Proyecto Fin de Carrera

Desarrollo de un sistema de recogida automática de datos aplicado a equipos autónomos

Autor

Adrián Peña Lapuente

Director y ponente

David Izquierdo Núñez
Carlos David Heras Vila

Escuela de Ingeniería y Arquitectura

Septiembre 2013

Desarrollo de un sistema de recogida automática de datos aplicado a equipos autónomos

RESUMEN

La gran evolución sufrida por las Tecnologías de la Información y de las Comunicaciones y por Internet, han propiciado su uso para el control remoto de todo tipo de instalaciones.

En este ámbito, el Grupo de Tecnologías Fotónicas de la Universidad de Zaragoza, que desde hace unos años diseña sensores para plantas termosolares, quiere obtener datos y controlar remotamente los equipos que instala, y es ahí donde se enmarca este proyecto.

Este proyecto consiste en el desarrollo de un sistema, capaz de transmitir los datos tomados por unos sensores situados en campo, apoyado en la tecnología GPRS de comunicaciones inalámbricas comerciales, hasta un servidor, capaz de almacenarlos y procesarlos.

Para ello, se ha instalado y configurado un servidor que aloja una web de recepción de datos, una web de gestión y control de los datos recibidos, y una web de configuración de los equipos remotos.

También se ha desarrollado el hardware y el software del dispositivo de medida, que se encarga de establecer la comunicación con el servidor. Este hardware se integra en una de las placas de control ya desarrolladas por el GTF.

Además, se han estudiado tanto el tráfico de datos que generarán estos equipos, como el consumo energético que este sistema añadirá al consumo que ya tenía la placa de control existente, además de caracterizar el sistema de carga de baterías a instalar.

Agradecimientos

A mi familia, por darme la posibilidad de estudiar esta carrera y por aguantarme en los periodos de exámenes.

A mis amigos, por el apoyo que he recibido de su parte, y en especial a Juan, porque sin él habría costado más estudiar.

Al GTF y en especial a David y Carlos, director y ponente de este proyecto, por darme la posibilidad de hacer este proyecto, y por la ayuda prestada en todo momento.

ÍNDICE

PARTE I

MEMORIA

Capítulo 1:

Introducción

Las Tecnologías de la Información y de las Comunicaciones (TIC) han sufrido un enorme avance en las últimas dos décadas gracias al fenómeno de Internet, que permite la interconexión de personas distantes, de forma que se pueden mantener informadas de los acontecimientos ocurridos en cualquier parte del globo. Esto ha dado como resultado un incremento exponencial en su número de usuarios, los que a su vez han demandado constantemente mayores prestaciones y aplicaciones, haciendo crecer el mercado que Internet abarca. Dichos avances han propiciado su inclusión también en el entorno profesional, permitiendo interconectar no solo ordenadores, generando así redes privadas de acceso a bases de datos, como por ejemplo en el sector bancario, sino cualquier pequeño dispositivo que necesite enviar datos a cualquier tipo de servidor, u obtenerlos de estos, dando lugar a lo que se conoce como sistemas *Machine-to-Machine* (*M2M*).

Dentro de los sistemas *M2M*, se debe hacer una mención especial a los equipos de medición, o sensores, ya que son empleados en una gran cantidad y diversidad de aplicaciones, como por ejemplo en el sector de la automoción, la meteorología, la geología… Estos equipos, tienen la particularidad además, de que pueden tenerse que instalar en localizaciones muy diversas, y en muchos casos esas ubicaciones no disponen de la red de acceso necesaria para que los sensores puedan enviar los datos tomados, ni son lo suficientemente accesibles como para que alguien se ocupe, cada cierto tiempo, de la recogida manual de las medidas tomadas. En estos casos se puede hacer uso de las tecnologías de comunicación inalámbricas, debido a que el gran avance experimentado por éstas en el último lustro, ha permitido a las compañías telefónicas garantizar coberturas casi totales en territorio español.

Por otra parte, la utilización de los sensores inalámbricos para llevar a cabo un control exhaustivo de grandes instalaciones, da lugar a extensas redes sensoriales inalámbricas conocidas como WSN (Wireless Sensor Network). Para el establecimiento de estas redes, se hace uso de tecnologías inalámbricas locales, que interconectan los dispositivos y permiten un control ordenado de todos ellos. A su vez, a través de estas conexiones, se puede hacer llegar las medidas que toman a un nodo colector, que se encarga de enviarlos a la central de procesado y almacenamiento.

El Grupo de Tecnologías Fotónicas (GTF) de la Universidad de Zaragoza, desde hace unos años, viene desarrollando diferentes sensores para su implantación en plantas termosolares. Es en este escenario donde se enmarca este proyecto, ya que el GTF requiere del desarrollo de una solución para poder controlar y obtener los datos leídos por dichos sensores de forma remota, como método de automatización completa de las medidas de control de las plantas termosolares. Hasta el

momento, para la lectura de los datos producidos por los equipos de medida, ha sido necesario que un operario se desplace al sensor y los extraiga del dispositivo. Debido a la falta de red de acceso en el interior de las plantas, se requiere controlar y obtener los datos obtenidos por dichos sensores de forma remota, sin necesidad de disponer de puntos fijos de acceso a Internet.

Dadas las características de los datos que deben ser enviados, la solución elegida se basa en el envío de los mismos a través de internet, directamente al servidor en el que se vayan a procesar y almacenar.

De este modo, el objetivo principal de este proyecto es el desarrollo de un sistema de comunicaciones inalámbrico, apoyado en el acceso a las redes de datos proporcionado por las tecnologías móviles comerciales, que permita, tanto la obtención de los datos tomados por los sensores remotos, como la configuración de los propios equipos. Además, estos datos y configuraciones tienen que poderse consultar rápidamente desde cualquier ubicación, y almacenarse de forma segura y privada en un servidor, que estará alejado de los sensores. También será un requisito indispensable el poder incluir con cierta facilidad este sistema en los equipos de medida ya diseñados por este grupo.

Dados los objetivos y las características de los equipos de campo, el sistema de recogida automática de datos a desarrollar debe cubrir los siguientes requisitos:

- Para el dispositivo autónomo: Que sea capaz de resolver los errores que puedan aparecer durante su funcionamiento y que garantice la autonomía energética, por lo que habrá que minimizar todo lo posible el consumo de este sistema.
- Para la comunicación: Que la comunicación se lleve a cabo sin errores, por lo que resulta conveniente el uso de protocolos que aseguren una comunicación extremo a extremo limpia, segura y fiable, garantizando así la integridad de las medidas. Por otra parte, no requerirá de grandes cantidades de recursos de red, ni de elevadas tasas de transmisión, debido a que el sistema de recogida de datos no cursará tráfico de forma continuada, ni serán grandes cantidades de datos cuando transmita.
- Para el servidor: Que sea capaz de recoger las medidas transmitidas por los dispositivos autónomos, tanto individuales como en archivos con las mediciones de todo un día, además de permitir la reconfiguración remota de los equipos situados en campo, para lo que requerirá de cierta flexibilidad y versatilidad.

Para poder alcanzar estos objetivos, el proyecto contempla dos fases bien diferenciadas: Estudio de alternativas tecnológicas y desarrollo del sistema de recogida automática de datos aplicado a equipos autónomos.

La primera, que se describe en el capítulo 2, consiste en el análisis y elección tanto de las alternativas tecnológicas para establecer la comunicación como del servidor.

La segunda, descrita en el capítulo 3, es el desarrollo e implementación del dispositivo autónomo de la comunicación y del servidor y la integración en el sistema de recogida automática de datos. Para ello, habrá que instalar un servidor que alcance el nivel de seguridad y privacidad que los datos requieren. También habrá que modificar el hardware de una de las placas de control de sensores creada por el GTF, integrando el dispositivo de comunicación y los componentes necesarios

para su control y gestión. Además, habrá que incluir en el software del microcontrolador de la placa las funciones necesarias para el establecimiento y la gestión de la comunicación con el servidor, que deberán controlar los errores que puedan aparecer, de forma que no se bloquee el equipo en ninguno de ellos.

Finalmente, en la evaluación del funcionamiento del sistema desarrollado, expuesta también en el capítulo 3, se realizarán las estimaciones de consumo energético y de tráfico de datos que permitan partir de unos valores aproximados a la hora de tomar decisiones en cuanto a los recursos de red a contratar y a las baterías y paneles solares a utilizar, que se caracterizarán en el mismo capítulo.

Capítulo 2:

Análisis de tecnologías, protocolos y plataformas

En este capítulo se realiza una comparación de las diferentes alternativas existentes en cuanto a tecnologías móviles comerciales que se puedan emplear para realizar la comunicación, así como otras tecnologías inalámbricas que permitan el establecimiento de redes sensoriales. También se estudian dos protocolos de nivel de transporte y dos de nivel de aplicación, que pueden utilizarse en el sistema de comunicación. Un análisis más extenso de cada tecnología se puede encontrar en el anexo 1, a partir del cual se obtienen los resultados expuestos en este capítulo.

	Alternativas	
Nivel de aplicación	HTTP	FTP
Nivel de transporte	TCP	UDP
Nivel de red	IP	
Nivel de enlace	GPRS	UMTS
Nivel físico		

Tabla 1: Pila de tecnologías susceptibles de ser empleadas en el sistema a desarrollar.

En la tabla 1 se muestran las diferentes alternativas que se van a analizar, y los niveles de la pila de protocolos que abarca cada una. El protocolo de nivel de red (IP) es fijo para las comunicaciones de datos a través de compañías móviles comerciales.

Además, se analizan las ventajas de instalar un servidor propio frente a la utilización de los servidores de datos disponibles en el mercado.

2.1. Tecnologías inalámbricas móviles comerciales

Inicialmente, se analizan las tres tecnologías existentes en el mercado de la telefonía móvil para la realización de comunicaciones inalámbricas: GSM (Global System for Mobile communicactions) [1], GPRS (General Packet Radio Service) [2] y UMTS (Universal Mobile Telecommunicacions System) [3]. Este tipo de tecnología es recomendable para su uso en sensores aislados, y para su uso en nodos colectores de datos de extensas redes de sensores.

Para introducir este análisis, se presenta la tabla 2 como comparativa inicial. En ella, se observa que la tecnología GSM se ha de descartar desde un principio, ya que no posee la capacidad de cursar tráfico de datos. Por otra parte, se observa una gran diferencia entre GPRS y UMTS en cuanto a la velocidad de transmisión, aunque no es un factor decisivo en la elección a realizar. Estas dos tecnologías se diferencian también en el método de acceso al medio, que otorga a UMTS una

menor probabilidad de bloqueo teórica. Si comparamos las coberturas, UMTS tiene una cobertura ligeramente menor que GPRS, aunque la diferencia entre las dos tecnologías no es demasiado significativa con respecto a este factor.

	GSM (2G)	GPRS (2,5G)	UMTS (3G)
Velocidad	9,6kbps	144kbps	2Mbps
Tráfico de datos	No	Si	Si
Acceso	TDMA/FDMA	TDMA/FDMA	CDMA
Cobertura	Muy Alta	Muy Alta	Alta
Consumo energético	Bajo	Medio	Alto

Tabla 2: Comparación de características de las principales tecnologías móviles comerciales.

De esta forma, el principal factor que diferencia estas dos tecnologías será el consumo energético. Así, destacar que el control de potencia en GPRS lo realiza el dispositivo móvil, adaptando la potencia transmitida según el nivel y la calidad de la señal recibida por un canal común. Por el contrario, en UMTS se hace uso de un control de potencia en el que tanto el dispositivo móvil como la estación base toman medidas, de forma que la estación base puede hacer que todos los equipos de usuario conectados a ella reduzcan su nivel de potencia, para poder ofrecer mayor calidad. Además, en UMTS, para conseguir mayores tasas de transmisión se reduce la codificación, por lo que se requiere del envío de mayores potencias, haciendo que UMTS posea mayor consumo energético que GPRS.

Finalmente, con respecto a la seguridad requerida en la transmisión, se ha de comentar que los chipsets GPRS no implementan el protocolo de seguridad de la capa TLS (Transport Layer Security), situada entre el nivel de aplicación y el nivel de transporte mostrados en la tabla 1. Por el contrario, los de UMTS sí que la contemplan, por lo que con estos últimos se puede establecer una comunicación con niveles muy elevados de integridad y seguridad.

Así, se concluye que ambas tecnologías podrían ser válidas, pudiéndose elegir una u otra dependiendo de a qué necesidades se les dé más peso.

2.2. Tecnologías para la creación de redes inalámbricas

Seguidamente, se comentan las tecnologías existentes en el mercado para la creación de redes inalámbricas, concretamente las tecnologías más extendidas en este tipo de aplicaciones, que son Bluetooth [4] y ZigBee [5]. Estas tecnologías son especialmente recomendables para su uso en la interconexión de los nodos de una red de equipos.

Por otra parte, se ha descartado entrar a analizar tecnologías como WiFi o WiMax, debido a que disponen de grandes cantidades de recursos, que para esta aplicación van a quedar sin uso, a costa de un gran incremento, tanto en la complejidad del sistema, como en el consumo energético.

Analizando la topología de red, Bluetooth ofrece una única posibilidad a la hora de establecer una red, que sería del tipo estrella, y con un máximo de 8 dispositivos. Sin embargo, ZigBee permite hasta 65535 dispositivos, con varias posibles jerarquías, repartidos en hasta 255 subredes y pudiéndose hacer uso de cualquier topología de red, como se muestra en la figura 1.

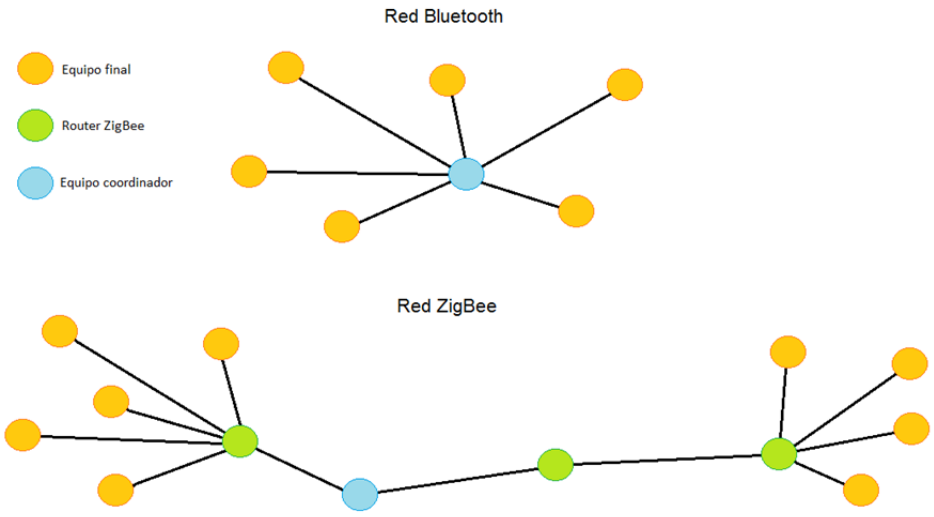


Figura 1: Comparación entre posible red Bluetooth y posible red ZigBee.

Por el contrario, Bluetooth permite tasas de transmisión de hasta 3000kbps mientras que las tasas de transmisión que puede cursar ZigBee son de entre 100 y 250kbps, lo que no resulta demasiado significativo debido a que, como se ha comentado anteriormente, no se han de transmitir grandes cantidades de datos, y además no se trata de un sistema de tiempo real.

Con respecto al ahorro energético, ambos sistemas tiene consumos eléctricos bajos, pero ZigBee algo menor. Además, éste alcanza coberturas mayores que Bluetooth, de hasta 1500 metros frente a 100 metros. Si a ello se le suma que ZigBee, al poder conectar muchos más equipos, puede emplear nodos cuyo único propósito sea hacer de repetidor o *router*, se tiene que el tamaño alcanzable por estas redes es muy superior a las redes Bluetooth, cubriendo así áreas muy extensas.

El desarrollo de esta funcionalidad no forma parte de este proyecto ya que se está desarrollando en paralelo en un trabajo fin de master dentro del GTF, en el que se hace uso de ZigBee. Sí que se contempla en el proyecto el rediseño de la parte hardware para la inclusión de la funcionalidad ZigBee.

2.3. Protocolo de nivel de transporte

En este punto, se analizan dos protocolos de nivel de transporte, para determinar cuál es el más adecuado para su utilización en el sistema a desarrollar. Se tratan los dos protocolos de este nivel más utilizados en la transmisión de datos: TCP (Transfer Control Protocol) [6] y UDP (User Datagram Protocol) [7].

TCP es un protocolo que proporciona fiabilidad debido al sistema de aceptaciones (ACK's) que utiliza. UDP no implementa nada parecido, por lo que en sus transmisiones pueden aparecer errores a nivel de aplicación. Esta fiabilidad, TCP la consigue a costa de incrementar el tráfico cursado en las comunicaciones, por lo que UDP únicamente cuenta con el incremento de tráfico debido a las cabeceras que incluye. Dicho incremento de tráfico genera una pérdida de eficiencia en la comunicación, pero esto no resulta demasiado significativo, sobre todo para transmisiones grandes, o si se emplean ventanas deslizantes de un tamaño óptimo.

TCP, al incluir tráfico de control, permite un menor número de conexiones a un servidor, pues parte de los recursos de los que éste dispone se emplean en dicho tráfico, cosa que no ocurre en UDP. Por el contrario, como TCP incorpora control de flujo, en caso de que los recursos empleados entre varias comunicaciones superasen los que posee en servidor, se reducirían las tasas de transmisión, evitando así incrementar los errores aparecidos, mientras que UDP no controla el flujo de datos, pudiendo saturar el servidor, perdiéndose datos y apareciendo errores en los recibidos.

Así, como en la aplicación para la que se van a requerir lo que interesa principalmente es la corrección de los datos obtenidos para su correcto análisis, y además el tráfico cursado no va a ser un parámetro limitante ni se va a requerir a la aplicación la transmisión en tiempo real, el protocolo de nivel de transporte más adecuado será TCP.

2.4. Protocolo de nivel de aplicación

Puesto que el protocolo de nivel de transporte elegido es TCP, en este punto se someten a análisis dos protocolos diferentes de nivel de aplicación que se apoyan sobre éste, como son FTP (File Transfer Protocol) [8] y HTTP (Hyper-Text Transfer Protocol) [9], destacando las características más importantes para la aplicación a desarrollar, y así determinar cuál es más recomendable en este caso.

El protocolo HTTP es un protocolo eficiente para el envío de datos, pues resulta muy flexible y adaptable, debido a las muchas formas de realizar peticiones que posee. Además, permite el envío, tanto de grandes bloques de información dentro de archivos de todo tipo, como de unos pocos bytes aislados, incluidos directamente en el cuerpo de una petición. Sin embargo, FTP está pensado para la transmisión de archivos, por lo que para transmitir pequeños bloques de información resulta ineficiente. Éste último, además, necesita dos conexiones TCP para realizar un envío, una para controlar el envío y otra para enviar u obtener el archivo, mientras que HTTP sólo requiere una. Además, las peticiones HTTP conllevan una respuesta por parte del servidor que incluye un código de error, en la que se puede introducir información para crear algo parecido a un sistema de aceptación, lo que no permite el protocolo FTP.

Por otra parte, hay que destacar que FTP incluye autenticación al inicio de la comunicación, de forma que no puede realizar un envío cualquier persona o dispositivo, únicamente quien disponga de la contraseña para acceder, mientras que con HTTP se tendría que implementar un sistema de autenticación.

Común a ambos protocolos es la falta de seguridad por medio de encriptación de datos, aunque en ambos casos se puede solucionar de la misma manera, que es implementando la capa de seguridad TLS comentada anteriormente, la cual provee de encriptación a los protocolos de nivel de aplicación, dando lugar a los protocolos HTTPS (Hyper-Text Transfer Protocol Secure) y SFTP (Secure File Transfer Protocol). Como se ha comentado anteriormente, esta capa sólo se implementa en los chipsets de UMTS, por lo que haciendo uso de GPRS no se pueden encriptar los datos.

Por último, destacar que la visualización de una página web conlleva el uso del protocolo HTTP, por lo que parece lógico emplearlo también en el envío de las medidas, y evitar mezclar protocolos. Además, permitirá tratar los datos directamente en el instante de recepción, en el *script*

de la página programada para ello, sin tener que esperar a lanzar ningún programa que los procese, como se debería hacer si se recibiesen a través de FTP.

Así, a pesar de que FTP facilita la autenticación en los envíos, se hará uso de HTTP para el envío de datos al servidor, y recepción de parámetros en los equipos, pues simplifica el procesado y almacenamiento de los datos, llevándose esto a cabo automáticamente, en el mismo instante de recepción de las medidas.

2.5. Estudio de servidores

Para la realización del servidor, se ha optado por desarrollar uno propio. Sin embargo, antes de acometer esta tarea, se han evaluado servidores y plataformas de almacenaje de datos comerciales. Dichos servidores comerciales hacen uso del protocolo HTTP, por lo que la elección anterior sería compatible con el uso de los mismos.

Existen en el mercado servidores comerciales que integran plataformas especialmente dedicadas al Internet of Things (IoT), de forma que permiten el almacenaje y visualización de los datos que se les remite, de las que se han estudiado tres: Xively [10], Exosite [11] y Carriots [12]. De éstas, destaca Xively debido al uso de HTTPS en la navegación por la web.

Inicialmente, el uso de servidores comerciales posee la ventaja de que las páginas web ya están programadas, y suelen ser muy intuitivas y estar bien explicadas, poseyendo en algunos casos extensas guías de aplicación. Además, no se ha de gestionar y mantener el servidor, tarea que puede resultar muy compleja.

Por otra parte, tienen un gran inconveniente para la aplicación a desarrollar y es que los datos enviados puedan ser públicos, pudiendo acceder a ellos cualquiera, además de que los gestionaría personal ajeno al grupo. Otra desventaja importante que conlleva el uso de servidores comerciales, es el no tener control sobre la dirección IP del servidor, que pueden cambiar, por lo que habría que estar pendiente para variarla manualmente en la configuración del equipo remoto.

Así, debido al control que permite mantener de todo el proceso y la privacidad y seguridad de la que se puede dotar a los datos, a pesar del incremento de complejidad que supone, se instala un servidor propio, al que habrá que dotar de los módulos necesarios para realizar las tareas que se implementen en las distintas secciones de la web. Además, la programación completa de las diferentes webs permite alcanzar la versatilidad necesaria para poder variar parámetros y extender el servicio todo lo que haga falta.

2.6. Conclusiones y diagrama final de la comunicación

En la figura 2, se muestran dos esquemas, a modo de pilas de protocolos, de la comunicación que se ha de establecer, diferenciando la posibilidad de tener o no una red de sensores conectada a un nodo colector (coordinador), que también es sensor. Además, se distinguen dos posibles casos,

dependiendo de si se hace uso de GPRS como tecnología de comunicación móvil, o si por el contrario se emplea UMTS.

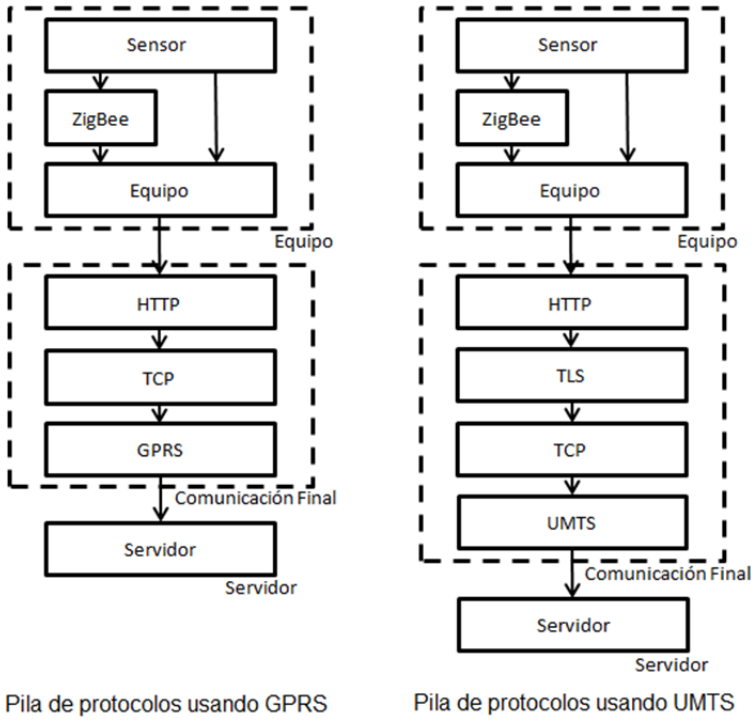


Figura 2: Representación esquemática de la comunicación resultante, según se emplee GPRS ó UMTS.

En esta figura se observa que la única diferencia entre el empleo de GPRS y UMTS, ya que con éste último, se puede implementar la capa de seguridad TLS, proveyendo a la comunicación de encriptación, que mejora la seguridad y la integridad de los datos en el tránsito entre equipo y servidor. Por otra parte, GPRS posee un menor consumo, permitiendo una mayor duración de las baterías, por lo que es esta tecnología la que se va a emplear en el desarrollo de la aplicación objetivo de este proyecto.

En ambos casos, el resto de protocolos son comunes, empezando por TCP, que se encarga de darle a la comunicación la fiabilidad necesaria, además de cierta integridad. El protocolo de nivel de aplicación será HTTP, que permite flexibilidad a la hora de realizar envíos y recepciones de datos, además de ser el más usado para este tipo de acciones, y el que se emplea en servidores y navegadores de internet.

Capítulo 3:

Desarrollo del sistema de recogida automática de datos

A continuación, se describe la fase del proyecto, en la que se desarrolla la aplicación final según las elecciones del capítulo anterior.

Primero, se va a desarrollar un servidor propio, en el que se va a programar una web destinada a la recolección, descarga y visualización de los datos obtenidos por los sensores, así como la configuración remota de los equipos de campo. La programación de cada sección de la web se encuentra detallada en el anexo 4. Se ha programado el conjunto de páginas web que precisa el servidor usando lenguajes de programación de páginas web como *HTML* [13], *PHP* [14] o *Javascript* [15], y partiendo de ejemplos encontrados en manuales de usuario de dichos lenguajes.

Segundo, se debe incluir el sistema de comunicación en las placas de los equipos de medida y reorganizar la placa existente, además de generar unas funciones para el microcontrolador que lleva el equipo, que configuren y realicen las comunicaciones con el servidor.

Tercero, se estimará tanto el tráfico cursado como el consumo energético que va a añadir al equipo la parte de comunicación, además de caracterizar el sistema de carga a utilizar.

3.1. Desarrollo del servidor

A continuación se describen las tres fases del desarrollo del servidor. La primera, en la que se monta y configura el servidor. La segunda, en la que se crea el conjunto de webs que se precisan para la interacción con los dispositivos autónomos, destinada a la recepción, el procesado y el almacenado de los datos. La tercera, en la que se crea la web requerida para la interacción con los usuarios, destinada a la gestión de los datos y los equipos remotos.

3.1.1. Montaje y configuración del servidor

A continuación, se describe el proceso seguido para el establecimiento de un servidor, a partir de un ordenador con sistema operativo *Windows*, proceso que se detalla más extensamente en el anexo 3. Como se precisa de un servidor con capacidad para generar páginas web dinámicas, hay que dar soporte al lenguaje de programación *PHP*.

Para la instalación del servidor, se hace uso de *Apache* [16] frente a la posibilidad de utilizar para ello *ISS* (Internet Information Service), ya que el primero, al ser de código abierto, cuenta con revisiones de seguridad cada cierto tiempo y es muy utilizado para el establecimiento de servidores personales, de forma que existe mucha información acerca de su instalación y configuración.

Como soporte de bases de datos, se emplea *MySQL* [17], que permite una gestión eficaz de las bases de datos que alberga, a las que sólo se permitirá el acceso desde el entorno local, para evitar ataques directos a las mismas. El principal inconveniente es que, al no contar con interfaz gráfica, es complejo de manipular y configurar, pero para eso se tienen soluciones programadas en *PHP*. En este caso se emplea *PHPMyAdmin* [18], el cual provee de interfaz gráfica a *MySQL* y permite visualizar los datos directamente en las tablas.

Para la representación en gráficas de los datos se hace uso de *FusionCharts* [19], cuya versión gratuita permite el uso de gran variedad de tipos de gráficas. Su programación resulta bastante sencilla al hacer uso de código *PHP* que genera instrucciones interpretadas por *Javascript*, por lo que las funciones a utilizar pueden ser incluidas directamente en el código de la página.

De modo que se instala *Apache* como servicio en el ordenador, realizando la configuración de éste y de los módulos *PHP*, *MySQL* y *PHPMyAdmin*, además de instalar *FusionCharts* y securizar dicho servidor y los módulos instalados [20] [21] [22] [23].

La distribución de los ficheros que conforman el servidor y la página web garantiza la seguridad de los datos enviados y los archivos con datos confidenciales. Ésta se muestra en la figura 3, en la que se encuentra en color azul la única zona del disco duro del ordenador a la que tiene acceso el servidor *Apache*. En dicha zona se encuentra la web con las secciones de visualización, descarga y gestión de la red (webs), accesible para los usuarios. Los dispositivos autónomos acceden a un directorio (sens) que se encuentra en la zona marcada y que contiene la web de recogida de datos. Además, se encuentra también en dicho directorio una zona de administración con el entorno gráfico para *MySQL* destinado a la gestión de las tablas de datos y un *script* de gestión, que permite la creación tanto de nuevos usuarios de la web como de sensores. Estas herramientas de administración sólo son accesibles para el administrador y desde el entorno local (modo *localhost*).

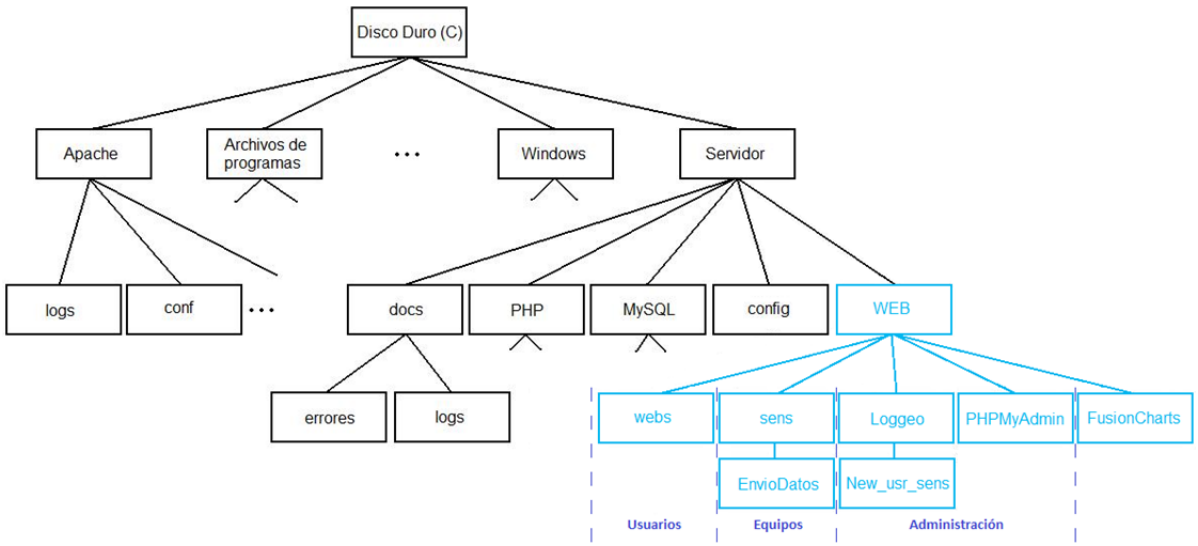


Figura 3: Esquema general de la organización de los ficheros dentro del disco duro. Los ficheros en color azul son los únicos accesibles para el servidor.

Por último, las bases de datos se organizan de modo que se separan las tablas en tres bases de datos. En una, se almacena la tabla con los usuarios y contraseñas de acceso a la web y la tabla con los identificadores de sensor y *tokens* para el envío de los datos y reconfiguración. En otra, se guardan las tablas de medidas y en la última se guardan las tablas con las configuraciones de los equipos remotos. De esta forma, se puede tener un usuario de *MySQL* especial dedicado al acceso desde las webs a las bases de datos, para protegerlas de ataques que traten de borrarlas. Por ello, dicho usuario tiene diferentes privilegios sobre las distintas bases de datos, limitados según los datos que almacenen y el uso que tenga que hacer de los mismos.

3.1.2. Web de interacción con los dispositivos autónomos

A continuación, se detalla la web encargada de la interacción con los equipos remotos. Primero hay que definir cómo se envían los datos de los equipos autónomos, a través del protocolo *HTTP*, del que se detallan algunos tipos de peticiones en el anexo 2. Estas peticiones, deben llevar ciertas cabeceras que permitan que el receptor del mensaje sepa descifrar el contenido del cuerpo de la petición. Así, de todo el conjunto de peticiones que posee este protocolo, se emplean GET, para el envío del *token* que realiza las funciones de contraseña, y POST, usada para el envío del identificador de sensor y los datos, ambas en la misma petición de envío de datos, como se muestra a continuación:

```
POST /camino_script/script.php?token=<token> HTTP/1.1
Host: <IP_servidor>
Connection: close
Content-Type: application/x-www-form-urlencoded;
Content-Length: <longitud_cuerpo>

Dato1=22&Dato2=22&Dato3=22
```

La petición necesaria para realizar el envío del archivo de comprobación, requiere cambiar la estructura de la parte correspondiente al método POST, de forma que queda:

```
POST /camino_script/script.php?token=<token> HTTP/1.1
Host: <IP_servidor>
Connection: close
Content-Type: multipart/form-data; boundary=<boundary>
Content-Length: <longitud_cuerpo>

--<boundary>
Content-Disposition: form-data; name="sensor"

<nombre_sensor>
--<boundary>
Content-Disposition: form-data; name="archivo"; filename=<nombre_archivo>
Content-Type: text/plain

<fecha>\t<dato1>\t<dato2>
<fecha>\t<dato1>\t<dato2>
--<boundary>--
```

Este uso conjunto de ambos protocolos se emplea mucho en las páginas web para la generación de sesiones de conexión. Destacar que debido a la inclusión de la cabecera "*Connection*", la sesión se cierra de forma controlada desde el servidor, por lo que hay que tener esto en cuenta en el software del dispositivo remoto.

El diagrama de flujo seguido en la web dedicada a la recepción de las medidas tomadas por los equipos remotos, es el mostrado en la figura 4, donde se puede ver que hay que autenticar el sensor con su identificador y su *token*, de forma que en caso de no ser correctos, se desechan los datos recibidos en esa comunicación.

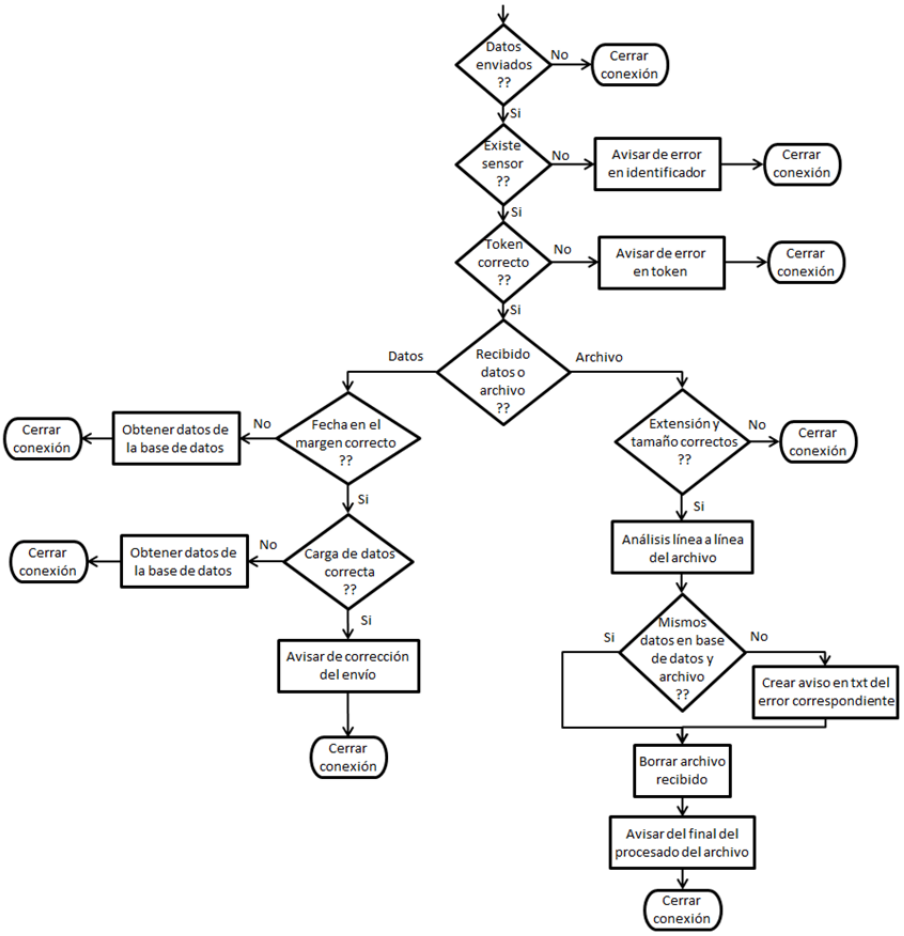


Figura 4: Esquema de la web a la que se envían los datos para que los procese y almacene en la base de datos.

Como se observa, el tratamiento recibido por los datos en caso de recibirse dentro de un archivo o recibirse aislados dentro del cuerpo de la petición, es diferente, de forma que si los datos se reciben aislados, hay que asegurarse de que pertenecen a un intervalo de tiempo posterior al último dato recibido desde ese sensor, por lo que se interroga a la base de datos en busca de éste.

Si por el contrario, llegan dentro de un archivo, al ser tomados como confirmación de los datos enviados ese día, se comprueba que todos están en la base de datos, y que todos los que están en la base de datos están también en el archivo. Si esto no ocurre, se toma como un error ocurrido en la comunicación, y para tenerlos organizados y facilitar su análisis, éstos se cargan en archivos diferentes, dependiendo de si son debidos a una medida que aparece en el archivo y no en la base de datos, o al revés.

Como respuesta a las peticiones de ingreso de datos, el servidor transmite una cadena que, en caso de que no haya que reconfigurar parámetros del equipo, es:

```
HTTP/1.1 200 OK
Date: Wed, 15 May 2013 14:32:17 GMT
Server: Apache
Content-Length: 16
Connection: close
Content-Type: text/html

Datos procesados
```

En caso de tenerse que añadir reconfiguraciones en la respuesta, éstas se situarían en la cadena, detrás de "*Datos procesados*".

Por otra parte, se ha creado un *script* encargado de dar la hora del servidor. Este *script* sólo será accesible desde los equipos remotos, pues se puede necesitar configurar la fecha y la hora de su reloj de forma remota, de modo que la podrían obtener automáticamente de este *script*.

3.1.3. Web de interacción con los usuarios

A continuación, se comenta la creación de la web destinada a la interacción con los usuarios. Destacar que todas las secciones de ésta, cuya estructura se presenta en la figura 5, en caso de no haber iniciado sesión previamente, redireccionan al usuario hacia la página de *login*. En esta página, se comprueba que alguna de las parejas usuario de web y contraseña, guardadas en la base de datos, se corresponde con la introducida en el formulario de acceso por el usuario, de forma que se restringe el acceso a intrusos. Por otra parte, y una vez iniciada la sesión con el usuario y la contraseña correctos, se accede a una web que dispone de diferentes secciones, donde se pueden visualizar y descargar los datos, completos o por rango de fechas, así como controlar los errores encontrados y reconfigurar el equipo remoto. También se incluye un enlace a la página de *logout*, que es la que destruye las *cookies* que generan la sesión de navegación.

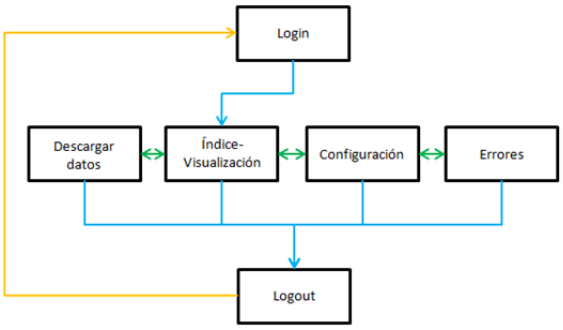


Figura 5: Esquema general de la web a crear. Los enlaces azules verdes unen todas las secciones de la web, pues desde una de ellas se puede acceder a cualquiera de las otras.

De esta forma, se concluye con el trabajo realizado sobre el servidor, quedando por realizarse el desarrollo de la parte correspondiente al equipo remoto.

3.2. Desarrollo del dispositivo autónomo

Finalmente, se presenta el desarrollo de la unidad de comunicaciones propiamente dicha, que debe integrarse tanto en el software como en el hardware existente de los equipos de medida, con las menores modificaciones posibles.

3.2.1. Hardware

El equipo final debe realizar el proceso de medida de la siguiente forma: Partiendo del estado de reposo (con el microcontrolador durmiendo y el *módulo GPRS* desconectado) el controlador recibe una interrupción de un reloj de tiempo real. Éste se despierta, realiza las medidas del sensor y las guarda en el archivo del día de la tarjeta SD (Secure Digital). Tras esto, enciende el *módulo GPRS* y envía las medidas al servidor. A continuación, el microcontrolador apaga el módulo, reprograma la alarma para la siguiente medida y se vuelve a dormir.

En el desarrollo hardware se parte de un diseño ya existente y se rediseña para incluir el *módulo GPRS*, el *módulo ZigBee* (aunque no se usará en este proyecto), el sistema de carga de las baterías con el panel solar y una conexión con un reloj que permita despertar al sistema de forma programada. Estas modificaciones se muestran en color azul en la figura 6.

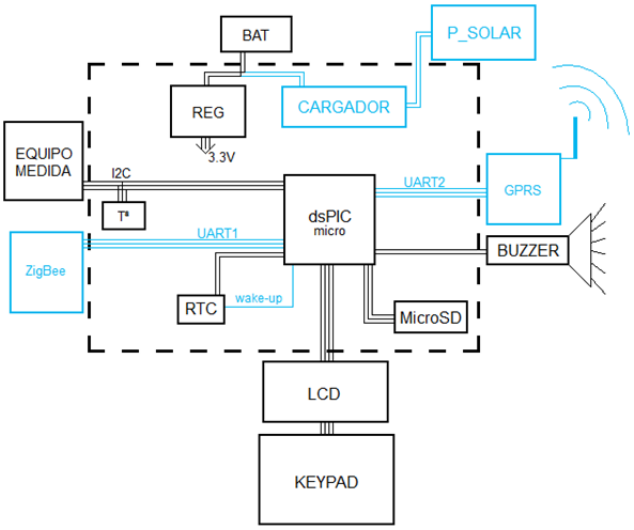


Figura 6: Diagrama de bloques de la composición de las placas controladoras de los sensores. En azul, los componentes a incluir o sustituir en la placa existente.

En la placa de partida se tiene ya conectado un reloj de tiempo real al bus I2C (Inter-Integrated Circuit) del microcontrolador, que es el *DSPIC33FJ128GP710* de *Microchip*, para tomar la fecha y la hora en el momento de realizar la medida. Además, está conectada una de las salidas del microcontrolador a un zumbador, empleado solamente para las pruebas, que se activa cuando se van a realizar medidas. También se tiene conectado al bus I2C un medidor de temperatura del equipo, cuyos datos son los utilizados para realizar las pruebas de funcionamiento del sistema final. Por otra parte, se tiene un teclado y una pantalla LCD (Liquid-Crystal Display), que sirven para realizar las pruebas y tareas de mantenimiento del equipo, desconectándose cuando se lleve el equipo a campo para eliminar su consumo, y que se conectan al microcontrolador por el bus I2C.

En este diseño, falta por integrar, como ya se ha dicho, el *módulo GPRS*, que debe conectarse al puerto serie 2 (UART2 - Universal Asynchronous Receiver-Transmitter) del controlador. También debe incluirse el *módulo ZigBee*, para su uso futuro, que se conecta al puerto serie 1 (UART1), además de la parte de carga del panel solar y la batería. Por otra parte, en la entrada de alimentación del *módulo GPRS* se conecta también un sistema para conectar y desconectar dicha alimentación. Finalmente, también se tiene que conectar una salida de alama del reloj a una entrada de interrupción del micro para poder despertarlo y que realice el proceso de medida.

El *módulo GPRS* de *Cooking Hacks* (figura 7) a incluir, dispone del chip de comunicaciones *HiLO* de *SAGEM*, el cual cuenta con un puerto serie para su control, y un pin para el encendido del módulo. Éste se puede alimentar con tensiones de entre 3.2V y 4.5V, por lo que como las baterías de litio utilizadas no alcanzarán los 4.5V, éstas se conectarán directamente al módulo sin atravesar el regulador que éste lleva, ya que la caída de tensión entre sus extremos es muy alta, haciendo que en muchas ocasiones, no se pueda alcanzar la tensión mínima requerida que funcione el chip de comunicaciones.

Figura 7: Módulo GPRS de Cooking-Hacks, con el chip de comunicaciones HiLO de SAGEM montado.

Como hay que insertar los módulos GPRS y ZigBee en el diseño, lo primero que hay que hacer es crear una librería, que contenga a ambos, para su uso en el programa de diseño de circuitos impresos *CadSoft Eagle*. Para ello, se crean los componentes, de los que se eliminan los pines que no se van a usar, para facilitar su inserción entre la circuitería ya existente. Para facilitar el montaje de los módulos en la placa, se emplean zócalos de modo que se evite el tener que cortar conexiones o desoldarlas.

El resultado final de la placa fabricada y montada se muestra en la figura 8. En ella se observa el módulo GPRS con su antena situado sobre los zócalos utilizados para su conexión con la placa que se ha rediseñado, junto al módulo ZigBee. También se observa la tarjeta SD junto a su ranura de conexión, y la posición en la que está colocado el reloj de tiempo real.

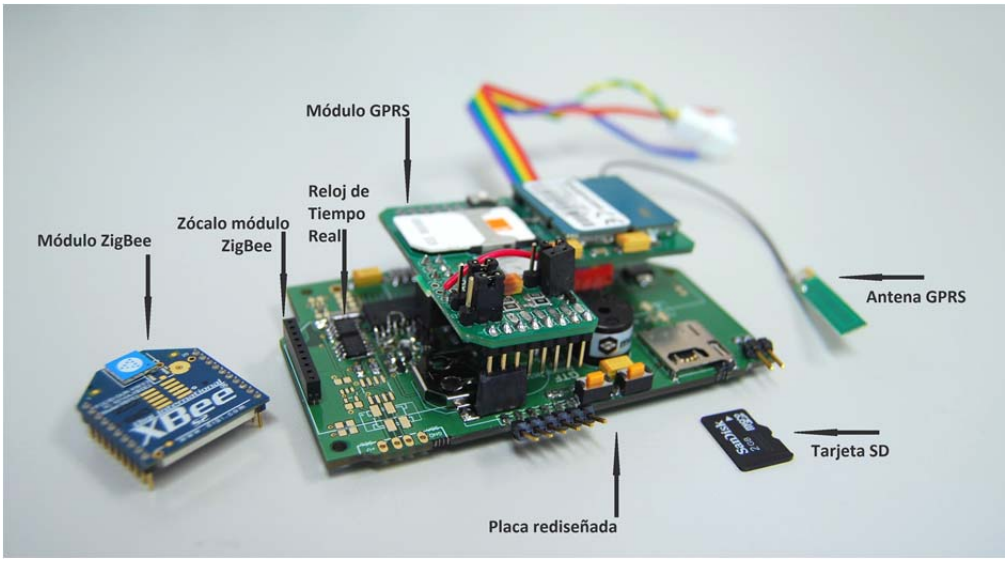


Figura 8: Imagen de la placa de control de sensores modificada, con el módulo GPRS montado.

3.2.2. Software

En este punto se implementan las funciones necesarias para comunicar el módulo GPRS con el servidor. Para ello, primero se tiene que generar un diagrama de flujo de la comunicación, que

muestre los diferentes errores que puedan aparecer en ella, de forma que se puedan buscar e implementar las soluciones que permitan el funcionamiento autónomo del equipo. Hay que tener en cuenta que las funciones que se vayan a crear para establecer y llevar a cabo la comunicación, deben integrarse perfectamente en el software ya desarrollado de los sensores, sin generar interferencias en el flujo de dicho software.

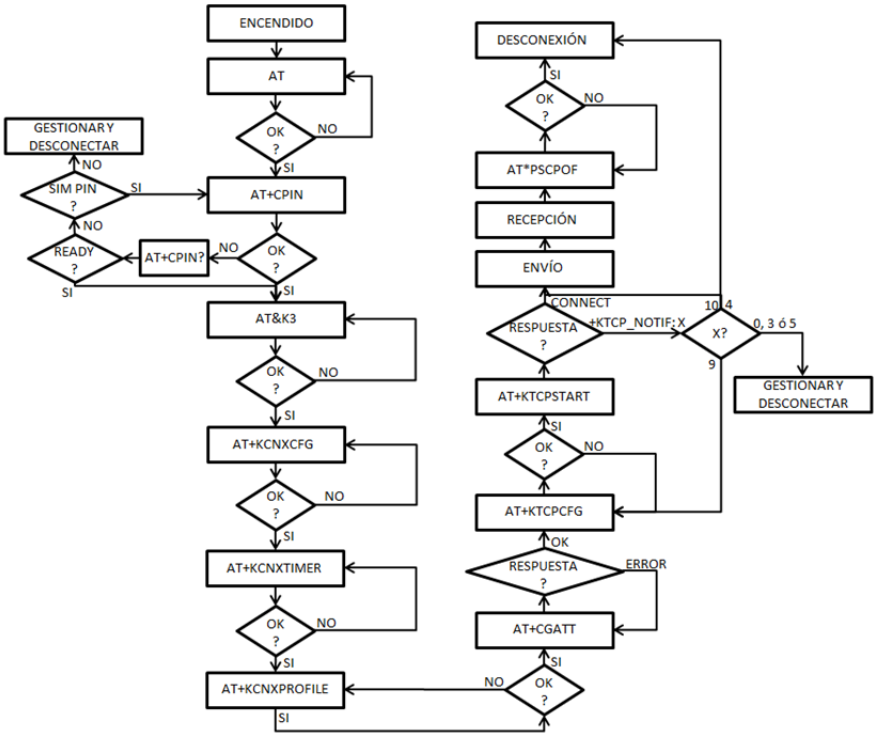


Figura 9: Diagrama de flujo y control de errores de los comandos implicados en el establecimiento de la comunicación.

En la figura 9, se muestra un diagrama de flujo con la secuencia de comandos necesaria para el establecimiento de la comunicación, junto con la gestión algunos errores que se pueden dar. Tras cada instrucción, se espera la respuesta del módulo ("*OK*") que indique que ha sido procesada. Estas esperas son críticas antes de la instrucción "*AT+CGATT*", que devolverá "*ERROR*" mientras no se haya completado la secuencia de ingreso en la red móvil producida tras la inserción del PIN, y después de la instrucción "*AT+KTCPSTART*", donde hay que esperar la respuesta "*CONNECT*", que indica que se ha iniciado la conexión con el servidor. Por otra parte, se puede dar un error relacionado con el comando de introducción del PIN, el cual tan sólo se podrá reintroducir una vez, si ha fallado, para evitar bloquear la SIM, por lo que si se dan dos errores, se desconectaría el módulo y se reportaría dicho error a un archivo contenido en la memoria del equipo. Otro punto crítico respecto a errores está localizado en el comando "*AT+KTCPSTART*", que puede notificar un fallo de la red, en cuyo caso no se podrá comunicar con el servidor, por lo que se desconectaría el módulo y reportaría el error al archivo mencionado.

Una vez conocidos los errores que pueden aparecer y cómo tratarlos, se pueden generar las funciones necesarias para establecer la comunicación. Para ello, se usa la herramienta "*Mplab X IDE*", que es la suministrada por el fabricante del microchip para su programación. Además, para facilitar su programación, se hace uso de las librerías de funciones que el GTF se ha creado para su manejo. Con esto, se crean las funciones que enciendan y apaguen el dispositivo, y que lleven a cabo

la transmisión, además de crear una variación de la función de lectura del puerto serie, adaptada a las necesidades de esta sección del programa.

3.3. Pruebas del sistema de recogida autónoma de datos

Como ya se ha comentado, puede resultar interesante el estudio de los consumos, tanto relativo al tráfico como energético, para poder hacer una buena elección en cuanto a la tarifa de datos a utilizar, y al sistema de alimentación formado por un panel solar y una batería, el cual también interesa caracterizar. Así, se comienza por el tráfico de datos cursado, se continúa con un análisis del consumo del módulo GPRS, para concluir con la caracterización del sistema de carga, siempre partiendo de unos parámetros de los envíos que no son definitivos, pero que pueden servir como una primera aproximación para establecer supuestos pesimistas y limitantes.

3.3.1. Estudio de tráfico generado

Se comienza por el estudio de tráfico, del que se lleva a cabo una mayor profundización, en cuanto a los supuestos y los cálculos realizados, en el anexo 5. En la estimación de este, se utiliza el tráfico a nivel de enlace, tanto transmitido como el recibido, puesto que las compañías telefónicas tienen en cuenta ambos tráficos a la hora de facturar.

Inicialmente, se supone una ventana deslizante de tamaño 1 paquete, MTU (Maximum Transfer Unit) de 1500 bytes, que no aparecen errores en la comunicación, y que se realizan transmisiones cada 15 minutos, enviándose una vez al día el archivo. Tanto la cabecera TCP como la IP serán de tamaño 20 bytes y el campo “boundary”, necesario para el envío de archivos, será de 10 bytes. Además, se tienen en cuenta en las estimaciones, los datos de la tabla 3:

Cabeceras HTTP para datos	182 bytes
String de datos	222 bytes
Respuesta del servidor	152 bytes
Cabeceras HTTP para archivo	395 bytes
Tamaño del archivo	12288 bytes

Tabla 3: Datos empleados en la estimación del tráfico cursado.

Con estos datos, el tráfico cursado en un mes sería de unos 3.24MB, que es una cantidad muy pequeña, como se suponía en un principio. La tarifa elegida finalmente es la “Delfín” de Orange, que proporciona 100MB de tráfico semanal. Teniendo esto en cuenta, se estima el tráfico generado semanalmente, que sería de 748.75KB, y se concluye que con esta tarifa, se podrían conectar hasta 136 sensores a través de una misma tarjeta SIM (Subscriber Identity Module). Si se simula un tamaño de ventana deslizante mayor o un valor de MTU mayor, la reducción de tráfico obtenida es muy pequeña, por lo que se puede suponer que el tráfico es aproximadamente independiente de este parámetro. Si se tiene en cuenta el *piggybacking*, al ahorrar una aceptación por transmisión, se obtiene un ahorro de entre 26KB y 27KB por semana, permitiéndose así el uso de la tarjeta SIM para, aproximadamente, 5 sensores más. Por otra parte, un incremento en el tamaño de la respuesta del servidor puede llegar a incrementar mucho el tráfico, ya que esta respuesta aparece en todas las comunicaciones. Como ejemplo de esto último, un incremento de tan sólo 30 bytes en la respuesta del servidor, conlleva un incremento de casi 20KB de tráfico semanal. En el otro lado de la

comunicación, si se reduce la cantidad de datos que envía cada sensor, se reduce mucho el tráfico, como por ejemplo si se envían sólo 3 datos, en cuyo caso se reduciría en casi 145KB semanales.

De este modo, queda claro que, como se suponía en un principio, el tráfico cursado por los equipos de medida no va a ser elevado, dependiendo principalmente de los datos a transmitir y el periodo de medida, por lo que con 100MB semanales se dispone de recursos de red suficientes.

Para concluir con este análisis comentar que, en las pruebas finales realizadas, se ha observado que el tráfico cursado es menor que el aquí estimado, y que se corresponde con el tráfico que se obtendría si no se tuviesen en cuenta las cabeceras de nivel IP.

3.3.2. Estudio de consumo energético

Es importante la estimación del consumo eléctrico que posee el *módulo GPRS* de *Arduino*, debido la naturaleza autónoma del equipo en el que se va a utilizar. Según las hojas de características del módulo, el consumo de éste puede llegar a ser de hasta 2.2A durante los picos producidos por el envío de datos, si la distancia a la estación base requiere de la máxima potencia de transmisión. Estos picos son periódicos durante la transferencia, correspondiéndose con los slots de transmisión que la estación base ha asignado al dispositivo. Se va a caracterizar, tanto el consumo mientras se mantiene el módulo dormido, como el consumo cuando se apaga el módulo, para así poder determinar si resulta mejor opción dormirlo, de modo que la configuración del módulo no tendría que hacerse cada vez que se va a transmitir, o apagarlo, en cuyo caso la configuración se perdería. En el anexo 6 se detallan tanto el montaje como las medidas obtenidas y los cálculos realizados para la estimación del consumo en cada caso.

Tras realizar las pruebas expuestas en el anexo 6, se puede representar una estimación del perfil de consumo correspondiente a la transmisión que se puede tener. Dicho perfil se muestra en la figura 10, donde se pueden observar diferentes zonas de consumo, en este caso apagando el módulo tras la transmisión:

- Zona A: Durante el proceso de encendido, mientras se configura el módulo, tiene un consumo medio de unos 50mA.
- Zona B: A continuación, el sistema está encendido y leyendo la información de todos los operadores que le llegan desde la estación base, consumiendo unos 74.9mA de media.
- Zona C: Cuando se introduce el PIN (Personal Identification Number) de la tarjeta SIM, comienza un intercambio de información del módulo con la estación base, de forma que éste se suscribe a la red móvil del operador marcado por dicha tarjeta, y obtiene los datos de la misma, consumiendo unos 106.41mA de media.
- Zona D: Una vez activado el acceso a la red, comienza el envío de comandos para configurar el acceso de datos antes de iniciar la transmisión. Por otra parte, la zona D que aparece tras la recepción (zona F), corresponde al envío del comando que apaga el módulo. En estas zonas, se consumen unos 50mA de media mientras interpreta el comando, y unos 34.4mA entre ellos.
- Zona E: Tras esto, se realiza la transmisión de los datos al servidor, por lo que se da el mayor consumo energético, que se de unos 183.6mA de media.

- Zona F: Una vez se ha realizado el envío, el servidor responde, y el módulo procesa los datos recibidos, consumiendo unos 146.88mA de media.
- Zona G: Tras la recepción de la orden de apagado, el módulo concluye su conexión a la red móvil del operador correspondiente, antes de apagarse, consumiendo unos 34.4mA de media.

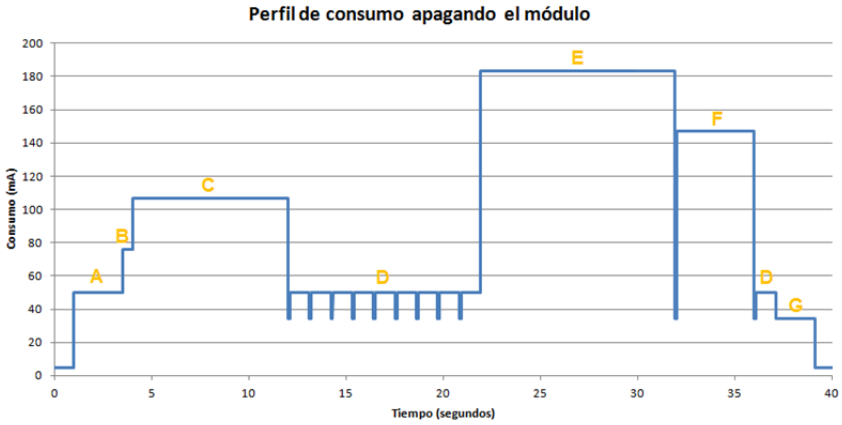


Figura 10: Perfil de consumo medio de una transmisión terminada apagando el módulo.

Con estos datos, se estima un consumo medio, realizando medidas y transmisiones cada 15 minutos, de 10.86mA apagando el módulo tras la transmisión y 12.87mA durmiendo el módulo, de modo que el ahorro de batería que se produce apagando el módulo frente a dormirlo es del 15%, aunque la diferencia sea de sólo unos 2mA.

El módulo, cuando se apaga, queda inaccesible desde la red, mientras que si se duerme, puede recibir llamadas y mensajes. Así, la conveniencia entre apagar el módulo o dormirlo depende del tiempo que vaya a quedar inactivo, ya que para esta aplicación no requiere escuchar las posibles llamadas o mensajes desde la red. En este caso, la idea inicial es realizar medidas en periodos de 15 minutos, por lo que la mejor opción sería apagar el módulo tras terminar la comunicación.

Por otra parte, el consumo obtenido con el módulo apagado es bastante grande, por lo que se opta por cortarle la alimentación, con unos transistores, mientras vaya a estar apagado, para así reducir dicho consumo. De esta forma, se obtiene un consumo medio, apagando el módulo, de 6.43mA.

Finalmente, comentar que en las pruebas finales realizadas del sistema completo, se han optimizado los tiempos de espera entre comandos, solapando también la ejecución, por parte del módulo, de alguno de ellos, consiguiendo así reducir el tiempo que requiere estar encendido el módulo. De esta forma, se ha reducido el consumo energético medio, que era de 6.43mA, hasta 2.3mA, que es un consumo muy pequeño comparado con el que generan el resto de los componentes.

3.3.3. Estudio de carga del panel solar

Hay que probar la capacidad de carga que tiene el panel solar del que se dispone, lo que se detalla más en profundidad en el anexo 7, ya que hay que cambiar la alimentación, que antes se realizaba a través del puerto USB (Universal Serial Bus). Para ello, se parte de que el consumo medio del *módulo GPRS*, estimado anteriormente, que está en torno a los 2.3mA. A este consumo, hay que

sumarle también el consumo total del microcontroloador, que es de unos 40-45mA. Así, se puede establecer un consumo medio aproximado de unos 50mA, el cual se puede simular, teniendo en cuenta que la placa de circuito trabaja a 3.3V, con una resistencia de aproximadamente 66Ω.

El panel solar utilizado puede generar una tensión de unos 7.5V entre sus terminales, por lo que, teniendo en cuenta que el cargador a usar trabaja correctamente con tensiones menores a 7V, se coloca un regulador de tensión de 5V a su salida para maximizar la corriente de carga.

Con esto, se monta un sistema de carga de baterías, con la resistencia comentada consumiendo unos 50mA, para medir principalmente la tensión de batería y la corriente de carga, obteniendo los resultados mostrados en la figura 11:

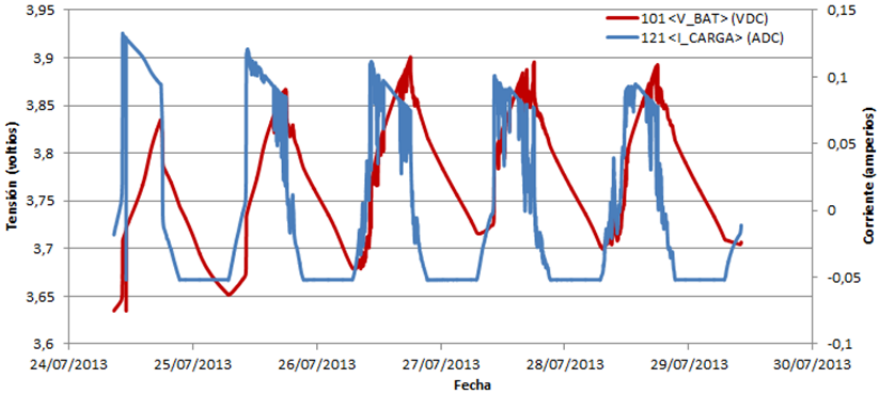


Figura 11: Tensiones de batería y corriente de carga obtenidas del montaje expuesto.

En esta gráfica, se observa como en los primeros días, la tensión de la batería al final de cada día es mayor que la tensión que se tenía 24 horas antes, por lo que se está cargando más de lo que se está gastando. En los dos últimos días, como consecuencia de un aumento de la nubosidad, que se ve reflejada como descenso de corriente de carga, la batería va perdiendo carga día a día. Por otra parte, también se observa que la corriente de carga llega a caer a unos -50mA de noche, cuando el panel no está cargando, que es la corriente que consume la resistencia. Además, se observan también dos irregularidades. Una de ellas es que la intensidad de carga no crece al principio del día linealmente hasta el punto máximo, defecto provocado por el hecho de que el panel solar está en sombra al principio del día, ocurriendo lo mismo al final del día. La otra es que la intensidad llega al punto máximo, y va cayendo conforme avanza el día, lo que se ha identificado con el hecho de que la batería esté casi cargada, de forma que requiere un menor cargado. Lo mismo se observa también en la tensión de batería, ya que va aumentando más lentamente. Esto habrá de ser tomado en consideración, pues en campo, el entorno puede ser más complejo, apareciendo sombras indeseadas. Además, en invierno los días son más cortos, y pueden darse días más nublados o con niebla que no permitan alcanzar la capacidad de carga observada en la figura 11.

3.4. Adaptación para el módulo 3G

Se han estudiado también las variaciones que habría que realizar para la inserción del *módulo 3G/GPRS* de *Arduino* (figura 12), en el lugar que ocupa el *módulo GPRS*, en caso de que se requiera encriptar los datos usando el protocolo HTTPS. En el anexo 8 se tiene un desarrollo más completo de los cambios necesarios para adaptar el sistema completo al *módulo 3G*.

Figura 12: Módulo 3G/GPRS de Arduino, con el chip de comunicaciones SIM5218 de SIMcom montado.

Por un lado, habrá que reconfigurar el servidor para que sea capaz de desencriptar las peticiones HTTPS. Esta se ha de realizar en el archivo existente expresamente para ello, y que posteriormente se incluirá en el de configuración general, donde además habrá que activar el uso del módulo correspondiente. Para el uso de este módulo, se tendrá que crear un certificado digital del servidor para las conexiones desde navegador, y una clave de encriptado de datos que se utilizará en todas las comunicaciones.

Por otro lado, dentro del equipo remoto, habrá que adaptar el hardware. Este módulo posee más pines que el *módulo GPRS*, de los que sólo se usarán el puerto serie, el pin de encendido, y los pines de alimentación, que al estar diseñados ambos módulos para *Arduino*, en ambos será los mismos pines, por lo que serán compatibles en este aspecto. Sin embargo, el tamaño del *módulo 3G* es mayor, por lo que, en la librería de los componentes de comunicaciones, habría que crear un elemento nuevo.

Con respecto al software, no se iniciará una sesión TCP tal cual, ya que para poder hacer uso de la encriptación proporcionada por la capa TLS hay que emplear los comandos propios de la conexión HTTPS, de forma que éstos se encargan de configurar toda la conexión en nivel de transporte, y de negociar la encriptación con el servidor. Dicha negociación, se llevará a cabo al inicio de la comunicación, para establecer los parámetros de encriptación que se van a utilizar, por lo que se generará un tráfico extra en cada comunicación. Como este tráfico extra será relativamente grande comparado con las cantidades de datos manejadas en la mayoría de conexiones, se incrementará bastante el consumo de datos, reduciéndose así la cantidad de sensores que se podrán establecer en una red controlada por un único dispositivo. Por el contrario, se conseguirá una mayor seguridad en la comunicación, además de una gran mejora en la integridad de los datos transportados.

Con esto, quedarían definidas todas las variaciones, tanto hardware como software, que se han de llevar a cabo para adaptar el diseño completo al uso del nuevo módulo.

Capítulo 4:

Conclusiones y líneas futuras

En este proyecto se ha desarrollado un sistema de comunicaciones inalámbrico, apoyado en el acceso a las redes de datos proporcionado por las tecnologías móviles comerciales, que permite tanto la obtención de los datos tomados por los sensores remotos como la configuración de los propios equipos y que se ha incluido en una de las placas de control diseñada por este grupo. Además, estos datos y variables se pueden consultar rápidamente desde cualquier ubicación, y almacenarse de forma segura y privada en un servidor alejado de los sensores.

Para ello, se ha desarrollado un servidor que alberga un conjunto de páginas web dedicadas, a la recepción de los datos de los equipos autónomos y a la gestión de los datos y los equipos. También se ha implementado la comunicación mediante peticiones HTTP realizadas sobre conexiones TCP a través del acceso a las redes de datos proporcionado por la tecnología de comunicaciones inalámbricas GPRS. Además, se ha desarrollado el dispositivo autónomo que tiene que establecer la comunicación desde el equipo remoto y que es compatible con los sensores del grupo.

Este sistema cumple con el requisito de un consumo eléctrico minimizado que garantice la máxima autonomía energética del equipo mediante el uso de baterías y paneles solares. Para ello, se han optimizado los tiempos de espera necesarios entre los diferentes comandos y se han solapado algunas acciones, reduciendo así el tiempo empleado en la comunicación. El sistema desarrollado también cumple con el requisito de establecer una comunicación fiable y que garantice un buen nivel de integridad de datos mediante el uso del protocolo TCP sobre la red de conmutación de paquetes accesible mediante GPRS. El sistema cumple con el requisito de seguridad y privacidad de datos en el almacenamiento de éstos. El sistema cumple con el requisito de compatibilidad con los sensores y con ZigBee.

Con el desarrollo realizado, la adaptación para una futura inserción del *módulo 3G* en el lugar del *módulo GPRS* no resulta complicada, de forma que con este cambio se conseguiría dar soporte a la encriptación de los datos, maximizando la integridad y seguridad de los datos. Por otra parte, queda abierta la posibilidad de desarrollar una funcionalidad de reconfiguración de los equipos remotos en tiempo real.

PARTE II

ANEXOS

Anexo 1:

Análisis de tecnologías y protocolos

En este anexo se pretende exponer las características principales de cada una de las tecnologías comparadas en el capítulo 2 de la memoria.

GPRS (General Packet Radio Service)

Puesto que el sistema GPRS se apoya en su predecesor, el sistema GSM (Global System for Mobile communications), se comenzará por definir éste último, que es un sistema de comunicaciones digitales de segunda generación (2G) basado en redes celulares radio, cuyas células se agrupan en clústeres donde no se reúsan frecuencias. Este sistema está diseñado para cursar principalmente tráfico de voz, por lo que se basa en conmutación de circuitos. Emplea acceso FDMA/TDMA/FDD (bandas pareadas), con 8 slots temporales para cada portadora frecuencial. Posee 500 portadoras frecuenciales por dúplex, repartidas en las bandas de 900MHz y 1800MHz, de 200Khz de ancho de banda cada una. Para la transmisión y recepción de la información, hace uso de un único slot temporal en cada dúplex, de forma que permite tasas de datos de hasta 9.6kbps, suficientes para tráfico de voz, debido al muestreo y codificación que se lleva a cabo.

GPRS es un sistema de comunicaciones digitales intermedio entre la segunda y tercera generación (2.5G) basado en GSM (figura 13), y cuyo propósito principal era el de permitir tráfico de datos con relativa eficiencia, por lo que se basa en conmutación de paquetes. GPRS puede llegar a hacer uso de hasta 4 slots temporales para la recepción de datos, y 2 para la transmisión (clase multislot 4+2), lo que le permite alcanzar tasas de datos de hasta 144kbps.

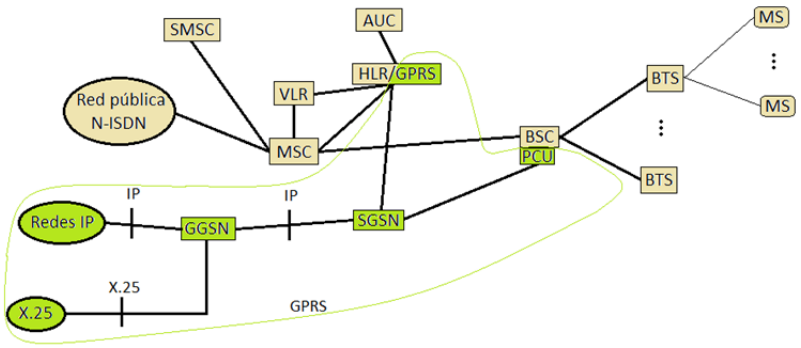


Figura 13: Arquitectura de red GPRS. En verde, módulos que agrega GPRS a la arquitectura de GSM ya existente.

Cabe destacar también, que la probabilidad de bloqueo de los recursos no es nula, aunque sí muy baja, ya que las redes celulares se diseñan suponiendo probabilidades de bloqueo típicamente del 2%, o como mucho del 5%.

En esta generación, se lleva a cabo el control de potencia por parte del dispositivo móvil, adaptando la potencia transmitida, según el nivel y la calidad de la señal recibida por un canal común de control, con la que estima las pérdidas de la señal en la transmisión entre equipo y estación base. Gracias a esto, los sistemas de segunda generación consiguen grandes ahorros de energía, permitiendo, tanto reducir el tamaño, como alargar la vida y duración de las baterías.

Acerca de estos sistemas, es destacable el hecho de que la tecnología esté ya en fase de madurez, lo que permite que se pueda garantizar la existencia de cobertura en prácticamente la totalidad del territorio nacional, manteniendo además niveles de disponibilidad elevados. Por otra parte, el número de portadoras en cada estación base viene limitado por la distancia de reúso de frecuencias que se tenga, no pudiendo tenerse todas las portadoras disponibles en todo el área. Es por esto por lo que la disponibilidad de recursos está bastante limitada, haciendo que el funcionamiento del sistema esté supeditado a que la disponibilidad, en la célula en la que se inscriba al acceder a la red, no cayese demasiado, lo que por otra parte es poco probable.

UMTS (Universal Mobile Telecommunications System)

El sistema UMTS es un sistema de comunicaciones digitales de tercera generación (3G) basado en redes celulares radio, e ideado para poder integrar diferentes servicios, por lo que cuenta con tasas de transmisión de entre 144kbps y 2Mbps, dependiendo del nivel de calidad requerido para cada servicio o que pueda garantizar la red en ese momento. Emplea acceso CDMA por secuencia directa (DS-CDMA (Direct Sequence Code Domain Multiple Access)), destinando un ancho de banda de 60+60MHz para acceso CDMA de banda ancha (WCDMA (Wideband Code Domain Multiple Access)) en banda pareada (FDD (Frequency Division Duplex)), y 20+15MHz para acceso CDMA por división temporal (TD-CDMA (Time Divixion Code Division Multiple Access)) en la banda no pareada (TDD (Time Division Duplex)).

UMTS requiere que todos los usuarios lleguen a la estación base con la misma potencia para poder maximizar la calidad. Así pues, utiliza un control de potencia en el que tanto el dispositivo móvil como la estación base toman medidas, de forma que la estación base puede hacer que todos los equipo de usuario conectados a ella reduzcan su nivel de potencia para poder ofrecer la mayor calidad posible. Como además, para conseguir mayores tasas de transmisión, se ha de reducir la codificación, se requiere del envío de mayores potencias que contrarresten la reducción de ganancia de código, por lo que esta tecnología va a tener un consumo energético considerable, haciendo que la duración de las baterías se vea mermada.

Si se tienen en cuenta el control de potencia y el uso de CDMA en UMTS, el reúso de frecuencias será total, disponiendo de todas las frecuencias en todas las estaciones base, por lo que la cantidad de recursos disponibles aumenta considerablemente. Por otra parte, como con el aumento del número de usuarios en un área determinada se reducen los recursos o la calidad de servicio asignado a cada usuario para mantener un nivel de errores aceptable, la probabilidad de bloqueo de canal es nula.

A pesar de ser una tecnología relativamente joven, el nivel de penetración en el mercado, debido a la gran aceptación obtenida en la sociedad, es elevado, por lo que la cobertura garantizada por las compañías telefónicas abarca casi por completo el territorio nacional. Además, los chips UMTS suelen implementar la capa de seguridad en la transmisión (TLS), que permite el encriptado de los datos, proveyendo así de seguridad y privacidad a la comunicación.

Bluetooth

Bluetooth es un protocolo de transmisión empleado en redes inalámbricas privadas que trabaja en la banda de frecuencias de 2.4GHz, y hace uso de espectro ensanchado por salto de frecuencias, para evitar el deterioro de la señal. Su principal objetivo es el servir de estándar para comunicaciones de muy corta distancia (máximo unos 100 metros), ya que ha sido ideado para tener un reducido consumo energético y un bajo coste. Este tipo de sistemas son capaces de cursar tasas de datos de hasta 3000kbps, ya que se emplea principalmente en aplicaciones en tiempo real.

El uso de Bluetooth permite únicamente la creación de redes con topología en estrella, como se muestra en la figura 14, ya que para que el sistema funcione debe existir un maestro que sirva como referencia de sincronismo, lo que limita mucho las posibilidades de éstos sistemas para el tipo de aplicación que se va a desarrollar. Además, permite un máximo de 8 dispositivos, lo que acota mucho la escalabilidad del sistema.

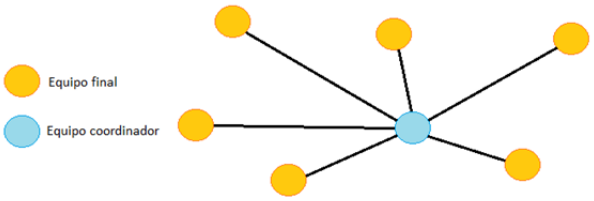


Figura 14: Bluetooth únicamente permite la implantación de topologías de tipo estrella.

Es destacable también que Bluetooth cuenta con encriptación de datos, y requiere de autenticación tras el establecimiento de la conexión, para el inicio de la transmisión de los datos, de forma que se maximiza la protección de los datos en dicho enlace. Además, considera que el canal físico no es fiable, por lo que realiza corrección de errores hacia adelante y comprueba la integridad de cabeceras y CRC (Cyclic Redundancy Check), de forma que provee de un servicio con gran integridad en los datos.

ZigBee

El estándar ZigBee define un conjunto de protocolos de alto nivel, que son utilizados sobre los protocolos de bajo nivel definidos por el estándar IEEE 802.15.4. Este último estándar, define la capa física y el nivel de control de acceso al medio (MAC) de sistemas de comunicación inalámbrica privados, cuyas tasas de transmisión son bajas. Trabaja en las bandas de frecuencia de 868MHz y 2.4GHz obteniendo tasas de transmisión de entre 100 y 250kbps. Puesto que no dispone de canales de control, como ocurría con las tecnologías móviles, para controlar el acceso al medio implementa CSMA/CA, evitando así las colisiones en el medio.

Sobre 802.15.4 se implementa ZigBee, que define una serie de protocolos de alto nivel para comunicaciones privadas con bajas tasas de transmisión. Permite crear redes de equipos con distintas topologías (figura 15), entre las que se encuentran la topología de malla y la topología en estrella, de hasta 65535 dispositivos repartidos en 255 subredes, de forma segura, gracias a que los módulos comerciales permiten la encriptación de la comunicación, y con un consumo bajo de energía. Dichas redes, se componen de diferentes tipos de equipos, como son los equipos coordinadores, los equipos finales, y los routers ZigBee, que permiten amoldar la red a las necesidades y espacios requeridos. De éstos, se podría decir que los equipos finales son dispositivos de funcionalidad reducida, y los equipos coordinadores y routers ZigBee son equipos de funcionalidad completa, ya que requieren poder interconectar con otros nodos.

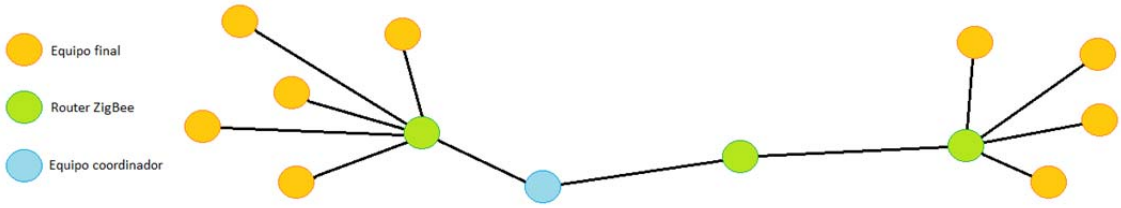


Figura 15: Una posible red de equipos interconectados mediante ZigBee.

Al tratarse de un protocolo cuyo objetivo es el del establecimiento de redes de equipos de bajo consumo, los alcances máximos de los módulos ZigBee son bastante pequeños, de unos 1500 metros como máximo en espacios abiertos y con línea de visión directa, aunque para su implementación en entornos más complejos, esto no es una limitación demasiado importante gracias a que, como se ha comentado anteriormente, permite la conexión de gran cantidad de dispositivos.

TCP (Transmission Control Protocol)

TCP es un protocolo de nivel de transporte caracterizado por ser orientado a conexión y fiable, por lo que al principio de la conexión se inicia una sesión de datos. En dicha sesión, todos los paquetes de datos llegan en orden, y se verifica la corrección de los datos recibidos. De este modo, garantiza que los datos llegan y que no llegan corruptos, por lo que añade fiabilidad entre aplicaciones del sistema de comunicación. Implementa un sistema de aceptaciones, con el que avisa al emisor de si los datos que llegan son correctos y están ordenados o no. Cuenta con un mecanismo de control de flujo de ventana deslizante, que controla cada cuántos paquetes correctos se envía una aceptación. Además, los números de secuencia de los paquetes TCP, garantizan que nadie se entromete en la comunicación, enviando paquetes falsos a ninguna de las partes.

Una de las limitaciones más importantes que posee este sistema, es que la inclusión del sistema de ACK's y el establecimiento/fin de la conexión, añaden una cantidad fija de tráfico independiente del tamaño de la comunicación, aunque el mecanismo de ventana deslizante reduce ligeramente esta sobrecarga. Además, contribuyendo al incremento de bytes a transmitir, hay que añadir que a cada paquete de datos se le adhieren al menos 20 bytes de cabeceras, y también, en caso de que la longitud de las cabeceras del paquete formado no sea múltiplo de 32 bits (4 bytes), se añade un relleno extra para que se cumpla esta condición. Así pues, la eficiencia de la conexión se ve reducida por estas sobrecargas, sobretodo en caso de tener pequeñas cantidades de tráfico.

UDP (User Datagram Protocol)

UDP es un protocolo de nivel de transporte caracterizado por ser no orientado a conexión y no fiable, por lo que al no iniciarse sesión, no sobrecarga el sistema de tráfico. De esta forma, los datagramas pueden emplear caminos diferentes, pudiendo llegar desordenados, por lo que su ordenación se establece en las cabeceras, cuya longitud será de por lo menos 20 bytes.

El único control de errores que se realiza, se da sobre las cabeceras, de forma que si una cabecera está corrupta, simplemente se descarta el datagrama entero, por lo que no necesita de mecanismo de aceptación, pero puede perder datos.

Debido al poco tráfico que añade, UDP permite un gran número de conexiones a un mismo servidor. Además, introduce poco retardo, por lo que resulta un protocolo muy adecuado para aplicaciones en las que la comunicación deba ser en tiempo real, y no importe que los paquetes puedan llegar corruptos. Por otra parte, el único control de flujo que implementa este protocolo es una limitación a una cantidad fija de los bytes que puede enviar el transmisor, a lo que el receptor se adapta reservando recursos según vea lo que va recibiendo, por lo que las conexiones UDP prevalecen sobre las demás.

HTTP/HTTPS (Hiper-Text Transfer Protocol (Secure))

El protocolo HTTP es el mayoritariamente utilizado para las conexiones con servidores web. Es un protocolo flexible que permite el intercambio de información de una manera eficiente. Dicha flexibilidad se la aporta en gran medida las cabeceras de dicho protocolo, que permiten definir gran cantidad de parámetros, así como los métodos que define, que permiten realizar diferentes acciones. Una petición HTTP debe contener, tanto el tipo de petición (método), como las cabeceras que permitan al servidor procesar dicha petición.

Este protocolo, inicia una conexión al puerto 80 del servidor, en la que envía datos y parámetros, además de recibir la información que el servidor le remita. Entre esta información remitida, aparecen también ciertas cabeceras especiales, que indican si la petición tiene un formato correcto, los recursos invocados existen, o incluso si se tiene permiso de acceso dichos recursos. Es por esto por lo que se podría decir que implementa algo parecido a un mecanismo de aceptaciones. En este caso, las respuestas no las procesa el protocolo, si no el propio cliente, lo que permite un gran control de la situación, a costa de incrementar ligeramente la complejidad del sistema.

El principal problema que conlleva el hacer uso de este protocolo, es su completa falta de seguridad, pues ni requiere de autenticación, ni encripta los datos, por lo que cualquier tercero podría corromper la comunicación. Para resolver esto, se inserta una capa extra entre las de aplicación y transporte, conocida como TLS (Transport Layer Security), encargada de la autenticación, y de la encriptación de los datos provenientes del nivel de aplicación y la desencriptación de los recibidos del nivel de transporte. De esta manera, se tiene una comunicación con autenticación y encriptada, haciendo uso de una capa invisible para el nivel de aplicación. Esta capa, añade una cantidad de tráfico al inicio de las conexiones, puesto que requiere comunicar las dos entidades de nivel TLS involucradas, para poder así acordar los parámetros de encriptación, por lo que el uso de esta capa incrementa el tráfico cursado.

Así pues, haciendo uso del protocolo HTTP sobre esta capa, se genera el protocolo HTTPS, agregando así seguridad e integridad a la comunicación.

FTP/FTPS ((Secure) File Transfer Protocol)

FTP es un protocolo establecido para la transmisión de archivos completos, como su propio nombre indica. Está basado en la arquitectura cliente-servidor, es decir, el cliente se conecta al servidor, lanzándole comandos para que éste realice unas u otras tareas. Este protocolo requiere de dos conexiones a puertos diferentes, de forma que hacia uno de estos puertos, el cliente inicia una conexión de control, mientras que hacia el otro, el cliente inicia una conexión de datos.

Destacable dentro de este protocolo es la necesidad de autenticación del remitente, lo que permite asegurar algo la conexión. No obstante, los datos no se encriptan, por lo que tanto datos como usuario y contraseña, están expuestos a una posible lectura de terceros que pretendan escuchar la conexión. Esto se resuelve con la inclusión de la capa de seguridad (TLS) expuesta anteriormente, lo que da lugar al protocolo SFTP (Secure File Transfer Protocol).

Anexo 2:

Protocolo HTTP

En este anexo se describe el estudio del protocolo HTTP realizado para encontrar el modelo de petición óptimo que resuelva las necesidades de este caso concreto, y además obtener el modelo de las respuestas transmitidas por dicho protocolo, para poder analizarlas y tratarlas correctamente en el equipo remoto.

El protocolo *HTTP* trata de realizar peticiones, mediante distintos métodos según el objetivo de la petición, las cuales deben llevar ciertas cabeceras que permiten que el receptor del mensaje sepa descifrarlo. Para el caso concreto de una petición mediante el método *GET*, que no permite introducir datos en dicha petición ya que es usado para obtener datos de una web o la propia web [13], puede no hacer falta el uso de ninguna cabecera, aunque también puede ser que se requiera de las cabeceras *"Host"* y *"User-Agent"* que definan, el host en el que se encuentra la página en caso de hacer uso de hosts virtuales, y el usuario que manda la petición. Por otra parte, el método *POST*, utilizado para enviar datos a la web, es imprescindible la aparición de la cabecera *"Content-Length"*, que marca el tamaño del cuerpo de la petición donde van los datos, así como la cabecera *"Content-Type"*, que marca el tipo de datos enviados para que puedan ser procesados correctamente, y la cabecera "*Host*" ya expuesta. Además, también es recomendable el envío de la cabecera *"Connection"*, que indica al servidor si la conexión se va a mantener iniciada, o tras la respuesta hay que cerrarla. Así pues, si por ejemplo se quiere obtener la página web principal de *Google España*, habría que conectarse vía *TCP* a la dirección *www.google.es* y, para pedir la dirección *http://www.google.es*, lanzar la siguiente petición:

```
GET / HTTP/1.0

```

También resulta imprescindible el salto de línea de la segunda línea, la cual ésta en blanco, pues es el que marca el final de las cabeceras, además de ser necesario marcar la versión de *HTTP* utilizada, en este caso con *"HTTP/1.0"*.

Como ejemplo de una petición *POST*, se puede exponer la utilizada para realizar el envío de *Dato1* con valor 22 y *Dato2* con valor 22, a la plataforma del servidor comercial *Exosite*, que sería:

```
POST /onep:v1/stack/alias HTTP/1.1
Host: m2.exosite.com
X-Exosite-CIK: <CIK_personal_de_40_caracteres>
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Content-Length: 17

Dato1=22&Dato2=22
```

En este caso, se incorpora una nueva cabecera creada para el propio servidor, con la que se introduce el identificador personal que marca la tabla y el usuario al que pertenecen dichos datos. Por otra parte, la petición para el ingreso de datos en *Xively* utiliza el método *PUT*, cuyo cometido es crear entidades en el servidor en caso de que éstas no existan, o modificarlas en caso de que existan [9], por lo que se puede ver que en *Xively* se almacenan los datos como entidades independientes y asociadas a cierto dispositivo.

Además, y de cara al sistema final a implementar, se pueden unir en una misma petición dos peticiones diferentes, una con el método *GET* y otra con el método *POST*, por lo que se puede realizar el envío del *token* por medio de *GET*, y la fecha y los datos por medio de *POST*, quedando una petición como la que sigue:

```
POST /camino_script/script.php?token=<token> HTTP/1.1
Host: <IP_servidor>
Connection: close
Content-Type: <tipo_aplicacion>;
Content-Length: <longitud_cuerpo>

Dato1=22&Dato2=22&Dato3=22
```

En esta petición, se envía el *token* en el campo de nombre *<token>*, que se sitúa después de la *URL* de la web, separado de ésta por un signo *"?"*. Además, se observa también la cabecera *"Connection"*, que indica al servidor que, tras realizar el procesado de los datos, cierre la conexión.

Por otra parte, para realizar el envío del archivo, es necesario variar el cuerpo de la petición POST, además de incluir alguna cabecera más, como el tipo de aplicación a la que va destinado ("*Content: Type*"). Esto es debido a que el contenido del archivo se transmite literalmente en el cuerpo de la petición, por lo que se requiere de separadores que marquen dónde empieza cada sección de dicho cuerpo. Igualmente, se pueden mandar datos mediante GET, y campos normales del método POST, quedando, como ejemplo de esto, una petición como la que sigue:

```
POST /camino_script/script.php?token=<token> HTTP/1.1
Host: <IP_servidor>
Connection: close
Content-Type: multipart/form-data; boundary=<boundary>
Content-Length: <longitud_cuerpo>

--<boundary>
Content-Disposition: form-data; name="sensor"

<nombre_sensor>
--<boundary>
Content-Disposition: form-data; name="archivo"; filename=<nombre_archivo>
Content-Type: text/plain

<fecha>\t<dato1>\t<dato2>
<fecha>\t<dato1>\t<dato2>
--<boundary>--"
```

Como se puede observar, *<boundary>* será un código cuyo único propósito es el de delimitar la extensión de cada sección, y marcar principio y el final del cuerpo de la petición. Al ser un código que no viene fijado, se ha de incluir una cabecera que lo defina. También se puede ver que cambia el tipo de contenido, pues en este caso tiene varias partes (*"multipart"*), ya que tiene que generar, dentro del cuerpo de la petición, una sección propia para el archivo. Otra sección, la primera, es la que envía los datos normales del *POST*, que en este caso se usa para enviar el identificador del equipo transmisor, mientras que el *token* correspondiente a éste se envía por el método *GET*. Dentro

de cada sección, se incluye una cabecera (*"Content-Disposition"*), que simplemente sugiere un nombre para el archivo o la variable que se envía, indicando cómo se envía, que en este caso es como parte de un formulario de datos. Para concluir el análisis de este tipo de peticiones, comentar que la longitud que hay que apuntar en *"Content-length"* es el tamaño total del cuerpo de la petición. Por otra parte, para adaptar esta petición a los archivos que se generan en los equipos del GTF, se observa que en el archivo, en vez de separa los distintos valores con comas, como se haría en el formato CSV típico, se emplean tabulaciones (*'\t'*), de forma que así, dicho grupo no ha de modificar ningún código implicado en la generación o lectura de archivos.

Para probar esto, se han realizado pruebas haciendo uso de la característica de *Windows* llamada *telnet*, y del módulo *Ethernet* de *Arduino* [24]. El uso de este módulo y de *Arduino* viene justificado por su sencillez de programación, que permite automatizar el envío de repetidas peticiones, lo que puede servir para analizar el comportamiento del servidor, por si se pudiese quedar bloqueado tras alguna de ellas. Además, los módulos de comunicaciones que se van a emplear corresponden con los diseñados para este hardware.

Tras las primeras pruebas, se ha observado que las respuestas recibidas también poseen cabeceras, aunque distintas, y que dichas cabeceras dan información acerca del servidor y de la petición recibida. La primera línea de la respuesta, lleva un código de error, que en caso de ser el *"200 OK"*, indica que la petición recibida es válida y ha sido procesada, y en caso contrario marca el error ocurrido. Esto define lo que antes se ha comentado era algo parecido a un mecanismo de aceptaciones, al que se puede incorporar algo de información en el cuerpo de la respuesta. Un ejemplo de respuesta es el mostrado en la figura 16, donde se expone la respuesta recibida a la petición GET expuesta anteriormente.

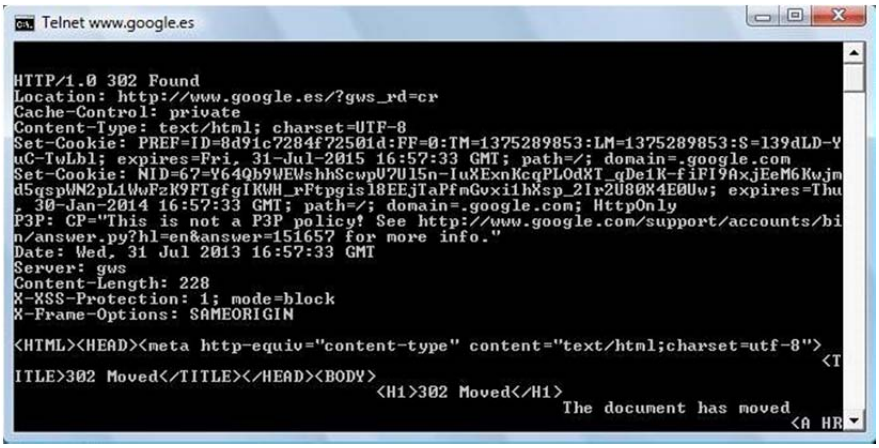


Figura 16: La figura muestra el principio de la respuesta obtenida a la petición GET anterior. El significado del error "302 FOUND" es que se ha localizado el elemento pedido, pero se redirecciona a otra página, lo que puede ser debido a que ahora Google implementa el protocolo HTTPS, y por lo tanto, redirecciona esta petición a https://www.google.es.

Anexo 3:

Montaje y configuración del servidor

En este anexo se profundiza más en los pasos seguidos para el montaje y la configuración del servidor, de forma que pueda servir como manual a quien se ocupe de su gestión futura dentro del GTF.

Para la instalación de *Apache* en el ordenador, se descarga de la página web oficial el paquete que no requiere de ejecutable, de tal forma que dicha carpeta se sitúa en el directorio deseado, y se instala *Apache* como servicio de *Windows*, mediante línea de comandos. A continuación, hay que realizar la configuración básica y la instalación del resto de módulos [20] [21]. Para instalar el complemento de creación de gráficas, sólo hay que descomprimir la carpeta descargada del sitio oficial [19] en el futuro directorio de la web.

El mayor inconveniente de los servidores no comerciales como *Apache*, es que están pensados para su uso sobre sistemas *Unix*. Por eso, para su securización, existen guías [22] pero están pensadas para estos sistemas y no para *Windows*. A pesar de esto, se realizan algunas acciones que permiten incrementar la seguridad del servidor.

La primera acción a llevar a cabo es denegar el acceso, por defecto, de *Apache* a todo el sistema, para permitir a continuación el acceso únicamente al directorio donde se situará la página web, pero sólo a cierto tipo de archivos, ya que es recomendable prohibir el acceso a archivos de configuración de usuarios y contraseñas. También se deberá limitar algunas opciones, como puede ser no permitir a *Apache* la resolución de enlaces simbólicos que pudieran permitirle acceder a zonas de memoria restringida. También es conveniente el evitar mostrar toda la información posible acerca del servidor, pues se estaría revelando información de las debilidades de éste, evitando también que el servidor muestre que posee *PHP* instalado, y su versión, de forma que además, se ahorrarán 25 bytes de tráfico en cada conexión al servidor. *Apache* no permite eliminar toda la información, ya que en las cabeceras de las respuestas siempre muestra que el servidor es *Apache*. También es recomendable desactivar, tanto la opción de listar ficheros en el directorio web, como de ejecutar los archivos "*.cgi*", además de deshabilitar todas las opciones en el resto de directorios. Hay que tener en cuenta que las peticiones tipo *TRACE* desvelan información del sistema, por lo que será recomendable deshabilitarlas, de forma que el servidor no las responda.

A continuación, también se puede realizar algún ajuste a la configuración de *PHP* que permita ganar algo de seguridad en el sistema. Por ejemplo, se le puede permitir el acceso únicamente al directorio que contiene los archivos de la web, además de permitir el acceso a las

cookies únicamente al protocolo *HTTP*, evitar que se abran archivos alojados en otros servidores, y activar el modo seguro de *PHP*.

Para evitar en la medida de lo posible ataques de denegación de servicio, es conveniente también limitar, tanto el tamaño de los archivos que puede admitir el servidor, como la cantidad de conexiones simultáneas que puede cursar, y el tiempo máximo dedicado a cada petición.

Con respecto a la gestión de los datos, es necesario asegurarse de no mostrar el código *PHP* de las webs en ningún momento, pero por si esto ocurriese, una buena práctica sería iniciar las conexiones a *MySQL* en archivos *PHP* fuera del directorio accesible por *Apache*, e incluirlo en el código de la página, de forma que dicho código se ejecute, pero nunca se pueda acceder a él. Otra protección más que se le puede incluir a la base de datos, es ocultarla tras el cortafuegos, ya que sólo cargaremos datos desde el propio servidor, por lo que no nos interesa que su puerto quede accesible desde el exterior. Por último, es recomendable crear un usuario de *MySQL*, únicamente con los privilegios necesarios para el envío de datos y la visualización de la web de gestión, de forma que un atacante no pueda borrar una base de datos a través de las propias páginas web. También se ha de configurar *PHP* para que destruya las *cookies* que generan la sesión de navegación, cuando se cierre el navegador, evitando que ésta se pueda quedar abierta.

Con respecto a la propia página web, es recomendable el uso de la directiva meta "*robots*" denegando la indexación y el seguimiento de enlaces, que evita que los robots buscadores, como el de Google, puedan listar la web y sus enlaces, ya que en este caso concreto, es para un uso privado, y no interesa que terceros puedan conocerla o acceder a ella. Por otra parte, para evitar la *inserción SQL*, es recomendable escapar todos los caracteres especiales de *MySQL*. Además, las conexiones a las bases de datos desde *PHP*, es recomendable hacerlas con las funciones de la librería "*mysqli*", en vez de las de "*mysql*", pues las funciones de la primera implementan mejoras en la seguridad.

Anexo 4:

Creación de las webs

En este anexo, al igual que en el anexo del servidor, se pretende crear un pequeño manual explicativo de cómo se han programados las distintas web y las diferentes secciones de las mismas, para facilitar su comprensión al futuro gestor de las mismas.

Web de interacción con los dispositivos autónomos

Inicialmente, se habrá de tener en cuenta que se requiere de procesar los datos recibidos, para lo que primero habrán de ser extraídos de la petición. Esto se consigue gracias a la existencia de las variables preestablecidas de *PHP* como "*$_POST[´<etiqueta_dato>´]*", de las que se genera una por cada dato introducido en el cuerpo de la petición POST. Así pues, una petición POST cuyo cuerpo sea "*dato1=22&dato2=17*", genera las variables "*$_POST[´dato1´]*" y "*$_POST[´dato2´]*", con valores 22 y 17 respectivamente. A partir de esto, se pueden fragmentar fácilmente los datos, para poder construir la petición a *MySQL* que los almacene en la tabla de la base de datos correcta. Es necesaria la transmisión del identificador del sensor, que es el primer campo en el cuerpo de la petición, y que viene seguido de la fecha y la hora a la que han sido tomadas las medidas, tras los que se insertan las medidas. Por otra parte, como ya se ha comentado, para hacer la función de contraseña, se va a asignar un *token* para cada equipo, de forma que éste, se envía por medio de la petición GET, que se une a la petición POST en la que van los datos. Para la recuperación de ese *token* en el *script*, al igual que ocurre con las peticiones POST, existe una variable preestablecida "*$_GET[‘<etiqueta_dato>’]*", cuyo contenido hay que comparar con el *token* almacenado en la tabla de identificadores de equipos.

Con respecto a las peticiones que se realizan a la base de datos, se pueden destacar dos, que son las más usadas en esta aplicación. La primera, para realizar el envío de los datos a una tabla de una base de datos concreta, debe ser del estilo de:

```
INSERT INTO <base_datos>.<tabla> (fecha, dato1, dato2) VALUES ('<fecha>', '<dato1>', '<dato2>');
```

En ella, la parte *"(fecha, dato1, dato2)"* marca el orden en que se envían los datos (nombres de las columnas de la tabla), cuyos valores se enumeran tras la marca *"VALUES"*.

Si lo que se pretende es obtener datos de la misma, de un intervalo temporal y por orden de fechas crecientes, la petición sería:

```
SELECT * FROM <base_datos>.<tabla> WHERE fecha >= '<fecha_desde>' AND fecha <= '<fecha_hasta>' ORDER BY fecha ASC;
```

En este caso, aparece un asterisco, que se encarga de marcar que se quieren obtener todos los elementos completos que cumplan la condición del intervalo de fechas. Dichos elementos, son los correspondientes con las filas de la tabla de la base de datos. Por otra parte, la petición no tiene por qué especificar ningún intervalo temporal, eliminándose en ese caso la parte *"WHERE fecha >= '<fecha_desde>' AND fecha <= '<fecha_hasta>'"*, ni es necesario que se descarguen ordenados, eliminándose la parte *"ORDER BY fecha ASC"*.

Con respecto a las peticiones a *MySQL*, hay que tener en cuenta que tras hacer una petición, no se cierra la conexión con este servidor de bases de datos, por lo que en caso de querer hacer otra petición a la misma base de datos, no hay que iniciar una nueva conexión. Igualmente, por seguridad, es recomendable cerrar todas las conexiones iniciadas con el servidor de bases de datos antes de terminar el *script*.

El esquema a seguir en el *script* de recepción de las medidas sería el mostrado en la figura 17, donde se puede ver que, tras comprobar que han sido enviados datos, habrá que autenticar el sensor con su identificador y su *token*.

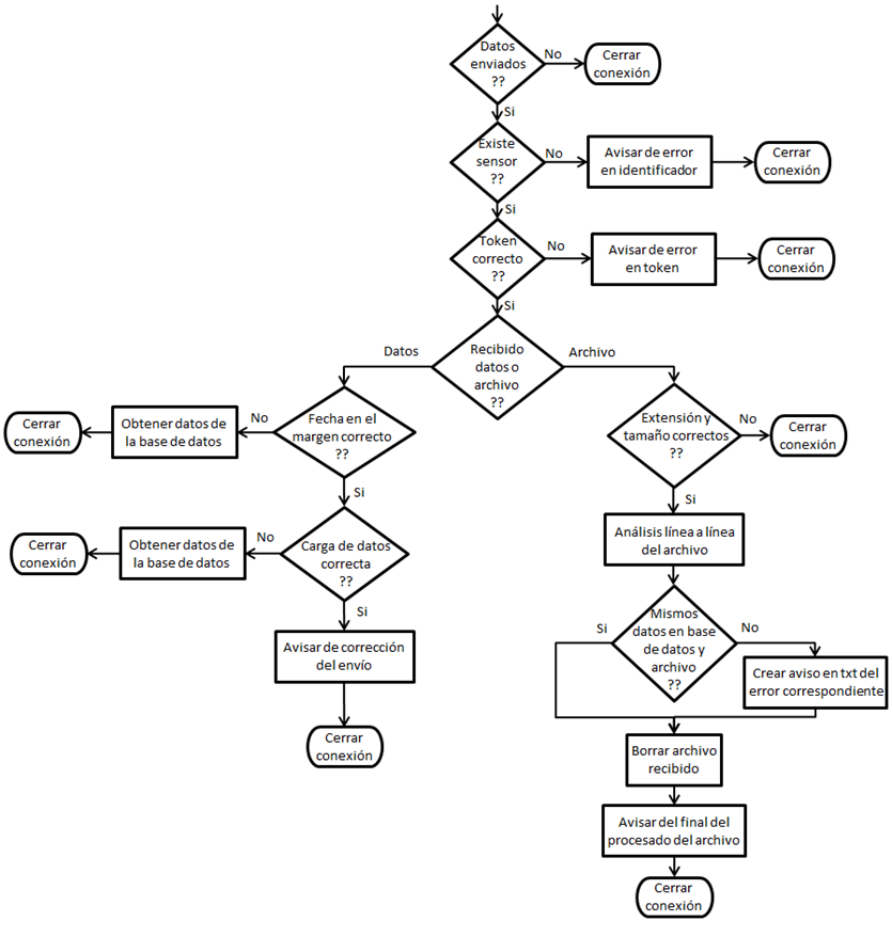


Figura 17: Esquema del script al que se envían los datos para que los procese y almacene en la base de datos.

Como se observa, el tratamiento recibido por los datos en caso de recibirse dentro de un archivo o recibirse aislados (dentro del cuerpo de una petición) es diferente, de forma que si se reciben aislados, habrá que asegurarse de que son de un intervalo de tiempo razonable, es decir, que son posteriores al último dato recibido por ese sensor, por lo que habrá que procesar y comparar las fechas de la base de datos con la fecha del envío. Para obtener la fecha del último envío registrado

en la base de datos, se realiza a *MySQL* una petición que es variación de la de obtención de datos, quedando una petición como ésta:

```
SELECT fecha FROM <tabla> ORDER BY fecha DESC LIMIT 1;
```

Ésta quiere decir que se seleccionan sólo las fechas, que en este caso es lo único que interesa, en orden de fechas descendentes, y tomando como máximo una, sólo la primera. También se podría revisar que no fuese demasiado posterior al último dato, pero eso podría resultar contraproducente en caso de que apareciese algún error que evitase el envío de alguna de las medidas.

Si los datos llegan dentro de un archivo, al ser tomados como comprobación de los datos enviados ese día, se comprobará que todos están en la base de datos, y que todos los que están en la base de datos están también en el archivo. En caso de que esto no fuese así, se anotaría en un archivo de texto, que se muestra en su correspondiente sección de la web. Además, para tener organizados los errores aparecidos y facilitar su análisis, éstos se cargan en archivos diferentes, dependiendo si son debidos a una medida que aparece en el archivo y no en la base de datos, o viceversa. Por otra parte, se puede realizar la comprobación de que la extensión del archivo es la correcta, y que el archivo no es de un tamaño excesivo, para evitar recibir peticiones de atacantes que traten de sobrecargar el servidor.

Web de interacción con los usuarios

La estructura de la página de gestión y control es la mostrada en la figura 18:

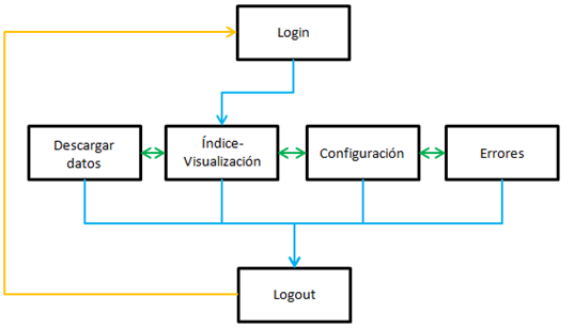


Figura 18: Esquema general de la web a crear. Los enlaces azules se siguen en sentido descendente, los verdes unen todas las secciones de un mismo nivel, y los naranjas se siguen en sentido ascendente.

Todas las secciones, en caso de no haber iniciado sesión previamente, redireccionan al usuario hacia la página de *login*. Tras el *loggeo* del usuario, se accede a una web que dispone de diferentes secciones, desde las que se puede también ir a la página de *logout*, que es la que destruye la sesión.

Así pues, como se ha comentado, para restringir el acceso a la página principal y las secciones de gestión y control de datos, se ha de implementar una página encargada de *loggear* a los visitantes, asegurándose de que son usuarios permitidos. A esta web, además, son redireccionados todos aquellos usuarios que traten de acceder a cualquier sección sin haber iniciado sesión.

En ella, inicialmente se revisa si el usuario ya ha rellenado o no el formulario de ingreso, y en caso de no haberlo hecho, lo saca por pantalla. Tras rellenarse dicho formulario, al igual que ocurre

con el resto de selecciones y formularios de las páginas, se envían los datos de inscripción al mismo *script* que se estaba ejecutando, de forma que en este caso, se comienza con el análisis de dichos datos. Si sólo se hubiesen enviado a medias, el *script* direccionaría a una web que imprimiría por pantalla el formulario de entrada de datos y un aviso de que algún campo estaba vacío.

En caso de que todos los datos se hayan enviado, como se ha comentado, se procesan, comenzando por escapar los caracteres peligrosos para las bases de datos, para evitar que a una petición que se lance a *MySQL* lleguen caracteres que puedan permitir un ataque que exponga la base de datos. Esto se puede llevar a cabo con la función "*mysqli_real_escape_string*".

Tras esto, se puede pedir a la base de datos los datos de todos aquellos usuarios cuyo nombre de usuario coincida con el de quien trata de acceder, para así, tras codificar la contraseña proveniente del formulario, compararla con las de los usuarios tomados de la base de datos. En caso de reconocimiento afirmativo, se genera una *cookie* con el usuario y contraseña utilizados para acceder, que se almacena en el equipo del usuario, de forma que se le pueda identificar cada vez que trata de acceder a una página o sección de la web. En caso de que dicha contraseña no coincida con ninguna de las traídas de la base de datos, o que el usuario no exista, se le direcciona a un *script* que imprime por pantalla el formulario de entrada de datos y un aviso de que al menos un dato de los introducidos es erróneo.

Por otra parte, en el *script* que se encarga de cerrar la sesión iniciada, la acción que realiza es cambiar el dato de fecha de la *cookie* por una fecha antigua, que haga que dicha *cookie* esté caducada.

Una vez se tienen los datos en la base de datos y se puede acceder de forma controlada, se puede proceder a su descarga, para lo que primero se deben obtener de la base de datos correspondiente, con la petición que se ha descrito anteriormente para la obtención de las medidas. Para cualquier descarga, es necesario decirle al navegador que lo que viene a continuación no es una página web, sino que es un archivo que debe guardar con cierto nombre y cierta extensión, y a continuación mandarle directamente el archivo.

En esta sección de la web, así como ocurre en casi todas las demás, se genera un menú donde elegir el sensor del que se quieren obtener datos. Este menú se genera dinámicamente, pues los sensores instalados pueden crecer en número, y hacerse complicado y tedioso el reprogramar todos los menús de las distintas secciones de la web. Para ello, se hace uso de la petición a la base de datos "*SHOW TABLES;*", que muestra los nombres de las tablas que se encuentran dentro de una base de datos. Puesto que los nombres de las tablas coinciden con los identificadores de los equipos, se listan los nombres de todas las tablas, permitiendo así su selección.

También se ha de tener en cuenta que no todos los sensores envían el mismo tipo de información, por lo que también se puede automatizar la obtención de ésta de la base de datos, haciendo uso de la petición:

```
DESCRIBE '<nombre_tabla>';
```

Esta petición devuelve la composición de la tabla, de forma que permite conocer de cuántas columnas se compone, y las etiquetas de las mismas, para así poder generar las cadenas de datos requeridas para las visualizaciones y descargas de forma más genérica y con menos código.

Para enviar los datos al usuario, se crea un archivo de *Excel* que se envía al navegador. Hay que tener en cuenta, que dependiendo de la configuración regional del ordenador, el separador decimal puede ser punto o coma, por lo que se implementa un selector en la web, antes de realizar la descarga, de forma que el usuario pueda elegir dicho separador, y así se representen los datos en *Excel* correctamente. Así, volviendo a la descarga de los datos, de lo que se trata es de generar una cadena de datos que simule una tabla que contenga los datos a descargar, por lo que hay que introducir etiquetas que marquen el inicio y el final de la tabla, así como delimitar las filas y las columnas. Un ejemplo sencillo de la estructura se muestra a continuación:

```
<table>
<tr>
<th>Cosa1</th><th>Cosa2</th>
</tr>
<tr>
<th>16</th><th>15,24</th>
</tr>
</table>
```

Se observa cómo la tabla viene delimitada por las etiquetas *<table>* y *</table>*, mientras que las columnas se marcan con *<th>* y *</th>*, y las filas con *<tr>* y *</tr>*. Por otra parte, las cabeceras que hay que enviar son:

```
Content-type: application/vnd.ms-excel
Content-Disposition: attachment; filename=<nombre_archivo>.xls
```

La primera, indica al explorador que lo que va a recibir es un archivo de *Excel* y no una página web, y la segunda le indica que debe recogerlo y guardarlo con el nombre *<nombre_archivo>* y la extensión *".xls"*.

Finalmente, como todo lo que se introduzca a continuación en la web va a ser tomado como parte del archivo, simplemente se cierra el *script*.

También se puede querer únicamente repasar la evolución de las medidas sin necesidad de descargarlas, por lo que otra opción que se incluye en la web es la de visualizar los datos graficados. Esta es la opción a la que se entra por defecto al acceder a la página. Para visualizar los datos, primeramente se obtienen de la misma forma que se obtienen para la descarga, sólo que en este caso, se tiene que crear una cadena diferente con los datos, ya que el formato de los datos introducidos en las funciones de creación de las gráficas es diferente del usado para la creación del *Excel*. En este caso, el formato de la cadena debe ser como el siguiente:

```
<chart>
  <categories>
    <category label='dia1'/>
    <category label='dia2'/>
  </categories>
  <dataset seriesName='<nombre_serie>' parentYAxis='P'>
    <set value='13'/>
    <set value='39.7'/>
  </dataset>
</chart>
```

En este caso, las etiquetas *<chart>* y *</chart>* delimitan el conjunto de datos utilizados para generar el gráfico, mientras que aquí, la separación de las dos columnas, puesto que se identifican con cada uno de los ejes, está muy definida, marcándose el eje X de la gráfica con las etiquetas *<categories>* y *</categories>* y los ejes verticales con *<dataset>* y *</dataset>*. El atributo *parentYAxis*

marca si dichos datos son representados con respecto a la escala marcada en el eje primario (P) o en el secundario (S).

Así, ya se tienen los datos preparados para su representación, lo cual se lleva a cabo con la siguiente instrucción:

```
echo renderChart("<path>", "", <datos>, "<nombre_grafica>", <anchura_pixels>, <altura_pixels>, false, true);
```

"echo" envía tal cual lo que le sigue al código *HTML* que forma la página web, de modo que, en el navegador, *Javascript* interpreta esta función y la ejecuta, pues anteriormente ha sido declarada en la cabecera de la web. Por otra parte, en *<path>* se introduce la dirección de los *scripts* que contienen las funciones, en *<datos>* se introduce la variable que contiene la cadena de datos, mientras que *<altura/anchura_pixels>* marcan el tamaño de la gráfica, *false* indica que no está activo el modo de depuración, y *true* se pone ya que es una opción por defecto que no está activa. El campo en blanco (*""*) es el campo en el que se introduciría la *URL* de la que se obtendrían los datos en formato *XML* (eXtensible Markup Language) o *JSON* (*JavaScript* Object Notation) si éstos se tomasen desde otra página web, en vez de introducirse desde una cadena.

Como se ha comentado anteriormente, también cabe la posibilidad de llevar un control de los errores aparecidos en la recepción de datos, almacenando éstos en archivos de texto. Mediante un bucle que recorre todos los archivos del directorio de errores, se abren los archivos y se imprimen por pantalla el nombre del archivo y su contenido. Para aplicaciones futuras con muchos sensores instalados, será necesario organizar este directorio, pudiéndose generar una carpeta por sensor.

Tras esto, tan sólo queda la creación de la sección de configuración remota de los equipos. Dicha configuración se hará llegar al dispositivo en la respuesta del servidor a la petición de almacenamiento de datos. Como puede darse la situación de que únicamente se reconfigure alguno de los parámetros, se almacenan en una base de datos dichos parámetros, marcados con un *flag*, de forma que si dicho *flag* está a uno, se ha cambiado y no ha sido enviado al equipo todavía, por lo que se deberá hacer en la siguiente petición. Así, se consigue reducir la cantidad de datos cursados, a costa de un mayor coste de procesado en el dispositivo remoto, pues debe evaluar qué parámetros le llegan, para reconfigurarlos.

Con esto, se completa la programación de las funciones básicas de la página web, que garantiza un acceso a dicha web sólo a usuarios permitidos, protegiendo así el sistema, de cambios maliciosos de la configuración de los equipos, o accesos de extraños a los datos almacenados, además de evitar en gran medida el envío de datos por parte de terceros.

Además, se incluye en el mismo directorio, pero no accesible desde ninguna de estas páginas, un *script* que permite al sensor remoto obtener la fecha y la hora local del servidor, por si llegase a hacer falta para la configuración inicial del equipo.

Tras esto, se realizan una serie de pruebas para comprobar el funcionamiento del servidor en determinadas situaciones que generasen error en algún punto. Una de estas situaciones fue realizar un envío a una dirección inexistente, en cuyo caso se observó que responde con el código de error "*404: Not Found*". Otra fue el envío erróneo de alguna de las cabeceras, a lo que responde con un "*400: Bad Request*", mientras que si se introduce mal alguna etiqueta de datos, da por buena la petición con un código de error "*200: OK*", pero *PHP* devolverá un error en el cuerpo de la respuesta.

Anexo 5:

Estudio de tráfico generado

Como se ha comentado, en este anexo se va a realizar un estudio del tráfico generado, profundizando en los cálculos realizados. Estos cálculos se han llevado a cabo en una hoja de *Excel*, que se ha preparado para efectuar las simulaciones de tráfico expuestas.

Para la estimación del tráfico de datos que va a generar el equipo, ha de tenerse en cuenta que, debido a que las compañías telefónicas facturan tanto el tráfico de datos generado como el recibido, se ha de realizar el cálculo del total de datos teniendo en cuenta ambos tráficos. Asimismo, los datos que se toman son los de nivel de enlace, presumiblemente, pues las cabeceras de este nivel variarán en el transcurso de los datagramas IP por las diferentes redes (redes X.25, Ethernet…) (figura 19), por lo que es de suponer que el nivel de enlace sea en el que se realice el cálculo del consumo.

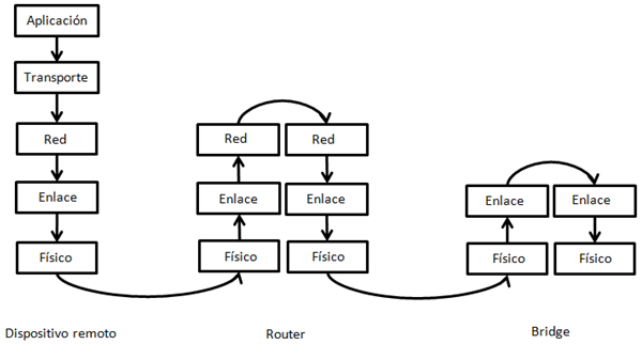


Figura 19: Niveles implementados en los distintos dispositivos de red. El dispositivo remoto irá generando datos de todos los niveles. Los routers, por sus propias características, deberán leer las cabeceras de nivel de red, mientras que los bridges tendrán que leer las cabeceras del nivel de enlace.

Como se observa en la figura 19, los routers trabajan a nivel de red, es decir, con los datagramas IP, por lo que como el primer elemento que tienen que atravesar éstos en la estación base es un router (el módulo SGSN de GPRS, como se ve en la figura 13 del anexo 1) que genera los datagramas que se introducen en la red, camino del destino, sean esos los datos que se contabilicen, y los que se tomen para la tarificación.

Así, se tendrán que contabilizar los datos de nivel de enlace, que incluirán los datos de nivel de red (protocolo IP) y sus cabeceras. Los datos de nivel de red estarán formados por los datos de nivel de transporte (protocolo TCP), correspondientes con los datos generados por la petición HTTP de nivel de aplicación, más las cabeceras de TCP. De todos estos datos, los únicos que se generarán directamente por parte del microprocesador que controla el equipo de medida, serán los datos de

nivel de transporte, por lo que se podrá apreciar que la eficiencia de la transmisión es menor del 100% debido al envío de las cabeceras de TCP y de IP, y a las aceptaciones y paquetes de inicio y final de la comunicación que incluye el protocolo TCP. No se tendrán en cuenta las posibles retransmisiones para el cálculo inicial, pero más adelante, se tomará como probabilidad de error la probabilidad de que alguno de los paquetes de datos o aceptaciones de paquetes de datos sean erróneos, suponiendo que nunca ocurren errores en los paquetes de inicio y final de la conexión, puesto que la cantidad de tráfico debido a éstos es poco significativa frente al total, y al ser paquetes muy pequeños, es más difícil que se den errores en ellos.

Analizando las cabeceras que aparecerán en los datos, que se muestran en la figura 20, como se ha comentado, se tendrán las cabeceras introducidas por el protocolo TCP y las cabeceras introducidas por IP. Las cabeceras de TCP contienen un campo llamado “Número de acuse de recibo” que permite llevar a cabo el *piggybacking*, que es el empleo de paquetes de datos para realizar aceptaciones que de otra forma irían sueltas, incrementando el tráfico en una cantidad igual a las cabeceras, de forma que se podría ahorra la aceptación anterior al envío de la respuesta del protocolo HTTP del servidor, que al darse en todas las comunicaciones supondrá un ahorro a tener en cuenta. Esto se puede simular fácilmente suponiendo que la respuesta del servidor es 40 bytes más corta de lo que realmente es, no teniéndose así que cambiar nada más que una variable, evitando complicar demasiado los cálculos. Por otra parte, el campo “opciones” no se empleará, por lo que en esta aplicación no se tendrá en cuenta en los cálculos, de forma que tanto la cabecera TCP como la cabecera IP tendrán una longitud de 20 bytes, que es múltiplo de 32 bits (4 bytes), así que al no necesitar relleno, esa será su longitud máxima. De esta forma, se tendrán 40 bytes fijos de cabeceras en cada paquete que se transmita.

Cabeceras del protocolo TCP

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Puerto Origen																Puerto Destino															
32	Nº de secuencia																															
64	Número de acuse de recibo (Número de aceptación) (ACK)																															
96	Longitud de la cabecera TCP				Reservado				Flags								Tamaño de la Ventana															
128	Suma de verificación (Checksum)																Puntero urgente															
160	Opciones + Relleno (Opcional)																															
224	Datos																															

Cabeceras del protocolo IP

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Versión				Tamaño de la cabecera				Tipo de servicio								Longitud total															
32	Identificador																Flags			Posición del fragmento											
64	TTL								Protocolo								Checksum de la cabecera															
96	Dirección IP origen																															
128	Dirección IP destino																															
160	Opciones + Relleno (Opcional)																															
224	Datos																															

Figura 20: Cabeceras de los protocolos TCP (arriba) e IP (abajo). Los campos de “Opciones + Relleno” no se tendrán en cuenta, pues no van a aparecer en la comunicación que se va a establecer.

Otra característica importante y que se ha de tener en cuenta, a pesar de su escasa influencia, es que cuando un paquete es fraccionado e insertado en diferentes datagramas IP, el tamaño de todos los datagramas, menos el del último, ha de ser múltiplo de 4 bytes, por lo que se añadirán bits de relleno hasta cumplir esta especificación. Si un paquete no requiere de ser fraccionado, será el último paquete de la serie, por lo que el datagrama del que forme parte tampoco tendrá por qué tener longitud múltiplo de 4. La longitud supuesta de los datos será de 8 bytes como máximo (3 de parte entera, el separador decimal y 4 de parte decimal), con una estructura del cuerpo de la petición como la mostrada en la memoria, por lo que el total del cuerpo

de una petición POST de datos tendrá una longitud de 222 bytes, suponiendo 12 datos en cada envío, y el total de cabeceras HTTP será de 182 bytes, para un total de tamaño de petición de 404 bytes. Con respecto a la petición POST de envío de archivo, con la estructura mostrada en la memoria, se tendrán 395 bytes de cabeceras, tomando un campo *boundary* de tamaño 10 bytes, separadores de secciones y principio y final del cuerpo de la petición, además del tamaño del archivo, que será de 12288 bytes, para un total de tamaño de petición de 12583 bytes.

Como el protocolo de nivel IP no cuenta con aceptaciones y teniendo en cuenta que los paquetes de nivel TCP son los datos de los datagramas de nivel IP, por lo general el diagrama de la transmisión será igual a nivel IP que a nivel TCP, cambiándose el significado de los paquetes, ya que en el nivel IP los datagramas no tendrán ningún significado especial, y será el mostrado en la figura 21:

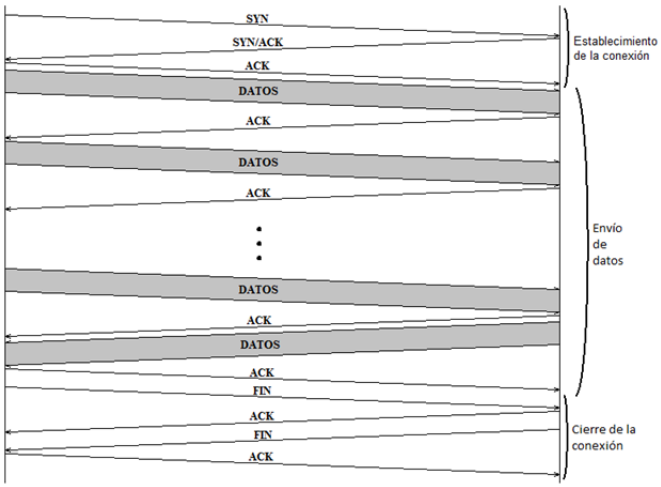


Figura 21: Esquema de una transmisión, en el nivel TCP, con tamaño de ventana deslizante de 1 paquete.

La respuesta del servidor, en caso de no tenerse que configurar ningún parámetro, será como la mostrada en la memoria del proyecto. Para realizar los cálculos de tráfico se supondrá que no se ha de reconfigurar el equipo, ya que la inmensa mayoría de las peticiones van a cumplir esta característica, por lo que la inclusión de unos cuantos bytes esporádicamente no va a variar en gran medida el tráfico, y se va a aproximar bastante bien al real. Aun así, se realizará alguna simulación también con reconfiguración en la respuesta, para así establecer cuánto podría afectar al tráfico si se llevase a cabo la configuración del equipo en cada transmisión, o si se enviase siempre y fuese el equipo el que se ocupase de ver si está bien o ha de cambiar algo.

Destacar también el papel de la Unidad Máxima de Transmisión (MTU), que es el mayor datagrama de nivel de red que se puede cursar, o lo que es lo mismo, la mayor cantidad de datos que caben en un paquete de nivel de enlace, y que varía entre tipos de redes y dependiendo de las condiciones de tráfico. Por este motivo, para la obtención de la mayor cantidad de datos de nivel de transporte que se pueden incluir en cada datagrama, se tendrán en cuenta las cabeceras de TCP y las de IP, pero no las de niveles inferiores.

Así, para las primeras estimaciones se supondrá que la ventana deslizante tiene un tamaño de 1 paquete, que la MTU es de 1500 bytes, como ocurre típicamente en las redes Ethernet, que además no aparecen errores, y que se realizan medidas y transmisiones cada 15 minutos, enviándose una vez al día el archivo con las medidas para la comprobación.

Con estos parámetros, el envío de datos a la web se comenzará con el establecimiento de la conexión, compuesto por un paquete SYN de nivel TCP, sin datos, es decir, únicamente contará con las cabeceras (40 bytes). A este paquete, el servidor contestará con una aceptación si tiene recursos disponibles, con la misma estructura que el paquete anterior, y en caso de no poder asignar todos los recursos pedidos, responderá con un paquete, a modo de corrección también sin datos, pero especificando los recursos asignados, de forma que se tendrá otro paquete también de 40 bytes. Para concluir con el inicio de sesión, el cliente responderá afirmativamente a la corrección o a la aceptación, enviando así otros 40 bytes. Es decir, en el inicio de sesión se consumirán 120 bytes fijos.

Tras esto, el cliente enviará el paquete de datos, que al ser de tamaño 444 bytes (404 bytes de paquete, formado por 182 bytes de cabeceras HTTP y 222 bytes de cuerpo, más 40 bytes de cabeceras), no se fraccionará nunca debido a que es menor que el MTU mínimo permitido, que es de 500 bytes. Así pues, y teniendo en cuenta que es un paquete único, no tendrá por qué tener una longitud múltiplo de 8 bytes, pudiéndose enviar tal cual. Cuando este paquete llegue al nivel de transporte del servidor, se generará una aceptación que no contendrá datos, por lo que su longitud será de nuevo de 40 bytes. En caso de emplearse el método de *piggybacking*, esta aceptación no se cursará, puesto que se enviará como parte de la cabecera TCP del paquete de datos de respuesta del servidor.

A continuación, se enviará la respuesta de la web, que como se ha comentado, formará un datagrama de 192 bytes (152 bytes + 40 bytes de cabeceras), por lo que tampoco se fraccionará. Esta respuesta también deberá ser aceptada, por lo que, como ocurría con la trama enviada al servidor, el cliente responderá con una aceptación sin datos, que tendrá una longitud de 40 bytes.

Como el servidor se encargará de cerrar la conexión, debido a la cabecera de la petición HTTP insertada, éste iniciará el proceso de desconexión enviando un paquete FIN de nivel TCP, que contará con las mismas características que el paquete SYN enviado por el cliente en el establecimiento (40 bytes). Este paquete deberá ser aceptado por el cliente (40 bytes), el cual, a continuación, enviará de vuelta otro paquete FIN (40 bytes), que será aceptado por el servidor (40 bytes), cerrándose así la conexión. De esta forma, el cierre de conexión contribuirá al tráfico con 160 bytes fijos.

Con esto, tendremos pues para esta conexión, que el total de tráfico cursado será de 996 bytes, y se realizará cada 15 minutos.

Analizando de la misma manera el envío del archivo de comprobación, se tiene que también se comenzará con el establecimiento de la conexión, que como se ha visto, supone 120 bytes.

A continuación viene la gran diferencia existente entre el envío de los datos y el del archivo. En este caso, el archivo será bastante grande, de forma que el tamaño del datagrama IP que generaría, superaría con creces el valor de MTU de la red supuesta. Con esta configuración, si en el envío del archivo se quiere obtener un único datagrama y que tenga un tamaño menor que 1500 bytes, se tendrían que tomar las medidas cada mucho más tiempo, aproximadamente cada 127 minutos, y así se obtendría un tráfico totalmente independiente del tamaño de ventana deslizante utilizado, lo que resulta interesante, sobre todo teniendo en cuenta que el tamaño de dicha ventana varía según las condiciones de tráfico de la red, y también de un tipo a otro de red. Continuando con el análisis del envío del archivo con la configuración que se estaba tratando, el tamaño del archivo es de 12288 bytes, con 395 bytes añadidos por la estructura de la petición HTTP, haciendo así un total

de datos TCP de 12683 bytes. El fraccionado generará 8 segmentos de 1456 bytes, sin contar las cabeceras y siendo múltiplo de 8 bytes, como se requería, y uno de 1035 bytes, también sin contar las cabeceras y que en este caso no será múltiplo de 8 bytes, al no ser necesario que sea así, que con las cabeceras TCP e IP correspondientes a cada segmento, hará un total de 13403 bytes transmitidos. Además, se podrán llegar a recibir en el equipo cliente hasta 9 aceptaciones, de 40 bytes cada una, correspondientes a los 9 paquetes TCP transmitidos, en caso de tener un tamaño de ventana deslizante de un paquete, y sin hacer uso del *piggybacking*. Si por el contrario, se hiciese uso un tamaño de ventana mayor que 8 paquetes y del *piggybacking*, se recibiría una única aceptación, que como iría en el paquete de datos de la respuesta del servidor, no generaría ningún tráfico extra.

A continuación, se enviará la respuesta de la web, que como se ha comentado, formará un datagrama de 192 bytes (152 bytes más 40 bytes de cabeceras), por lo que tampoco se fraccionará. De nuevo se aceptará la trama con un paquete de longitud 40 bytes, y a continuación se cerrará la conexión, agregando otra vez los 160 bytes de tráfico de cierre de conexión.

Así que en este caso, se tendrá un total de tráfico cursado de 13915 bytes, que se realizará una vez al día.

Así, al final del día, el tráfico cursado total será el equivalente a 24x4 veces el tráfico del envío de datos, más el tráfico del envío del archivo, haciendo un total de 109531 bytes, es decir, 106.96KB. Esto, al final de un mes de 31 días, hace un tráfico total de unos 3.24MB, que como se ve, es una cantidad muy pequeña, por lo que a la hora de contratar una tarifa con una compañía telefónica, se podrá buscar una que incluya una cantidad de tráfico de datos muy baja, reduciendo así el coste económico. Con respecto a la eficiencia de la comunicación, que se calculará tomando como datos válidos los daros de nivel TCP, es decir, cabecera y cuerpo de la petición HTTP y son contar la respuesta del servidor, aunque esta también posea datos válidos cuando se transmite una reconfiguración. De esta forma, se obtiene una eficiencia en la transmisión del archivo del 91.15%, mientras que la eficiencia de las transmisiones simples será del 40,56%, debido a que las cabeceras y el tráfico de establecimiento/cierre de la conexión suman una parte importante del tráfico total.

La tarifa elegida proporciona 100MB a la semana. Así pues, calculando el tráfico generado semanalmente, se tiene una cantidad de 748.75KB, por lo que con esos 100MB por semana contratados, se podrán conectar hasta 136 sensores con la misma tarjeta SIM, lo que se puede explotar instalando una red de sensores, y usando sólo uno de los dispositivos para recoger todos los datos y realizar la comunicación con el servidor.

La influencia del *piggybacking* será notable, debido a que se lleva a cabo en todos los envíos, ahorrando una aceptación en cada uno, de forma que se obtiene un ahorro de entre 26KB y 27KB por semana, permitiéndose así el uso de la tarjeta SIM para realizar las comunicaciones de unos 5 sensores más.

Con respecto al tamaño de la ventana deslizante, cabe destacar que no va a afectar casi nada, pues el fraccionado de los paquetes se da únicamente en el envío del archivo, y al ser éste de un tamaño elevado, el tráfico añadido por unos ACK's extra es ínfimo en comparación con el total de la comunicación, sobre todo teniendo en cuenta que la mayor carga de tráfico se genera en las conexiones para el envío de datos. Algo parecido ocurre con respecto a la MTU, ya que la reducción de ésta puede hacer que se incrementen el número de paquetes en los que fraccionar, añadiendo

tanto aceptaciones como cabeceras. Utilizando el valor mínimo de MTU permitido, que es 500, se podría como mucho triplicar el tráfico generado por ACK's y cabeceras, incrementando el tráfico semanal en unos 10KB.

Por otra parte, tomando como ejemplo el primer caso expuesto, un incremento en el tamaño de la respuesta del servidor puede llegar a incrementar mucho el tráfico, ya que ésta respuesta aparece en todas las comunicaciones. Lo mismo ocurre si se cambia la cantidad de datos que se deben enviar con cada sensor, ya que reduciéndola, se obtiene un ahorro mensual considerable.

Llegados a este punto, se puede simular la influencia del periodo de medida en el tráfico cursado por las conexiones. En la figura 22, se muestra ésta simulación, con una red que no genera errores, un MTU de 1500 bytes, tamaño de la ventana deslizante de 1 y envío de 12 datos.

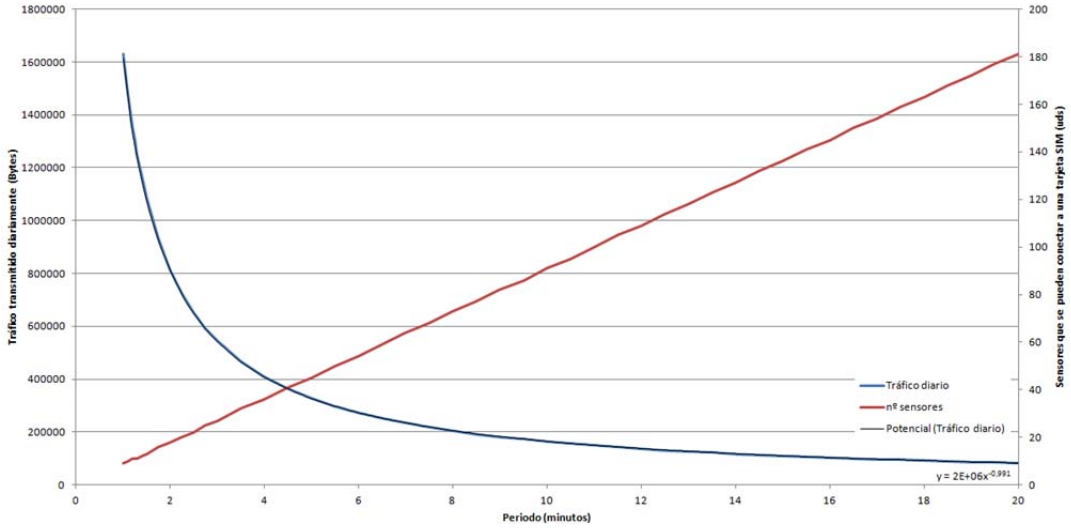


Figura 22: Dependencia con el periodo de medida, del tráfico y el número de sensores que permite una única tarjeta SIM.

En dicha figura, se observa que el tráfico generado tiene una relación inversa con el periodo de medida, pues quitando las cabeceras de los paquetes, las aceptaciones y el tráfico de establecimiento/cierre de la conexión, que supone una carga no excesivamente significativa, la cantidad de datos generados decrece con el crecimiento del periodo de medida. Igualmente, como el número de sensores es aproximadamente inversamente proporcional al tráfico generado, su relación con el periodo de medida es prácticamente directa, salvo para periodos muy pequeños, donde la sobrecarga introducida por el tráfico de control y las cabeceras resulta significativa.

Para terminar, se puede observar, en la figura 23, la dependencia del tráfico generado con la probabilidad de error producida por la red:

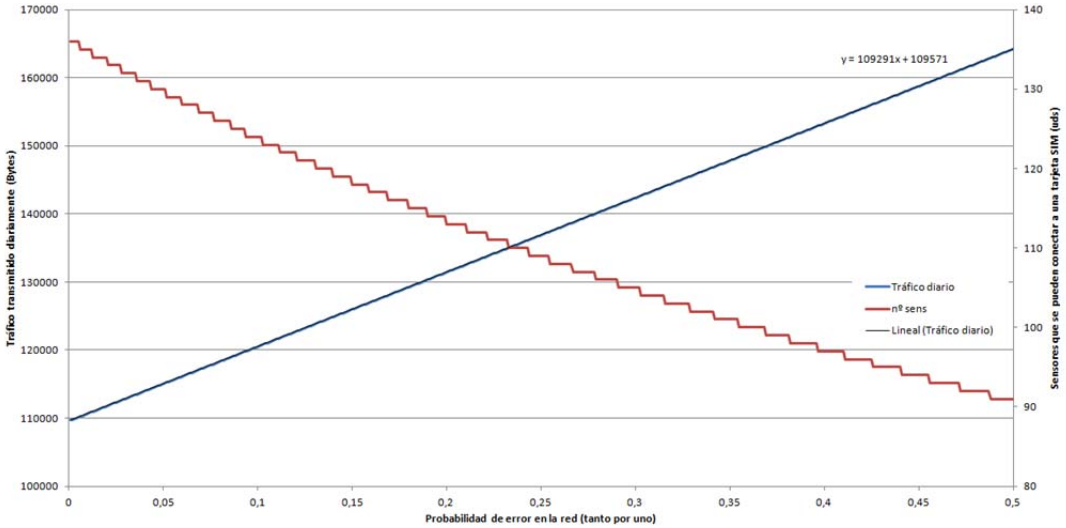


Figura 23: Dependencia con el periodo de medida, del tráfico y el número de sensores que permite una única tarjeta SIM.

En esta figura se ve cómo el tráfico generado sigue una relación casi directa con (1+p), que se correspondería con el caso de una probabilidad de error de (100*p)%, lo que ocurre debido a que, como se ha comentado anteriormente, se ha supuesto que tanto el establecimiento como el cierre de la conexión están carentes de errores.

Anexo 6:

Estudio de consumo energético

Para la realización de este estudio, se ha preparado una hoja de cálculo donde se establecerán los cálculos a realizar a partir del periodo de medida y de los datos obtenidos de las medidas realizadas. Por otra parte, debido a las diferencias de consumo según la acción que esté llevando a cabo el módulo en cada momento, se harán capturas del osciloscopio que permitan analizar en cada momento qué está ocurriendo y a qué es debido, para comprender mejor el comportamiento de los dispositivos GPRS.

La importancia de realizar un minucioso análisis del consumo del *módulo GPRS* radica principalmente en la variabilidad de dicho consumo y su alto nivel en algunas ocasiones, por lo que resulta interesante poder prever diferentes situaciones de consumo, debido a que se va a instalar en módulos que deben ser autónomos, y cuyas baterías deben mantener carga suficiente para dar servicio a todos los componentes. Por lo tanto, se caracterizará, tanto el consumo mientras se mantiene el módulo dormido, como el consumo cuando se apaga el módulo, para así poder determinar si resulta mejor opción dormirlo o apagarlo.

Para ello, se realizará un montaje en el cual, el *módulo GPRS* irá conectado a un *Arduino Uno rev3* de *Cooking Hacks* [24], que será usado como *gateway*, para que así se le puedan enviar los comandos al módulo directamente desde un puerto serie cualquiera. El *módulo GPRS* se alimentará con una fuente externa (*E3649A* de *Agilent*), para así poder medir la corriente de entrada en un osciloscopio (*MSO3034* de *Tektronix*) con una sonda de corriente (*TCP0030* de *Tektronix*), lo que determinará el consumo que se está dando en cada momento.

En las hojas de características del módulo se puede observar que el consumo puede llegar a valores de 2.2A durante los picos producidos por la transmisión, cuando la distancia a la estación base requiera dicha potencia.

Como ejemplo de algunas capturas del osciloscopio analizadas para obtener los consumos por tramos, se presentan a continuación las capturas correspondientes a la introducción del PIN (figura 24) y el apagado del módulo (figura 25).

Analizando el consumo de corriente generado al enviar el PIN (figura 24), se observa que al principio de la captura se tiene un consumo importante. Esto, presumiblemente, es debido a que al no tenerse todavía constancia de la operadora a la que se debe conectar, mantiene un análisis de todas las operadoras a las que puede acceder, para acelerar el proceso de negociación que se inicie con la introducción del PIN. Tras esto, se introduce el PIN, con lo que aparecen picos grandes de

consumo, debidos a transmisiones. Estas transmisiones son debidas al acceso del módulo, a la red del operador correspondiente a la tarjeta SIM que se acaba de activar. Finalmente, cuando ya ha completado toda la negociación de parámetros y configuración de los mismos en el módulo, se mantiene a la espera, con menor consumo, pues sólo debe atender a los slots por los que le llegan los canales de control de la operadora a la que ha accedido.

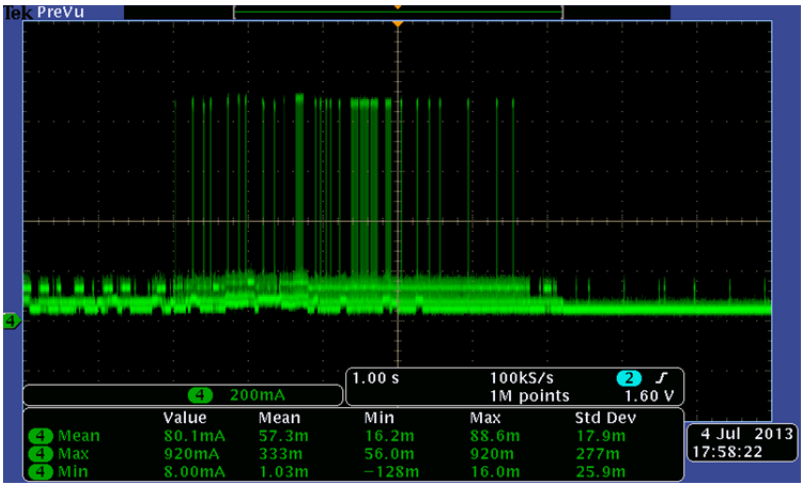


Figura 24: Captura en la que se representa el consumo previo a la introducción del PIN, el consumo durante la negociación de ingreso a la red, y el consumo en reposo tras esto.

Igualmente, si se analiza la captura correspondiente al apagado del módulo (figura 25), se puede ver que antes de apagarse, se tiene un periodo de reposo, en el que el módulo ya ha avisado a la red de que se va a apagar, hasta que recibe la confirmación y envía la respuesta "*OK*" por el puerto serie.

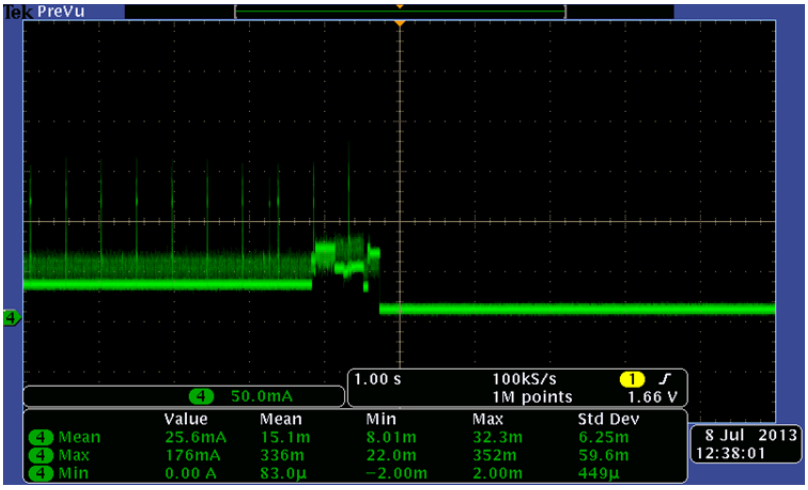


Figura 25: Captura en la que se observa el periodo de reposo anterior al apagado del módulo.

Falta por comentar que en el proceso de dormir/despertar del módulo, se observa que partiendo del módulo apagado, el despertado se hace en dos fases. En un principio, el módulo consume poco durante unos segundos, previsiblemente mientras activa las diferentes funcionalidades, hasta que por fin se despierta del todo, mientras que el dormido del módulo se lleva a cabo de golpe. En todas las fases, dormido, despierto y despertándose, sigue manteniendo el contacto con la estación base, como en reposo. Además, el dormido lo realiza cuando lleva unos segundos sin recibir comandos, mientras que para despertarse, necesita recibir algún comando por el puerto serie, que no será capaz de interpretar.

Destacar también que en la recepción de los datos, se obtienen también unos picos de consumo como en transmisión, pero sin llegar al nivel de estos otros, ya que el módulo sólo tiene que amplificar la señal recibida y decodificarla, y no hacerla llegar hasta la estación base.

Por otra parte, el consumo del módulo durante la recepción de un comando, es como el consumo aparecido justo antes del apagado definitivo del mismo en la figura 25, ligeramente superior al consumo en reposo.

De estas capturas, se pueden obtener los consumos medios que tiene cada parte de la conexión, obteniéndose los resultados de tiempos y consumos de corriente reflejados en la tabla 4:

Consumo apagado	4.7 mA
Consumo dormido	8.4 mA
Consumo despertándose	20 mA
Consumo encendido	34.4 mA
Consumo recibiendo un comando	50 mA
Consumo antes de introducir el PIN	74.9 mA
Consumo tras introducir el PIN	106.41 mA
Consumo transmitiendo datos	183.6 mA
Consumo recibiendo datos	146.88 mA
Tiempo que tarda en despertarse	1.4 segs.
Tiempo empleado por PIN	6 segs.
Tiempo empleado en transmitir	15 segs.
Tiempo empleado en recibir	8 segs.

Tabla 4: Resultados obtenidos en las simulaciones realizadas para el cálculo del consumo

Estos resultados se pueden reflejar en una simulación de una transmisión, de forma que se pueda obtener el consumo medio y un perfil de consumo de la transmisión. Dicho perfil de consumo se muestra en la figura 26, donde se pueden observar diferentes zonas de consumo:

- Zona A: Durante el proceso de encendido, mientras se configura el módulo y se activan todas las funcionalidades, tiene un consumo medio de unos 50mA.
- Zona B: A continuación, el sistema está encendido y leyendo la información de todos los operadores que le llegan desde la estación base, consumiendo unos 74.9mA de media.
- Zona C: Cuando se introduce el PIN, comienza un intercambio de información del módulo con la estación base, de forma que éste se suscribe a la red móvil del operador marcado por la tarjeta SIM, y obtiene los datos de la misma, consumiendo unos 106.41mA de media.
- Zona D: Una vez activado el acceso a la red, comienza el envío de comandos para configurar el acceso de datos antes de iniciar la transmisión. Por otra parte, la zona D que aparece tras la recepción (zona F), corresponde al envío del comando que apaga el módulo. En esta zona, se consumen unos 50mA de media mientras interpreta el comando y unos 34.4mA entre ellos.
- Zona E: Tras esto, se realiza la transmisión de los datos al servidor, por lo que se da el mayor consumo energético, consumiendo unos 183.6mA de media.
- Zona F: Una vez se ha realizado el envío, el servidor responde, y el módulo procesa los datos recibidos, consumiendo unos 146.88mA de media.
- Zona G: Tras la recepción de la orden de apagado, el módulo concluye su conexión a la red móvil del operador correspondiente, antes de apagarse, consumiendo unos 34.4mA de media.

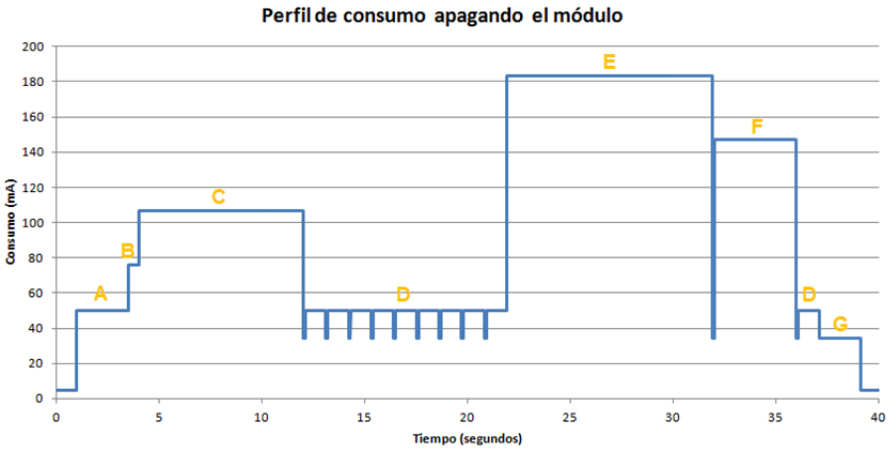


Figura 26: Perfil de consumo medio de una transmisión terminada apagando el módulo.

Con esto, se puede realizar un cálculo aproximado del consumo medio del módulo, llegando a que, si suponemos que se realiza un envío cada 15 minutos, se tiene un consumo de unos 10.86mA apagando el módulo, y unos 12.87mA si se duerme el módulo en vez de apagarlo. Así pues, en este caso se puede ver que, aunque la diferencia parece pequeña, pues es de 2mA, supone un ahorro de batería de más del 15%, por lo que parece lógico pensar que la mejor opción es apagar el módulo.

Por otra parte, como se puede ver en la figura 27, conforme se reduzca el periodo de envío, la mejora que supone el apagado del módulo frente a dormirlo se verá reducido, y llegado cierto punto, es preferible dormirlo, pues la configuración se mantiene, haciendo que el módulo tenga que estar menos tiempo encendido. Esta relación se puede observar en la figura 27, donde se observa que para periodos menores de 7 minutos, es preferible dormir el módulo manteniendo la configuración. Esto permite observar que si se usa el módulo como elemento final en una red de equipos remotos, como no se sabe cuándo va a tenerse la necesidad de establecer una comunicación, pudiendo ser periodos muy pequeños, resulta más conveniente dormir el módulo.

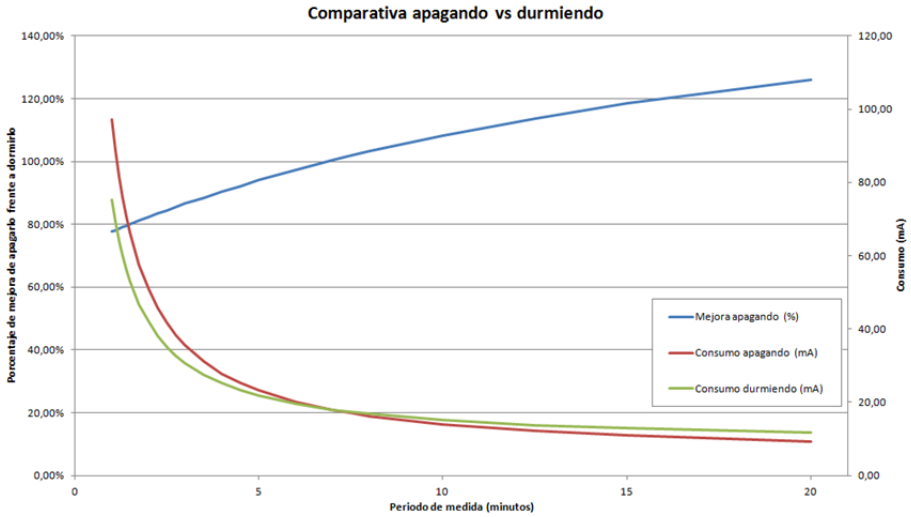


Figura 27: En el eje de la derecha, comparación del consumo medio apagando el módulo (rojo) frente al consumo medio durmiéndolo (verde); En el eje de la izquierda y en azul, el porcentaje de mejora que supone apagar el módulo

También se puede realizar este análisis manteniendo el módulo despierto, obteniendo mejor resultado al apagar el módulo siempre y cuando el periodo sea mayor que 1 minuto 35 segundos, pero siempre va a ser mejor dormir el dispositivo, pues en ambos casos se ahorrará reintroducir la

misma configuración. Del perfil de consumo de la transmisión se puede también llegar a la conclusión de que no tiene sentido reducir el periodo de medición por debajo de 40 segundos, pues en ese caso ya casi no se tiene tiempo para que el dispositivo se pueda apagar.

Como se estima que el consumo del módulo sigue siendo muy elevado, debido al consumo del regulador, se opta por cortar la alimentación cuando se apague el módulo reduciendo así dicho consumo. De esta forma, se obtiene un consumo medio, apagando el módulo, de 6.43mA.

Igualmente, como se comenta en la memoria, en las pruebas finales realizadas del sistema completo, se han optimizado los tiempos de espera entre comandos, solapando también la ejecución, por parte del módulo, de alguno de ellos, consiguiendo así reducir el tiempo que requiere estar encendido el módulo. De esta forma, se ha reducido el consumo energético medio, que era de 6.43mA, hasta 2.3mA, que es un consumo muy pequeño comparado con el que generan el resto de los componentes.

Anexo 7:

Caracterización del sistema de carga y alimentación

En este anexo se comenta el montaje realizado para la medida de los parámetros de carga del panel solar y el cargador utilizados en el equipo final. También se mostrarán y analizarán los resultados obtenidos de este montaje.

Como punto de partida se ha de tener en cuenta que el consumo medio estimado del módulo esta en torno a los 2.3mA, como se expone en la memoria, pudiéndose llegar a picos de hasta 2.2A en caso de que el camino entre el transmisor y la estación base presentase muchas pérdidas, debidas a la distancia, desvanecimientos, u otras causas. A este consumo, hay que sumarle también el consumo de los distintos componentes de la placa, que es de unos 40-45mA, por lo que se puede establecer un consumo medio aproximado de unos 50mA en total.

Para realizar las pruebas de carga de la batería, se emplea el cargador final, conectándole en la entrada el panel solar, y en la salida, la batería y un regulador, al que se conecta una carga que simula el conjunto del equipo. Dicha carga, debe generar un flujo de 50mA por ella, por lo que, como el regulador da en su salida 3.3V, se utiliza una resistencia de 0.25W, ya que el consumo es de aproximadamente 0.165W, con un valor de 66Ω. Al no disponerse de una resistencia de dicho valor exacto, se emplea una de 64.9Ω, simulando así una corriente casi idéntica, más concretamente de unos 50.84mA, pero con un consumo aun así menor de 0.25W.

El panel solar utilizado puede llegar a generar una tensión de unos 7.5V entre sus terminales, por lo que, teniendo en cuenta que el cargador a usar trabaja correctamente con tensiones menores a 7V, se coloca un regulador de tensión de 5V a su salida, de forma que así se pueda aprovechar al máximo la capacidad de carga proporcionada por el panel y el cargador.

Así, haciendo uso de un *datalogger* (*34972A LXI Data Acquisition/Data Logger Switch Unit* de *Agilent*), se medirán, durante varios días, las tensiones de batería, del regulador de la carga, del regulador del cargador y la proporcionada por el panel solar, y las corrientes a la entrada y la salida del cargador. La tensión de la batería proporciona una medida de la carga que ésta almacena, mientras que la corriente a la salida del cargador da la medida de cuánto se está cargando en ese instante la batería, que debería ser aproximadamente algo menor que la intensidad a la entrada del cargador, pues antes del cargador se deriva parte de la corriente a la carga. La tensión del regulador de la carga indica siempre el mismo valor, a no ser que en algún momento la batería se descargue tanto que no sea capaz de proporcionar la corriente suficiente a la carga. La tensión proporcionada por el panel solar permite observar las diferencias de carga en ciertos momentos del día.

Con esto, se monta un circuito como el mostrado en la figura 28, donde los voltímetros y amperímetros son las entradas de tensión y corriente del *datalogger*, montaje que se puede ver en la figura 29.

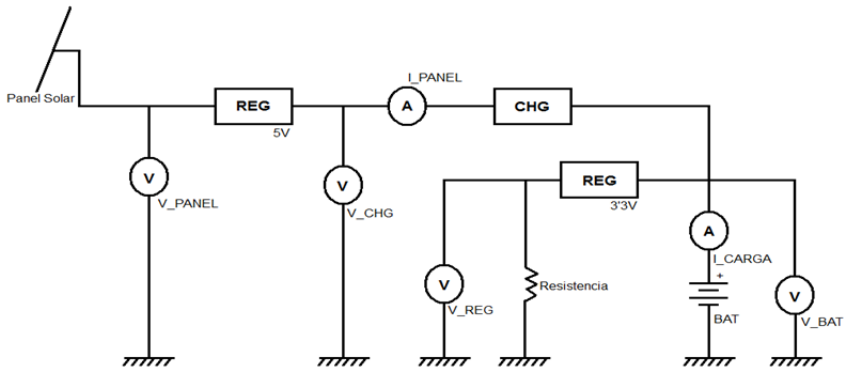
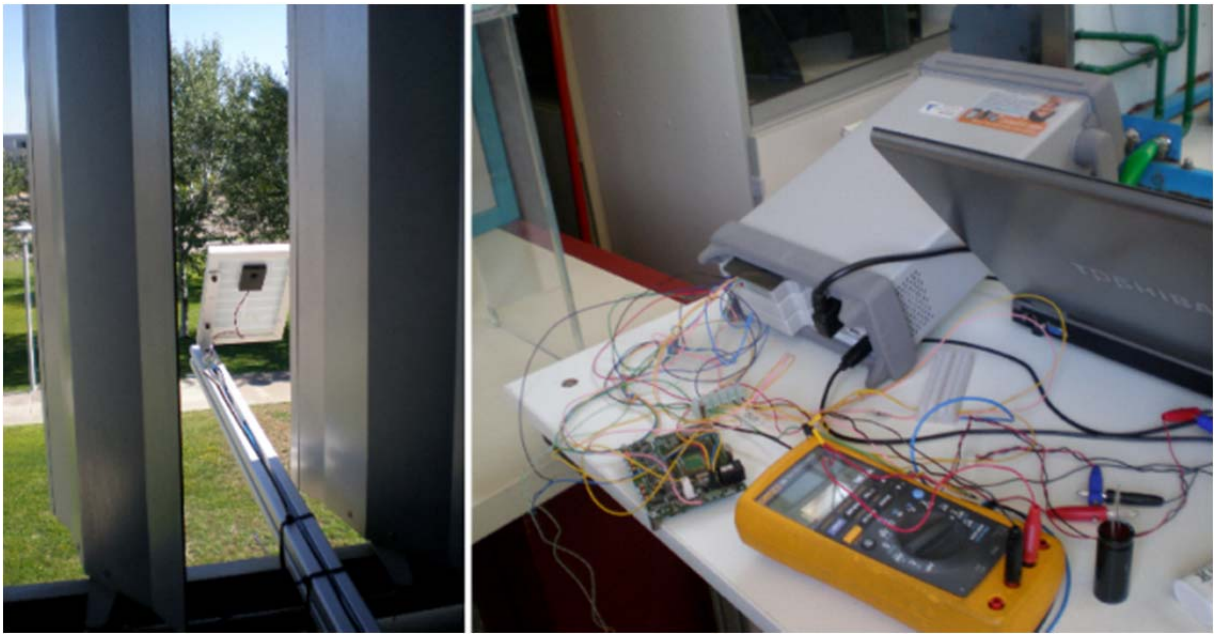


Figura 28: Esquema del montaje realizado para la toma de medidas de carga de la batería por medio del panel solar.



(a) (b)

Figura 29: (a) colocación del panel solar en dirección al sur y con una inclinación de 56º. (b) montaje para efectuar las medidas con el datalogger.

Con esto, se obtienen las medidas mostradas en las figuras 30, 31 y 32:

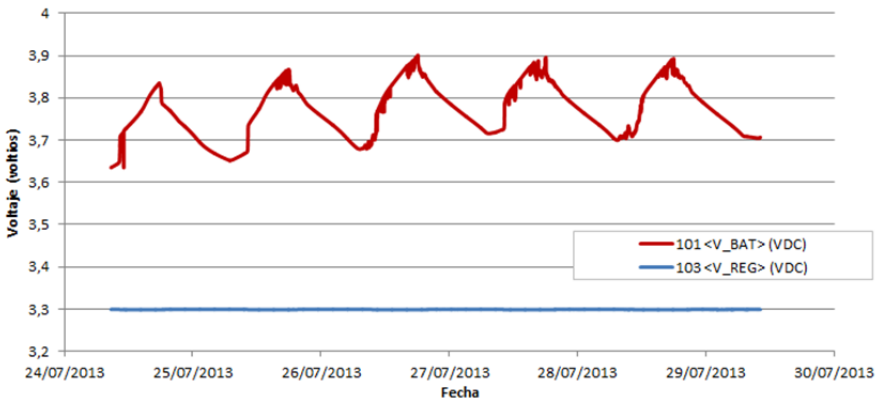


Figura 30: Tensiones de batería y a la salida del regulador obtenidas del montaje expuesto.

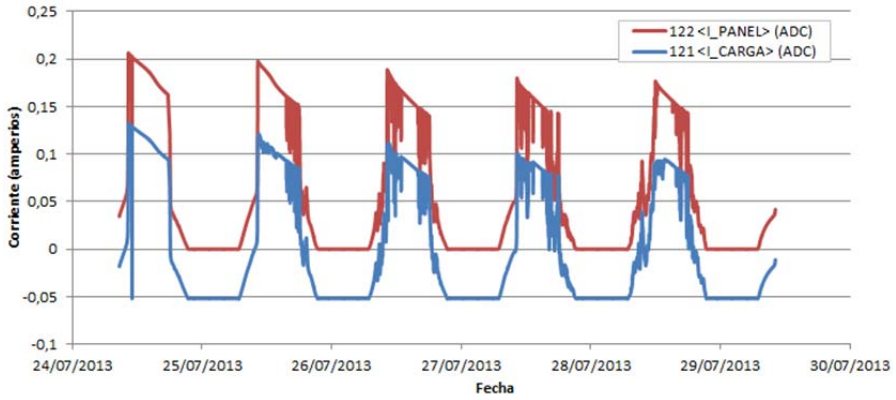


Figura 31: Corrientes de carga y proporcionada por el panel del montaje expuesto.

En la figura 31, se observa cómo en los primeros días, la tensión de la batería al final de cada día es mayor que la tensión que se tenía el día anterior, por lo que se está cargando más de lo que se está gastando. En los dos últimos días, como consecuencia de un aumento de la nubosidad, la corriente total de carga disminuye, haciendo que la batería pierda carga día a día. Como esta tensión no decae demasiado en ninguno de los días, la tensión suministrada por el regulador está fija entorno a los 3.3V, como cabía esperar. Por otra parte, en la figura 32, se observa que la corriente de carga es siempre unos 50mA menor que la corriente suministrada por el panel, lo que se corresponde con la intensidad que está consumiendo la resistencia. Además, se observan también dos no idealidades. Una de ellas es que la intensidad de carga no crece al principio del día linealmente, ni cae linealmente al final del día, hasta el punto máximo, defecto que se ha identificado con que el panel solar está en sombra al principio y al final del día. La otra no idealidad es que la intensidad llega al punto máximo, y va cayendo conforme avanza el día, lo que se ha identificado con el hecho de que la batería esté casi cargada, de forma que requiere un menor cargado. Esto se observa también en la tensión de batería, ya que va aumentando más lentamente. A este mismo respecto, comentar también que el máximo de intensidad de carga obtenido en la gráfica está en torno a los 200mA, mientras que en unas pruebas previas, se obtenían valores de corriente de carga de hasta 280mA, por lo que se puede suponer que en caso de que la carga de la batería fuese menor, se podría llegar a generar una corriente de carga superior. Además, se aprecia también el efecto de las nubes, que hacen que la intensidad de carga disminuya cuando éstas tapan el sol. Estos dos últimos efectos se observan más claramente en la figura 33:

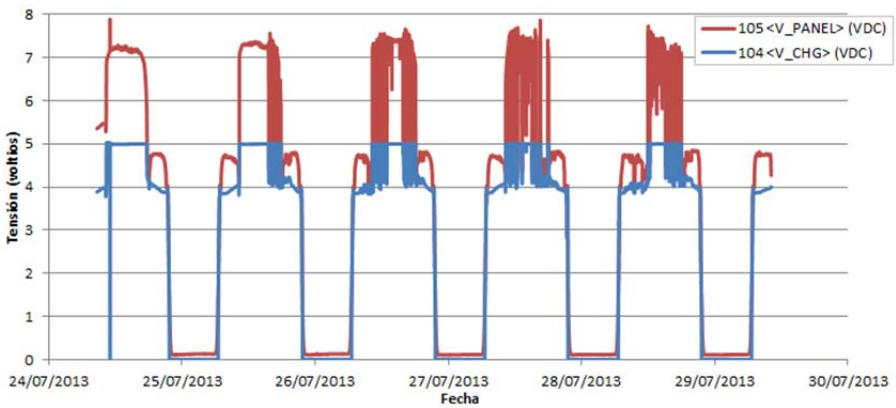


Figura 33: Tensiones de panel y de carga obtenidas del montaje expuesto.

En ella se puede ver que tanto al principio como al final del día se dan cambios bruscos de tensión de panel, debido como se ha comentado a que el edificio hace sombra al panel. Es muy

importante tener esto en cuenta, pues al situar el equipo en campo, puede haber elementos que hagan sombra en determinados momentos del día. También se observa que la obstrucción de rayos solares por parte de las nubes hace caer bastante la tensión del panel.

Anexo 8:

Modificaciones para la inserción del módulo 3G

Se han estudiado también las variaciones que habría que realizar para la inserción del *módulo 3G* en lugar del *módulo GPRS*, para hacer uso de la encriptación de datos que éste proporciona.

Por un lado, estaría la reconfiguración del servidor, para que admitiese los datos encriptados. Esta se ha de realizar en el archivo existente expresamente para ello, y que posteriormente se incluirá en el de configuración general, donde además habrá que activar el uso del módulo correspondiente. Para el uso de este módulo, se tendrá que crear un certificado digital del servidor para las conexiones desde navegador, y una clave de encriptado de datos que se utilizará en todas las comunicaciones. El certificado, como no se validará por ninguna empresa de certificación, dará en el navegador desde el que se acceda a la web una alerta de no autenticidad, pero al ser para uso privado, no será un problema, ya que el usuario conocerá la veracidad del mismo, y de esta forma no se deberá pagar una certificación, pues son bastante caras.

Por otro lado, se procederá a analizar el *módulo 3G* al igual que se ha hecho con el *módulo GPRS* (figura 34), para dejar así preparada el posible cambio de tecnología usada. Para ello, se tendrá que hacer uso del manual de *comandos AT* del chip *SIM5218* de *SIMcom*, que es el chip de comunicaciones 3G que incluye este módulo, puesto que los comandos variarán de forma, además de tenerse que usar otros diferentes y establecer una comunicación diferente.

Figura 34: Módulo 3G de Arduino, con el chip de comunicaciones SIM5218 de SIMcom montado.

Lo primero, será observar qué ocurre con el hardware. En este caso, se puede observar que la placa es mayor, pues el propio chipset tiene mayor tamaño, y además incorpora más pines, utilizados principalmente para conectar periféricos como altavoces, micrófono o cámara de video. Estos pines no van a interesar para esta aplicación, por lo que se puede llevar a cabo una selección

de conexiones como se hizo con el *módulo GPRS*, para ahorrar espacio en la placa de circuito impreso. De esta forma, se llega a que van a ser necesarios los pines de transmisión y recepción, el de encendido del módulo, y los de alimentación del módulo. Esto, junto a que su posición coincide con las posiciones que ocupaban en el *módulo GPRS*, hacen que ambos sean compatibles en cuanto a este aspecto, pudiéndose utilizar la misma circuitería, aunque el tamaño y la forma de la placa cambien, lo que por otra parte puede resultar insignificante en caso de que el receptáculo donde se fuese a ubicar la misma tiene el suficiente espacio para el *módulo 3G*. Así pues, si se quiere crear también el elemento en la librería de *CadSoft Eagle*, únicamente habrá que cambiar la forma y dimensiones de la placa, manteniendo los mismos pines para el elemento adaptado. Además, la antena usada con el *módulo GPRS*, al usar ambos las mismas bandas de frecuencias, es compatible también con este módulo. Además, las alimentaciones para el *módulo 3G* y para el *módulo GPRS* son iguales, por lo que la parte de alimentación tampoco deberá ser tocada, pero sí asegurarse de que el mayor consumo de éste nuevo módulo, que puede llegar a tener picos de corriente de hasta 3A debido a las diferencias entre ambas tecnologías comentadas en el capítulo 2, no incrementa demasiado el consumo.

Observando el software necesario, habrá de tenerse en cuenta que el método de encendido será el mismo que para el *módulo GPRS*, pero para apagarlo, además poderse hacer enviando un *comando AT*, en este caso el comando "*AT+CPOF*", también se podrá enviar el mismo pulso que para encenderlo. De este modo, tras avisar a la red de que el dispositivo va a ser apagado, dejará de funcionar. Así pues, se puede crear únicamente una función de encendido/apagado, que se use para realizar ambas acciones enviando un pulso de nivel bajo, y obviando la existencia del *comando AT* de apagado. Dicho comando podría ser útil para mantener la fecha y la hora correctas del reloj de tiempo real, ya que apagando así este módulo, no se corta la alimentación del mismo, y se mantiene en marcha este servicio, pudiendo de esta forma ser utilizado, aunque por otro lado, seguirá consumiendo, aunque menos, reduciéndose así la duración de las baterías.

Por otra parte, los comandos requeridos para llevar a cabo la comunicación también cambiarán, teniendo que usarse en este caso los específicos del protocolo *HTTPS* (Hyper-Text Transfer Protocol Secure) y no pudiendo abrirse simplemente una comunicación *TCP*, para así poder implementarse la capa de seguridad. Además, la configuración previa necesaria también será diferente, ya que en este caso, además de usarse comandos diferentes, no hay que configurar los caracteres de terminación de línea y línea nueva, ni la clase multislot a utilizar, pero debiéndose añadir la configuración de la actualización de fecha y hora. Además, las bandas de frecuencias a utilizar, en este caso incluirán las de *WCDMA* (Wideband Code Domain Multiple Access), y se deberá configurar en qué orden se han de usar las tecnologías, si primero probar *GPRS* y si no está disponible, probar con *WCDMA*, o al revés. Por último, la tasa de transmisión por puerto serie en este caso no se puede seleccionar automáticamente, si no que se le ha de dar un valor fijo, que por defecto es 115200 baudios, que es compatible con el microchip controlador que se utiliza en la placa de los sensores. El comando de introducción del PIN de la tarjeta SIM no cambia, ya que sigue siendo "*AT+CPIN*". Lo mismo ocurre con el comando de selección de operador, que será "*AT+COPS*", que también se fijará en automático en este módulo, para que se conecte al operador fijado por la tarjeta SIM.

Para establecer la comunicación, habrá que asegurarse al principio de estar conectado a la red ("*AT+CREG?*"). En caso de estarlo, lo primero será iniciar la pila de protocolos usando la capa de

seguridad ("*AT+CHTTPSSTART*"). Tras esto, ya se podrá iniciar una sesión al puerto *<puerto>* del servidor *<servidor>* con el comando "*AT+CHTTPSOPSE=<servidor>,<puerto>*", de forma que se realiza el *handshake* de inicio de sesión. En este punto, ya se puede iniciar el envío de la petición. Para ello, se envía el comando "*AT+CHTTPSSEND=<longitud>*", con *<longitud>* la longitud de la cadena de la petición completa a enviar. Una vez hecho esto, el módulo responderá con el carácter '*>*', momento en el que se puede comenzar a enviar la cadena. Por último, para recibir la respuesta del servidor, se hará uso del comando "*AT+CHTTPSRECV=<min_longitud>*", donde *<min_longitud>* es la mínima longitud a ser recibida, a lo que el módulo contestará con la cantidad de datos recibidos, y los enviará por el puerto serie.

Para concluir con el *módulo 3G*, destacar que el tráfico generado será mayor al hacer uso de la capa de seguridad, pues se establecerá un periodo de *handshake*, el iniciado con el comando "*AT+CHTTPSOPSE*", en el que se configurarán los parámetros de encriptado de la comunicación en el cliente y en el servidor. Este *handshake*, como se trabajaría sin certificados digitales, sigue los siguientes pasos: Cliente envía *ClienHello*, servidor envía *ServerHello*, ambos negocian el cifrado, y finalmente se inicia la comunicación HTTP. Este incremento de tráfico, además de ser bastante grande para los tamaños de peticiones que se generan, afectará a todas las comunicaciones, por lo que su efecto no será despreciable, haciendo que el número máximo de equipos conectables a través de una única tarjeta SIM sea bastante menor que con el *módulo GPRS*.

PARTE III

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] Especificaciones técnicas de GSM:
http://www.3gpp.org/ftp/Specs/html-info/45-series.htm

[2] Manual de GPRS:
http://www.radio-electronics.com/info/cellulartelecomms/gprs/gprs_tutorial.php

[3] Manual de UMTS:
http://www.radio-electronics.com/info/cellulartelecomms/umts/umts_wcdma_tutorial.php

[4] Información técnica acerca de Bluetooth:
http://www.bluetooth.com/Pages/Tech-Info.aspx

[5] Especificaciones técnicas de ZigBee:
http://www.zigbee.org/Specifications.aspx

[6] Estándar TCP:
http://www.rfc-es.org/rfc/rfc0793-es.txt

[7] Estándar UDP:
http://www.ietf.org/rfc/rfc0768.txt

[8] Estándar FTP:
http://tools.ietf.org/html/rfc959

[9] Estándar HTTP:
http://www.w3.org/Protocols/rfc2616/rfc2616.html

[10] Web oficial de la plataforma Xively:
https://xively.com/

[11] Web oficial de la plataforma Exosite:
http://exosite.com/

[12] Web oficial de la plataforma Carriots:
https://www.carriots.com/

[13] Manual de usuario de HTML:
http://www.desarrolloweb.com/manuales/21/

[14] Manual de usuario de PHP:
http://php.net/manual/es/index.php

[15] Tutorial de Javascript:
http://www.w3schools.com/js/

[16] Web oficial del proyecto del servidor web Apache:
http://httpd.apache.org/

[17] Web oficial del servidor de bases de datos MySQL:
http://www.mysql.com/

[18] Web oficial de la herramienta de administración de MySQL, PHPMyAdmin:
http://www.phpmyadmin.net/

[19] Web oficial del módulo de creación de gráficas FusionCharts:
http://www.fusioncharts.com/

[20] Manual no oficial de instalación de Apache, PHP, MySQL, y PHPMyAdmin:
http://www.lawebdelphp.com/manuales/lampp.pdf

[21] Manual no oficial de instalación de Apache, PHP, MySQL, y PHPMyAdmin:
http://www.maestrosdelweb.com/editorial/phpmysqlap/

[22] Guía básica para la securización del servidor web Apache (Instituto Nacional de Tecnologías de la Comunicación):
http://www.inteco.es/guias/guia_apache

[23] Gestión de sesiones web: ataques y medidas de seguridad (Instituto Nacional de Tecnologías de la Comunicación):
http://www.inteco.es/guias/guia_sesiones_web

[24] Web oficial de Libelium:
http://www.libelium.com/es/