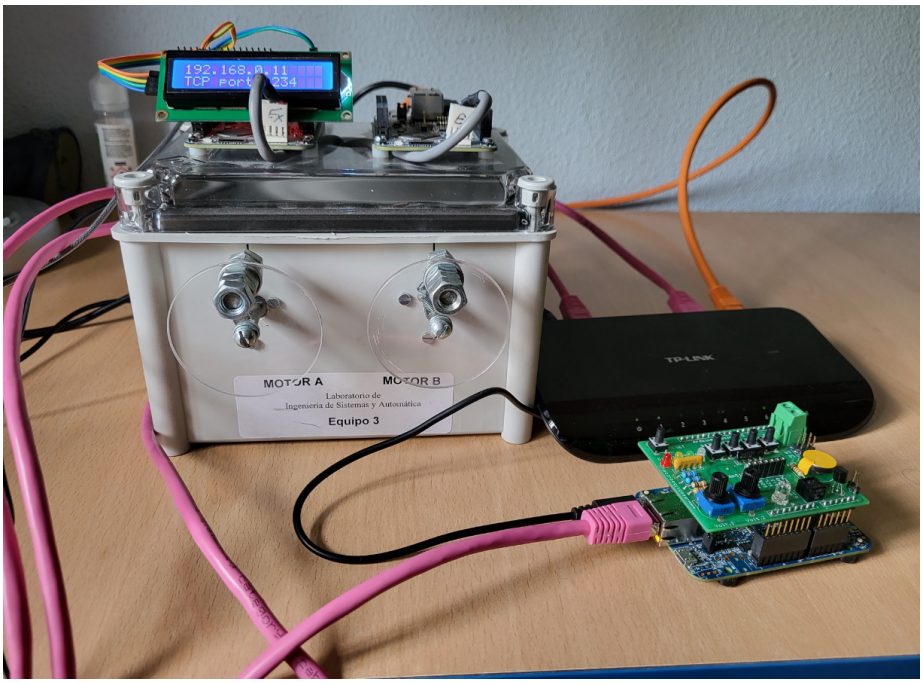

Figura 1.1: Laboratorio del Proyecto

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. En este apartado se dan unos conceptos básicos para que el lector pueda entender adecuadamente el objetivo del trabajo y como se ha desarrollado.

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, funcionamiento del wifi y el ethernet, etc.

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

Se añade información en forma de anexos a la ya presentada en la memoria que ayudarán a la comprensión de la misma y a conocer el funcionamiento del resultado del proyecto.

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 "Shieldz "Placa motor Az "Placa motor B".

2. Laboratorio compuesto por dos motores y su placa controladora, una pantalla lcd, tres placas k64f y una placa de expansión.

3. Repositorio de GitHub [**EOD1001**] con todo el contenido de desarrollador

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. Hay tres tipos de objetivos. Por un lado tenemos los objetivos generales de un proyecto, por otro los objetivos técnicos que tiene este proyecto concretamente y por ultimo los objetivos que yo mismo me marco y por lo que realizo este TFG.

2.1. Objetivos Generales

- Configuración de una red de sistemas empotrados que sean capaces de conectarse y transmitir paquetes entre ellos mediante conexión ethernet. Para ello se creará de una relación maestro-esclavo con sistemas embebidos.

- Uso de sistemas embebidos y exploración de todo su ecosistema, sensores, tipos de placas, relojes internos, 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.
- Configuración y programacion de una comunicación I2C para la representación de caracteres por la pantalla LCD.
- Utilización de la comunicación UART para el envío de comandos a los motores y conseguir así su correcto funcionamiento
- Investigación para conseguir que la interacción de este tipo de sistemas con las personas sea mas sencilla.
- Manejo y configuración de conversores analógico y digitales (ADC).
- Programación de aplicaciones de red en un sistema empotrado mediante la implementacion del protocolo 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. Capas de red, organigrama de envío de paquetes e nacionalización de una conexión, 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 la obtención de documentación profesional.

Conceptos teóricos

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

3.1. Sistemas Embebidos

Ya hablamos anteriormente de que es un sistema empotrado, en este apartado profundizaremos mas 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 un sistema embebido?

Los sistemas embebidos o empotrados son herramientas de computación programadas con una o varias funcionalidades concretas. 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 ademas del hardware mínimo para que se ejecute un programa se le pueden añadir infinidad de periféricos que nutren de distintos usos a las placas mas básicas. Al mismo tiempo el hecho de que se puedan comunicar entre ellas añade ademas varios usos muy interesantes, como podremos ver en el apartado de Trabajos Relacionados.

Hardware

En los sistemas embebidos prácticamente todos los componentes están integrados en microcontrolador. En este caso el microcontrolador viene

instalado en una placa de demostración que provee la opción de conectar periféricos mediante pines o entradas especificas para un periférico en concreto.

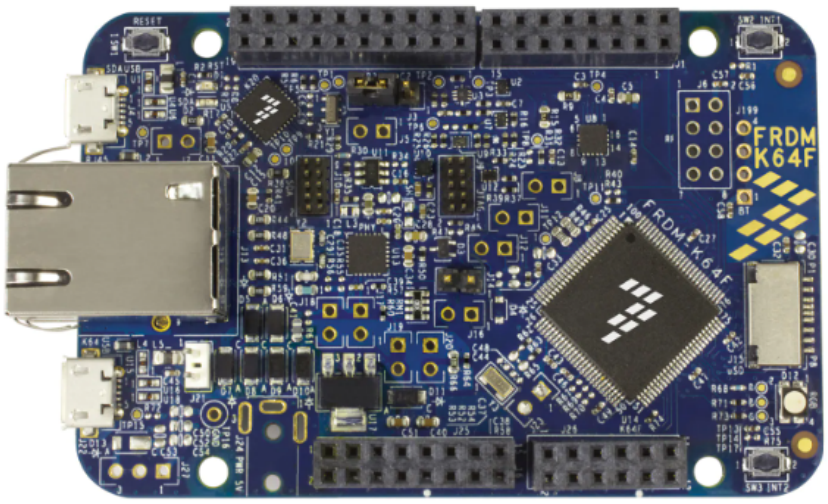

Figura 3.1: Placa FRDM-K64F

En el caso de los sistemas embebidos cada uno de ellos se construye según el propósito especifico que se va a realizar con ello, es decir, dependiendo de su objetivo tendrá unas características hardware u otras. Sin embargo, si 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. Ademas de este componente, necesitaríamos de mas periféricos para obtener algunas funcionalidades mas especificas, algunos de estos periféricos son:

Puertos de comunicación

Sensores

spositivos de interfaz humana

Actuadores

versores ADC y DAC

Ultrasonidos

rtos de comunicación

Software

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

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.

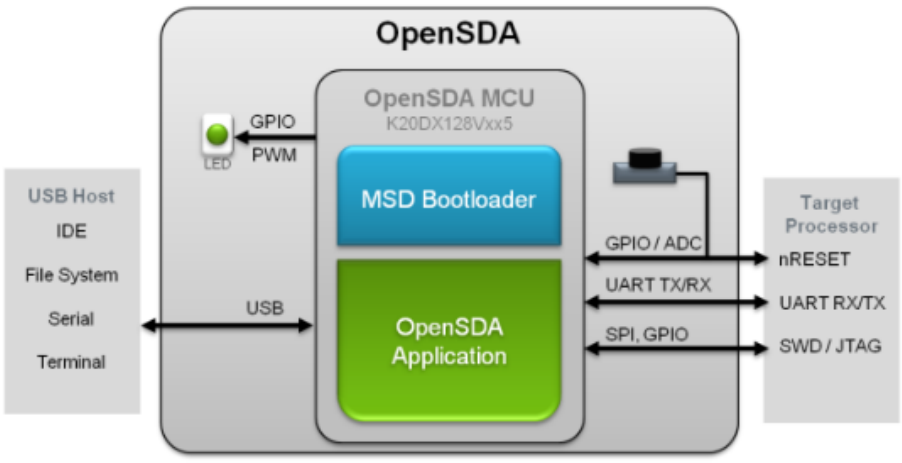

Figura 3.2: Diagrama de Bloques de OpenSDA

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

SO en tiempo real: Es un sistema operativo que se utiliza para facilitar la gestión de multitareas y el uso de tareas en dispositivos con recursos y tiempos limitados. Ademas 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 podríamos referirnos como procesos o hilos. Estos hilos se ejecutan de manera independiente, es decir, 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 esta siendo utilizada.

En Ejecución: La tarea se esta llevando a cabo.

Programador: Con programador nos referimos a la persona que desarrolla el funcionamiento del sistema operativo en tiempo real. Existen distintas técnicas para la programacion 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.

No expropiativo: En este caso solo se interrumpe una tarea si esta aborta.

En el caso de este proyecto FreeRTOS utiliza la programacion Preventiva permitiendo así cumplir con las normas de tiempo real. Por otro lado esto tiene una mayor carga en el microcontrolador ya que tiene que gestionar el cambio de tarea.

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 mas 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.

Middelware: 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.

Drivers: Los *drivers* o controladores de dispositivos proveen las instrucciones necesarias para que un dispositivo externo, un periférico, pueda ser controlado por el sistema operativo que el sistema embebido utilice.

3.2. Tecnologías de Comunicación para SE

Existen varias formas de comunicarse con los SE, tanto con componentes externos como entre dos o mas micro controladores. Por lo general, se pueden conectar los SE a periféricos que lo doten de conexiones *bluetooth*, infrarrojos o wifi, entre otros. De esta manera conseguimos que podamos enviar y recibir información de otros sistemas. Por otro lado tenemos otro tipo de comunicaciones, en este caso para la comunicación con otros elementos, los periféricos. Según la funcionalidad que estemos usando se elegirán una tecnología acorde para el intercambio de datos. En los próximos dos apartados hablaremos sobre este tipo de tecnologías de comunicación y sus utilidades especificas.

Comunicación de Red

En este apartado vamos a hablar sobre las conexiones de red. Por un lado, trataré la conexión vía ethernet que es la que se ha usado en el resultado final del proyecto. Por otro lado, también tocaré la conexión vía wifi ya que durante el desarrollo de este trabajo también se trato de utilizar este modo. Aunque, finalmente se acabo por elegir solo la conexión ethernet, se realizó bastante investigación sobre este tipo de conexión y como implementarla y puesto que si se consiguió transmitir datos mediante comandos 'AT' vía wifi, voy a exponer toda esta información en esta sección.

TCP/IP

Podriamos definir el modelo TCP/IP como un conjunto de normas que hace que varios equipos puedan comunicarse adecuadamente. Seria algo así como las normas sintácticas que seguimos los humanos para hablar y entendernos los unos a los otros. Este modelo es utilizado en Internet y se divide en cuatro grandes capas o niveles.

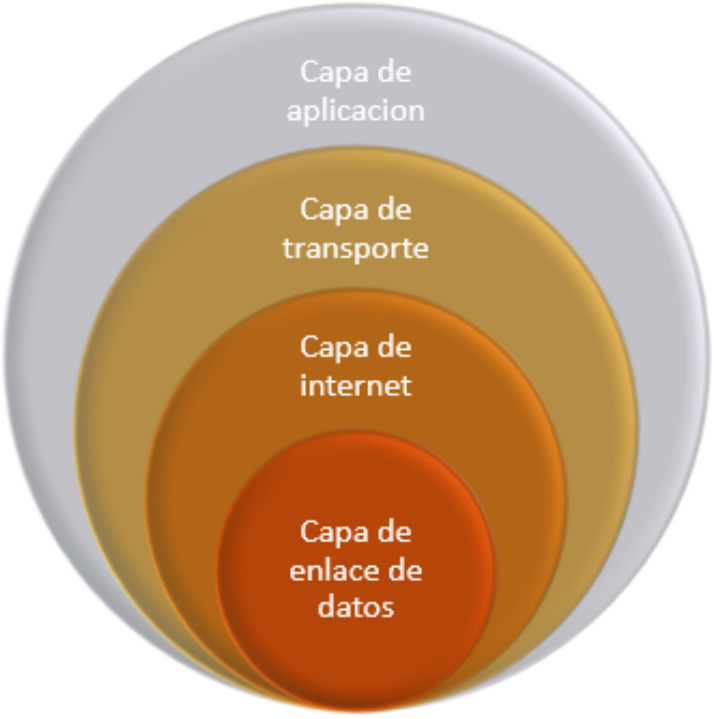

Figura 3.3: Capas del modelo TCP/IP

Capa de enlace de datos Esta capa debe manejar las partes físicas del envío y recepción de datos. Esta comunicación se puede llevar a cabo mediante el cable Ethernet o de manera inalámbrica, la tarjeta de interfaz de red, el controlador del dispositivo en el equipo, etc.

Capa de Internet //MEJORAR//////// La capa de Internet (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. Esta capa utiliza las versiones IPV4 e IPV6 para el control y notificación de errores, existe ademas, IGMP (Internet Group Management Protocol) y MLD (Multicast Listener Discovery) que se usan para establecer grupos de difusión múltiple.

Capa de transporte Esta capa se asegura de conseguir que la conexión de datos sea fiable entre dos dispositivos. Para ello envia 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 seguir 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. Las transmisiones usan una ventana deslizante que permite detectar aquellas no reconocidas para que sean retransmitidas.

UDP A diferencia de TCP, UDP no utiliza ningun mecanismo de establecimiento de la conexion, no se realizan retransmisiones, por lo tanto, determinadas transmisiones pueden llegar a perderse. Su uso se justifica en aquellos escenarios donde la velocidad de la transmisión prima por encima de todo aunque se incurra ocasionalmente en la perdida de información.

Capa de aplicación Esta capa nos ofrece la posibilidad de acceder a otras capas para usar sus servicios. Además a esta capa, pertenecen los protocolos como POP y SMTP que se utilizan para la comunicación, por ejemplo el correo electrónico. Otro protocolo incluido en esta capa seria HTTP, que define la sintaxis y la semántica que se utiliza en la arquitectura web (clientes, servidores *proxies*) para comunicarse. En este caso no es el usuario quien interactua directamente con esta capa sino que interactua con programas que a su vez lo hacen con la capa de aplicación.

Lwip

La libreria lwIP 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 Informática Ciencias (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.

Características

- 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 (Autoconfiguración 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)

Wifi: módulo ESP8266

Segun la informacion mostrada en [**moduloEsp8266**] el modulo ESP8266 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.

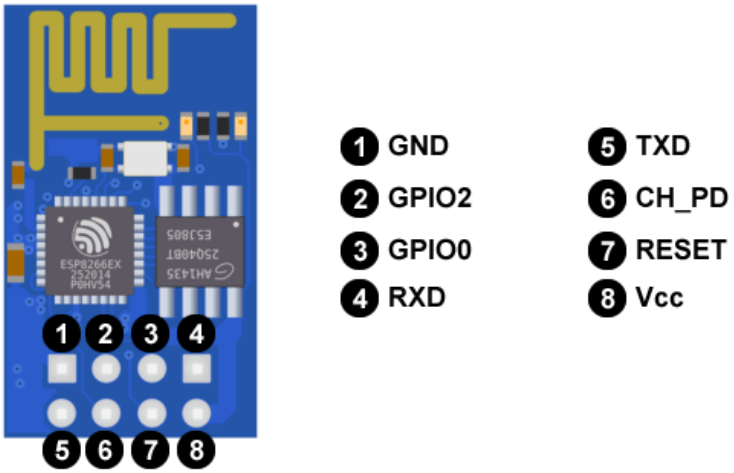

Figura 3.4: Pinnes del modulo wifi 8266

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 200m en campo abierto.

Pero también tiene sus limitaciones, las cuales son:

- 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 250 bytes como mucho.

3.3. FreeRTOS

Algunos apartados mas 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 es robusto, tiene un tamaño reducido y prácticamente una compatibilidad del 100 %, por todo ello es el sistema operativo en tiempo real mas utilizado mundialmente en microcontroladores pequeños. Ademas cuenta con varias demostraciones preconfiguradas y detalladas referentes al Internet de las cosas (IoT). Esta mantenido por AWS en beneficio de la comunidad de FreeRTOS. Incluye, también, un soporte a largo plazo y cuenta con una comunidad que promueve la creación de nuevas librerías aumentando cada cierto tiempo las funcionalidades de este software. Como hemos podido observar es un software completísimo, referente mundial, fácil de usar y con un soporte que saca actualizaciones constantemente.

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, ademas de vía internet, que hemos utilizado en este trabajo.

UART

Uart, 'Universal Asynchronous Receiver-Transmitter' o en castellano Receptor-transmisor asíncrono universal. Su funcionamiento es sencillo, las conexiones son cruzadas entre los dos dispositivos, es decir el pin de transmisión (TX) del transmisor estará conectado al pin receptor (RX) del dispositivo que recibe la comunicación y viceversa. Esta comunicacion es asincrona por lo que no utiliza relojes para el envio y recepcion de mensajes. Para que ambos dispositivos sepan cuando tienen que empezar y dejar leer bits se añaden a cada envio 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 asi la comunicacion fallaria ya que un dispositivo enviaria bits a una velocidad diferente de la que se leen ocasionando malas lecturas. La velocidad predeterminada suele ser de 115200 baudios.

Por otro lado tenemos la comunicación usart 'Universal Synchronous and Asynchronous Receiver-Transmitter' que puede realizar procesos de comunicación con relojes.

Caracteristicas	Usart	Uart
Modo	Semidúplex	Duplex
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 UART vs USART

El principal motivo de hacerlo con Uart en vez de con Usart, es que los comandos de los motores se envían en bytes y no en bloques, por lo que, como hemos visto en la tabla, es necesario hacerlo por uart.

I2C

El protocolo I2C, Ïnter-Integrated Circuitüsa dos lineas para comunicarse con otros dispositivos. Por un lado tenemos SCL (línea de reloj en serie) y por otro lado tenemos SDA (línea de datos), esta ultima es la encargada de transmitir los datos. Este tipo de comunicacion suele estar destinado al intercambio de datos con modulos o sensores y se usa en arquitecturas maestro-esclavo:

- Maestro: Dispositivo que proporciona un reloj para la comunicación
- Esclavo: Dispositivo que utiliza el reloj del maestro

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

Potenciometro y sensor de Temperatura

El potenciometro 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, que pueden cambiar de valor y una tercera resistencia con la cual podemos interactuar. Esta tercera resistencia nos permite aumentar o disminuir su resistencia propiamente dicha. De esta forma conseguimos poder variar el valor entre las conexiones.

Un sensor de temperatura o termómetro es un dispositivo que transforma los cambios de temperatura (magnitud física) en señales eléctricas(voltaje). Por lo general esta formado por un sensor encapsulado en una cubierta protectora. Entre el sensor y la capsula 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 [**TempSensTipos**]:

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.

Termoresistencias 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 capitulo 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 para el desarrollo y organización del proyecto elegí la metodología SCRUM ya que ademas de ser la mas 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 Sprints y tareas. El desarrollo de todo el proyecto se separa en diferentes Sprints, que contienen una serie de tareas necesarias para conseguir un software final. Cada uno de los Sprints debe ser completado en unas 3 semanas por lo que en la reunión inicial se deben discutir que 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 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.

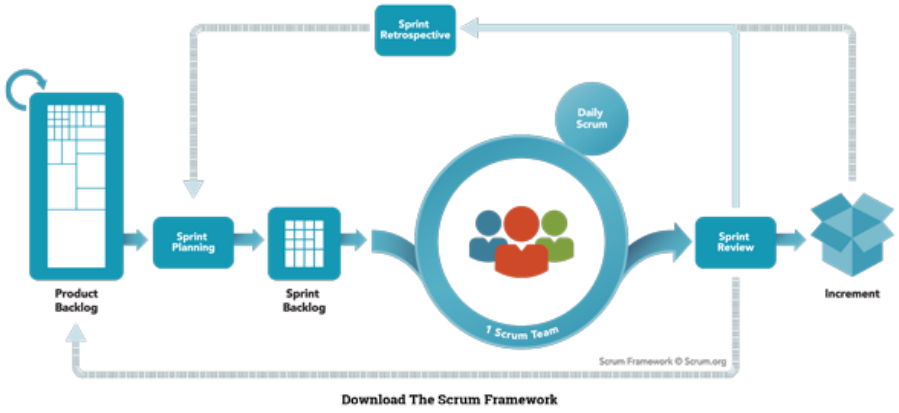

Figura 4.1: Diagrama resumen de la Metodología SCRUM

Aún nos queda por hablar sobre las figuras mas importantes de esta metodología:

El Product Owner Es el encargado de hablar con el cliente y exponer sus peticiones al scrum manager y al equipo de desarrollo. En otras palabras es quien define las tareas, condiciones y prioridades del proyecto centrándose en el 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 esta muy ligada al product owner ayudandole 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 mi, a diferencia de la figura de Scrum Master la habrían representado mis cotutores.

Todo esto nos ayuda a poder lograr un resultado final optimo en el desarrollo de proyectos complejos donde 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 mas detalle los componentes hardware de este proyecto.

Placa FRDM K64F

Esta placa es un conjunto hardware que, ademas de contener el microcontrolador, nos ofrece las características necesarias para poder conectar infinidad de periféricos a este controlador. Veamos las caracteristicas de la placa FRDM K64F:

4FN1M0VLL12 MCU (120 MHz, 1 MB de memoria flash, 256 KB RAM, USB de bajo consumo, sin cristal, y 100 Low profile Quad Flat (LQFP))

- 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

Además de todo esto también se le puede añadir placas de expansión. Como veremos en el siguiente apartado nosotros hemos utilizado la ARDUINO BASIC I/O.

Placa de expansion Arduino Basic I/O

Esta placa se sitúa sobre la placa K64F dotándola de varias características, como por ejemplo: altavoz, 4 leds de diferentes colores, 4 botones, sensor de infrarrojos, 2 potenciometros, entre otros. De esta manera se consigue dotar a la placa de mas funciones.

LCD

Tanto para el desarrollador como sobretodo para el cliente final que recibe el proyecto es interesante el uso de una pantalla que muestre información sobre las operaciones que esta realizando el sistema empotrado. En este proyecto se ha utilizado una pantalla de 2 lineas de 16 caracteres cada linea. Por lo general estas pantallas necesitan de la conexión de mas de 15 pines para la transmisión de datos, la alimentación, la iluminación y el control de la transmisión. Sin embargo en este caso esta pantalla incorpora un modulo que deja la conexión en tan solo 4 pines que serian:

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 12v totalmente equipado con codificadores y un reductor 30:1. 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 una placa que sera la encargada de controlar los bytes recibidos y enviados.

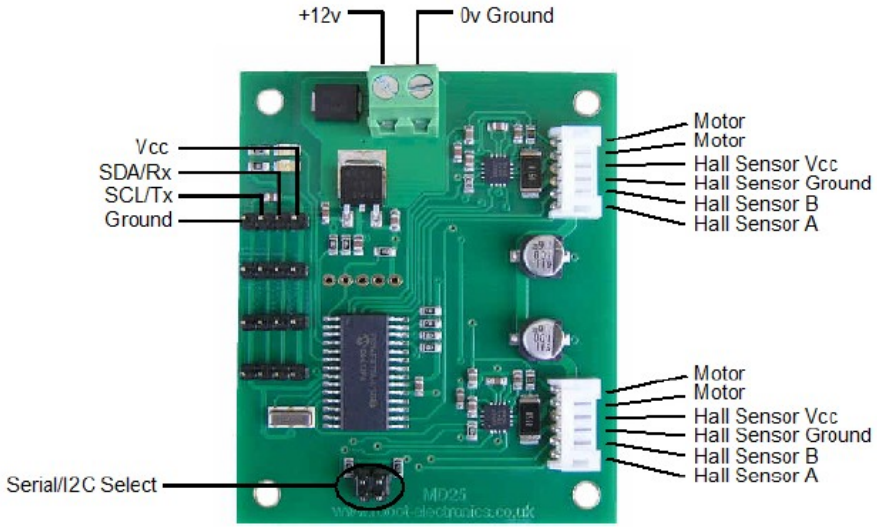

Figura 4.2: Placa controladora de los motores

La placa K64F envía a través de conexión UART 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. La placa de los motores puede recibir y transmitir mediante comunicación UART o I2C. Por otro lado los bytes

enviados consisten, en la mayoría de instrucciones, en un byte de sincronizacion, un byte para elegir el motor puesto que hay dos motores, y 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 velocidad1 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 cuanto a los comandos voy a explicar los mas interesantes:

CMD	Nombre	Bytes Send-Recv	Descripción
0x21	Get Speed 1	2 - 1	Devuelve la velocidad del motor 1
0x22	Get Speed 1	2 - 1	Devuelve la velocidad del motor 2
0x2A	Get Acceleration	2 - 1	Devuelve la aceleración
0x31	Set Speed 1	3 - 0	Fija la velocidad del motor 1
0x32	Set Speed 1	3 - 0	Fija la velocidad del motor 2
0x33	Set Acceleration	3 - 0	Fija la aceleración
0x34	Set Mode	3 - 0	Fija el modo
0x38	Disable TimeOut	2 - 0	No apagarse tras 2s sin comunicación
0x39	Enable TimeOut	2 - 0	Apagarse tras 2s sin comunicación

Tabla 4.1: Comandos Motores MD25

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. 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 esta basado en eclipse y sobre el que se ha desarrollado el proyecto. Este software es especifico para la programacion de MCU NXP basados en nucleos ARM Cortex -M que, recordemos, son los que lleva la placa FRDM K64F. El IDE de MCUXpresso ofrece funciones avanzadas de edición, compilación y depuración con la adición de vistas de depuración específicas de MCU, rastreo y creación de perfiles de código, depuración multinúcleo y herramientas de configuración integradas.

Dockligth Docklight es una herramienta de prueba, análisis y simulación para 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 mas 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, ademas de el numero de bits, paridad, etc. Por otro lado este software cuenta con algunas características añadidas como poder guardar comandos para utilizarlos de una forma mas 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 interacción con el usuario mas simple. En este caso el programa se configura 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 por la placa. 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.

Otras Herramientas

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

lwIP Implementación liviana de la pila de protocolos TCP/IP. Permite a la placa utilizar la mayoría de los protocolos habituales: IPv4, IPv6, TCP, UDP, ICMP, IGMP... Estas herramientas se han utilizado por su conocimiento previo más que por un exhaustivo examen de todas las alternativas existentes.

4.4. Herramientas de Documentación

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

Texmaker Es una herramienta gratuita que nos ayuda a escribir documentos de texto integrando las funciones necesarias para poder realizar documentos con Latex. Ademas este software es multiplataforma.

Latex 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 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 realizacion de documentos con LaTeX.

4.5. Herramientas de Comunicación

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

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

Microsoft Teams 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.

4.6. Herramientas de Gestión de Proyectos

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

GitHub GitHub es una plataforma pensada para que los desarrolladores puedan alojar su repositorios de codigo de forma segura en la nube. Ademas incluye un sistema de control de versiones conocido como Git. Por otro lado tambien permite el desarrollo colaborativo entre distintos desarrolladores y ofrece todas las herramientas para poder trabajar con SCRUM.

GitKraken GitKraken es una aplicación que nos permite manejar Git y por tanto nuestros archivos de GitHub de forma mas 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 capitulo.

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 practicas realizadas con ese programa de cara a famirializarme con ello. Tras la realización de las practicas enseguida me di cuenta de que estaba bastante anticuado y que las nuevas versiones de *drivers* y *middelware* no funcionaban en él, por lo que decidí migrar hacia MCUXpresso. Este cambio hizo que tuviera que volver a famirializarme con el nuevo IDE y la manera de programar el hardware. Aunque KDS estuviera anticuado en la parte de programacion hardware, es decir en dotar a los pines de la placa de una funcionalidad y de sus respectivos relojes para el correcto funcionamiento, algo que MCUXpresso no tiene como tal y dificulta su programacion. A favor de la version nueva diré que incluye decenas de programas de ejemplo con los que poder çacharrearz comprender su funcionamiento, ademas de un montón de *drivers* que habilitan la instalación de un gran numero de sensores. También se puede observar en MCUXpresso como la interfaz es mas limpia y simple a la hora de interactuar con ella. Una vez comprendes como funciona la programacion 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 funcionamiento y periféricos que permiten utilizar a la FRDM-K64F. Ademas, utilizar Arduino hubiera sido mas 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 mas 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 un laboratorio. 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, para poder aprender como funciona el microcontrolador y sus extensiones mediante los pines de la placa. Se ha pretendido desde el principio 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 necesitara que el SE cumpla con unos requisitos específicos y no exceda de ellos para disminuir el costo y tamaño del sistema. En cada proyecto se desarrollan unas placas especificas para esos requisitos y es conveniente saber su funcionamiento interno para poder diseñarlas correctamente.

5.3. Ethernet vs Wifi

Como ya vimos en el apartado comunicaciones por red, existen dos formas de comunicarse, una es por cable de red ethernet y la otra vía wifi utilizando una antena wifi. En este punto veremos las características para cada uno para poder entender por qué nos decidimos por hacerlo por cable en el trabajo. Pese a que las redes wifi se han convertido en la forma mas común de conectarnos en nuestros hogares y lugares de ocio no son las mas apropiadas en todos los casos. Este tipo de conexión no es tan segura, ni estable como la conexión por cable. Veamos las ventajas e inconvenientes de cada una.

Usar cable de red ethernet es mucho más rápido que usar una conexión WiFi. Según un estudio realizado por ADSLZone [**AdslWifivsEthernet**], se pierde hasta el 65 % de tu conexión a Internet mediante wifi. Esto se debe

a factores como la distancia, obstáculos o interferencias con otros dispositivos. Por otro lado, si vives en una comunidad de vecinos o estas rodeados de varias conexiones wifi también se puede producir una saturacion de los canales que utilizan estas redes. En el caso del uso de cable no tienes ninguno de los inconvenientes anteriores sin embargo existe una gran desventaja y que resulta bastante mas engorroso tener que llevar un cable de red hasta cada equipo para que pueda conectarse a Internet. Ademas nos obliga a disponer de conexiones *switch* y en algunos casos habrá que configurar estos *switches* según nuestras necesidades. En el caso de este proyecto solo se iban a utilizar 3 placas que ademas se iban a conectar entre ellas por lo que aparentemente el uso de ethernet es mas sencillo.

Por todos estos motivos y puesto que la placa nos otorgaba la posibilidad de utilizar cable de red, decidí usar esta opción.

5.4. RTOS

En el caso de los sistemas operativos en tiempo real encontramos algunas alternativas a FreeRTOS. Las características y conceptos mas importantes de FreeRTOS ya los vimos en el apartado 3.3. 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. El motivo de haber elegido la utilización de FreeRTOS es su presencia en un mayor numero de proyectos que sus competidores. Esto hace que existan comunidades en Internet que nos brindan mayor información y soluciones para su correcta utilización. Ademas 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.

5.5. Metodologías Ágiles: SCRUM

Como alternativas a SCRUM teníamos dos metodologías ágiles muy utilizadas y quizás mas 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 procesoz las que ya se terminaron: "Terminadas". Cada usuario o equipo tiene la posibilidad de aumentar el numero de columnas según les sea de mayor utilidad. D 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 mas estudiada durante el grado. Ademas, 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.

5.6. 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, sobretodo 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 especifico ya que están pensadas para usuarios con ciertos conocimientos en el ámbito de los SE. Al mismo tiempo va dirigido para personas que tampoco son expertos en la materia y no saben diseñar sus propias placas. Esto hace que se cierre mucho el nicho de personas a las que van dirigidos.

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 tenia 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 practicas curriculares y el tiempo libre disminuye complicando tener una estabilidad diaria para realizar este proyecto.

Volviendo al desarrollo, la adaptación del tipo de comunicaciones, sobretodo 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.

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

Trabajos relacionados

Este capitulo 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. Un ejemplo seria un medidor de presión sanguínea, cuyo esquema arquitectónico seria el siguiente:

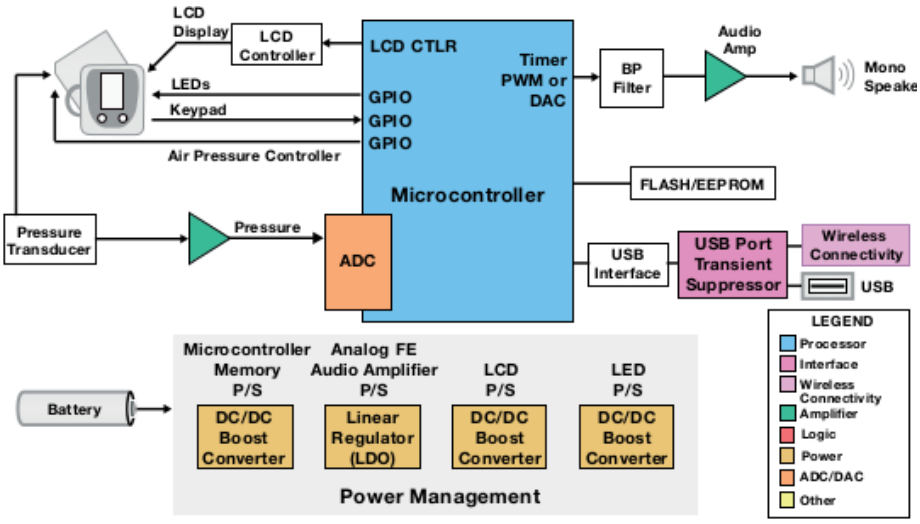

Figura 6.1: Esquema arquitectónico de un SE de medición de sangre

En la imagen podemos apreciar como existen dos partes importantes, por un lado tenemos todos los elementos que necesitan corriente y como se conectan a una fuente de alimentación y por otro lado esta 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 utilizara un valor analógico y lo convertirá en uno digital (ADC) para poder saber la presión sanguínea del cliente. Pero ahí no acaba todo, hay inmensidad de utilidades para este ámbito. Otro ejemplo seria la utilización de un SE para controlar un desfibrilador (DESA). Como todos sabréis 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 micro controlador que mediante un altavoz explica como debes usarlo y aplica de forma cíclica una serie de descargas dependiendo del estado del paciente y con una ligera interacción humana. En este caso el esquema arquitectónico seria 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 ademas 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:

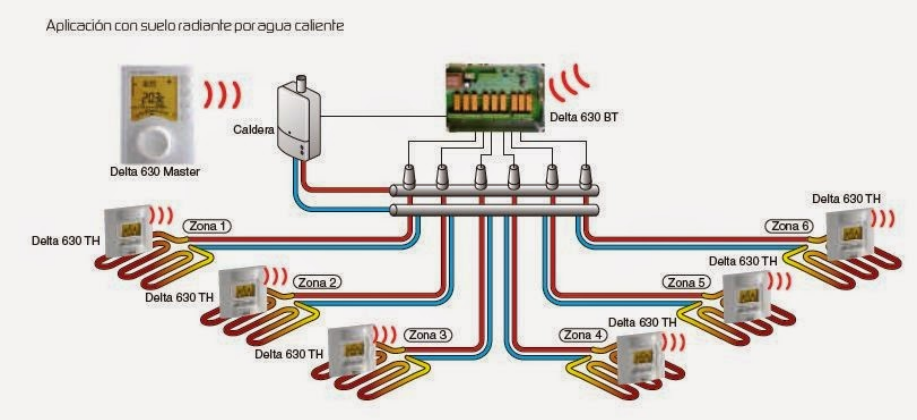

Figura 6.2: Sistema de Calefaccion con SE

Como podemos apreciar en la figura se muestran 2 sistemas embebidos, ademas 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 dejaran 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 este usando en ese momento.

6.3. Conexión desde otros dispositivos

En este apartado me gustaría hacer mención al trabajo de fin de máster que desarrollo un compañero, [**RPC0027**] , de esta misma universidad en el año 2018. Su TFG estaba relacionado con los sistemas empotrados. En este caso el realizo un la conexión a Internet de una sola placa que se controlaba a través de un servidor que el 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, potenciometros, 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.

Ademas de esta manera conseguimos poder comunicarnos con las otras placas desde cualquier sitio y no desde donde este físicamente ese sistema.

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

Conclusiones y Líneas de trabajo futuras

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

7.1. Conclusiones

Mediante la realización de este trabajo he podido demostrarme a mi mismo los conocimientos que he adquirido durante estos cuatro años en la realización de esta carrera. 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 adquirido. Otra de las partes mas 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 los errores. 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 como se realiza un proyecto de este tipo en un ejemplo real y aprender como funciona esta herramienta. Ademas usarla ha permitido que en caso de fallos en la programacion del sistema embebido dispusiera siempre de una copia anterior que me servia como *backUp*.

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

Personalmente ha sido de gran interés el uso de sistemas embebidos en mi proyecto puesto que a medida que avanzaba con el proyecto y comprendía su utilización cada vez surgían mas y mas utilidades que se le pueden dar, tal y como veremos en el próximo apartado, Lineas de trabajo futuras. Ademas su estructura permite ampliar sus funcionalidades fácilmente por lo que cada vez se utilizan mas 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 numero de usos también tienen puntos débiles. Debemos poner de manifiesto que su programacion a bajo nivel es en algunas ocasiones bastante complicado y en muchas ocasiones se necesitan placas específicamente creadas para algunos periféricos concretos, debido a su voltaje numero 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 especifico por no hablar de que cada sistema embebido dispone de una programacion diferente. Ademas 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 serian:

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 linea de futuro seria implementar algún sistema domótica, pudiendo llegar al punto de utilizarlo en tu propia casa. Algunos ejemplos serian, continuando con la idea del sensor de temperatura conectar o bien por infrarrojos o por bluetooth una conexión de la placa al aire acondicionado y que este variara la temperatura automáticamente, incluso la implementacion 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. Ademas se pueden conectar a la nube para subir y descargarse datos de manera que pueden crear informes sobre el funcionamiento de un sensor.

Relaciona 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.