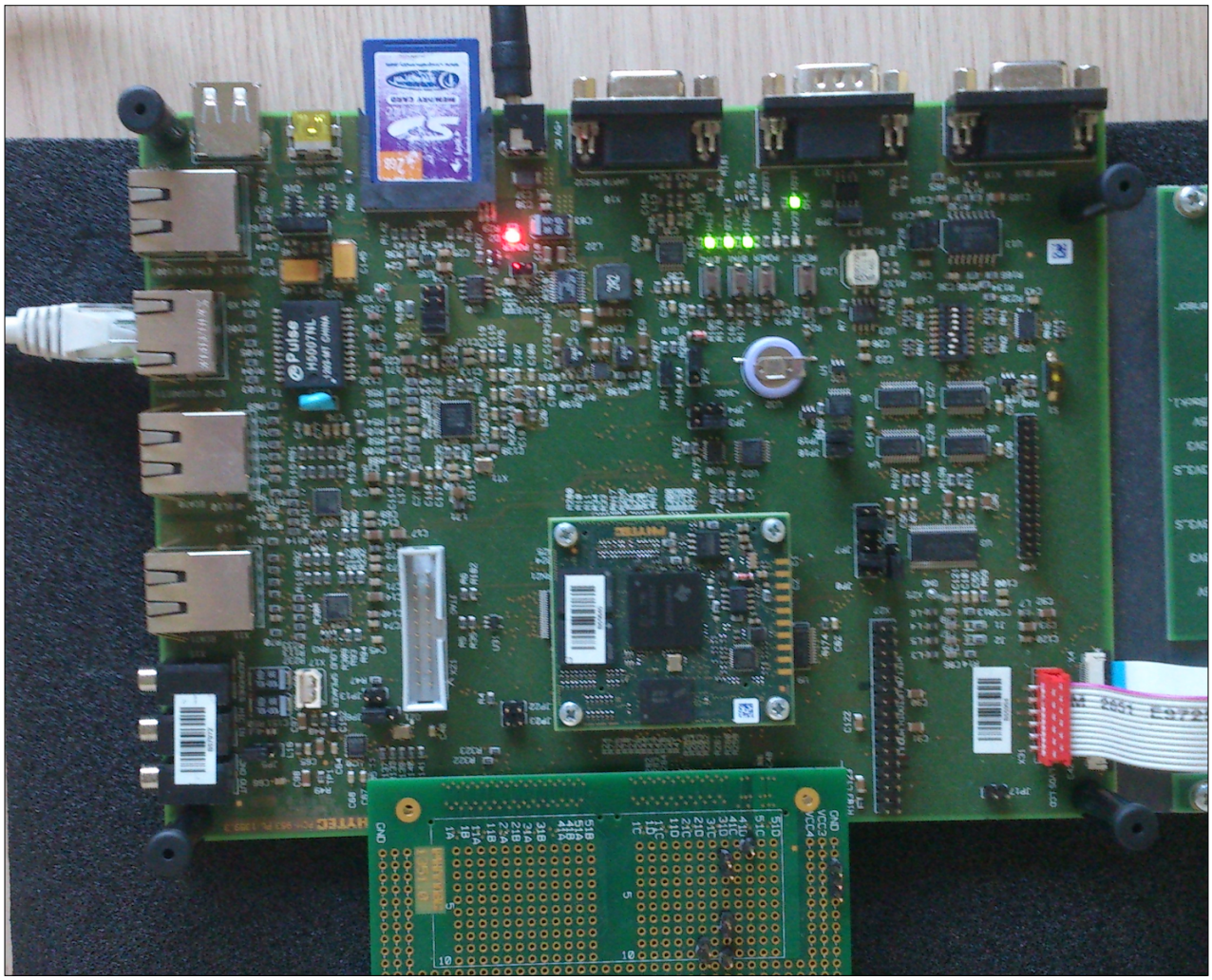
Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

Grado en Ingeniería de Telecomunicación



Gateway para red de sensores

Autor: Koldo Iribarren Azcárate
UPNA - Tutor: Javier Goicoechea Fernández
Kunak - Tutor: Francisco Javier Alonso Caballero
Pamplona, 20 de junio de 2014

Dedicatoria

A Oscar y Javier, por haberme ayudado a llegar hasta aquí.

Agradecimientos

A mis tutores, a mis compañeros de Kunak y a Guillermo,
por su colaboración para con este trabajo.

RESUMEN

El presente trabajo trata sobre el estudio de los sistemas embebidos a nivel de hardware y software, a fin de hallar la solución más adecuada para su utilización como un a pasarela de datos entre una red de sensores comercial y el usuario de dicha red. Para esto, primero se explica de qué se componen dichos sistemas embebidos a nivel físico y las opciones de sistemas operativos más interesantes en la actualidad y en función de esto, se decide cuál será el sistema que se utilizará para la solución. A continuación se muestra qué se necesita para que el sistema funcione y cómo se consigue. Posteriormente, se explica la preparación del dispositivo y las pruebas realizadas de forma práctica y por último se analizan los resultados y se dan las pautas para continuar.

PALABRAS CLAVE

Sistemas, Embebidos, Pasarela, Embedded, Systems, Gateway.

ÍNDICE

1. INTRODUCCIÓN

El presente trabajo trata sobre el estudio y pruebas de diferentes tecnologías tanto software como hardware para el diseño de una pasarela de comunicaciones entre la red de sensores en estrella de la empresa Kunak y el usuario de dicha red, a través de internet.

La conexión entre esta pasarela y el nodo central será cableada entre puertos serie tipo UART, y por ahí se recibirán todos los datos producidos por los sensores. Con dichos datos, la pasarela podrá almacenarlos en una base de datos alojada en su memoria interna, de forma que el usuario acceda a ella directamente, o bien los reenviará a otra base de datos en internet, adonde se conectará el usuario. De cualquiera de las formas, los datos se deberán transmitir por una conexión TCP sobre Ethernet.

Para este cometido, a nivel de hardware se hará uso de una placa de circuitos embebidos, con un procesador de arquitectura ARM similar a los que utilizan los teléfonos móviles. Esto es fundamental, ya que este tipo de procesadores requiere un consumo reducido pero proporciona grandes prestaciones, lo cual permite que el dispositivo pueda funcionar en cualquier lugar, simplemente con una batería y un panel solar que la recargue, durante varios años sin que lo tenga que manipular nadie.

Por otro lado, el sistema operativo deberá estar basado en un software abierto y que permita flexibilidad para poder adecuarlo a la aplicación que sea necesario, e incluso, permitir la posibilidad de utilizar la pasarela en otro tipo de redes a parte de la contemplada en este proyecto sin tener que cambiar el sistema operativo ni las herramientas para construirlo. Además, otro requisito importante a nivel de software es que la placa debe ser capaz de alcanzar su estado normal de trabajo automáticamente en su primera conexión y de recuperarlo en caso de que el sistema se reinicie debido a algún fallo.

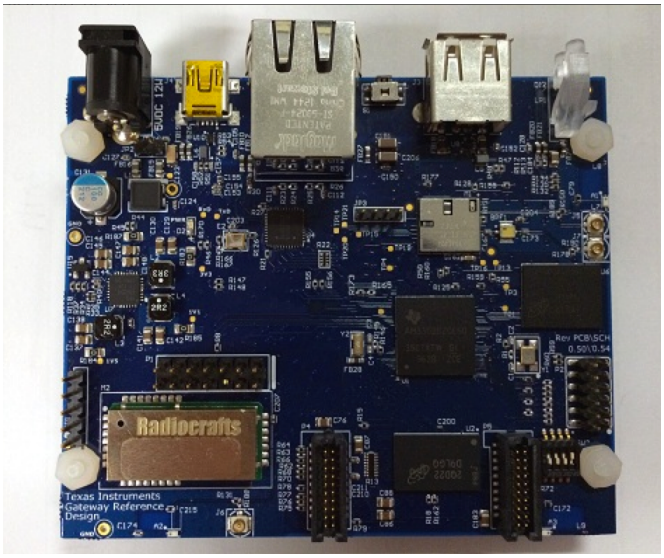


Fig. 1.1 Concentrador para smartgrid - Texas Instruments

1.1 Objetivos

El objetivo es encontrar la solución embebida (HW & SW) más adecuada, para el producto de Kunak, de entre las diversas estudiadas, para lo cual, se contemplarán las siguientes características:

- Técnicas

1. Cumplimiento de las funciones requeridas por el producto: recepción de datos de la red, almacenamiento y/o transmisión de dichos datos, versatilidad…
2. Prestaciones: consumo, velocidades de ejecución, tiempos de respuesta…
3. Fiabilidad, robustez frente a fallos y recuperación ante caídas.

- Comerciales

1. Sencillez de reproducibilidad, para que el proceso de fabricación sea lo más simple posible.
2. Sencillez de programación, a fin de que el proceso de modificación del software no de muchos problemas. Existencia de diferentes herramientas de desarrollo, tanto abiertas como de pago.
3. Costes, de materiales y de manufactura. Existencia de diversos fabricantes, ampliamente usada.

Aparte del Gateway, interesa que la plataforma elegida sirva a para más proyectos embebidos de bajo consumo.

1.2 Contexto

Kunak es una startup ubicada en Noain (Navarra) que se dedica al desarrollo de dispositivos electrónicos de medición, basados en tecnologías inalámbricas, comunicaciones máquina a máquina y el internet de las cosas, que ofrecen un mecanismo de monitorización remota para una gran variedad de escenarios posibles, según las necesidades del usuario.

Así pues, su producto estrella es el sistema integral de Kunak, consistente en una red de sensores inalámbricos para diversas aplicaciones, los cuales son gestionados por una plataforma residente en la nube, que además recibe los datos producidos por ellos y se los presenta al usuario, permitiéndole así la monitorización y el control de cualquier situación y evento.

Fig. 1.2 Esquema de el sistema de Kunak

Uno de los aspectos más importantes del producto es su versatilidad, pues los sensores pueden ser de muchos tipos, dependiendo de la aplicación y su necesidad, que van desde los típicos termómetros, higrómetros y anemómetros para meteorología, a medidores de vibraciones para maquinaria industrial, pasando por sensores de concentración de gases como CO2 para estudios medioambientales o de perniciosidad de un ambiente de trabajo.

Además, estos sensores están pensados para realizar medidas periódicamente, con mayor o menor frecuencia según lo requiera el usuario, de forma que se obtiene una colección de datos significativa, capaz de mostrar la evolución del sistema a lo largo del tiempo, dando así una perspectiva mejor al usuario de qué es lo que está pasando.

Por otro lado, los sensores tienen una disposición en estrella, y la comunicación entre ellos es a través de radioenlaces de larga distancia y bajo consumo y del nodo central con la plataforma en la red mediante internet móvil. El bajo consumo es un aspecto muy importante ya que los sensores pueden estar colocados en puntos de difícil acceso y a mucha distancia unos de otros, por lo que tienen que funcionar con pequeñas baterías y células solares, y con un formato compacto.

Por último, la plataforma en la nube es la encargada del control de los sensores y de la recepción de los datos y su almacenamiento y presentación. Esta plataforma guarda las medidas en una base de datos y también trabaja con ellas, a fin de poder realizar un análisis previo automático y enviar alarmas al usuario en caso de que se sobrepase algún nivel crítico o se produzca alguna incidencia en los sensores.

La presentación final de los datos al usuario puede ser de dos formas, desde un ordenador a través de la página web que soporta la plataforma, o mediante una aplicación para smartphones y tablets, ambas con un funcionamiento similar, se presentan los datos en forma de gráficas, hay un menú de alarmas, y ofrecen la capacidad de gestionar el funcionamiento de la red, como la frecuencia de medición, los niveles de alarma, etc.

Como ya se ha dicho, el sistema está completo y puede ser utilizado directamente así, sin embargo, en algunas situaciones y aplicaciones puede no ser una solución óptima. Por ejemplo, si el sistema se utiliza en un sitio cercado, como una fábrica, puede resultar más conveniente que los datos no salgan del recinto, sino que se conserven en un servidor privado interno, propiedad de la empresa, por motivos de sencillez y seguridad en el control de los datos.

Fig. 1.3 Esquema del sistema con gateway

Así pues, se hace necesaria la presencia de un elemento que sirva de interfaz entre la red de sensores y el sistema informático del usuario, así como una base de datos no dependiente de agentes externos al usuario del producto.

Para esta tarea se ha decidido utilizar una tarjeta programable de sistemas embebidos, capaz de recibir los datos del nodo central de la red de sensores a través de un puerto UART, almacenarlos en una base de datos interna, y retransmitirlos por un puerto ethernet.

Para poder funcionar adecuadamente, la tarjeta deberá ser capaz de ejecutar programas en Java y Python, y tener instalados Apache, Tomcat y MySQL, para poder trabajar como servidor. Además, siguiendo con la filosofía del resto del sistema, se buscará que la placa tenga un consumo lo más bajo posible y capacidad de adaptación a posibles nuevas necesidades y tecnologías que puedan surgir en el futuro.

2. SISTEMAS EMBEBIDOS

2.1 Introducción

2.1.1 Definición

Los sistemas embebidos son unidades de computación destinadas al desarrollo de una o varias tareas concretas y de forma dedicada, a diferencia de un ordenador de propósito general, que puede utilizarse para diversas tareas que pueden ir cambiando.

Generalmente los sistemas embebidos se utilizan para aplicaciones en tiempo real, es decir, que el procesado de las señales de entrada se produce en cuanto se dan estas, e inmediatamente se produce la respuesta, y esto se consigue teniendo el procesador en escucha constante a la espera de eventos.

Además, estos sistemas suelen estar físicamente formados por una sola tarjeta, en la que se encuentran la unidad de procesado central (CPU) y todos los periféricos y elementos necesarios. Debido a su especificidad, los sistemas embebidos sufren importantes limitaciones de espacio, ya que suelen estar integrados en el interior de los dispositivos que controlan, por ejemplo, en una lavadora, la placa tiene que caber dentro de la carcasa junto con el tambor y los motores, dentro de unos márgenes de tamaño estandarizados.

Por último, los sistemas embebidos suelen estar bastante limitados en memoria, tanto RAM como ROM, especialmente cuanto más dedicados y sencillos son sus procesadores, por lo tanto, su codificación ha de ser lo más ligera posible, o si no, no funcionará adecuadamente.

2.1.2 Utilización

Originalmente, en los años 80, los sistemas embebidos estaban desarrollados en torno a procesadores pequeños y se centraban en realizar muy pocas acciones y muy sencillas, las cuales se programaban directamente en el microcontrolador. Actualmente, sin embargo, la capacidad de los micros ha aumentado enormemente, por lo que los sistemas embebidos pueden contener hasta sistemas operativos similares a los de los ordenadores domésticos, basados en sistemas de ficheros complejos, de forma que, si bien suelen estar más limitados, la línea que marca la diferencia entre ellos se ha difuminado bastante.

De esta forma, al igual que hay gran variedad de sistemas embebidos hay multitud de aplicaciones en las que se utilizan, desde las más sencillas como por ejemplo el control de un ascensor, a otras mucho más complejas, como una plataforma para alojar un servidor de internet. Es por esta versatilidad y funcionalidad que hoy en día se pueden encontrar sistemas embebidos en una gran cantidad de dispositivos de muy variados ambientes, como termostatos, lavadoras o teléfonos en el entorno doméstico, o robots, vehículos o incluso satélites en el medio industrial.

En aquellas aplicaciones en las que el tiempo de respuesta en ejecución resulta muy crítico o que la aplicación sea muy simple y concreta, se utilizan sistemas con procesadores más sencillos, con los programas escritos directamente a muy bajo nivel, en ensamblador, de forma que se consigue optimizar todo, tanto tiempos de respuesta y ejecución como uso de memoria y consumo, aunque todo ello resulte más complicado de lograr.

Si los tiempos de respuesta son menos importantes, o si el dispositivo puede requerir cambios en su aplicación con el tiempo, los sistemas empleados son más complejos, más evolucionados que los anteriores, y se configuran a un nivel mayor, con un sistema operativo el cual se encarga de la conversión a bajo nivel y el acceso a los drivers, aunque esta versatilidad se paga con un mayor consumo, menos agilidad en la respuesta y mayor ocupación de memoria por parte del sistema.

Sin embargo, en la actualidad, con algunos sistemas embebidos de los de más alto nivel, como por ejemplo los que se basan en Linux embebido en procesadores de bajo consumo, se pueden conseguir buenos tiempos de respuesta, combinados con gran versatilidad y eficiencia, lo cual los hace idóneos para casi cualquier tipo de aplicación automatizada.

2.2 Hardware

El término hardware hace referencia a toda la parte física del dispositivo. Si bien esto incluye la parte de soporte, como la caja de plástico que pueda contener un router, para esta explicación sólo se entiende como hardware los elementos necesarios para el funcionamiento de un sistema embebido, es decir, el procesador, los periféricos y demás componentes electrónicos y la tarjeta con las pistas electrónicas donde éstos se colocan.

Como se ha dicho, los sistemas embebidos están formados por una única placa donde se encuentran todos los elementos juntos, sin embargo, dependiendo de la complejidad de algunos modelos, los sistemas pueden dividirse en una pequeña placa con la CPU, también llamada system on chip (SoC), que contiene todos los elementos de computación, y otra placa más grande que contiene los periféricos y a la que se acopla la anterior. Esto sirve para poder producir diferentes sistemas embebidos con mayor o menor funcionalidad, ajustándose mejor a las necesidades del usuario, pero sin tener que cambiar el diseño del núcleo.

2.2.1 Procesador

El procesador o CPU es la parte central y más importante del sistema embebido, ya que es la encargada de interpretar las instrucciones de los programas y procesar los datos para hacer que todo funcione adecuadamente. Originalmente, los procesadores eran muy grandes y formaban parte de superordenadores, pero en la actualidad son muy pequeños (microprocesadores) y se encuentran en prácticamente todos los aparatos electrónicos.

En su estructura más extendida, la conocida como arquitectura Harvard, el procesador se conecta por separado dos bloques de memoria, uno que contiene las instrucciones del programa y otro con los datos que se han de utilizar. Por otro lado se comunica con los terminales de entrada y salida.

Internamente, el procesador tiene un conjunto de registros que se emplean por un lado para llevar el contador de programa y por otro para interpretar las instrucciones y trabajar con los datos de la memoria, una unidad de control que dirige las operaciones y una unidad aritmético-lógica (ALU), que opera con los registros de bits.

A través de los sistemas de entrada y salida se comunica con los periféricos y los controla, recibe los datos que le transmiten, los interpreta y procesa y les devuelve la respuesta que proceda.

ARM

La familia de procesadores ARM (Advanced RISC [Reduced instruction set computer] Machine) es muy variada pues la empresa que los diseña, ARM Holdings, no es la que los fabrica, sino que vende las licencias a otros fabricantes para que éstos puedan hacer sus propios procesadores basados en la arquitectura ARM. Esto supone que hay una oferta muy diversa con algunas diferencias de producto entre fabricantes, como la alimentación del chip y los periféricos soportados, si bien todas las versiones comparten el mismo juego de instrucciones, por lo que todas las versiones que soporten alguna misma revisión de ARM son compatibles a nivel de software.

Esto no significa que todos los dispositivos puedan ser programados y preparados de la misma manera, sino que el lenguaje ensamblador y los códigos binarios resultantes son idénticos para los procesadores que cumplan una versión de la arquitectura.

Las versiones de 32 bits en uso actualmente incluyen ARMv4T, empleada por las familias ARM7TDMI (p. ej. en Game Boy Advance) y ARM9TDMI (como algunos navegadores TomTom), ARMv5TE, en las familias ARM9E (como en algunos Sony Ericsson o en la Nintendo DS), ARM10E y Xscale (p.ej. en algunos modelos de BlackBerry), ARMv6, utilizado por la familia ARM11 (p.ej. en varios teléfonos de Nokia y los primeros iPhone y iPod Touch) y parte de Cortex, y ARMv7, contemplado en el resto de la familia Cortex (en muchos dispositivos móviles como iPhone 4 y HTC Desire, consolas como Ouya y PSVita, y sistemas embebidos como BeagleBoard). Por otro lado, hay una versión de 64, la ARMv8-A que se puede encontrar en los Cortex A53 y A57, siendo la primera versión de ARM en soportar esta anchura de palabras de bits.

Como se puede apreciar por los ejemplos anteriores, los procesadores ARM dominan en el mercado de la electrónica móvil e integrada, a causa de su simplicidad, tamaño reducido, bajo consumo y bajo coste. Además de lo anterior, otra característica muy interesante es que estos procesadores soportan gran cantidad de sistemas operativos como Windows Embedded, Symbian, QNX, algunos UNIX/tipo UNIX, como Solaris, iOS, BSD o diversos Linux.

Intel x86

La familia x86 comenzó en 1985 con el i386 de Intel y llega hasta la actualidad incluyendo todos sus descendientes, como el 486, la familia Pentium, NetBurst, Xeon, Core, Core 2 y los Core i3, i5 e i7, junto con los procesadores compatibles de otros vendedores como National Semiconductor y AMD, que fue el primero que popularizó la extensión de 64 bits de los x86 (x86_64). Sin embargo, Intel sigue siento la principal referencia en cuanto a la familia x86 y sigue siendo su principal distribuidor.

Los procesadores x86/x86_64 son los más extendidos en el ámbito de los ordenadores personales de sobremesa y portátiles, por lo que la mayoría de los sistemas operativos tienen versiones que corren en este tipo de arquitectura, como diversas distribuciones de Windows, MacOS X y Linux, e incluso versiones adaptadas de sistemas operativos pensados para móvil, como Android.

Sin embargo, a pesar de esto, los x86 sólo representan una pequeña fracción del mercado de los sistemas embebidos tradicionales, pues en la mayoría de los casos, los diseñadores prefieren otras arquitecturas como ARM, MIPS o PowerPC. Esto se debe a razones de complejidad, ya que x86 es una arquitectura poco limpia por haberse diseñado para ser compatible con las arquitecturas anteriores de Intel, por consumo de potencia, pues el de los x86 es bastante más elevado que los de los otros, aunque esto está cambiando actualmente con los nuevos Core iX y por costes totales.

No obstante, la arquitectura basada en el i386 es la más ampliamente documentada en general, habiendo gran cantidad de libros y documentos en internet, en muchos idiomas, de vendedores y de usuarios, tratando sobre los entresijos de este tipo de procesadores.

Power PC

La arquitectura Power PC (Performance Optimization With Enhanced RISC - Performance Computing) surge como resultado de una colaboración entre Apple, IBM y Motorola (Ahora Freescale), y hereda ideas del trabajo previo de las tres compañías, especialmente la optimización de rendimiento de IBM. Estos procesadores se hicieron famosos porque se usaban en los primeros Macs de Apple, pero esta alianza ya no continúa. Sin embargo, también hay otros dispositivos basados en PowerPC de IBM y otros vendedores, así como muchos sistemas embebidos, como en los rovers de Marte, o en consolas como Wii, PS3 y Xbox 360.

Multitud de distribuciones soportan PowerPC en sus versiones de 32 y 64 bits, algunas exclusivamente, como, por ejemplo, Yellow Dog Linux, que sólo provee Linux para esta arquitectura, otras, dan soporte a PowerPC a parte de a otras arquitecturas, como Debian, Fedora o Ubuntu, pero a menudo, es una arquitectura considerada como secundaria, de modo que el soporte y el mantenimiento lo da la propia comunidad de usuarios.

MIPS

Los procesadores MIPS (Microprocessor without Interlocked Pipeline Stages) se hicieron famosos por su utilización en consolas como la Play Station original o la Nintendo 64. La compañía diseñadora del procesador, MIPS Technologies (Anteriormente MIPS Computer Systems), vende licencias de la arquitectura a los fabricantes, de una forma similar a ARM, para que estos hagan su versión, sin embargo, a diferencia que con los ARM, con MIPS hay gran cantidad de juegos de instrucciones diferentes entre sí. Actualmente ya no se utilizan para estaciones de trabajo, pero sí que se siguen utilizando en servidores, placas de desarrollo y sistemas embebidos.

Entre otras opciones, se pueden encontrar implementaciones de MIPS de 32 bits de fabricantes como por ejemplo IDT, Toshiba, NXP (Antes Philips) y LSI, e implementaciones de 64 bits de IDT, LSI, NEC, NXP, Broadcom y Toshiba, pero el mercado de procesadores MIPS es mucho más amplio y sus aplicaciones principales son en sistemas embebidos, como routers, gateways y videoconsolas, aunque también se han logrado estaciones de trabajo completas basados en MIPS.

Esta arquitectura es soportada por algunas versiones de sistemas operativos portadas especialmente, como por ejemplo IRIX, Windows Embedded, Linux, BSD, QNX y el propio sistema RISC desarrollado por MIPS Computer Systems.

2.2.2 Periféricos

Los periféricos son todos los elementos que colaboran en el desempeño de la aplicación y son controlados y gestionados por el procesador. Los periféricos le envían los datos al procesador el cual los interpreta y con ellos elabora la respuesta que proceda, según cómo esté programado, y éste se la devuelve para que actúen en función de ésta.

La categoría de periféricos es muy amplia e incluye componentes muy variados como por ejemplo, chips de convertidores A/D-D/A o serie-paralelo, sensores y actuadores, pantallas, y otras CPUs. Lo más importante de estos elementos es la forma de comunicarse con el procesador, ya que de esto depende el medio físico y el protocolo empleado. Algunos de los más importantes son los siguientes.

Bus

Los buses son mecanismos de comunicación digital que sirve para transferir datos entre los componentes de un sistema o entre varios sistemas, incluyendo todos los componentes de hardware relacionados, como cable y fibra óptica, y de software, como los protocolos de comunicación, por lo que existen muchos tipos de buses.

Originalmente los buses eran de tipo paralelo, es decir, los datos se transmitían en bloques de varios bits simultáneos, lo cual requería que las comunicaciones se hicieran por medio de cintas o de numerosas pistas en el circuito impreso, de forma que cada conductor tuviera una función fija y la conexión solo requiriera puertos de entrada y salida para cada dispositivo. Sin embargo, en la actualidad los buses en paralelo están menos extendidos, en favor de los buses en serie, que, aunque requieren una lógica más compleja, consiguen velocidades y eficacias mayores.

CAN

El bus CAN (Controller Area Network) es un protocolo de comunicaciones sobre topología bus serie basado en la transmisión de mensajes para entornos distribuidos y que permite la comunicación entre CPUs. Originalmente fue diseñado para su uso en aplicaciones de automóviles, aunque también se emplea en el sector aerospacial, marítimo, automatización industrial y equipamiento médico.

Como fue pensado para su uso en coches, el diseño pretende sustituir al cableado complejo y soporta un control distribuido en tiempo real, es decir, la topología es un bus serie multi-maestro, al que se conectan las unidades de control electrónico (ECU), que son los sensores y actuadores, como el control de estabilidad, o el disparador del airbag.

El sistema permite conectar hasta 70 dispositivos, con comunicaciones de hasta 1 Mbps a través de mensajes, con bajos tiempos de latencia, flexibilidad en la configuración y robustez en consistencia de los datos por detección y corrección de errores. Los nodos pueden emitir y recibir datos pero no a la vez, y en caso de querer transmitir varios a la vez, el orden en el que lo hacen se dirime por prioridad

que, de modo que el medio de acceso es CSMA/BA. Los datagramas contienen un número identificador, asignados en la instalación, de forma que la prioridad del nodo será mayor cuanto menor su identificador, por lo que al coincidir mensajes, el de menor número permanecerá y los demás desaparecerán.

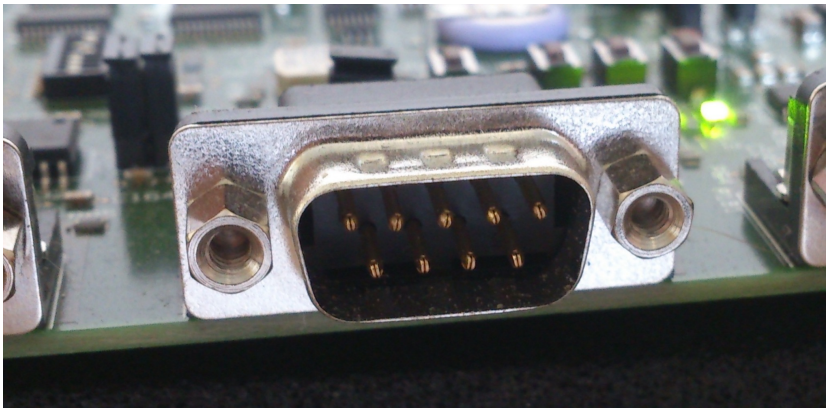


Fig. 2.1 Puerto serie CAN

SPI

El bus SPI (Serial Peripheral Interface) es un estándar de facto de comunicaciones, usado para la conexión de circuitos integrados en equipos electrónicos, como sistemas embebidos y sensores. Es un mecanismo muy extendido pues permite la conexión de periféricos de distintos tipos de una forma común, permitiendo así diseños flexibles y de coste reducido por simplificación de la electrónica.

Los dispositivos se comunican en modo maestro-esclavo, en el que un único maestro se conecta con los esclavos mediante un bus serie de tres líneas común a todos ellos, y una cuarta línea de selección, individual a cada uno. Uno de los cables comunes conecta la salida de reloj del maestro con las entradas de reloj de los esclavos, mientras que los otros dos cables son de transmisión de datos, uno en sentido maestro a esclavo y el otro en el recíproco.

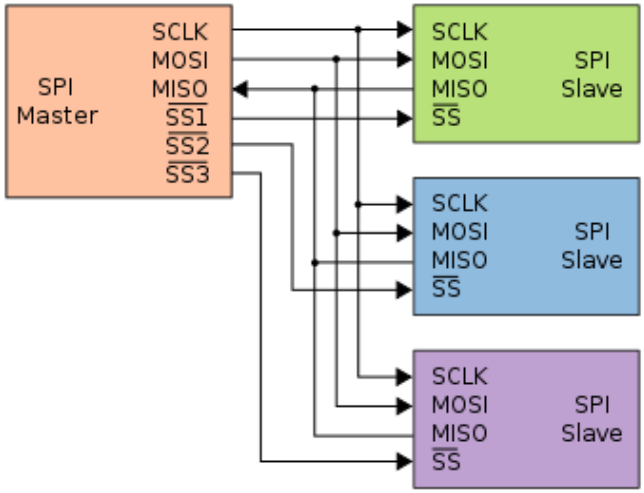


Fig. 2.2 Esquema bus SPI - Wikipedia

El hecho de tener un chip-select permite que el sistema funcione en un modo de comunicación full-duplex síncrono, con la posibilidad de tramas variables y una implementación sencilla, con menos conductores y un consumo de potencia inferior a otros estándares. Por el contrario, el protocolo no contempla una señal de confirmación, por lo que pueden perderse datos sin notarlo el maestro y no soporta conexión en caliente.

I2C

El bus I2C (Inter Integrated Circuits) es un estándar de comunicación en serie muy usado en la industria para conectar dispositivos electrónicos, generalmente dentro del mismo circuito impreso. Puede conseguir velocidades de 10 kbps en modo de baja velocidad, 100 kbps en modo normal y hasta 3.4 Mbps en modo de alta velocidad y puede conectar simultáneamente hasta 128 nodos con direccionamiento de 7 bits y hasta 1024 con direccionamiento de 10 bits.

Los dispositivos conectados al bus funcionan conectados a un sistema de dos líneas, una de datos (SDA) y otra de reloj (SCL), ambas conectadas mediante resistencias de pull-up a otra línea a 3.3 o 5 V, en una configuración de drenador-abierto. Adicionalmente podría haber una tercera línea sirviendo de terminal de referencia conectado a masa, pero como generalmente los elementos forman parte de una misma placa y comparten la masa, esta línea no suele hacer falta.

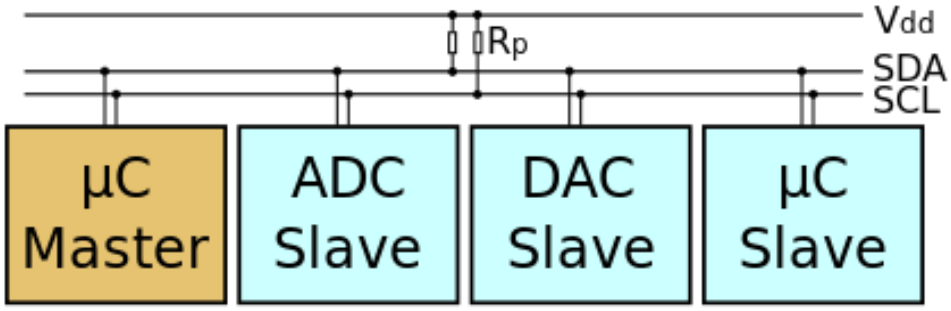


Fig. 2.3 Esquema I2C - Wikipedia

El protocolo contempla la presencia de múltiples maestros, que son los que inician la transferencia de datos y generan la señal de reloj. Se considera que el bus se encuentra en estado libre cuando SDA y SCL están en estado lógico alto y en este estado, cualquier dispositivo puede ocupar el bus como maestro. Las comunicaciones comienzan cuando el maestro envía un mensaje de condición de comienzo que pone a los demás dispositivos en escucha, y selecciona a quién se dirige mediante un byte con la dirección de 7 bits que lo identifica, quedando el bit menos significativo para indicar la operación deseada, de lectura (LSB=1) o escritura (LSB=0).

Los esclavos comparan el byte enviado con sus identificadores y el aludido responde al maestro con un mensaje de confirmación tras el cual comienza el intercambio. La comunicación es byte a byte, que son confirmados con un bit de ACK tanto por el maestro como por el esclavo, por lo que el proceso es bastante lento. El maestro controla la señal de reloj pero el esclavo puede pararla y dejar al maestro en un estado de espera de respuesta en caso de que no sea capaz de responder más rápido. La comunicación termina con un código de condición de parada tras el cual se vuelve al estado libre dejando así paso a otras comunicaciones.

USB

El estándar USB (Universal Serial Bus) fue desarrollado originalmente para sustituir a los lentos interfaces de conexión que eran los puertos serie y paralelo que se usaban tradicionalmente para conectar periféricos, pero se ha establecido como el interfaz preferido para éstos por su bajo coste, facilidad de uso y alta velocidad de transmisión.

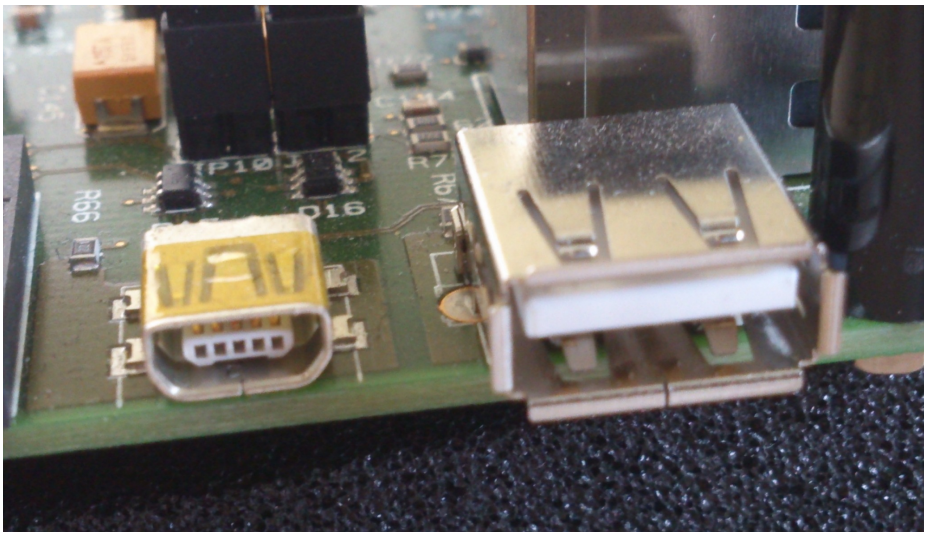


Fig. 2.4 Puertos USB

Es probablemente el periférico más conocido entre el público general, ya que se emplea en gran cantidad de dispositivos empleados en el día a día, como periféricos de ordenador en la forma de memorias, ratones y teclados y reproductores multimedia, pero también están presentes en otros aparatos como minicadenas, reproductores de vídeo y videoconsolas o incluso las radios de los coches. Está tan extendido que ha acabado por sustituir a interfaces de ordenador anteriores como los puertos serie y paralelo, y también a puesto fin a la separación entre el interfaz de datos y la alimentación de los dispositivos.

Existen seis tipos de conectores USB, clasificados entre normal, mini y micro, con versiones A y B de cada uno, pero todos siguen el mismo estándar en cuanto a los pines que emplean, siendo uno para la alimentación a 5V, dos para la transmisión de datos, uno en positivo y otro en negativo, y por último un pin de masa. Además, en los tipos mini y micro, hay un quinto pin, que en la versión A está conectado a tierra, mientras que en el B queda flotante.

Por otro lado, respecto a su velocidad, los USB se clasifican en 1.x, de baja velocidad, hasta 12 Mbps, 2.0, de alta velocidad, hasta 280 Mbps y 3.x, de super-velocidad, hasta 10 Gbps, siendo compatibles las versiones actualizadas con las anteriores.

UART/RS-232

El UART (Universal Asynchronous Receiver-Transmitter) es una pieza de hardware encargada de controlar los puertos y dispositivos serie, manejando las interrupciones de los dispositivos conectados al puerto y convirtiendo los datos entre serie y paralelo. Los UART suelen trabajar en conjunto con estándares de comunicación, de entre los cuales, el más conocido es el RS-232, que, si bien ya se ha quedado obsoleto en ordenadores a causa de la proliferación del USB, algunos dispositivos todavía lo utilizan para conexiones directas entre ellos. No obstante, los UART sólo necesitan dos pines para poder realizar comunicaciones, uno de transmisión y otro de recepción.

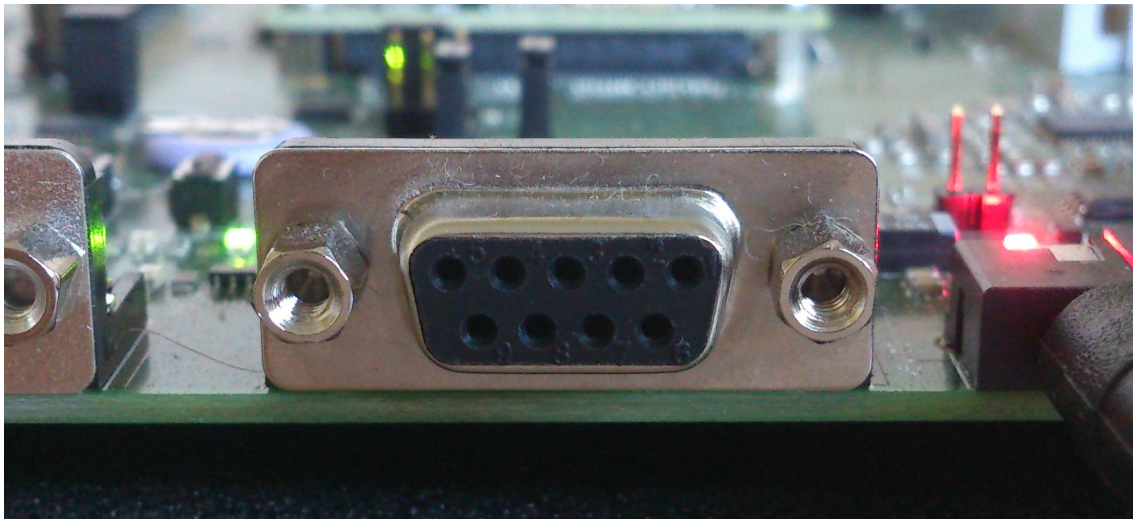


Fig. 2.5 Puerto RS232

Los UART tienen un registro de desplazamiento que se encarga de la transformación entre los formatos serie y paralelo, y los envía en grupos de ocho u once bits, en orden de menos significativo a más significativo, siendo el primero un bit de comienzo y los dos últimos uno de parada. El hecho de permitir la transmisión en serie permite la disminución de costes en los cables, ya que la tecnología es más sencilla pues transmitir en paralelo es muy complicado.

En el RS-232 hay nueve líneas que conforman el cable, una de tierra, dos de transmisión y recepción, dos de indicación de preparación de terminal y datos, dos de petición y aviso de envío, una de detección de portadora y la última de indicación de anillo. El estándar está diseñado para distancias cortas, menos de 15 m, y velocidades bajas, menores de 20 kbps, y la comunicación puede ser síncrona, asíncrona, simplex, half-duplex o full-duplex. Los conectores que emplea el estándar son los DB-25, que tienen 25 pines, aunque también se utilizan ampliamente los DE-9, que solo tienen los 9 pinas necesarios.

GPIO

Los puertos de entrada/salida de propósito genérico (GPIO) son pines de circuitos integrados que pueden ser utilizados a voluntad como entrada y como salida, y con un comportamiento controlado por el usuario, según le convenga. Estos pines no tienen ninguna función predefinida por lo que en origen permanecen inutilizados.

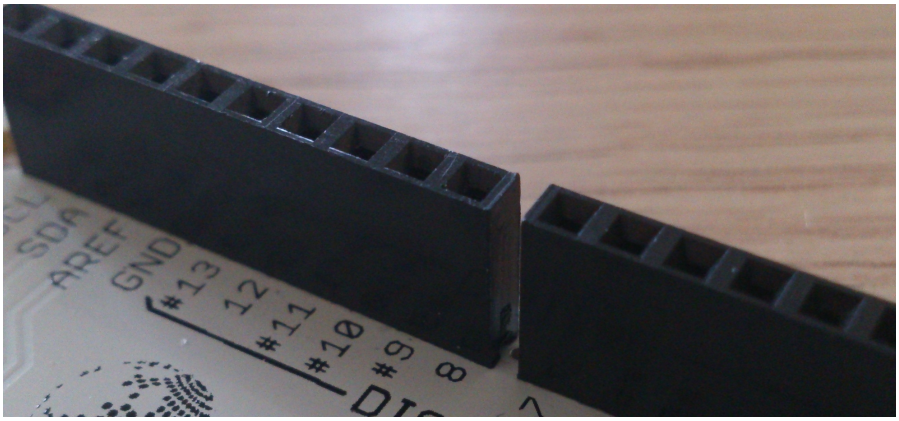


Fig. 2.6 GPIOs

Generalmente se utiliza en chips con pocos pines y también está extendido en sistemas embebidos, ya que el hecho de poder configurarlo según las necesidades de la aplicación le da al dispositivo una gran versatilidad, pudiendo conectarlo con todo tipo de dispositivos, como sensores de luminosidad, acelerómetros o pulsadores, y actuadores como motores, zumbadores o leds e incluso puede trabajar como entrada de señales de interrupción.

SATA

El SATA (Serial Advanced Technology Attachment) es un interfaz de conexión entre la placa base de un ordenador y dispositivos de almacenamiento masivo como discos duros. Es la evolución del Parallel ATA, respecto al cual aumenta la longitud de los cables, reduciendo su número y el coste, y además permite mayores velocidades y el intercambio de dispositivos en caliente.

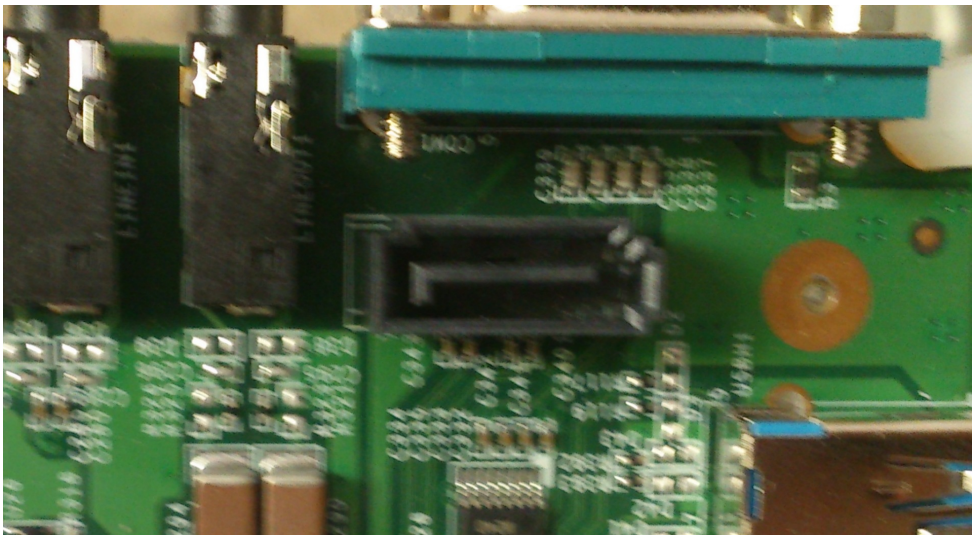


Fig. 2.7 Puerto SATA

Los conectores constan de 7 pines, 3 de ellos son masas, y los otros 4 se reparten en parejas, una para transmisión diferencial y la otra para recepción diferencial. Según sus velocidades se pueden clasificar como 3 generaciones de SATA internos, la 1.0 con velocidad de 150 Mbps, la 2.0 a 300 Mbps, 3.0 a 600 Mbps y su evolución, 3.2 hasta 1969 Mbps y por otro lado está el SATA externo, eSATA, hasta 300 Mbps.

2.2.3 Memoria

Junto con el procesador, la memoria es otra de las partes más importantes de los sistemas embebidos, ya que, por un lado, es en ésta donde se almacenan todos los programas que el primero ha de ejecutar, o que guardan información temporalmente de forma que el procesador pueda utilizarla de una forma rápida en la ejecución de programas.

La memoria RAM (Random Access Memory), es una clase de medio de almacenamiento de datos usado en ordenadores y otros dispositivos electrónicos que generalmente es de tipo volátil, lo que significa que cuando la alimentación eléctrica se corta, la información contenida en ella se pierde, por lo que este tipo de memoria sólo se usa cuando la máquina está encendida y funcionando, por lo que su uso se reduce al de memoria de trabajo para el software. No obstante también existen memorias de acceso aleatorio no volátiles, como las flash.

Este tipo de memoria se utiliza para cargar las instrucciones que ha de ejecutar el procesador y datos que pueda necesitar, de modo que éste pueda acceder a ello de una forma más rápida que leyendo del disco duro. Esta velocidad es lo que caracteriza a las memorias RAM y les da el nombre, pues “acceso aleatorio” significa que puede acceder a cualquier posición de memoria, tanto para escritura como para lectura, con los mismos tiempos de espera, sin necesidad de seguir un orden para los accesos.

RAM

Las memorias RAM se pueden separar en dos tipos diferentes, las SRAM (Static RAM) y las DRAM (Dynamic RAM). Las SRAM se caracterizan por estar basada en transistores que forman un biestable, por lo que no hace falta refrescar la información, sólo hay que mantener la alimentación para que no se pierda el estado de cada bit. Por el contrario en las DRAM, las células de memoria están compuestas por un transistor y un condensador, de modo que hay que refrescar los datos almacenados cada cierto tiempo porque los condensadores pierden carga paulatinamente y hay que restaurársela.

Por otro lado, las SRAM son más caras que las DRAM, porque la tecnología es evidentemente más compleja (6 transistores frente a 1 transistor y 1 condensador), pero generalmente es más rápida y requiere menos potencia, de modo que las DRAM predominan en los ordenadores, quedando las SRAM relegadas a uso como memoria de caché.

Dentro de las DRAM cabe destacar el subtipo DDR3 SDRAM (Double Data Rate type 3 Synchronous Dynamic RAM), una forma muy rápida de memoria RAM pues envía los datos dos veces por cada ciclo de reloj, permitiendo así trabajar al doble de la velocidad del bus del sistema sin necesidad de aumentar la frecuencia de reloj. El tipo 3 es la versión más actual de la tecnología, que está bastante extendida por su elevada velocidad de acceso, mayor que en DDR y DDR2.

ROM

Por otro lado está la ROM (Read-Only Memory), un tipo de memoria no volátil en la que los datos solo están accesibles para su lectura, pues las celdas de memoria son fijas, como por ejemplo en una matriz de diodos. Así pues, no se puede modificar la información en estos dispositivos, al menos no de una forma rápida o fácil, ya que se pueden estropear y quedar inutilizados.

Actualmente hay otros tipos de memorias que se siguen llamando ROM a pesar de que efectivamente se pueden borrar y reprogramar, porque en su modo normal de funcionamiento se utilizan sólo para lectura ya que el procedimiento de reprogramación es largo y pude requerir aparatos externos.

Originalmente, las memorias ROM venían programadas ya de fábrica para una aplicación en concreto, de forma que el diseñador le tenía que enviar los programas al fabricante a la hora de comprar las memorias. Posteriormente salieron al mercado las memorias PROM (Programmable ROM) que permitían programar los contenidos en las memorias mediante la aplicación de pulsos eléctricos de algo voltaje, pero sólo una única vez, tras lo cual se convertían en memorias ROM normales. Esto fue una gran ventaja porque agilizaba el proceso de diseño de dispositivos, pero seguía sin permitir corregir errores o aplicar modificaciones sencillas al mismo chip.

El siguiente paso que se logró fueron las memorias EPROM (Erasable PROM), que se podían devolver a su estado original desprogramado mediante la exposición de las memorias a fuerte luz ultravioleta, pero este procedimiento seguía siendo demasiado complejo, por lo que la evolución fue a las EEPROM, (Electrically EPROM), que se podían borrar y reprogramar eléctricamente en el mismo dispositivo final, simplificando el proceso y reduciendo su tiempo.

FLASH
La memoria flash es un tipo de memoria no volátil que surge como una versión mejorada de las memorias EEPROM, ya que se puede escribir y borrar mediante impulsos eléctricos, pero que consigue unas velocidades mucho más rápidas (de ahí su nombre) pues es capaz de borrar y escribir varias posiciones de memoria en cada operación.

Dentro de esta clase de memoria se pueden encontrar dos subclases diferentes, las flash NAND y las flash NOR. Ambas están formadas por transistores MOSFET pero por su configuración, cada variante tiene sus propias particularidades y limitaciones.

Por un lado, la flash tipo NOR puede ser mapeada directamente, aunque requiere de herramientas externas para reescribirla y otras acciones, sin embargo, en el caso de las flash NAND, no hay mapeado directo, por lo que para poder acceder a los datos que contiene, el kernel tiene que utilizar buffers para copiar los datos de la flash en la RAM.

Por otro lado si bien las memorias NAND son mucho más baratas y permiten una mayor densidad de almacenamiento, las NOR son mucho más rápidas en lectura, escritura y borrado, pudiendo escribir con precisión de un bit, en vez de palabras, y ofreciendo una mayor fiabilidad en general. Es por esto que en las flash NOR es donde se almacena el bootloader del sistema operativo, ya que este no puede contener fallos.

2.2.4 Comunicaciones

El apartado de comunicaciones también es muy importante en un sistema embebido, ya que es la forma de acceder a internet y de intercambiar datos con otras máquinas o con los usuarios.

Ethernet

Ethernet es una familia de tecnologías de comunicación de redes de ordenadores de área local (LAN), muy extendida en la actualidad, al punto de haber sustituido casi completamente a otras tecnologías LAN como token ring, FDDI y ARCNET convirtiéndose así en el estándar principal.

Existe una gran variedad de versiones del estándar de ethernet, el IEEE 802.3, que definen por un lado las características físicas del enlace, como la velocidad máxima posible, o el tipo de cable que se emplea y su longitud máxima, y por otro lado también define características del protocolo de la comunicación, como la longitud de las tramas o si se permite o no la negociación de velocidades. Así pues, las velocidades más típicas para ethernet son 10/100/1000 Mbps aunque también existen estándares de 10 Gbps, pero no tienen uso doméstico práctico, y los medios físicos son el cable coaxial (fino y grueso), el par trenzado de cobre (apantallado o sin apantallar) y la fibra óptica (monomodo y multimodo).

De todas estas opciones la más común es el par trenzado sin apantallar a 10/100/1000 Mbps con conectores RJ-45 ya que es el estándar utilizado para la conexión de los ordenadores a switches y routers para el acceso a internet en redes de ordenadores domésticas y profesionales. Los cables están compuestos por cuatro pares de hilos de cobre, trenzados para reducir interferencias, de los cuales se usan dos, uno para transmisión y otro para recepción.

La transmisión de los datos es diferencial, es decir, se envían de forma positiva por uno de los cables del par y de forma invertida por el otro y se restan en la recepción, mitigando así el efecto del ruido aditivo que pueda afectar a la comunicación. Por otro lado, la comunicación puede ser half-duplex o full-duplex según lo permitan los equipos conectados ya que, aunque las vías en cada sentido están separadas, el modo de transmisión es de múltiple acceso y detección de portadora, con detección de colisiones (CSMA/CD), por lo que en redes con repetidores, la comunicación requiere esperar a que el medio quede libre, impidiéndose así la bidireccionalidad.

Bluetooth

Bluetooth es un estándar de comunicación inalámbrica para la conexión de dispositivos a corta distancia mediante ondas de radio en la banda ISM, entre 2.4 y 2.485 GHz. Aunque originalmente estaba pensado para sustituir al estándar cableado RS232, en la actualidad se utiliza como método de creación de redes inalámbricas de área personal y de bajo consumo, permitiendo conectar entre sí diversos dispositivos como teléfonos, auriculares, ordenadores o impresoras, entre otros.

Por el momento existen 4 versiones del estándar, cada una con mayor velocidad y otras mejoras respecto a la anterior. La última modificación de la primera versión, la 1.2 presenta una velocidad máxima de 721 kbps, realiza saltos de frecuencia adaptable de espectro ampliado para reducir las interferencias de RF y permite conexiones síncronas extendidas para mejorar la calidad de enlaces de audio por reenvío de paquetes corruptos. La versión 2.1+EDR (Enhanced Data Rate) puede alcanzar hasta 2.1 Mbps gracias a la combinación de la modulación GFSK y PSK con dos variantes, pi/4DPSK y 8DPSK y además incluye una mejora de seguridad y otra de reducción de consumo. La versión 3.0+HS (High Speed) permite una velocidad teórica de hasta 24 Mbps, aunque la conexión no es propiamente por bluetooth, sino que la negociación y el establecimiento es por bluetooth y la transmisión por WiFi, y también incluye otra mejora de reducción del consumo de potencia. La versión

más moderna, la 4.1 se caracteriza por ser una versión de ultra bajo consumo, implementando todas las versiones anteriores mejoradas y añadiendo nuevas características que reducen el consumo y aumentan la velocidad empleando WiFi 802.11n.

GPRS

El GPRS (General Packet Radio Service) es una tecnología de transmisión de datos basada en conmutación de paquetes extendiendo la red GSM (Global System for Mobile communications) conformando la red conocida como 2.5G. Estas conexiones permiten servicios de mensajería SMS y MMS, servicios punto a punto (P2P) y punto a multipunto (P2M) como llamadas en grupo, WAP para el acceso a internet para el correo electrónico y páginas web y permite utilizar el dispositivo como un módem USB.

Este estándar de comunicación provee transmisiones de datos de velocidad moderada, entre 56 y 114 kbps, para múltiples usuarios simultáneos mediante la utilización de divisiones duales de frecuencia y método de acceso TDMA (Time Division Multiple Access). Por un lado, la división de frecuencias supone que el canal de subida funciona a una frecuencia diferente de la del canal de bajada, por lo que se puede realizar una transmisión full-duplex, si bien esta es asimétrica, porque la velocidad de subida es inferior a la de bajada. Por otro lado, para permitir el acceso múltiple, se asigna a cada usuario una franja temporal en la que puede enviar paquetes de longitud constante y el análogo para la bajada, el tiempo de emisión de la estación se divide entre los usuarios.

3G/UMTS

La tercera generación de redes móviles, UMTS (Universal Mobile Telecommunications System) es una evolución de la red GPRS y, como esta, a parte de las llamadas de voz permite la transmisión de datos, pero a mayor velocidad, lo que supone que se puedan añadir servicios, como la visualización de vídeo en streaming, la descarga de programas o las videollamadas. Además, aunque originalmente era un estándar pensado para terminales móviles, también se distribuyen dispositivos USB que proveen internet móvil para ordenadores, pues las velocidades son adecuadas para ordenadores personales, llegando hasta 56 Mbps teóricos de bajada y 22 Mbps de subida en la última versión de UMTS, HSPA+ (Evolved High Speed Packet Access).

A diferencia del sistema GPRS, el control de acceso al medio por los usuarios no se realiza por división temporal, TDMA, sino por división de código, CDMA (Code Division Multple Access), de forma que varios usuarios comparten el mismo canal simultáneamente, diferenciándose unos y otros por un código individual, diferente y ortogonal al resto, por el que se multiplican los paquetes quedando así codificados. Esta forma de multiplexación se basa en la técnica de espectro ensanchado, que resulta mucho más eficiente que las técnicas de las generaciones de redes anteriores y permite mayores tasas binarias.

WiFi

La tecnología WiFi es una forma de conexión inalámbrica de área local, que permite el intercambio de datos entre los dispositivos hallados en esa red y es uno de los métodos más extendidos para la conexión de equipos con routers de acceso a internet. Trabajando a 2.4 o 5 GHz según la versión del estándar 802.11, WiFi permite alcances de hasta 20 m para un punto de acceso en interiores y mayores distancias en exteriores puntos de acceso solapados.

La primera versión de WiFi, la 802.11a, opera en la banda de 5 GHz, aprovecha la desocupación de dicha banda para tener pocas interferencias, sin embargo, debido a la alta frecuencia, la distancia de cobertura es menor que en las versiones que funcionan a 2.4 GHz, porque las señales a mayor frecuencia sufren mayor atenuación. Este estándar provee una velocidad máxima teórica de 54 Mbps, que puede reducirse a varias velocidades inferiores, en caso de que el dispositivo conectado lo requiera, aunque en realidad la velocidad máxima posible ronda los 20 Mbps.

Las siguientes versiones extendidas de WiFi, 802.11b, 802.11g, operan en la banda de 2.4 GHz, lo cual permite mayor alcance con menor consumo de potencia, pero al tratarse de una banda ocupada por otras tecnologías inalámbricas, como Bluetooth, pueden aparecer interferencias severas. Las velocidades máximas reales para estos estándares son de unos 6 Mbps y 22 Mbps respectivamente para “b” y “g”.

El estándar comercial más actual es el 802.11n, permite un rendimiento mucho mayor que todos los anteriores pues trabaja simultáneamente en las bandas de 2.4 y 5 GHz, mediante el empleo de dos antenas. En esta versión, la velocidad máxima teórica está en los 600 Mbps y la máxima real se ha conseguido hasta de 300 Mbps.

Existen otros estándares WiFi, sin embargo, no son versiones domésticas y comerciales, o están todavía en fase de desarrollo, por lo que no hay routers que dispongan de esa tecnología en el mercado actual, si bien sí que hay algunos ordenadores modernos de gama alta que implementan tarjetas de red inalámbrica según la norma 802.11ac, versión mejorada de la 802.11n.

2.3 Sistemas Operativos

El sistema operativo es la parte no física, que se ejecuta por el procesador, se guarda en la memoria de programa y hace que funcione el sistema. A grandes rasgos es un código que contiene las instrucciones que se encargan de llevar a cabo las aplicaciones, contiene la información de cómo acceder a los puertos y a los programas y de cómo tratar los datos, de dónde cogerlos y adónde y de qué forma llevarlos.

Al igual que el procesador era la parte más fundamental del hardware, el sistema operativo es la parte más indispensable del software, ya que es la pieza central sobre la que se incluyen el resto de programas, por lo que sin él, es dispositivo no funciona.

2.3.1 Con o sin SO (barebone)

A la hora de instalar un sistema operativo en una tarjeta de sistema embebido hay dos opciones entre las que elegir dependiendo de las necesidades de la aplicación y de las limitaciones del procesador: el sistema operativo completo, similar a un ordenador, con entorno gráfico como Debian, Ubuntu, etc., o un sistema operativo reducido al mínimo, que solo tiene el bootloader, el kernel y el sistema de archivos.

La primera diferencia entre ellos es la preparación del sistema, ya que en un sistema operativo completo, la instalación pasa por descargar una distribución adecuada a la arquitectura del procesador y más o menos optimizada para el sistema embebido en cuestión, y el resto de la gestión de programas y archivos se hace internamente accediendo al sistema como si se tratara de un ordenador. Por el contrario, en un sistema barebone, hay que preparar a mano el kernel y el sistema de archivos raíz, incluyendo la descarga de archivos en código fuente, la compilación, la instalación en el sistema y la creación de la imagen.

Esto resulta muy interesante puesto que el sistema operativo incluye muchos servicios implementados de antemano y además, suelen existir repositorios o servidores con programas ya compilados para esa distribución específica, lo cual simplifica mucho la instalación y modificación de archivos, pues están ya listos y no hace falta ninguna acción adicional. Por el contrario, como ya se ha dicho, en un sistema barebone se hace necesario compilar todos los programas y aplicaciones antes de la instalación con un compilador, ya que, como el software es una distribución particular, y puede ir en muchos tipos de procesadores, no suele haber un lugar de donde bajarlos ya preparados.

Por todo lo anterior, es evidente que el empleo de un sistema operativo resulta más sencillo que el diseño de una distribución de software barebone. Sin embargo, el sistema operativo es mucho más pesado que un barebone a medida, ya que el primero es mucho más complejo y necesita muchas más cosas para funcionar, y por esto, también resulta más lento en su ejecución y supone un mayor consumo de recursos, tanto de memoria como de potencia, por tener que ejecutar procesos innecesarios.

Así pues, un sistema muy simplificado y hecho a medida permite realizar un dispositivo más optimizado en tamaño, tiempos de respuesta y consumo energético, ya que solo tiene y gasta lo que necesita para realizar las funciones, mientras que un sistema operativo al que se le quita todo lo que sobra, que pueden ser muchos programas, al final presenta un peor resultado porque la metodología es más sucia ya que hay cosas que no se pueden sacar y el sistema sigue siendo grande.

Por lo tanto, para los sistemas embebidos, el barebone es generalmente la mejor opción, porque suelen tener importantes limitaciones de potencia, memoria y tamaño, ya que no se pueden ampliar como un PC. Por el contrario, una placa con un sistema operativo completo también puede resultar útil a la hora del desarrollo y prueba rápida de posibles aplicaciones antes de hacer la versión definitiva, pues el proceso es más sencillo, da menos problemas y la poca optimización de recursos no es crítica pues no es un producto final. No obstante, una vez hecha la prueba, sí que hay que adecuarlo todo a la versión reducida.

Por último, el hecho de aprender a trabajar con un sistema embebido sin sistema operativo de alto nivel ayuda a entender bien cómo es el funcionamiento interno de un ordenador o sistema similar, lo cual puede resultar muy interesante de cara al desarrollo de aplicaciones y resolución de problemas.

2.3.2 Linux

Creado por Linus Torvalds en 1991, Linux es probablemente el kernel de sistema operativo, libre y de código abierto, más conocido en la actualidad. Su amplia expansión en los últimos es debida a que

existe una gran comunidad de usuarios y desarrolladores que contribuyen a diseñar nuevo código y mejorar el presente.

Linux es un sistema basado en Unix, por lo que su estructura de directorios es la misma que la de dicho sistema, siguiendo la norma de la jerarquía del sistema de archivos. Este sistema de directorios tiene como origen el directorio raíz, de símbolo "/ ", que es el que contiene todos los demás. Así pues, los directorios contenidos en la raíz, denotados como "/directorio", contienen a su vez otros directorios y archivos, cuyas rutas se indican como "/directorio/subdirectorio/archivo", y así sucesivamente, creando de este modo una estructura ramificada que es la base del funcionamiento de Linux.

El kernel de Linux puede utilizarse solo mediante una consola de comandos, cambiando de directorios y ejecutando los programas a través de ella, lo cual era la forma original de empleo, de manera similar al MS-DOS, pero con el tiempo y la difusión del sistema entre el público general, acostumbrado a sistemas con interfaz gráfico con ventanas, han surgido multitud de distribuciones de sistemas operativos basados en Linux, que incorporan estas características, como Ubuntu o Fedora, más atractivas y amigables para los usuarios.

A parte de la inclusión del entorno gráfico, también hay diversas distribuciones que incluyen otras características adecuadas para diversas utilidades que se requieran, por ejemplo, del propio Fedora existen varias versiones preconstruidas orientadas a diferentes cosas, como diseño gráfico, seguridad o análisis matemático, ente otras.

El mayor atractivo del kernel de Linux y sus distribuciones es, como se ha dicho, que es un sistema libre, por lo que se puede modificar y configurar como convenga, y que hay una gran comunidad detrás, apoyándolo, por lo que es sencillo conseguir ayuda técnica. Además, por ser libre, se puede comercializar un producto que funcione con Linux sin tener que pagar royalties al dueño del software.

Concretamente, para sistemas embebidos, otra ventaja a parte de lo anterior es que, como se puede crear un kernel a medida de las necesidades de la aplicación, se pueden conseguir distribuciones de tamaño muy reducido y bastante optimizadas para conseguir un consumo reducido y mejor aprovechamiento de los recursos de la tarjeta. Además, por el hecho de que está muy extendido, Linux es compatible con prácticamente todas las arquitecturas de los procesadores comerciales actuales.

Sin embargo, Linux no es un sistema en tiempo real, si bien se puede aproximar a esa característica mediante el empleo de módulos, por lo que no resulta una opción especialmente idónea para casos en los que la respuesta en tiempo real sea demasiado crítica.

2.3.3 Windows Embedded

Windows embebido es una distribución del sistema operativo de Microsoft para su uso en sistemas embebidos, que puede ser de diversos tipos, los cuales se ajustan a las necesidades de la aplicación para la que van a ser empleados. Así pues, hay versiones para industria, para el sector automovilístico, para dispositivos móviles y navegadores, versiones para servidores, compactas estándar y profesionales.

De todas estas, las más importantes para sistemas embebidos como los que contempla este trabajo son la versión servidor, que es una adaptación de Windows Server para ordenador, y la versión compacta, que es un sistema de tamaño súper reducido, ya que está pensada para ordenadores y sistemas embebidos muy pequeños, como descodificadores y teléfonos móviles, y es compatible con varios procesadores como ARM, x86 y MIPS.

La ventaja de estas distribuciones es que vienen preparadas y ajustadas expresamente para las aplicaciones para las que están pensadas y además cuentan con el respaldo de una compañía grande y potente, por lo que estos sistemas operativos están muy optimizados para dichas tareas, son muy robustos frente a fallos por las exhaustivas pruebas a las que son sometidos, y además, hay un servicio técnico profesional que se encarga de resolver los posibles problemas que puedan aparecer, lo que en conjunto supone una garantía de funcionamiento.

Sin embargo, este software no es libre ni de código abierto, por lo que, en caso de querer usarlo, es necesario pagar una licencia y unas tasas por venta. Además no se puede modificar ni versionar, de forma que si el servicio no se ajusta a las necesidades del producto, no se puede remediar salvo que Microsoft ofrezca una solución, y, por otro lado, si la empresa en algún momento interrumpiera el soporte al software, no se podría actualizar, ni mejorar o corregir fallos.

2.3.4 Android

Android es un sistema operativo de Google basado en el kernel de Linux, con una interfaz de usuario basada en el control táctil y diseñado principalmente para dispositivos con esta forma de control, como smartphones y tablets, aunque también se puede encontrar en otros dispositivos electrónicos como televisiones o videoconsolas.

Al igual que Linux, Android es un sistema operativo libre y de código abierto por lo que se puede versionar y distribuir gratuitamente, tal y como hacen los fabricantes de terminales, los cuales, por ejemplo, adaptan el estilo gráfico a una forma personalizada, que si bien funcionan todas igual, contienen señas distintivas de la marca.

El sistema es compatible con diversas arquitectruas entre las que se encuentran ARM, x86 y MIPS, y su estructura se basa en aplicaciones que se ejecutan en un arco de trabajo de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución.

Por lo mismo que Linux, Android tiene una gran comunidad de usuarios y diseñadores de aplicaciones dándole soporte, así como las empresas fabricantes de los dispositivos, por lo que hay gran cantidad de software y de documentación disponible, de cara a la utilización y programación de software.

2.3.5 QNX

El sistema QNX es un sistema operativo en tiempo real tipo Unix, diseñado para el mercado de los sistemas embebidos por QNX Software Systems, posteriormente comprada por BlackBerry. Este sistema fue el primer microkernel de sistema operativo en ser comercialmente exitoso y se utiliza en

diversos dispositivos como routers de alta capacidad, simuladores de vuelo o sistemas de efectos especiales.

El sistema basado en microkernel consiste en hacer correr el kernel del sistema operativo dividido en múltiples tareas pequeñas denominadas servidores, en vez de la forma más tradicional de kernel monolítico, que todo el sistema operativo es un programa muy largo con muchas partes con características particulares. De esta forma, QNX permite a los usuarios apagar cualquier funcionalidad no requerida en vez de tener que cambiar el sistema operativo entero, simplemente esos servidores no corren. Esto supone que se puede conseguir una enorme flexibilidad de una forma muy sencilla y con un sistema bastante pequeño.

El sistema funciona prácticamente en todos los procesadores de sistemas embebidos en la actualidad, como PowerPC, ARM, x86 y MIPS entre otros y dispone de licencias no comerciales y para estudiantes, pero no está disponible gratuitamente para aficionados o publico general.

2.3.6 Xenomai

El sistema Xenomai es un marco de trabajo en cooperación con el kernel de Linux para conseguir un soporte en tiempo real duro para aplicaciones de usuario y perfectamente integrado en el entorno de Linux.

Xenomai está basado en un núcleo abstracto de sistema operativo en tiempo real (RTOS), sable para construir cualquier tipo de interfaces de tiempo real, sobre un núcleo que exporte un juego de servicios de RTOS.

El sistema puede correr sobre 7 arquitecturas, entre las cuales están ARM, x86, x86_64, PowerPC y PowerPC64.

2.4 Elección

Una vez descrito cuáles son las características que componen un sistema embebido hay que elegir algo de entre lo disponible que se ajuste lo mejor posible a los requisitos de la aplicación contemplada.

De entrada, como el sistema tiene que consumir lo menos posible, se ha decidido que el procesador tiene que estar basado en arquitectura ARM, y, puesto que se quiere hacer que el sistema operativo cumpla unos requisitos particulares y el producto se va a comercializar, el sistema operativo más adecuado será uno basado en Linux (barebone o con sistema operativo por encima), ya que al ser libre, es personalizable y se puede distribuir gratuitamente.

Por esto, se van a analizar las distintas opciones actuales de sistemas embebidos, en cuanto a procesador ARM y sistemas operativos basados en Linux. En función de todo esto se elegirá una opción que se considere la más adecuada del mercado y se procederá a su estudio más en profundidad.

2.4.1 Alternativas

Tal y como se ha dicho, el mercado actual de los sistemas embebidos está creciendo y extendiéndose enormemente en múltiples sectores y para diferentes aplicaciones por lo que hay una gran variedad de procesadores por distintos fabricantes y varios sistemas operativos tanto distribuidos por fabricantes, como desarrollados por comunidades usuarios aficionados al software libre.

2.4.1.1 Procesador

AM335x de Texas Instruments

La mejor oferta de Texas Instruments en el campo de sistemas embebidos es la familia de procesadores AM335x, que son sistemas ARM Cortex-A8, muy apropiados para aplicaciones que necesiten un rango amplio de interfaces y conectividad de red. Con un núcleo de hasta 1 GHz, una RAM DDR3 de hasta 1 GB y una flash de hasta 2 GB, hay una gran variedad de procesadores para ajustarse adecuadamente a las necesidades de la aplicación y, además, los AM335x pueden configurarse de distintas maneras para controlar la relación entre rendimiento y eficiencia energética, pudiendo conseguir un dispositivo optimizado en uno de los dos aspectos o uno neutral, según convenga al diseño.

Por otro lado, los AM335x ofrecen un gran número de interfaces y periféricos distintos, como seis UART, SPI, CAN, I2C, dos USB, un Ethernet 10/100 y otro 10/100/1000, pantalla y convertidores A/D de 12 bits. Además, estos procesadores ofrecen la posibilidad de trabajar con aplicaciones de tiempo real y soportan como sistemas operativos, Linux, Android y Windows embebido entre otros.

iMX 6 de Freescale

La gama iMX 6 de Freescale está formada por procesadores ARM Cortex-A9, disponibles en versiones de uno (Solo), dos (Dual) y cuatro (Quad) núcleos a 1 GHz y 512 MB, 1 GB y 2 GB de RAM DDR3 respectivamente.

Estos procesadores ofrecen como puertos y periféricos un HDMI, un interfaz de cámara, un UART, un USB y un Ethernet 10/100/1000, añadiendo módulos WiFi y Bluetooth en las versiones Dual y Quad, y un puerto SATA sólo para la Quad. Además, los iMX6 son compatibles con varios sistemas operativos como por ejemplo Ubuntu, Yocto y Android.

Existen otras empresas fabricantes de procesadores con arquitectura ARMv7 recomendados por la propia empresa ARM, pero al buscar información ha resultado que o bien no eran adecuados para esta aplicación, como en el caso de Samsung, o que no eran de la familia Cortex, como en el caso de ST o simplemente, no aparecía información, como con Broadcom.

2.4.1.2 Software

Ubuntu

Existen distribuciones de Ubuntu compatibles con ARM pensadas para su instalación en sistemas embebidos, con imágenes preparadas para copiar en una tarjeta SD desde donde ejecutarlas. Esta es una opción que facilita mucho las cosas en cuanto a la gestión de la placa, pues funciona igual que una distribución de Ubuntu para PC.

Sin embargo, estas distribuciones suelen ser bastante pesadas y no están optimizadas en cuanto a programas y complementos incluidos, pues son distribuciones genéricas, no personalizadas. Si el procesador está muy limitado, la ejecución del sistema y de los programas puede hacerse muy lenta y no funcionar del todo bien, o al menos, no cumpliendo los requisitos de velocidad y consumo que se esperan de un sistema embebido como el de la aplicación que se está tratando.

Kernel Linux de Pengutronix
Pengutronix es una empresa que se dedica a desarrollar herramientas para la confección y desarrollo de núcleos de Linux personalizados para empresas. Distribuyen todo lo necesario para construir un kernel a medida, el compilador, los toolchain para la compilación cruzada, el bootloader y la documentación, y además ofrecen soporte técnico a través de listas de correo.

El resultado que se obtiene con este método es un núcleo de Linux muy reducido, adecuado a las necesidades de la aplicación desarrollada, por lo que se consigue optimizar en gran medida el consumo de recursos, ya que solo contiene lo que necesita. A pesar de esto, el sistema puede sufrir importantes limitaciones pues, a pesar de que existir una comunidad en torno a la herramienta, se depende directamente de la empresa para solucionar los posibles problemas y producir las mejoras. Por lo tanto, es posible que no se puedan conseguir las prestaciones requeridas por la aplicación.

Yocto + OpenEmbedded
Yocto Project es un grupo de trabajo de la Fundación Linux cuya objetivo es producir herramientas buenas para la creación de distribuciones de Linux para sistemas embebidos, independientemente de la arquitectura subyacente. El proyecto es libre y abierto y está enfocado en mejorar el proceso de desarrollo de software, por lo que se asoció con OpenEmbedded, otro marco de software abierto para la creación de núcleos de Linux para embebidos con el que Yocto comparte objetivos, dando como resultado el proyecto OpenEmbedded-Core.

El sistema se basa en recetas de BitBake, una herramienta de compilación similar a “make” enfocada especialmente en compilación cruzada para sistemas embebidos y permite crear distribuciones de Linux que soporten las principales arquitecturas de sistemas embebidos, ARM, MIPS, PowerPC y x86/x86_64. Además es un sistema que permite hacer personalizaciones de forma sencilla, puede cross-compilar miles de paquetes y con él se pueden hacer versiones de distribuciones de Linux conocidas como Ångström u Openmoko.

2.4.2 SBC Basado En AM335x Con Linux

Finalmente, según todo lo anteriormente dicho, y con base en las necesidades y características definidas en los objetivos, la opción elegida para la aplicación del gateway es una basada en el AM335x con linux porque es de bajo consumo, tiene buenas prestaciones lo cual promete buenos resultados y al ser linux es fácilmente modificable y tiene una buena comunidad detrás y además, se puede distribuir gratuitamente. Además, este procesador aparece como ejemplo de un gateway para redes inteligentes de baja energía recomendado por Texas Instruments.

De entre las opciones comerciales que se ajustan a esta disposición, se ha elegido la placa de desarrollo phyCORE-AM335x de la empresa Phytec, pues dispone de todo lo necesario para conseguir y probar la aplicación contemplada en este trabajo y todas las posibles aplicaciones que pudieran interesar a la empresa en un futuro. Esta tarjeta, sin embargo, no es el modelo definitivo que se va a usar cuando el proyecto se lleve a cabo, pero el procesador y los puertos utilizados son idénticos, por lo que el software desarrollado será completamente compatible, y el resultado de la ejecución será el mismo.

Como software se ha elegido la distribución de Linux hecha con la herramienta de pengutronix que la propia Phytec suministra con la placa.

Fig. 2.8 CPU de la phyCORE-AM335x

3. SBC AM335 CON LINUX

3.1 Características Técnicas Hardware De La SBC

SOFTWARE	
Sistema operativo	Linux 3.x
Tiempo real	Rtppreempt
BSP / Image	Sí
Boot loader	Barebox
Tool Chain + IDE	OSELAS Toolchain + Eclipse
Compilador	GNU
Interfaz de debug	JTAG
CPU	TI AM3359
Arquitectura	Cortex A8
Fabricante	Texas Instruments
Frecuencia de reloj	720 MHz
MEMORIA	
Flash NAND	512 MB
Flash NOR	8 MB (SPI)
DDR RAM	512 MB DDR3
EEPROM	4 kB
INTERFACES	
Ethernet	1x 10/100 Mbps, 1x 10/100/1000 Mbps
RS232	1 (SUB-D 9)
USB-OTG	1
USB-HOST	1
MMC / SDIO Ranura de tarjeta	1
Profibus	1 (SUB-D 9)
ADC	8 vía conector de expansión
UART	4 vía conector de expansión
SPI	1 vía conector de expansión
I2C	1 vía conector de expansión
CAN	1
Pantalla	1
Fuente de voltaje	DC 5V

Tabla 3.1 Características phyCORE-AM335x - Phytec

3.2 Desarrollo De Linux Embebidos

3.2.1 Componentes software

3.2.1.1 Cross-compilation Toolchain

Una cadena de herramientas, o toolchain, es el conjunto de herramientas de software necesarias para construir software de ordenador, que tradicionalmente incluye el paquete de binutils, el compilador de C y otros lenguajes y la librería de C y sus cabeceras.

Las binutils son las herramientas encargadas de generar y manipular los binarios para un procesador concreto. De entre todas estas, las más importantes son el ensamblador, llamado as, de assambler, que se encarga de convertir el código de programa de ensamblador a código fuente y el enlazador, que se encarga de crear archivos ejecutables o librerías a partir de ficheros objeto en las primeras etapas de la compilación. En la siguiente figura se puede ver una lista completa de las herramientas de binutils.

Utility	Use
as	GNU assembler
ld	GNU linker
gasp	GNU assembler pre-processor
ar	Creates and manipulates archive content
nmu	Lists the symbols in an object file
objcopy	Copies and translates object files
objdump	Displays information about the content of object files
ranlib	Generates an index to the content of an archive
readelf	Displays information about an ELF format object file
size	Lists the sizes of sections within an object file
strings	Prints the strings of printable characters in object files
strip	Strips symbols from object files
c++filt	Converts low-level, mangled assembly labels resulting from overloaded C++ functions to their user-level names
addr2line	Converts addresses into line numbers within original source files

Tabla 3.2 Lista de binutils - Building Embedded Linux (Ver 6.2)

La colección de compiladores de GNU (GCC) son las herramientas encargadas de la compilación de programas de diversos lenguajes, como C, C++ y Java para un gran número de arquitecturas, incluyendo ARM, x86, PowerPC entre otras.

La librería de C es una librería de código compartido que actúa como un una interfaz entre las aplicaciones y el kernel de Linux y se usa prácticamente por todas aquellas que corren en un sistema

de Linux. Existen diferentes tipos que se pueden usar, pero se debe elegir cuál en el momento de crear el toolchain, pues la GCC se compila usando dicha librería.

Algunas toolchains incluyen librerías de código adicionales y otras herramientas suplementarias como un comprobador de memoria o un debugger. Adicionalmente, aunque no se suele contar como parte de la toolchain, se puede trabajar con un IDE que dé un interfaz para trabajar con esas herramientas más fácilmente.

Una toolchain multiplataforma, o cross toolchain, es aquella que está hecha para funcionar en un dispositivo de desarrollo, pero se utiliza para hacer programas que corren en otro dispositivo, como suele pasar cuando el software que hay que desarrollar es para sistemas embebidos.

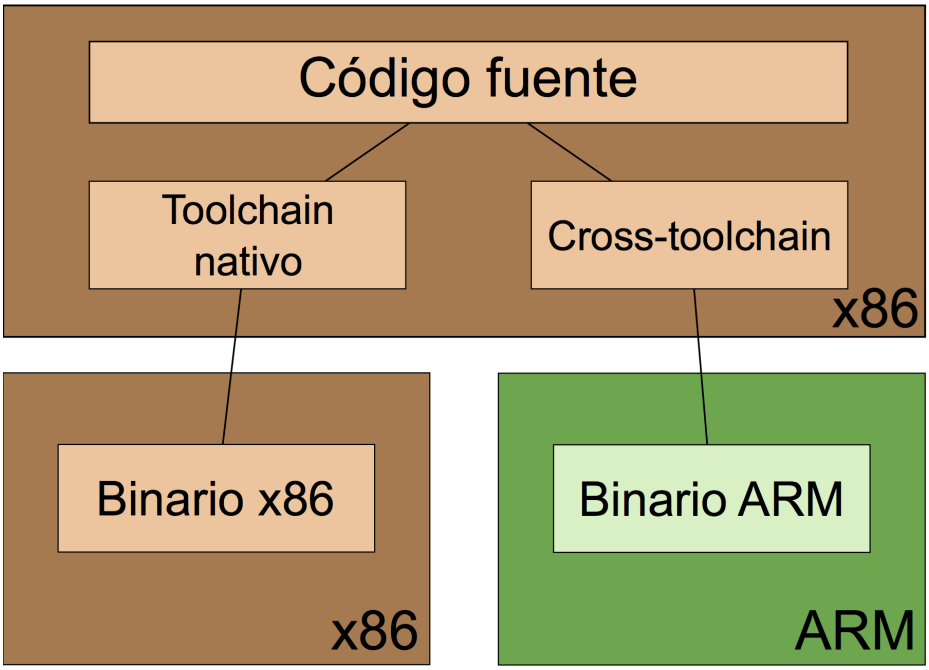


Fig. 3.1 Toolchain Vs CrossToolchain - Linux para sistemas embebidos (Chiesa/De Andrés)

En este caso se va a utilizar la toolchain de OSELAS v2012.12 incluido en el software del kit de desarrollo, que está preparado para construir el software necesario para utilizar linux en el procesador Cortex A8 de la tarjeta phyCORE.

3.2.1.2 Bootloader

El gestor de arranque o bootloader, es el programa que, iniciado por el hardware tras el encendido, se encarga de la inicialización básica del sistema, obtiene el kernel de una flash u otro tipo de almacenamiento no volátil y lo carga en la RAM desde donde lo ejecuta. A menudo, los bootloaders tienen varias etapas, es decir, hay más de uno de esos pequeños programas que se van llamando entre sí, de más sencillo a más complejo.

Además de estas funciones, la mayoría de los bootloaders realizan otras en el arranque, como generar una consola serie con línea de comandos, o inspeccionar la memoria para realizar un diagnóstico de

error. Cuantas más funciones simultáneas, más complejo es el bootloader, por lo que en sistemas con varias etapas, son las que más tarde se ejecutan las que más funciones proveen.

En los procesadores de la familia x86 para ordenador existe una memoria no volátil con un programa, la BIOS, que se ejecuta cada vez que el sistema se enciende o resetea, y se encarga de inicializar el hardware y cargar un programa pequeño, generalmente de 512 bytes, de un medio no volátil. Este programa suele ser un bootloader de primera etapa que luego carga el bootloader completo, como GRUB, que se encarga de ejecutar el sistema operativo y el sistema alcanza su estado normal.

En los sistemas embebidos, el sistema de arranque primario depende de cada procesador y cada placa, pues algunas tienen el bootloader escrito en una flash NOR desde donde el procesador comienza a ejecutar instrucciones pero otras tienen el código integrado en una ROM que automáticamente carga una porción de una flash NAND a una SRAM la cual se encarga de llamar al bootloader principal.

Para el caso particular de la placa elegida para el proyecto, hay dos bootloaders, barebox, de etapa primaria y MLO, de etapa secundaria, que los suministran Phytec y pengutronix listos para usar.

3.2.1.3 Kernel

El núcleo o kernel es la parte más fundamental del software de un sistema de linux, pues es el responsable de administrar el acceso al hardware por parte de los programas, o de otra forma, gestiona los recursos a través de servicios de llamada al sistema.

Algunos de los recursos administrados por el kernel incluyen el tiempo de procesador dado a los programas, el uso de la RAM disponible, y el acceso a los dispositivos. El kernel provee varias abstracciones por las cuales los programas pueden pueden pedir acceso a los recursos del sistema sin tener que comunicarse directamente con el hardware.

Estas capacidades ofrecidas por el kernel de Linux son configurables cuando éste se está construyendo, lo cual permite eliminar capacidades que nunca se van a usar, o, por el contrario, añadir la posibilidad de usar algún dispositivo particular que sea exclusivo del sistema objetivo elegido. Por lo tanto, para poder funcionar bien, cada sistema necesita un kernel determinado, adaptado a sus características propias.

Existen distintos tipos de núcleos que se diferencian en cómo gestionan las llamadas de sistema y los recursos disponibles. Así pues, en un kernel de tipo monolítico, tradicionalmente usados por los sistemas tipo Unix, tienen implementadas todas las funciones del núcleo y los controladores de dispositivos, lo que permite una gran variedad de abstracciones muy potentes.

Por el contrario, su principal desventaja es que hay muchas dependencias entre componentes del sistema, de forma que un fallo puede destruir el sistema entero, y, además, los kernels muy grandes son difíciles de mantener.

Por otro lado los microkernels, o micronúcleos, se basan en pasar del núcleo tradicional a unos pequeños núcleos llamados “servidores” como se ha dicho hablando del sistema operativo QNX. Así, un microkernel se diseña para una plataforma concreta, liberando todo lo que sobra y quedándose con lo que se necesita para funcionar y consiguiendo así mayor funcionalidad. De este modo, el mantenimiento resulta más sencillo que en los monolíticos, los fallos individuales no resultan catastróficos y el tiempo de desarrollo de software es menor porque no requiere reiniciar la tarjeta para probar los programas nuevos.

Sin embargo, también tiene algunos problemas como el hecho de que la gestión de los procesos es complicada, consume más memoria y además, el rendimiento depende del contexto, si hay muchos procesos se comporta diferente que si hay pocos.

Los núcleos híbridos o modulares se utilizan en la mayoría de sistemas operativos comerciales, su comportamiento es similar a los microkernels pero además incluyen código adicional para conseguir un mejor rendimiento, es decir, consiguiendo algunas propiedades de monolíticos.

Por último, los exokernels son un tipo de núcleo que todavía está en fase experimental que se diferencia de otros tipos de kernel en cuanto a que no otorga ninguna abstracción sobre la que desarrollar una aplicación. Sin embargo, esto permite una grandísima funcionalidad puesto que permite a los desarrolladores determinar cómo hacer el uso más eficiente del hardware para un determinado programa.

Para la tarjeta phyCORE se utiliza un kernel que se genera cada vez que se ejecuta el compilador del sistema de archivos, bajo el nombre “linuximage”.

3.2.1.4 Drivers

Un controlador de dispositivos, o driver, es un programa informático que se encarga de operar o controlar a un periférico de un ordenador u otro sistema electrónico. El driver proporciona una interfaz para usar dicho dispositivo, una forma de abstracción que permite que otro programa pueda acceder al él sin necesidad de saber cómo se controla físicamente.

En otras palabras, los controladores actúan como traductores entre un dispositivo físico y las aplicaciones o sistemas operativos que las quieran usar. Por lo tanto, los drivers resultan indispensables para poder usar el hardware.

Un ejemplo de el funcionamiento de un driver puede ser una programa de alto nivel para la comunicación por un puerto serie, que sólo tenga dos funciones, como recibir y enviar. A bajo nivel, el driver que implemente las funciones tendrá que comunicarse con el controlador del puerto, cosa que hará de forma muy diferente si tiene que controlar un UART o un FTDI, pero sin embargo, el driver abstrae esos detalles de forma que para la aplicación no hay diferencia.

Existe una gran cantidad de tipos de drivers, puesto que cada periférico necesita uno y muchas veces hay varios para el mismo dispositivo, cada uno dando distintas funcionalidades. Generalmente los

controladores los escriben y distribuyen los propios fabricantes de los periféricos, pues son los que mejor lo conocen, pero también se pueden encontrar versiones libres hechas por los creadores de un sistema operativo libre, generalmente en colaboración con los fabricantes del dispositivo, y por último también hay versiones no oficiales conseguidas mediante ingeniería inversa.

Una variación de los controladores son los drivers virtuales, que se usan para emular un dispositivo en sistemas operativos virtualizados, por ejemplo al utilizarlos en una máquina virtual como virtualbox. De este modo, los drivers del sistema simulado actúan como si pudieran acceder realmente al periférico, pero en realidad lo que hacen es acceder a los drivers del sistema real como si fuera una llamada a una función.

Evidentemente, para este proyecto se van a utilizar los drivers que proporciona el fabricante de la placa incluidos en el software del kit de desarrollo.

3.2.1.5 Root Filesystem

Como ya se ha dicho, Linux es un sistema operativo que se basa en Unix y de él toma la estructura jerárquica de ficheros. El “root filesystem” es el sistema de ficheros que se encuentran contenidos en el directorio raíz, el primero de todos, denominado con el símbolo “/ “. A partir de este sistema de archivos se derivan todos los demás, dando así al sistema global una estructura ramificada con forma de árbol, que crece y se expande desde la raíz, y a la que por tanto se le llama árbol de directorios.

El root filesystem, por tanto, contiene todas las librerías, aplicaciones y datos del sistema, por lo que armarlo es una de las tareas más importantes al crear el sistema operativo. No obstante, el kernel y el bootloader suelen estar separados fuera del root filesystem, de forma que el usuario no pueda tenerlos próximos y pueda estropearlos.

Algunos directorios típicos del sistema raíz son los siguientes

Símbolo	**Contenido**
/	Raíz
/bin	Binarios necesarios
/lib	Librerías fundamentales del sistema
/etc	Descriptores de dispositivos
/sbin	Archivos de Configuración
/home	Directorios de los usuarios
/root	Archivos del usuario root
/usr	Binarios y librerías complementarios
/tmp	Directorio para archivos temporales
/var	Servicios, páginas web, BBDD

Tabla 3.3 Root filesystem - Linux para sistemas embebidos (Chiesa/De Andrés)

Los contenidos exactos del sistema de archivos raíz cambia dependiendo del ordenador que lo contenga, pero todas las versiones tienen que contener los archivos que son necesarios para el arranque del sistema y llevarlo a un estado en el que el resto de archivos puedan ser montados y también herramientas para reparar sistemas de archivos y para recuperar archivos desde un backup.

El root filesystem generalmente debería ser bastante pequeño, porque contiene archivos críticos y por tanto, cuanto más pequeño sea, menor probabilidad de que se corrompa habrá. Un root filesystem estropeado generalmente supone que el sistema no pueda arrancar desde el medio que se haya corrompido y necesitará de una forma de arranque alternativa.

Además, el tamaño pequeño del filesystem también resulta importante desde el punto de vista de eficiencia de recursos, ya que un sistema muy grande, que contenga programas y archivos indeseados, seguramente consuma recursos de memoria y procesador y el dispositivo que lo contenga funcionará peor que si estuviera limpio.

Para este trabajo, el root filesystem se construirá sólo con las opciones necesarias y deseadas mediante la herramienta ptxdist como se explicará a continuación.

3.2.2 Creación Del Sistema Linux

Para la poder utilizar el sistema Linux en la phyCORE-AM335x de Phytec hacen falta cuatro archivos, los bootloaders primario y secundario, la imagen del kernel y la imagen del root filesystem, de los cuales, los dos primeros vienen ya creados con el software y los segundos se crean con una de las herramientas incluidas, la aplicación ptxdist.

3.2.1.1 BSP

El BSP (Board Support Package) es el conjunto de los archivos necesarios para el arranque del Linux en el sistema embebido para el que estén diseñados, que son, los archivos de arranque, bootloaders, y el kernel con la configuración del entorno.

El bootloader para la tarjeta phyCORE es Barebox, el cual se puede configurar con variables de entorno y scripts, pero la distribución de Phytec ya tiene una versión adecuada para ella, por lo que no hace falta modificarlo de ninguna forma.

Lo mismo pasa con el bootloader de segunda etapa, el MLO, que no necesita ningún cambio en ningún momento, siempre se utiliza el mismo archivo que viene de origen. La única ocasión en las que conviene cambiar de archivos es cuando la propia Phytec saca una nueva versión actualizada del BSP para la placa.

Por último, el kernel, linuximage, sí que necesita ser modificado cada vez que se se actualiza el sistema de archivos raíz, porque el kernel se encarga del control del hardware y por lo tanto, tiene que actualizarse también o si no, no podrá gestionar el hardware para las nuevas aplicaciones.

3.2.1.2 Instalación De Aplicaciones

Para instalar las aplicaciones necesarias para el correcto funcionamiento de la tarjeta phyCORE hay que usar el menú de configuración de menú de la herramienta ptxdist desde un terminal con el directorio de trabajo en donde se encuentran los archivos que utiliza dicha herramienta, comando "ptxdist menuconfig", lo cual abre el siguiente interfaz.

En esta pantalla se puede navegar con las flechas del teclado por los diversos menús y submenús, que contienen todas las opciones por defecto incluidas en el directorio de ficheros de reglas que venía incluido en el software del kit de desarrollo, y que también se puede descargar de la página del desarrollador de la aplicación.

Así pues, se puede ir pasando por todas las opciones y seleccionar aquellas que nos interesen para nuestra aplicación, y deseleccionando todas aquellas que no queramos. Sin embargo, puede haber distintas utilidades que sean necesarias para el dispositivo diseñado, pero que no aparezcan reflejadas en el menú. Para conseguir que se incluyan es necesario crear dos archivos de reglas, uno con extensión ".in" que hace que lo tenga en cuenta el menú de configuración y otro con extensión ".make" que es el que contiene las instrucciones de preparación, compilación e instalación. Estos archivos tienen que ir en una de las dos carpetas de reglas donde mira la herramienta para que el programa pueda aparecer como opción seleccionable.

Una vez seleccionados todos los archivos hay que ejecutar la orden de compilación de la herramienta, "ptxdist go" y de esta forma comienza el proceso automático por el cual todos los programas y aplicaciones seleccionados se descargan, luego se preparan para su compilación, después se compilan, y por último, se instalan en un sistema de archivos que posteriormente se convertirá en el root filesystem del Linux embebido que se meterá en la tarjeta.

Por otro lado, los archivos ".in" y ".make" no sólo sirven para instalar programas nuevos, sino que también sirve para actualizar las versiones antiguas e incluso, para añadir archivos que no se contemplan de entrada en la instalación de una aplicación.

Así pues, si hay un programa que está muy desactualizado, se puede modificar su archivo de reglas de compilación para que en la primera fase, en vez de descargar el archivo anticuado, busque y descargue una versión más nueva. El otro caso puede resultar muy útil cuando por ejemplo, un programa que se va a instalar necesita un complemento que no se consigue durante la fase de descarga del paquete normal. Para ello, se descarga el complemento a mano y se coloca en una carpeta conocida, después se modifica el fichero ".make" que corresponda

3.2.1.3 Creación De Imágenes

Una vez ejecutada la fase de compilación e instalación, la imagen del kernel está creada, como se ha dicho, pero no la del root fliesystem. Para esto, hay que ejecutar el comando "ptxdist images" que genera la imagen de estos tres archivos, el root filesystem comprimido en tar, root.tgz, el mismo sistema pero en un sistema de archivos UBI, y la imagen física UBI que se debe instalar en la memoria NAND en caso de querer arrancar desde la placa.

Una vez creadas estas imágenes, junto con el BSP, ya se puede instalar el Linux en la memoria de la phyCORE, si se ha elegido esa opción de arranque. De este modo, primero hay que copiar los bootloaders de primera y segunda etapa, barebox y MLO respectivamente, en sus correspondientes regiones de la memoria flash. La siguiente fase es instalar el kernel que se ha creado antes, linuximage, el cual también tiene una parte de la memoria propia.

Este método sin embargo resulta poco útil para la instalación, desde el punto de vista comercial, como se tiene en este trabajo, ya que si hay que producir muchas placas así, se perdería muchísimo tiempo en la instalación a mano porque es un proceso largo y hay que tener cuidado de no borrar lo que no se debe o se puede estropear al punto de no poder arrancarla más
Por eso, el método más práctico, rápido y cómodo es la instalación y arranque desde una tarjeta SD. Para esto, es necesario cambiar la configuración de un selector con varios interruptores en la placa que se encarga de elegir el medio de arranque. Además, es necesario formatear la tarjeta e introducir los archivos de una forma específica, para la cual, Phytec facilita un par de scripts que se encargan del proceso de forma automática sin realizar a penas cambios en ellos más que para ajustarlos al ordenador donde se van a ejecutar.

De esta forma, se pueden crear muchas tarjetas con el sistema operativo ya metido y listo para ejecutar en una placa, en un tiempo muy bajo comparado con el método explicado antes.

4. GATEWAY

4.1 Concepto

El objetivo del proyecto que motiva el diseño del gateway es conseguir un dispositivo autónomo basado en sistemas embebidos para situarlo en la frontera entre una red de sensores genérica, y la red de ordenadores privada del usuario propietario de la red. La función principal del gateway es pues, servir de interfaz entre ambas redes que separa, de forma que concentre los datos generados por los sensores que se comunican inalámbricamente, los guarde en su interior y después los redistribuya a los servidores de la empresa por vía cableada, cuando estos se lo pidan. Para esto, el dispositivo debe ser capaz de recibir los datos por un puerto serie UART, almacenarlos en una base de datos interna y actuar como un servidor poniendo dicha base de datos a disposición del usuario que accede a través de una conexión TCP/IP sobre Ethernet.

4.2 Comunicación serie por UART

La comunicación por el puerto serie es necesaria en la aplicación para la comunicación entre el gateway y la estación de radio de la red de sensores encargada del control de dicha red. El gateway tiene que ejecutar un programa que se encargará de enviar órdenes al master, el cual las atenderá y actuará como convenga, por ejemplo, dando información al gateway o modificando algún parámetro de algún sensor.

A su vez, el gateway tendrá que ser capaz de ponerse a la espera y escuchar en el puerto para poder recibir mensajes de la estación, que pueden ser, datos generados por los sensores y que se envían de forma automática o señales de alarma que puedan saltar y tendrá que discernir el tipo de mensaje que le llega y actuar en consecuencia, ya que no debe comportarse igual con un dato que tiene que almacenar como con una alarma que tiene que transmitir al usuario.

Los puertos serie constan básicamente de 3 hilos, uno para la transmisión de las señales, otro para la recepción y otro de masa, para la tensión de referencia. A la hora de escribir, el programa que quiere acceder al dispositivo serie /dev/ttyO3 (en el caso de la phyCORE) para escribir hace una llamada de sistema al kernel y este controla al driver que se encarga de convertir el código en una señal eléctrica. En el caso de la recepción, el procedimiento es el inverso, el driver controlado por el kernel recibe la señal y traduce los datos. Antes de poder hacer nada, es necesario configurar los drivers de ambos extremos de la conexión para que tengan la misma velocidad o si no, no se podrá transmitir nada.

En primera instancia, para realizar la prueba de funcionamiento de los dos terminales UART libres que presenta la tarjeta utilizada, se usó un convertidor de serie a USB para conectar dicha tarjeta con un ordenador. Desde un hiperterminal en el PC, con el programa minicom, se ajustó la velocidad del puerto serie en el USB a 9600 bps enviados en bloques de 8 y sin control de flujo y se hizo el proceso análogo en la placa, posteriormente, se puso a escuchar a la placa en el puerto serie con el comando “cat” y se escribió en el hiperterminal del PC. Una vez comprobado el funcionamiento, se hizo el proceso a la inversa, se puso a escuchar al ordenador y se escribió en el puerto de la tarjeta, solo que por probar otros métodos, se hizo con el comando “echo”.

Estas pruebas sirvieron para ver que efectivamente la comunicación era posible, pero también había que confirmar que el sistema era capaz de realizar comunicaciones serie de manera autónoma, con un programa corriendo en su interior. Para esto se decidió hacer dos programas sencillos, uno que consistía en configurar el puerto y enviar varios mensajes al PC con cierto retraso entre unos y otros, y otro programa que escuchara en el puerto comenzando por la velocidad más baja y cada vez que recibiera algo, hiciera un eco del mismo y cambiara a la velocidad siguiente, hasta pasar por todas las posibles, probando así además que dichas velocidades son soportadas.

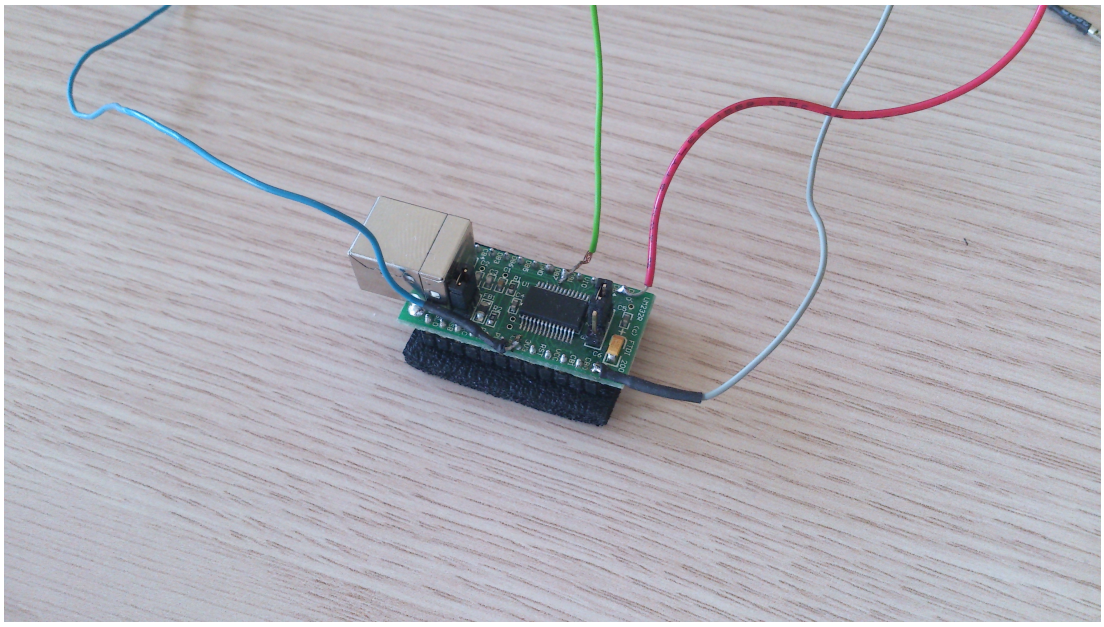


Fig. 5.1 Convertidor UART- USB

Si bien el acceso a los puertos serie es una función a muy bajo nivel, en vez de hacer los programas en lenguaje C se eligió hacerlos en los lenguajes de alto nivel Java y Python por simplicidad, ya que son lenguajes más cómodos para el programador y, aunque en este caso, los programas no son muy complejos, el programa final con el que ha de trabajar sí que lo es, y así se facilita el trabajo futuro. Por otro lado, también resulta muy útil el hecho de que sean lenguajes interpretados, porque así no hace falta utilizar compiladores cruzados.

No obstante, dado que estos lenguajes son de alto nivel, no están pensados para el acceso a puertos serie, así que para poder habilitar esta función para los lenguajes, es necesario añadir unos archivos complementarios, que son RxTx (RXTXcomm.jar y librxtxSerial.so) para Java y pySerial2.7 para Python.

Con esto se confirmó que la tarjeta puede realizar comunicaciones serie autónomas, por lo que el siguiente paso será realizar el programa que se encargue de las comunicaciones reales con la estación de radio de la red de sensores.

4.3 Comunicación TCP/IP por Ethernet

En el caso de la comunicación por Ethernet, el procedimiento es el inverso a por el puerto serie. En este caso, el programa corriendo en el gateway tiene que estar a la espera de posibles instrucciones del usuario, como peticiones de acceso a la información de la base de datos u órdenes para el control de los dispositivos de la red de sensores. Por otro lado, el programa del gateway enviará datos al usuario sin necesidad de que se le hayan pedido, como sucedía en el caso anterior, por motivo de alarma o porque la aplicación requiera el envío de datos automáticamente.

Las conexiones a través de Ethernet se realizan mediante el protocolo TCP/IP que consiste en la transmisión de los datos en paquetes de tamaño fijo, a través de un socket TCP, que es una forma de intercambio de paquetes de forma que el archivo transmitido de un lado del canal se reciba completo y sin errores en el otro lado.

Para esto, el emisor deberá dividir el fichero a transmitir en varias partes según haga falta y después los empaquetará en unas tramas junto con unas cabeceras de datos que llevan información de la transmisión, como, identificación del emisor, identificación del receptor y número de paquete. De esta forma, el receptor es capaz de ordenar los paquetes que recibe de forma numérica y en función de quién los envía, y posteriormente mandarle un mensaje de confirmación al emisor para que éste compruebe que todo el archivo ha llegado correctamente, y en caso contrario, poder reenviar los fragmentos perdidos.

Para comprobar que los puertos Ethernet funcionaban correctamente, la primera y más sencilla prueba fue realizar un acceso remoto a la tarjeta desde el ordenador mediante “ssh”. De esta forma, al conseguir acceder a la tarjeta a través de Ethernet, se comprobó que el puerto recibía y transmitía datos adecuadamente, pues si no los hubiera recibido bien, no se habría podido controlar remotamente, y si no los hubiera enviado, no se habría visto nada en la consola de comandos desde la que se realizó el acceso.

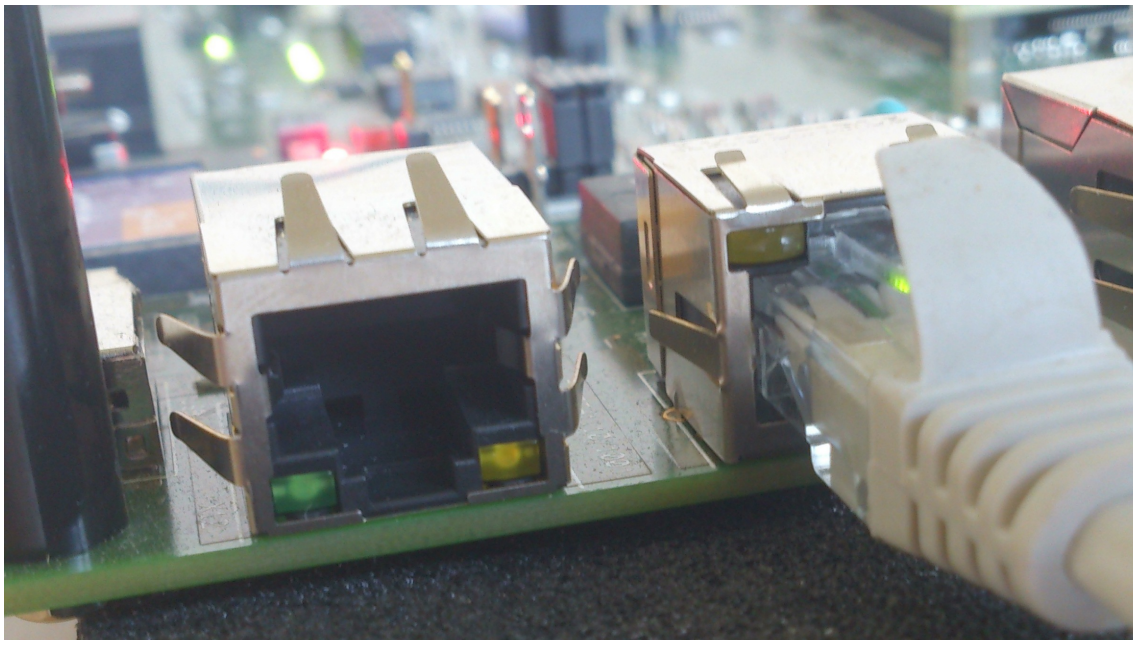


Fig. 5.2 Conector Ethernet

Seguidamente, al igual que en el caso de los puertos serie, se realizaron un par de programas de servidor TCP/IP para Java y Python. Los programas consistían simplemente en poner a escuchar al sistema en un puerto TCP de la dirección de uno de sus terminales de Ethernet y, cuando le llegara algún texto ahí, el programa se lo devolviera a quien lo había enviado, pero con otro texto añadido.

Como las conexiones de red son de más alto nivel que las conexiones por puerto serie, y su uso es más común, en este caso no hubo que añadir ningún complemento a las librerías de Java y Python, porque traían todo lo necesario de origen.

Así pues, se comprobó que el gateway es capaz de comunicarse con un ordenador a través de Ethernet, por lo que el siguiente paso fue introducir el programa de protocolo de comunicación por TCP/IP desarrollado por Kunak y cuyas pruebas se muestran más adelante.

4.4 Base de datos

La base de datos es la etapa intermedia en el sistema del gateway, se encarga de almacenar en tablas los datos recibidos por el puerto serie desde la estación de la red y luego de mostrárselos al usuario. Dado que se utilizan unos sistemas embebidos de capacidad limitada, la base de datos contendrá registros de la últimas 24 horas, además del núcleo de información necesario con la información de los equipos conectados.

Para esta función se eligió utilizar MySQL dado que es la BD que se usa en el sistema actual previo a la inclusión del gateway. MySQL es una base de datos relacional que trabaja con un sencillo lenguaje de peticiones, como “insert” y “select” para escribir y leer datos respectivamente. Así pues, los programas antes mencionados que se encargarán de las comunicaciones con los elementos conectados al gateway tendrán que hacer uso de estos comandos para trabajar con los datos y las tablas, de modo que el que escucha al UART escriba en la BD y el asociado al TCP será el que envíe los datos bajo demanda a las estaciones de control, que serán las que tengan la potencia suficiente para actuar como servidor interno.

Para enviar la información se ha elegido el protocolo HTTP mediante el uso de web services rest que corren sobre un servidor Tomcat, dado que se ha utilizado el lenguaje java como core de desarrollo.

Para la prueba de MySQL se ha hecho uso del GUI MySQL Workbench que permite visualizar el contenido de la base de datos y el estado del servidor, así como los propios web services llamados desde un navegador de internet como alternativa más inmediata, pudiendo observar la salida en formato XML.

4.5 Pruebas

4.5.1 Escenario

Web services: REST, con autenticación (user, password) y cifrado (https)

DB server: MySQL
TCP server: Programado en java, con un hilo por conexión. Se envía un paquete por cada serial number (el serial number sería como la IP del protocolo de Kunak).

Se parte de un escenario inicial de 20 equipos transmitiendo, hacia el gateway, a lo sumo una vez por minuto y con dos servidores internos de la empresa haciendo peticiones a dicho gateway.

A nivel de web services se simularán 2 hilos que equivalen a 2 interfaces Ethernet (por motivos de redundancia) pidiendo información a la gateway

4.5.2 Herramientas

Se ha utilizado un SW propio para emular varios nodos enviando lecturas en paralelo con delays entre lectura y lectura de cada nodo de 1 segundo, de manera que nos aseguramos que funcionará bien en un entorno real donde se darán como máximo aproximadamente un ciclo de lectura por minuto y por equipo.

Para medir el rendimiento (tiempo de respuesta) de los web services se ha utilizado el sw jmeter.

4.5.3 Resultados servidor TCP

Para hacer esta prueba se han emulado 50 equipos, mandando lecturas de 2 sensores 1 vez por segundo al gateway, un sobredimensionamiento mucho mayor que la peor situación real posible para probar los límites del equipo.

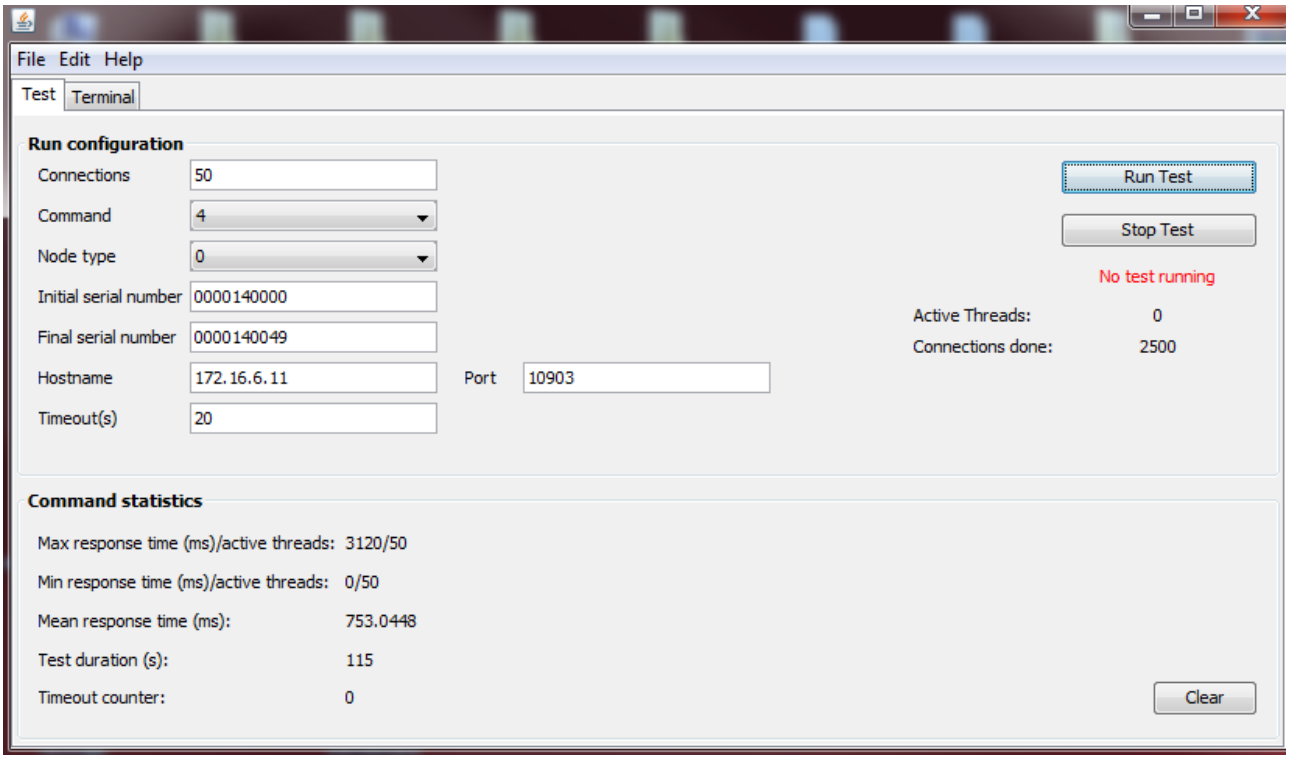


Fig. 5.3 Prueba TCP 1

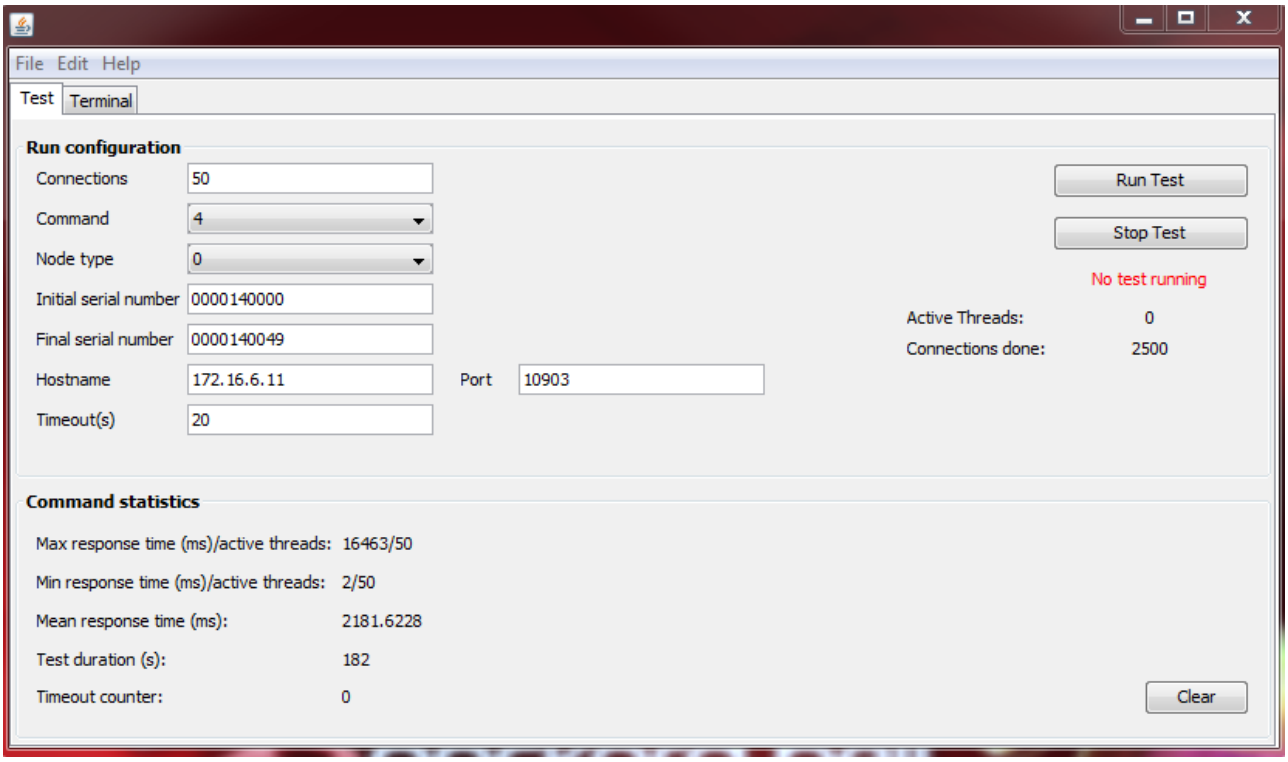


Fig. 5.4 Prueba TCP 2

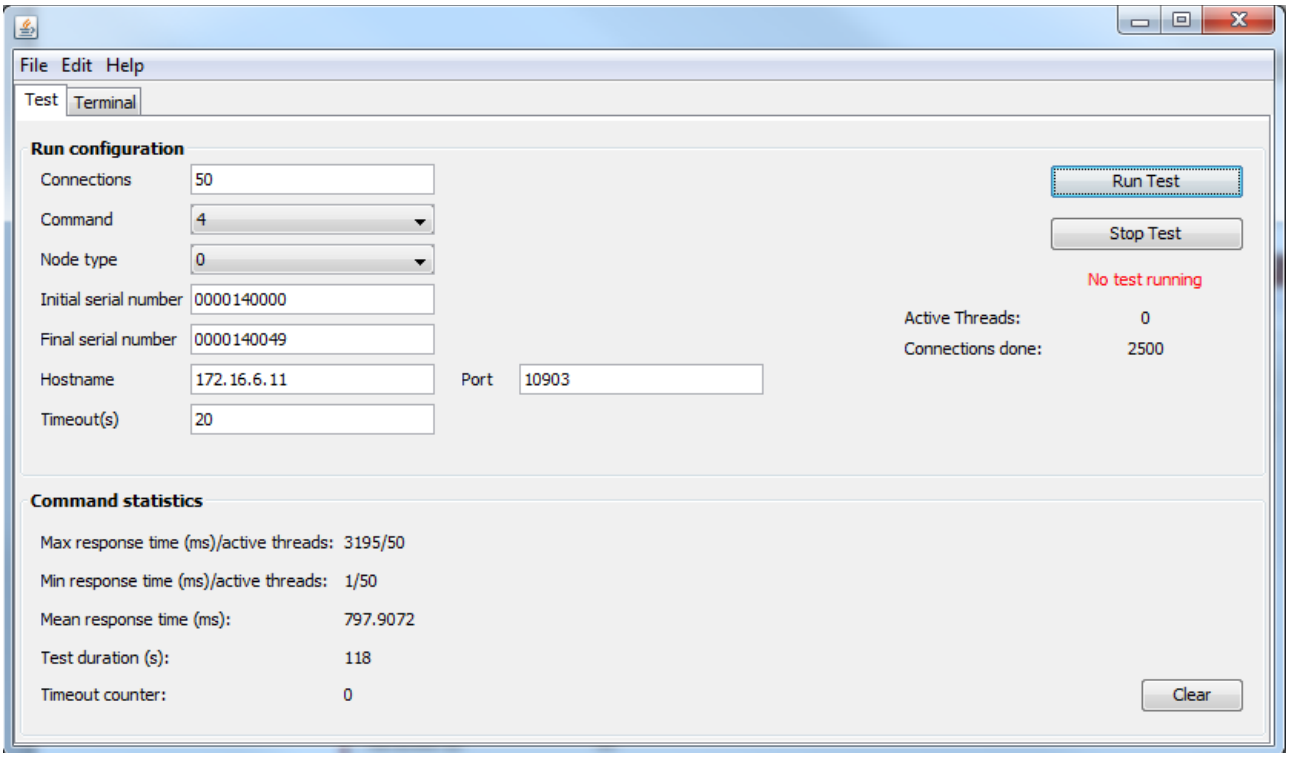


Fig. 5.5 Prueba TCP 3

Como se puede apreciar en las figuras anteriores, el gateway ha sido capaz de aguantar 50 equipos en paralelo manteniendo unos retardos de respuesta aceptables. También se puede ver que no ha habido ninguna pérdida de hilos o lecturas durante el proceso, como consecuencia de limitaciones propias, del procesador, la memoria o la base de datos.

4.5.4 Pruebas de Web Services

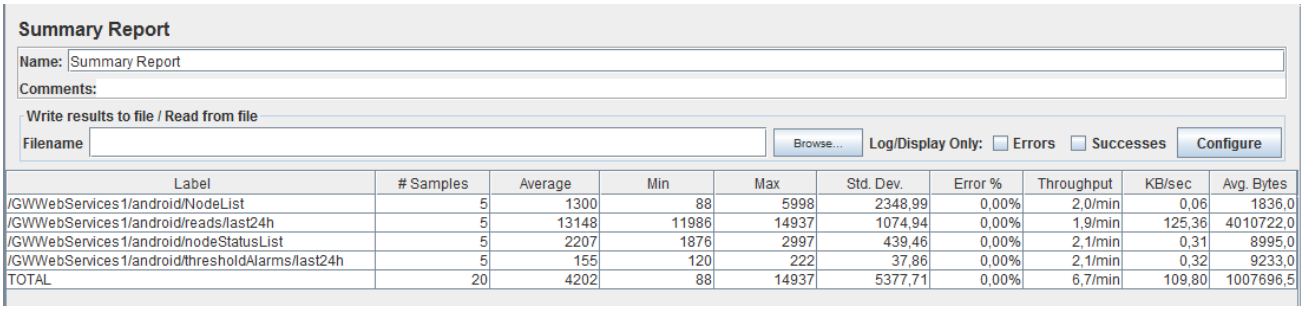
Cada usuario pedirá 4 web services diferentes:

- Lecturas últimas 24 h
- Alarmas últimas 24 h
- Estado actual de nodo (alarmas activas)
- Lista de nodos

Los web services piden información de todos los nodos. No filtran, luego serán pesados, sobre todo el de lecturas ya que se han hecho las pruebas con unas 58000 lecturas (2 sensores siendo leídos una vez por minuto, por cada nodo, en 24 horas)

Los resultados son los siguientes:

Un hilo, lecturas 20 equipos



Summary Report

Name: Summary Report

Comments:

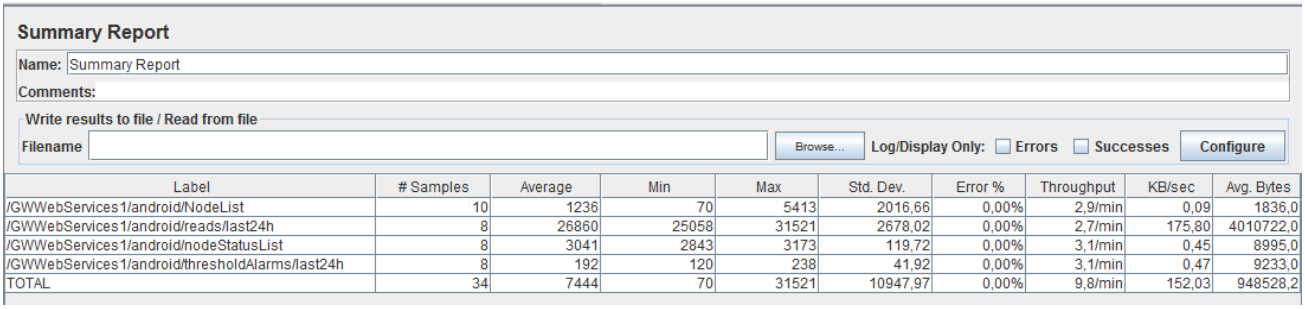
Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
/GWWebServices1/android/NodeList	5	1300	88	5998	2348,99	0,00%	2,0/min	0,06	1836,0
/GWWebServices1/android/reads/last24h	5	13148	11986	14937	1074,94	0,00%	1,9/min	125,36	4010722,0
/GWWebServices1/android/nodeStatusList	5	2207	1876	2997	439,46	0,00%	2,1/min	0,31	8995,0
/GWWebServices1/android/thresholdAlarms/last24h	5	155	120	222	37,86	0,00%	2,1/min	0,32	9233,0
TOTAL	20	4202	88	14937	5377,71	0,00%	6,7/min	109,80	1007696,5

Fig. 5.6 1 hilo, 20 equipos

Dos hilos, lecturas 20 equipos



Summary Report

Name: Summary Report

Comments:

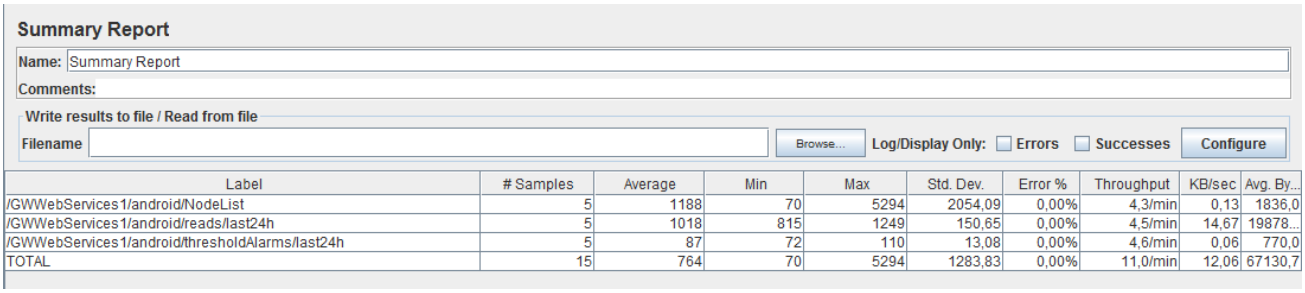
Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
/GWWebServices1/android/NodeList	10	1236	70	5413	2016,66	0,00%	2,9/min	0,09	1836,0
/GWWebServices1/android/reads/last24h	8	26860	25058	31521	2678,02	0,00%	2,7/min	175,80	4010722,0
/GWWebServices1/android/nodeStatusList	8	3041	2843	3173	119,72	0,00%	3,1/min	0,45	8995,0
/GWWebServices1/android/thresholdAlarms/last24h	8	192	120	238	41,92	0,00%	3,1/min	0,47	9233,0
TOTAL	34	7444	70	31521	10947,97	0,00%	9,8/min	152,03	948528,2

Fig. 5.7 2 hilos, 20 equipos

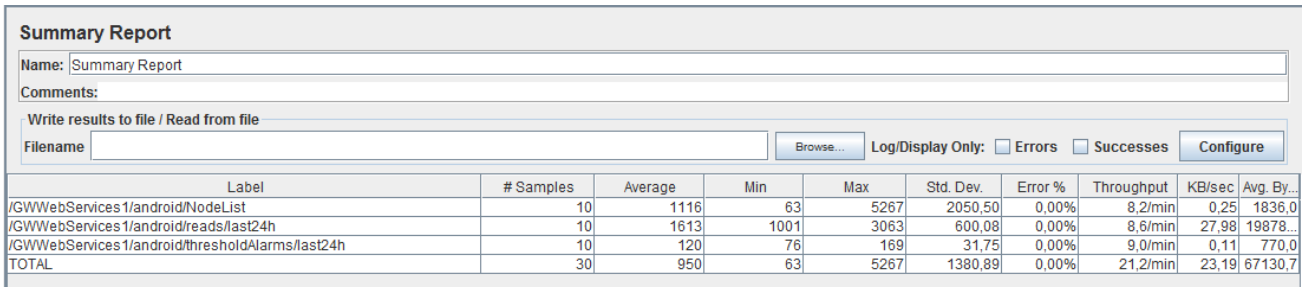
Un hilo, información filtrada para un solo equipo



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. By...
/GWWebServices1/android/NodeList	5	1188	70	5294	2054,09	0,00%	4,3/min	0,13	1836,0
/GWWebServices1/android/reads/last24h	5	1018	815	1249	150,65	0,00%	4,5/min	14,67	19878...
/GWWebServices1/android/thresholdAlarms/last24h	5	87	72	110	13,08	0,00%	4,6/min	0,06	770,0
TOTAL	15	764	70	5294	1283,83	0,00%	11,0/min	12,06	67130,7

Fig. 5.8 1 hilo, 1 equipo

Dos hilos, información filtrada para un solo equipo



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. By...
/GWWebServices1/android/NodeList	10	1116	63	5267	2050,50	0,00%	8,2/min	0,25	1836,0
/GWWebServices1/android/reads/last24h	10	1613	1001	3063	600,08	0,00%	8,6/min	27,98	19878...
/GWWebServices1/android/thresholdAlarms/last24h	10	120	76	169	31,75	0,00%	9,0/min	0,11	770,0
TOTAL	30	950	63	5267	1380,89	0,00%	21,2/min	23,19	67130,7

Fig. 5.9 2 hilos, 1 equipo

Obviamente el rendimiento empeora cuando se tienen más hilos haciendo peticiones a los web services y cuando dichas peticiones son sin filtrar, es decir, piden información de todos los equipos. Cuando se pide información filtrada por equipos, los tiempos de respuesta mejoran hasta reducirse a unos 2-3 para las lecturas acumuladas por un nodo al final del día.

4.5.5 Conclusiones de las pruebas

Como las pruebas se han realizado para la parte TCP/Ethernet y no se ha podido comprobar el rendimiento de las comunicaciones serie, es de esperar unos resultados muy similares ya que la arquitectura serie y TCP (sockets) difieren únicamente en la fase de conexión, pues no hay un proceso de saludo, ni se puede tener múltiples conexiones desde diferentes equipos al puerto serie.

5. CONCLUSIÓN Y LÍNEAS FUTURAS

Visto todo lo anterior y teniendo en cuenta las características descritas, la tarjeta de Phytec es una buena solución, ya que con la distribución hecha con las herramientas que ellos proporcionan permite un diseño muy ajustado a las necesidades del producto, sin derrochar capacidades, y con un consumo bastante reducido, cumpliendo, además, con todo el resto de prestaciones de una forma completamente aceptable por el momento.

Si bien lo anterior es cierto, también lo es que por los problemas acaecidos a la hora de compilar el kernel a causa de las carencias de las herramientas de Pengutronix, algunos de los programas instalados se encuentran en versiones más o menos antiguas sin poder actualizarse, por lo que es posible que si se intenta cambiar algo del sistema, como añadir alguna función o modificar algún protocolo, la pasarela deje de funcionar correctamente y no se pueda actualizar.

De esta forma, resulta muy recomendable probar alguna otra placa que sea capaz de funcionar con una distribución como Ubuntu, ya que con un comportamiento similar a un ordenador, es probable que la gestión de archivos y los procesos de instalación sean mucho más sencillos y rápidos, consiguiendo por otra parte versiones más nuevas y probadas.

Debido a la limitación de tiempo y recursos, además de la ralentización por los problemas encontrados, hay varias ideas que se han quedado en el tintero y que habrá que probar e implementar en el futuro.

En primer lugar, falta por diseñar el juego de programas que tienen que correr en la tarjeta para dotarla de las funcionalidades de gateway que se han tratado en el proyecto.

También queda contemplar el control elemento capaz de realizar comunicaciones inalámbricas WiFi o 3G/4G conectado al puerto USB de la tarjeta, para ofrecer más posibilidades de conectividad, o capacidad de conexión a internet por medio inalámbrico si la conexión cableada sobre ethernet es directa a un ordenador en vez de a un switch o a un router que lo permita.

Además, como se ha dicho, habría que probar otro tipo de placas como la del ejemplo, para estudiar su comportamiento y comparar con la otra, o por ejemplo, para la prueba rápida de nuevas ideas, antes de hacer la versión definitiva en una tarjeta con el kernel hecho desde cero como la de Phytec.

Por último queda también pendiente la elección de las fuentes de alimentación, así como del soporte físico que lo contenga todo, de cara a obtener el producto final.

6. BIBLIOGRAFÍA

6.1 Enlaces Web

http://www.kunak.es/
http://en.wikipedia.org/
http://es.wikipedia.org/
http://www.phytec.eu/
http://www.pengutronix.de/
http://www.openembedded.org/
http://www.yoctoproject.org/
http://www.microsoft.com/
http://www.qnx.com/
http://www.xenomai.com/
http://www.ti.com/
http://www.samsung.com/
http://www.freescale.com
http://broadcom.com/
http://linuxfoundation.com/
http://www.wandboard.org/
http://www.iearobotics.com/

6.2 Libros y presentaciones

Building Embedded Linux Systems; Yaghmour, Masters, Ben-Yossef & Gerum - O'Reilly 2008
EMbedded Linux Primer: A Practical, Real-World Approach; Hallinan - Prentice Hall 2006
Linux para sistemas embebidos; Chiesa, De Andrés - Universidad de Buenos Aires 2010
Linux Sin Vértigo; Zabalza - Cemitec
Workshop Linux Embebido; Chiesa, De Andrés, Bassi - SASE/FIUBA
Embedded Linux system development; Free Electrons - Free Electrons 2014