UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

CONCENTRADOR DE MEDICIONES CLÍNICAS EN HOSPITALES

Carlos Andradas Martínez

Enero 2015

CONCENTRADOR DE MEDICIONES CLÍNICAS EN HOSPITALES

AUTOR: Carlos Andradas Martínez

TUTOR: Guillermo González de Rivera Peces

HCTLab

Dpto. de Tecnología Electrónica y Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Enero 2015

A todos los que me ofrecieron su apoyo

Especialmente a mis padres

Agradecimientos

Como no, este punto de mi vida no se ha generado sin más. Ha sido producto de grandes esfuerzos por mi parte pero también por parte de otras personas cuyo interés en mis progresos académicos han servido siempre como empuje durante el día a día desde hace ya unos cuantos años.

La principal dedicación que tiene este proyecto se centra en mis padres. Incansablemente pesados desde el día en que pisé por primera vez un colegio con la idea de que estar entre los mejores de clase se basaba sencillamente en dedicar un pequeño esfuerzo diario estudiando, consiguieron generar en mí ese afán de superación responsable de que, tarde o temprano, haya logrado concluir una carrera como esta ingeniería de telecomunicación con el correspondiente proyecto que se expone en el presente documento. Por todo el respaldo que me habéis aportado todo este tiempo, gracias.

Estos logros de los que hablo, por supuesto no habrían tenido cabida de no ser por el profesorado que me ha asistido durante todo este tramo de mi vida. Principalmente, me siento obligado a admitir que en la gran mayoría de los casos me he sentido muy agradecido por la ayuda y la atención que he obtenido por parte de los profesores de este escuela. Centrándome más en este proyecto quiero agradecer a mi tutor, Guillermo González de Rivera, la oportunidad, la sensación de cercanía y los consejos, así como el buen ambiente que siempre ha creado durante el transcurso del proyecto. No en menor medida, también quiero mencionar a mi "segundo tutor" Eduardo Boemo, por todos los momentos en los que me he sentido más desenganchado del proyecto, en los cuales influyó para mejorar mi rendimiento, y por todo el material aportado que me sirvió de tanta ayuda. Entre los dos generaron, por un lado un ambiente cercano en el que no faltaron las risas y por el otro lado un nivel de exigencia adecuado para que todo el transcurso del proyecto fuese propicio.

De una manera más indirecta pero no menos importante también han influido terceras personas, como mis compañeros de clase que siempre aportaron fluidez al día a día y que en tantas ocasiones me han echado una mano en los momentos en los que me quedé rezagado. A Cecilia, Guillermo, Juan Pablo, Gonzalo y Diego han ido dedicadas principalmente estas líneas. Del mismo modo, sin salir de la universidad, también creo necesario mencionar a todas las personas de la asociación de estudiante AET que generaron momentos tan épicos e inolvidables, con especial mención a Alex. Por último también mencionar a mis compañeros de primero que por unos motivos u otros no continuaron más adelante, pero que fuera de la universidad también sirvieron de apoyo, como Siman o Eloy.

Fuera de la universidad, a mi hermana y a mis amigos más cercanos, principalmente a los que forman parte de mi vecindario, capitaneados por Juan y Sergio y seguidos de cerca por toda esa gentuza que tantos buenos ratos me ha hecho pasar. Sin vuestra forma de ser no sería el mismo, y eso influye en absolutamente todo. De la misma forma y dedicándoles el mismo párrafo, merecen una mención de honor Alex y José, los cuales han estado presentes en mi vida casi desde el principio y que ahora, de la misma manera que muchas personas ya mencionadas, me sirven de referente. También quiero mencionar a David y a Miguel por todas las aportaciones léxicas que me han cedido entre la gran cantidad de tonterías que han llegado a ser capaces de decir.

También considero éste un buen apartado para nombrar personas que siempre he tenido de referentes. Por mi admiración quiero mencionar a mis abuelos y a la gran mayoría de mis tíos, al igual que ya mencioné a mis padres, por todo el ejemplo de trabajo y sacrificio que de maneras tan diversas me han ido mostrando a lo largo de mi vida, en especial a mis abuelos Carlos y Ángel, y a mis tíos Carlos (ambos), Fernando, José, Alberto, Margarita y a mi difunto tío Miguel Ángel. Mis abuelas Conchita y Bea también aportaron grandes dosis de ánimo y, en ocasiones, lugares idóneos para el estudio.

Como mezcla de un poco de admiración, y un poco de grandes momentos juntos, también quiero aprovechar para acordarme de la gran mayoría de mis primos.

Si bien es verdad que el apoyo de las personas más cercanas influye, y mucho, a la hora de imponerse metas y de lograr la motivación suficiente para conseguirlas, también opino que es digno de mención acordarse de las personas que intentaron de algún modo producir el efecto contrario. Especial mención tienen también en este apartado de agradecimientos las personas que, aunque sólo fuese a través de comentarios esporádicos, no creyeron en mí. No mencionaré los nombres y apellidos de principalmente aquellos profesores de instituto (y un jefe concreto de catering) que desde mi punto de vista mostraron ese ego tan absurdamente alto a lo largo del periodo que compartí con ellos, pero sí indicaré que acordarme de ellos ha servido para ganar la motivación que en tantas ocasiones me ha faltado por las circunstancias que en esos momentos estuvieran teniendo lugar.

Por último quiero añadir que en este momento, también tengo muchos nombres en mi mente que por diversos motivos no quiero mencionar pero que también han influido enormemente en este periodo de tiempo, especialmente a C y a L, aunque existen una infinidad más.

Porque nadie sería nada sin la influencia, cariño y ayuda de los que le rodean, muchas gracias a todos. Espero que esto no sea lo último que os dedique.

Resumen

Resumen

El presente proyecto desarrolla una plataforma diseñada específicamente para su uso en hospitales y entornos biomédicos. Su propósito general es el de facilitar la toma de medidas de constantes vitales a los pacientes en dichos tipos de entornos con la finalidad de simplificar el trabajo del personal médico. Dicha plataforma, además de facilitar el trabajo, tiene como objetivo reducir los errores durante el manejo de datos, principalmente humanos, que puedan darse en la toma, gestión y envío de los datos. Para ello se presenta un sistema de tratamiento de datos, desensamblado, envío y automatización capaz de realizar todas las gestiones referentes a los datos de una manera autónoma, rápida y eficiente que presente a su vez una interfaz de usuario clara e intuitiva.

Dicho sistema a su vez, está basado en la normativa de dispositivos electrónicos en entornos biomédicos tales como hospitales o centros de salud, concretamente en una zona concreta del sistema completo a las que las normas se refieren.

Por último, el sistema contendrá un sistema de alimentación independiente formado por una batería recargable que aporte autonomía durante su uso. Este aspecto presenta una gran relevancia ya que ofrecerá la posibilidad de que el dispositivo pueda ser usado por parte del personal médico sin necesidad de apagarlo e iniciarlo cada vez que pueda ser usado, generando así una mayor versatilidad en la toma de las constantes vitales de varios pacientes consecutivamente.

Palabras clave

Sistema embebido, circuito de propósito general, Linux embebido, programación, USB, programación orientada a objetos, normativa, estandarización, baterías.

Abstract

Abstract

The objetive of this project is to develop a system whose main purpose is manage all the activities related to take measures of the vital signals of the patients in a biomedical environement, processing the data obtained, store it and send it if necessary. The Project has been divided into three parts. The most importante part has been the design and implementation of a device which is able to carry on with all the activities mentioned before. In the second part of the project (wich have been developed in first place) it is been collected a lot of information about the standars which are followed in the implementation of any health care device. The objetive of this part is get the enough knowledgement about this regulations thinking in the possibilities of a future implementation in a health care environment. The last part has as aim goal giving a special skill to the device, the autonomy. This autonomy will give a better performance to the device and to the general system, and it will be reached adding a battery. In this way, the battery will act as a power supply of the device.

Keywords

Embedded system, general purpose system, embedded Linux, programming, USB, objects oriented programming, regulations, standarization, batteries.

Índice general

Bloque I. Capítulos

Índice de figuras

.

Índice de tablas

PARTE I

Capítulos

Capítulo 1

Introducción

1.1. Motivación

Los centros de salud y hospitales cuentan con una plataforma tecnológica de comunicaciones suficiente para aumentar y mejorar la automatización de las labores básicas del personal de dichas zonas de trabajo pero aún se está muy lejos de conseguir un uso eficiente de dicha plataforma. Para aumentar la funcionalidad del sistema se le desea añadir un dispositivo que sea capaz de proporcionar servicios al paciente como televisión, Internet o telefonía y que, a su vez, pueda servir de interfaz para el monitoreado de las constantes vitales. Éstas deberán ser provistas por un dispositivo capaz de recibir las medidas de dichas constantes desde los distintos sensores encargados de obtenerlas.

1.2. Objetivos

El objetivo de este proyecto es llevar a cabo el diseño y el desarrollo del dispositivo que se encargará de recoger las medidas de los sensores biométricos.

Como objetivos intermedios se presentarán una serie de hitos que marcarán las diferentes fases del desarrollo del proyecto. En primer lugar se realizará un estudio sobre la normativa referente al uso de dispositivos electrónicos dentro de entornos médicos. Otra tarea que podría ser llevada a cabo al mismo tiempo que la anterior consistirá en realizar un estudio sobre el estado actual de los sistemas embebidos así como de los sistemas operativos para poder elegir cuáles podrían ser más convenientes para

desarrollar el proyecto de la manera más producente posible.

A continuación, se procederá a la programación de una aplicación capaz de interactuar correctamente con los distintos sensores que puedan ser usados por parte del personal médico del hospital. Dicho programa deberá realizar un uso responsable de los recursos del sistema embebido que se haya elegido para realizar el proyecto. Para el desarrollo de este software será imprescindible alcanzar otro objetivo que consistirá en determinar los protocolos de comunicación de los sensores proporcionados.

En última instancia se montará el dispositivo completo y se realizará un estudio sobre el consumo eléctrico del dispositivo finalmente desarrollado, así como el de los sensores proporcionados con el fin de elegir una batería capaz de aportar la autonomía que el dispositivo necesita para poder llevar a cabo sus funciones de la manera más cómoda posible. El montaje final del Concentrador de Señales (CS) incluirá la adhesión de la batería mencionada.

1.3. Organización de la memoria

El presente documento se estructura en dos bloques: los capítulos y los apéndices. En los capítulos se detalla todo el proceso llevado a cabo durante el desarrollo del proyecto mientras que los apéndices se han reservado para añadir aspectos más técnicos o referencias a características o imágenes de fuentes externas.

Los capítulos se han organizado siguiendo como principales pautas los objetivos mencionados anteriormente. En primer lugar se expone el estudio sobre el estado del arte de los dispositivos y regulaciones que conciernen a este proyecto. A continuación se detallan todos los aspectos referentes al diseño del sistema y a su desarrollo, exponiendo los puntos más relevantes así como los problemas surgidos durante el proceso. En último lugar, se hablará del montaje final y de las conclusiones sacadas durante el proyecto para concluir finalmente sugiriendo una serie de posibles pautas para continuar o mejorar el trabajo en el futuro.

Los apéndices incluyen una serie de aspectos más técnicos del sistema, tales como las especificaciones de los elementos del sistema o el manual de uso de éste. También en los apéndices se ha añadido información complementaria como los detalles de las herramientas usadas para la realización del proyecto (entorno Linux, Matlab, compilador *gcc*, analizador Agilent) o detalles sobre el código desarrollado a lo largo del proyecto. En los anexos también se añaden aspectos legales como el presupuesto del proyecto o el pliego de condiciones de éste. Por último, se añade información adicional referente a la propia memoria, más concretamente un glosario en el que aparecen los acrónimos y siglas presentes en los capítulos con su significado y la bibliografía.

Capítulo 2

Estado del arte

2.1. Introducción

La finalidad de este capítulo es exponer el estado actual de los dispositivos electrónicos, protocolos de comunicación y sistemas operativos que puedan resultar de interés en el desarrollo de este proyecto.

En primer lugar se habla brevemente sobre las principales interfaces de conexión entre los sensores y el sistema desarrollado (CS) haciendo especial hincapié en la interfaz USB, que será la que finalmente se use.

Seguidamente, se expondrá el estado de los sistemas embebidos, Tablet y baterías en el mercado actual, detallando los modelos comerciales que han sido tenidos en cuenta durante el desarrollo del proyecto.

Todo el software correrá a cargo de un sistema operativo, para el cual también será interesante realizar un estudio sobre el estado del arte de cara a elegir el que mejor se adapte a las necesidades del presente proyecto.

Por último se mencionarán y caracterizarán una serie de dispositivos similares al que se pretende diseñar en este proyecto, presentes en el mercado actual, con la finalidad de presentar un dispositivo que tenga al menos las mismas prestaciones que las que se ofertan actualmente. Este apartado, desde un punto de vista comercial, será especialmente importante para poder realizar una comparativa entre el dispositivo desarrollado y la oferta del mercado al finalizar su elaboración, con el fin de poder prever su demanda y aceptación.

2.2. Interfaces de conexión

En electrónica, se entiende como interfaz de conexión a la conexión física y funcional entre dos sistemas o dispositivos de cualquier tipo capaz de ofrecer una comunicación entre distintos niveles.

Puede ser llevada a cabo por medio de conexiones inalámbricas o por medio de conexiones cableadas. Dentro del primer grupo destacan Bluetooth, ZigBee y WiFi (por medio de un punto de acceso a la red inalámbrica) mientras que del segundo grupo destacan USB, el puerto serie y el puerto paralelo.

2.2.1. Conexiones inalámbricas

Son aquellas que no necesitan de un medio cableado para transmitir información, sino que lo hacen a través de ondas electromagnéticas. Dependiendo de la frecuencia que utilicen para realizar los envíos presentarán unas prestaciones u otras. Su principal ventaja reside en el ahorro económico y comodidad al no requerir de ninguna superficie de transmisión eléctrica, mientras que su principal desventaja reside en que requieren de una seguridad mayor para evitar el posible robo de datos.

- Bluetooth

Es una tecnología creada por las empresas Ericsson, IBM, Intel, Toshiba y Nokia en 1994 con la finalidad de elaborar un reemplazo del cable. Se basa en la tecnología de saltos de frecuencia de amplio espectro.

Bluetooth cuenta con un protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance y basados en transceptores de bajo costo.

Debido a su naturaleza inalámbrica, su arquitectura hardware incluye un dispositivo de radio, así como un controlador digital compuesto de una CPU y de unas interfaces con el dispositivo anfitrión, encargado de procesar la banda base y del manejo de los protocolos de capa física.

En sus características técnicas se define un canal de comunicación máximo a un máximo de 720 Kbit/s con un rango óptimo de 10 metros. Opera en la frecuencia de 2,4 a 2,48 GHz con saltos de frecuencia con posibilidad de transmitir en Full Duplex. Sin embargo, algunas características varían entre las diferentes versiones de su especificación (versión v1.1 de 2002 hasta v4.0 de 2010).

Sus principales aplicaciones residen en entornos donde pueda haber dos o más dispositivos en un área reducida sin grandes necesidades de ancho de banda. Como

ejemplos se mencionan los teléfonos móviles, impresoras, módems, auriculares o mandos de consolas.

❖ ZigBee

ZigBee es el nombre que recibe la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica. Su finalidad es ser utilizada con radiodifusión digital de bajo consumo en aplicaciones que requieran comunicaciones seguras con baja tasa de envío de datos.

ZigBee trabaja sobre las bandas ISM, definiendo hasta 16 canales en el rango de 2,4 GHz permitiendo tasas de transmisión en el aire de hasta 250 Kbit/s con rango de 10 metros hasta 75 metros.

Al igual que Bluetooth, su arquitectura hardware se compone de un transmisor de radio y de un microcontrolador capaces de funcionar en márgenes muy pequeños de consumo y de ancho de banda.

Si bien es verdad que sus aplicaciones podrían ser las mismas que las de la tecnología Bluetooth, las diferencias que existen entre ellas las limitan a aplicaciones más concretas.

- ZigBee presenta un menor consumo y un menor coste que Bluetooth, pese a que trabaja con un rendimiento menor ya que permite una tasa de transferencia de datos menor y se queda dormido durante los periodos de inactividad.
- Bluetooth, pese a tener un coste mayor y un consumo mayor, trabaja con un mayor rendimiento, permitiendo tasas de transferencia de datos mucho mayores (del orden de 250 vs 3000 kbit/s).

Por tanto, mientras que Bluetooth se ha extendido más en dispositivos como móviles y ordenadores, ZigBee y demostrado ser más eficaz en dispositivos con transferencia de datos menores tales como sensores médicos, artículos de juguetería o domótica.

❖ WiFi

WiFi (Wireless Fidelity) es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Surge de la creación WECA (Wireless Ethernet Compatibility Alliance) en 1999 por las empresas 3Com, Airones, Intersil, Lucent Tchnologies, Nokia y Symbol Technologies con la finalidad de asegurar la compatibilidad inalámbrica de distintos dispositivos, sean del fabricante que sean.

Su principal aplicación es aportar un punto de acceso a Internet a partir de dispositivos de acceso como los routers. Por medio de un despliegue de estos puntos de acceso, se puede elaborar una red que sea capaz de proveer un servicio de transferencias de datos muy eficaz.

Las principales ventajas es la comodidad que aporta cualquier tecnología inalámbrica, la universalidad en cuanto a su compatibilidad con cualquier dispositivo y las tasas de transferencia. Sin embargo, también presenta serias desventajas, entre las que destacan la necesidad de instalar una red de puntos de acceso, la falta de seguridad en cuanto a la privacidad de las transferencias de datos y la mayor lentitud que presenta frente a las tecnologías cableadas.

2.2.2. Conexiones cableadas

Como su nombre indica, son aquellas que requieren de un medio cableado para realizar transferencias de datos. Pese a la desventaja que supone tener que conectar y desconectar los medios de transmisión que se requieran, las interfaces de conexión cableadas presentan una serie de ventajas muy importantes frente a las inalámbricas, entre las que destacan una mayor tasa de envío de datos (debido a la mínima atenuación que experimentan y a la reducida presencia de interferencias) y una mayor seguridad y privacidad en las transferencias, haciendo innecesaria la encriptación de datos a nivel general.

❖ Puerto Serie

El puerto serie es una interfaz de comunicaciones en el que la información es transmitida enviando un solo bit a la vez, en contraste con el puerto paralelo. Normalmente, el término “puerto serie” identifica el hardware conforme al estándar RS-232.

Fue ampliamente utilizado en todo tipo de computadoras aunque, por motivos de compatibilidad y rendimiento, en la actualidad se considera un puerto obsoleto en favor del puerto USB.

No obstante continúa teniendo aplicación en máquinas industriales, siguiendo el estándar RS-422.

❖ Puerto Paralelo

El puerto paralelo es otra interfaz de comunicaciones con la peculiaridad de que en este puerto los bits se transmiten a la vez, enviándose mediante paquetes de byte.

Los puertos paralelos más usados son los puertos de tipo ATA (IDE, P-ATA) y los puertos SCSI que siguen usándose en la actualidad para interconexión de dispositivos internos de un ordenador, tales como discos duros.

❖ USB

USB (Universal Serial Bus) es un estándar industrial desarrollado a mediados de los años noventa por las empresas Intel, IBM, Northern Telecom, Compaq, Microsoft,

DEC y NEC. Define los cables, conectores y protocolos usados en un bus serie para comunicar y proveer alimentación entre ordenadores y periféricos.

En la actualidad ha reemplazado a casi todos los puertos usados con anterioridad y es desarrollado por más de 685 compañías debido a su universalidad (se encuentra disponible en prácticamente todos los dispositivos electrónicos) y a su alto rendimiento (la especificación 3.0 puede llegar a alcanzar velocidades de 5 Gb/s).

El conector dispone de 4 puertos, dos dedicados a envíos y recepciones de datos, otro con la alimentación de 5 V y otro conectado a tierra. Estos dos últimos son los encargados de proveer alimentación a los periféricos. Los periféricos que requieran mayor alimentación suelen tener una entrada adicional de alimentación.

Las transferencias de datos se realizan a través de un buffers de datos llamados *endpoints*, que disponen de una dirección en la que se especifica el identificador, el sentido del envío de datos, el tipo de envío y el tamaño máximo de la trama a enviar. Cada transmisión de datos se realiza en tres fases: inicialización (token), donde se configuran los parámetros de la transmisión; envío (data), donde se realiza el envío de datos; y confirmación (handshake), donde se reportan errores de transmisión o éxito en el envío. La última fase es posible gracias a que los paquetes de datos se generan con un identificador de paquete y unos bits de corrección de errores.

El encargado de gestionar toda la conexión y las transferencias es el *host* USB, que suelen ser ordenadores o sistemas embebidos. Los dispositivos periféricos sólo se encargan de atender las peticiones que les realiza el *host*. La información sobre el dispositivo conectado es de vital importancia, puesto que dependiendo del dispositivo la conexión se gestionará de una manera concreta o de otra (se cargarán unos *drivers* u otros). Esta información es proporcionada por el propio dispositivo a través de los descriptores, y a este proceso se le denomina enumeración.

Existen 4 tipos de envíos en el protocolo USB:

- *Control*: Se usan para llevar a cabo la configuración de la conexión a nivel general. Los descriptores del dispositivo son enviados de este modo.
- *Bulk (masivo)*: Se usan para transferencia masiva de datos. Son útiles para gestionar grandes cantidades de datos en cada envío. Las impresoras, o los dispositivos de almacenamiento masivo usan este tipo de envíos.
- *Interrupción*: Se usan para transferencias de datos pequeñas. Dispositivos como los ratones, los teclados o los sensores médicos usados en este proyecto usan este tipo de envío.
- *Isócronos*: Se usan para transferencias de datos en tiempo real. Son útiles para dispositivos como cámaras, reproductores de vídeo o reproductores de audio.

En este proyecto no se pretende explicar en detalle el protocolo USB, por lo que sólo se explicarán los aspectos relevantes que serán mencionados más adelante (capítulos 3 y 4). El sensor utilizado en el proyecto usa dos tipos de envío de los cuatro anteriores: envíos de control y de interrupción. Los envíos de control se realizan por medio del endpoint 0 a través de tramas en cuyos campos se especifican detalles como la dirección de envío, el tipo de envío, el endpoint al que se debe enviar una petición concreta, los parámetros necesarios o datos concretos y la longitud de dichos datos. Estos detalles se especifican como sigue:

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

Tabla 2.1: Campos de un paquete setup

Estos campos conforman una trama completa de 8 bytes llamada setup package. Se puede observar que en el primer byte llamado *bmRequestType* se especifican tanto la dirección del envío como el tipo de trama que se envía (por ejemplo Class Interface). El campo *bRequest* indica el tipo de petición que se está llevando a cabo (por ejemplo 0x9 se correspondería con SET_REPORT, para envíos de Class Interface) mientras que el campo wValue se usaría para establecer algún parámetro que pudiera ser necesario. Los dos últimos campos correspondientes a los cuatro últimos bytes de la trama se corresponden con wIndex, usado para establecer el Offset al que va dirigida la petición, y con wLength, que indicaría la longitud de la trama de datos en el caso de que hubiera.

Los envíos de interrupción son usados para intercambiar los datos de la capa de aplicación del protocolo. Aquí es donde el sensor envía las medidas que ha tomado del paciente. En este tipo de tramas, sólo es necesario indicar el endpoint al que va dirigida la trama, los datos que se transmitan y su longitud.

2.3. Sensores

Los sensores se definen como dispositivos capaces de detectar una determinada acción externa y transmitirla adecuadamente. Ejemplos muy comunes de sensores son los sensores de luz, de humedad, de presión, de temperatura o de proximidad. En el ámbito biomédico, se definen como dispositivos capaces de determinar magnitudes fisiológicas de interés y de transmitirlas correctamente, tales como temperatura, presión sanguínea, oxigenación, actividad cardíaca, actividad neuronal… En este apartado, se hablará de los sensores más comunes en los entornos biomédicos: el termómetro, el tensiómetro, el oxímetro y el electrocardiógrafo.

El objetivo de estos dispositivos es obtener medidas cuantitativas de las variables fisiológicas importantes de los pacientes durante los periodos críticos de sus funciones biológicas.

❖ Termómetro:

Un termómetro es un instrumento capaz de medir la temperatura corporal de un paciente en un instante dado. En los entornos biomédicos es común encontrar termómetros digitales que, valiéndose de dispositivos transductores (termistores) son capaces de convertir las variaciones de tensión en valores binarios, los cuales son muy útiles a la hora de ser transmitidos. El dispositivo a usar en esta práctica es un termómetro infrarrojo con interfaz USB.

❖ Tensiómetro:

Un tensiómetro es un instrumento capaz de medir la presión sanguínea de un paciente. Pueden medir tres tipos de presión: la presión arterial (entre 30 y 300 mmHg), la presión venosa (entre 5 y 15 mmHg) y la presión pulmonar (entre 6 y 25 mmHg) siendo la presión venosa la más común. Esta medida junto con otras medidas fisiológicas ofrece una inestimable ayuda para diagnosticar la situación cardiovascular y el funcionamiento cardiaco del paciente. El tensiómetro es capaz realizar medidas de manera invasiva (mediante el uso de vías) o de manera no invasiva (medidas tomadas desde el exterior) ofreciendo información sobre la presión máxima (sistólica) y mínima (diastólica). Existen tensiómetros manuales y digitales, entre los que destacan éstos últimos ya que son más prácticos y más fiables. El tensiómetro usado en este proyecto es un tensiómetro digital de brazo.

❖ Oxímetro:

Un oxímetro es un instrumento capaz de medir la oxigenación en la sangre del paciente o, dicho de otro modo, la capacidad de la hemoglobina para transportar oxígeno. Sus valores más comunes están entre el 95% y el 100%. Los oxímetros más comunes son los oxímetros de pulso (o pulsioxímetros), que constan de dos LED ubicados de cara a un fotodiodo. Entre los LED y el fotodiodo se ubica el dedo del paciente. Cada

LED emite una señal de luz roja e infrarroja respectivamente. La absorción de estas dos longitudes de onda es muy diferente entre la oxihemoglobina y su forma no oxigenada, por lo que de la relación entre la absorción de ambas longitudes de onda se deduce la concentración de oxigenación en la sangre. En este proyecto se usa un oxímetro de este tipo.

- Electrocardiógrafo:

Un electrocardiógrafo es un instrumento capaz de captar la actividad eléctrica del corazón mediante el uso de cuatro electrodos situados en las extremidades y otros seis electrodos situados en posiciones precordiales. El registro de dicha actividad se denomina electrocardiograma (ECG). En el presente proyecto no se usará instrumentación de este tipo.

- Otros sensores:

En los entornos biomédicos también es común encontrarse más tipos de sensores como los electroencefalógrafos, sensores de flujo aéreo nasal, sensores de movimientos o caleidoscopios entre muchos otros.

2.4. Sistemas embebidos

Un sistema embebido o empotrado es un sistema de computación diseñado para cubrir una (o varias) funciones, generalmente en tiempo real. La principal diferencia entre este tipo de sistemas y los PC de propósito general radica en este aspecto: mientras que los PC de propósito general están diseñados para cubrir un amplio rango de funciones y necesidades, los sistemas embebidos suelen tener un propósito mucho más concreto.

Este tipo de sistemas se comenzó a desarrollar durante la década de 1980 por la empresa IBM. Hoy se pueden encontrar en todos los dispositivos que requieran un sistema de control o gestión electrónico, como pueden ser máquinas expendedoras, impresoras o fotocopiadoras, taxímetros, control de accesos…

La manera habitual de programar este tipo de sistemas se basa generalmente en el lenguaje ensamblador del microprocesador o microcontrolador que se incluya en el sistema. Sin embargo, existen compiladores de lenguajes más “universales” como C o C++ e incluso, en caso de que el tiempo no sea un factor determinante, leguajes interpretados como Java.

Los componentes principales de un sistema embebido son el microprocesador o unidad de procesamiento (microcontrolador, DSP…), unidades de memoria, actuadores (elementos con propósitos específicos como motores o conmutadores de relé), puertos

de comunicación (RS-232, USB, Wi-Fi…), módulo de energía, módulo de reloj, conversores A/D y D/A, y módulos de entrada y salida (teclados, pulsadores, monitores…). En general, un sistema embebido simple contará con un microprocesador, memoria, unos pocos periféricos de E/S y un programa dedicado a una aplicación concreta almacenado permanentemente en la memoria.

Los microprocesadores, en esencia, consisten en una serie de elementos básicos, (transistores, diodos o resistencias) que combinados forman elementos lógicos (puertas AND, OR, INV…), capaces de formar a su vez elementos básicos de la electrónica digital tales como contadores, ALU o registros. Las diferentes configuraciones de estos elementos pueden llegar a formar el microprocesador completo. Su funcionamiento es posible gracias a la existencia de memorias capaces de almacenar las instrucciones que el microprocesador es capaz de procesar así como datos de entrada o salida. Dichas memorias se pueden dividir en dos tipos: memorias de sólo lectura (ROM) o memorias de acceso aleatorio (RAM). Estos elementos junto con los módulos E/S proveen un sistema de computación completo en un sistema embebido. Los lenguajes de programación más eficientes son el lenguaje ensamblador capaz de proveer una traducción directa entre instrucción escrita e instrucción de código máquina y el lenguaje C que, además de estar estandarizado y ser fácilmente transferible a cualquier procesador, ofrece una eficiencia similar a los lenguajes ensambladores propios del procesador.

Los microcontroladores proveen en un mismo chip integrado la unidad de procesamiento (CPU), memorias RAM y ROM, circuito de reloj y módulos E/S de modo que no requieren de chips adicionales para ofrecer una funcionalidad completa de computación. Las principales ventajas a la hora de diseñar sistemas embebidos con microcontroladores son la notable reducción de coste y espacio así como la disminución de componentes en el sistema, lo que implica una gran ventaja a la hora de diseñar sistemas médicos portables. Asimismo, incluyen conversores analógico-digital lo que facilita el inmediato uso en instrumentación.

El uso de PC en actividades cotidianas ha conseguido generar una gran popularidad de estos dispositivos en actividades médicas debido a su alto rendimiento en recogida de datos y procesamiento. Hoy día emergen como auténticas estaciones de tratamiento de datos, siendo capaces de recoger, procesar, almacenar y mostrar los datos procedentes de una gran variedad de sensores.

A nivel comercial, existen una serie de sistemas basados en circuitos embebidos muy amplia, ofreciendo por precios bastante razonables una gran gama de placas con funcionalidades muy diversas, sólo delimitadas por la capacidad de procesado de éstas. A nivel de diseño biomédico, cualquier placa capaz de manejar eficientemente los datos procedentes de los diferentes tipos de sensores (en este proyecto bastaría con un termómetro, un tensiómetro y un pulsioxímetro) bastaría para abordar los requisitos del sistema a diseñar. A modo de ejemplo, en la actualidad serían una buena opción las placas BeagleBone, Raspberry Pi o Arduino Uno, ya que ofrecen unas especificaciones técnicas muy satisfactorias y su precio no asciende hasta cifras inabordables. Dichas especificaciones técnicas se adjuntan en el anexo B de la memoria.

Ilustración 2.1: Sistemas basados en microprocesador o microcontrolador

2.5. Sistemas operativos

En cualquier dispositivo que use una CPU como módulo de procesamiento de datos debe haber un programa encargado de realizar tareas básicas para poder llevar a cabo un correcto funcionamiento del sistema a nivel general. Este programa es generalmente el sistema operativo. En los sistemas embebidos se puede optar entre programar un pequeño sistema operativo en el lenguaje ensamblador propio del procesador y elegir un sistema operativo ya desarrollado que sea compatible con la familia de procesadores a la que pertenezca la CPU del sistema. En el pasado lo más común era desarrollar la primera opción, escribiendo en lenguaje ensamblador un código propietario en el que se incluían los controladores de los elementos hardware del sistema y las interfaces desde cero.

El sistema operativo es el programa encargado de realizar las tareas más básicas, entre las que destacan la planificación de procesos, la gestión de memoria del sistema, la gestión de archivos o la gestión de los módulos E/S.

La familia de procesadores más común en el mercado de sistemas embebidos es la familia ARM, para la cual existe compatibilidad con prácticamente todos los sistemas operativos dedicados a sistemas embebidos.

Entre los sistemas operativos dedicados a sistemas embebidos, destaca un tipo especial denominado sistema operativo en tiempo real, cuya principal característica es la restricción estricta en el tiempo de respuesta por parte del sistema.

Los sistemas operativos dedicados a sistemas embebidos más comunes se presentan a continuación.

❖ Linux embebido

Linux embebido se refiere al uso del núcleo Linux en un sistema embebido. Éste combina el núcleo Linux con otras utilidades de software libre, como pueden ser los controladores desarrollados en código abierto para cualquier elemento hardware del sistema para generar un sistema operativo útil en el uso de estos sistemas. Su principal ventaja es que es un sistema operativo modular, de manera que ofrece una gran facilidad a la hora de cargar o eliminar diferentes módulos del sistema como controladores de dispositivos hardware que no se encuentren en el sistema embebido en cuestión. Este hecho genera que el sistema operativo pueda llegar a ocupar un espacio más reducido que otros sistemas operativos.

Otras grandes ventajas que ofrecen los sistemas operativos basados en el núcleo de Linux son el uso de código abierto (ofrece la posibilidad de modificación del código para adaptarlo de formas más eficientes bajo la responsabilidad del programador), su reducido espacio que puede llegar a los 2MB frente a los 21MB que ocupa Windows CE, generalmente la ausencia de costes por derechos de autor, estabilidad y respaldo por parte de la comunidad del software libre.

Una gran cantidad de sistemas embebidos usan este tipo de sistemas operativos, como Raspberry Pi o BeagleBone (distribución Angstrom).

Algunos ejemplos de plataformas soportadas por Linux son PowerPC, ARM o MIPS.

❖ Windows CE

Es el sistema operativo desarrollado por Microsoft para sistema embebidos. El núcleo de este sistema operativo ha sido utilizado en numerosas aplicaciones, tales como telefonía móvil (destacando Windows Phone) o videojuegos. Su principal ventaja a la hora de desarrollar software son las herramientas de programación, ya que son similares a las usadas para programar en los sistemas operativos Windows de PC, lo cual no requiere del aprendizaje de nuevas plataformas de programación por parte de los desarrolladores de software para PC. Sus principales desventajas son los costos por derechos de autor y su indisponibilidad para eliminar controladores innecesarios.

Su versión más actualizada es la 2013, presentada en Junio de ese mismo año.

Windows CE puede ser usado en arquitecturas como Intel x86 y otras compatibles, ARM o MIPS.

❖ VxWorks

VxWorks es un sistema operativo de tiempo real, basado en Unix, vendido y fabricado por Wind River Systems. Como la mayoría de los sistemas operativos en tiempo real, vxWorks incluye kernel multitarea con un planificador capaz de permitir que los procesos puedan tomar la CPU arbitrariamente. Otras ventajas de este sistema operativo

es la capacidad para ofrecer una respuesta rápida a las interrupciones, comunicación entre procesos, sincronización y sistema de archivos. Es compatible con el sistema POSIX, que es el sistema de estándares desarrollado por el IEEE usado por la gran mayoría de sistemas operativos de código abierto como Linux.

Las plataformas sobre las que se puede implementar VxWorks son entre otras la familia Intel x86, MIPS, PowerPC o ARM.

- QNX

QNX es un sistema operativo de tiempo real de tipo Unix que cumple con la norma POSIX. Fue desarrollado por la empresa canadiense QNX Software Systems, y su finalidad es principalmente su uso en sistemas embebidos. Actualmente es gestionado y actualizado por BlackBerry. Está basado en una estructura de micronúcleo, lo cual proporciona buena estabilidad frente a cualquier tipo de fallos del sistema.

QNX está disponibles para plataformas como las pertenecientes a la familia Intel x86, MIPS, PowerPC o ARM.

Se puede comprobar que todos los sistemas operativos expuestos anteriormente y en general la mayoría de sistemas operativos diseñados para sistemas embebidos soportan la arquitectura ARM, presente en la gran mayoría de sistema sobre placas del mercado, como los mencionados en el apartado de sistemas embebidos (Arduino, BeagleBone o Raspberry Pi).

2.6. Tablet

Una Tablet es una computadora de mayor tamaño que un móvil o PDA pero de menor tamaño que un PC generalmente, integrada en una pantalla táctil que en la mayoría de casos hace de interfaz tanto de entrada como de salida (se interactúa usando los dedos o un estilete). Suelen tener tamaños comprendidos entre 7 y 12 pulgadas.

Durante la década de 2000 se han diseñado diversos prototipos, entre los que cabe destacar la desarrollada por Nokia en 2001 (el *Nokia 510 webtablet*), aunque no ha sido hasta el año 2010 con el lanzamiento por parte de la empresa Apple Inc del *iPad* basado en su exitoso móvil *iPhone* que no se alcanzó un éxito comercial. Desde ese año se han generado una gran cantidad de dispositivos de este tipo por parte de la gran mayoría de empresas dedicadas al desarrollo de dispositivos de computación (Asus, Microsoft, Samsung, Sony, Hewlett-Packard…).

A nivel de usuario ofrecen una gran serie de utilidades, entre las que cabe destacar la capacidad de almacenamiento y lectura de libros, función de reproductor de música y vídeo, conexión a Internet, cámaras fotográficas, videoconferencias o servicio de posicionamiento (GPS). Sin embargo, a nivel de desarrollador, las Tablet ofrecen una plataforma electrónica muy avanzada, puesto que en el mercado pueden hallarse dispositivos con interfaz USB, Bluetooth y conexión a Internet por diversas interfaces, que pueden ser usadas para generar comunicación entre diversos dispositivos.

Del mismo modo que sucede con los sistemas embebidos, las Tablet también requieren de un sistema operativo con el cual poder llevar a cabo sus funciones más básicas. Los sistemas operativos más comunes en Tablet son Android, iOS, BlackBerry Tablet OS y Windows 8.

❖ Android:

Es un sistema operativo desarrollado por Google Inc basado en el núcleo de Linux. Es usado en la gran mayoría de dispositivos con interfaz táctil del mercado. Como principal inconveniente presenta una mayor dificultad en cuanto a compatibilidad entre diferentes dispositivos debido los diferentes tamaños de pantalla y resoluciones de éstos. Sin embargo, ofrece una amplísima variedad de aplicaciones disponibles (Play Store), así como una mayor facilidad a la hora de desarrollar aplicaciones (no se requiere tasas anuales ni registrarse como desarrollador en su base de datos). Dispone de una máquina virtual JAVA lo cual ofrece una forma universal de programación para el desarrollador. Las diferentes versiones de Android se muestran a continuación:

Versión	Nombre	Fecha	Nivel API	Distribución
4.4	*KitKat*	31/10/2013	19	8.5%
4.3.x	*Jelly Bean*	24/7/2013	18	8.5%
4.2.x		13/11/2012	17	18.8%
4.1.x		9/7/2012	16	33.5%
4.0.3 - 4.0.4	*Ice Cream Snadwich*	16/12/2011	15	13.4%
3.2	*Honeycomb*	15/7/2011	13	0.1%
2.3.3 – 2.3.7	*Gingerbread*	9/2/2011	10	16.2%
2.2	*Froyo*	20/5/2010	8	1.0%

Tabla 2.2: Versiones de Android y características

En un principio, Android se presentaba en dispositivos económicos, diferenciándose así del dispositivo que sobresalía en cuanto a prestaciones y precio durante esos años: el *iPhone*. Sin embargo, no tardaron en aparecer dispositivos de alta gama que incluso hoy día compiten en prestaciones, con similares precios, con el *iPhone*.

❖ iOS:

Es el sistema operativo creado y desarrollado por la empresa Apple Inc. Aparece únicamente en los dispositivos desarrollados por dicha marca, como el *iPhone*, el *iPad* o los *iPod* táctiles. El dispositivo original en usarlo fue el *iPhone*, para el cual se creó una tienda de aplicaciones (AppStore). Para desarrollar aplicaciones en este sistema operativo se requiere pagar una licencia de desarrollador anual, y se debe usar un lenguaje de programación especial llamado Objetive-C (similar a C), que sólo puede ser desarrollado en los PC de la marca Apple (los Mac). Por ser el primero en ofrecer un servicio de aplicaciones como el que conocemos hoy día, durante un largo periodo de tiempo se mantuvo como el sistema operativo con mayor número de aplicaciones disponibles, aunque con el tiempo ha perdido terreno en favor de Android.

❖ BlackBerry OS:

Es el sistema operativo creado por la empresa Research in Motion para la gama de dispositivos móviles *BlackBerry*. Fue uno de los primeros sistemas operativos en ofrecer una interfaz rápida y eficaz a servicios de Internet (navegación, correo electrónico, Youtube…) en dispositivos móviles. Hoy en día ha perdido una gran cantidad de dispositivos vendidos en el mercado a favor de iOS y Android.

❖ Windows 8:

En los últimos años se ha comprobado como los sistemas operativos mencionados anteriormente han irrumpido con fuerza en el mercado, adquiriendo una presencia muy notable en toda la gama de dispositivos móviles actuales. Era de esperar que tarde o temprano Windows pasara a desarrollar un sistema operativo que compitiera con los sistemas operativos dominantes de este mercado, lanzando así Windows Phone 7 el 21 de Noviembre de 2010. Este sistema operativo apareció en un principio implantado en la gama de móviles Nokia Lumia, aunque actualmente se puede encontrar en una mayor variedad de dispositivos.

En cuanto a fabricantes de Tablet en el mercado actual, se pueden destacar grandes empresas como Apple, Samsung, Sony, Asus, Motorola, BlackBerry o Hewlett-Packard, así como empresas de menor tamaño como BQ, Woxter, Leotec, Lenovo o Energy Sistem. Del mismo modo, también se han lanzado empresas de software a este mercado, ofreciendo dispositivos como la Microsoft Surface o Google Nexus.

2.7. Baterías

Una batería eléctrica (o acumulador eléctrico) es un dispositivo capaz de proporcionar electricidad a partir de la energía química contenida en una celda electroquímica. Dichas celdas constan de un electrodo positivo (cátodo) y un electrodo negativo (ánodo) a través de los cuales se genera una diferencia de potencial capaz de generar una corriente si se ubica un conductor entre ambos. Su funcionamiento se basa en las reacciones químicas de reducción-oxidación, en las cuales uno de los elementos se oxida (pierde electrones) mientras que otro se reduce (gana electrones). Gracias a este fenómeno se produce la corriente eléctrica.

Las pilas son acumuladores eléctricos formados por una celda galvánica, mientras que las baterías están formadas por varias conectadas en serie o en paralelo. Existen dos tipos de celdas: las celdas primarias y las celdas secundarias. Las celdas primarias convierten la energía química en electricidad de forma irreversible, de modo que cuando concluye su uso se desechan. Las celdas secundarias son capaces de regenerar su energía química por medio del suministro de energía eléctrica a la celda.

Los parámetros que caracterizan a un acumulador eléctrico son:

- El voltaje: se define como la diferencia de potencial eléctrico que hay entre dos puntos espaciales. Es el primer parámetro a tener en cuenta puesto que los dispositivos eléctricos o electrónicos funcionan con un único valor de voltaje.
- La corriente: es la cantidad de carga que fluye por un punto espacial en cada unidad de tiempo. Para un valor fijado de tensión, dependiendo de la composición del circuito eléctrico, las baterías proporcionarán una intensidad de corriente eléctrica u otra. En este proyecto, para caracterizar el consumo eléctrico del sistema a diseñar se analizará el consumo de corriente que realice éste.
- La capacidad de carga: es otro parámetro de grandísima importancia. Se define como la cantidad de carga que contiene la batería y repercute directamente en el tiempo de autonomía que es capaz de ofrecer la batería. Se mide en miliamperios-hora, lo cual indica la cantidad de miliamperios que es capaz de proporcionar la batería durante una hora. De este modo, una batería que contenga una capacidad de carga de 1200mAh durará la mitad de tiempo que otra batería que contenga una capacidad de 2400mAh.
- Resistencia interna: hace referencia a la oposición al paso de corriente eléctrica a través de la propia batería. La resistencias internas de las baterías suelen tener un valor muy pequeño, por lo que son capaces de suministrar corrientes eléctricas mucho mayores que las de las pilas.

- El peso: es otro gran importante parámetro a tener en cuenta puesto que el diseño del concentrador de señales que se persigue en este proyecto tiene como finalidad ofrecer un sistema de gestión de medidas hospitalarias más cómodo que los actuales, para lo cual es importante que dicho dispositivo tenga un peso adecuado.
- Las dimensiones de la batería también pueden llegar a ser críticas, ya que ésta puede llegar a ser el elemento más grande del sistema.

Como ya se definió, una batería como un dispositivo capaz de generar electricidad a partir de la energía química contenida en su interior. Existen en la actualidad muchas maneras de lograr este propósito (experimentos básicos de electrólisis actuarían como pequeñas baterías). Los tipos de baterías más importantes se mencionan a continuación:

- Baterías de estado sólido de ion litio.
- Baterías de plomo ácido.
- Baterías de níquel-cadmio.
- Baterías de níquel-hidruro metálico.
- Baterías de níquel-hierro.

De las anteriores, las más comunes a día de hoy son las baterías de plomo ácido. No obstante, las baterías de ion litio de estado sólido han tenido un desarrollo más reciente y actualmente se encuentran en constante desarrollo.

Otra gran característica de las baterías es el llamado efecto memoria. Este se manifiesta en la reducción de la capacidad de carga que se presenta cuando una batería no se carga (o descarga) completamente.

2.8. Concentradores comerciales

Los concentradores de señales son equipos capaces de recibir las señales de varios sensores al mismo tiempo, de analizarlas, gestionarlas y mostrarlas a través de un dispositivo de salida. Para su diseño es necesario disponer de la mayoría de elementos mencionados en este capítulo, ya que en esencia consisten en un soporte hardware (como un sistema embebido) sobre el que se ejecuta un sistema operativo capaz de ejecutar programas encargados de la gestión de las interfaces de conexión a través de las cuales se puede comunicar el equipo con los sensores. Todo el dispositivo funcionará provisto de una batería como fuente de alimentación. La información contenida en las señales recibidas de los sensores podrá ser mostrada gráficamente a través de un dispositivo de salida, como por ejemplo una Tablet.

Estos dispositivos tienen como finalidad proporcionar una toma de medidas del paciente rápida, cómoda y eficiente. Para ello, por lo general, son capaces de tomar una

gran variedad de tipos de medidas (temperatura, oxigenación, presión sanguínea…) en el menor tiempo posible, y suelen ser capaces de gestionar estos datos y enviarlos a un sistema de datos central capaz de almacenarlos para futuras consultas.

A modo de ejemplo se expondrán una serie de dispositivos que se pueden encontrar actualmente en el mercado.

❖ Welch Allyn Spot Vital Signs:

Dispositivo desarrollado por Welch Allyn, como indica su nombre. Es un dispositivo muy similar al que se persigue desarrollar en el presente proyecto. Entre sus funcionalidades se encuentran las siguientes:

- Rapidez y precisión en la toma de las diferentes medidas que es capaz de tomar (temperatura, presión sanguínea, oxigenación y pulso en 30 segundos aproximadamente).
- Modular. Posibilidad de optar por una medida concreta.
- Conectividad a interfaces con almacenamiento de datos de los pacientes.
- Portabilidad. Se puede montar sobre un carrito para transportarlo.
- Pantalla LCD para observar las medidas en tiempo real.
- Precio económico.
- Fácilmente adaptable a otras necesidades.
- Batería integrada que permite un continuo funcionamiento.

Welch Allyn
Advancing Frontline Care™

Ilustración 2.2: CS Welch Allyn

❖ Saruro Tele Care Box:

Desarrollado por la Universidad Nacional de Colombia, este dispositivo está diseñado para manejar las señales vitales de los pacientes remotamente. Contiene una serie de módulos encargados de tareas concretas destinadas, conjuntamente, a ofrecer la posibilidad del diagnóstico remoto del paciente. Sus funcionalidades son:

- Hardware independiente para la toma de medidas de electrocardiograma (ECG), presión sanguínea no invasiva y oximetría de pulso.
- Batería o fuente de alimentación externa con tecnología de conmutación para reducir el ruido eléctrico.
- Capacidad de monitorización y representación gráfica de las señales medidas en tiempo real.
- Portabilidad.
- Conectividad a bases de datos remotas para el almacenamiento y gestión de datos del paciente.

Ilustración 2.3: Saruro Tele Care Box

❖ Sure Signs VS3:

Desarrollado por Philips, sus funcionalidades son:

- Medidas de señales biológicas de temperatura, presión sanguínea, pulso cardiaco y oximetría.
- Condicionamiento para uso en cuidados generales, emergencias, UCI o cirugía.
- Capacidad de actualización vía USB.
- Monitor para representaciones gráficas en tiempo real.
- Batería de litio para una mayor autonomía.
- Atención de soporte técnico 24 horas.
- Posibilidad de venta con carrito de transporte.
- Piezas de recambio.

❖ Omecrom FT Surveyor

Desarrollado por RGB Medical Devices. Sus características de funcionamiento son las siguientes:

- Capaz de tomar medidas de temperatura, presión sanguínea no invasiva, oximetría, electrocardiogramas y respiración.
- Posibilidad de uso en UCI, UCI neonatal, quirófanos o unidades de reanimación.
- Monitor de tiempo real y filtración de ruido en pulsioximetría.
- Capacidad para mostrar 6 curvas de ECG a la vez y para grabar 12 segundos de secuencia de ECG.
- Conexión a central de vigilancia.

❖ CareTec

Es un sistema desarrollado por EDP Ingeniería. Consta de sensores inalámbricos que se conectan de manera inalámbrica a un concentrador de señales que, a su vez, envía los datos medidos a un monitor de constantes vitales. Tiene como finalidad presentar un sistema en el que el contacto con pacientes que padezcan enfermedades infecciosas no sea necesario, evitando así posibles contagios. Tiene un sistema de seguimiento, de manera que es capaz de conocer la localización de los pacientes en cada momento, ofreciendo de esta manera la posibilidad de conocer qué personas han tenido contacto con este paciente. Los datos tomados son digitalizados y enviados a una base de datos con un registro de las medidas tomadas a todos los pacientes.

Capítulo 3

Normativa de dispositivos médicos

3.1. Introducción

En este capítulo se hará un repaso de la normativa referente a la seguridad en dispositivos electrónicos en hospitales y entornos médicos así como de la normativa referente a los protocolos e interfaces de conexión que puedan involucrarse en el desarrollo de este proyecto.

Como en cualquier tipo de proyecto que involucre el desarrollo de algún dispositivo, un paso imprescindible será el conocimiento de las normas y regulaciones que actualmente rigen el uso de dicho dispositivo. Para ello se buscarán y expondrán inicialmente qué organismos son los encargados de realizar dichas normas.

La normalización o estandarización es el proceso de redacción, aprobación y mejora de normas (o estándares) que se establecen para garantizar el acoplamiento de elementos elaborados independientemente, su calidad, la seguridad de funcionamiento y su responsabilidad social.

Asimismo, un estándar o norma es un acuerdo de múltiples entidades que establece una referencia de criterios para situaciones arbitrarias. Se compone de un set de normas, condiciones o requisitos correspondientes a la definición de términos, clasificación de componentes, enumeración de procedimientos, especificación de recursos, funcionamiento, diseño de operaciones y medidas de calidad de los recursos, productos, sistemas, servicios y prácticas.

Una especificación es un documento usado para controlar la adquisición de recursos teniendo en cuenta los criterios de diseño, el funcionamiento del sistema, los materiales y la información técnica.

3.2. Organismos de regulación

Los organismos de regulación se definen como organismos sin ánimo de lucro encargados de elaborar, comprobar y actualizar una serie de normas para su aplicación en el área en el que éstos imperen. Se dividen según el área geográfica sobre la que influyen. Referentes al ámbito de este proyecto, a nivel internacional, los más importantes son:

- ISO: Organización Internacional para la normalización.
- IEC: International Electrotechnical Comission.
- IEEE: Institute of Electrical and Electronic Engineers.
- ITU: Unión Internacional de Telecomunicaciones.

Ilustración 3.1: Principales organismos de normalización

A nivel regional, destaca CENELEC (Comité Europeo de Normalización Electrotécnica) y ETSI (European Telecommunications Standards Institute) en el marco europeo. En España la entidad de normalización es AENOR quien es responsable de la edición en español de las distintas normas europeas. Ésta última es la encargada de representar a España dentro de los organismos internacionales ISO, IEC y CENELEC.

ISO es un organismo encargado de promover normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica. Para temas relacionados con estas ramas se asocia con IEC, que es una organización de normalización en los campos eléctrico y electrónico. En ocasiones también se asocia con IEEE, que es una asociación internacional de técnicos e ingenieros dedicada a la estandarización, la mayor del mundo en este ámbito.

ITU es el organismo especializado de Telecomunicaciones de la ONU encargado de regular las telecomunicaciones a nivel internacional entre distintas administraciones y empresas operadoras.

En este proyecto no se atenderán las normas del organismo ITU puesto que no llegan a usarse sistemas de telecomunicación nuevos, sino que se usan tecnologías como Bluetooth o USB que ya están reguladas en sus especificaciones. Sin embargo, sí se tendrán en cuenta las normas de los organismos ISO, IEEE e IEC.

Concernientes al ámbito del presente proyecto, los organismos de regulación presentan los siguientes documentos sobre normativa:

- ISO IEEE 1107 Health informatics - Medical / health device communication standards.

- IEC 60601-1 - Medical electrical equipment.

En este apartado se expondrán a gran escala los principales puntos contenidos en los estándares anteriores, realizando una síntesis resumida de los aspectos más relevantes, evitando dar demasiados detalles que puedan ser innecesarios o redundantes.

3.3. Normas CEN ISO/IEEE 11073

CEN ISO/IEEE 11073 *Health informatics - Medical / health device communication standards* habilita un sistema de comunicación entre dispositivos biomédicos (tales como sensores o concentradores) y sistemas computacionales externos. Proporciona un sistema de intercambio de datos automático y detallado sobre señales vitales y el estado de los dispositivos que, usado apropiadamente, pueden contribuir al desarrollo de una mejoría en el cuidado y tratamiento de pacientes acelerando el proceso, haciéndolo más seguro y reduciendo su coste. En el propio estándar, aparecen las palabras *"el estándar atiende la necesidad de una definición abierta de un estándar que convierta la información de los dispositivos en un formato de transmisión común para lograr un intercambio de datos entre los dispositivos biomédicos personales y los servidores (teléfonos móviles, PC o concentradores biomédicos)"*.

La norma CEN ISO/IEEE 11073 se subdivide en un amplio rango de normas específicas, entre las que cabe destacar las normas referentes a los sensores biomédicos. Dichas normas se listan a continuación:

11073 – 10404 Device Specialization	Pulse Oximeter
11073 – 10406 Device Specialization	Basic Electrocardiograph
11073 – 10407 Device Specialization	Blood Pressure Monitor
11073 – 10408 Device Specialization	Thermometer
11073 – 10415 Device Specialization	Weighing Scale
11073 – 10417 Device Specialization	Glucose Meter
11073 – 10418 Device Specialization	INR Monitor
11073 – 10419 Device Specialization	Insulin Pump
11073 – 10420 Device Specialization	Body Composition Analyzer
11073 – 10421 Device Specialization	Peak Flow
11073 – 10441 Device Specialization	Cardiovascular Monitor
11073 – 10442 Device Specialization	Strength Fitness Equipment
11073 – 10471 Device Specialization	Independent Living Activity Hub
11073 – 10472 Device Specialization	Medical Monitor

Tabla 3.1: Estándares para dispositivos médicos

Dichas normas no serán explicadas en el presente proyecto debido a que la normativa y estandarización de los sensores se queda fuera del ámbito de éste, ya que el objetivo de este proyecto no es el desarrollo de sensores.

Las normas concretas de las que se ha obtenido la información necesaria para desarrollar el proyecto se listan a continuación:

11073 – 10101 Point of Care	Nomenclature
11073 – 10201 Point of Care	Domain Information Model
11073 – 20101 Application Profile	Base Standard
11073 – 20601 Application Profile	Optimized Exchange Protocol
11073 – 30200 Transport Profile	Cable Connected

Tabla 3.2: Estándares base

A continuación se explicarán por encima los puntos más relevantes de la normativa nombrada anteriormente.

❖ ANTECENTES

- Estos estándares **están destinados** a dispositivos de salud personal tales como termómetros, monitores de actividad o monitores de glucosa así como a dispositivos de cuidado más intensivo. Componen una familia de estándares que pueden ser implementados de manera conjunta ofreciendo una conectividad optimizada para los dispositivos que pretenden ser usados en las diferentes operaciones. Dichos estándares se pueden dividir en cuatro partes:
  - Datos de dispositivos, incluyendo nomenclatura, optimizada para la representación de señales vitales basada en modelos de datos orientados a objetos, y dispositivos especializados.
  - Aplicaciones de servicios generales.
  - Interconexión y especificaciones de interfaces.
  - Especificaciones de nivel físico (conexiones cableadas o inalámbricas)

  Las funcionalidades que se buscan abarcan la interoperabilidad en tiempo real (los datos de los dispositivos son almacenados, sincronizados, procesados y representados en cuestión de fracciones de segundo), conexiones plug-and-play (el usuario sólo tiene que conectar los dispositivos, éstos se configuran automáticamente) y el intercambio optimizado de datos entre dispositivos (los datos referentes a señales vitales del paciente son tomados, almacenados y procesados por una serie de aplicaciones definidas para evitar el uso de equipos y software adicionales sin pérdidas de información innecesarias).

- En la ausencia de estándares relativos a este entorno, los **principales problemas** están relacionados con la toma de datos. Sin un estándar, los datos deben ser tomados por el personal del hospital (enfermería), con

probabilidad de que se den errores humanos, o por medio de aparatos especializados, lo que implicaría la aparición de protocolos propios incompatibles entre diferentes marcas y por tanto un aumento de coste. Dicho aumento de coste genera un menor uso de dichos equipos, delimitándolos a pacientes que requieran de un cuidado más intensivo. Del mismo modo, el desarrollo de nuevos equipos más avanzados quedarían restringidos a dispositivos propios de cada empresa, lo cual dificultaría su incorporación al mercado y el avance conjunto de la tecnología biomédica a nivel general, e incluso dispositivos con características similares tenderían a ofrecer menos consistencia en las comunicaciones que pudieran tener entre sí.

- Por tanto, la motivación de esta estandarización reside en la creación de una normativa única que abarque desde la capa física (conexiones cableadas o inalámbricas) hasta la abstracción en la gestión y el intercambio de datos por medio de una interfaz de usuario que ofrezca una representación de los datos más asequible y adecuada para el usuario. El estándar IEEE 11073 está elaborado y mantenido con un alto nivel de participación internacional. De hecho, está integrado en los estándares ISO bajo el nombre de ISO CT215 Health Informatics al igual que en los estándares CEN. Ha sido organizado en colaboración con muchas otras entidades de normalización como IHTSDO, IrDA, HL7, DICOM o CLSI. La nomenclatura establecida en esta normativa ha sido muy ampliamente adoptada por un gran número de centros hospitalarios, y son compatibles con los estándares definidos por organizaciones estadounidenses como la FDA o el CDRH así como organizaciones británicas como la NPfIT. Además, el coste de integración de estas nuevas tecnologías en las líneas de productos ya existentes es bastante reducida.
- Su disponibilidad es gratuita para aquellos que han participado activamente en su desarrollo. El resto de entidades pueden acceder a esta normativa comprándola en las organizaciones de estandarización nacionales, dependiendo de cuál sea el estado donde se pretenda realizar el pago. En España el organismo competente es AENOR.

❖ RAZONES

El objetivo principal de este estándar es proveer un conjunto de directrices que optimicen el funcionamiento en la capa de aplicación (lo que el usuario maneja) y la implementación en el dominio de la información.

- Modelo de comunicación: Los siguientes aspectos hacen referencia al estándar base, considerando los medios de comunicación y la definición de protocolos referentes a ellos.
  - Los medios de comunicación deben ser lo más cercanos posible a estándares ya existentes. Es necesario que incluso los dispositivos menos comunes puedan ser integrados en el sistema definido.

- Con el fin de reducir el número de envíos y recibos durante la comunicación, se tratará de simplificar lo más posible las cabeceras de cada envío, sugiriendo un tamaño fijo en los paquetes que sean transferidos. De este modo también se simplificará el desarrollo del software de los elementos.
- La comunicación debe de ser flexible para facilitar la integración de otros tipos de protocolos de comunicación.

Los estándares relativos a la capa física son los 11073-30000, que incluyen la conexión cableada, la conexión por medio de infrarrojos y la conexión Ethernet. Se debe considerar la calidad del servicio (QoS) en cuanto a probabilidad de error. La capa de sesión no tiene por qué ser demasiado compleja, ya que incluso la detección de errores se puede llevar a cabo con objetos definidos en el MDDL (scanner objects).

- Modelo de información: Los siguientes aspectos hacen referencia al modelo de objeto Conmmunication Controller (CC).
  - Los objetos definidos en este modelo representan información que será compartida a través de los enlaces de comunicación, y forman parte del Información base de los datos sobre dispositivos médicos (MDIB).
  - Los objetos se basarán en parámetros sobre la capacidad de las interfaces de comunicación tales como velocidad, calidad o detección de errores.
  - Se podrá usar UML para definir el sistema.
  - El modelo deberá ser dinámico para poder definir el estado de las conexiones entre los dispositivos en cada instante.
  - Deberán definirse también objetos referentes a la gestión del sistema con el fin de poder ofrecer soporte técnico.

❖ ESTANDARIZACIÓN DEL NÚCLEO DEL SISTEMA

La nomenclatura del sistema está estandarizada basándose en la orientación a objetos, de manera que a la hora de desarrollar el software del sistema será más sencillo usando lenguajes de programación orientados a objetos (Java, C++…).

El escenario de ensamblaje de datos y la transmisión de los objetos anteriormente mencionados está definido en un estándar base que se subdivide en dos partes: la capa de comunicación y la capa de información, como ya se ha indicado previamente. La primera representa las funciones y métodos correspondientes a las capas 5, 6 y 7 del modelo OSI mientras que la segunda define los procesos, los formatos y la sintaxis del código de transmisión de los objetos que hayan sido descritos en el sistema. Este sistema quedaría formado por dos tipos de componentes: los agentes y los gestores (agent y manager en inglés).

- Los agentes son la parte del sistema encargada de ofrecer datos. Se corresponderían con los dispositivos conectados a los dispositivos médicos.
- Los gestores son capaces de copiar los datos aportados por los agentes, responder a cualquier cambio que pueda darse en ellos e incluso realizar cambios en dichos datos. Son utilizados generalmente con la finalidad de representar los datos remotamente, es decir, como salida de datos, pero también pueden usarse para controlar remotamente los agentes.

Estos tipos de sistemas pueden llegar a variar de muy diversas formas. De hecho, en un mismo sistema puede darse el caso de que los agentes puedan actuar como gerentes y viceversa. Otro tipo de sistemas podrían usar capas intermedias, con elementos capaces de realizar funciones de ambas partes.

Los elementos principales de este sistema formado por agentes y gestores serían:

- Aplicación de agente: consiste en un módulo encargado de hacer de interfaz entre dispositivos que usen protocolos de propietarios.
- Modelo del dominio de la información (DIM): consiste en una estructura de árbol en la que se almacenan objetos de los definidos en el estándar, referentes en lo general a señales de constantes vitales y a las relaciones entre ellas.
- Módulo de control de servicios de asociación: es un módulo definido en otros estándares que es compatible con el ISO/IEEE 11073. Su principal función es realizar las tareas de control durante los procesos de ensamblaje, negociando la posibilidad de conexiones y sus condiciones. No se transmitirían objetos en este módulo.
- Módulo de servicios de información entre dispositivos médicos: los intercambios de objetos entre agentes y gestores se definen en este módulo. Las funciones que pueden aplicarse sobre los objetos son creación, modificación y eliminación que son llevadas a cabo por las instrucciones CREATE, UPDATE y DELETE. Otras funciones más complejas pueden ser llevadas a cabo a través de peticiones especiales definidas en los atributos de los objetos.
- Capa de presentación: los objetos, atributos comunes a grupos de objetos y atributos únicos son codificados en esta capa, usando la representación ASN.1.
- Capa de sesión: todos los temas referentes a la interconexión de dispositivos son gestionados en esta capa.

❖ MODELO DEL DOMINIO DE LA INFORMACIÓN (DIM)

Como ya se explicó, la parte más “cercana al usuario” es el dominio de la información, que es a su vez la parte más importante de todas. Los objetos encargados de almacenar las constantes vitales de los pacientes y las relaciones entre ellas se definen en este modelo.

En primer lugar se mostrará un esquema que contiene todos los objetos definidos en el estándar para la elaboración de las aplicaciones que puedan ser implementadas en el sistema. Dichos objetos son muy numerosos, y pueden ser agrupados en paquetes específicos para cada tipo de funcionalidad.

En segundo lugar se dará una explicación de la funcionalidad y utilidad de cada uno de los paquetes que intervienen, poniendo en algunos casos más evidentes ejemplos de algún objeto que sea más comúnmente utilizado.

A continuación se muestra un esquema con los tipos de paquetes que conforman el modelo y las diferentes interconexiones que tienen lugar dentro de dicho esquema:

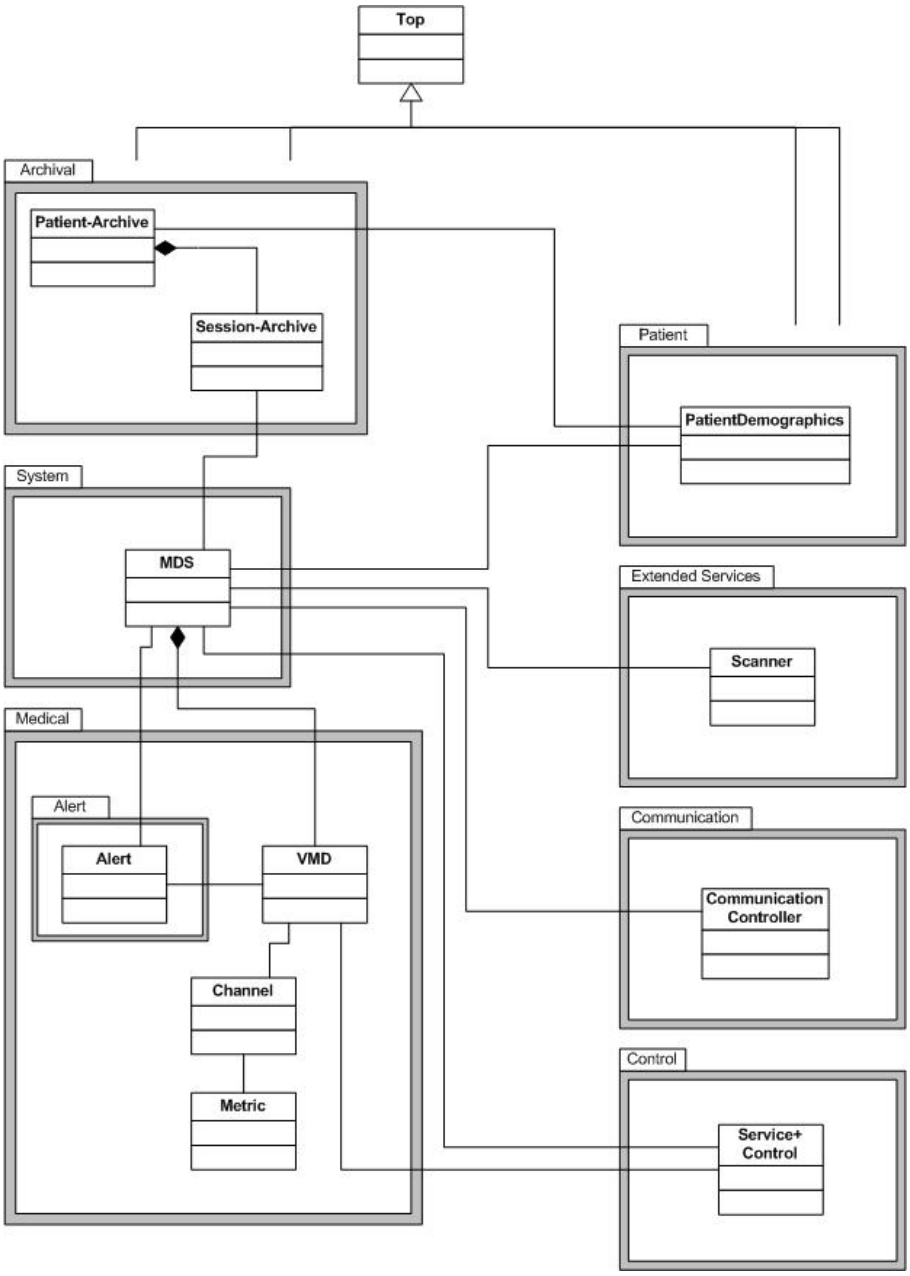


Ilustración 3.2: Esquema UML del DIM

Cada uno de los paquetes representados tendrá una funcionalidad concreta, que se definirá a continuación:

- Medical Package (paquete médico): en este paquete se definen los objetos que serán usados para representar las constantes vitales de los pacientes. Por ejemplo, se define un objeto llamado RealTimeSampleArray para representar los datos del ECG.
- Alert Package (paquete de alertas): es un paquete definido dentro del paquete médico. Su función es gestionar y generar parámetros de control y alertas para los objetos definidos en el paquete médico.
- System Package (paquete de sistema): en este paquete se definen objetos útiles para representar dispositivos médicos. Se basan en el objeto Medical Device System (MDS). Dos ejemplos de objetos definidos en este paquete son el Battery y el Clock, que representan respectivamente la batería y el reloj a través del cual se sincronizan las transferencias de datos del dispositivo. También se define el objeto raíz del árbol DIM.
- Control Package (paquete de control): en este paquete se definen los objetos encargados del control remoto de los dispositivos.
- Extended services Package (paquete de servicios extendidos): aquí se definen objetos destinados a ofrecer servicios como la lectura de datos o la generación de eventos. Ofrecen un amplio rango de atributos que hacen posible las diferentes aplicaciones que puede tener el árbol DIM. Por ejemplo, el objeto FastPeriCfgScanner está diseñado específicamente para llevar a cabo la transferencia de datos en tiempo real del objeto RealTimeSampleArray (definido en el paquete Medical Package), que es el encargado de gestionar los datos de los dispositivos de ECG.
- Communication Package (paquete de comunicación): en este paquete se definen objetos cuya finalidad es realizar el intercambio entre dispositivos que usen protocolos propietarios. Este set de paquetes es muy abierto de cara a que pueda ser modificado (o, generalmente, ampliado) para incluir nuevos paquetes que permitan la comunicación con dispositivos ajenos al desarrollo del sistema.
- Archival Package (paquete de archivado): la finalidad de este paquete reside en el almacenamiento de los datos referentes al paciente. Por ejemplo, el objeto Patient Archive es capaz de almacenar las constantes vitales del paciente, su información demográfica y su correspondiente tratamiento en un solo objeto.
- Patient Package (paquete de paciente): contiene un único objeto llamado Patient Demographics que contiene información referente a los datos del paciente. Esta información puede ser referenciada a un objeto MDS o a un objeto del paquete Archival Package con la finalidad de crear un enlace entre una serie de datos tomados de manera “anónima” a un objeto de datos relativos a un paciente concreto, generando un sistema de información completo (datos del paciente, constantes vitales y tratamiento).

Los objetos descritos representan las herramientas que aporta el sistema para poder representar y elaborar los procesos de la capa de aplicación. Sin embargo, entre la capa de aplicación y la capa física también existen otras capas que hacen posible la transmisión de datos entre diferentes dispositivos. Esto se define en el modelo de comunicación.

❖ MODELO DE COMUNICACIÓN

En este apartado se explican brevemente los componentes que intervienen en la comunicación entre los dispositivos que conforman el sistema completo. Dicha comunicación puede llegar a ser muy compleja, de modo que no se entrará en demasiado detalle en esta explicación.

- La parte más básica del sistema de comunicación se elabora partiendo de una máquina de estados finita en la que están definidos todos los posibles estados por los que puede pasar el dispositivo en cuestión y las condiciones de paso entre un estado y otro.
- Inicialización del MDIB. Se realiza durante la fase asociación, momento en el cual el dispositivo agente y el dispositivo gestor intercambian dato sobre su estado, alcanzándose así el estado de configuración de la máquina mencionada en el punto anterior. En este proceso el objeto MDSCreateEvent genera una copia del MDS del MDIB del agente en el MDIB del gestor y el objeto ContextScanner se crea en el agente realizando una copia del MDIB de éste para ser transferido al gestor (exceptuando el MDS que ya había sido enviado). El gestor evalúa esta información y genera los objetos necesarios para tener una representación completa del agente. De este modo quedan configurados los dispositivos y se pasa al estado de configurado.
- La forma de proceder a la hora de realizar transferencias entre dispositivos es la siguiente. El CMDISE proporciona una función GET que se encarga de enviar la información solicitada por el gestor. Dicha función crea un conjunto de los identificadores del agente requeridos por el gestor, que son enviados a éste.
- Las actualizaciones en el estado de los agentes son comunicadas al gestor a través de objetos scanner. Estos objetos tienen como función realizar una comprobación del estado del agente de manera periódica. En el momento en el que detectan un cambio se lo comunican al gestor. Dichos objetos scanner son creados a través de la función CREATE del MDIB del agente y pueden ser eliminados en cualquier momento a través de la función DELETE también del MDIB.

Los elementos que han sido mencionados anteriormente se ubican en una serie de capas intermedias localizadas entre la capa de aplicación y la capa física. La siguiente imagen representa esquemáticamente la composición de dichas capas:

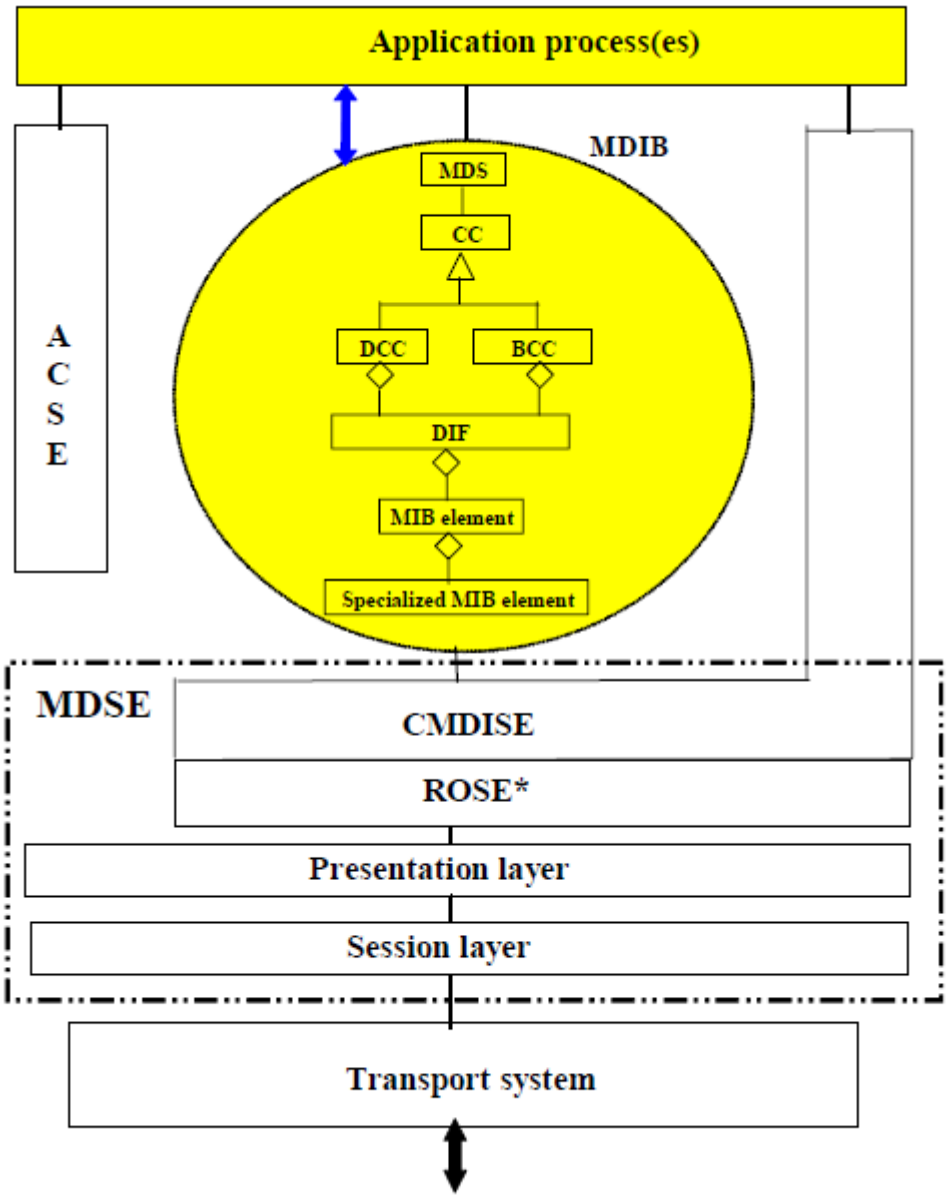


Ilustración 3.3: Modelo de comunicación

Los elementos de la imagen se describen a continuación:

- ACSE es el estándar de control de asociación definido en ISO/OSI. Proporciona flexibilidad a la hora de adoptar futuros cambios en los formatos de datos de otras capas (tales como imágenes por ejemplo). También provee herramientas de comprobación de compatibilidad entre dispositivos.
- La gestión del dispositivo es llevada a cabo por el CMDISE. Es una versión reducida del elemento de gestión definido por ISO/OSI, el CMISE.
- Cualquier operación remota es gestionada por el elemento ROSE, tales como generar eventos, devolver errores o reportar el resultado de una acción.

- Las capas de sesión y presentación añaden cabeceras con información útil para llevar a cabo los envíos y recibos de datos correctamente. En este estándar se busca que dichas cabeceras tengan un tamaño fijo mínimo para lograr una optimización durante las transferencias de datos.
- El conjunto de los elementos anteriores conforma el MDSE, que es una abstracción del servicio que se pretende ofrecer. Se caracteriza por una transparencia y flexibilidad capaces de lograr que las aplicaciones programadas no necesiten conocer los detalles del envío, ofreciendo así un mayor margen para los desarrolladores.

Los componentes del MDIB se definen por norma en el DIM. La información referente al MIB se puede clasificar brevemente como:

- MDS: elemento capaz de abstraer las características del dispositivo y de representarlo.
- Los controladores de comunicación (CC): objeto genérico que puede especificarse de dos maneras:
  - Device CC (DCC): objeto que representa el CC del agente.
  - BedSide CC (BCC): objeto que representa un gestor DCC.
- El DIF representa la interfaz de conexión (punto de acceso) del dispositivo.
- Elemento MIB: abstracción del estado, funcionamiento y de otra información relevante del dispositivo. Existen elementos MIB específicos dedicados a representar implementaciones y configuraciones de algunos DIF más particulares.

En relación con este sistema, se puede esquematizar la comunicación entre un agente y un gestor como se muestra en la siguiente imagen:

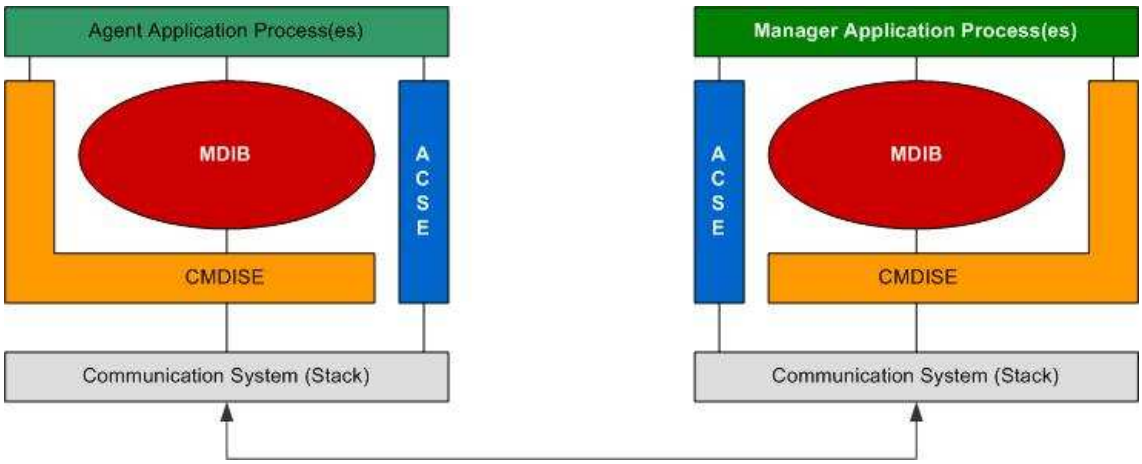


Ilustración 3.4: Esquema de comunicación entre agente y gestor

❖ MODELO DE SERVICIO

En este modelo es donde se definen comandos como el GET, SET, EVENTE REPORT, que son usados para intercambiar datos entre los agentes y los gestores. En este

estándar se definen los mensajes que pueden ser transmitidos y los instantes en los cuales pueden tener lugar. Los mensajes pueden ser de diferentes tipos:

- Mensajes destinados a iniciar y concluir interconexiones entre agentes y gestores.
- Mensajes a través de los cuales el gestor es capaz de acceder a la información del DIM de los agentes, leer sus atributos o controlar algunos aspectos de su funcionamiento.
- Mensajes enviados desde el agente hasta el gestor con los datos requeridos. Estos mensajes son denominados eventos y son usados para configurar la interconexión o para enviar información sobre las medidas.

A través de estos últimos mensajes el gestor puede realizar una copia de los objetos definidos en los agentes, lo cual es útil para conocer si el agente tiene una configuración definida en el estándar o no. Otra gran utilidad de esta característica es que se hace posible llevar a cabo la funcionalidad plug-and-play.

❖ MODELO DE INFORMACIÓN

Este modelo impone las normas referentes a la encapsulación, formatos y sintaxis usados/as en las capas de aplicación y transporte en las que se usa generalmente el estándar ASN.1 (Abstract Syntax Notation). Este estándar se usa para representar objetos con sus atributos. Su principal utilidad es proporcionar un sistema de representación de la información independiente de las técnicas de codificación específicas de cada dispositivo. Los objetos son convertidos en datos binarios usando las técnicas MDER (Medical Device Encoding Rules).

3.4. Normas IEC 60601

IEC 60601 es un conjunto de normas referentes a la seguridad de los dispositivos biomédicos. Concretamente la norma *IEC 60601-1 – 1 – Medical electrical equipment – Part 1: General requirements for basic safety and essential perfomance* difiere de la norma anterior (CEN ISO/IEEE 11073) en que este conjunto de estándares hace una referencia exclusiva a la seguridad eléctrica de los dispositivos, dejando de lado la definición de un sistema con sus elementos, formatos y protocolos de comunicación.

Los principales aspectos que se tienen en cuenta son:

- Evitar someter al paciente a corrientes, tensiones o campos que puedan resultar dañinos, tanto en condiciones normales como durante posibles

fallos, los cuales son situaciones críticas puesto que pueden producir corrientes o tensiones más intensas que las normales.

- Proteger al personal del hospital y a los pacientes de eventos anómalos pero posibles en su alimentación (sobretensiones).
- Avisar de riesgos de mal funcionamiento como bajadas de tensión de alimentación del dispositivo.
- Comprobar que los elementos del sistema funcionen correctamente.

En primer lugar, se explicarán los aspectos más importantes referentes a la protección de los pacientes y usuarios del sistema. A continuación, se mencionarán las medidas de seguridad posibles ante sobretensiones y pérdidas de tensión en la alimentación del sistema, mostrando dichas soluciones en imágenes que lo ilustren esquemáticamente.

Los aparatos y su entorno no deberán alcanzar temperaturas excesivas en uso normal o en condición de primer defecto, de modo que se eviten los riesgos por calentamiento del propio aparato, riesgos al entorno del aparato y riesgos directos sobre las personas. Para ello, se miden las temperaturas máximas alcanzadas en las distintas partes o componentes del equipo, mientras se encuentra en las condiciones de funcionamiento más desfavorables (a plena carga). En los ensayos correspondientes a condiciones anormales o condiciones de fallo se comprobará que tras sobrecargas, cortocircuitos, etc… el equipo no incumple ningún requisito esencial de seguridad.

❖ PROTECCIÓN FRENTE A FALLOS

Referente a la protección frente a fallos o comportamientos anómalos del sistema, se deberán tener en cuenta una variedad de posibilidades que pueden dar lugar a este tipo de funcionamientos, como las corrientes de fuga, la resistencia a la humedad, la resistencia al fuego o las líneas de fuga.

- Protección contra corrientes de fuga: mediante el ensayo de corrientes de fuga se pretende determinar la corriente que circularía por el cuerpo humano cuando se entrase en contacto con el equipo. La sensibilidad del cuerpo humano a las corrientes eléctricas depende del grado y naturaleza del contacto con el equipo, lo cual se clasifica por medio del grado y calidad de la protección en equipos tipo B, BF y CF.
  Los equipos tipo B definen un tipo de contacto con la piel tomando como referencia la tierra. Los equipos tipo BF también definen el contacto a través de la piel, pero toman como referencia una entrada flotante. Por último, los equipos CF son aquellos a través de los cuales se pueda establecer un camino directo al corazón del paciente.
  La forma de combatir este problema consiste en aislar las partes metálicas propensas a generar estas corrientes de fuga. Esto se logra recubriendo estas zonas con materiales aislantes, caracterizados por su rigidez dieléctrica, y conectando a tierra las zonas metálicas. Existen ciertos elementos del sistema que no pueden ser aislados tales como electrodos

o palas de desfibriladores, para los cuales la única precaución posible es establecer unos límites de tensión, corriente o capacidad.
Un ejemplo de corriente de fuga medida para los diferentes tipos de equipos son los siguientes:

Tipo de aparato	B,BF		CF	
Condición	Normal	Fallo	Normal	Fallo
Caja–tierra	0.1	0.5	0.01	0.5
Paciente–tierra	0.1	0.5	0.01	0.05
Línea–tierra	—	5	—	0.05
I electrodo AC	0.1	0.5	—	0.05
I electrodo DC	0.01	0.5	0.01	0.05

Tabla 3.3: Corrientes de fuga para distintos tipos de dispositivos

En cuanto al aislamiento, para asegurar que los equipos disponen de una rigidez dieléctrica apropiada, se realizan una serie de ensayos específicos que muestren la resistencia a su perforación. Para ello, los equipos se someten a tensiones con forma de onda y frecuencia similares a las que el equipo se sometería en condiciones normales.
Las siguientes ilustraciones muestran los distintos tipos de aislamiento posible:

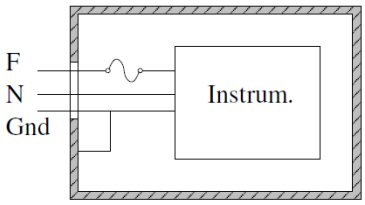


Ilustración 3.5: Conexión a tierra

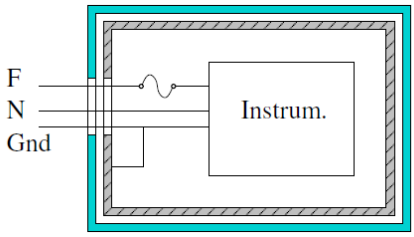


Ilustración 3.6: Aislamiento y conexión a tierra

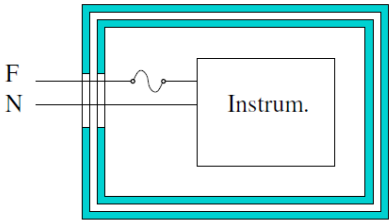


Ilustración 3.7: Doble aislamiento

- Resistencia a la humedad: Otra faceta a tener en cuenta en este tipo de precaución es la resistencia a la humedad y a la penetración de agua. En este sentido los equipos deberán estar clasificados siguiendo la nomenclatura IPXX, donde la primera X hace referencia al grado de penetración de polvo y la segunda al grado de penetración del agua.
  Respecto a esta precaución, los equipos son sometidos a un ambiente con una humedad relativa del 91-95% y una temperatura de 20-32ºC durante un periodo de tiempo que dependerá del tipo de dispositivo.
- Resistencia al fuego: el uso indebido del equipamiento médico es la segunda causa de incendios en hospitales. Por ello, en esta norma se especifica que los equipos deben tener la resistencia y rigidez necesaria

para evitar posibles incendios que puedan generarse durante un colapso total o parcial producido por los abusos a los que puedan someterse los equipos durante su normal uso.

De este modo, el sistema debe estar formado por elementos elaborados con materiales que minimicen el riesgo tanto de incendio como de propagación de la llama, tanto en su interior como en el exterior. Partes de los equipos como las carcasas, zonas externas aisladas o material aislante, que soportan partes activas incluyendo conexiones cuyo deterioro pueda ser causa de riesgo, deberán ser suficientemente resistentes al calor.

- Líneas de fuga y distancias en el aire: por último se atenderán a las distancias mínimas entre equipos referentes a las líneas de fuga y distancias en el aire. Se definen como dimensiones que hacen referencia al camino más corto entre dos partes conductoras o entre una parte conductora y una superficie accesible del equipo. Están condicionadas por factores como la tensión de trabajo, la polución de la superficie, la humedad, los materiales, las topografías de las superficies y las consecuencias de cualquier fallo del equipo. Las líneas de fuga concretamente son medidas a lo largo de la superficie de aislamiento, mientras que las distancias en el aire son medidas a través del aire.

❖ ANOMALÍAS EN LA ALIMENTACIÓN

Aunque normalmente la red eléctrica de los hospitales aporta 230 V de tensión, en ocasiones se pueden presentar picos de corrientes debidos a anomalías en la red eléctrica (*spikes*). Estos picos de corriente pueden deberse a:

- Cortocircuitos con líneas de tensiones más altas o con fases distintas de la normal (el contacto neutro-fase puede elevar la tensión hasta 380 V).
- Desconexión de una carga inductiva con corrientes altas sin las necesarias precauciones.
- La caída de rayos.
- Efectos inducidos por descargas cercanas a cualquier fenómeno de los anteriores.

Los circuitos de alimentación de los aparatos biomédicos deben garantizar que, en caso de presencia de sobretensiones, el paciente quede aislado de las mismas. En caso de que los equipos usados sean aparatos críticos, se debe garantizar que no se alterará su correcto funcionamiento.

Para solventar este problema existen una serie de medidas independientes que pueden incluso implementarse sobre un mismo dispositivo:

- La sobretensión puede evitarse mediante elementos en serie, entre los cuales es muy común encontrar los fusibles. Estos dispositivos se queman en presencia de corrientes altas (comunes en presencia de

sobretensiones) dejando el circuito abierto (impidiendo así la entrada de tensiones altas). Sus desventajas es que deben reemplazarse cada vez que se quemen y que no son muy eficaces frente a tensiones que suben lentamente. Existe otros dispositivos electromecánicos que pueden volver a cerrar el circuito y elementos llamados PTC que se reactivan automáticamente.

- También existen elementos que pueden proteger el circuito colocándolos en paralelo con este. Estos elementos son capaces de cortocircuitar la alimentación en presencia de una sobretensión, impidiendo así el paso de corriente al circuito. Se suelen usar con elementos serie para aumentar su eficiencia y tienen como principal ventaja su velocidad de reacción.
- En elementos más críticos como los equipos de tipo CF, debe existir algún elemento que aísle lo más posible el circuito frente a posibles anomalías. Estos elementos están basados generalmente en amplificadores de aislamiento. Estos amplificadores siguen el siguiente esquema:

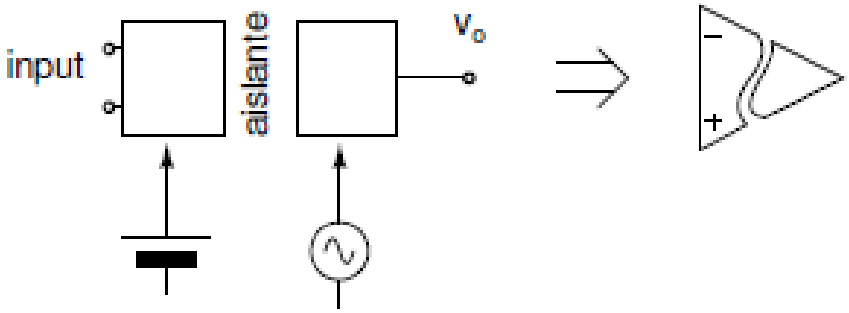


Ilustración 3.8: Amplificador de aislamiento

Existen diferentes formas de conseguir esquivar el aislante de los amplificadores como elementos ópticos, transformadores, o condensadores AT. Un ejemplo con elementos ópticos se muestra a continuación:

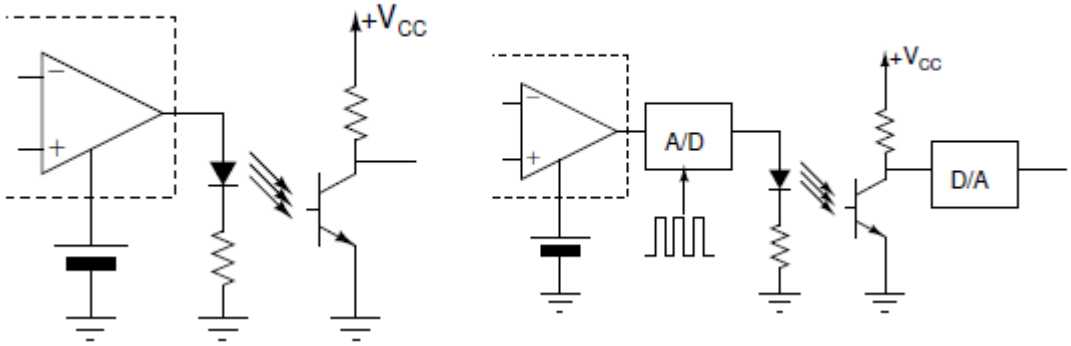


Ilustración 3.9: Aislamiento con elementos ópticos

El primero usa electrónica analógica (baja linealidad, barato) mientras que el segundo usa electrónica digital (buena linealidad, más caro y complejo por la sincronización).

Los transformadores aportan buena linealidad a un precio más económico, aunque tienen menor capacidad de aislamiento (mejorado con modulación). Un ejemplo es el que se muestra a continuación:

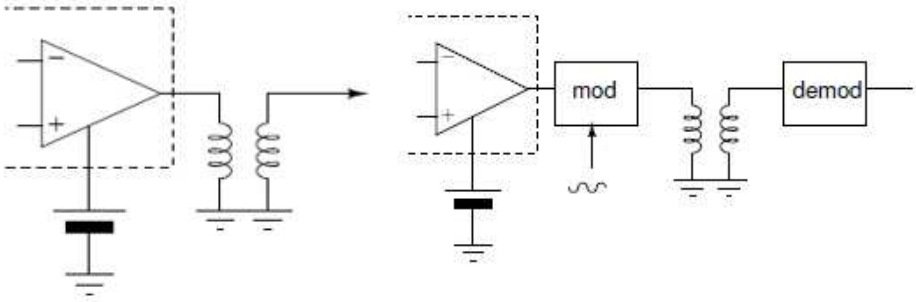


Ilustración 3.10: Aislamiento con transformadores

Los circuitos capacitivos presentan buen aislamiento y buena linealidad. Un ejemplo esquemático sobre su implementación es el siguiente:

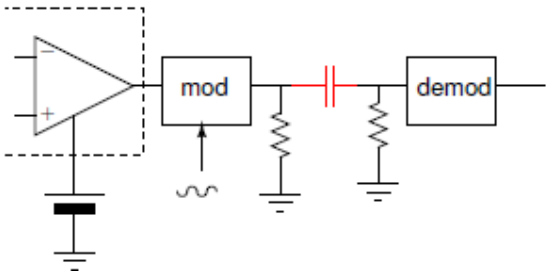


Ilustración 3.11: Aislamiento con elementos capacitivos

Por último, se mostrará un ejemplo de un circuito de aislamiento híbrido en el cual se combinan varias técnicas para conseguir mejores prestaciones. Este ejemplo presenta una linealidad perfecta, banda buena, buen aislamiento con alta versatilidad pero con un precio ligeramente alto:

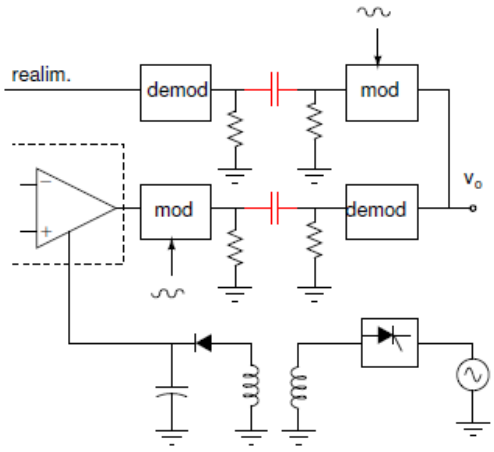


Ilustración 3.12: Aislamiento híbrido

❖ PROTECCIÓN FRENTE A BAJADAS DE TENSIÓN

Las bajadas de tensión pueden ser impulsivas o permanentes. En el primer caso suelen depender de sobrecargas instantáneas y transitorias de la red de alimentación o del mismo instrumento, las cuales pueden estar causadas por el retraso de los componentes tras una sobretensión o por la inductancia de pérdida en los cables de alimentación. Las bajadas de tensión permanentes se producen por fallos en la red de alimentación. En cualquier caso, se deben dar alarmas sobre el mal funcionamiento y activar una alimentación secundaria con la finalidad de no interrumpir el correcto funcionamiento del dispositivo.

Sin elementos capacitivos, un pico de corriente puede generar una bajada de tensión de alimentación. La capacidad proporciona la sobredemanda instantánea de corriente. Por ello es necesario añadir elementos capacitivos (condensadores cerámicos dan buen resultado) en paralelo al circuito, como se indica en la siguiente imagen:

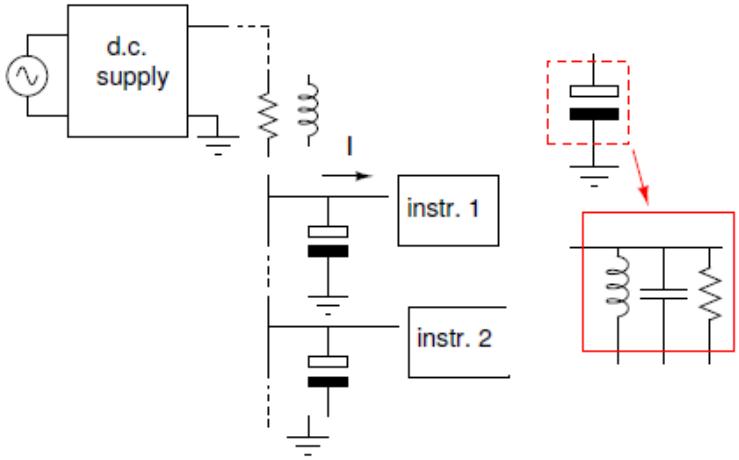


Ilustración 3.13: Protección con elementos capacitivos

Además, esta solución proporciona protección frente al ruido que pueda interferir en la propagación de las señales.

Un ejemplo de alarmas puede ser el siguiente:

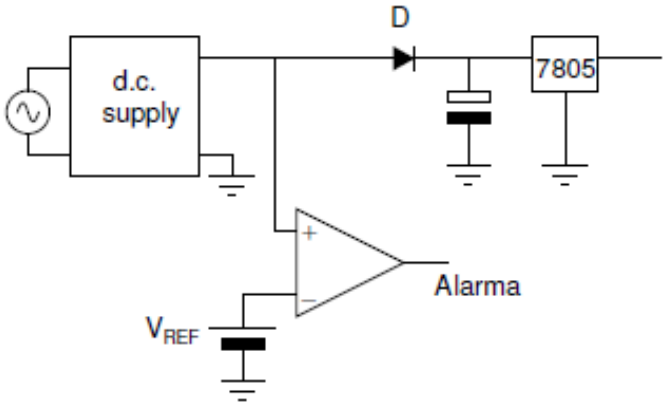


Ilustración 3.14: Protección con alarma

El diodo D tiene como finalidad evitar que la energía contenida en el elemento capacitivo se dirija hacia la fuente de alimentación.

Por último, se mostrará un ejemplo de alimentación secundaria (o alimentación *backup*):

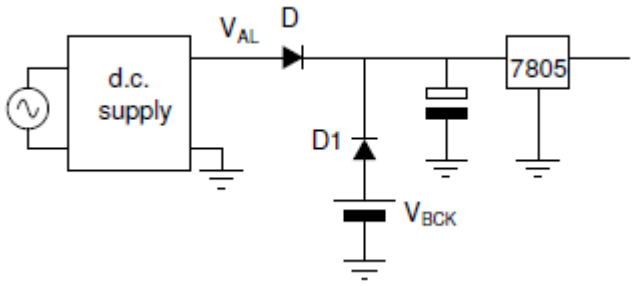


Ilustración 3.15: Protección con alimentación secundaria

El circuito anterior quedaría completado con el circuito de carga de la batería.

3.5. Instrumentos de radiofrecuencia

Las principales ventajas que ofrece el uso de un sistema inalámbrico son, como cabía esperar, la mayor comodidad en la toma de medidas por parte del personal médico y la disminución de cableado en el área del paciente, lo que se traduce en un aumento de confortabilidad de ésta. Por otro lado, el sistema podría aumentar la automaticidad de cara al almacenamiento inmediato de los datos tomados sin necesidad de tener que conectar ningún dispositivo a un ordenador central. Este hecho también se traduciría en una mayor flexibilidad para sistemas de asistencia móvil (ya sean ambulancias o sistemas de monitorización en viviendas para pacientes crónicos) y en el aumento de la facilidad de acceso al historial de datos de cualquier paciente, que puede agilizarse incluso a la consulta en tiempo real de las medidas tomadas en cada instante.

Además de los evidentes beneficios que este tipo de tecnologías aporta, también han de tenerse en cuenta una serie de inconvenientes acordes con la calidad del servicio. Este hecho se acentúa por el posible entorpecimiento de la interoperabilidad entre los dispositivos inalámbricos y los dispositivos que usen otros estándares como el IEEE 802.3 Ethernet para enlaces cableados, ya que podría reducir considerablemente la tasa de envíos afectando directamente a la calidad del servicio (QoS).

Respecto al uso de instrumentos de radiofrecuencia, se presenta como principal estándar la norma *CEN ISO/IEEE 11073. Part 00101: Guide - Guidelines for the use of RF Wireless technology.* Este documento expone una serie de beneficios, riesgos, indicaciones y recomendaciones sobre el uso de tecnologías de radiofrecuencia en

entornos hospitalarios. Existe a su vez un estándar elaborado por el mismo organismo e incluido en el mismo set de normas (CEN ISO/IEEE 11073) encargado de detallar los protocolos (de la capa de transporte) que se recomiendan usar para instrumentos que usen frecuencias infrarrojas, el *30300 Transport Profile: Infrared Wireless.*

El primero de estos documentos está más enfocado al desarrollo de un sistema completo que incluya dispositivos próximos al paciente, una red de comunicación que interconecte sensores con servidores y unos terminales encargados de almacenar, procesar o disponer los datos tomados, lo cual queda fuera del ámbito de este proyecto. No obstante también tiene una serie de indicaciones importantes de cara al uso de dispositivos inalámbricos que podrían añadirse al CS como parte de un posible trabajo futuro.

Dicho documento ofrece una serie de recomendaciones para los distintos grupos de trabajo que pueden intervenir en la elaboración de un sistema hospitalario basado en tecnologías de radiofrecuencia, con la intención de agilizar el proceso de desarrollo compatibilizando el trabajo de éstos. Estos grupos se dividen en términos genéricos como sigue:

- Proveedores de atención médica.
- Fabricantes y vendedores de dispositivos.
- Fabricantes y vendedores del equipamiento de soporte inalámbrico.
- Agencias gubernamentales.

Las recomendaciones persiguen generar una intercomunicación entre los grupos que sirva para llevar a cabo las tareas más fundamentales en el desarrollo de este tipo de sistemas, como identificar las tecnologías inalámbricas idóneas (teniendo en cuenta sus limitaciones, costes y disponibilidad de drivers), elaborar una implementación que sea práctica en el entorno escogido y evitar obstáculos durante el desarrollo. Para ello, se deberán evaluar temas como el tráfico de datos en dispositivos médicos o interconexiones entre dispositivos que usen diferente tecnología inalámbrica o tecnologías cableadas.

Las recomendaciones asociadas a los diferentes grupos son las siguientes:

- Dentro del grupo de proveedores de atención médica se ubican diferentes perfiles:
    - Personal médico: son los responsables de indicar nuevas necesidades y reportar fallos, así como de resumir el funcionamiento del sistema.
    - Personal técnico del hospital: se encargan de evaluar el sistema (comprando, instalando y comprobando el sistema), buscar carencias, localizar posibles fuentes de interferencias (como ordenadores portátiles, móviles, routers…) y realizar inventarios de dispositivos.
    - Ingenieros de IT: son los encargados de diseñar y dimensionar el sistema. Para ello, deben analizar la infraestructura (y sus capacidades) y las diferentes tecnologías inalámbricas sobre las que se pretende implementar el sistema. Asimismo, también tienen la responsabilidad de prever el tráfico que se debe soportar para conseguir un soporte físico satisfactorio en el transporte de datos. La detección de un uso no autorizado así como la implantación de protocolos de autenticación también recaen sobre este grupo.

    - Coordinadores de espectro: son los encargados de asignar las bandas espectrales de los hospitales.
    - Administradores: se encargan de analizar los informes del resto de personal de salud y de evaluar si las peticiones son viables.
- Fabricantes de dispositivos médicos: son los encargados de desarrollar los dispositivos inalámbricos. Sus responsabilidades abarcan especificar unos parámetros que exijan un QoS concreto para cada dispositivo, definir requisitos como los tipos de datos, la latencia del sistema y la tasa de envío, indicar los anchos de banda necesarios y proveer sistemas de evaluación de los propios dispositivos.
- Fabricantes de equipos inalámbricos: tienen la responsabilidad de encuadrar el funcionamiento de los sistemas inalámbricos con las necesidades del personal del hospital en lo referente a los requerimientos implícitos de los equipos usados y de la calidad deseada (QoS). De mismo modo deben de definir y configurar los parámetros necesarios e informar al resto de grupos sobre el funcionamiento del sistema bajo cada una de las posibles situaciones que puedan tener lugar.
- Vendedores de equipos inalámbricos: son los encargados de evaluar el entorno en el que se implantará el sistema y de diseñar el despliegue de éste basándose en la cobertura de los dispositivos, en los posibles tipos de dispositivos que pudieran aparecer en un futuro para hospitales, en el tráfico de datos previsto, en las prioridades de los diferentes tipos de peticiones que serán generadas (alertas, medidas…) y en las posibles tareas referentes a EMI (interferencias electromagnéticas) o a EMC (compatibilidad electromagnética).

La finalidad de este apartado es dar una serie de directivas sobre las precauciones y las recomendaciones que deben seguirse si se añadieran sensores que usen tecnologías inalámbricas. Por ello, los aspectos más relevantes dentro de las normas referentes al uso de tecnologías inalámbricas en hospitales y entornos biomédicos serán las que abarquen temas como compatibilidad entre diferentes tecnologías, interferencias, seguridad y riesgos, sin llegar a entrar en temas de producción, topologías, tipos de instalación (dependiendo del entorno) o jerarquías de un sistema de comunicación completo.

En lo referente a compatibilidad entre sistemas, está indicado explícitamente que no existen estándares oficiales que ofrezcan recomendaciones sobre cómo llevar a cabo una óptima interconexión entre dispositivos que usen tecnologías normalizadas (como 802.11) y dispositivos que usen tecnologías o protocolos propios, de modo que la responsabilidad de que todo quede perfectamente interconectado recae en el desarrollador. No obstante, sí se detalla que los requisitos QoS de los sistemas IEEE 802.11 proporcionan una calidad garantizada y se recomienda su implementación a la hora de dar prioridades especiales a tramas importantes como las alertas (aunque dicha norma fuera desarrollada para soportar transmisiones de vídeo y audio inicialmente y no garantice por tanto una solución general para abastecer los requisitos QoS de dispositivos médicos). En definitiva, como en el caso de las interferencias, deberá ser el usuario final (en este caso los ingenieros clínicos o ingenieros IT) el que tome las decisiones sobre cómo desarrollar una compatibilidad dentro del sistema general.

Dentro de los hospitales, existen una gran cantidad de sistemas que usan un amplio rango de tecnologías RF tanto estandarizadas como no. Las pruebas referentes a las interferencias que pueden darse entre los dispositivos de un hospital y fuentes externas se han estado llevando a cabo en los últimos años (debido al auge del uso de tecnología inalámbrica en hospitales que ha habido recientemente). Un ejemplo de la importancia que tiene este fenómeno se dio en un hospital en Dallas en 1998, donde se desarrolló un sistema que usaba tecnología inalámbrica que funcionaba en unas bandas de frecuencias similares a las que usaban algunos canales de televisión digital. La presencia de una antena de televisión en las proximidades del hospital generó que el 50% del sistema se bloqueara por las interferencias, lo cual evidenció el extremo cuidado que se debe tener a la hora de asignar las bandas de frecuencia de funcionamiento de estos dispositivos.

Por ello, para poder desplegar una red inalámbrica, que posibilite el uso de este tipo de sensores, será necesario encontrar unas bandas de frecuencia en las que no se encuentren otros sistemas con antenas emisoras operando a una potencia media o alta, cuyas emisiones radioeléctricas no conlleven un riesgo y que puedan garantizar una plena compatibilidad electromagnética entre el equipamiento utilizado y las redes desplegadas.

Un sistema que basado en los estándares 802.11b, 802.11g o 802.11n proporcionaría buenos resultados en un corto espacio (50 metros al aire libre), aunque la cobertura dentro de un hospital se vería degradada por su estructura (ascensores, zonas de aislamiento, gran cantidad de placas metálicas…). Es por ello que deberá realizarse un buen dimensionado de cobertura para ubicar correctamente las antenas y establecer así las frecuencias de funcionamiento (actualmente en Europa hay 13 canales disponibles). Dentro de este sistema, los sensores deberán ajustarse para alejarse lo máximo posible de las frecuencias de funcionamiento del resto de equipos. Otro detalle a tener en cuenta será la protección de los datos para preservar su privacidad ya que los sistemas inalámbricos son mucho más vulnerables a posibles ataques que los cableados.

Otra instalación piloto, realizada en el hospital Niño Jesús, sirvió para realizar un estudio en el que se tomó, entre otras conclusiones, que las instalaciones de sistemas WLAN como las definidas anteriormente no son perjudiciales para la salud de los pacientes ya que los rangos de frecuencias usados están muy por debajo de los considerados perjudiciales (no se detectó ninguna relación causa-efecto entre la exposición a estas radicaciones y patologías conocidas). Por otro lado también se detalló en este estudio que no existen interferencias entre WLAN, servicios de comunicación clásicos e instrumentos médicos propios del entorno.

Capítulo 4

Diseño del sistema

4.1. Introducción

Este capítulo tiene como finalidad realizar una explicación detallada sobre todo el proceso de diseño del Concentrador de Señales, exponiendo los requisitos fundamentales del sistema y proponiendo soluciones a los problemas que se presenten.

Para ello se detallará la evolución en las distintas propuestas sobre el diseño hardware, proponiendo y presentando las características de un diseño final.

A continuación se presentará el diseño software, exponiendo las partes en las que se dividirá el software como módulos de operación, detallando las funciones de cada uno de ellos.

Por último, se expondrán los requisitos de alimentación que tendrá el sistema en general de cara a realizar un posterior dimensionado de la batería que será requerida.

4.2. Diseño Hardware

En este apartado se realiza el diseño del hardware del sistema.

En primer lugar se expondrán los requisitos que debe cumplir y, en consecuencia, los componentes necesarios para su elaboración.

Seguidamente se resumirán las diferentes propuestas de diseño explicando en última instancia el que ha sido elegido como definitivo.

4.2.1. Requisitos del sistema

El primer paso para la elaboración del CS es elaborar una lista con las especificaciones que debe tener para llevar a cabo sus funciones correctamente. Estas especificaciones son:

- Los sensores deben estar homologados para uso hospitalario.
- El número máximo de sensores que se pueden conectar por cable al CS será de 4.
- Debe ser configurable para distintos tipos de sensores.
- Los sensores serán conectados al CS a través de un hub USB, que hará de interfaz de entrada de datos.
- El CS deberá tener conectividad Bluetooth para enviar los datos procesados de los sensores a una interfaz de usuario.
- El peso del CS debe ser reducido.

4.2.2. Componentes necesarios para el diseño

El Concentrador de Señales estará compuesto de varios elementos fundamentales en todo tipo de sistemas basados en microcontrolador o microprocesador, así como de una serie de elementos destinados a cumplir las especificaciones del equipo a diseñar:

- Procesador (familia ARM).
- Memoria RAM superior a 512 MB.
- Almacenamiento de datos, ya sea usando discos duros o usando ranuras para tarjetas microSD de almacenamiento masivo.
- Batería o módulos de alimentación para el suministro de energía del sistema.
- Conector de alimentación para el cargador de la batería.
- Interfaz para el modo autónomo del concentrador, que incluirá una pantalla y una serie de botones que sirvan para llevar a cabo las diferentes acciones a emprender por el personal médico.
- Módulo de conexión inalámbrica Bluetooth, que permita la conexión del concentrador con la interfaz de usuario.
- Puertos USB que permitan la conexión de los sensores por cable.
- Sistema operativo capaz de gestionar los elementos Software del sistema.
- Plataforma para desarrollar o depurar el software del dispositivo.
- Módulos de conversión y acondicionamiento de señal.

4.2.3. Evolución del diseño

Una vez definidos los requisitos y los elementos a incluir en el diseño HW se procede al diseño del hardware del subsistema.

❖ Concentrador de Señales basado en PCB

La primera propuesta de diseño consistió en la elaboración de un PCB (Printed Circuit Board, tarjeta de circuito impreso) que incluyera todos los elementos hardware necesarios. Este circuito impreso se habría diseñado y elaborado en el laboratorio HCTLab de la Escuela Politécnica Superior de la UAM.

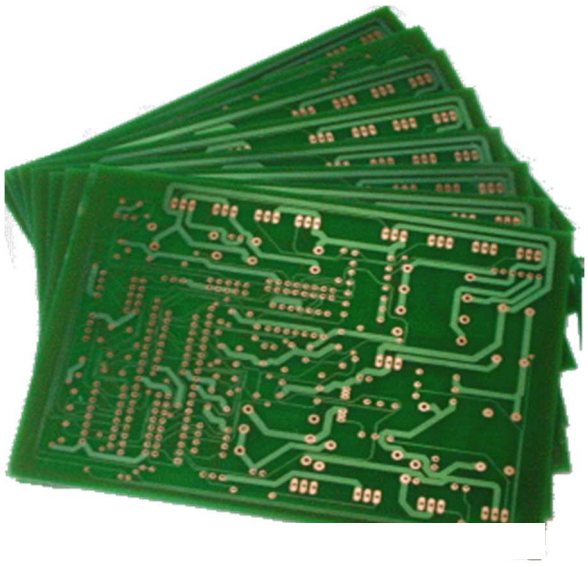


Ilustración 4.1: PCBs

- Diseño: En este punto se hubiera usado como software de diseño electrónico el programa ***Altium Designer***. Dicho programa, además además de ofrecer todas las herramientas necesarias para el desarrollo electrónico, es capaz de generar los archivos necesarios para la elaboración de la tarjeta de circuito impreso física.
- Elaboración: La elaboración del primer prototipo del Concentrador de Señales se llevaría a cabo en el *taller de circuitos impresos* de la Escuela Politécnica Superior de la UAM. En este taller se hace uso de una máquina de prototipo de PCB de la empresa LPKF: la **plotter fresadora protoMat s100**, que que es capaz de realizar circuitos impresos a partir de los archivos generados por el programa Altium Designer. Una vez elaborada la tarjeta impresa se soldarían los elementos hardware en su superficie.
- El firmware del CS se elaboraría con herramientas software de uso libre.

Ilustración 4.3: LPKF s100

Ilustración 4.2: Altium Designer

❖ Concentrador de Señales usando sistemas basados en microcontrolador

La propuesta anterior fue mejorada con la idea de usar sistemas basados en microcontrolador comerciales. Esta idea supone una mejora por varias razones:

- Usar dispositivos comerciales garantiza unas prestaciones mínimas por parte del fabricante.
- Supone un ahorro significativo en el diseño del sistema completo.
- Permite ahorrar costes si se elige una tarjeta económica, adecuada a las prestaciones que se requieren.
- Limita el diseño a la incorporación y programación de periféricos a dicho sistema, y a la elaboración del software necesario para llevar a cabo las acciones requeridas por el CS.

El sistema basado en microcontrolador sería la tarjeta BeagleBone, que incluye un microprocesador ARM Cortex A8, ranura para tarjetas microSD (a través de la cual se instalaría el sistema operativo), memoria RAM de 256 MB, alimentación con puerto USB, puerto microUSB para conexión de periféricos, Ethernet *on the chip*, puerto de salida HDMI y sistema operativo Linux Angstrom.

La tarjeta BeagleBone sería la encargada de recibir la información procedente de los sensores por medio de su puerto microUSB. Para ello, se usaría un *hub* USB que hiciera posible el uso de varios sensores USB a la vez y de un USB con antena Bluetooth para los sensores inalámbricos. Los datos procedentes de los sensores serían analizados y gestionados desde un software diseñado con ese fin en Linux. Los nuevos datos (obtenidos tras la ejecución de dicho software) serían empaquetados y enviados a la red IP a través de la conexión Ethernet que ofrece la tarjeta.

Ilustración 4.4: Tarjeta Beaglebone

Para el modo autónomo del CS (caso en el que el terminal de destino de los datos no estuviese disponible) se añadiría un pequeño display con varios botones con el fin de ofrecer un interfaz alternativo al usuario.

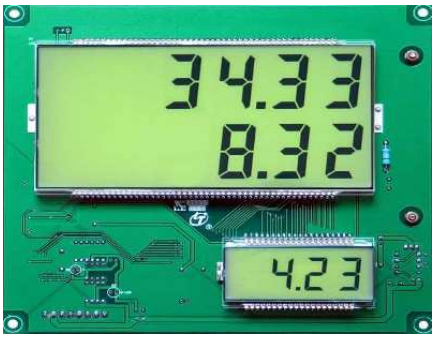


Ilustración 4.5: Display

❖ Sistema compuesto de Tablet y tarjeta de adaptación

Dado el precio y prestaciones actuales de las Tablet, se propone el diseño del CS usando una de éstas como elemento central. El principal motivo es la disponibilidad de varios elementos hardware necesarios para el diseño del CS dentro de la Tablet, además de que la interfaz con el usuario es infinitamente mejor que el display del diseño anterior, haciendo que la toma de medidas en modo autónomo sea mucho más amigable. Una Tablet estándar incluye elementos como una CPU, una pantalla, un teclado, una conexión Wifi y ranuras de conexión de tarjetas de almacenamiento de datos, útiles para introducir el software necesario para su correcto funcionamiento. La presencia de estos elementos en la Tablet simplificaría mucho el diseño del CS, limitando dicho diseño a la elaboración de una tarjeta de adaptación que incluyera las interfaces hardware necesarias para la interconexión del CS con los sensores. Esta interfaz hardware incluiría varios puertos USB para los sensores de este tipo, así como un módulo de transferencia de datos Bluetooth para los sensores inalámbricos. Para la interconexión de la Tablet con la tarjeta de adaptación se incluiría un puerto USB con función USB OTG, a través del cual se transferirían las medidas tomadas por los sensores.

Ilustración 4.6: Energy Tablet s7

La Tablet elegida para este diseño provisionalmente fue la Energy Tablet s7 Dual, desarrollada por Energy System. Al final de este documento se anexionan las especificaciones tanto de esta Tablet como de las que fueron consideradas como candidatas.

La tarjeta de adaptación se habría elaborado como un PCB, usando el programa *Altium Designer*, según el proceso descrito en el apartado 4.1.

❖ Sistema compuesto de Tablet y sistema basado en microcontrolador

El diseño basado en Tablet y tarjeta de adaptación tiene como principales problemas los siguientes:

- Todas las operaciones a realizar sobre las medidas tomadas por los sensores deben ser procesadas por la Tablet, cuya principal motivación era ofrecer una interfaz para el modo autónomo.
- El software del sistema se programaría mediante aplicaciones Android, que presentan dos inconvenientes muy importantes respecto a cualquier programa de Linux: tienen menor rendimiento ya que los programas escritos en Java tardan más en ejecutarse que los programas escritos en C, y se darían problemas de compatibilidad con el software de los sensores inalámbricos (escritos en C y testados y comprobados sobre Linux).

Por estos motivos, se propone un sistema que combine las ventajas de los sistemas presentados en 4.2 y 4.3. Este sistema se compondría de un sistema basado en microcontrolador (se volvería a optar por la tarjeta BeagleBone) encargado de recibir los datos procedentes de los sensores tanto USB como inalámbricos, y por una Tablet conectada a la tarjeta (se volvería a optar por la Energy Tablet s7 Dual) encargada de mostrar los datos de los sensores y de ofrecer una interfaz con el usuario para el modo autónomo.

Este sistema permitiría dividir el diseño del software en dos bloques:

- Software destinado a analizar y gestionar los datos procedentes de los sensores. Este software desensamblaría las tramas procedentes de los sensores, aislaría los datos de interés clínico (pulso, tensión, temperatura…) y empaquetaría dichos datos para enviarlos a través de la red IP según un protocolo de aplicación desarrollado con ese fin. Se desarrollaría en el sistema basado en microcontrolador, es decir, sobre Linux, mediante programas escritos en C.
- Software destinado a recibir las medidas (tomadas por los sensores) ya analizadas y mostrarlas por pantalla usando un interfaz que incluya: autenticación del paciente y del clínico, las medidas tomadas por todos los sensores (indicando con un código de colores si están en rango o no), el estado de las conexiones de los sensores y el estado de la batería y de la conexión Wifi. Se desarrollaría mediante aplicaciones Android en un ordenador que disponga de un entorno de programación (IDE) destinado a dichas aplicaciones (Eclipse IDE, por ejemplo) y sería ejecutado en la Tablet.

También es digna de mención la edición que debe hacerse del propio sistema operativo Android con el objetivo de realizar un arranque del sistema personalizado y de

que se carguen únicamente las aplicaciones clínicas desarrolladas exclusivamente para este proyecto.

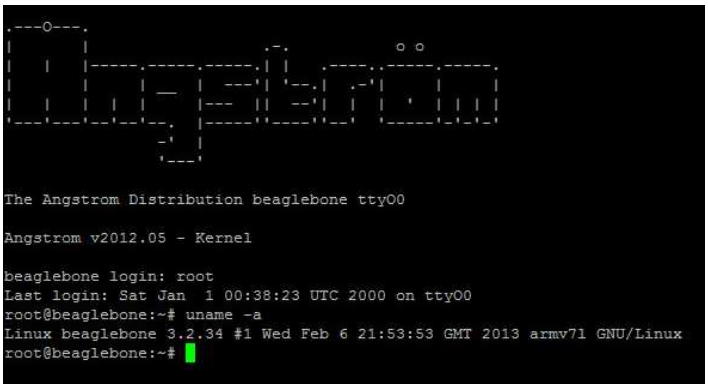


Ilustración 4.7: Linux Angstrom

4.2.4. Características del diseño final

El Concentrador de Señales que será desarrollado finalmente es el correspondiente al diseño expuesto en el apartado 3.2.3.4. Este diseño se divide en dos grandes bloques: la Tablet y el sistema basado en microcontrolador, que a partir de aquí será referida como tarjeta de procesado. Las características de la Tablet se indican a continuación.

❖ Interfaz en modo autónomo: Tablet

En este diseño, la Tablet tiene como principal función ofrecer una interfaz para el modo autónomo con el usuario, cediendo gran parte de las funciones de procesado a la tarjeta de procesado. Las características necesarias del Tablet son las siguientes:

- CPU con Android 4.1.
- Memoria RAM de al menos 512 MB.
- Memoria Flash de al menos 4 GB.
- Pantalla de 7''.
- Wifi g/n.
- Bluetooth 4.
- Entrada de alimentación independiente del cable USB.

Las características opcionales, para dotar de mejoras al sistema:

- Cámara Web frontal.
- Altavoz integrado.

Para la elección de la Tablet se ha optado por usar una comercial, entre las que han destacado como candidatas las siguientes:

1. BQ Elcano: cuyas características más relevantes son las siguientes:
    a. CPU Dual Core Cortex A9 1 GHz.
    b. Memoria RAM: 1 GB.
    c. Pantalla de 7''.
    d. Memoria: 16 GB.
    e. Android 4.0.
    f. WiFi 802.11 b/g/n.
    g. Bluetooth 4.0.
    h. Micro-USB 2.0 OTG.
    i. Entrada de 5V – 2A.
    j. Slot micro-SD.
    k. Precio: 199,90€.
2. Leotec Space III: cuyas características más relevantes son:
    a. CPU Boxchip A13 1,2GHz (Cortex A8).
    b. Memoria RAM: 512 MB.
    c. Pantalla de 7''.
    d. Memoria: 4 GB.
    e. Android 4.0.
    f. WiFi 802.11 b/g/n.
    g. Micro-USB 2.0 OTG.
    h. Entrada 5V DC.
    i. Slot micro-SD.
    j. Precio: Depende del punto de venta.
3. Energy Tablet s7: cuyas características más relevantes son:
    a. CPU Dual Core Cortex A9 1,6 GHz.
    b. Memoria RAM: 1 GB.
    c. Pantalla de 7''.
    d. Memoria: 8 GB.
    e. Android 4.1.
    f. WiFi 802.11 b/g/n.
    g. Bluetooth 4.0.
    h. Función USB OTG.
    i. Alimentación mediante coaxial XVDC 2.000mA.
    j. Slot micro-SD.
    k. Precio: 139,00€.

De entre las opciones anteriores, la primera en ser descartada fue la Leotec Space III debido a la ausencia de Bluetooth y a su indisponibilidad por parte del fabricante (el diseño del CS dependería de la disponibilidad de los puntos de venta de esta Tablet).

Finalmente, entre la BQ Elcano y la Energy Tablet s7 Dual se optó, como ya se indicó en apartados anteriores, por la Energy Tablet s7 debido a que teniendo unas prestaciones similares es notablemente más barata.

La hoja de características de la Energy Tablet s7 Dual está en el Anexo B.

❖ Tarjeta de procesado

La tarjeta de procesado es la parte del diseño encargada de realizar todo el tratamiento de la información procedente de los sensores. Sus principales tareas serán recibir y desempaquetar los datos, procesándolos y enviándolos (ya sea a través de la red IP o por medio de Bluetooth) a través un protocolo de comunicación propio que deberá ser diseñado y desarrollado para tal fin basado en un sistema cliente-servidor.

Éste constará de un módulo de entrada formado por un hub USB con capacidad suficiente para conectar todos los sensores USB que se deseen así como una entrada Bluetooth para sensores inalámbricos, un módulo de procesado formado por el sistema embebido y un módulo de salida por el que se puedan visualizar los datos medidos y procesados.

El esquema de la tarjeta de procesado es el siguiente:

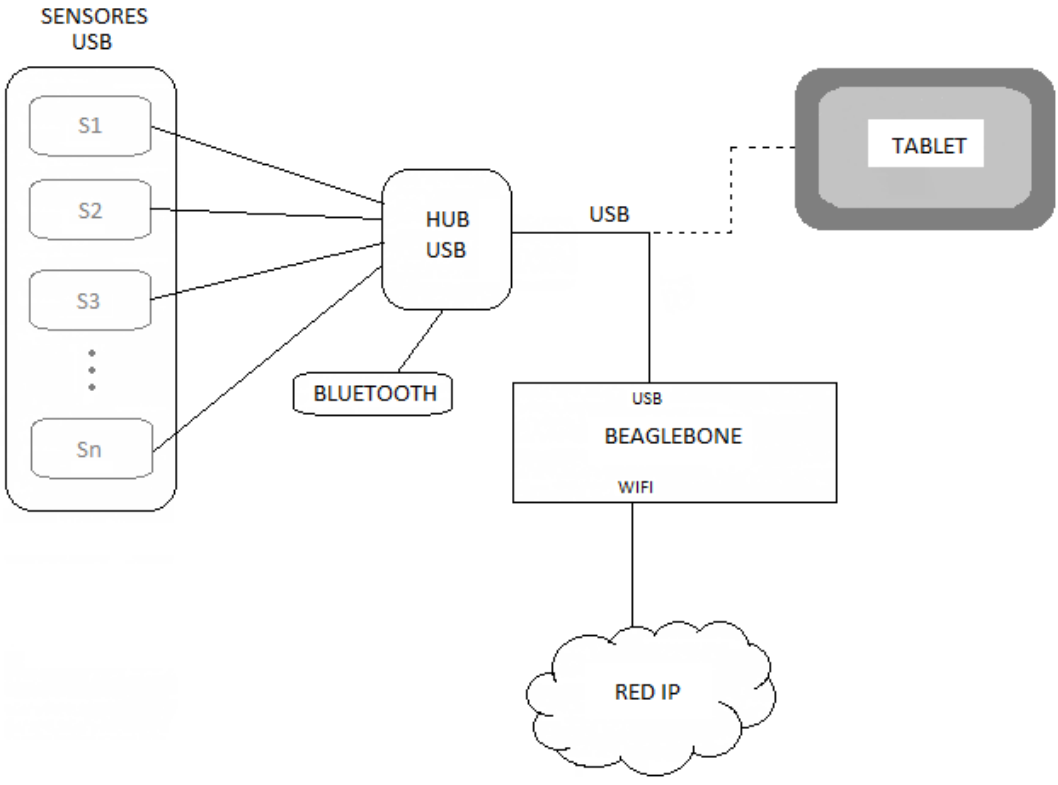


Ilustración 4.8: Tarjeta de procesado

El esquema anterior se resume a continuación:

1. El corazón del diseño será la tarjeta BeagleBone, un sistema basado en microcontrolador con un puerto USB por el que entrarán los datos de las constantes vitales.
2. Este sistema llevará instalado como sistema operativo un Linux embebido (distribución Angstrom) sobre el que se elaborará la parte del diseño software encargada del procesado de datos.
3. Los datos provendrán de un hub USB que recibirá los datos de los sensores USB.
4. En el caso de ampliar el diseño con sensores Bluetooth, éstos serán gestionados desde un adaptador Bluetooth-USB que se conectará a la tarjeta o al hub USB.

4.3. Diseño Software

En este apartado se lleva a cabo el diseño del software del sistema.

Como en el apartado del diseño hardware, se comenzará resumiendo los requisitos básicos del software para posteriormente definir las partes en las que se dividirá el diseño.

Por último, se realizará una descripción de cada una de las partes del software y una explicación esquemática detallada del sistema finalmente diseñado.

4.3.1. Requisitos del sistema

La propia definición hardware del CS y sus especificaciones funcionales dan lugar a una serie de requisitos imprescindibles a la hora de realizar el diseño software del sistema. Estos requisitos son:

- El software debe ser capaz de admitir nuevos sensores.
- El software tiene que tener la capacidad de trabajar con varios sensores al mismo tiempo.
- El sistema debe ser capaz de recibir datos de sensores conectados por medio de una interfaz USB.

- Los datos obtenidos deben ser desensamblados y tratados para su consecuente envío al monitor.
- Para poder llevar todas sus labores correctamente, el sistema debe ser modular, es decir, debe estar compuestos de apartados específicos encargados de realizar tareas concretas.
- Las operaciones llevadas a cabo por el software no deben dar lugar a un consumo excesivo de energía ni de recursos del sistema.

4.3.2. Componentes del diseño

Como ya se ha indicado en el apartado de requisitos del sistema, el software del sistema debe ser modular con el fin de poder desarrollar un sistema formado a partir de subsistemas independientes entre sí. Dichos subsistemas deben ser capaces de abordar las principales tareas del Concentrador de Señales, a saber: recibir los datos de los sensores USB, procesarlos y enviarlos desensamblados y procesados al dispositivo que contenga la interfaz de usuario.

En el caso de los componentes encargados de realizar transferencias de datos entre diferentes dispositivos (recibo y envío de datos) se hará uso del sistema cliente-servidor. En cada caso, el papel del servidor lo llevará a cabo el dispositivo que contenga los datos deseados y el cliente, el que los requiera. Las siguientes figuras muestran los dos tipos de conexiones de las que hará uso el CS:

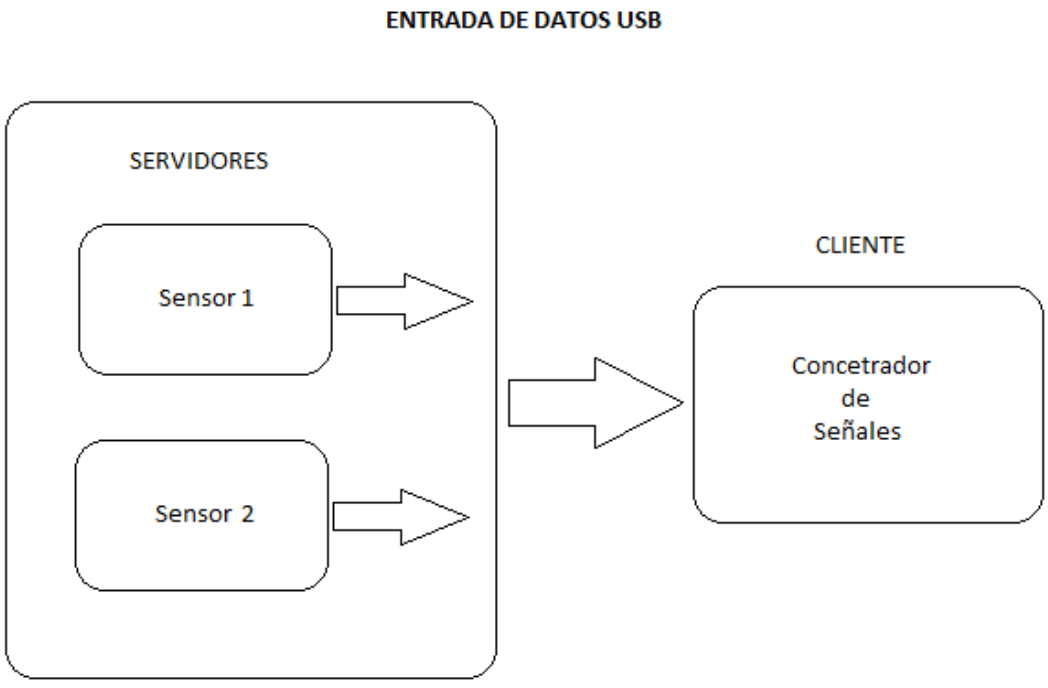


Ilustración 4.9: Entrada de datos

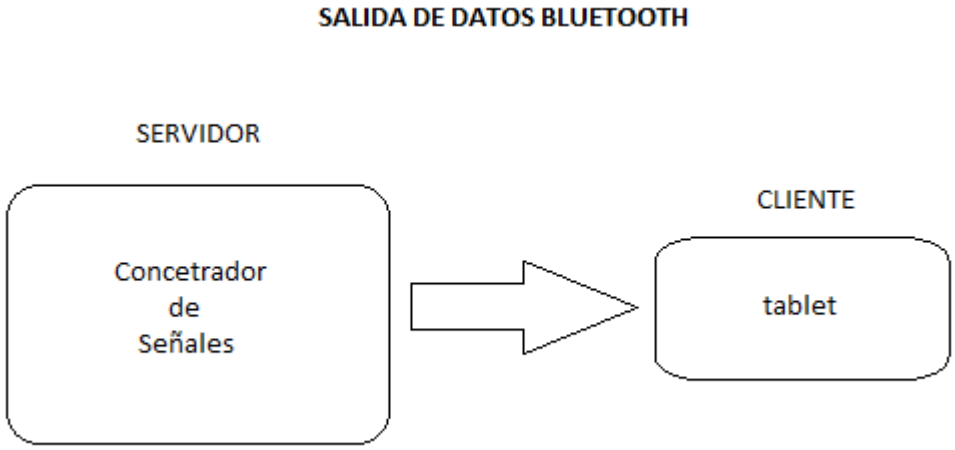


Ilustración 4.10: Salida de datos

En el primer caso, el programa servidor de los sensores ya estará integrado en los propios sensores, por lo que el trabajo se centrará en crear un programa cliente compatible con dicho programa servidor.

Por lo tanto, dentro del software del Concentrador de Señales debe haber un subsistema encargado de llevar a cabo cada una de estas tres tareas:

1. Módulo de entrada de datos: será el encargado de comprobar la conexión de periféricos USB al CS y de realizar las tareas necesarias para generar una correcta transferencia de datos desde los sensores. Los datos de entrada serán almacenados en un fichero que será usado como base para el siguiente módulo.
2. Módulo de procesado: será el encargado de tomar los datos generados por el módulo de entrada de datos y de procesarlos para generar una serie de datos que incluyan únicamente la información útil (medidas, fecha, hora) proporcionada por los sensores, desechando datos propios del protocolo de comunicación del sensor. Dicha información será almacenada en un fichero que será usado como base para el siguiente módulo.

3. Módulo de salida: será el encargado de enviar los datos al terminal de usuario por medio de Bluetooth. En este caso, será necesario programar un cliente y un servidor que funcionarán en el terminal y en el CS respectivamente. Para llevar a cabo dicha tarea, será necesario elaborar un protocolo de comunicación entre ambos. Éste será el punto más importante en la elaboración de este módulo.

4.3.3. Módulo de entrada de datos

Como ya se ha explicado, este módulo será el encargado de conectar los sensores con el Concentrador de Señales de modo que se pueda llevar a cabo la correcta transferencia de datos entre ambos dispositivos.

❖ Subsección de gestión de conexiones

Al comienzo del apartado de diseño software, se indicó entre los requisitos básicos que el CS debía ser capaz de manejar varios sensores al mismo tiempo. Para ello, el módulo será implementado como un programa multiproceso modelado como una máquina de estados con el siguiente esquema:

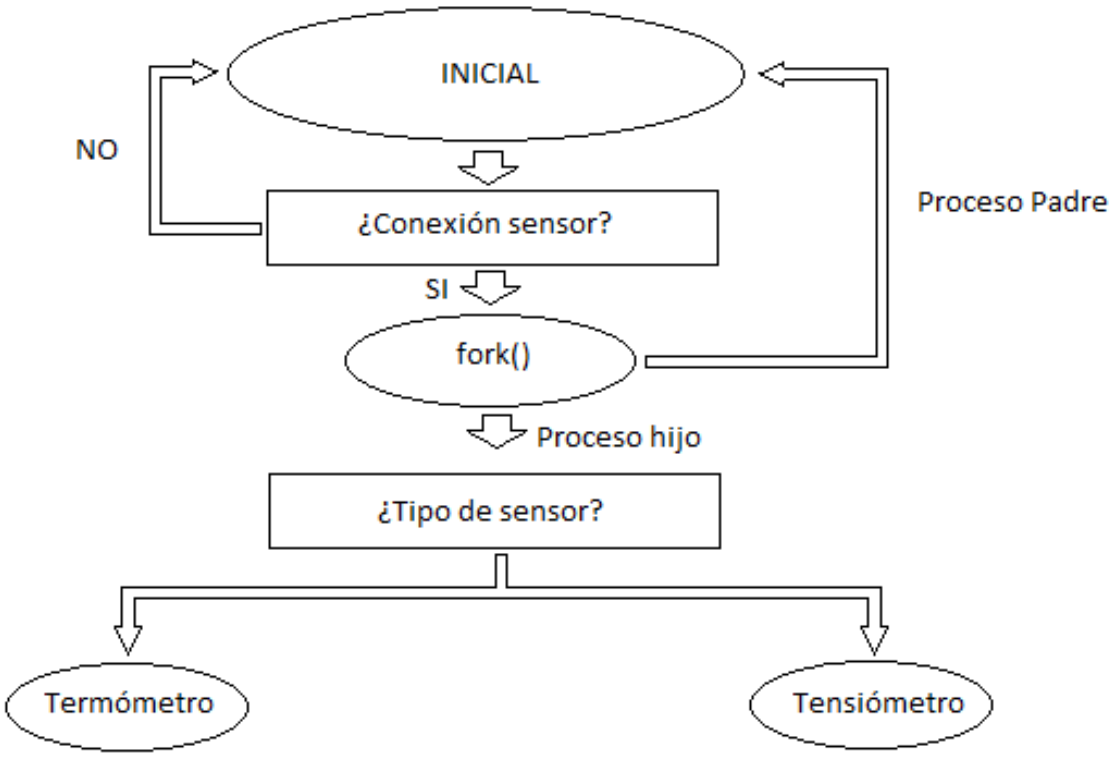


Ilustración 4.11: Gestión multiproceso

La sincronización entre los diferentes procesos (padre e hijos) se realizará por medio de semáforos y la comunicación entre ellos por medio de una zona de memoria compartida.

❖ Subsección de comunicación USB

La gestión explicada en el punto anterior da lugar a un proceso específico por cada sensor que esté conectado encargado de gestionar la comunicación con el sensor que le corresponda. La única diferencia que se da entre los sensores a este nivel es el protocolo de comunicación que usan para su conexión con un PC, por lo que el diseño de las estructuras y el flujo de datos interno en el Concentrador de Señales podrán ser implementados de la misma manera:

- En primer lugar, se deben almacenar en una base de datos tanto los datos a transferir como las características (o parámetros) de cada transmisión.
- Esta información será obtenida mediante ingeniería inversa realizando transmisiones con un analizador USB.
- Dichos datos y características se almacenarán en un fichero inalterable, que será leído por el programa guardando la información en un TAD.
- El TAD será consultado en el momento previo a realizar cada transmisión.
- Los datos recibidos deberán ser almacenados importando el orden de llegada (TAD cola) y serán guardados en otro archivo que servirá como entrada para el módulo de procesamiento.
- El driver a nivel general, debe valer para cualquier tipo de dispositivo, y por tanto, debe ser capaz de soportar cualquier tipo de envío: control, bulk, iterrupt e isochronous transfers.

Esquemáticamente, la sección de comunicación USB tendrá la siguiente estructura:

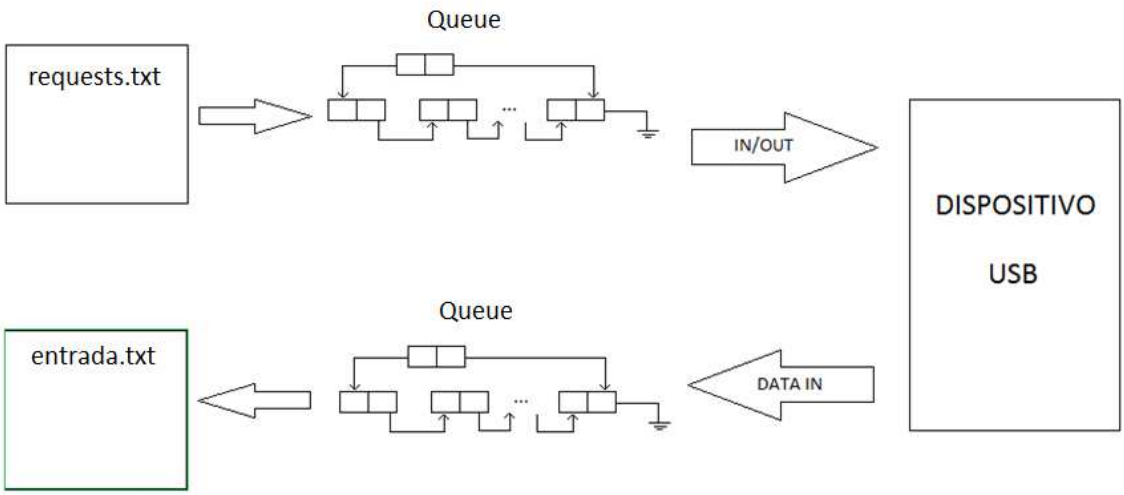


Ilustración 4.12: Comunicación USB

El programa, de este modo, deberá generar un archivo *requests.txt* donde se indiquen los parámetros y datos necesarios para la comunicación USB. A continuación un TAD cola (en inglés *queue*) se encargará de almacenarlos en memoria dinámica para poder usarlos más versátilmente durante los envíos. Finalmente, los datos recibidos se irán almacenando en otro TAD cola según se vayan recibiendo para posteriormente registrarlos en otro fichero *entrada.txt* (marcado en verde) que será usado por el módulo de procesado.

4.3.4. Módulo de procesado de datos

En este punto, el material del que se dispone es un archivo que contiene los datos que ha enviado el sensor USB en cuestión. En principio, estos datos incluyen tanto los datos de interés como los datos de control de las transmisiones USB.

El objetivo de este módulo es generar un fichero de salida (que será llamado *salida.txt*) en el que se encuentren los datos y parámetros necesarios para llevar a cabo las transmisiones por medio del Bluetooth.

❖ Subsección de desensamblado y extracción de datos

Para llevar a cabo la extracción de los datos de interés de entre todo el conjunto de datos recibidos, es necesario desensamblarlos en primer lugar. Este paso será implementado de diferente forma para cada sensor debido a que los datos a procesar en este punto son los que ha generado el sensor directamente. Por ello, el paso previo será descifrar por medio de ingeniería inversa los protocolos de comunicación de cada sensor.

Una vez hallado dicho protocolo, se procederá al desensamblado. Este proceso consiste únicamente en separar los datos útiles de los datos de control

❖ Subsección de ensamblado de datos

La salida final del módulo de procesado de datos debe ser un fichero que contenga los datos y parámetros necesarios para la comunicación entre la tarjeta de procesado y la interfaz Bluetooth, similar al fichero *requests.txt* del módulo de entrada de datos. La creación de dicho fichero se lleva a cabo en esta subsección. Para ello:

- Se debe elaborar un protocolo de comunicación cliente-servidor entre el CS y la interfaz de usuario.
- En base a este protocolo se crearán los parámetros que deberán ser usados durante las transmisiones.
- Dichos parámetros serán incluidos en el fichero siguiendo un formato similar al seguido en *requests.txt.*

4.3.5. Módulo de salida de datos

Como ya se ha indicado, este módulo será el encargado de llevar a cabo la salida de datos a través del sistema cliente-servidor. En este caso, el programa servidor se hallará en la tarjeta de procesado mientras que el programa cliente estará ubicado en el terminal que se pretenda usar como interfaz de usuario.

La principal tarea a desarrollar en este punto del software sería elaborar un protocolo de comunicación para el envío de datos. El terminal le enviaría al CS el dispositivo del que se desea conocer las medidas y el número de éstas.

❖ Programa servidor

El CS funcionará con un sistema operativo Linux, lo que permitirá escribir el programa en C.

Dicho programa tomará los datos del fichero salida.txt y los almacenará en un TAD que hará de soporte en la comunicación entre la tarjeta de procesado y el terminal de interfaz de usuario de una forma similar a la que usa el módulo de entrada de datos USB.

El método de transmisión de datos consistirá en una primera confirmación de interconexión entre el la tarjeta de procesado y el terminal para poder, a continuación, proceder con los correspondientes envíos de datos.

El envío se realizaría de forma desfragmentada, enviando una trama compuesta de una serie de bytes de confirmación de petición y otra serie de datos cada vez. En las tramas de datos se alternarían los datos de fecha y hora con los del valor de las medidas, de manera que harán falta dos tramas de datos por cada medida.

❖ Programa cliente

La interfaz de usuario, en primera instancia, constará en una Tablet que usará el sistema operativo Android. Esto implicará que el programa cliente será implementado como una aplicación Android programada en Java.

La aplicación, además de tener una parte dedicada a la comunicación con la tarjeta de procesado, deberá ofrecer una interfaz con el usuario que constará de un menú que se mostrará por pantalla y una entrada táctil a través de la cual el usuario indicará qué datos desea conocer. La interfaz con el usuario tendrá una forma similar a la siguiente:

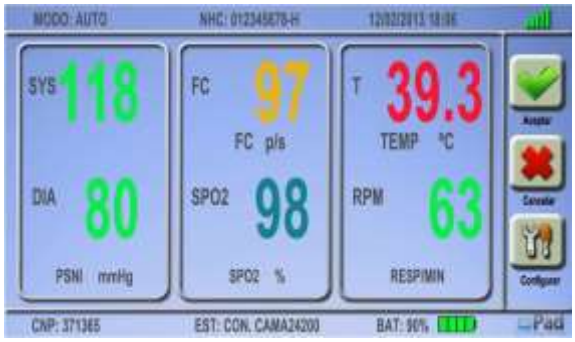


Ilustración 4.13: Menú de la interfaz

En la imagen se observa un ejemplo en el que aparecen medidas de los tres sensores más comunes en los hospitales (tensiómetro, pulsioxímetro y termómetro). Además, la aplicación albergará una base de datos con las medidas registradas clasificadas por el nombre del paciente que esté siendo tratado.

La parte dedicada a la comunicación con la tarjeta de procesado tendrá un modo de funcionamiento idéntico al explicado en el apartado del módulo de entrada de datos, puesto que en la práctica también lo es.

4.3.6. Esquema final del diseño software

En resumen, el usuario del sistema introducirá el dispositivo que desea usar en ese momento y, en caso de que éste esté correctamente conectado, introducirá también el número de medidas que desea conocer. Dicho número de medidas será usado por el módulo de entrada para obtener los datos del sensor. Seguidamente, el módulo de procesado obtendrá los datos de interés de entre todos los enviados por el sensor para posteriormente enviarlos a través de Bluetooth a la interfaz de usuario.

Esquemáticamente, el diseño software final compuesto por todos los módulos descritos anteriormente tendría la siguiente forma:

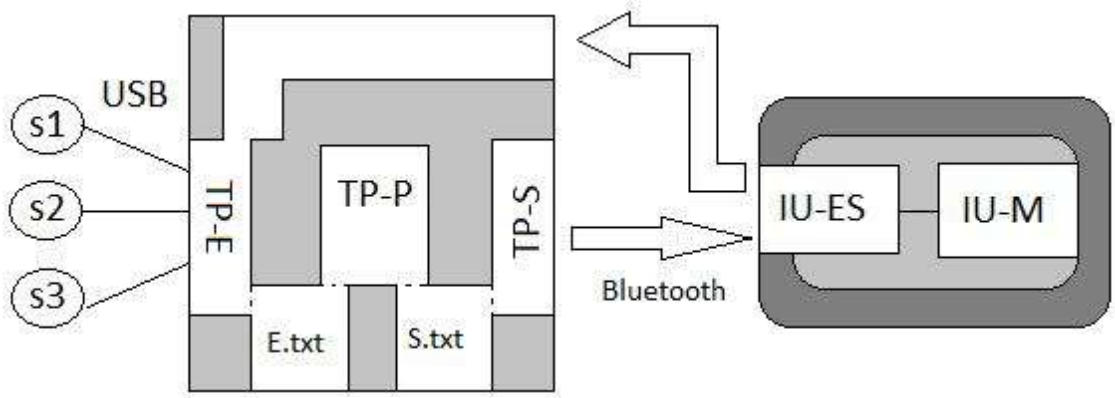


Ilustración 4.14: Esquema del diseño software

El esquema anterior se detalla a continuación:

- s1, s2, s3: sensores.
- TP-E: Tarjeta de Procesado, módulo de Entrada. Se conecta con los sensores a través de USB y se comunica con el módulo de procesado a través del fichero entrada.txt. Se rige por las peticiones del IU.
- E.txt: fichero entrada.txt.
- TP-P: Tarjeta de Procesado, módulo de Procesado. Recibe datos del TP-E y se comunica con el TP-S a través del fichero salida.txt.
- S.txt: fichero salida.txt.
- TP-S: Tarjeta de Procesado, módulo de Salida. Recibe datos del TP-P y se comunica con la IU-ES a través de Bluetooth.
- IU-ES: Interfaz de Usuario, módulo de Entrada-Salida. Recibe los datos del IU-M y los envía a la TP. Al mismo tiempo recibe datos del TP-S a través de Bluetooth.
- IU-M: Interfaz de Usuario, Menú. Recibe las peticiones del usuario y envía dichas peticiones al IU-ES, recibiendo y mostrando las respuestas recibidas por el IU-ES.

4.4. Requisitos de alimentación

Una parte fundamental del diseño del Concentrador de Señales es la fuente de alimentación, encargada de proporcionar energía a todo el sistema. La finalidad de dicha fuente es conseguir la autonomía que requiere el dispositivo para poder ser usado de forma portátil.

Para implantarla se seguirá el siguiente esquema:

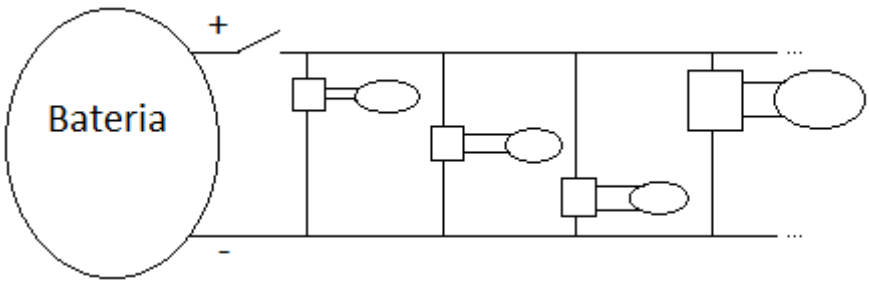


Ilustración 4.15: Esquema del sistema de alimentación

Los elementos con forma cuadrada simbolizan conversores de voltaje cuya principal función es ofrecer un voltaje concreto a los elementos que alimenta a continuación. Asimismo los elementos con forma elíptica simbolizan los elementos del sistema que pretendan ser alimentados con la batería, ya sean los sensores, la tarjeta de

procesado o la Tablet. Por último, se añade un interruptor para encender y apagar el sistema completo.

Los conversores de tensión tienen como finalidad ofrecer al elemento que alimenta una tensión adecuada para su funcionamiento ya que la batería dispondrá un voltaje que puede ser excesivo para alguno de ellos.

El esquema anterior se situaría dentro de la carcasa del CS, ofreciendo ranuras para conectar los elementos portátiles del sistema como los sensores.

En cuanto a los requisitos que se buscan referentes a la alimentación del sistema a nivel general son los siguientes:

- El CS debe ser autónomo, con alimentación para al menos 8 horas de trabajo.
- El tiempo de carga de la batería del CS debe ser menor de 1 hora.
- Para proceder a la carga de la batería del CS éste se introducirá en una base de carga que se conecta a la red de alimentación (220 VAC).

Para elegir la batería del sistema se atenderán a los parámetros que caracterizan las baterías a nivel general: el voltaje, la corriente de pico, la carga (amperios por hora) y su peso.

La batería deberá ofrecer como mínimo el voltaje máximo que sea requerido por un elemento del sistema. Los conversores DC-DC se encargarán de adaptar el resto de elementos del sistema a dicho voltaje.

Por otro lado, se deberá cubrir el peor aunque improbable caso de que todas las corrientes de pico se den a la vez, de manera que la corriente de pico que debe ofrecer la batería deberá tener el valor de, como mínimo, la suma de todas las posibles corrientes de pico de los elementos individuales del sistema.

De este modo, para dimensionar la batería se medirán las características de consumo anteriores de los elementos que conformarán el sistema individualmente. Una vez obtenidos estos datos y atendiendo a los requisitos sobre los tiempos de carga y de autonomía expuestos anteriormente, se hará una aproximación sobre el uso del dispositivo (carga mínima que pueda ofrecer) para poder elegir finalmente una batería comercial que satisfaga todas las necesidades energéticas del sistema.

Capítulo 5:

Desarrollo e implementación del sistema

5.1. Introducción

Este capítulo explica los procedimientos utilizados para llevar a cabo la implementación del diseño expuesto en el capítulo anterior exponiendo los métodos llevados a cabo y los principales problemas afrontados con sus respectivas soluciones tomadas para solventarlos

En primer lugar se detallarán los pasos seguidos durante el desarrollo de las aplicaciones software del sistema.

Seguidamente, se describirán los datos obtenidos durante las mediciones del consumo de los elementos del sistema para elegir en última instancia la batería comercial del sistema.

5.2. Desarrollo Software

Durante el desarrollo software del sistema se comenzará por obtener en primer lugar los formatos de las interfaces de los sensores a conectar para poder, por consiguiente, desarrollar las aplicaciones software encargadas de gestionar la comunicación entre el sistema y el sensor correspondiente.

A continuación se implementarán las estructuras y archivos que harán de base para el flujo de datos, realizando una explicación detallada de sus formatos.

5.2.1. Determinación de protocolos USB

En este apartado se obtienen los formatos de las interfaces de los sensores a conectar.

El método utilizado ha sido ingeniería inversa usando los programas aportados por el fabricante para realizar peticiones básicas a los sensores y un analizador USB (USBTrace) capaz de mostrar los detalles software de las tramas enviadas y recibidas.

Ilustración 5.1: USBTrace

Ilustración 5.2: ION Health

❖ Termómetro

En primer lugar se han analizado las tramas enviadas y recibidas por el termómetro. Con el fin de ilustrar el protocolo se mostrará un ejemplo correspondiente a la petición de dos medidas. La siguiente captura muestra los datos recogidos por el software proporcionado con el termómetro:

La captura de pantalla de todas las tramas detectadas por el analizador USBTrace se muestra en el Anexo A. En este punto se irán conjuntos de tramas sueltas para su análisis.

Fecha	Tiempo	Categoria	Valor Medido	Estado
07/04/2014	19:05:00	Temperatura	33,9	Guardado
07/04/2014	19:05:00	Temperatura	35,7	Guardado

Ilustración 5.3: Medidas del termómetro

Se observan tres tipos de transmisión: CONTROL_TRANSFER, CLASS_INTERFACE y BULK_OR_INTERRUPT_TRANSFER.

Los dos primeros tipos de transmisión son usados para iniciar la transferencia de datos, y serán analizados más adelante. Las tramas del tercer tipo son usadas para transmitir los datos relevantes, y van a ser analizadas a continuación:

- El primer byte de cada trama indicada el número de bytes (útiles) restantes que hay en dicha trama.
- Cada petición es respondida por 8 bytes útiles, de los cuales los 2 últimos tienen la función de checksum.
- Al comienzo de cualquier transmisión se envían siempre 3 peticiones respondidas con sus 8 respectivos bytes útiles de respuesta sin intercambio de datos.
- Finalmente, se envían dos peticiones (con sus 8 bytes de respuesta) por cada medida deseada. Este tipo de envíos serán estudiados a continuación.

Las tres primeras peticiones son las siguientes:

BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 24 00 00 00 00 A3 18	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 24	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 17	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 11	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 03	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	02 A5 45	3
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 24 00 00 00 00 A3 18	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 24	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 17	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 11	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 03	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 45	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 2B 00 00 00 00 A3 1F	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 2B	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 02	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	02 A5 23	3

Ilustración 5.4: Captura de tramas (1)

Estas peticiones consisten en un envío desde el PC de 8 bytes útiles con una respuesta de 8 bytes útiles desde el termómetro. Estas tramas son las que se envían cuando el software comprueba la correcta conexión del dispositivo al PC por lo que concluimos que esa es su principal finalidad.

A continuación, se envían dos peticiones por cada medida. Un ejemplo es el siguiente:

BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 25 00 00 00 00 A3 19	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 25	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 87	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 1C	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 05	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 13	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 D6	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 26 00 00 00 00 A3 1A	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 26	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 53	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 53	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 C4	2

Ilustración 5.5: Captura de tramas (2)

En la primera petición se envían los datos referentes a la fecha y hora de la medida tomada, y la segunda petición a la medida de la temperatura. El formato de intercambio de datos se detalla a continuación:

1º Ráfaga: 0187 011C 0105 0113 → 871C 0513

Si se ha usado el formato Little Endian: 1C87 1305

1C87 = 0001 1100 1000 0111 → 7 bits de año, 4 de mes, 5 de día.

- 00111: Día 7
- 0100: Mes 4
- 0001110: Año 14

1305: Primer byte de hora y segundo byte de minuto

- 13: $(13)_{16} = (19)_{10}$ Hora 19
- 05: $(05)_{16} = (05)_{10}$ Minuto 05

2º Ráfaga: 0153 0101 0153 0101 → 5301 5301

Si se ha usado el formato Little Endian: 0153 0153

0153: $(0153)_{16} = (339)_{10}$

- Temperatura: 33,9 ºC

Un ejemplo más:

BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 25 01 00 00 00 A3 1A	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 25	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 87	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 1C	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 05	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 13	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 D6	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 26 01 00 00 00 A3 1B	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 26	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 65	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 65	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 E8	2

Ilustración 5.6: Captura de tramas (3)

1º Ráfaga: 0187 011C 0105 0113 → 871C 0513

Si se ha usado el formato Little Endian: 1C87 1305

1C87 = 0001 1100 1000 0111 → 7 bits de año, 4 de mes, 5 de día.

- 00111: Día 7
- 0100: Mes 4
- 0001110: Año 14

1305: Primer byte de hora y segundo byte de minuto

- 13: $(13)_{16} = (19)_{10}$ Hora 19
- 05: $(05)_{16} = (05)_{10}$ Minuto 05

2º Ráfaga: 0165 0101 0165 0101 → 6501 6501

Si se ha usado el formato Little Endian: 0165 0165

0165: $(0165)_{16} = (357)_{10}$

- Temperatura: 35,7 ºC

Se comprueba que coinciden con los datos de la captura de imagen del software del dispositivo.

En cuanto a los datos de control enviados en primer lugar, como ya se indicó en el apartado del estado del arte, se usan para configurar los dispositivos con la finalidad de realizar los intercambios de datos correctamente.

En este caso, con otro analizador de tramas USB (USBLyzer) se ha comprobado que la finalidad concreta de estas tramas era establecer una vía lógica de comunicación (un *pipe*). Aunque se han mostrado 4 tramas de las cuales 2 enviaban datos, observando la especificación del USB HID se ha comprobado que las tramas de tipo CLASS_INTERFACE tenían una configuración concreta que encajaba con los datos de configuración de las tramas CONTROL_TRANSFER, motivo por el cual se intentó enviar únicamente las tramas de control al dispositivo con sus correspondientes buffer de datos y el dispositivo contestó correctamente.

Por ello se concluye que las tramas de tipo CLAS_INTERFACE no son más que las tramas de control que realmente se han enviado mostradas de otra forma por los analizadores USB.

❖ Tensiómetro. Modo usual

El tensiómetro en modo usual siempre envía tramas de 8 bytes. La transmisión comienza con una trama enviada desde el ordenador al dispositivo: 04 12 16 18 28 34 35 36.

Los datos que entran al ordenador se componen de 1 byte de cabecera formado por una F y un número comprendido entre 1 y 7. Este número indica el número de bytes que son útiles dentro de la trama. Por ejemplo, en la trama F4 84 54 32 CA 23 43 12 sólo se toman 84 54 32 CA.

Siguiendo este convenio y ordenando los bytes útiles de cada trama se puede deducir el siguiente formato:

- 1 byte de valor sistólico.
- 1 byte de valor diastólico.
- 1 byte de valor de pulso.
- 1 byte para el año y el mes.
- 2 bytes para día, hora y minuto.

Siguiendo este protocolo podemos obtener la información como se indica en el siguiente ejemplo:

Fecha	Tiempo	Sys	Dia	Pulso
23/10/2012	11:44	138	84	50
23/10/2012	11:48	136	79	58
23/10/2012	13:58	144	83	68
31/10/2012	16:45	161	93	51
31/10/2012	16:48	108	66	67
Promedio	**---**	**137**	**81**	**59**

Ilustración 5.7: Medidas del tensiómetro

Tramas enviadas:
F4845432CA000000 F4BAEC2488000000 F64F3ACABAF02400
F5905344CABB2400 F47A20A15DBB2400 F533CAFC2D202400
F66C4243CAFC3000 F42089513BFC3000 F500000005F83000

Bytes útiles:
8A 54 32 CA BAEC 24 88 4F 3A CA BA F0 24 90 53 44 CA BB 7A 20 A1 5D 33 CA FC 2D 20 6C 42 43 CA FC 30 20 89 51 3B **00 00 00** 05 F8

Los ceros en negrita se corresponden con la fecha y la hora del promedio.

- Sistólica, diastólica y pulso: Conversión hexadecimal en decimal.
  - 8A 54 32 = 138 84 50
  - 88 4F 3A = 136 79 58
  - 90 53 44 = 144 83 68
  - A1 5D 33 = 161 93 51
  - 6C 42 43 = 108 66 67
  - 89 51 3B = 137 81 59
  - Promedio
- Año y mes: Primeros 4 bits del año y los últimos 4 bits del mes.
  - CA: C=12 A=10 CA → …/10/2012
- Día, hora y minutos: Descomposición binaria. 5 bits de día, 5 bits de hora y 6 bits de minuto.
  - BAEC=1011 1010 1110 1100

Día: 10111=23 Hora: 01011=11 Min: 101100=44
BAF0=1011 1010 1111 0000
Día: 10111=23 Hora: 01011=11 Min: 110000=48
BB7A=1011 1011 0111 1010
Día: 10111=23 Hora: 01101=13 Min: 111010=58
FC2D=1111 1100 0010 1101
Día: 11111=31 Hora: 10000=16 Min: 101101=45
FC30=1111 1100 0011 0000
Día: 11111=31 Hora: 10000=16 Min: 110000=48

❖ Tensiómetro. Modo Diag

En el modo Diag las tramas tienen la misma estructura que en el modo Usual con una cabecera de 1 byte en la que se indica el número de bytes útiles en la trama. En el modo Diag en cambio, siempre se transmiten el mismo número de bytes útiles, que se dividen en 7 días. Cada día contiene 56 bytes en los que se incluye la información de 2 medidas tomadas por la mañana y 2 medidas tomadas por la tarde. Un día tiene la siguiente estructura:

- 1ª medida de la mañana.
- 2ª medida de la mañana.
- 1ª medida de la tarde.
- 2ª medida de la tarde.

Cada una de las anterior contiene el mismo formato que el modo usual con un byte para el valor sistólico, otro para el valor diastólico, otro para el pulso, otro para el año y mes, y otros dos bytes para el día, la hora y el minuto.

En cada día (56 bytes) aparecen las dos medidas de la mañana consecutivamente, 14 bytes a 0 y las dos medidas de la tarde consecutivamente también.

En el caso de que haya algún día sin ninguna medida, todos los bytes correspondientes a ese día estarán a 0.

En el caso de que en algún día sólo se haya tomado una de las medidas (o bien por la tarde o bien por la mañana) los bytes correspondientes a las medidas no tomadas estarán a cero excepto los encargados de indicar la fecha.

❖ Pulsioxímetro

El pulsioxímetro está constantemente enviando información al ordenador, pero la información de importancia (la oxigenación y la pulsación) sólo es enviada cuando el

software del dispositivo la muestra por pantalla. Hay dos tipos de transferencia de datos entre el ordenador y el pulsioxímetro: en tiempo real y cargando datos memorizados por el dispositivo.

En el caso de datos memorizados por el dispositivo, la información es enviada en una gigantesca trama de una vez. A continuación se muestra un ejemplo:

00 F2 80 00 F2 80 00 F2 80 00 80 81 77 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 62 F0 C9 62 F0 C9 62 F0 C9 62 F0 CA 62 F0 CA 62 F0 CA 62 F0 CA 62 F0 CA 62 F0 CA 62 F0 CA 62 F0 CA 61 F0 CA 61 F0 C9 61 F0 C9 61 F0 C8 61 F0 C8 61 F0 C8 61 F0 C8 61 F0 C8 61 F0 C8 61 F0 C8 61 F0 C7 61 F0 C7 61 F0 C6 61 F0 C6 61 F0 C6 61 F0 C6 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00 F0 80 00

En el caso anterior se han obtenido los siguientes datos:
Oxigenación: 98 97
Pulsación: 73 74 73 72 71 70

Se puede observar que la gran trama está divida en “subtramas” con una cabecera 111100001 seguida de 15 bits de información. La información se divide a su vez en 7 bits de pulsaciones y 8 bits de oxigenación.

Por ejemplo: F0C962: 1001001=73 01100010=98

Si se comprueban sucesivamente los datos no nulos de la trama se obtienen los siguientes datos:

- Pulsaciones sucesivamente:
  73(C9) 74(CA) 73(C9) 72(C8) 71(C7) 70(C6)
- Prescindiendo del bit más significativo de la C:
  Oxigenación: 98(62) 97(61)

En el caso de envío de datos en tiempo real, se envían constantemente dos tipos de tramas: unas de 2 bytes y otras de 3 bytes. Las tramas de dos bytes contienen los valores de la pulsación y la oxigenación sin cabeceras. Las tramas de 3 bytes probablemente tengan como principal finalidad mantener la sincronización del tiempo real.

5.2.2. Flujo de datos

Como ya se indicó en el comienzo de la memoria, en este proyecto sólo se desarrollará una aplicación que gestione la entrada de datos de los sensores. Esta aplicación tendrá como principal finalidad recibir los datos del dispositivo en cuestión, para lo cual será necesario crear una estructura sobre la que se asentará todo el flujo de datos necesario.

Dicha estructuración se detalló en la ilustración 10 del diseño software del sistema.

Para poder realizar el diseño de la aplicación, se han seguido unas pautas que se muestran a continuación:

- Lo primero que hará el programa será pedir el número de medidas a recibir. Éste, podrá ser enviado desde el terminal de usuario o a través de un teclado en su defecto.
- En base a este número, se generará un archivo que hará de base de datos.
- De este archivo se obtendrá la información sobre cada transferencia de datos entre el PC y el sensor.
- Con esta información se procederá con las transferencias de datos con el sensor.
- Los datos recibidos serán almacenados en otro archivo que servirá de base para su posterior procesado.

Para lograr los objetivos anteriores se han elaborado una serie de Tipos Abstractos de Datos (TAD) encargados de almacenar de forma temporal en memoria dinámica los datos usados durante las transmisiones. El más importante de todos es la cola, que se ha implementado en dos situaciones.

La primera de las dos es en el punto en el que se extraen los datos del primer archivo (al que se ha llamado *requests.txt*). En dicho punto, la cola ha servido para ofrecer un soporte de datos en el momento de realizar los intercambios de datos entre PC y sensor.

La segunda de las dos ha tenido como finalidad recibir los datos del sensor para almacenarlos posteriormente en otro archivo (al que se ha llamado *data_in.txt*).

El resto de detalles sobre los TAD usados se explicarán en el siguiente apartado.

Los datos guardados en el archivo *data_in.txt* han servido como entrada para el módulo de procesado de datos. Dicho módulo ha hecho uso de otro TAD, cuya principal finalidad ha sido ordenar los datos recibidos para su posterior procesado.

Dicho módulo, en resumidas cuentas, elimina en una primera instancia los bloques de datos que no aportan información relevante y en una segunda instancia, dentro de los bloques útiles, elimina los bytes concretos que tampoco aportan información de interés.

A continuación analiza las tramas o bloques de bytes resultantes como se ha explicado en el apartado "Determinación de protocolos USB" haciendo uso de funciones de conversión de tipos de datos, concretamente:

- Para obtener la fecha y la hora:
  - DecToBin: paso de decimal a binario.
  - BinToDec: paso de binario a decimal.
- Para obtener temperatura:
  - DecToHex: paso de decimal a hexadecimal.
  - HexToDec: paso de hexadecimal a decimal.

Uno de los principales problemas que surgió en este punto fue precisamente el de realizar correctamente la conversión de tipos. Para solucionar este problema se optó por almacenar los datos en el archivo *data_in.txt* en su forma decimal y almacenarlos en tipos de datos ***int***. Los valores en binario y hexadecimal se implementaron como cadenas de tipo ***char.*** De este modo, realizando reservas dinámicas de memoria (con sus correspondientes liberaciones) se logró finalmente obtener un resultado óptimo.

Tras la conversión de datos ya se dispone de los datos de interés del termómetro. Estos datos se enviarán por medio de un sistema Bluetooth al terminal de usuario o a una pantalla por medio de una conexión HDMI en su defecto.

5.2.3. TAD y funciones propias

En este apartado se mostrará una lista con los Tipos Abstractos de Datos que han sido utilizados durante la implementación del software del sistema. A continuación se presentan sus nombres y sus funciones:

- Referentes a la cola:
  - Node: nodo genérico de la cola. Alberga información sobre el paquete que almacena (tipo de transferencia y puntero genérico al paquete de información).
  - tInfoPacket: puntero genérico que puede apuntar a la estructura ControlPacket o InterruptPacket.
  - tNode: puntero a la estructura Node.
  - tNodeQueue: Nodo de cabecera de la cola. Contiene un puntero al primer elemento de la cola y otro al último.
  - tQueue: puntero a tNodeQueue. Desde este tipo de dato es desde donde se usarán los TAD cola.

- Referentes a los datos a enviar/recibir:
  - tControlPacket: estructura que alberga los datos necesarios para enviar o recibir un paquete de control.
  - tControl: puntero a tControlPacket.
  - tInterruptPacket: estructura que alberga los datos necesarios para enviar o recibir un paquete de interrupción.
  - tInterrupt: puntero a tInterruptPacket.
  - tData: puntero genérico que contiene el campo data a enviar al sensor.

Los TAD referentes a los datos que se desean enviar o recibir contienen todos los parámetros necesarios para realizar el envío o recibo correspondiente. El TAD tData se utiliza para albergar un buffer con los datos a enviar o recibir.

La jerarquía que siguen los TAD es la siguiente:

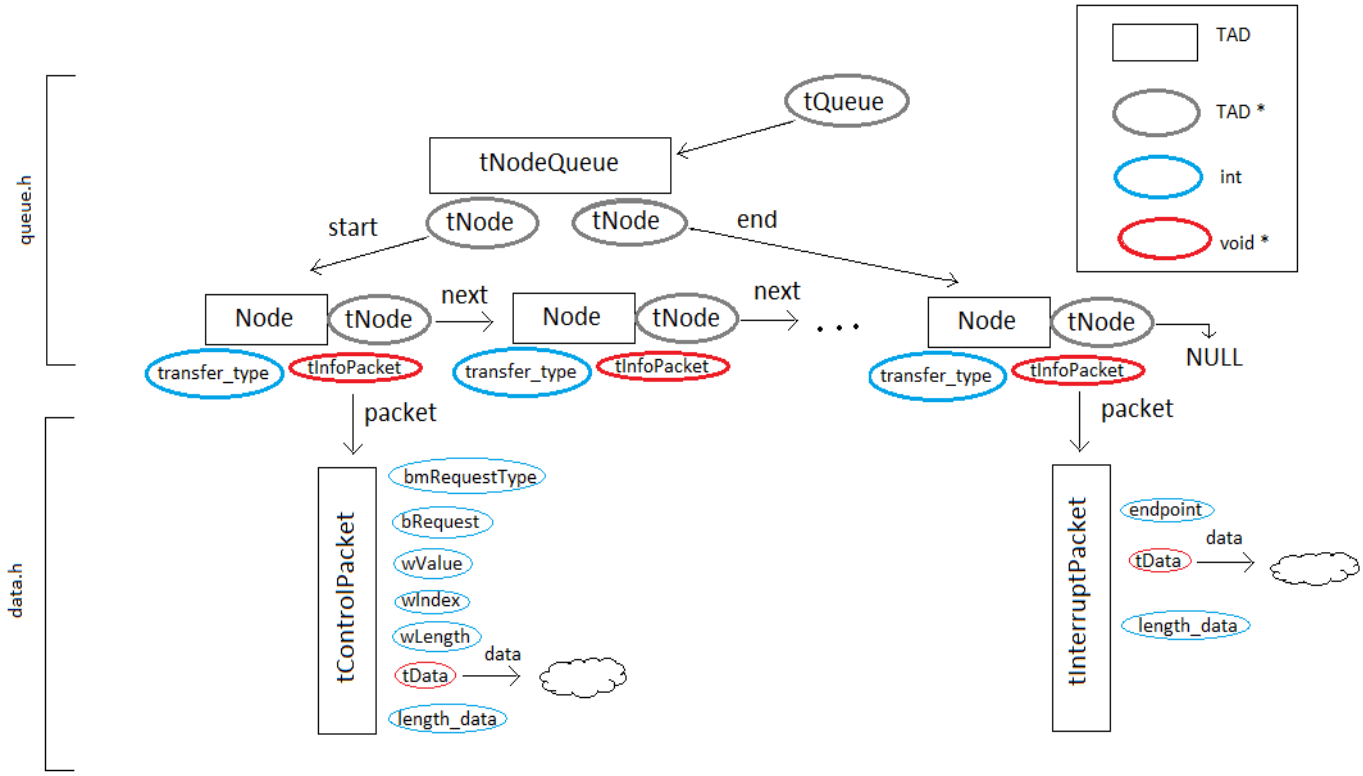


Ilustración 5.8: TAD de entrada de datos

Los TAD tControl y tInterrupt son asignados a los punteros genéricos de las estructuras Node (indicados con una elipse roja), los TAD tInfoPacket.

Por otro lado, en el módulo de procesado de datos se ha declarado y desarrollado otro TAD más, llamado tBlock. Este ha consistido en una estructura capaz de albergar un puntero genérico, en el que se ha almacenado una cadena de datos, y un número identificativo.

La finalidad de tBlock ha sido almacenar los datos devueltos en el archivo *data_in.txt* por el módulo de entrada de datos de una forma ordenada con el fin de tener

más capacidad a la hora de eliminarlos y manipularlos. Resumiendo, este TAD ha sido el pilar fundamental para el módulo de procesado.

Todos los TAD han tenido una serie de funciones o métodos propios definidos implementados como funciones en C, que han tenido un papel de vital importancia a la hora de crear, eliminar o manipular los datos contenidos en los TAD del sistema. Dichas funciones, así como los atributos de cada TAD se explican más ampliamente en los anexos, concretamente en el ANEXO C.

5.2.4. Formatos de archivos de transición

Por último, se va a cerrar este apartado hablando sobre el formato de los archivos de transición, es decir, de los archivos *requests.txt* y *data_in.txt.* En el caso del archivo *requests.txt* se deben cumplir las siguientes características previas:

- Se debe incluir en primer lugar el tipo de envío.
- Por cómo están definidos los tipos de envíos en la librería *libusb.h* el tipo de envío vendrá dado por los valores 0 y 3 para los tipos de envío de control y de interrupción respectivamente.
- Se deben incluir todos los parámetros necesarios para realizar los envíos y recibos correctamente.
- Deben incluirse tanto los parámetros de entrada como de salida.
- Los parámetros que serán requeridos para las transferencias de interrupción serán el endpoint, el buffer de datos y el tamaño de dicho buffer.
- Los parámetros para las transferencias de control serán el bmRequestType, el bRequest, los valores wValue, wIndex y wLength, y un buffer de datos con su tamaño.

De esta manera, el formato del archivo consistirá en una línea por cada envío o recibo que pretenda realizarse, con la siguiente distribución de datos:

- Control: 0 char char[2] char[2] char* int char[2]
- Interrupción: 3 int char* int

En el caso de que cualquiera de estas líneas de corresponda con un envío, a continuación de estos datos aparecerán ordenadamente los valores en decimal de los bytes que se deseen enviar.

Usando este formato se ha podido desarrollar una función propia del TAD cola que lee todos los datos incluidos en dicho archivo y los almacena ordenadamente en una

cola (llamada database) que es la que se consulta a la hora de realizar las transferencias entre el PC y el sensor.

5.2.5. Compilación para sistemas embebidos

Para compilar los programas elaborados en el lenguaje de programación C desde el PC para que sean ejecutados en la tarjeta BeagleBone se han seguido una serie de pasos que se detallarán a continuación:

- Instalación de un *Toolchain*.
- Uso de Eclipse IDE con las herramientas de desarrollo de C/C++.
- Creación y configuración de un proyecto.
- Compilación y ejecución de una aplicación.

❖ INSTALACIÓN DE UN TOOLCHAIN

Para poder llevar a cabo esta tarea el primer paso es conseguir una herramienta capaz de realizar compilaciones para arquitecturas diferentes de la arquitectura en la que se desarrolla el programa. Esta herramienta recibe el nombre de *toolchain*. En este proyecto, el sistema operativo en el que se ha desarrollado el software ha sido Ubuntu 13.04 sobre una arquitectura IA-64 (arquitectura de Intel para procesadores de 64 bits), mientras que el sistema embebido BeagleBone usa el sistema operativo Angstrom en una arquitectura ARMv7. Para esta configuración, la herramienta que se ha instalado ha sido *angstrom-2011.03-i86_64-linux-armv7a-linux-gnueabi-toolchain.tar.bz2*.

❖ USO DE ECLIPSE IDE

El IDE elegido para desarrollar el software ha sido Eclipse debido a su gran versatilidad. Este IDE de por sí no incluye las herramientas necesarias para programar en el lenguaje C o C++, por lo que es necesario descargar en primer lugar las herramientas de desarrollo correspondientes. Esta tarea se puede llevar a cabo desde el propio Eclipse con bastante facilidad. Tras este paso se reinicia el programa y ya estará todo preparado para empezar a programar.

❖ CREACIÓN Y CONFIGURACIÓN DE UN PROYECTO

La creación de un proyecto se realiza del mismo modo que en cualquier otro IDE para programar de la manera que se desee. La configuración del proyecto, en cambio, es el paso importante para llevar a cabo la compilación cruzada necesaria para llevar a cabo la ejecución del software en el sistema embebido. En esta configuración se le indicará a Eclipse que los programas desarrollados se compilarán para otra arquitectura. Para ello, se seleccionará en primer lugar el compilador del *toolchain* para que la compilación se realice de cara a su uso en el sistema embebido. A continuación se realizarán los mismos

pasos que fueron seguidos para configurar el compilador para compilar en este caso el *linker* del programa. Por último, también se tomará el ensamblador del sistema del *toolchain*. De esta manera los programas desarrollados en adelante en Eclipse generarán un código máquina ejecutable en el sistema embebido.

❖ COMPILACIÓN Y EJECUCIÓN DE UNA APLICACIÓN

En este punto el sistema ya está preparado para desarrollar el código fuente de los programas que serán ejecutados en el sistema embebido. El desarrollo del código fuente se realizará sin ningún tipo de modificación y, en el caso de que esté bien desarrollado y no dé ningún tipo de error, se deberá transferir a la placa.

Para transferir el archivo a la placa lo primero será tener en cuenta que la conexión USB de la BeagleBone al PC está configurado como una conexión Ethernet, por lo que escribiendo en una terminal el comando *ifconfig* aparecerá la dirección IP y sus parámetros correspondientes referentes a la dirección a través de la cual se conectará el PC a la placa. Esta configuración se puede modificar al antojo del programador a través del archivo /etc/network/interfaces. Una vez configurada la conexión se requerirán los comandos *ssh* y *scp*. El primero será usado en una terminal con la finalidad de enlazar ésta a la placa. El segundo se usará para transferir el archivo ejecutable obtenido al compilar el código fuente a la placa.

Otro modo de transmisión es usando la placa como un periférico, montando la unidad extraíble con el comando *mount*. Tras esto se usará el comando *cp* para copiar el archivo a la unidad extraíble y se cambiarán los permisos de dicha unidad con el comando *chmod*.

Tras esto, sencillamente hará falta desplazarse al directorio en el cual haya sido guardado el ejecutable y escribir en la terminal el comando *./<nombre_ejecutable>*. El punto de este comando hace referencia al directorio en el cuál se ubica la terminal, de modo que el comando anterior únicamente enuncia el nombre del ejecutable para que éste sea ejecutado.

Como se puede observar, el objetivo sencillamente es lograr un ejecutable que contenga las instrucciones del código máquina de la arquitectura correspondiente. Otros métodos que consigan este objetivo serán igual de válidos puesto que el software del sistema podrá ser ejecutado en la placa de la misma manera.

El método anteriormente descrito ha sido el usado a lo largo del desarrollo del sistema, aunque en el montaje final, al añadirle cómo dispositivo de salida un display compartible con la placa, no ha hecho falta puesto que a través de la terminal propia de la placa (más fácilmente accesible gracias a la pantalla añadida) se ha podido compilar el código fuente usando el comando *gcc* directamente sobre ella. Dicho comando ha sido capaz de generar el ejecutable deseado directamente en la placa.

5.3. Dimensionamiento de la batería

Como ya se indicó en el capítulo dedicado al diseño del sistema el Concentrador de Señales funcionará de manera autónoma, lo que implica el inevitable uso de una batería como fuente de alimentación.

En este apartado se estudiará el consumo de todos los componentes del sistema con la finalidad de conocer el consumo total del concentrador de señales integrado.

Para obtener las medidas del consumo de todos los elementos se ha usado el analizador AGILENT DC POWER ANALYZER modelo N6705 con capacidad para alimentar los dispositivos y tomar medidas detalladas sobre su consumo al mismo tiempo. La toma de medidas se realiza en intervalos de 30 segundos y pueden ser importadas a un medio USB extraíble como archivos con formato csv. Dichos archivos pueden ser fácilmente tratados con Excel o algún software similar. En este caso, se optará por el uso del software Matlab debido a que ofrece una mayor cantidad de recursos para manipular y representar los datos generados.

Ilustración 5.9: Agilent N6705

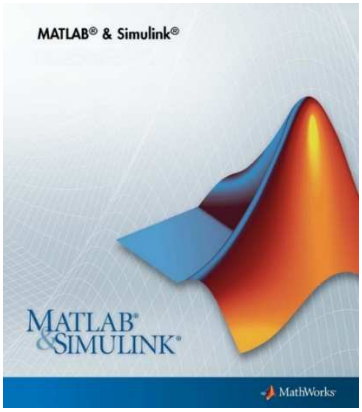


Ilustración 5.10: Matlab

5.3.1. Medida del consumo del sensor

Para dimensionar la batería, el primer paso será medir el consumo individual del sensor. Para ello, se sobreentiende que éste será usado sin sus pilas correspondientes y que estará alimentado por la batería del sistema en todo momento. Con el fin de caracterizar su consumo, se tomarán medidas en cada una de sus principales formas de funcionamiento.

El voltaje que requiere el termómetro para su correcto funcionamiento es de 3V. Para conocer la corriente de pico y la carga consumida se usará el analizador. Las medidas

serán tomadas durante sus principales acciones: encendido, apagado, medición de temperatura y transferencia de datos con el ordenador. La carga se calcula como:

$$i(t) = \frac{dq}{dt} \rightarrow dq = i(t)dt \rightarrow Q = \int_{t_0}^{t_1} i(t)dt$$

Como los datos ofrecidos por el analizador no son continuos, se calculará:

$$Q = \sum_{t_0}^{t_1} i(t)\Delta t \ \ donde\ \Delta t = 0.1\ s$$

Durante el encendido y apagado del termómetro, el analizador ha obtenidos los siguientes datos:

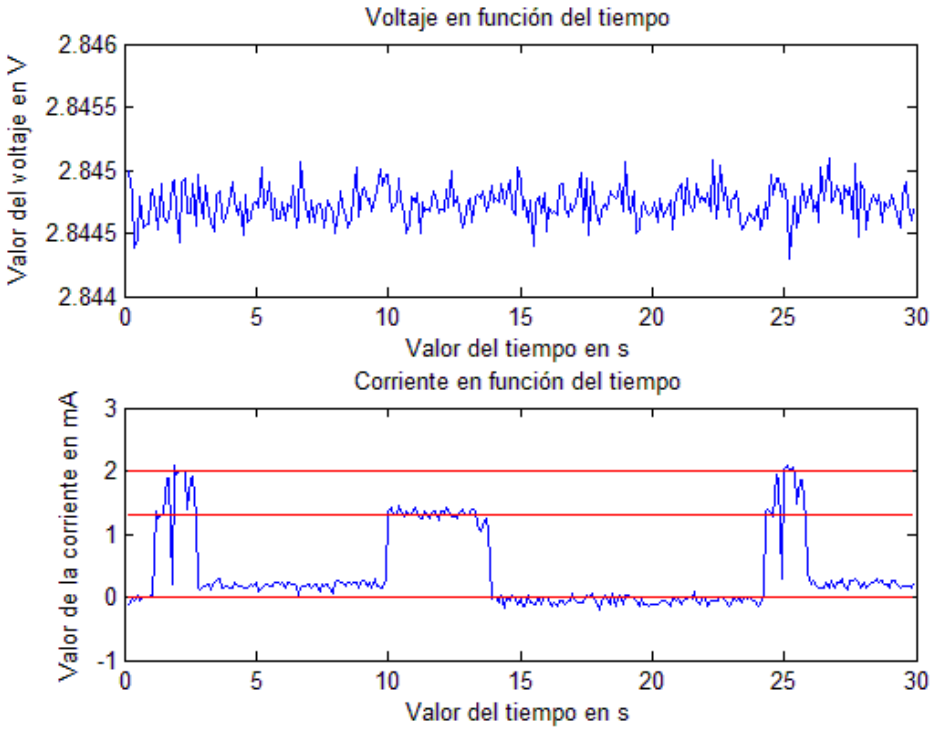


Ilustración 1.11: Encendido, apagado y encendido del termómetro

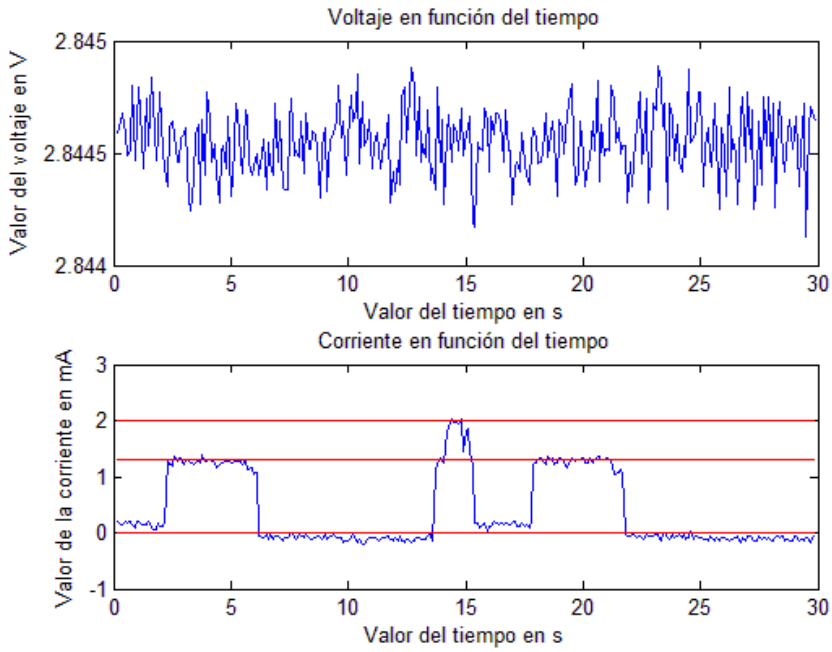


Ilustración 5.12: Apagado, encendido y apagado del termómetro

Se comprueba que durante el encendido y apagado se producen 2 mA de corriente de pico correspondientes a el momento de encendido del termómetro. A continuación se muestra el consumo medido al encender el termómetro y tomar tres medidas:

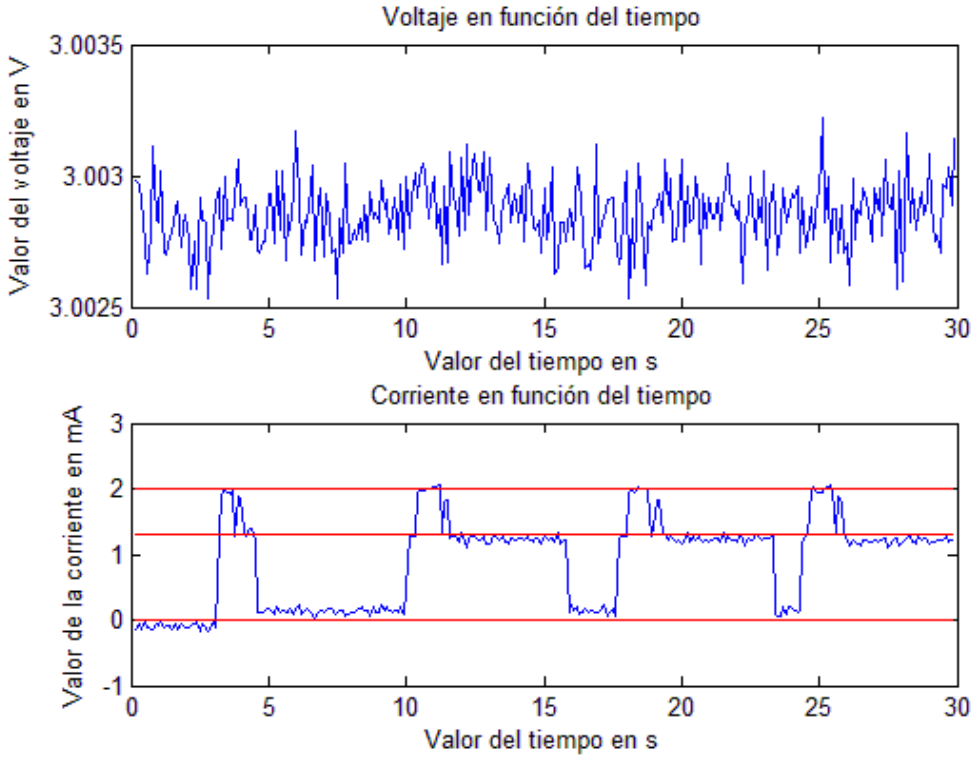


Ilustración 5.13: Encendido y toma de tres medidas

Se comprueba que al tomar medidas se produce una corriente de pico igual a la generada durante el encendido del sensor. Tras la corriente de pico se genera un consumo estable de 1 mA durante un breve periodo de tiempo. Por último se muestran los datos obtenidos al medir el consumo durante una operación consistente en encender el termómetro, tomar una medida y pasar dicha medida al ordenador por medio del USB. Dicha operación se ha realizado dos veces:

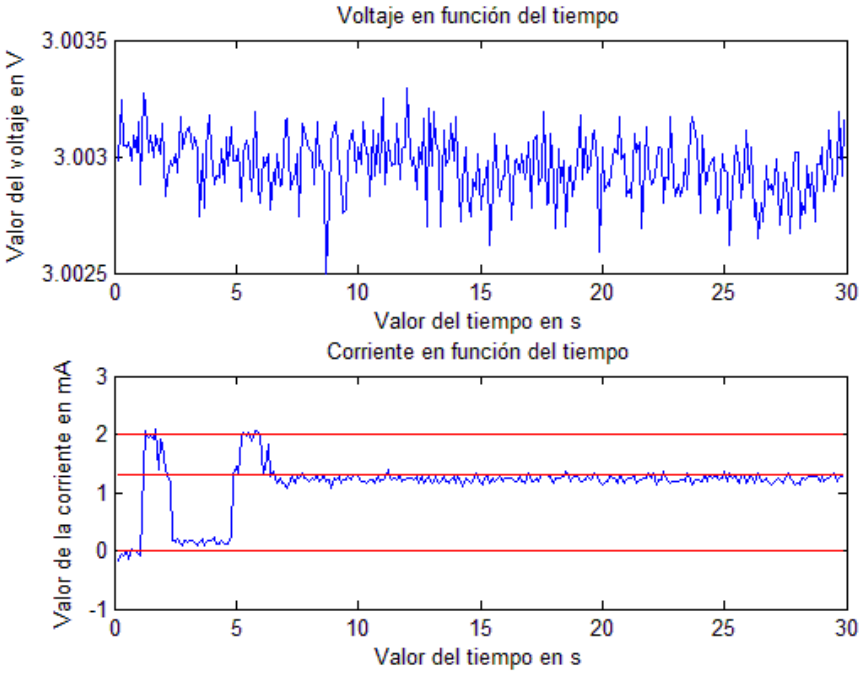


Ilustración 5.14: Encendido, medición y transferencia de datos (1)

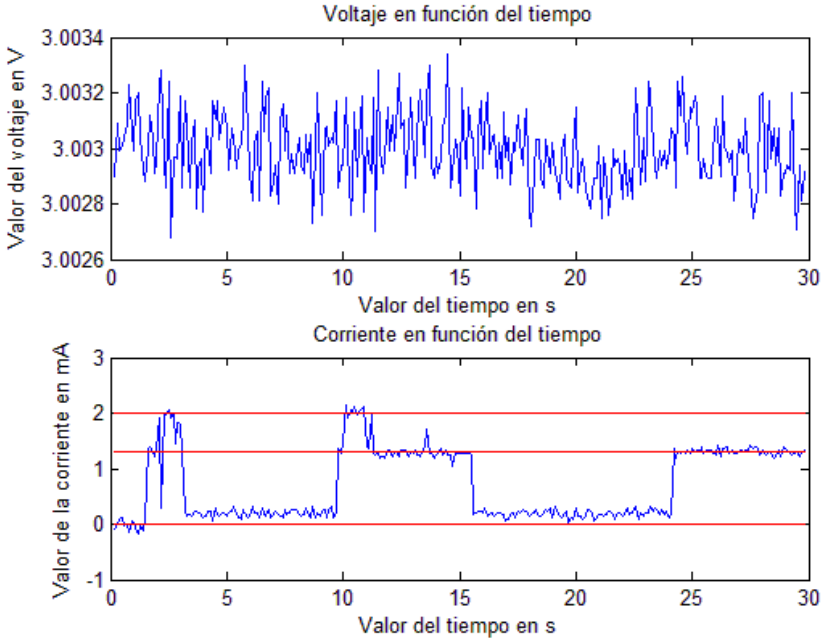


Ilustración 5.15: Encendido, medición y transferencia de datos (2)

En las mediciones anteriores se comprueba que durante la conexión USB y la transferencia de datos no se produce ningún pico de corriente aislado, sino que el consumo de corriente se mantiene en torno a 1 mA. En el primero de los casos, la conexión se realiza suficientemente deprisa como para enlazar el consumo de corriente durante la medición de temperatura y la conexión del termómetro con el ordenador, comprobando que ambos consumos no se superponen. En el segundo caso se ha dejado pasar un tiempo suficiente como para no enlazar dichos consumos, de manera que se pueda apreciar el consumo del termómetro durante la conexión aisladamente.

El consumo de carga será calculado por medio del software Matlab como la corriente consumida durante el tiempo de dicho consumo. Para ello se aislarán cada uno de los funcionamientos del termómetro y se calculará la media de los valores obtenidos. De este modo, para las medidas representadas anteriormente se han obtenido los siguientes valores:

- Carga media al encender = 0.0007 mAh
- Carga media al apagar = 0.0014 mAh
- Carga media de las mediciones = 0.0021 mAh
- Corriente media durante el encendido = 0.16 mA
- Corriente media durante el apagado = -0.0664 mA ~ 0 mA
- Corriente media conectado = 1.3 mA

Estos valores serán usados durante las aproximaciones sobre el uso del CS y sobre el consumo de la batería que tendrá finalmente, con el fin de establecer unos parámetros básicos a la hora de elegir una batería comercial.

5.3.2. Medida del consumo de la placa

En este apartado se realiza un estudio sobre el consumo de la tarjeta BeagleBone con su funcionamiento esperado. Las medidas se han tomado usando un teclado como método de entrada que no estará presente en el montaje final del sistema.

Asimismo, también cabe indicar que el sistema funciona correctamente con una alimentación de 5 Voltios y 2.5 miliamperios.

❖ VOLTAJE

En primer lugar se mostrará el consumo del voltaje de dicha tarjeta, que se espera que se estabilice en torno a 5 Voltios. Las dos siguientes figuras muestran el voltaje medido para diferentes fases del funcionamiento:

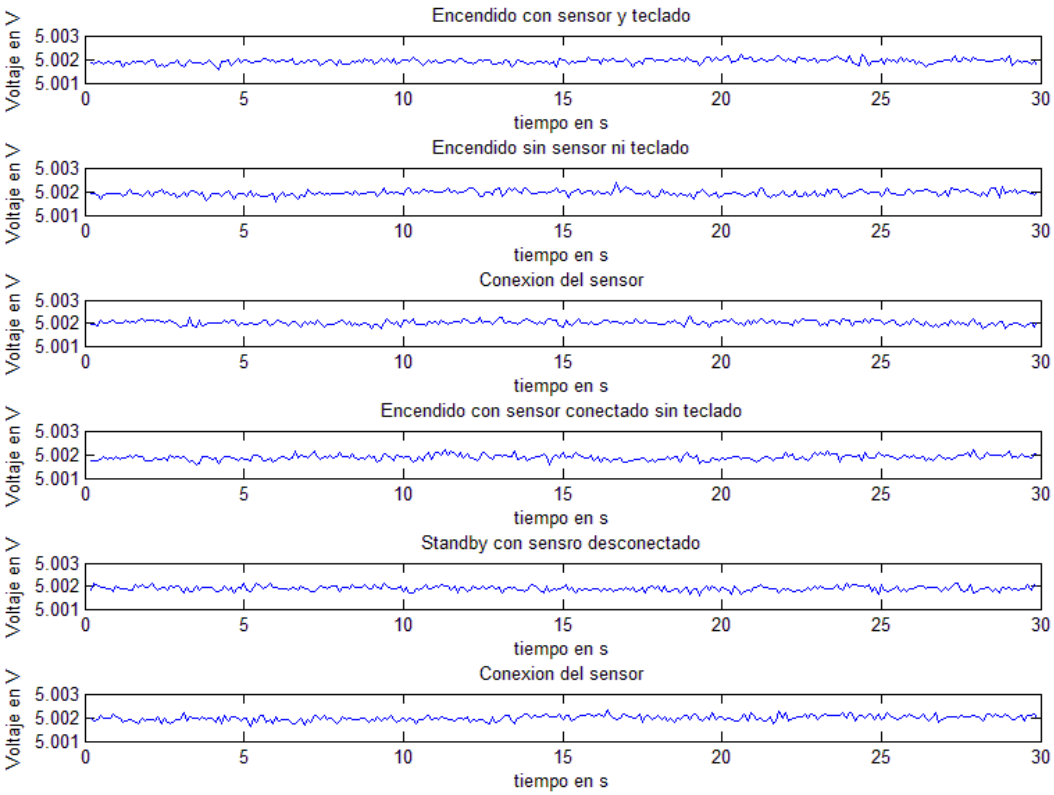


Ilustración 5.16: Voltajes en Beaglebone (1)

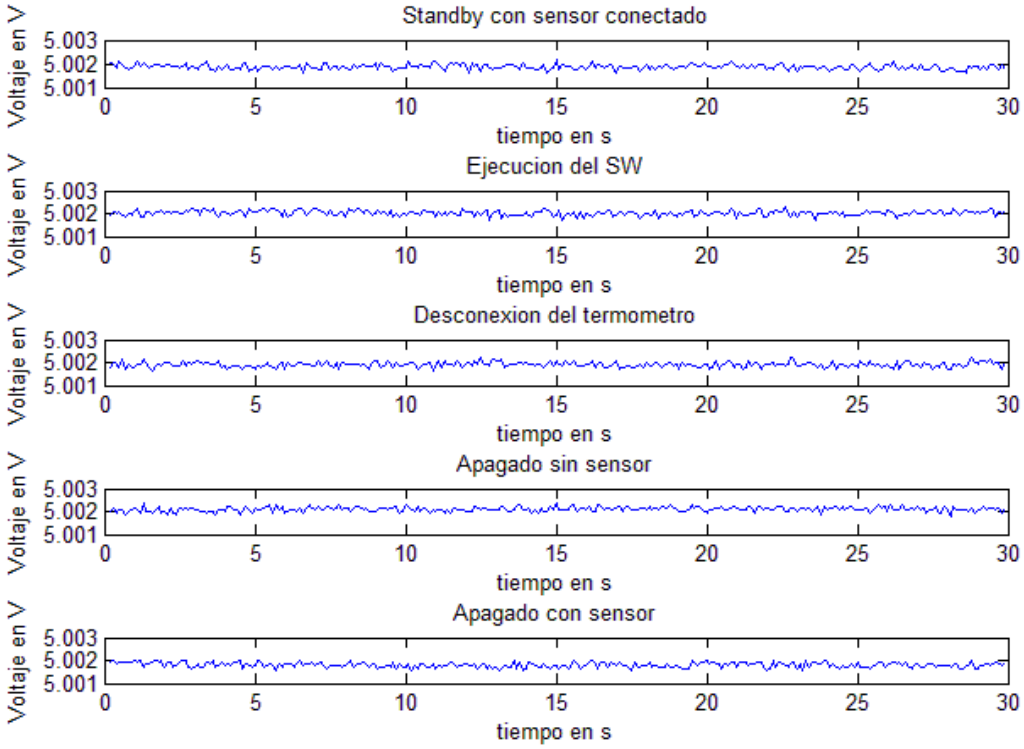


Ilustración 5.17: Voltajes en Beaglebone (2)

Como conclusión de este apartado se confirma que el consumo del voltaje se mantiene constante con pequeñísimas fluctuaciones despreciables en torno a 5 Voltios.

❖ CORRIENTE

En el presente apartado se va a realizar un estudio sobre el consumo de corriente eléctrica por parte de la tarjeta de procesado. Las fases del funcionamiento que se analizarán serán las mismas que en el apartado del voltaje.

A continuación se mostrarán las gráficas obtenidas referentes a la corriente que ha circulado por la tarjeta de procesado durante su arranque inicial:

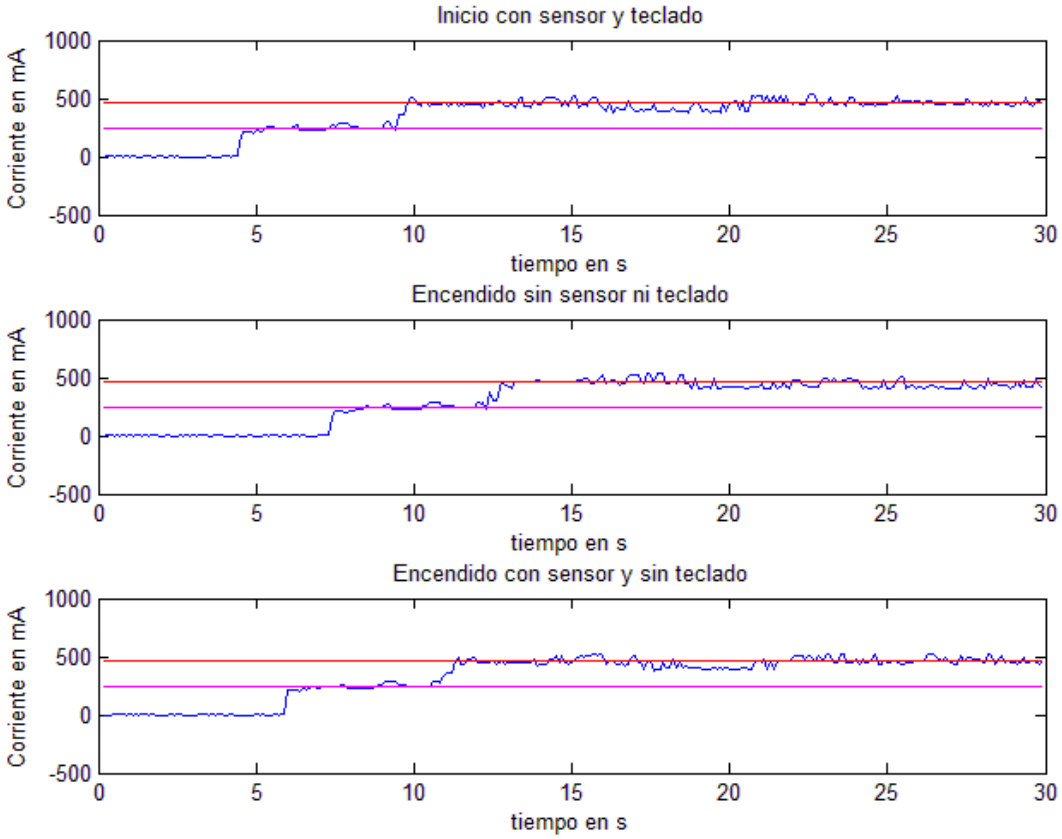


Ilustración 5.18: Corriente durante el encendido

Se comprueba que tienen el mismo comportamiento independientemente de los periféricos que estén conectados en el momento del encendido. Éste se divide en tres partes: consumo de cero amperios, consumo estable de 250 mA y consumo estable de 460 mA. El primero de los casos se corresponde con el momento en el que aún no se ha iniciado el sistema y el segundo, probablemente, con la configuración del sistema antes de su comienzo (lectura de la memoria BIOS, carga del sistema operativo en memoria…). En el tercero suponemos que el sistema ya se ha configurado y es un periodo de carga del sistema operativo, además de que justo en el instante en el que comienza este periodo se activan los periféricos.

En las tres gráficas anteriores, el valor marcado en rojo, en torno al que todas se estabilizan al pasar un breve periodo de tiempo, es 460 mA. En todos los casos se comprueba que se estabiliza con el mismo consumo. Del mismo modo, el valor marcado en magenta se corresponde con 250 mA. Esta segunda fase del inicio del sistema tiene una duración en los tres casos de 5 segundos aproximadamente.

Una vez analizado el consumo durante el inicio del sistema, se va a proceder a analizar el consumo de corriente durante el resto de periodos. En primer lugar se mostrará el consumo durante un periodo de inactividad, tras el cual aparecerá representado el consumo durante la conexión del termómetro y por último se mostrará también el consumo durante un periodo de inactividad tras dicha conexión:

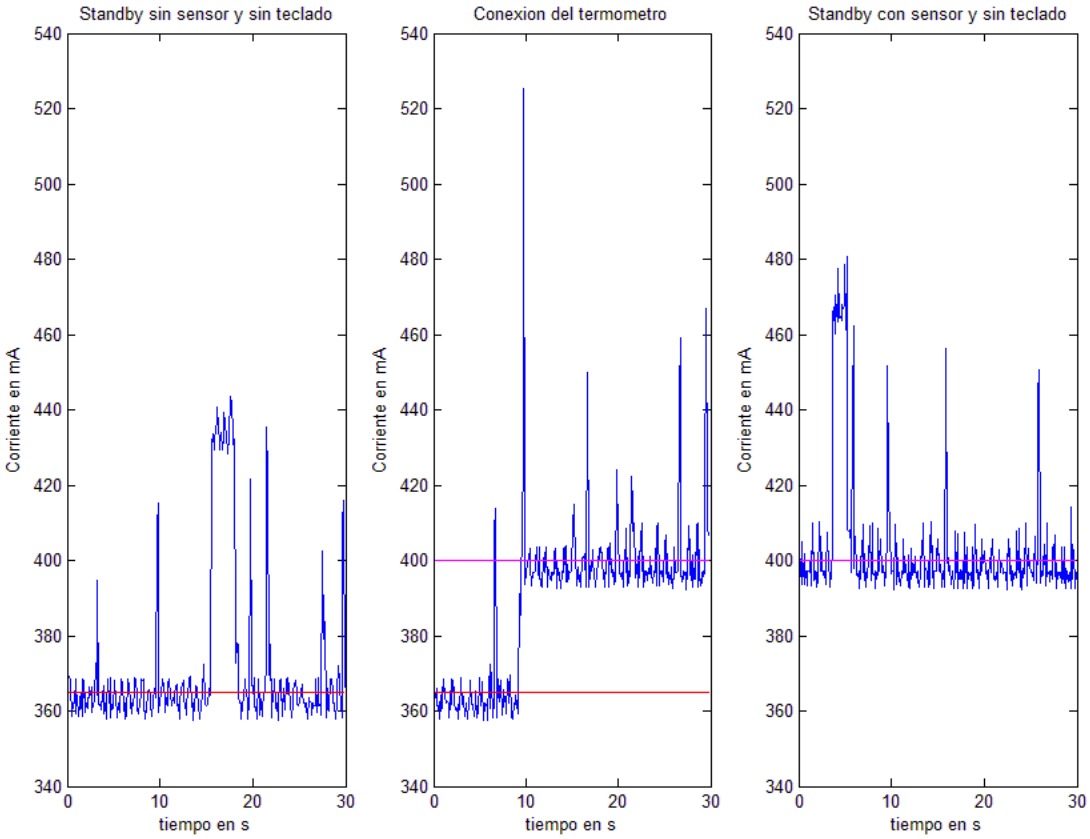


Ilustración 5.19: Corriente durante la conexión del sensor (1)

La línea magenta que aparece en las imágenes central y derecha se corresponde con el valor 400 mA, mientras que la roja que aparece en las imágenes central e izquierda se corresponde con el valor 365 mA. Se comprueba la continuidad del consumo durante estos tres casos.

En las gráficas anteriores correspondientes a los periodos de inactividad se pueden observar una serie de picos de corriente que tienen lugar cada 10 segundos. Dichos picos, sea cual sea la corriente que está siendo consumida en cada instante, alcanzan un valor de 50 mA adicionales. Seguramente se correspondan con operaciones de sincronización del sistema.

A continuación se muestra una representación del consumo durante la ejecución del programa, el cual será compilado y ejecutado de forma consecutiva. Los picos correspondientes a la compilación del programa no serán tenidos en cuenta debido a que, una vez generado el ejecutable, no volverá a ser necesario realizar la compilación del programa para que éste pueda ser usado en el sistema. El consumo para estas tareas se muestra en la siguiente figura:

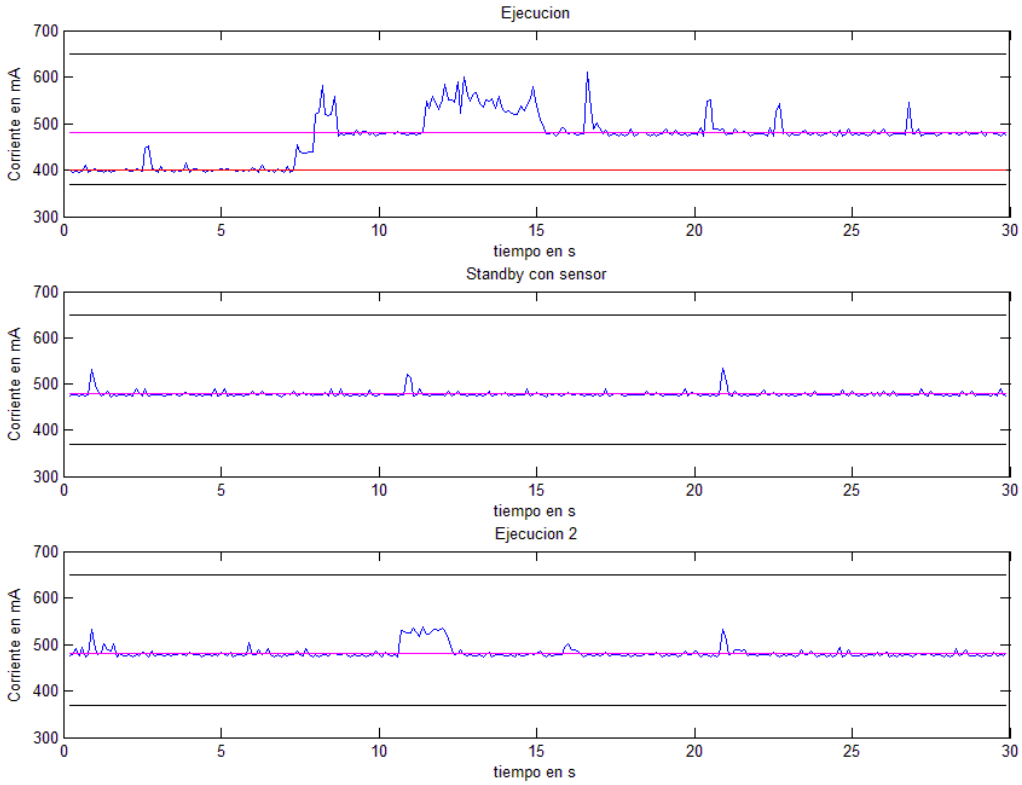


Ilustración 5.20: Ejecución, standby y ejecución

La línea roja de la primera imagen se corresponde con el valor de 400 mA, que coincide con el valor que tenía el consumo de corriente eléctrica cuando el sistema permanecía inactivo tras conectar el sensor. Al ejecutar el software se observa un pico que se corresponde con la compilación del programa y una etapa de estabilidad que se corresponde con el momento en el cual el programa pide al usuario el número de medidas que desea comprobar. Tras la introducción de este dato el pr46ograma termina de ejecutarse mostrando un breve periodo de mayor consumo para luego estabilizarse de nuevo en el valor que indica la línea magenta (480 mA) presente en todas las gráficas.

Tras la primera ejecución del programa, en la segunda gráfica, se genera un consumo constante de 480 mA, que permanece estable mientras no se realice ninguna acción más sobre el sistema.

Por último, en la tercera gráfica se muestra el consumo de corriente durante una segunda ejecución del programa. En esta gráfica se comprueba que el consumo inicial se corresponde con los 480 mA anteriormente mencionados y que los consumos durante la compilación (primeros valores de la gráfica) y la ejecución (pequeña “meseta” que

aparece más adelante) del programa son notablemente menores que los consumos medidos en las mismas condiciones durante la primera ejecución del programa. Este fenómeno se debe a que durante la primera ejecución se realizan operaciones de carga del programa en memoria mientras que en las sucesivas ejecuciones el programa ya está almacenado en la memoria RAM del sistema y únicamente se ejecuta, sin pasos previos.

En todas las gráficas anteriores se pueden observar los picos de corriente generados cada 10 segundos que ya han sido mencionados con anterioridad, por lo que serán tenidos en cuenta para el cálculo final del consumo de corriente eléctrica sea cual sea la fase de funcionamiento que se esté tratando.

Las gráficas siguientes reflejan los valores del consumo de corriente en el momento de desconexión del termómetro:

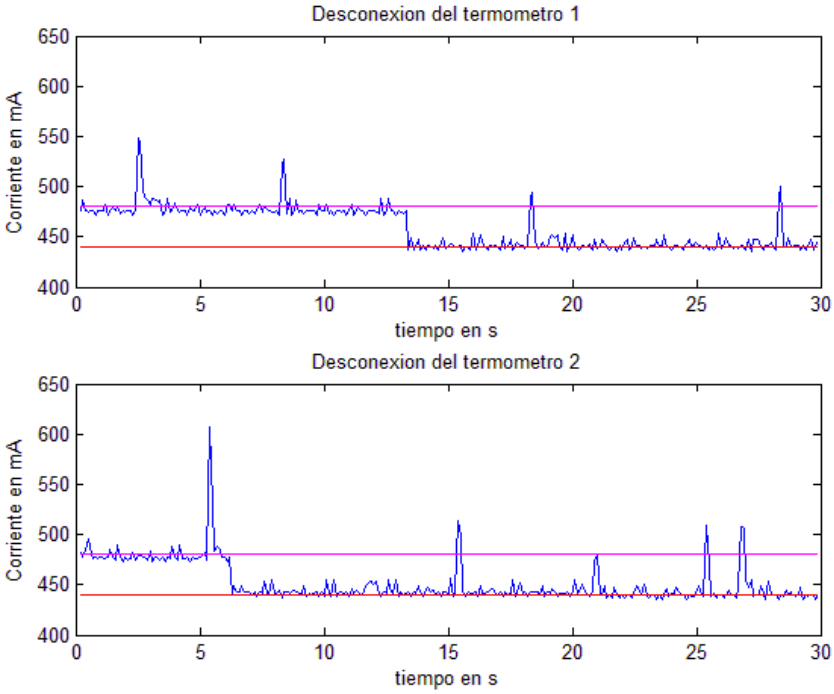


Ilustración 5.21: Desconexión del termómetro

Se puede comprobar que el consumo inicial, exceptuando los picos de corriente ya tratados, se estabiliza en torno a 480 mA (línea magenta), que coincide con el consumo de corriente durante los periodos de inactividad (con el sensor conectado). En el momento de la desconexión del sensor se produce una caída del consumo hasta el valor indicado por la línea roja (440 mA). El valor permanece estable mientras no se produzca ningún evento nuevo en el sistema.

En este punto pueden darse dos casos: o bien se vuelve a conectar el sensor, o bien se apaga el equipo. En el primero de los casos, se puede esperar que el comportamiento sea exactamente el contrario que lo observado en el último caso estudiado (desconexión del sensor), es decir, que partiendo de un consumo constante de 440 mA salte tras la nueva conexión a 480 mA. Este fenómeno se comprueba con la siguiente gráfica, que representa los datos recogidos realizando la conexión mencionada:

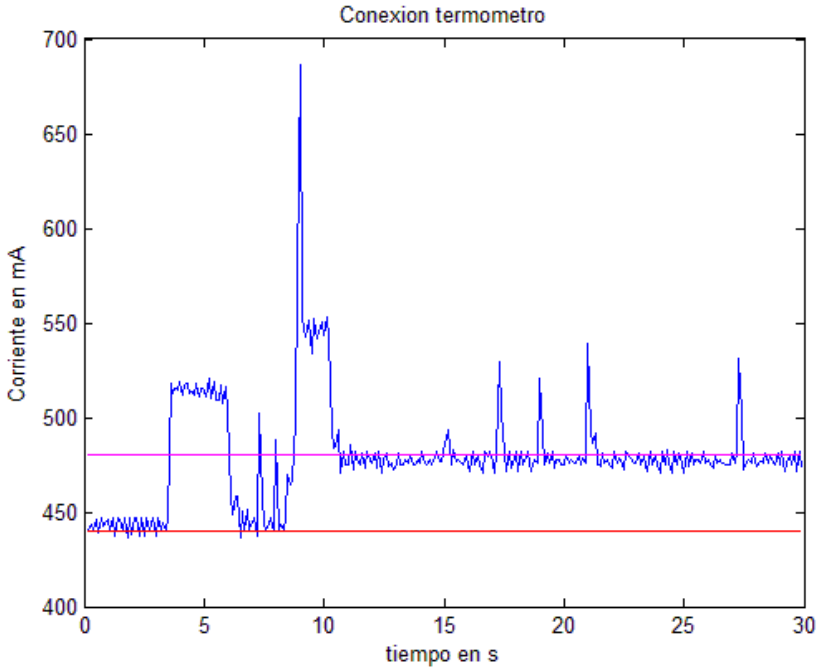


Ilustración 5.22: Corriente durante la conexión del sensor (2)

Las líneas roja y magenta representan los valores de 440 mA y 480 mA respectivamente. Se comprueba que el sistema cumple con lo esperado, es decir, realiza justo el comportamiento opuesto a la desconexión del sensor.

En el segundo de los casos mencionados anteriormente, apagar el sistema, se han recogido los siguientes datos:

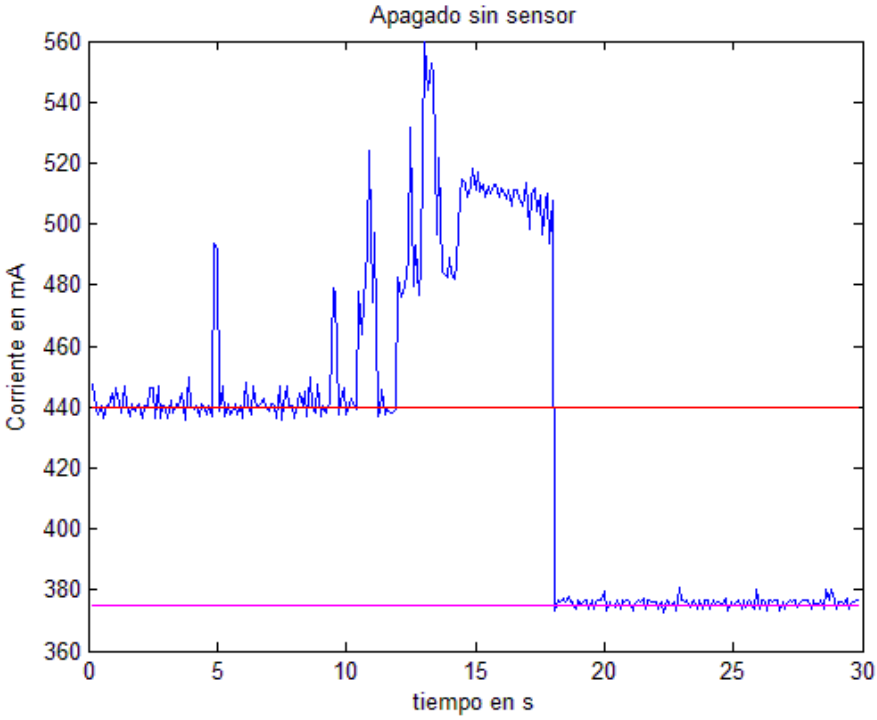


Ilustración 5.23: Apagado del sistema sin sensor

Se observa que, partiendo del consumo que se había estabilizado tras la desconexión del sensor (440 mA), se realiza un último salto de consumo de corriente que se corresponde con el proceso a través del cual el sistema concluye los procesos en ejecución y cierra el sistema operativo. A partir de este punto, el sistema permanece con un consumo constante de 370 mA, y puede ser desconectado en cualquier momento.

Las fases descritas anteriormente se corresponden con las diferentes posibilidades de uso que se pueden llevar a cabo con el sistema realizando las tareas para las que está diseñado. Sin embargo pueden darse casos más extraños a través de los cuales el consumo de corriente puede variar. Por ejemplo, se probó entre otros experimentos qué pasaría si se encendiera el sistema, se conectase el sensor y, sin ejecutar el software del sistema ni desconectar dicho sensor, se apagase. La gráfica resultante es la siguiente:

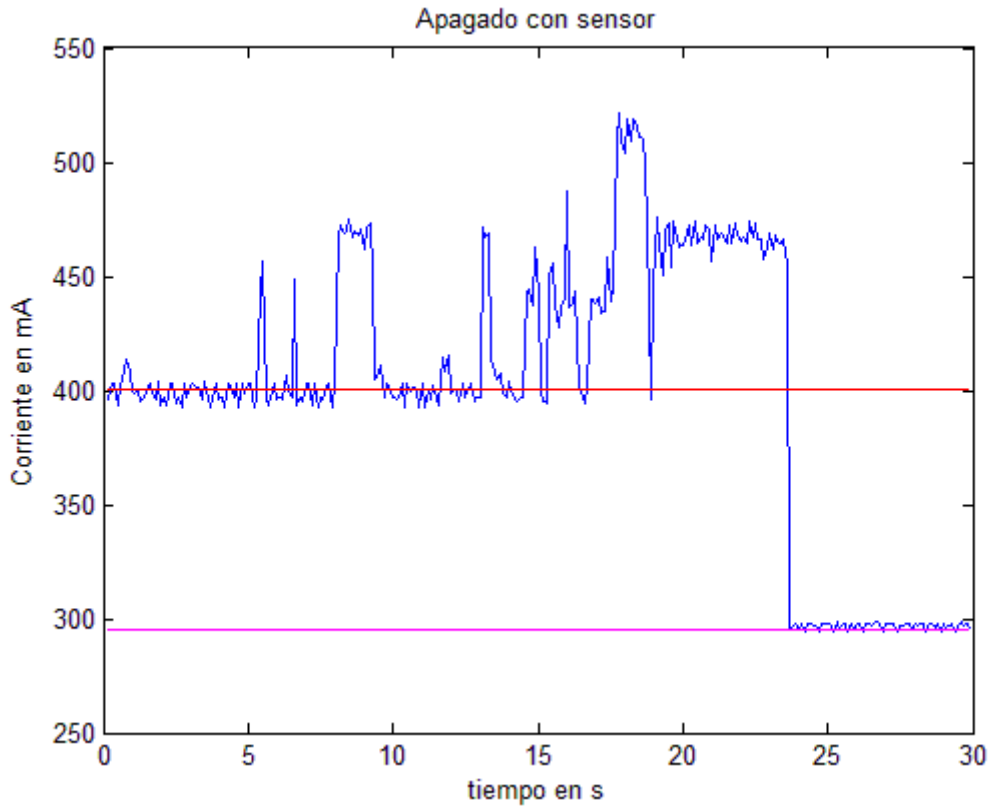


Ilustración 5.24: Apagado del sistema con sensor

Se puede observar que, a pesar de tener el sensor conectado (lo que a priori hace pensar que debería generar un consumo de corriente mayor) el sistema realiza un consumo mucho menor, partiendo de un consumo estable de 400 mA (en vez de los 440 mA mencionados en el caso anterior) a un consumo de 295 mA (en vez de los 370 mA del caso anterior).

Del mismo modo, otros usos atípicos del sistema generan un consumo menor que el realizado llevando a cabo un uso más convencional, de modo que se tomarán los datos

referentes al uso adecuado del sistema ya que, además de ser lo más común, genera un consumo de energía mayor. Esto se corresponde con lo que se podría denominar "el peor caso" de consumo, es decir, el caso en el que se consume mayor cantidad de energía, que es el que debe tenerse en cuenta a la hora de elegir la batería o fuente de energía que alimentará al sistema completo. Esto es así porque el sistema debe de estar suficientemente bien alimentado para no apagarse o llevar a cabo un funcionamiento anómalo o inadecuado sean cuales sean las acciones que se lleven a cabo.

Por último se va explicar el único paso que ha quedado incompleto, y éste es el paso que da la gráfica de la corriente entre el inicio del sistema y el estado de standby sin sensor. Entre estas dos gráficas se observa que hay un salto entre los 460 mA que se daban cuando se estabilizaba el consumo de corriente durante el inicio del sistema y los 365 mA en los que se estabiliza el sistema cuando está inoperativo, sin sensores conectados. Las medidas obtenidas para registrar ese cambio ha requerido de tomar medidas durante tres fases del inicio del sistema por separado: carga del sistema operativo (caracterizado por la aparición por pantalla del logotipo del SO Angstrom), el inicio automático de sesión y el inicio del escritorio. Estas fases se han tomado durante tres inicios del sistema diferentes debido a que el analizador sólo permite tomar medidas de 30 segundos, y tras cada medida se deben guardar los datos lo cual lleva un tiempo demasiado largo para tomar medidas completas. Se ha supuesto que el consumo de corriente no varía demasiado entre un inicio y otro. De este modo las medidas tomadas son las siguientes:

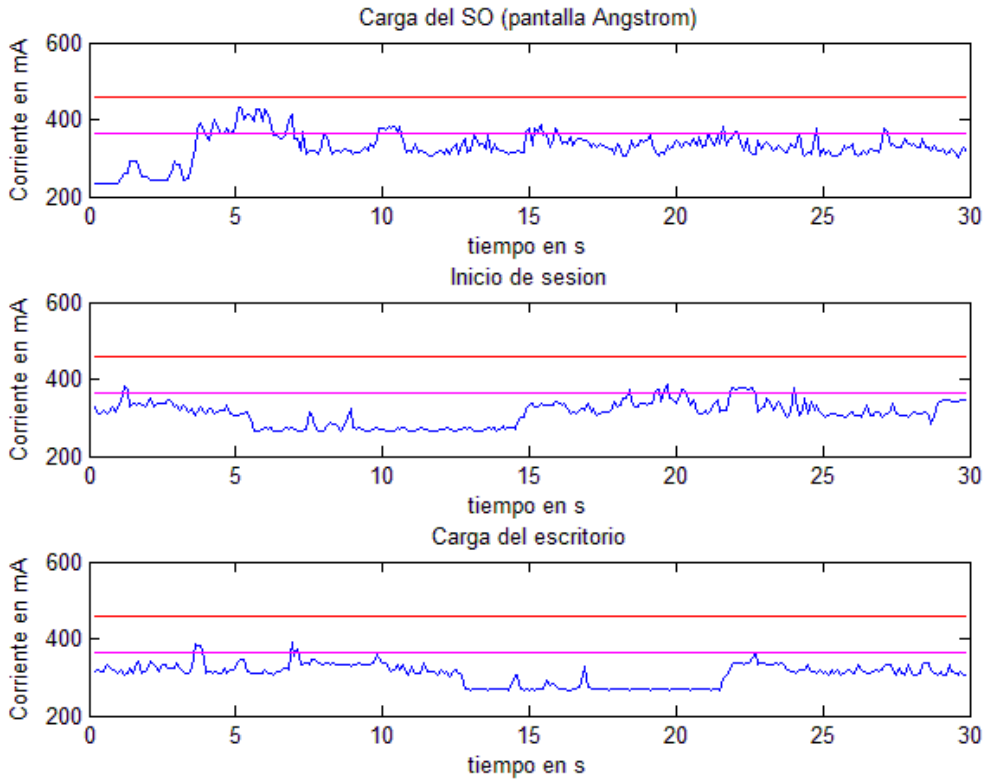


Ilustración 5.25: Encendido del sistema. Tres imágenes

Se observa como dato relevante que el consumo cae considerablemente entre cada una de las fases descritas anteriormente. La zona de bajo consumo de la segunda imagen se corresponde con el paso entre la carga del SO y el inicio de sesión, así como la zona de bajo consumo de la tercera imagen se corresponde con el paso entre el inicio de sesión y la carga del escritorio. Se comprueba que hay grandes variaciones de consumo durante las últimas etapas del inicio del sistema, por lo que el paso entre el inicio del sistema y el estado inoperativo queda reflejado en las gráficas anteriores.

❖ CARGA Y CORRIENTE CONSUMIDAS

Con las medidas anteriormente tomadas se puede realizar el cálculo del consumo de carga y de corriente del sistema completo.

- Durante el encendido, el sistema tarda una media de 65 segundos entre el inicio y la carga completa del escritorio. Durante este proceso el consumo de corriente varía notablemente llegando a alcanzar corrientes de 541 mA. Durante varias mediciones independientes, se ha podido comprobar que durante el inicio del sistema se consumen 5.3 mAh como media (calculada a partir de un gran número de muestras).
- Durante los periodos de inactividad, se calculará la intensidad media que consume el sistema con el fin de realizar más adelante la estimación del tiempo en el que permanecerá de esta manera para calcular finalmente el consumo de carga. Existen cuatro tipos de periodos de inactividad bastante diferenciados:
    - Entre el inicio del sistema y la conexión del sensor, el consumo de corriente medio es de 365 mA.
    - Una vez conectado el sensor, el consumo asciende a una media de 400 mA.
    - Tras la ejecución del software, los periodos de inactividad presentan un consumo medio de 480 mA, sean cuantas sean las veces que se ejecute.
    - Por último, tras la desconexión del sensor, los periodos de inactividad presentan un consumo medio de corriente eléctrica de 440 mA.
- En lo referente a la ejecución del software se han observado diferencias entre el consumo durante la primera ejecución y las demás. La primera ejecución presenta un pico de corriente mayor (llegando a alcanzar los 612 mA mientras que en el resto de ejecuciones se ha llegado a registrar un pico de 537 mA) además de un consumo de carga media igualmente mayor. El consumo de carga durante la primera ejecución es aproximadamente de 1.1 mAh mientras que durante la segunda ejecución es de 0.92 mAh.
- Durante la conexión del sensor al sistema embebido también se generan corrientes adicionales a tener en cuenta. Dicha corriente produce un

aporte de carga de 0.092 mAh. La desconexión del sensor en cambio no produce ninguna corriente adicional.

- Por último, durante el apagado, el sistema consume una cantidad de energía extra para cerrar todos los procesos que se encuentren activos y finalizar su funcionamiento. Esta energía se traduce en una corriente que de media da lugar a un aporte de carga de 1.2 mAh.

De este modo se han obtenido los parámetros requeridos para poder efectuar finalmente el cálculo correspondiente al consumo estimado de energía durante una jornada de trabajo, que será realizado más adelante.

5.3.3. Estudio sobre el uso del sistema

Con los datos de los dos apartados anteriores se va a proceder a hacer un estudio del consumo total requerido por el sistema. Las medidas obtenidas en el apartado 4.3.2 ofrecen una buena aproximación al consumo real que tiene el sistema, pero para tener una idea más certera hay que tener en cuenta la forma de usar el sistema, puesto que dichas medidas también abarcan la alimentación al sensor durante la conexión de éste:

A grandes rasgos, el uso del sistema se resume a continuación:

- El dispositivo tiene como principal finalidad recibir las señales referentes a las constantes vitales del paciente, que son aportadas por los sensores.
- Dicho dispositivo se transportará en un carrito que disponga de una fuente de alimentación propia (autonomía), que será la encargada de alimentar tanto el dispositivo como los sensores.
- El personal del hospital llevará el Concentrador de Señales hasta la cama del paciente, donde los sensores serán usados para tomar las medidas necesarias.
- Estos sensores serán usados en este punto alimentados por la fuente de alimentación del CS, para lo que será necesario activar el sistema completo.
- En este punto se dará la primera fase del uso del dispositivo. Esta fase se caracteriza porque el CS permanece inactivo mientras que los sensores son usados para tomar las medidas del paciente.
- Tras esta acción, el sensor usado se conectará al CS, el cual ejecutará el software de extracción de datos.
- Por último, se desconectará el sensor y se apagará el sistema.
- Entre la toma de medidas de dos pacientes consecutivos, lo normal será que el personal del hospital no apague el dispositivo hasta que no terminen de tomarse todas las medidas que sean requeridas, por lo que será necesario realizar una aproximación del tiempo de uso del sistema.

Además del funcionamiento general del sistema (que es el aspecto más importante a la hora de realizar el análisis del consumo) se deben tener otros aspectos en cuenta como el hecho de que la compilación del programa (tenida en cuenta en las medidas del apartado anterior) no se darán en ningún caso una vez esté concluido el desarrollo del sistema. Por ello, dicho consumo no será contemplado en los cálculos.

❖ APROXIMACIÓN DEL CONSUMO EN UN PACIENTE

En primer lugar se realizará una aproximación sobre el uso que se le dará al sistema durante las tomas de medidas de un único paciente, para tener una idea de la alimentación total que requiere el sistema durante dicho periodo. A continuación se realizará una aproximación del uso del dispositivo en una jornada completa en el hospital, suponiendo un hospital real con la ocupación al 100% (peor caso).

En la toma de medidas de un paciente, en primer lugar se encenderá el sensor y se realizarán dichas medidas para, a continuación, activar el concentrador de señales que extraerá dichas medidas del sensor.

En lo referente al sensor, el consumo se resume a continuación:

- Inicio del sensor: carga de 0.0007 mAh.
- Supondremos un tiempo entre el inicio del termómetro y la toma de medidas de un minuto, lo cual consumiría una carga de 0.003 mAh.
- Medida: carga de 0.0021 mAh.
- Supondremos otros cinco minutos entre la toma de medidas y la conexión del sensor al concentrador de señales, es decir, otros 0.014 mAh.
- Conexión al concentrador de señales: suponiendo 3 minutos se entrega una carga de 0.066 mAh.
- Por último, para su finalización también se mencionó que se requería un consumo de carga de 0.0014 mAh.

El concentrador de señales se espera que sea activado para realizar la primera medida y que, entre distintos pacientes, se mantenga inactivo. De esta manera se concluye el siguiente consumo total para cada uso con pacientes individuales:

- Para el primer paciente se da un consumo total aproximado de **40.1492 mAh**:
  - Inicio del sistema: 5.3 mAh.
  - Se supondrán dos minutos entre el inicio del sistema y la conexión del sensor (corriente media de 365 mA): 12.17 mAh.
  - Conexión del sensor: 0.092 mAh.
  - Se supondrán dos minutos entre la conexión del sensor y la ejecución del software (corriente media de 400 mA): 13.4 mAh.
  - Primera ejecución: 1.1 mAh.
  - Se supondrá un minuto entre la ejecución del software y la desconexión del sensor (corriente media de 480 mA): 8 mAh.
  - La desconexión del termómetro no produce ningún consumo.

    - Durante este proceso, el termómetro consume 0.0872 mAh.

- Para pacientes intermedios se consume un total aproximado de 25.0992 mAh:
    - Conexión del sensor: 0.092 mAh.
    - En estos casos, la corriente media entre la conexión del sensor y la ejecución del software es de 480 mA, por lo que suponiendo de nuevo dos minutos entre las dos acciones se da un consumo de 16 mAh.
    - Ejecución del software: 0.92 mAh.
    - Suponiendo también un minuto entre la ejecución y la desconexión del sensor: 8 mAh.
    - Consumo del termómetro durante el proceso: 0.0872 mAh.
- Para el último paciente se da un consumo total aproximado de 38.6392 mAh:
    - Mismo proceso que para pacientes intermedios: 25.1102 mAh.
    - Cierre del sistema: 1.2 mAh.
    - Se supondrán dos minutos entre el cierre del sistema y su desconexión de la fuente de alimentación (corriente media de 370 mA): 12.34 mAh.

Por último, en términos de tiempo, se supondrán un total de 8 minutos de funcionamiento durante el inicio del sistema y la toma de medidas del primer paciente, 8 minutos para la toma de medidas de los pacientes intermedios, y 10 minutos para la toma de medidas del último paciente y el apagado y cierre del sistema.

A los cálculos anteriores quedaría añadir el consumo que generará el sistema durante el tiempo que se invierta en tomar las medidas de dos pacientes consecutivos. Para ello se estimará el tiempo medio que puede invertir el personal del hospital en trasladarse de una habitación a otra. Teniendo en cuenta que una enfermera será capaz de atender dos habitaciones en escasos minutos (supongamos cinco) y que también se darán descansos o momentos de inactividad por parte del dispositivo (supongamos un máximo de veinte minutos) se supondrá finalmente un tiempo medio de doce minutos entre las consecutivas tomas de medidas.

Cabe mencionar que este tiempo es el que se invierte entre el instante en el que se desconecta el sensor (tras atender a un paciente) y el instante en el que vuelve a conectarse (para atender al siguiente).

El periodo que se estudia en esta sección se corresponde con uno de los periodos de inactividad mencionados en el apartado 5.3.2, concretamente con el periodo de tiempo entre la desconexión del sensor y su reconexión en la siguiente sala. Durante este tiempo, el sistema consume una corriente media de 440 mA. Como ya se indicó que de media se supondrán doce minutos consumiendo esa corriente, se puede calcular que el consumo de carga durante este proceso es de 88 mAh.

Con estos cálculos y aproximaciones, cabe mencionar que el CS podrá ser utilizado para atender a unos 40 pacientes, lo cual indica un número de pacientes bastante realista. Para grandes hospitales será necesario un número mayor de dispositivos.

5.3.4. Consumo total y elección de batería

Como ya se indicó en el apartado 4.4 (requisitos de la batería), uno de los objetivos perseguidos en el desarrollo del CS es que tenga una autonomía de al menos 8 horas. En las aproximaciones estimadas en apartados anteriores, se expusieron los siguientes tiempos:

- 8 minutos invertidos en cada paciente (entre la conexión del sensor y su desconexión del CS).
- 12 minutos invertidos durante el traslado del CS entre las habitaciones (entre la desconexión del sensor y su nueva conexión al CS).
- 10 minutos invertidos para el último paciente (entre la conexión del sensor y el cierre definitivo del sistema).

Contando que para cada paciente se invertirán 20 minutos (sus 8 minutos más los 20 minutos de traslado hasta el siguiente) excepto para el último, para el que se invertirán únicamente 10 minutos (ya que no habrá traslado posterior del sistema), se obtienen un total de:

$$8\ h = 480\ min \rightarrow\ 470\ \text{min}\ \ (quitando\ \text{último})$$

$$^{470min}/_{20min} = 23.5 \rightarrow 23\ pacientes$$

Sumando el último se obtiene que se pueden atender un total de 24 pacientes en 8 horas. Como el requisito es que el CS tenga una autonomía de *al menos* 8 horas, se considerará un periodo de tiempo mayor que abarque 25 pacientes, de modo que se deberá encontrar una batería que atienda como mínimo a estos 25.

Para calcular el consumo total del sistema se usará la siguiente expresión:

$$C_{total} = C_{prim} + C_{tras} + n(C_{inter} + C_{tras}) + C_{ult}$$

Donde C_{total} es el consumo total, C_{prim} el del primer paciente, C_{inter} el de cualquier paciente intermedio, C_{ult} el del último paciente, C_{tras} el consumo durante el traslado del CS de una habitación a otra y n el número de pacientes total.

Se obtiene 40.1492 + 88 + 25(25.0992+88) + 38.6392 = 2906.27 mAh = 2.90627 Ah.

A este valor, por último, hay que añadirle los picos de consumo que se generaban en la placa cada 10 segundos (6 picos por minuto, cada uno de 50 mA durante un brevísimo periodo de tiempo) descritos y mencionados en diversos puntos del apartado

5.3.2. La forma de realizar el cálculo es, en primer lugar, calcular el número de picos que se han dado. Una vez hecho esto, se considerará un intervalo de 0.1s para cada pico. El cálculo queda:

$$6 \times 480 = 2880\ picos \rightarrow 2880 \times 0.1s \times 50mA = 14400mAs = 4mAh$$

Finalmente, el valor calculado para el consumo de carga ha sido de 2.91027 Ah.

El resumen de los requisitos que debe tener la fuente de alimentación del sistema, finalmente, son los siguientes:

- Voltaje mayor o igual que 5 V.
- Corriente de pico mayor o igual que 700 mA.
- Carga total igual o mayor que 3 Ah.

Por tanto, consultando la web www.amidata.es encontramos que la única batería de plomo ácido que encaja dentro de los requisitos anteriores (con un margen razonable) es la RS 727-0385, cuyas características técnicas se listan a continuación:

- Tensión nominal (voltaje): 6V.
- Capacidad (carga total) 3.2 Ah.
- Dimensiones: 134 x 34 x 66 mm.
- Peso: 0.67 Kg.
- Rango de temperatura: -15 → 50 ^{o}C.
- Tipo de terminal: Tabs.
- Tipo de plomo ácido: Ciclo profundo.

Los detalles restantes de las especificaciones se muestran en el anexo B.

Ilustración 5.26: Batería RS 727-0385

Capítulo 6

Integración, pruebas y resultados

6.1. Introducción

En este capítulo se presentan los puntos que han sido seguidos durante el montaje del sistema completo, indicando además qué partes de dicho montaje han diferido del diseño explicado en el capítulo 3 de la memoria.

La gran mayoría de pruebas de interés se han ido realizando durante la implementación del sistema. No obstante se expondrán errores que han requerido de un estudio para ser solventados, imprevistos de cara al montaje final del sistema o temas referentes al uso responsable de elementos del sistema tales como la memoria.

Otro detalle de gran interés es el hecho de que, aunque en un principio el sistema iba a disponer de tres sensores proporcionados al inicio del proyecto, finalmente se elaboró con uno solamente, el termómetro. Esto se debe a que en primer lugar, el oxímetro se descartó debido a que su funcionamiento no era afín con la finalidad del sistema completo. Para conseguir las medidas tomadas por este sensor se debía conectar en primer lugar al PC, una vez conectado tomar las medidas y por último, por medio de una combinación de botones de éste los datos eran transmitidos de golpe, imposibilitando la posibilidad de proveer una funcionalidad plug-and-play. El segundo sensor excluido del proyecto fue el tensiómetro debido a que se sometió a una gran tensión por accidente durante la medición de su consumo en el laboratorio, dejándolo inoperativo.

6.2. Montaje final del sistema

El montaje del sistema ha consistido esencialmente en la interconexión de todos los elementos hardware del sistema. Como todo sistema computacional, ha consistido en un sistema de procesado y una memoria, ambos incluidos en el sistema basado en microcontrolador, y en periféricos de entrada y de salida.

Los elementos que han sido usados finalmente se listan a continuación:

- Tarjeta de procesado: placa BeagleBone.
- Dispositivos de entrada: sensores (y teclado opcional).
- Hub USB, que permita la conexión de varios periféricos simultáneamente.
- Dispositivo de salida: pantalla BeagleBone Cape LCD4.
- Fuente de alimentación (batería).

El elemento que marca la principal diferencia entre el diseño y el montaje final es el dispositivo de salida, que en vez de ser la Tablet es un display diseñado específicamente para la tarjeta de procesado BeagleBone. La ventaja más notable de este cambio es la plena compatibilidad entre este nuevo elemento y la tarjeta, lo cual supone un ahorro muy significativo de tiempo y esfuerzo ya que desaparece la necesidad de elaborar un protocolo de comunicación con sus respectivas aplicaciones *frontend* y *backend*.

6.2.1. BeagleBone Cape LCD4

Para utilizar esta expansión de la BeagleBone se requiere disponer de la propia BeagleBone y del display, así como de una fuente de alimentación que recomendablemente debe proporcionar una tensión de 5 V y una corriente de 2 A. Los pasos a seguir durante su primer uso han sido los expuestos en la página web *http://elinux.org/CircuitCo:BeagleBone_LCD4*:

- Conectar ambas tarjetas a través de los pines de expansión.
- Asegurarse de que la imagen del sistema operativo almacenado en la tarjeta SD es compatible con el display.
- Conectar la fuente de alimentación de 5 V al puerto de alimentación de la BeagleBone.
- Seguir las instrucciones de calibración para la entrada táctil de la pantalla.
- Tras el calibrado, se mostrará el escritorio para poder ser usado libremente.

El primer problema presentado fue la incompatibilidad de la versión del sistema operativo proporcionado con la tarjeta con la pantalla.

Para solventarlo se acudió a la página mencionada anteriormente, donde aparece en un apartado llamado "Soporte Software" la indicación "*BeagleBone LCD4 Cape revision A1 es soportado por Angstrom release 11-22-12 en adelante*". Los pasos para actualizar el sistema operativo del sistema y crear una tarjeta SD con dicha actualización en Windows han requerido de software adicional, concretamente:

- Win32DiskImager: una utilidad para escribir imágenes en unidades extraíbles.
- 7-zip: programa de compresión y descomprensión de datos.
- HP Format Tool: herramienta capaz de formatear la tarjeta SD.

Dichos pasos han sido descargar la imagen *BeagleBone Rev A6A Production 11-22-12*, descomprimirla con la herramienta 7-zip, formatear la tarjeta SD con HP Format Tool y escribir la imagen descomprimida en la tarjeta con el programa Win32DiskImager.

Con estos pasos se obtiene una tarjeta SD desde la que iniciar la tarjeta BeagleBone con el sistema operativo actualizado. Con ella insertada la pantalla LCD4 funciona automáticamente sin necesidad de programar nada.

6.2.2. Inicio del sistema

En el capítulo 4 se explicó la forma de compilar los programas elaborados desde el ordenador para que puedan funcionar correctamente en procesadores ARM. Este método usaba el software Eclipse IDE y una serie de herramientas que hacían posible la compilación cruzada. Ésta, se hizo innecesaria con la incorporación del display al sistema ya que con este dispositivo se hizo posible la compilación por medio del terminal del sistema operativo Angstrom a través de un teclado que ejecutase los comandos necesarios. Con este nuevo sistema, se puede introducir el código fuente del software por medio de la tarjeta SD.

Con el sistema actualizado, se introduce el software del sistema en la tarjeta SD, se copia en la carpeta en la que se inicia la terminal de Linux por defecto y se elabora un archivo Makefile que automatice su inicio.

A continuación, para realizar el montaje del hardware del sistema, se conecta el Hub USB en el puerto USB de la BeagleBone y se conectan los sensores (con las medidas correspondientes ya tomadas) en los puertos de dicho Hub. La pantalla se montará sobre la superficie de la tarjeta, quedando ésta como base del montaje. Por último, se conecta la batería al sensor y al puerto de alimentación de la BeagleBone a través de un interruptor para cada conexión. En este punto ya se podría iniciar el sistema.

6.3. Pruebas y ajustes

En este apartado se expondrán aspectos relativos al funcionamiento del sistema durante las primeras pruebas de éste, los problemas encontrados y su solución, así como los aspectos referentes al uso responsable de los recursos del sistema por parte del software (uso de memoria y de procesador).

Los problemas más básicos fueron solventados durante el diseño del sistema, tales como la elaboración de una estructura software capaz de registrar y almacenar datos de forma temporal y el flujo de estos datos a través de dicha estructura. Durante la implementación del software también se encontraron problemas a la hora de desarrollar los diferentes TAD que han formado la estructura del sistema, tanto a la hora de establecer sus atributos como sus métodos.

Sin embargo, a pesar de todo el desarrollo de la estructura software que a priori debería haber funcionado perfectamente, durante las pruebas apareció un nuevo problema imprevisto que repercutió de forma más notable en la salida de los datos. Dicho problema se debió a un desbordamiento en la representación de datos. Será explicado con más detalle a continuación.

6.3.1. Overflow en el registro de datos

El problema de desbordamiento (*overflow*) no fue detectado durante un gran periodo de tiempo. Esto se debió a que durante el primer periodo de pruebas el sistema completo pareció funcionar perfectamente. Sin embargo, un día al azar, se generó una salida de datos bastante inesperada e inapropiada, representando los días de las medidas con números negativos y los meses con un cero. La siguiente imagen muestra la salida descrita con un trozo de código cuyo detalle se explicará más adelante:

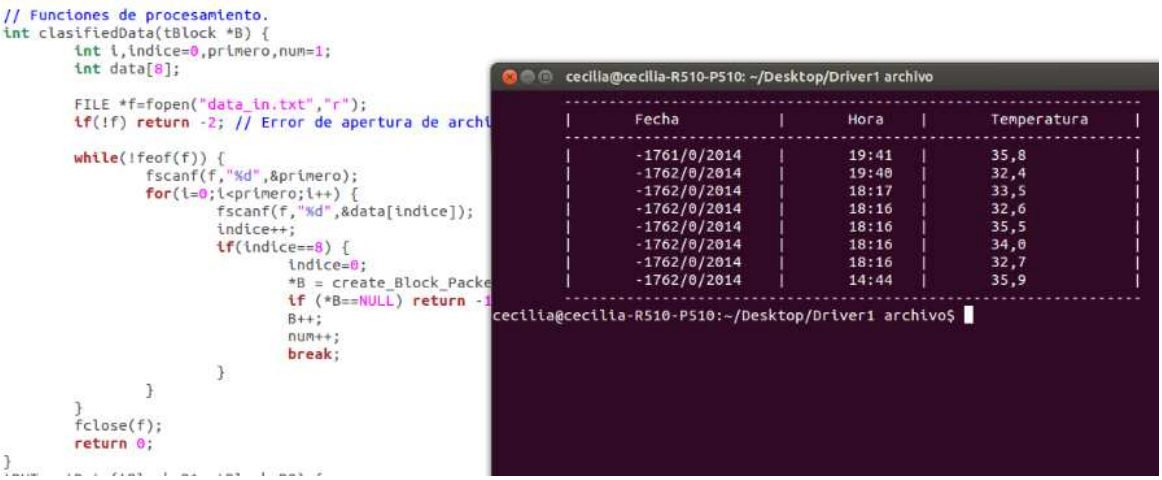


Ilustración 6.1: Error con overflow

Se observan los fallos mencionados anteriormente (día negativo y mes nulo). Los datos referentes a la hora y a la temperatura sin embargo, se representan correctamente.

Tras realizar un estudio sobre el flujo de los datos procedentes del termómetro se encontró que el error de representación procedía de un error anterior de desbordamiento de datos. En primer lugar, a modo de recordatorio, se expondrá el esquema del flujo de datos para poder basar la explicación del problema en dicha estructura:

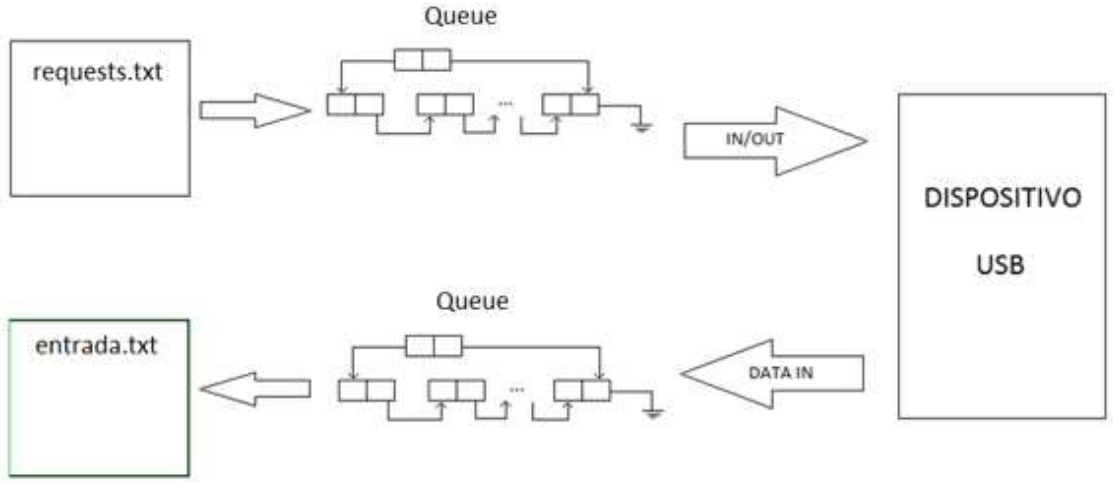


Ilustración 6.2: Flujo de datos

Suponemos con toda certeza que el termómetro envía los datos correctamente, por lo que se comenzará la depuración del programa a partir de la cola que recibe los datos. Esta cola almacena punteros genéricos a estructuras que almacenan los datos correspondientes a los parámetros y datos de los envíos y recibos entre el PC y el sensor. Uno de los campos de estas estructuras es un array que contiene los datos que han sido recibidos desde el termómetro. Dichos datos son escritos sin ningún tipo de procesado en un archivo llamado *entrada.txt* y son leídos desde otro TAD que almacena dichos datos como enteros para tratarlos más adelante.

El problema en este planteamiento es que, por requerimiento de la propia librería *libusb.h* los datos tanto enviados como recibidos deben ser declarados como datos de tipo *char*. Sin embargo los datos son leídos como datos enteros. Esto generará un problema en los casos en los que los datos registrados como *char* tengan un 1 como bit más significativo, ya que al ser leídos como datos enteros se interpretará que el valor del registro es negativo.

Los datos de tipo *char* tienen un registro de 1 byte, mientras que los enteros constan de 4 bytes. De este modo si, por el motivo que sea, en el archivo *entrada.txt* se almacena un valor con dicho bit a 1, por ejemplo 10010011, será representado como un entero con sus bits más signifivativos a 1, en el ejemplo anterior quedaría como:

11111111 11111111 11111111 10010011

Este problema sólo se dio en los días concretos en los que dicho bit se representó con un 1, generando que el software funcionase correctamente algunos días y otros no.

Para solventar este problema se consideró por realizar un tratamiento condicional de los datos antes de escribirlos en el archivo diseñando funciones concretas que evitasen dicho problema, pero finalmente se optó por una solución mucho más simple tanto en esfuerzo como en consumo por parte del PC. La idea de dicha solución nace en el hecho de que el dato deseado se ubica inalterado (a pesar de todo) en el byte menos significativo del registro de 4 bytes de los datos enteros (como puede comprobarse en el ejemplo anterior). Por tanto, con una simple operación lógica, concretamente la operación AND se obtendría el dato deseado. La operación AND se realiza de la siguiente manera:

Dato 1	Dato 2	Resultado
0	0	0
0	1	0
1	0	0
1	1	1

El lenguaje de programación C ofrece está funcionalidad por medio del ampersand, el cual cuando es usado con registros en este caso de 4 bytes realiza la operación bit a bit. De modo que con añadir una simple instrucción "*dato*&0x000000FF" ya se obtendría el dato deseado para su posterior tratamiento y representación. El sistema corregido de esta manera funcionó correctamente, tal como se muestra en la siguiente figura:

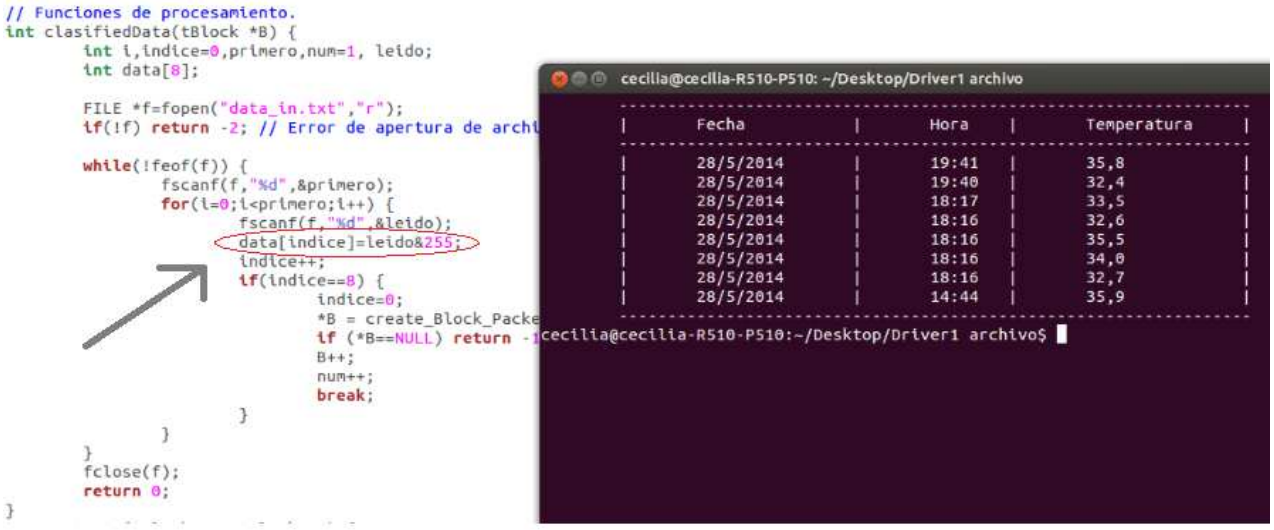


Ilustración 6.3: Error con overflow. Solucionado

En esta imagen se muestran las mismas medidas que en la imagen que mostraba el fallo. Se comprueba la corrección del problema. Por otro lado, se aprecia el código fuente del programa de fondo, donde se ha realzado la instrucción AND que ha

solventado dicho error. En el código el operador AND se aplica con un valor de 255 porque es el valor decimal de 0xFF.

Frente a la aparición de este fallo se consideró la posibilidad de que el resto de magnitudes pudieran sufrir el mismo error, por lo que se realizó un estudio sobre el posible desbordamiento en ellas. La posibilidad de fallos también resultó ser solventada con la misma instrucción, sin necesidad de añadir más código.

6.3.2. Uso de recursos del sistema

La estructura software del sistema ha requerido crear reservas dinámicas de memoria para registrar eficientemente los TAD descritos en el apartado 4.2.3. Además de estas reservas de memoria, han hecho falta algunas reservas más adicionales para almacenar información de manera temporal.

Estos hechos han generado un problema cuya resolución ha sido prioritaria: la correcta gestión de la memoria del sistema.

Al estar usando el lenguaje de programación C en Linux, las instrucciones usadas para realizar las reservas dinámicas de memoria han sido las incluidas en la librería *stdlib.h*, que pueden ser liberadas con el comando *free*.

Asimismo, otra herramienta de vital importancia ha sido la herramienta *valgrind* usada durante la compilación del programa. Su funcionamiento abarca tanto la devolución de errores y advertencias de todo tipo durante la compilación del programa hasta la devolución de errores, advertencias y resúmenes sobre el uso de la memoria del sistema durante la ejecución del programa. Usando esta herramienta se ha obtenido lo siguiente:

```
Cola database eliminada correctamente
Cola receive eliminada correctamente
Released Interface

Device closed correctly

        ------------------------------------------------------------------
        |       Fecha           |       Hora    |       Temperatura     |
        ------------------------------------------------------------------
        |       24/3/2014       |       23:42   |       36,2            |
        |       24/3/2014       |       22:29   |       35,5            |
        |       24/3/2014       |       22:29   |       35,8            |
        |       24/3/2014       |       22:25   |       35,7            |
        |       24/3/2014       |       21:47   |       36,2            |
        |       24/3/2014       |       21:47   |       35,7            |
        |       24/3/2014       |       16:54   |       34,2            |
        |       24/3/2014       |       16:53   |       36,0            |
        ------------------------------------------------------------------
==6659==
==6659== HEAP SUMMARY:
==6659==     in use at exit: 0 bytes in 0 blocks
==6659==   total heap usage: 1,881 allocs, 1,881 frees, 243,978 bytes allocated
==6659==
==6659== All heap blocks were freed -- no leaks are possible
==6659==
==6659== For counts of detected and suppressed errors, rerun with: -v
```

Ilustración 6.4: Memoria usada

Se recalca que en las últimas líneas se ubican los mensajes devueltos por el programa *valgrind* como se muestra a continuación:

Total heap usage: 1881 allocs, 1881 frees, 243978 bytes allocated

All heap blocks were freed – no leaks are posible

Se concluye afirmando que todos los bloques de memoria reservados dinámicamente durante la ejecución del software del sistema han sido liberados correctamente.

Por otro lado, usando otros comandos de Linux se analizará el uso de memoria y procesador por parte del programa durante su ejecución. Dicho comando será usado en un PC con Ubuntu, mientras que la tarjeta BeagleBone dispone de otro sistema operativo, Angstrom. Ambos sistemas operativos son distribuciones de Linux, por lo que comparten kernel lo cual, además de que ambas proporcionan los comandos (propio de Linux), se espera que den un resultado similar. La razón por la cual se usa el PC es porque en la tarjeta, la pantalla es demasiado pequeña, y sólo muestra los tres primeros procesos de la lista (entre los cuales no se encuentra el programa).

Los comandos que han sido utilizados para comprobar este consumo han sido *top* y *ps aux.* El primero de ellos es capaz de proporcionar el consumo de memoria y de CPU, entre otros parámetros, para cada proceso en tiempo real, actualizándose cada pocos segundos. El segundo sencillamente proporciona una lista con todos los procesos (en cualquier estado, incluso zombis) que se existan en el momento en el que se ejecuta el comando con sus consumos de CPU y memoria, entre otros.

Usando el comando *top* directamente no se ha llegado a apreciar la aparición del proceso correspondiente al programa. Esto se debe a que el programa *top* sólo muestra los procesos con mayor consumo de todos, lo cual sólo da lugar a dos opciones: o bien nuestro programa consume muy pocos recursos, o bien su duración es suficientemente breve para que el comando *top* no lo detecte.

A continuación se ha usado el comando *ps aux*, el cual ha sido capaz de mostrar el consumo del programa, indicando que el consumo era del 0.0% tanto de memoria como de CPU. Estos datos indican que el consumo de recursos del sistema por parte del programa es suficientemente pequeño como para que no suponga un problema.

6.3.3. Pruebas sobre el consumo de la batería

En cuanto al consumo de batería, en primer lugar se va a detallar el montaje del sistema que se ha llevado a cabo:

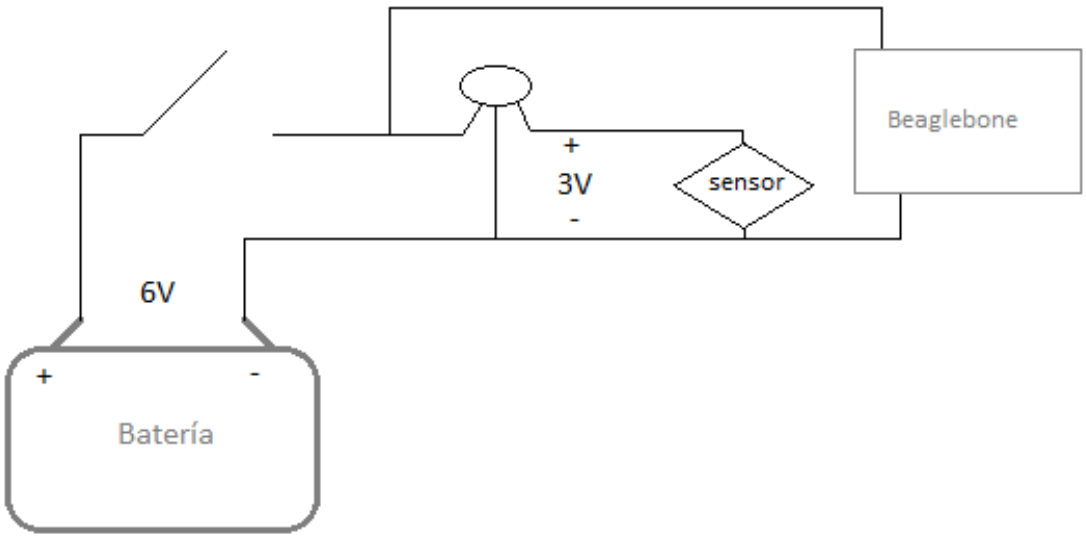


Ilustración 6.5: Esquema del montaje con batería

En el esquema anterior se muestra la batería alimentando los dos elementos que requieren una fuente de alimentación autónoma (el resto de elementos serán alimentados a través de los puertos USB). El elemento situado sobre la batería es un interruptor que estará situado frente al nodo en el que se bifurca el circuito para alimentar los dos elementos en paralelo con la finalidad de cerrar y abrir el sistema totalmente. Con los 6 voltios de la batería se alimentará el sistema completo mientras que para el sensor hará falta añadir un elemento que rebaje el voltaje hasta los 3 voltios que necesita. Dicho elemento dispone de tres patas: la del medio corresponde con la toma de tierra, la izquierda con el voltaje de entrada y la derecha con el voltaje de salida.

Con este montaje se ha comprobado que el sistema funciona correctamente. Para cargar la batería bastará con conectar un adaptador a los bornes de ésta.

Se recomendará tener el sistema con el interruptor abierto (sistema apagado) durante la carga.

6.4. Comentario del resultado

El sistema diseñado finalmente consiste en un sistema capaz de obtener las medidas tomadas con un sensor totalmente ajeno al proyecto (usando su propio protocolo de comunicación), de procesar dichas medidas y de mostrar tanto la medida tomada como la fecha y la hora a la que fueron tomadas.

Este proyecto se complementaría con el uso de un dispositivo que hiciese de base de datos al que se conectase el CS de forma inalámbrica, de modo que se pudiesen registrar las medidas tomadas por el CS relacionadas con el nombre de los pacientes. Con este otro dispositivo se crearía un sistema con funcionalidad completa en lo que se refiere a toma de medidas, clasificación y registro, dejando los datos referentes a las constantes vitales de los pacientes de forma clara, ordenada y amigable para el personal del hospital.

Con este sistema se logra un avance en las tomas de medidas de constantes vitales en hospitales, ya que el personal de enfermería sólo se limitaría a realizar la toma de medidas con los sensores correspondientes. Toda la gestión de los datos de los sensores sería llevada a cabo por el dispositivo desarrollado en el presente proyecto, encargándose éste de su obtención, tratamiento y clasificación.

A toda esta funcionalidad es necesario añadir también el importante papel que juega la batería del sistema, funcionando como fuente de alimentación independiente ya que gracias a este componente, el sistema es capaz de ofrecer dos importantes características o ventajas: la autonomía y la movilidad.

A continuación se muestra una imagen del sistema completo, formado por la tarjeta de procesado, su pantalla, el *hub* USB, el sensor y un pequeño teclado USB que se usará como método de entrada para el prototipo:

Ilustración 6.7: Sistema en funcionamiento

Por último cabe mencionar que el dispositivo se completaría añadiendo una simple carcasa que cumpliese las funciones de soporte y protección del sistema completo. Dicha carcasa se equiparía con una capa de aislamiento y con una conexión a tierra con el fin de aportar seguridad al equipo así como de cumplir con la normativa IEC 60601.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Características del dispositivo final

El primer aspecto a mencionar dentro de las conclusiones del proyecto es realizar un listado con las especificaciones técnicas del dispositivo final. Dichas especificaciones serán las que se mostrarían si este dispositivo saliera al mercado, y se resumen a continuación:

- Procesador ARM Cortex-A8 a 720 MHz.
- 2 CPU de 32 bits tipo RISC.
- RAM DDR2 256 MB.
- Pantalla de 4 pulgadas con entrada táctil como interfaz de usuario.
- Batería de plomo ácido con capacidad para una autonomía de 8 horas.
- Autonomía durante tiempo de uso:
- Autonomía durante tiempo en espera:
- Capacidad para gestionar las medidas del termómetro iONHealth.
- Posibilidad de ampliación en la gestión de medidas a otros tipos de sensores tales como tensiómetros u oxímetros.
- Rango de temperaturas: -15 → 50 ^{o}C. (*El rango más restrictivo es el de la batería.)

7.2. Cumplimiento de la normativa

Otra tarea importante para cerrar el proyecto será ubicar el dispositivo dentro de un sistema estándar de medidas clínicas y de comprobar su cumplimiento con la normativa.

En el primer caso, respecto a la norma CEN/ISO IEEE 11073, formaría parte de la arquitectura general del sistema, concretamente a un **elemento agente** que, a modo de recordatorio, eran aquellos encargados de recopilar los datos y de ofrecérselos a un sistema de gestión de datos central. Esto se hace evidente en el comportamiento del CS, ya que como ya se ha definido en numerosas ocasiones, es un dispositivo encargado de recoger las señales procedentes de los sensores y de desensamblarlas para obtener los datos sobre los que trabajará el sistema completo para enviarlos posteriormente a un dispositivo de salida.

Entrando en el árbol definido en este estándar, llamado DIM (Modelo del Dominio de la Información) este dispositivo se ubicaría en el **paquete de comunicación** (Communication Package), que se definía como el elemento encargado de hacer posible el intercambio de información con dispositivos propietarios, siendo posible la modificación o la ampliación de su estructura interna de cara a poder incluir la comunicación con dispositivos ajenos al desarrollo del sistema. Esto se comprueba en el hecho de que el CS tiene como finalidad actuar de interfaz con sensores con protocolos propios, ajenos al estándar. Del mismo modo, no sería difícil modificar el dispositivo para que fuese compatible con un mayor número de sensores cuyo número quedaría únicamente limitado por la cantidad de interfaces físicas de conexión de las que dispusiera el CS (que también podría adaptarse para aumentarlo).

Asimismo, también se definía en el núcleo del sistema unos elementos llamados **aplicaciones agentes** que eran las encargadas de adaptarse a los protocolos propietarios de los dispositivos ajenos al desarrollo del sistema (los sensores en este caso). El módulo de entrada de datos del CS sería el responsable de llevar esta tarea a cabo.

En el caso de las normas IEC 60601 referentes a la seguridad de los dispositivos electrónicos en entornos biomédicos, será necesario añadir una carcasa al dispositivo que tuviera conexión a tierra con alguna de las estructuras indicadas en el apartado 3.4. Del mismo modo también sería necesario añadir elementos como los amplificadores de aislamiento para cumplir con la protección frente a las posibles sobretensiones, que pueden ser producidas por errores aleatorios tales como rayos o fallos en la alimentación del sistema. Por último, también sería necesario añadir condensadores o dispositivos similares para proteger el sistema frente a caídas de tensión. Estos dos últimos elementos podrían ser añadidos en un mismo módulo hardware, pero se dejarán como trabajo pendiente para trabajo futuro ya que se quedan fuera del objetivo del presente proyecto.

Las normas referentes a elementos con comunicación inalámbrica se añadirían en el caso de incluir un módulo de comunicación para sensores inalámbricos (bluetooth), que también se quedan fuera del ámbito del proyecto.

7.3. Cumplimiento de los requisitos

A continuación se realizará un recordatorio de los requisitos iniciales del dispositivo para comprobar si han sido cumplidos o no. Estos requisitos son:

- El software debe ser capaz de admitir nuevos sensores. Cumplido.
- El software tiene que tener la capacidad de trabajar con varios sensores al mismo tiempo. No cumplido.
- El sistema debe ser capaz de recibir datos de sensores conectados por medio de una interfaz USB. Cumplido.
- Los datos obtenidos deben ser desensamblados y tratados para su consecuente envío al monitor. Cumplido.
- Para poder llevar todas sus labores correctamente, el sistema debe ser modular, es decir, debe estar compuestos de apartados específicos encargados de realizar tareas concretas. Cumplido, expuesto en el capítulo del diseño del sistema.
- Las operaciones llevadas a cabo por el software no deben dar lugar a un consumo excesivo de energía ni de recursos del sistema. Cumplido, indicado en el capítulo de pruebas y resultados.

Los requisitos iniciales referentes a la fuente de alimentación del sistema eran que debía tener una duración estimada de 8 horas con un tiempo de carga menor a 1 hora conectada a la red eléctrica. Dichos requisitos también se han cumplido satisfactoriamente.

7.4. Trabajo futuro

En el capítulo del estado del arte, se mencionó que la finalidad de dicho capítulo era exponer (entre otras cosas) cuál es el estado de los dispositivos comerciales similares al desarrollado en este proyecto en la actualidad. Es evidente que para que el dispositivo

desarrollado fuese competitivo en el mercado sería necesario ampliar sus funcionalidades.

En este apartado se listarán las posibles futuras líneas de trabajo de cara a mejorar el diseño del dispositivo o a complementar sus funcionalidades en lo que respecta a rendimiento, interactividad con otros dispositivos o aumento de compatibilidad con un mayor número de sensores. Así, los principales puntos sobre los que se basará el trabajo futuro de este proyecto son:

- Desarrollar drivers para otros sensores (ya sean sensores de otros tipos o de otras marcas) que podrían estar basados en el driver desarrollado en este proyecto de cara a ahorrar tiempo.
- Diseñar nuevas interfaces de conexión con dispositivos como Tablet o similares, ya sean inalámbricos o por cable, para dar una mayor flexibilidad en cuanto a la gestión de los datos y a su representación gráfica.
- Complementar la funcionalidad con capacidad para conectar el dispositivo a Internet. La idea sería facilitar el intercambio de datos entre el CS y otros dispositivos ya sean de almacenamiento o de salida de datos. No obstante, la pantalla del CS seguiría siendo interesante mantenerla de cara a disponer de un modo autónomo para la toma de medidas.
- Con la finalidad de cumplir con las normas IEC 60601 sobre la seguridad del dispositivo, será importante diseñar una caja o carcasa para el dispositivo que cumpliera con dichas normas siguiendo las líneas expuestas en el apartado 7.2 así como un módulo con amplificadores de aislamiento y condensadores de protección frente a sobretensiones y bajadas en la alimentación del sistema.

PARTE II

Apéndices

Anexo A

Captura de datos recibidos

Request	I/O	EndPoint	Buffer Snippet	Buffer Size
CLASS_INTERFACE	OUT	0		0
CONTROL_TRANSFER	IN	0	46 0A 01	3
CLASS_INTERFACE	OUT	0	41 01	2
CONTROL_TRANSFER	IN	0		0
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 24 00 00 00 00 A3 18	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 24	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 17	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 11	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 03	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	02 A5 45	3
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 24 00 00 00 00 A3 18	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 24	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 17	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 11	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 03	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 45	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 2B 00 00 00 00 A3 1F	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 2B	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 02	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 00	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	02 A5 23	3
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 25 00 00 00 00 A3 19	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 25	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 87	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 1C	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 05	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 13	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 D6	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 26 00 00 00 00 A3 1A	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 26	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 53	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 53	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 C4	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 25 01 00 00 00 A3 1A	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 25	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 87	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 1C	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 05	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 13	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 D6	2
BULK_OR_INTERRUPT_TRANSFER	OUT	3	08 51 26 01 00 00 00 A3 1B	9
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 51	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 26	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 65	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 65	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 01	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 A5	2
BULK_OR_INTERRUPT_TRANSFER	IN	83	01 E8	2

Anexo B

Especificaciones de los dispositivos

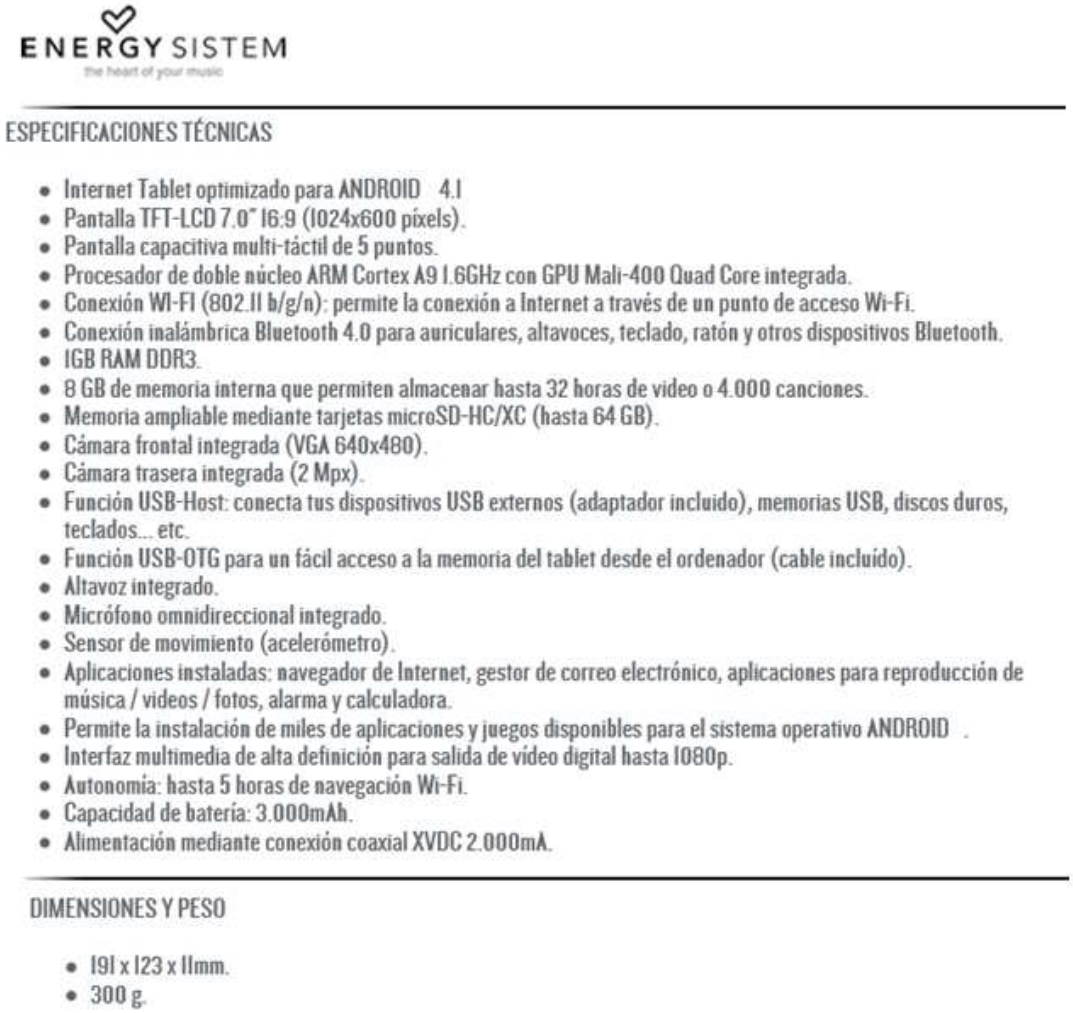
B.1. Comparación de las placas

En la siguiente tabla se pueden observar las especificaciones de las tres placas propuestas para realizar el soporte hardware del proyecto:

Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	$29.95	$35	$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

B.2. Especificaciones de la Tablet

Este apartado muestra una imagen extraída de la web del fabricante de la Tablet en la que se muestran todas las especificaciones técnicas del dispositivo.



ENERGY SISTEM
the heart of your music

ESPECIFICACIONES TÉCNICAS

- Internet Tablet optimizado para ANDROID 4.1
- Pantalla TFT-LCD 7.0" 16:9 (1024x600 píxels).
- Pantalla capacitiva multi-táctil de 5 puntos.
- Procesador de doble núcleo ARM Cortex A9 1.6GHz con GPU Mali-400 Quad Core integrada.
- Conexión WI-FI (802.11 b/g/n): permite la conexión a Internet a través de un punto de acceso Wi-Fi.
- Conexión inalámbrica Bluetooth 4.0 para auriculares, altavoces, teclado, ratón y otros dispositivos Bluetooth.
- 1GB RAM DDR3.
- 8 GB de memoria interna que permiten almacenar hasta 32 horas de video o 4.000 canciones.
- Memoria ampliable mediante tarjetas microSD-HC/XC (hasta 64 GB).
- Cámara frontal integrada (VGA 640x480).
- Cámara trasera integrada (2 Mpx).
- Función USB-Host: conecta tus dispositivos USB externos (adaptador incluido), memorias USB, discos duros, teclados... etc.
- Función USB-OTG para un fácil acceso a la memoria del tablet desde el ordenador (cable incluido).
- Altavoz integrado.
- Micrófono omnidireccional integrado.
- Sensor de movimiento (acelerómetro).
- Aplicaciones instaladas: navegador de Internet, gestor de correo electrónico, aplicaciones para reproducción de música / videos / fotos, alarma y calculadora.
- Permite la instalación de miles de aplicaciones y juegos disponibles para el sistema operativo ANDROID .
- Interfaz multimedia de alta definición para salida de vídeo digital hasta 1080p.
- Autonomía: hasta 5 horas de navegación Wi-Fi.
- Capacidad de batería: 3.000mAh.
- Alimentación mediante conexión coaxial XVDC 2.000mA.

DIMENSIONES Y PESO

- 191 x 123 x 11mm.
- 300 g.

B.3. Especificaciones de la batería

Del mismo modo que el apartado anterior, este apartado ofrece las especificaciones técnicas más importantes de la batería elegida, la RS 727-0385. Parte de las especificaciones se indicaron en el apartado de elección de la batería (5.3.4). No obstante existen otros detalles que caracterizan más completamente el dispositivo, como

el tipo de terminal o el tipo de plomo ácido, así como las aplicaciones típicas que propone el vendedor del producto:

Aplicación Típica	Señales aéreas, sistemas de alarma y seguridad, automoción, equipos de comunicación, fuentes de alimentación dc, aparatos electrónicos, equipo electrónico, equipo de emergencia, iluminación de emergencia, EPS, señales de ferrocarril, UPS
Capacidad	3.2Ah
Construcción	AGM
Dimensiones	134 x 34 x 66mm
Gama de la Marca	RS
Peso	0.67kg
Química	Plomo Ácido
Rango de Temperatura de Funcionamiento	-15 → +50°C
Tensión Nominal	6V
Tipo de Terminal	Tabs
Tipo Plomo Ácido	Ciclo Profundo

Anexo C

Herramientas usadas

Durante el desarrollo del proyecto se han usado una serie de herramientas que han sido fundamentales para lograr los objetivos exitosamente. Estas herramientas, principalmente, han sido tres: el sistema operativo Linux, el analizador Agilent DC Power Analyzer modelo N6705 y el software Matlab (ejecutado en el sistema operativo Windows 7).

C.1. Entorno Linux

Para el desarrollo del software del sistema se ha usado el entorno Linux por diversos motivos, entre que destacan dos.

- Es un sistema operativo completo, totalmente gratuito y muy eficiente.
- Tiene un núcleo modulado, es decir, se pueden añadir y quitar módulos de forma libre haciendo que pueda tener un tamaño considerablemente menor que otros sistemas operativos.

La primera característica tiene como principal ventaja la facilidad para obtenerlo y utilizarlo. La segunda característica, además de ofrecer la posibilidad de reducir notablemente el tamaño del sistema operativo, le da a Linux la propiedad de hacerlo más especializado para una finalidad concreta. Es por esto que es el sistema operativo ideal (y el más utilizado) para implementar dispositivos con funcionalidades muy concretas como routers, servidores, centrales, concentradores…

Otra gran ventaja que ofrece Linux frente a otros sistemas operativos es la facilidad para desarrollar software sin necesidad de permisos o de programas (IDE) destinados a tal fin. Dispone de un compilador para programas escritos en C: el compilador gcc. Dicho compilador se ejecuta desde la terminal de Linux por medio de un comando en el cual se indica el archivo que contiene el código fuente y genera el archivo ejecutable.

El comando gcc es usado para programas escritos en C++ Objective C y FORTRAN además de C. Lleva a cabo la compilación en cuatro pasos:

- Preprocesado: en esta etapa se realizan tareas concernientes al código fuente en sí mismo. Un buen ejemplo es la asignación de los datos *#define*, que son reemplazados por su valor numérico en todos los puntos donde se referencia.
- Compilación: es la traducción del código fuente al lenguaje ensamblador propio del procesador del sistema en el que se pretende ejecutar el programa. El siguiente ejemplo lleva a cabo esta tarea:
  - gcc -S circulo.c: genera un archivo con código ensamblador *circulo.s* a partir del archivo de código fuente circulo.c.
- Ensamblado: convierte el código ensamblador en el código máquina con las instrucciones binarias que irán al procesador (código objeto). No es muy común ejecutar este paso independientemente, sino que suele llevarse a cabo a la par que la compilación como en el siguiente ejemplo:
  - gcc -c circulo.c: se genera el archivo *circulo.o* a partir de circulo.c.
- Enlazado: en este paso se enlazan los archivos del código fuente con las librerías utilizadas (como stdio.h) constituyendo de este modo el programa ejecutable completo. Las funciones pertenecientes a librerías externas no se compilan junto al código fuente porque ya están compiladas y ensambladas en otros códigos objetos. Todos los módulos en código objeto se juntan en esta etapa como en el siguiente ejemplo:
  - gcc -o circulo circulo.o: genera el ejecutable *circulo* a partir del código objeto contenido en el fichero circulo.o.

Todos los pasos anteriores pueden realizarse en un solo paso usando el comando gcc, tal como se muestra a continuación:

gcc -o circulo circulo.c

Cuando se desea realizar un programa extenso es común dividir el código fuente en diferentes archivos. Para automatizar los procesos de compilación y ejecución de estos programas se pueden realizar archivos Makefile, que son ejecutados con el comando *make*. Dicho comando ejecuta el script contenido en el archivo. Para este proyecto se ha realizado el siguiente archivo Makefile:

main:

gcc -o main main.c ./term/USBin/term.c ./term/USBin/functions.c ./term/USBin/queue.c ./term/USBin/data.c ./term/proc/proc.c ./term/proc/functions.c -I/usr/local -L/usr/local -lusb-1.0

sudo ./main

mem:

```
		gcc -Wall -g main.c ./term/USBin/term.c ./term/USBin/functions.c ./term/USBin/queue.c ./term/USBin/data.c ./term/proc/proc.c ./term/proc/functions.c -I/usr/local -L/usr/local -lusb-1.0 -o main

		sudo valgrind --leak-check=full ./main

	clean:

		sudo rm main

		sudo rm ./term/USBin/requests.txt

		sudo rm ./term/data_in.txt
```

Con este archivo, al ejecutar *make*, el sistema compilará y ejecutará el programa. Con el comando *make clean* se eliminarán todos los archivos ejecutados durante el proceso, limpiando los directorios de archivos innecesarios. Finalmente, con el comando *make mem* se ejecutará el programa *valgrind* que generará una serie de indicaciones sobre el uso de algunos recursos del sistema tales como el procesador o la memoria, útiles para conocer si el programa se ha ejecutado de manera solidaria con el sistema o no.

Otra importante herramienta disponible en la distribución Angstrom de Linux es *libusb*, una librería capaz de facilitar en una gran medida el desarrollo del software encargado de interactuar con dispositivos USB. Esta librería ofrece una gran variedad de funciones capaces de realizar cualquier tipo de acción, petición o gestión en lo referente a comunicación a través del puerto USB. En lo referente al presente proyecto, las funciones indicadas anteriormente han sido útiles para iniciar y configurar los buffers necesarios, realizar las transferencias de datos y, por último, liberar la memoria dinámica reservada, cerrando de este modo todos los buffers inicializados.

La distribución Angstrom viene provista de esta herramienta, pero otras distribuciones como Ubuntu (donde se ha desarrollado realmente el software) no disponen inicialmente de esta facilidad. Para obtenerla se debe descargar de Internet tanto la librería como el código fuente e instalarla a través de la consola de comandos:

```
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libusb-1.0-0
```

C.2. Analizador Agilent N6705

El dimensionamiento de la batería ha sido un paso fundamental para conseguir desarrollar el sistema. Para llevar a cabo este paso se ha utilizado este analizador.

Este dispositivo ha sido elegido por su disponibilidad en el laboratorio. No obstante, de entre todos los que estaban disponibles ha resaltado por una serie de funcionalidades que lo han hecho ideal para tomar las medidas sobre los parámetros eléctricos deseadas.

- Capacidad para manejar hasta cuatro salidas.
- Por cada salida, ofrece una fuente de alimentación configurable y un multímetro capaz de medir la tensión y la corriente entre dos puntos.
- Resultados mostrados gráficamente.
- Posibilidad de extracción de datos medidos durante 30 segundos en formato csv (Excel).
- Interfaz amigable.

Con estas funcionalidades, el proceso a seguir (apartado 5.3) ha consistido en tomar una de las interfaces del analizador y usarla para alimentar el dispositivo bajo estudio (sensor o placa). La entrada del multímetro se ha colocado sobre los mismos bornes que la fuente de alimentación. Una vez montado, se ha comenzado a tomar las medidas "extraíbles" y durante los 30 segundos de duración se han ido realizando las funciones básicas de cada uno de los dispositivos bajo estudio.

De este modo se obtuvieron una serie de archivos csv con los datos referentes al consumo de corriente y tensión de los dispositivos en los bornes de alimentación (donde se sitúa la batería del sistema) durante todas las posibles funciones de dichos dispositivos.

Con estas medidas se ha realizado el estudio sobre el consumo de batería que harían los componentes del CS una vez realizado el montaje completo. Dichas medidas deberán ser analizadas con algún tipo de software especial.

C.3. Matlab

El software elegido para realizar el análisis de los datos recogidos por el analizador ha sido Matlab. Consiste en un programa consistente en un IDE capaz de ejecutar una serie de instrucciones propias a un nivel bastante alto.

Sus principales ventajas de este software frente a otros (como Excel) residen en la plena libertad que tiene el usuario para realizar programas que manipulen los datos extraídos, así como en la disponibilidad de una grandísima cantidad de funciones ya

elaboradas capaces de ofrecer funcionalidades como la obtención de datos desde archivos con diversos formatos o la facilidad de representar datos gráficamente de muchas formas (gráficas continuas, discretas, histogramas, diagramas…).

De este modo, se han realizado funciones capaces de obtener los datos recogidos en los archivos csv generador por el analizador para almacenarlos en vectores de datos y analizarlos posteriormente, para cada uno de los archivos obtenidos. El análisis de los datos consistió únicamente en representarlos gráficamente y en realizar los cálculos necesarios para conocer la carga consumida durante los 30 segundos de duración y la corriente media y máxima durante dicho periodo.

Anexo D

Atributos y funciones de los TAD

D.1. Variables y funciones globales

Antes de comenzar con los TAD se introducirán dos funciones fundamentales que son usadas por los TAD sin llegar a ser métodos de éstos:

- int printDatabaseFile (int num): genera la base de datos con la información necesaria para llenar los TAD con los datos que serán usados durante las transferencias en un fichero llamado *requests.txt* (usando las funciones set de los TAD tControl y tInterrupt).
- int sendReceiveData (libusb_device_handle **handle, tQueue database, tQueue *receive): esta es la función encargada de generar todos los envíos (leyendo a través de las funciones get de un TAD Queue los parámetros del envío) al dispositivo USB y de gestionar los recibos desde dicho dispositivo (guardando los datos en otro TAD Queue con las funciones set de éste).
- Variables globales VID_TERM y PID_TERM: son los identificadores del sensor a nivel USB, con valores 0x10C4 y 0xEA80 respectivamente.

D.2. Módulo de entrada de datos

En el módulo de entrada de datos se han usado los siguientes TAD:

- tData: puntero genérico encargado de guardar los datos recibidos o enviados durante el intercambio de información. No tiene atributos ni funciones propias.

- tControl/tControlPacket: TAD encargado de almacenar la información necesaria para llevar a cabo las transferencias de control con el dispositivo USB. Sus atributos son:
    - bmRequestType: entero donde se almacenan los bits correspondientes al campo bmRequestType (ver apartado 2.2.2).
    - bRequest: entero donde se almacenan los bits correspondientes al campo bRequest (ver apartado 2.2.2).
    - wValue: entero donde se almacenan los bits correspondientes al campo wValue (ver apartado 2.2.2).
    - wIndex: entero donde se almacenan los bits correspondientes al campo wIndex (ver apartado 2.2.2).
    - wLength: entero donde se almacenan los bits correspondientes al campo wLength (ver apartado 2.2.2).
    - data: TAD del tipo tData.
    - length_data: entero que indica el número de bytes que contiene el campo data.

  Las funciones propias de este TAD son del tipo set y get:
    - int get_Ctrl_bmRT (tControl ctrl): obtiene el atributo bmRequestType.
    - void set_Ctrl_bmRT (tControl *ctrl, int bmRT): asigna el valor del atributo bmRequestType.
    - int get_Ctrl_bRq (tControl ctrl): obtiene el atributo bRequest.
    - void set_Ctrl_bRq (tControl *ctrl, int bRq): asigna el valor del atributo bRequest.
    - int get_Ctrl_wVal (tControl ctrl): obtiene el atributo wValue.
    - void set_Ctrl_wVal (tControl *ctrl, int wVal): asigna el valor del atributo wValue.
    - int get_Ctrl_wInx (tControl ctrl): obtiene el atributo wIndex.
    - void set_Ctrl_wInx (tControl *ctrl, int wIndx): asigna el valor del atributo wIndex.
    - int get_Ctrl_wLn (tControl ctrl): obtiene el atributo wLength.
    - void set_Ctrl_wLn (tControl *ctrl, int wLn): asigna el valor del atributo wLength.
    - unsigned char* get_Ctrl_data (tControl ctrl): obtiene el atributo data.
    - int set_Ctrl_data (tControl *ctrl, int dat[], int length): asigna el valor del atributo data.
    - int get_Ctrl_lnDt (tControl ctrl): obtiene el atributo length_data.
    - void set_Ctrl_lnDt (tControl *ctrl, int lnDt): asigna el valor del atributo length_data.
    - tControl create_Ctrl_Packet (int bmRT, int bR, int wV, int wI, int wL, int data[], int ld): genera un puntero a un TAD tControlPacket.
    - void delete_Ctrl_Packet (tControl ctrl): Vacía y elimina un TAD tControlPacket.

- tInterrupt/tInterruptPacket: TAD encargado de almacenar la información necesaria para llevar a cabo las transferencias de interrupción con el dispositivo USB. Sus **atributos** son:
    - endpoint: entero encargado de indicar el endpoint al que van dirigidos los envíos o desde el cuál serán enviados las transferencias a recibir.
    - data: TAD del tipo tData.
    - length_data: entero que indica el número de bytes que serán enviados o recibidos.

  Las **funciones** propias de este TAD son del tipo set y get:
    - int get_Int_ep (tInterrupt inter): obtiene el atributo endpoint.
    - void set_Int_ep (tInterrupt *inter, int ep): asigna el valor del atributo enpoint.
    - unsigned char* get_Int_data (tInterrupt inter): obtiene el atributo data.
    - int set_Int_data (tInterrupt *inter, int dat[], int length): asigna el valor del atributo data.
    - int get_Int_lnDt (tInterrupt inter): obtiene el atributo length_data.
    - void set_Int_lnDt (tInterrupt *inter, int lnDt): asigna el valor del atributo length_data.
    - tInterrupt create_Int_Packet (int ep, int data[], int ld): genera un puntero a un TAD tInterruptPacket.
    - void delete_Int_Packet (tInterrupt inter): vacía y elimina un TAD tInterruptPacket.
- tReceive/tReceivePacket: TAD encargado de almacenar los datos recibidos para escribirlos posteriormente en el archivo *data_in.txt*. Sus **atributos** son:
    - data: TAD del tipo tData.
    - length: número de bytes del campo data.

  Las **funciones** propias de este TAD son únicamente del tipo set, ya que no sus atributos no serán leídos directamente (se leerán a través del archivo).

    - void set_Rec_ln (tReceive *rec, int lnDt): asigna el valor del atributo length.
    - int set_Rec_Data(tReceive *rec,unsigned char *data,int length): asigna el valor del atributo data.
    - tReceive create_receive_Packet(unsigned char *data, int length): genera un puntero a un TAD tReceivePacket.
    - void delete_Rec_Packet (tReceive rec): vacía y elimina un TAD tReceivePacket.
- tInfoPacket: puntero genérico al que se le asignará o bien la dirección de un TAD tControlPacket o de un TAD tInterruptPacket, según se corresponda con un nodo de envío de control o de interrupción. No tiene atributos ni funciones.
- tNode/Node: TAD que representa un nodo del TAD cola (queue en inglés). Cada nodo se corresponderá con un envío/recibo y serán usados por orden, motivo por el cual se han implementado sobre una cola. Sus atributos son:

  - transfer_type: entero que indica si la transferencia es de control (1) o de interrupción (3).
  - Packet: puntero a tInfoPacket.
  - Next: puntero que apunta al siguiente nodo.
- tQueue/tNodeQueue: TAD cola. Consiste en un nodo especial con dos **atributos** únicamente:
  - start: puntero que apunta al primer nodo de la cola.
  - end: puntero que apunta al último nodo de la cola.

  Las **funciones** de este último conjunto de TAD son las siguientes:

  - void queue_init (tQueue *C): inicializa los campos del TAD cola.
  - int isEmpty (tQueue C): comprueba si el TAD cola está vacío.
  - int addNode (int type, tInfoPacket packet, tQueue *C): agrega un nuevo nodo al TAD. Para ello crea un nodo nuevo, le asigna los atributos correspondientes y lo enlaza a la cola como el último elemento. Este último paso se lleva a cabo asignando al atributo next del último nodo de la cola el nuevo nodo, y apuntando al mismo como nuevo último elemento.
  - int deleteNode (tQueue *C): elimina el primer nodo del TAD cola y asigna su contenido a un TAD tReceive. Para ello usa un puntero independiente para apuntar el nodo objetivo, reorganiza la cola haciendo que el primer elemento sea ahora el que apuntaba el nodo objetivo como next, extrae la información y elimina el nodo.
  - int readDatabaseFile (tQueue *C, int num): lee la base de datos y almacena la información distribuida en nodos. El argumento num indica el número de medidas que han sido solicitadas.
  - int printQueueFile (tQueue C): escribe los datos de los nodos de la cola en un archivo de texto (*data_in.txt*).
  - void printQueue (tQueue C): imprime todos los datos de los nodos de la cola por pantalla. Ha sido usada para depurar el software.
  - void deleteQueue (tQueue *C): Elimina el TAD cola.
  - const char* queue_error_name (int error): traduce los errores propagados a lo largo de la ejecución del software en mensajes legibles. La traducción es la siguiente:
    - case 0: "SUCCESS";
    - case -1: "TAD_ERROR_NODE_CREATION";
    - case -2: "TAD_ERROR_EMPTY_QUEUE";
    - case -3: "TAD_ERROR_DATA_FILE";
    - default: "OTHER";

D.3. Módulo de procesado de datos

En este módulo se definen los TAD referentes al almacenamiento y procesado de los datos:

- tDataBlock: puntero genérico donde se almacenarán los datos a procesar.
- tBlock/tBlockPacket: TAD encargado de almacenar los datos leídos desde el fichero *data_in.txt*, es decir, los datos enviados por el sensor. Sus **atributos** son:
    - num: entero que almacena el número de dato a procesar. Su utilidad reside en la ordenación de los datos a la hora de procesarlos.
    - dataBlock: puntero genérico del tipo tDataBlock en el que se almacenan los datos leídos en el fichero.

  Sus **funciones** se definen a continuación:

    - void set_Block_num (tBlock *B, int num): asigna el valor del atributo num.
    - int get_Block_num (tBlock B): obtiene el atributo num.
    - int set_Block_data (tBlock *B, int dat[]): asigna el valor del atributo dataBlock.
    - int *get_Block_data (tBlock B): obtiene el atributo dataBlock.
    - tBlock create_Block_Packet (int data[],int num): genera un puntero a un TAD tBlockPacket y ejecuta las funciones set_Block_num y set_Block_data para rellenar sus campos.
    - void deleteBlock (tBlock *B, int num): vacía y elimina una TAD tBlockPacket.
    - int clasifiedData(tBlock *B): es la función que lee cada línea del fichero *data_in.txt* (cada línea se corresponde con una trama de envío del sensor) y genera un TAD tBlock/tBlockPacket por cada una, rellenando sus campos consecutivamente. De este modo, cuando finaliza esta función, los datos recibidos desde el sensor quedan clasificados y guardados en buffers de memoria dinámica.
- tDHT/tDateHourTemp: TAD encargado de almacenar los datos relevantes recibidos desde el sensor y almacenados en el TAD tBlock/tBlockData. Dichos datos relevantes son la temperatura tomada (en una medida concreta), la fecha y la hora a la que se realizó dicha medida. Sus **atributos** son:
    - minute: entero que guarda el minuto en el que se tomó la medida.
    - hour: entero que guarda la hora en la que se tomó la medida.
    - day: entero que guarda el día en el que se tomó la medida.
    - month: entero que guarda el mes en el que se tomó la medida.
    - year: entero que guarda el año en el que se tomó la medida.
    - temp: entero que guarda la temperatura medida.

Sus funciones son las siguientes:

- int set_Date (int date[], tDHT *DHT): asigna los atributos día, mes y año.
- void set_Hour (int hour[], tDHT *DHT): asigna los atributos minuto y hora.
- int set_Temp (int temp[], tDHT *DHT): asigna el atributo temp.
- tDHT create_DHT_Packet (int date[], int hour[], int temp[]): genera un puntero a un TAD tDateHourTemp y rellena sus campos con las tres funciones anteriores.
- void delete_DHT (tDHT *DHT, int num): vacía y elimina un TAD tDateHourTemp.
- tDHT getData(tBlock B1, tBlock B2): es la función que obtiene los datos relevantes de los TAD tBlock/tBlockData. Consulta dichos TAD y extrae la hora, fecha y temperatura para luego llamar a la función créate_DHT_Packet enviándole estos datos como argumentos. Esta función (getData) tiene dos argumentos de tipo tBlock porque el sensor envía dos tramas "relevantes": la primera con la fecha y hora y la segunda con la temperatura.

Por otro lado, existen otras funciones que también son muy relevantes para el correcto funcionamiento del módulo de procesado de datos. Dichas funciones son las siguientes:

- char *invert(char *cad, int length): reinvierte el orden de los datos de una cadena.
- char *DecToBin(int num, int *length): convierte un dato decimal a binario.
- int BinToDec(char data[], int length): convierte un dato binario a decimal.
- char *DecToHex(int num): convierte un dato decimal a hexadecimal.
- int HexToDec(char data[]): convierte un dato hexadecimal a decimal.

Para la salida de datos:

- void printBlock(tBlock B): imprime por pantalla los datos del TAD tBlock. Usado para depuración.
- void printResults(tDHT *data, int length): imprime por pantalla los resultados finales del software del programa. La presentación es similar a la siguiente:

Fecha	Hora	Temperatura
dd/mm/yyyy	HH:MM	tt,t
dd/mm/yyyy	HH:MM	tt,t

dd: día (con una o dos cifras).
mm: mes (con una o dos cifras).
yyyy: año (con cuatro cifras).
HH: hora (con una o dos cifras).
MM: minuto (con una o dos cifras).
tt,t: temperatura (dos cifras y un decimal).

Anexo E

Propagación de errores

Dentro del desarrollo software, de cara a ofrecer un buen servicio de soporte, es imprescindible dotar al dispositivo de un buen sistema de gestión de errores capaz de detectarlos y propagaros eficazmente. En este anexo se presentan los posibles errores que pueden darse y la forma de propagarlos.

Durante la ejecución del software, pueden darse dos tipos de errores, que pueden manifestarse de diferentes maneras. Estos dos tipos de errores son:

- Errores en la creación de buffers de memoria: se deben a la imposibilidad de realizar reservas dinámicas de memoria (generalmente por la falta de espacio). Dichas reservas se realizan a través de la función *malloc* ubicada en la librería *sdtlib.h* de C, la cual devuelve un puntero a la primera posición de memoria reservada en el caso de ejecutarse exitosamente o un puntero con valor NULL en caso de error. Este error puede manifestarse en las siguientes situaciones:
    - Apertura de archivos: los archivos usados por el sistema se abren con la función *fopen* ubicada en la librería *stdio.h* de C, la cual usa la función *malloc* para crear un buffer en el que almacenar los datos necesarios para poder abrir y editar el archivo en cuestión. Del mismo modo que *malloc*, la función *fopen* devuelve un puntero al "archivo" (representado a través de la estructura *FILE*) en caso de éxito o NULL en caso de error. Este error se puede presentar en dos ocasiones en este sistema: al abrir el archivo *requests.txt* o al abrir el archivo *data_in.txt*. Las funciones encargadas de realizar estas aperturas recibirán el valor NULL en caso de error, y propagarán el error hacia las funciones superiores con el valor -3.
    - Creación de nodos: para generar los TAD es necesario ir realizando reservas dinámicas de memoria para la creación de cada nodo. Dicha reserva se realiza con la función *malloc*, lo cual puede dar lugar a errores como ya se ha indicado. Las funciones encargadas de realizar estas reservas propagarán el error con el valor -1.

- Errores al liberar un puntero de memoria inexistente: la liberación de la memoria dinámica se lleva a cabo con la función *free* ubicada también en la librería *stdlib.h* de C. Como cabe esperar, sólo podrán liberarse espacios de memoria que hayan sido reservados con anterioridad por la función *malloc*, por tanto, si el puntero que se le indique a la función *free* no se corresponde con uno generado por *malloc* la función devolverá un error, del mismo modo que si se intenta liberar un espacio de memoria que haya sido liberada ya. Dentro del sistema, este tipo de error se genera cuando se intenta liberar un nodo de un TAD que ya haya sido liberado y las funciones encargadas de detectarlos los propagan con el valor -2.

Otros errores que pueden darse son los producidos durante la comunicación con los dispositivos USB, que son generados por las funciones de la librería *libusb.h*, cuyos detalles no se indicarán por quedar fuera del ámbito del proyecto. Pueden consultarse en la web de su API indicada en la bilbiografía.

En el caso de que estas funciones susceptibles de error no detecten niguna anomalía devolverán el valor 0, indicando que todas las tareas se han realizado exitosamente.

Anexo F

Manual de usuario

En este anexo se indicarán cuáles son los pasos que han de seguirse para poder usar el dispositivo correctamente.

Como cabe esperar, lo primero es disponer de la batería del sistema cargada para comenzar a utilizar el dispositivo. Un mal uso de ésta (interrumpir la carga antes de que se cargue totalmente o conectar a la red de alimentación antes de que se haya descargado completamente) provocará que la carga de la batería disminuya, reduciendo de este modo la autonomía del dispositivo.

Para que el CS comience a funcionar se pulsará el interruptor general de éste. Una vez pulsado el sistema tardará tres minutos aproximadamente en estar preparado.

El dispositivo mostrará un panel de acceso de usuario. En su configuración por defecto, bastará con pulsar la opción *Login* para acceder, sin necesidad de introducir contraseñas.

Cuando esté listo, bastará con abrir una terminal pulsando sobre la pantalla táctil. El acceso directo de la terminal se ubica en el escritorio, con el nombre Terminal. Una vez abierta la consola (pantalla negra) habrá que teclear el comando *make* y ya estará el dispositivo listo para un uso continuado.

Mientras el sistema esté en funcionamiento, el programa pedirá por pantalla el número de medidas que el usuario desea extraer del sensor. Durante este proceso se recomienda no desconectar ningún sensor puesto que el CS no recibirá ningún dato hasta que el usuario indique dicho número de medidas y, por tanto, la desconexión del sensor que corresponda conllevará la pérdida de los datos que no hayan sido almacenados.

De este modo, el correcto uso del dispositivo consistirá en (una vez esté encendido) tomar las medidas con los sensores y de forma inmediata extraer dichas medidas con el CS. Este proceso deberá llevarse a cabo con cada paciente.

Una vez se hayan tomado todas las medidas deseadas, el sistema deberá apagarse para ahorrar energía. Para apagarlo se deberá acceder al escritorio y una vez ahí seleccionar el botón de apagado situado en la barra superior de la pantalla en el extremo derecho. Una vez se seleccione la opción de apagado del sistema, éste cerrará todos los procesos y se quedará en un modo de suspensión en el que se mostrará en una pantalla con fondo negro el símbolo del sistema operativo (angstrom). En este punto se podrá desconectar el sistema para apagarlo definitivamente.

Para recargar la batería del sistema, será muy recomendable apagar el sistema antes de comenzar la carga.

Anexo G

Glosario

CS: Concentrador de Señales.

USB: Universal Serial Bus.

USB OTG: USB On-the-go

HID: Human Interface Device

PC: Personal Computer

CPU: Central Processing Unit

ISM (bandas): Industrial, Scientific and Medical

ATA: AdvancedTechnology Attachment

IDE: Integrated Device Electronics

SCSI: Small Computer System Interface

LED: Light-Emitting Diode

ECG: ElectroCardioGraphy

DSP: Digital Signal Processor

ALU: Arithmetic Logic Unit

ARM: Advanced RISC Machine

RISC: Reduced Instruction Set Computer

PDA: Personal Digital Assistant

GPS: Global Positioning System

OS/SO: Operative System / Sistema Operativo

UCI: Unidad de Cuidados Intensivos

ISO: International Organization for Standardization

IEC: International Electrotechnical Comission

IEEE: Institute of Electrical and Electronic Engineers

ITU: International Telecommunication Union

ETSI: European Telecommunications Standards Insitute

CENELEC: Comité Européen de Normalisation Electrotechnique

ONU: Organización de Naciones Unidas

AENOR: Asociación Española de Normalización y Certificación

QoS: Quality of Service

OSI: Open System Interconnection

PCB: Printed Circuit Board

IDE: Integrated Development Environment

TAD: Tipo Abstracto de Dato

VID: Vendor ID

PID: Product ID

Anexo H

Presupuesto

- Ejecución Material
  - Compra de ordenador personal (Office, Matlab incluidos). 2.000€
  - Beaglebone, cape lcd4. 118€
  - Sensores. 450€
  - Componentes electrónicos (batería incluida). 20€
  - Material de oficina. 50€
  - Total de ejecución material. 2.638€
- Normativa
  - Part 00101: Guidelines for the use of Wireless technology. 118€
  - Part 20101: Application Profiles - Base Standard. 69€
  - Part 30200: Transport Profile - Cable Connected. 69€
  - Part 30400: Transport Profile - Cabled Ethernet. 61€
  - Otros. 700€
  - Total normativa. 1.017€
- Honorarios proyecto
  - 1400 horas a 22 €/horas. 30.800€
- Material fungible
  - Gastos de impresión. 50€
  - Encuadernación. 50€
- Subtotal del presupuesto
  - Subtotal Presupuesto. 34.555€
- I.V.A. aplicable
  - 21% Subtotal Presupuesto. 7.256,55€
- Total presupuesto
  - Total Presupuesto. 41.811,55€

Madrid, Enero 2015

El Ingeniero Jefe de Proyecto

Fdo.: Carlos Andradas Martínez

Ingeniero Superior de Telecomunicación

Anexo I

Pliego de condiciones

En este apartado se incluyen todas las pautas referentes al desarrollo del proyecto en términos legales. Para ello, se supondrá el caso hipotético de que una empresa comercial realice el encargo a otra empresa (consultora, por ejemplo) que se encargará de desarrollar el dispositivo. Dicha empresa habrá tenido que desarrollar una línea de investigación para llevar el proyecto a cabo, comprobar su viabilidad y su posible utilidad y competencia en el mercado, por lo que deberá establecer unos términos legales con el objetivo de quedar amparada. Por otro lado, también debe considerarse que será la empresa comercial la que indique las funcionalidades que se desean obtener del dispositivo.

En este pliego se presentan las condiciones legales que se aplicarán al desarrollo de este proyecto y deberá ser entregado por la empresa desarrolladora (consultora) a la empresa comercial bajo las condiciones generales ya mencionadas así como una serie de condiciones particulares que se detallan a continuación:

1. La empresa que desarrolla el producto (consultora) deberá entregar junto al primer prototipo una documentación en la que queden bien indicados los casos de uso y el funcionamiento del producto en cada uno de ellos, resaltando los límites en los cuales el rendimiento del dispositivo no sea óptimo o en los que pueda acarrear un riesgo para la salud de los pacientes así como del personal que lo manipule o utilice. En esta documentación también se indicará el cumplimiento de la normativa.
2. El primer paso será que la empresa comercial verifique que se cumplen todas las expectativas respecto a las funcionalidades deseadas del dispositivo.
3. Para corroborar que todo está en orden (prototipo y manual de uso) la empresa que comercializará el producto deberá probar el dispositivo, tanto su correcto funcionamiento como sus límites, de manera que se pueda realizar el traspaso de responsabilidades de la empresa desarrolladora a la comercial.
4. Esta comprobación deberá realizarse en un plazo de dos semanas.

5. La empresa comercial a su vez, debe elaborar un set de instrucciones en el que aparezcan todas las indicaciones aportadas por la empresa desarrolladora así como funcionalidades o indicaciones adicionales sobre posibles casos en los que la empresa no se haga responsable frente a usos incorrectos o indebidos del dispositivo, resaltando los casos en los que cualquier fallo que pueda tener lugar no generen una negligencia de la que se tengan que hacer responsables.
6. Una vez realizada la comprobación, la empresa comercial pasará a adquirir toda la responsabilidad respecto a la distribución del producto y de las instrucciones de uso, de manera que el cliente final esté en posesión de todo lo necesario para poder usar el dispositivo bajo su propia responsabilidad en el momento de la venta.
7. En la documentación entregada al usuario final deberá indicarse un periodo de garantía de funcionamiento del dispositivo en el cual, en caso de errores, deberá ofrecerse soporte gratuito o incluso la posibilidad de un dispositivo de sustitución.
8. En caso de errores debidos al mal uso, la responsabilidad recaerá sobre el usuario final del producto, siempre y cuando la empresa vendedora del producto haya estipulado correctamente las condiciones en las que debe ser usado el producto así como sus posibles aplicaciones con sus métodos de uso.
9. En el momento de la venta, la documentación aportada por la empresa vendedora será lo que limite los aspectos legales, es decir, cualquier anomalía registrada en el funcionamiento del dispositivo que no esté contemplado en éstas pasará a ser responsabilidad de la empresa vendedora, mientras que en caso contrario, las negligencias cometidas serán responsabilidad del usuario final.
10. En el caso de que las anomalías no estén reflejadas en la documentación del dispositivo, en caso de que se produzcan accidentes que afecten a la salud del paciente o del personal, la empresa vendedora será la responsable de los hechos. Si el dispositivo deja de funcionar o simplemente no cumple de manera total o parcial con sus funciones de manera temporal o definitiva, será la empresa vendedora la encargada de ofrecer el soporte necesario o, incluso en el peor de los casos, de ofrecer un dispositivo que reemplace al inicial.
11. La empresa comercial del producto será la responsable de hacer llegar el manual de uso y las limitaciones al cliente que finalmente compre el producto, de manera que éste pueda comenzar a usarlo de manera inmediata.
12. El montaje del sistema deberá ser llevado a cabo por la empresa que comercialice el dispositivo, de manera que puedan comprobar que los sensores de los que disponen sean compatibles con éste.

Bibliografía

Internet. Documentación para BeagleBone

[1] http://www.lvr.com/eclipse1.htm
[2] http://www.lvr.com/beaglebone.htm
[3] http://www.lvr.com/beagleboard.htm

[4] http://beagleboard.org/bone

[5] http://elinux.org/BeagleBoardUbuntu
[6] http://elinux.org/BeagleBone
[7] http://elinux.org/Beagleboard:C/C%2B%2B_Programming
[8] http://elinux.org/Beagleboard:BeagleBone
[9] http://elinux.org/ARMCompilers

Internet. Documentación sobre desarrollo USB

[10] http://devel.no-ip.org/hardware/USB0000/index.html
[11] http://es.scribd.com/doc/15636319/Creacion-Drivers-Usb
[12] http://libusb.sourceforge.net/api-1.0/
[13] http://www.dreamincode.net/forums/topic/148707-introduction-to-using-libusb-10/
[14] http://www.usbmadesimple.co.uk/
[15] http://www.beyondlogic.org/usbnutshell/

Guía Agilent

[16] Agilent Technologies DC Power Analyzer. Model N6705. User's Guide

Libros

[17] Linux Device Drivers. Jonathan Corbet
[18] USB Embedded Hosts. Jan Axelson
[19] Biomedical Instrumentation, Technology and applications. R.S. Khandpur
[20] Building Embedded Linux Systems. Karim Yaghmour.

Estado del arte

[21] www.wikipedia.org
[22] http://www.windriver.com/products/vxworks/
[23] http://www.microsoft.com/windowsembedded/en-us/windows-embedded-compact-2013.aspx
[24] http://www.qnx.com/

Normativa

[25] http://shop.ieee.org/ieeestore
[26] http://www.cen.eu
[27] http://www.iso.org/iso/search.htm)qt=11073&searchSubmit=Search&short=rel&type=simple&published=true

11073 - Health informatics – PoC mediacl device communication:

[28] Part 00101: Guide – Guidelines for the use of Wireless technology
[29] Part 10101: Point of Care – Nomenclature
[30] Part 10201: Point of Care - Domain Information Model
[31] Part 20101: Application Profiles - Base Standard
[32] Part 20601: Application Profile - Optimazed Exchange Protocol
[33] Part 30200: Transport Profile - Cable Connected
[34] Part 30300: Transport Profile - Infrared Wireless
[35] Part 30400: Transport Profile - Cabled Ethernet
[36] Goga, NIcolae and Costache, Stefania, "A formal analysis of ISO/IEEE P11073-20601 standard of medical device communication", IEEE SysCon 2009 —3rd Annual IEEE International Systems Conference, 2009, Vancouver, Canada, March 23–26, 2009.
[37] Giannetti, Romano. "Apuntes informales de Electrónica Biomédica. Aparatos biomédicos y seguridad".
[38] Tesche, Günter. "European Standardization in Medical Informatics and its Relevance to Medical Image Communication". Philips Medizin Systeme GmbH. Röntgenstr. 24, D-2000 Hamburg 63, Germany.
[39] Gumudavelli, Suman; McKneely, Paul; Chapman, Frank. "Medical Instrument Data Exchange". 30th Annual International IEEE EMBS Conference. Vancouver, British Columbia, Canada, August 20-24, 2008.

[40] http://www.coit.es/foro/pub/ficheros/libroscapitulo_5_2d1661b7.pdf