Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DEPARTAMENTO DE SISTEMAS TELEMÁTICOS Y COMPUTACIÓN (GSyC)

Proyecto Fin de Carrera:

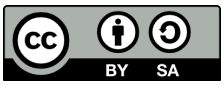
SISTEMA DOMÓTICO DISTRIBUIDO, INALÁMBRICO, INTEROPERABLE Y OPENSOURCE

PFC para la carrera superior de Ingeniería de la telecomunicación en la Universidad Rey Juan Carlos

---

Tutor:
José María Cañas Plaza

Alumno:
Daniel Castellano Bonilla



Dedicado a todas las dificultades
que me he encontrado en el camino y
a quienes me han ayudado a superarlas.

Agradecimientos

A mis padres, que han renunciado a mucho para poder verme llegar hasta aquí.
La mayor parte de este triunfo es suyo. Gracias.

A mi novia, por aguantarme incluso en los días de más estrés (realmente difícil).

A mi hermano, quien ha creado desinteresadamente el diseño de la web de Surveillance 4.

A toda mi familia, que siempre se ha preocupado por mí y por mis estudios, en especial a mi bisabuela, recientemente fallecida.

A José Mª, por ayudarme y aconsejarme durante toda esta larga aventura.

A todos mis profesores, pasados y ¿futuros?, por enseñarme con dedicación y esmero.

Y, en general, a todos los que han hecho posible que pudiese llegar hasta aquí.

Índice general

Índice de figuras

Capítulo 1

Introducción

Se entiende por sistema domótico (del latín *domus* y el griego *automatos* e *-ico*) aquel sistema que implementa dispositivos (sensores, actuadores y/o elementos de control) que permiten la automatización de distintas tareas relacionadas con el hogar con distintos objetivos. Generalmente se busca el aumento del confort, reducción del consumo energético y aumento de la seguridad. Dichos dispositivos pueden estar integrados por medio de redes interiores y/o exteriores de comunicación, cableadas y/o inalámbricas, y cuyo control se puede realizar desde dentro y/o fuera del hogar.

También podemos contemplar dentro del marco de este proyecto el término "inmótica", el cual se refiere a los sistemas citados en el párrafo anterior o similares, pero aplicados a la industria o a otro sector que no sea estríctamente el inmobiliario (centros comerciales, fábricas, oficinas, etc.). En este término no se incluye la automatización de la producción. Los sistemas utilizados son prácticamente idénticos y en muchos casos no se hace distinciones entre ambos términos.

El origen de la domótica se remota a la década de los setenta, cuando tras muchas investigaciones aparecieron los primeros dispositivos de automatización de edificios basados en la tecnología X-10[7]. Posteriormente surgieron otros estándares que permiten más flexibilidad en

Figura 1.1: Figuración artística de un sistema domótico.

las configuraciones de los sistemas, como KNX[8], actual estándar europeo.

Este sector tiene varios campos de aplicación, posiblemente el más extendido sea el de la seguridad. Es por ello que el sensor más utilizado son las cámaras RGB (no funcionan en la oscuridad) y los detectores de movimiento (actualmente no permiten distinguir entre personas y animales u otros objetos). Por supuesto, el actuador más utilizado es la alarma (por ejemplo, un altavoz, aunque suelen ofrecer más funcionalidades). Fuera de este sector, el sensor de movimiento sigue siendo más utilizado, junto a relés para encender o apagar luces al paso de personas, ahorrando electricidad.

El manejo eficiente de los recursos es cada vez más una obligación para la sociedad, pues la mayoría de estos recursos son finitos y, en muchos casos, escasos. Poder racionalizar el uso de la electricidad, el agua y el gas, manteniendo el confort (o mejorándolo comparado con una casa sin domotizar) con menos recursos es beneficioso no sólo para los bolsillos del cliente, si no también para el medio ambiente.

Domotizar una casa también tiene otros beneficios: se pueden implementar módulos de seguridad (algunos aplicables incluso estando dentro de la casa), simulación de presencia por si nos ausentamos un tiempo largo, podemos personalizar las alarmas según la hora y el día en que ocurran y un largo etcétera.

Los orígenes de la domótica en España deben buscarse a primeros de los años noventa, en los que tienen lugar las primeras iniciativas en promociones inmobiliarias y un mayor conocimiento de sus beneficios. Del mismo modo que en nuestros días no es aceptable que una vivienda no tenga corriente eléctrica o agua corriente, dentro de poco no se concebirán viviendas que no estén mínimamente domotizadas.

La domótica y la inmótica están evolucionando a un ritmo vertiginoso tecnológicamente hablando. Hasta hace poco sabíamos lo que era gracias únicamente a las películas, ahora se empieza a implantar en hogares, principalmente como elemento de seguridad o en hogares de personas con capacidades reducidas. Pero aún falta mucho para llegar a los sistemas que podemos ver en la gran pantalla, donde se junta seguridad, comodidad y un ahorro energía subyacente, que no se ve, pero existe. A fin de llenar ese gran hueco nace este proyecto.

El principal problema a la hora de comercializar estos sistemas es que el sector de la vivienda es uno de los sectores que se han mantenido más reticentes a la incorporación de las nuevas tecnologías. Existe un elevado desconocimiento entre los principales usuarios de estas tecnologías, los propietarios de viviendas, generalmente propiciado por desconocimiento técnico. Si los potenciales usuarios desconocen las virtudes y prestaciones de estos sistemas, difícilmente van a demandarlos. Esta falta de información ha conducido en la mayoría de los casos a asociar estas tecnologías con sistemas caros, de difícil uso, de poca utilidad, etc.

1.1. Estándares domóticos y conceptos

Los primeros sistemas comerciales fueron instalados, sobre todo, en Estados Unidos. Estos se limitaban a la regulación de la temperatura ambiente de los edificios de oficinas y poco más. Posteriormente la tecnología se fue desarrollando, aunque lentamente. Al ver la necesidad de interoperar distintos sistemas domóticos (el caso más típico es el de ampliar una red ya existente con nuevos componentes), se comenzaron a definir estándares domóticos.

Si bien el estándar inicial (x-10) fue suficiente para soportar estas pequeñas aplicaciones, posteriormente los automatismos destinados a edificios de oficinas (junto con otros específicos), se han ido aplicando también a las viviendas de particulares u otro tipo de edificios, donde el número de necesidades a cubrir es mucho más amplio, dando lugar a nuevos estándares de comunicación.

X10 Protocolo estándar de comunicaciones para el control remoto de dispositivos eléctricos, desarrollado en 1975. Utiliza la red eléctrica para comunicar los distintos dispositivos del sistema a través de comandos predefinidos en el estándar.[7]

Fue la primera tecnología domótica en aparecer, dando solución específica a las comunicaciones domóticas, diferentes de las comunicaciones arbitrarias entre dispositivos.

Este estándar sigue en uso hoy en día, aunque ha sido redefinido para otorgarle más funcionalidades y existen protocolos alternativos más recientes, pero con menor número de dispositivos comerciales (debido principalmente al extenso tiempo que X10 lleva en el mercado).

KNX Estándar europeo para la automatización de las viviendas y oficinas. Para la transmisión de datos puede utilizar cableado de baja tensión (el más utilizado actualmente), de alta tensión (red eléctrica), transmisión inalámbrica o por Internet.[8]

Es un estándar domótico abierto que permite implementar un mayor número de comandos y sensores más heterogéneos que *X10*. Si bien *KNX* no es compatible con el estándar *X10*, existen numerosos gateways que permiten el uso de sistemas multiprotocolo.

SCP Simple Control Protocol. Estándar de *Microsoft* y *General Electric* y actualmente libre de royalties.

Necesita conexiones punto a punto y cableado de baja tensión, por lo que es prácticamente inviable para sistemas con un gran número de sensores.

En general, este estándar no está teniendo gran acogida, además, es incompatible con la legislación Europea (en concreto, con la norma *EN-50065*[9]).

Redes de sensores Este concepto es bastante amplio, pues puede englobar cualquier red formada por un grupo de sensores con ciertas capacidades sensitivas y de comunicación inalámbrica, generalmente para formar redes ad hoc sin infraestructura física preestablecida ni administración central.

Estas redes están siendo más utilizadas en el control de grandes áreas y/o de difícil acceso (por ejemplo, para prevenir incendios en bosques), pues se caracterizan por su facilidad de despliegue, mantenimiento y control. La idea inicial que tras estas redes, fue repartir aleatoriamente los nodos en un territorio grande, el cual sería monitorizado hasta que los recursos energéticos de estos nodos se agotasen.

Aunque siguen en fase de investigación, empiezan a existir aplicaciones comerciales basadas en este tipo de redes.

Si bien los estándares están definidos para albergar cualquier tipo de funcionalidad, podemos dividir la domótica actual en tres grandes sectores.

Figura 1.2: Extracto de elementos "domotizables".

1.1.1. Seguridad

Como elemento de seguridad, el sistema tiene dos vertientes, la **seguridad personal** estando dentro de casa, y la seguridad de los bienes materiales; ambos son similares en cuanto a los elementos utilizados, pero pueden tratados de forma diferente.

Por ejemplo, para proteger los bienes, se puede simular la presencia de personas en la casa o hacer sonar la alarma si hay indicios de que se encuentran personas en la vivienda, cosa que, evidentemente, no aplica si hay usuarios legítimos del sistema en el interior. Sin embargo, hay otros métodos para proteger a los usuarios ante un ataque o robo. Probablemente el más conocido es la creación de una "habitación del pánico", que básicamnete consiste en construir una pequeña estancia blindada en los que los atacantes no puedan entrar y esperar de forma segura la llegada de las autoridades.

Los sensores más utilizados para abordar la seguridad en la domótica, son las cámaras. Han evolucionado según las necesidades del cliente, por ejemplo, existen cámaras IP para poder visualizarlas desde Internet, o todo lo contrario, circuitos cerrados de TV; pueden estar siendo monitorizadas remotamente por un operario o pueden conectarse a un procesador que ejecute algoritmos de detección automática para ciertas acciones, incluso hay cámaras simuladas (de plástico) para asustar al posible asaltante. También pueden ser utilizados en ciertos negocios para controlar el acceso a alguna instalación, cámaras para evitar robos, etc. todo esto nos da una idea de la gran utilidad que ofrecen estos sensores. Por supuesto, tienen sus limitaciones: son especialmente sensibles a la falta de luz; baja mucho el rendimiento de la cámara (aumenta el ruido en la señal RGB) cuando disminuye, hasta inhabilitar la misma si no hay suficiente luz.

1.1.2. Ahorro de energía

Con sensores como el sensor de intensidad lumínica podemos mantener una cantidad de luz suficiente en una habitación controlando a su vez las persianas para que el consumo eléctrico sea mínimo. Si combinamos los sensores de luz con sensores de presencia, podemos también apagar las luces de las habitaciones donde no se requieran, reduciendo así drásticamente la potencia eléctrica consumida.

Figura 1.3: Estándares y marcas comerciales de dispositivos domóticos.

1.1.3. Comodidad

Otro de los pilares de la domótica es la comodidad en el hogar, por ejemplo, manteniendo una temperatura agradable o una luz constante en los distintos habitáculos de la casa aunque las condiciones exteriores cambien (puesta de sol, rachas de aire, nubosidad, etc.).

Como todo el sistema, estas funcionalidades están entrelazadas, por ejemplo, mantener la misma temperatura en todas las habitaciones, incluso con las puertas cerradas, nos evitará tener que aumentar la intensidad de la calefacción al entrar a una habitación más fria, o al contrario, llegar a un habitáculo y comprobar que hace más calor; de esta forma también podemos disminuir el consumo en gas o electricidad.

La domótica es una tecnología jóven, por lo que no se puede acotar su funcionalidad estricatamente a lo indicado anteriormente, aunque la mayoría de las aplicaciones comerciales actuales se centran en estos casos.

Como vemos, es un poco temeroso acotar las funcionalidades de un sistema domótico genérico, pues como toda tecnología, no para de avanzar.

1.2. Mercado domótico

Recientemente ha crecido el número de empresas que ofrecen sistemas domóticos interoperables entre sí, permitiendo crear sistemas de forma flexible a un bajo precio.

Generalmente se necesitan conocimientos técnicos para poner a punto este tipo de redes, por contra, con estos sistemas podemos abarcar muchas soluciones. Por ejemplo, los tres casos típicos de la domótica (seguridad ante robos y averías domésticas, comodidad en el hogar y ahorro de energía), tanto juntos como por separado, adquiriendo únicamente los nodos que requiramos.

Hay empresas que venden sensores aislados que no necesitan interconexión (por ejemplo, sensores de vibración que emiten sonido) o que tienen una conexión limitada para, por ejemplo, instalar varios sensores del mismo tipo a una centralita deslocalizada que será la que emita la alarma (principalmente interconectados por X10) o visualizar una cámara IP. Estos sistemas son generalmente baratos y fáciles de instalar y mantener, pero apenas ofrecen flexibilidad.

Otras empresas ofrecen servicios de diseño e implementación de sistemas domóticos e inmóticos; esto conlleva todas las ventajas y desventajas de sistemas creados a medida, no pueden ser reutilizados y suelen tener un alto precio.

Veamos a continuación algunos ejemplos representativos del marco comercial de la domótica en España:

Figura 1.4: Sistemas domóticos comerciales actuales.

Z-Wave Se auto definen como estándar internacional[10]. Dispone de varios módulos de distintos fabricantes interoperables, con una gran variedad de sensores y actuadores. Permite crear sistemas inalámbricos y gestión a través de telefonos inteligentes.

Al ser sistemas de distintos fabricantes, son sistemas propietarios, además, aunque el sistema está en venta, algunos módulos siguen en desarrollo.

Al tener distintos fabricantes, los precios varían mucho, por ejemplo, un controlador (nodo central) cuesta entre 60euros y 850 euros (el más básico no dispone de conexión Ethernet, entre otras desventajas), un sensor cuesta entre 40e. y 60e., una cámara, entre 70e. y 130e., etc.

Precio no muy elevado (depende del módulo y del fabricante), disponen de servicios de suscripción de pago.

Verisure Sistema de alarmas de la empresa *Securitas Direct*[11]. No se trata de un sistema domótico completo, pues únicamente disponen de sensores de alarma.

Los nodos que se ofrecen para este sistema son siempre los mismos, no parece que se pueda crear un kit de sensores a medida (al menos, no se oferta en la web, es posible que se pueda pagando más). Este kit contiene un nodo central, dos sensores de movimiento con cámara, un sensor de puerta abierta, un lector NFC y una sirena. La principal ventaja de este sistema consiste en el uso de sonsores volumétricos adaptado a mascotas. Esto permite el uso de dichos sensores ante la presencia de animales sin provocar falsas alarmas.

Su modelo de negocio es algo diferente, ya que cobran una cuota mensual de 99e. y disponen de servicio de llamada a la policía (vídeo vigilancia en la centralita de *Securitas direct*).

Prosegur [12] Empresa similar a *Securitas Direct* y que ofrece servicios de vigilancia y "domótica". En su página web indican que disponen de un sistema domótico completo (seguridad, confort y ahorro energético), aunque sólo se muestran módulos de vídeo-vigilancia (cámaras IP), detectores de movimiento, switchs magnéticos y centralita con tag NCF.

El modelo de negocio es parecido al de *Verisure*, cobran "desde 30 euros al mes". El equipamiento incluido es similar al de *Verisure*, salvo que incluyen otro sensor de movimiento en lugar de el de puerta abierta.

Delta Dore Ejemplo de sistema domótico actualmente comercializado, genérico, multipropósito e inalámbrico [13]. Se compone de distintos módulos según la necesidad del cliente; el sis-

tema puede tener capacidad para gestionar, la climatización y la luz de la casa, alarmas y apertura de puertas automáticas. El sistema se controla a través de un mando a distancia, desde un cuadro de mando o desde una aplicación para el móvil.

A pesar de ser un sistema muy similar en concepto al propuesto en este PFC, su precio es elevado. Por ejemplo, los nodos centrales cuestan a partir de 200 euros, sensores a partir de 65 euros, etc. aunque existen packs para reducir el precio final.

Libelium Sistema de red de sensores, de fabricación española que implementa *router* multi-protocolo, el cual puede interactuar con las tecnologías Wi-Fi, ZigBee, GPRS, Bluetooth y GPS en una misma máquina.[14]

Es un sistema principalmente pensado para desarrollo (utiliza Arduino y XBee como base), aunque la propia empresa Libelium puede crear sistemas a medida a partir de sus placas y sensores.

1.3. Domótica en la *URJC*

Surveillance 1.0 y 2.0 Surveillance 1.0 es el productos del trabajo fin de máster de *Roberto Calvo Palomino*[3]. Consiste en un sistema de vídeo vigilancia distribuido integrado en dispositivos móviles basados en la plataforma Android.

El sistema realiza grabaciones a través de las cámaras instaladas y lleva asociada una generación de alarmas mediante detección de movimiento. Todo ello es gestionado y visualizado desde un dispositivo móvil inteligente basado en Android. La conectividad entre las cámaras y el dispositivo se realiza de forma alámbrica.

Surveillance 2.0 se constituye como un exploración aparte, realizada por *Roberto Calvo*, para visualizar el valor de diversos sensores a través de un telefono inteligente con Android. Para ellos se implementó una aplicación compuesta por diversos sensores conectado a una placa Arduino y cuyos datos finales eran visualizados en un dispositivo móvil. Para lograr las comunicaciones, tanto los sensores con la placa como la placa con el dispositivo móvil estaban unidos mediante cables.

Se integraron sensores de infrarrojos (movimiento), temperatura y puerta abierta (sensor magnético). Para ello, los sensores se unen a una placa Arduino mediante un cable USB, la placa procesa los datos y los envía a través de su interfaz Ethernet para poder visualizarlo en un tercer dispositivo. La comunicación entre el Arduino y el dispositivo final (la muestra se realizó con un dispositivo Android) se basa en los protocolos HTTP y JSON.

Sistema de riego Trabajo fin de máster de *Rubén Orellana Galloso*[4], realizado en el departamento de *tecnología electrónica* de la *URJC*.

El proyecto plantea la modernización de un sistema de riego de accionamiento manual a otro automatizado que haga un uso óptimo del agua. Se basa en la creación de un sistema de riego automático utilizando las tecnologías Arduino, ZigBee, Android, Bluetooth y otros elementos de bajo coste para monitorizar el estado de la tierra y mejorar el consumo del agua en un emplazamiento gaditano a partir de un sistema de regadío ya existente.

Plataforma Jderobot Este entorno de desarrollo software ha sido creado por el departamento *GSYC* de la *URJC*, aunque está centrado en el desarrollo de aplicaciones, testeado y control para robots, tiene componentes que pueden resultar útiles en este proyecto.[5]

Entre los proyectos aplicables en domótica destacan *ElderCare*[6], soporte para dispositivos *X10* y *Surveillance* 1 y 2.

ElderCare es un sistema autónomo y no intrusivo, diseñado para detectar caídas mediante el uso de cámaras y *Kinect*s (o *Xtion*s). Su principal mercado es el cuidado de personas mayores, por lo que su uso principal se encuentra en residencias de ancianos o domicilios particulares donde vivan ancianos. El sistema permite reducir drásticamente el tiempo de respuesta a la asistencia en el caso de caída, estimando la posición del usuario caído y avisando a sus cuidadores o a la asistencia médica.

1.4. Motivación

Si bien existen multitud de sistemas domóticos, ninguno de ellos contempla todo lo que la domótica puede aportar, si no que la mayoría se centran en la televigilancia (una de las ramas de la domótica); además, todos tienen varios puntos negativos importantes, prueba de ello es que los sistemas domóticos aún no se han introducido realmente en los hogares, a pesar de que actualmente existe la tecnología necesaria para ello.

El principal "fallo" de estos sistemas suele ser el precio, generalmente excesivo para el consumidor medio y, aunque es cierto que estos sistemas nos ayudarán a reducir el consumo eléctrico y, por lo tanto, la factura de la luz, el consumidor tiene una gran barrera que es el coste inicial.

El segundo gran problema de los sistemas actuales es que la mayoría están enfocados a la seguridad, dejando de lado aspectos tan importantes como el ahorro de energía o la comodidad extra que podría aportar un sistema de estas características al usuario final, ya sea manteniendo un entorno agradable o automatizando diversas tareas.

Otra deficiencia general que este proyecto trata de suplir es la imposibilidad de interconexión entre distintos sistemas, pues los sistemas de una marca suelen hablar únicamente un protocolo e incluso ser incompatibles con otros sistemas aunque usen como base el mismo protocolo.

En este proyecto se pretende introducir una mejora sustancial sobre los trabajos anteriores1.3 (Surveillance 1 y 2), hasta el punto de poder ser usado como sistema real en una casa cualquiera, por ello, en lugar de tratar de mejorar el trabajo de *Roberto Calvo*, se ha decidido comenzar desde cero.

Capítulo 2

Objetivos

La tendencia actual de la domótica es bastante clara, aunque generalmente cada compañía se centra más en una vertiente en concreto para obtener una diferenciación en el mercado. Nuestro sistema pretende ser universal, esto es, que pueda interconectarse con el mayor número de protocolos, sensores y cualquier fuente de información posible. Además, el sistema tiene que ser altamente escalable y versátil, de este modo conseguiremos que pueda perdurar en el tiempo, evolucionando y siendo complementado a través de aportaciones de otras personas. En otras palabras, no se pretende conseguir un producto cerrado para presentar un PFC y olvidarse de él.

La pretensión de este PFC es diseñar, construir y programar un sistema domótico. En concreto, interesa diseñar la arquitectura y crear la infrestructura necesaria para el sistema. Cómo objetivos finales se ha propuesto el construir un sistema domótico inalámbrico, accesible desde Internet y que permita el uso de cámaras de vigilancia.

Inalámbrico Tecnología *wireless*, pues al ser un sistema inalámbrico, tendremos flexibilidad a la hora de distribuir los sensores, así como una instalación más sencilla y menos costosa que con sensores cableados.

Acceso desde Internet Internet está presente en todo el mundo, por lo que el acceso mediante web a nuestro sistema nos permite conocer el estado de nuestra casa estemos donde estemos.

Cámaras de vigilancia La cámara RGB es el sensor más usado actualmente en la domótica, por lo que se hace necesario ofrecer esta posibilidad al usuario.

Adicionalmente nos hemos fijado unos requisitos mínimos que lo hagan atractivo, pues el sistema que se pretende construir debe tener unas cualidades que le hagan destacar por encima de sus competidores. Estos requisitos se detallan a continuación:

Bajo coste Uno de los principales problemas de los sistemas domóticos es su alto coste. Surveillance 4.0 pretende romper esa tendencia para poder convertirse en un bien de consumo masivo y estar presente en la mayoría de los hogares.

Debido a la coyuntura económica actual, este requisito es indispensable para cualquier sistema que pretenda llegar a un número grande de hogares.

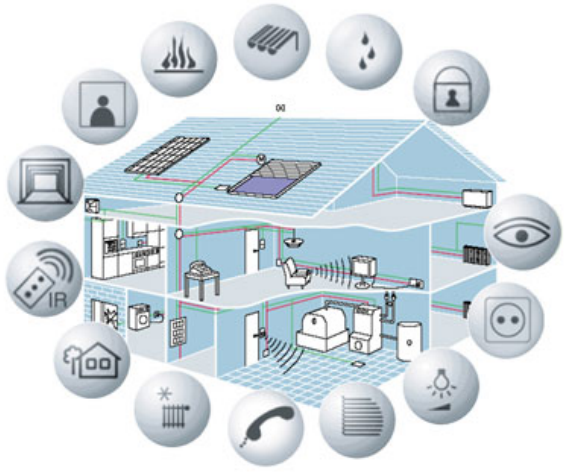


Figura 2.1: Posibles utilidades y conexiones del sistema.

Bajo consumo eléctrico Esta es una característica necesaria por ser un sistema inalámbrico, pues el ciclo de vida de las baterías depende de ello.

Estética atractiva Interfaz de usuario con alto grado de impacto visual y tratamiento complejo de los datos recibidos (apariencia moderna, creación de estadísticas).

Facilidad de uso Interfaces sencillas para su uso por personas sin conocimientos informáticos o tecnológicos. El sistema está enfocado para ser usado por gente que potencialmente no dispone de conocimientos técnicos avanzados, por lo que es necesario que pueda entender y controlar el sistema sin complicaciones.

Interoperabilidad Queremos ofrecer un sistema que permita el uso de distintos tipos de sensores, para que sea el usuario quien elija cómo monitorizar su casa, de acuerdo a sus necesidades.

Espacio reducido Los hogares de tamaño medio, en especial en España, no disponen de mucho espacio, por lo que el tamaño de todos los componentes del sistema ha de ser el menor posible. Este requisito a su vez nos exige usar sensores menos visibles, lo cual puede ser otra ventaja a los ojos del usuario.

2.1. Metodología

Para realizar el proyecto se ha realizado una gestión por cadena crítica. Si bien en este caso los recursos son fijos, estos son escasos (un único alumno), por lo que se puede aprovechar mejor el tiempo que si se gestionase el proyecto basado en hitos.

Se establecieron reuniones periódicas semanales entre el alumno y el tutor, en las cuales se verificaba el avance del proyecto, se exponían las dudas o ideas y se definía el flujo crítico y secundario para las acciones a corto y largo plazo.

Durante el transcurso del proyecto, se evitó el bloqueo del mismo teniendo siempre presente varias vías secundarias (objetivos secundarios) de investigación o desarrollo. Del mismo modo,

los objetivos intermedios se adaptaban según avanzaba el entorno, por ejemplo, puede darse la dualizadad de que en el mismo departamento se investigue la misma tecnología en dos proyectos diferentes, en tal caso, si no son complementarios, sólo se continuará una línea de investigación (el que disponga de mayor recursos, mayor experiencia, etc.). Una vez desbloqueada la línea de investigación, ambos proyectos podrán aplicar el conocimineto adquirido.

Para el mantener versionado el código del proyecto se ha utilizado una rama SVN [1] en el servidor de *Jderobot*. Durante la realización del proyecto, se ha mantenido una bitácora en la "wiki" de *Jderobot*[2], donde se informaba del progreso del proyecto.

El proyecto se ha dividido en dos grandes bloques: Surveillance 3.0 y Surveillance 4.0. En la primera solución, se abordó el uso de *ZigBee* y una interfaz de usuario básica, basandose todo el sistema en *Arduino*. Con Surveillance 4.0 integramos los aportes de la versión 3.0, creando una interfaz de usuario más atractiva y aumentando la funcionalidad del sistema mediante cámaras y actuadores. Este último sistema utiliza tanto al tecnología *Arduino* de su predecesor como placas *Raspberry Pi*.

Capítulo 3

Tecnologías aplicadas

El progreso se puede entender cómo una sucesión de tecnologás, más sencillas o más complejas, pero todas ellas se apoyan en otras previamente descubiertas. Este proyecto, como no puede ser de otra manera, también utiliza otras tecnologías ya desarrolladas. En este capítulo se explican someramente dichas tecnologías.

3.1. Lenguajes de programación

C/C++ C y C++ [15] son lenguajes de programación desarrollados en los años 1970 y 1980 respectivamente. Son lenguajes de bajo nivel (C++ tiene librerías orientadas a objetos) altamente eficiente en cuanto a recursos necesarios y con posibilidad de controlar todo el hardware (memoria, pines E/S, etc.).

En este proyecto vamos a programar las placas Arduino (y, por ende, los microcontroladores de las mismas) en C++.

Python Python [18], al igual que Java, es un lenguaje de programación interpretado y orientado a objetos. Fue desarrollado a comienzos de los años 80.

Se caracteriza por ser código abierto, fuertemente tipado y tener resolución de nombres dinámica.

En este proyecto usaremos el lenguaje Python para desarrollar los manejadores (programas que procesan las peticiones web de usuario y generan los datos que serán devueltos

Figura 3.1: Lenguajes utilizados en el proyecto.

al usuario) en *Django*3.6, puesto que está escrito en este mismo lenguaje y nos proporcionará mayor afinidad. Además, la placa *Raspberry Pi* está diseñada especialmente para ejecutar programas en Python, por lo que en este "miniordenador" obtendremos un buen rendimiento mediante el lenguaje Python.

3.2. Arduino

Arduino [22] es una plataforma de hardware libre, creada para el prototipado rápido de sistemas embebidos. Esencialmente es una placa (existen multitud de tipos diferentes de placa, según la aplicación a la que va dirigida) con un microprocesador y varios puertos de entrada/salida, junto con algunas funcionalidades hardware añadidas (regulador de voltaje, chip Ethernet, etc.). Incluye además un entorno de desarrollo libre y gratuito que permite la programación en C/C++ de los microprocesadores sin que el usuario tenga que tener en cuenta las particularidades de la placa que esté usando. Además, el programador dispone de varias librerías creadas específicamente para esta plataforma.

Arduino es una plataforma de electrónica abierta para la creación de prototipos. Una vez realizado el prototipo en la placa Arduino, es posible, con relativamente poco cambio en el código, extraer el microprocesador y prescindir de la placa, eliminando así alguno costes del hardware y consumo eléctrico.

En el prototipo Surveillance 3.0, todo el sistema domótico está construido sobre Arduino (nodo central y satélites); en la versión 4.0, el nodo central es una Raspberry Pi y los nodos satélites prescinden de la placa Arduino y utilizan únicamente su microprocesador.

La placa se programa en C/C++ (se hace uso de librerías propias proporcionadas por el fabricante) mediante un compilador propio de código abierto.

3.2.1. *Shields* para Arduino

Además de la placa Arduino, es posible extender su funcionalidad mediante placas hardware accesorias, llamadas *shield*. En este proyecto se ha utilizado la *shield* de Ethernet para utilizarla en el nodo central (en la versión 3) e inicialmente la *shield* XBee para facilitar el uso de ZigBee tanto en los nodos satélites como en el nodo central; posteriormente esta *shield* se dejó de utilizar para ahorrar costes y minimizar el consumo eléctrico.

3.3. Raspberry Pi

Raspberry Pi [23] es una placa computadora, esto es, un ordenador con todos sus componentes en una pequeña placa. Puesta a la venta en el 2012, hay disponibles dos modelos; nosotros utilizaremos el modelo "B", puesto que el modelo "A" no dispone de puerto Ethernet.

La placa tiene, entre otros, un procesador a 700MHz, un procesador gráfico (GPU), y corre bajo el sistema operativo Raspbian (un derivado de Debian, es decir, Linux), como soporte de almacenamiento utiliza tarjetas de memoria SD (de hasta 16GB).

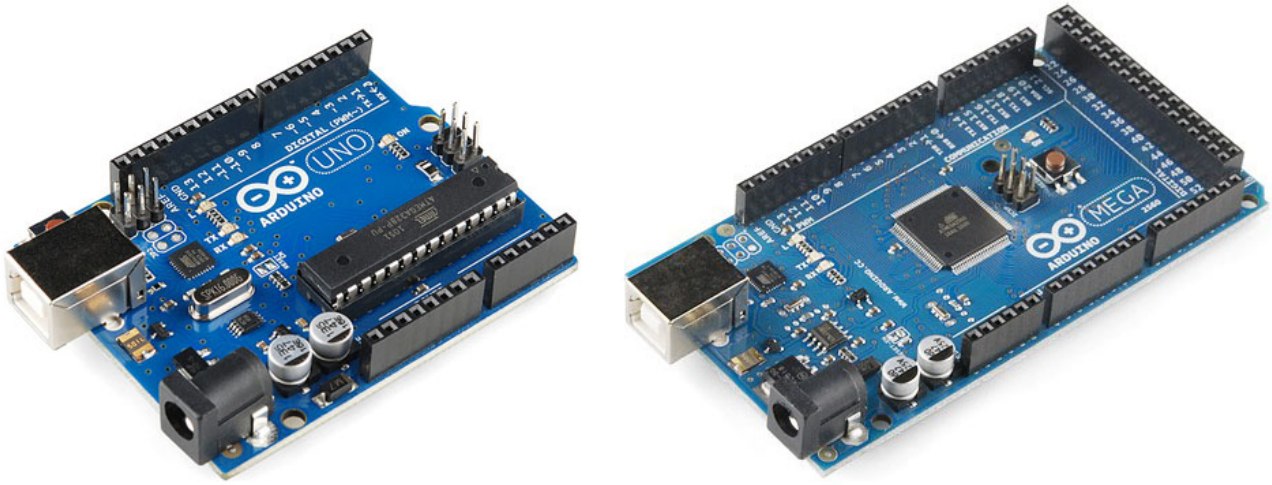


Figura 3.2: Placas Arduino UNO y Arduino MEGA 2560, ambas con procesador.

Nótese el salto en la disponibilidad de recursos respecto del Arduino (procesamiento a 700MHz vs 16 MHz, memoria RAM de hasta 16 GB vs 256 KB, etc.), lo cual puede permitir la realización de procesamientos más complejos y almacenamiento de datos. Por otro lado, el consumo eléctrico y el precio de la placa Raspberry Pi es algo superior a la del Arduino

Raspberry Pi no dispone de un reloj de tiempo real (los fabricantes justifican esta ausencia por el alto precio de este), por lo que le usuario tendrá que poner en hora el sistema al arrancar el mismo tras cada reinicio, así como cuando el usuario considere que el reloj está desviado con respecto a la hora real.

Además de los chips y puertos integrados, Raspberry Pi ofrece una interfaz GPIO programable para la comunicación con periféricos externos, como un emisor/receptor ZigBee.

3.4. WiFi

Wi-Fi es un sistema de envío de datos sobre redes inalámbricas utilizando ondas de radio, sus especificaciones del nivel físico y MAC están basadas en las definiciones del estándar IEEE 802.11[28].

El estándar dispone de varias versiones, siendo las más extendidas las versiones b, g y n (compatibles entre sí). La versión más reciente de las cuatro, el estándar IEEE 802.11n, permite operar a tasas teóricas mayores de 100 Mbps, aunque con un consumo energético elevado, pero con la ventaja de ofrecer un mayor alcance que sus predecesores (hasta 100 metros).

Si bien esta tecnología puede actuar como red "ad-hoc", la configuración típica es en modo infraestructura, el cual requiere un dispositivo que otorgue las funcionalidades de un punto de acceso (AP) y varios dispositivos que se conectarán de forma directa al primero.

Esta tecnología soporta protocolos de cifrado de datos ("WEP", "WPA", "WPA2", etc.) que se encargan de codificar la información transmitida para proteger su confidencialidad, también son comunes los filtros basados en MAC o IP, aunque estos dependen de la implementación de

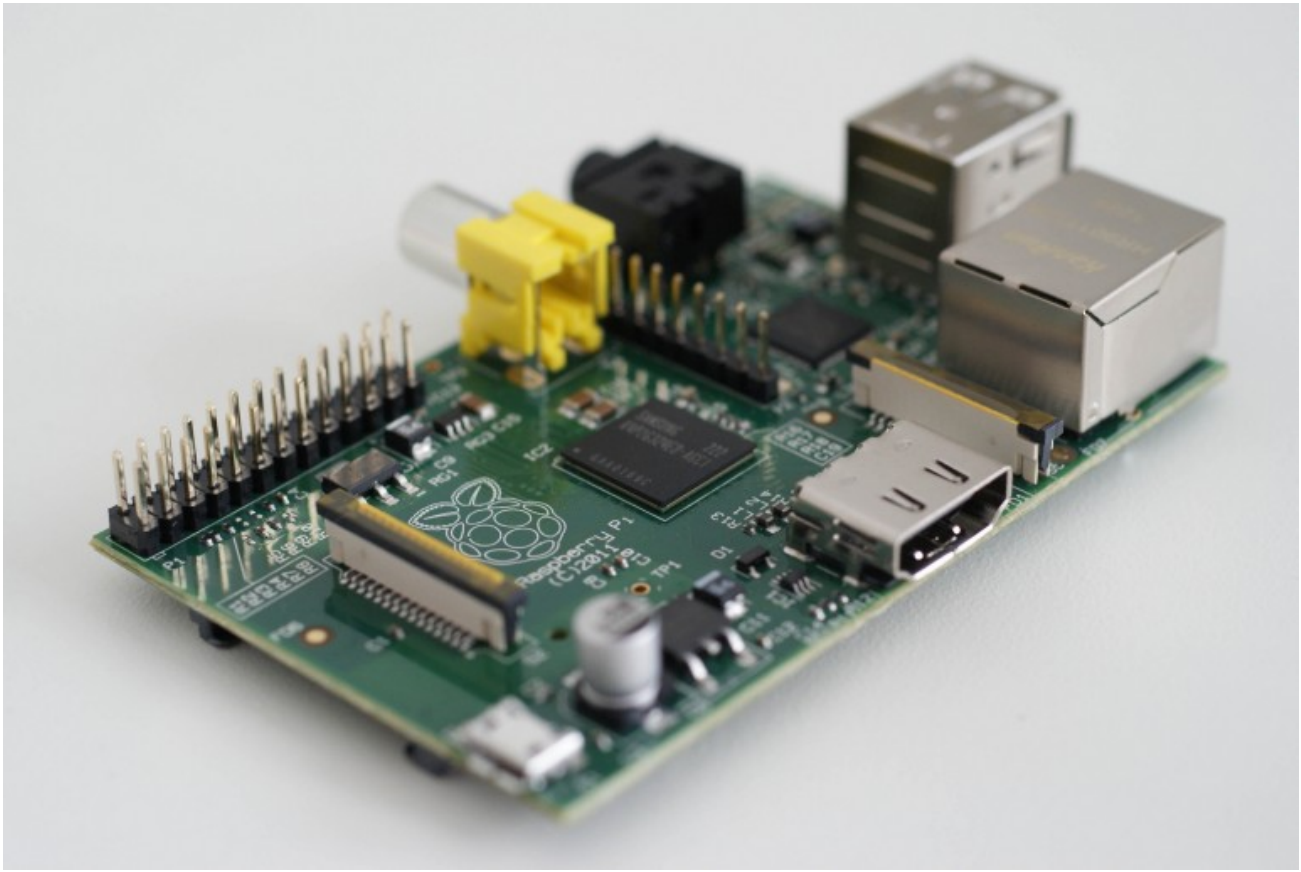


Figura 3.3: Raspberry Pi modelo "B".

los dispositivos.

En Surveillance 4.0, esta tecnología nos permitirá comunicar los nodos satélite con cámara y el nodo central entre sí y con Internet.

3.5. ZigBee

Basado en el estándar IEEE 802.15.4[29] y promovido por la asociación "ZigBee Alliance"[30], este protocolo de alto nivel para comunicaciones inalámbricas, está diseñado para ser usado por aplicaciones que requieran comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías. La primera especificación ZigBee vio la luz en diciembre de 2004, mientras que la especificación actual data del 2007 (perfil "home automation").

ZigBee contempla el uso de la banda ISM o los 2'4GHz, en nuestro caso usaremos la banda de 2'4GHz por ser libre en todo el mundo (la banda ISM varía según la región donde se utilice). Permite una velocidad de hasta 250 Kbit/s en campo abierto, mucho menor que otros estándares como Bluetooth o WiFi, pero suficiente para el uso en domótica que queremos otorgarle.

Para crear una red ZigBee, disponemos de tres tipos distintos de dispositivos.

Coordinador Es obligatoria la existencia de uno de estos nodos por red. Se encarga de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos, así como de los parámetros configurables de la red (uso del canal con menor ruido) y otros aspectos necesarios (promoción de la red, negociación para la inclusión de un nodo en la misma, etc.).

En nuestro proyecto, el coordinador estará incluido en el nodo central, pues igualmente es necesario que haya uno por sistema. Además, como este nodo será cableado, se puede introducir una antena ZigBee con mayor potencia (y mayor consumo) para aumentar el alcance de la red.

Router Interconecta dispositivos separados en la topología de la red. Aunque actualmente este tipo de nodo no es usado en el proyecto, es posible utilizarlo para su propósito original sin ningún tipo de modificación en el sistema, pues la funcionalidad que ofrece es exclusivamente a nivel de ZigBee y es transparente para el resto de componentes. Se puede interpretar como un proxy transparente para ZigBee.

Dispositivo final Este tipo de nodo es capaz de comunicarse con el coordinador de forma bidireccional, pero no se puede comunicar con otros dispositivos finales.

Los nodos satélite de nuestro proyecto, donde se ubican los sensores o actuadores, dispondrán de este tipo de dispositivo, pues sólo necesitan comunicarse con el nodo central.

Utilizaremos esta tecnología para transmitir datos de forma inalámbrica datos entre los nodos remotos y el nodo central, especialmente por el bajo consumo de batería que obtendremos.

La tecnología de ZigBee está especialmente indicada para su uso en la vivienda, pues dispone de un rango suficiente para una casa sin necesidad de desperdiciar energía fuera de la misma (si se necesita ampliar este rango, se puede utilizar routers ZigBee).

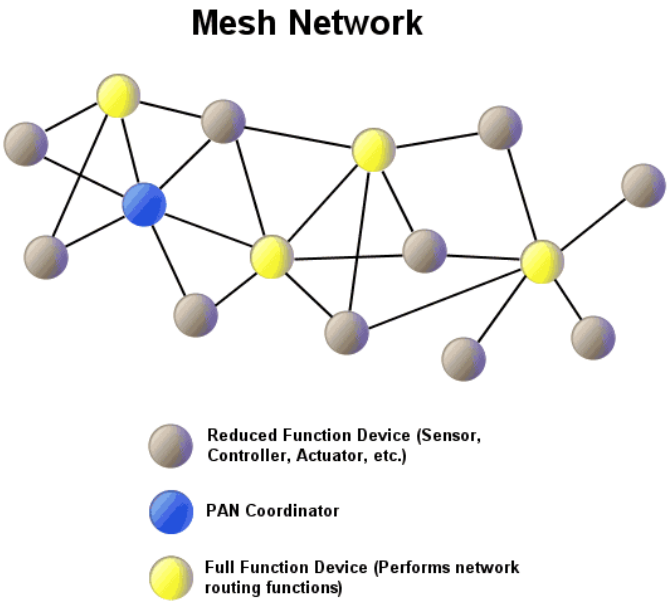


Figura 3.4: Ejemplo de malla ZigBee.

Al utilizar una tecnología inalámbrica, evitamos el uso de cables con un múltiples propósitos en el hogar:

- Una necesidad de reforma casi inexistente para la instalación (y por lo tanto, facilidad para trasladar el sistema si fuese necesario).
- Mejor estética visual, al no tener que ocultar o disimular los cables.
- Mayor facilidad de instalación, pues no es necesario saber dónde enchufar cada cable y menor posibilidad de errores durante la misma.

3.5.1. Seguridad

Al ser una tecnología de envío de datos inalámbrica, es relativamente fácil interceptar estos datos incluso sin que el emisor o el receptor se percaten, por lo que hay que tener muy en cuenta la seguridad aplicada en las transmisiones.

La capa de aplicación añade un factor extra de seguridad, pues además de añadir el nodo a la red ZigBee, hay que darlo de alta en el sistema a través de la aplicación que se está ejecutando en el nodo central. De esta manera evitamos que un nodo que no hemos instalado nosotros se añada a la red y pueda provocar alarmas inexistentes, datos extraños o, en general, mal funcionamiento del sistema. Si se implementa la auto-configuración de nodos, este aporte extra quedará inhabilitado.

Para implementar la seguridad, ZigBee utiliza claves de 128 bits en todos sus mecanismos de seguridad; en nuestro caso, otorgaremos seguridad a nivel de enlace (es posible otorgarla a nivel de red) mediante clave preestablecida, la cual ha de ser escrita en los parámetros del módulo *XBee*. Será el propio módulo el que gestione todo lo relacionado con el cifrado/descifrado de las comunicaciones.

Si tenemos un sistema al que queramos añadir un nuevo nodo, habremos de incluir dicho parámetro en la configuración del módulo, pues no es posible tener parte de las comunicaciones cifradas y otra parte en claro. Si es necesario, el sistema se puede configurar para realizar todas las comunicaciones sin cifrar, aunque esto implica exponerse a los riesgos antes citados.

3.5.2. *XBee*

XBee[21] es una marca comercial, distribuida por "Digi International", que vende chips integrados con todo lo necesario para realizar comunicaciones ZigBee (incluye pila de protocolo, antena, etc.), aunque no provee ningún mecanismo para la ejecución de código de usuario (únicamente el firmware), por lo que es necesario el uso de un microprocesador separado de este módulo.

Dentro de la marca *XBee*, hay diferentes tipos de módulos según el protocolo utilizado. Básicamente se distingue la "serie 1", el cual implementa un protocolo propio, y la "serie 2", el cual usaremos por utilizar el protocolo ZigBee[30]. Además, se pueden distinguir los módulos "Pro", con mayor consumo energético, mayor alcance y mayor precio, y los módulos "normales", que son los que usaremos en este proyecto.

3.6. *Django*

Django[25] es un contenedor de aplicaciones web que proporciona un entorno de desarrollo de código abierto y escrito en Python[18]. Contiene funcionalidades tanto de servidor de ficheros (para las páginas y elementos estáticos) como de ejecución de código Python para la creación de páginas dinámicas. Contiene también un manejador de base de datos en varios formatos, en este proyecto, al no necesitar una base de datos distribuida, se utilizará SQLite3 en este proyecto.

La estructura de *Django* está diseñada para usar los programas modularmente, reutilizando fácilmente el código, reduciendo el número de líneas de código y facilitando su comprensión global. La forma de trabajar simplificada de *Django* es la siguiente:

Para conseguir la modularidad mencionada, se pueden descomponer las distintas funciones de la página en aplicaciones; cada aplicación tendrá su propio espacio de trabajo que posteriormente se integrará en el proyecto.

En el fichero "urls.py" se detalla el mapeo de las URLs a sus respectivos manejadores, los cuales se encontrarán en el fichero "models.py" de cada aplicación; este manejador recoge la información enviada por el navegador y modifica los datos según la lógica programada. La salida de esta ejecución, o bien es enviada directamente de vuelta al navegador o, como es nuestro caso, se envían a las plantillas HTML creadas para el proyecto, en las cuales se resuelven las variables y con ello se compone la página web final que será devuelta al navegador.

3.7. Arquitectura "REST"

La arquitectura *REST* surgió a partir de la tesis doctoral[19] sobre la web escrita por *Roy Fielding*, en el año 2000. En dicha tesis se analiza el éxito de los servicios de la *World Wide Web*

y sostiene que este éxito se debe a los siguientes factores:

Protocolo sin estado Cada mensaje HTTP contiene toda la información necesaria para comprender la petición, por lo que ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes, simplificando así la lógica de la aplicación.

Operaciones bien definidas Se define todo el espectro de posibles operaciones para todos los recursos de información (por ejemplo, "post", "get", "put" y "delete" para HTTP).

Recursos con sintaxis universal Cada recurso puede ser consumido por un componente únicamente mediante su URI, no pudiendo haber más de un recurso bajo la misma URI.

Hipermedia Posibilidad de representación de datos de distinta índole (imágenes, audio) y no exclusivamente texto ha permitido una expansión más rápida al conseguir un mayor interés por parte de la población.

Por otro lado, dado que esta característica no fue tenida en cuenta en la creación de la web, se han tenido que crear elementos externos a la arquitectura inicial (plugins y reproductores) para solventarlo. La arquitectura *REST* permite el uso de hipermedia siempre que no viole ninguna de las restricciones enumeradas en el siguiente apartado.

Todo esto lleva a una gran escalabilidad e independencia con el lenguaje del cliente y del servidor, por ello se definen una serie de restricciones que una aplicación *REST* ha de cumplir siempre.

Paradigma cliente-servidor El cliente no puede guardar datos de transacciones anteriores (toda la información necesaria para acceder al servidor web ha de contenerse en la petición del cliente) ni contener lógica de ejecución más allá de la necesaria para la correcta visualización de los datos recibidos. Si el procesamiento de los datos en el lado del cliente es necesario, este tratamiento debe ser el menos posible, por ello *REST* permite ejecutar código en el cliente bajo demanda (así sólo ejecutamos el código estrictamente necesario).

Por su parte, el servidor no podrá controlar la interface de usuario ni tener acceso a los posibles estados de éste (por ejemplo, si está visualizando la página o la tiene en segundo plano).

Cacheable Para utilizar un menor uso de recursos en caso de repetición de la petición, aumentando así la escalabilidad del sistema, la respuesta del servidor ha de ser cacheable.

Comunicación transparente El cliente lanza la petición y no debe saber si hay intermediarios o no, cualquier elemento que la petición o la respuesta encuentre entre el cliente y el servidor, ha de ser totalmente transparente para la petición/respuesta.

Idealmente, estos intermediarios utilizarán políticas de balanceo de carga y cacheado para minimizar el tiempo de respuesta.

Interfaz uniforme Al no alterar la interfaz entre el cliente y el servidor, se asegura que ambos componentes podrán seguir comunicándose de la misma manera aunque sean desarrollados y actualizados de forma independiente. Esto es, la actualización del cliente, podrá alterar

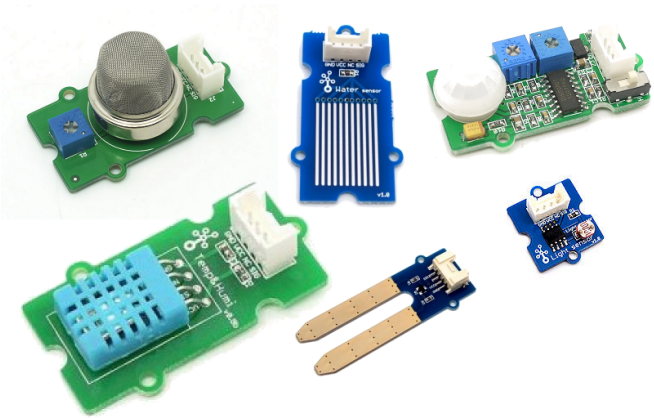


Figura 3.5: Algunos sensores utilizados en este PFC.

la forma de visualización de los datos, pero los datos que le envía el servidor serán los mismos; igualmente, si actualizamos el servidor, pueden cambiar los datos enviados al cliente, pero al enviarse de la misma manera que anteriormente, el cliente podrá extraer los datos de la misma forma con la versión anterior del servidor.

Puesto que este tipo de arquitectura simplifica enormemente la interconexión de sistemas, se ha decidido utilizarlo en este proyecto.

3.8. Sensores/actuadores domóticos

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas, posibilitando así el tratamiento de los datos a través de microprocesadores. Los sensores utilizados en Surveillance proceden todos de la empresa *seeedstudio*[26], la cual provee sensores simples de código abierto (código y esquemas de libre acceso por el público), encajando por ello muy bien en la filosofía de este proyecto.

Los componentes que se van utilizar en estre proyecto son:

Sensor de humedad En el proyecto se han utilizado dos tipo diferentes. Uno integrado junto al sensor de temperatura, más sensible, que nos va a permitir medir la humedad ambiente de la casa. Y otro menos sensible, ideado para monitorizar ambientes mucho más húmedos (ideado para controlar la humedad en macetas con plantas).

Sensor de temperatura Integrado junto a un sensor de humedad, nos permite monitorizar la temperatura ambiente. Contiene una resistencia variable con la temperatura, lo cual hace variar el voltaje obtenido al leer la salida del sensor.

Sensor lumínico Contiene una resistencia que aumenta conforme a la intensidad lumínica recibida, lo cual, junto con otra resistencia fija, hace variar el voltaje obtenido al leer la salida del sensor.

Sensor de vibración El sensor crea corriente al deformarse; si se deforma lo suficiente, crea suficiente corriente como para activar una interrupción de alarma.

Sensor de gas Al leer el voltaje devuelto por el sensor, obtenemos una medida directamente relacionada con la cantidad de elementos detectados en el aire. Hay diferentes sensores para detectar varios elementos, en este proyecto se utilizará el detector de LPG, i-butano (gas natural), metano, alcohol, hidrógeno y humo.

Sensor de agua El sensor de inundación se activará cuando el agua toque su superficie por completo (no se activa si hay mucha humedad en el aire).

Sensor magnético También conocido cómo sensor de puerta abierta o cerrada. Se compone del sensor y de un pequeño imán; si hay un campo magnético suficientemente fuerte (provocado por el imán), se cierra el circuito y fluye la corriente; si el campo disminuye, el circuito queda abierto y no obtendremos voltaje del sensor.

Cámara Comúnmente utilizado en tele-vigilancia. Dispone de un sensor de intensidad lumínica RGB y ciertos componentes ópticos.

Altavoz Actuador primario que nos permite emitir cierto tipo de sonidos.

Relé Actuador que corta o permite el paso de corriente eléctrica hacia un tercer componente.

Capítulo 4

Surveillance 3.0

En esta primera versión del proyecto vamos a abordar el objetivo de contruir un sistema domótico. Para ello utilizaremos placas *Arduino*, junto a módulos *ZigBee* para conseguir que estos sean inalámbricos.

4.1. Diseño

Para reducir el tamaño al mínimo, se ha optado por una arquitectura centralizada en la que el nodo central sea un sistema embebido, prescindiendo de ordenadores o aparatos similares en volumen. El nodo central estará constituido por un único dispositivo (Arduino para la versión 3 de Surveillance) que podrá incluir algún tipo de componente adicional (como son las *shields*) y que necesariamente tendrá capacidad de comunicación *ZigBee*. Esta arquitectura en estrella nos permite distribuir la mayor parte de la carga en los nodos satélite, lo que ayuda a no tener elementos especialmente grandes que puedan molestar al usuario (cada nodo tiene capacidad de procesamiento suficiente para simplificar la siguiente capa, pero sin comprometer su tamaño, bajo consumo, etc.) tanto en el nodo central como en los satélites. Podemos ver un ejemplo ilustrativo en la figura 4.1, en la que vemos un nodo central, una serie de satélites inalámbricos y dispositivos para la comunicación con el usuario.

Esta arquitectura tiene la gran ventaja de que una mejora en el nodo central (por ejemplo, utilizando un *XBee* "Pro" en lugar del normal) implica una mejora directa en todo el sistema (con cambiar sólo el emisor *ZigBee*, obtendríamos mayor alcance para toda la red). Otra ventaja es que el nodo central puede estar cableado, por ejemplo, a la red eléctrica para tener corriente de forma contínua en el tiempo, permitiendo mayor procesamiento y mayor potencia de emisión, aumentando así la capacidad de toda la red y no sólo de un segmento, mientras que el resto de nodos pueden comunicarse con el central de forma más liviana, pudiendo *dormir* (no realizar operaciones durante cierto tiempo), gastando menos energía y facilitando el uso de baterías.

4.2. Nodo central

Este nodo tendrá conexión a la red eléctrica y, si se quiere hacer más robusto el sistema, a un USP u otro tipo de alimentación continua. El nodo central constituye un punto único de fallo

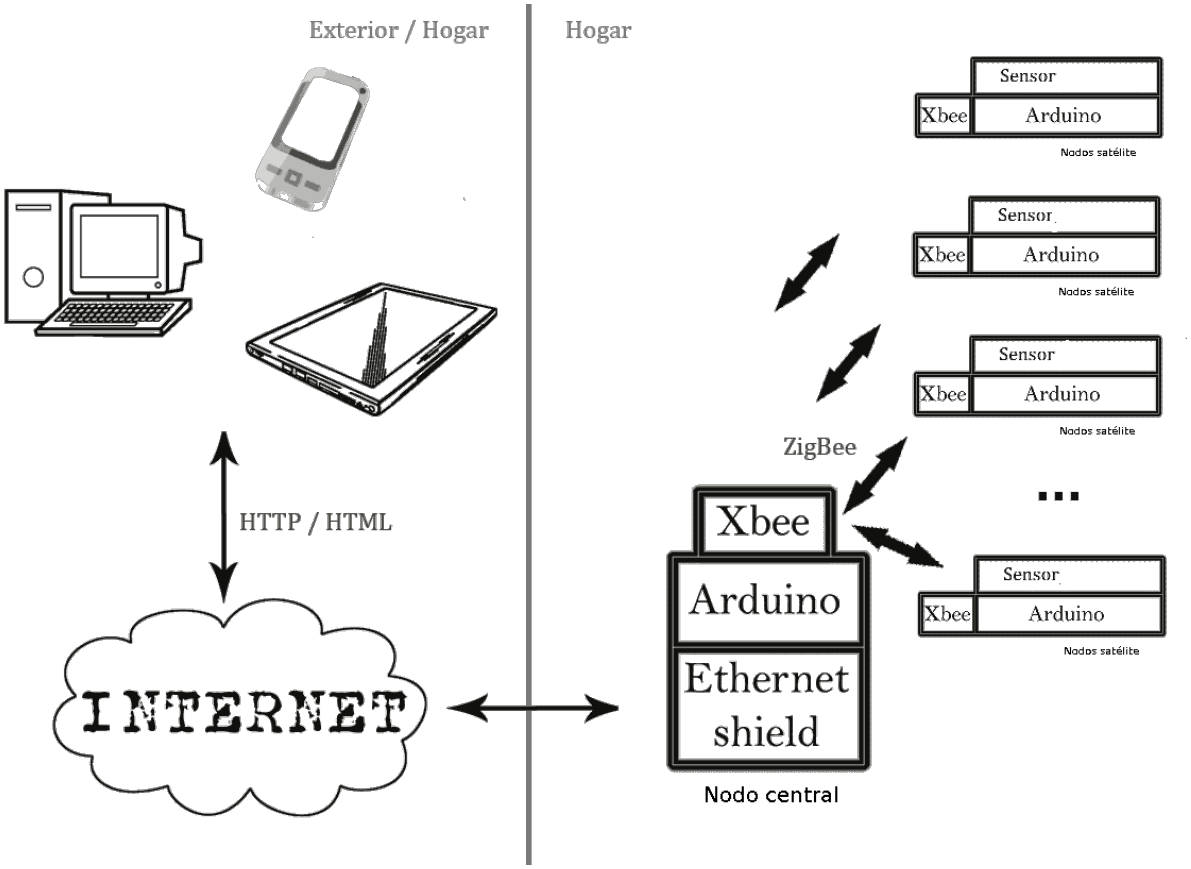


Figura 4.1: Esquema sobre la arquitectura de Surveillance 3.0.

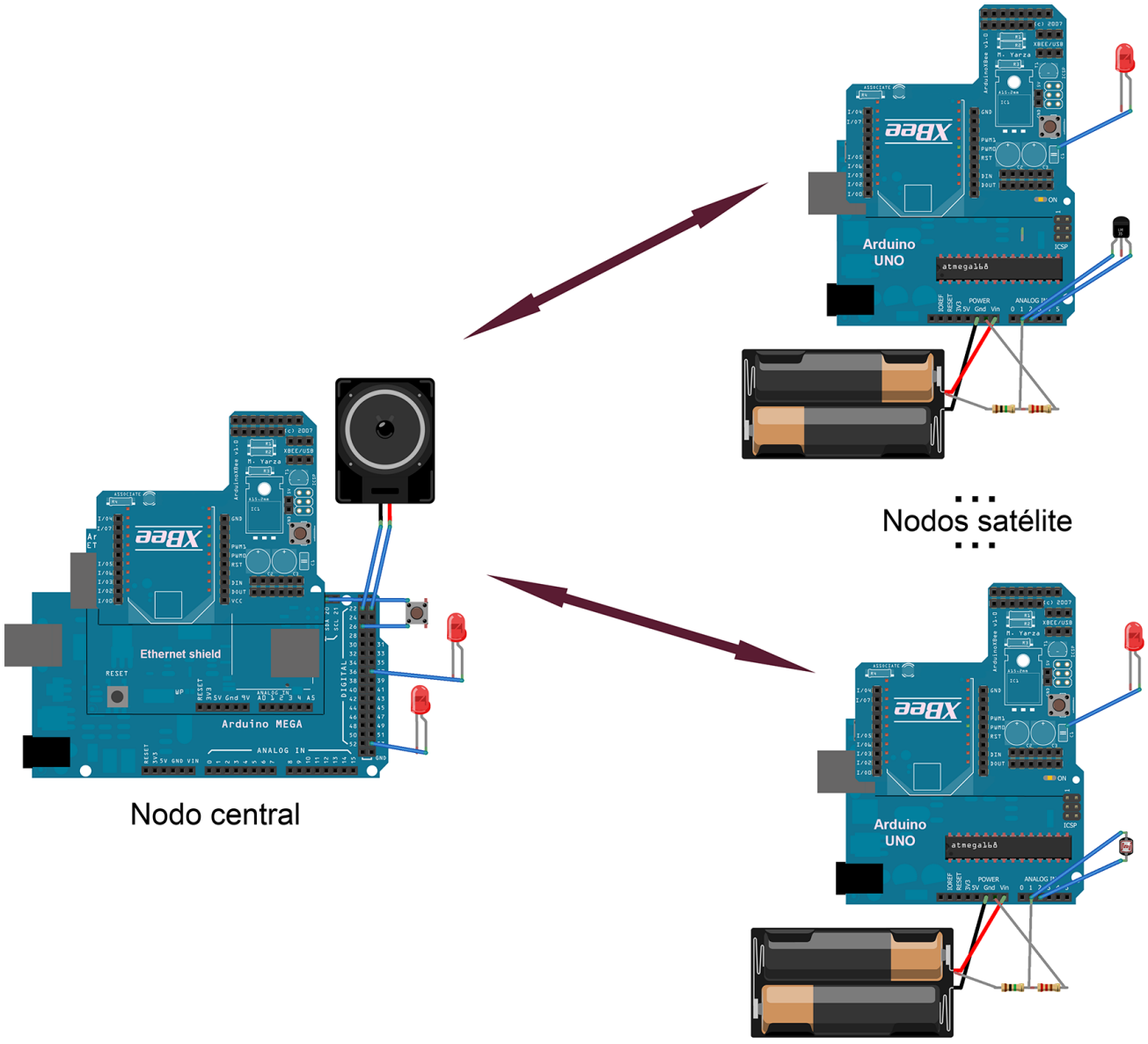


Figura 4.2: Surveillance 3.0. Recreación esquemática hardware.

y si éste deja de recibir corriente eléctrica, se apagará automáticamente y no podrá utilizarse ninguno de los servicios ofrecidos por el sistema (alarmas, medidas de variables y configuración del sistema). Es conveniente que el nodo esté conectado a Internet, si así fuere, el usuario podrá consultar el estado de las alarmas/sensores del sistema, así como configurar el mismo de forma remota. En caso contrario, las alarmas se activarán igual, con las indicaciones visuales (leds) y sonora, pero no será posible visualizar los datos de los sensores.

El nodo central está compuesto por los siguientes componentes:

Arduino Mega 2560 Se eligió esta placa como CPU debido a la gran popularidad de la placa Arduino (aunque reciente, es una plataforma muy bien valorada) y la posibilidad de conectar módulos para simplificar el ensamblado y ayudar a realizar el prototipo más rápidamente (aunque a un coste económico alto).

Además, se decidió utilizar el modelo "Mega" por tener mayor capacidad de cómputo, así como mayor cantidad de puertos de entrada y salida (entre ellos, tres puertos de comunicación en lugar de uno, como contiene la versión utilizada en los nodo satélite) de datos para realizar pruebas.

Ethernet *shield* Dado que la placa anteriormente citada no dispone de puerto Ethernet, y siendo este puerto necesario para la comunicación con el usuario en esta versión (dicha comunicación se realizará íntegramente mediante HTTP), se decidió la utilización de una expansión (*shield*) para dotar el nodo central de dicha conectividad.

Esta expansión, además de añadir el puerto físico, contiene un circuito integrado que implementa la abstracción del protocolo HTTP, permitiendo crear clientes o servidores web sin necesidad de implementar este protocolo. Por supuesto, el resto de interacciones y protocolos deben ser programados. En este caso, al ser un servidor web, he implementado el mapeo de URLs, el manejo de datos y el código HTML de las páginas web a presentar.

XBee shield* y *XBee Z-24B Para poder realizar la comunicación mediante *ZigBee* entre todos los nodos, se decidió utilizar los módulos *XBee Z-24B*[21], los cuales implementan tanto la pila *ZigBee* como una antena PCB para realizar la comunicación.

Dado que la comunicación, tanto del módulo Ethernet como del *XBee* con la placa Arduino se realiza a través los pines serie de la placa, es necesario modificar la conexión estándar del *shield XBee* para que utilice otros pines de comunicación distintos a los de la *shield* Ethernet. Por supuesto, el código debe reflejar en qué pines se ha conectado y ser escrito en consecuencia.

Además, hay que tener en cuenta que ambas expansiones (*XBee* y Ethernet) necesitan conexión ICSP, aunque es utilizado por ambos únicamente como circuito de alimentación auxiliar. Al no ser estos pines propagados (únicamente permiten la conexión con un *shield*, pero no con dos) por ninguna de los *shields*, se han de conectar los tres conectores ICSP de forma artesanal.

Otros componentes Se precisan una serie de componentes más genéricos, pero necesarios para el funcionamiento del sistema: una tarjeta SD, para contener las páginas web que harán de

interfaz con el usuario; un altavoz genérico (2'5W), para poder emitir una alarma sonora; dos leds, para indicar si el sistema se encuentra en estado de aviso o alarma; un botón, para restear las alarmas o avisos (este elemento es prescindible, pues el reseteo se puede realizar mediante la interfaz de usuario), y el resto de pequeños componentes necesarios para el funcionamiento del hardware (resistencias, condensadores, etc.).

Se hará uso tanto del conector de alimentación, a través de un trasformador, como del conector Ethernet, a través del cual conectaremos el nodo con un *router*, un switch o un ordenador, por lo que es necesario disponer de los cables y el transformador pertinente (es posible conectar un cable USB en lugar del transformador para alimentar la placa).

A la hora de analizar la arquitectura software del nodo central, podemos dividirla en tres grandes partes: la comunicación con los satélites y la comunicación web con el usuario y el procesamiento de los datos enviados por los satélites.

4.2.1. Comunicación con los nodos satélite

La parte correspondiente las comunicaciones físicas mediante *ZigBee* 4.3.3 se realiza a través del módulo *XBee*, el cual recibe/envía los bytes por la salida "serie 1" del Arduino, pudiendo configurarse para cifrar los datos de forma automática, entre otras utilidades. Hay que destacar que las cabeceras de *ZigBee* y todo lo necesario para las comunicaciones mediante este protocolo es construido por el módulo, pero es necesario utilizar un protocolo de nivel de aplicación para la comunicación entre nodos, por ello se ha diseñado el protocolo propio WHAP, el cual se describe con detalle en la sección 4.3.3.

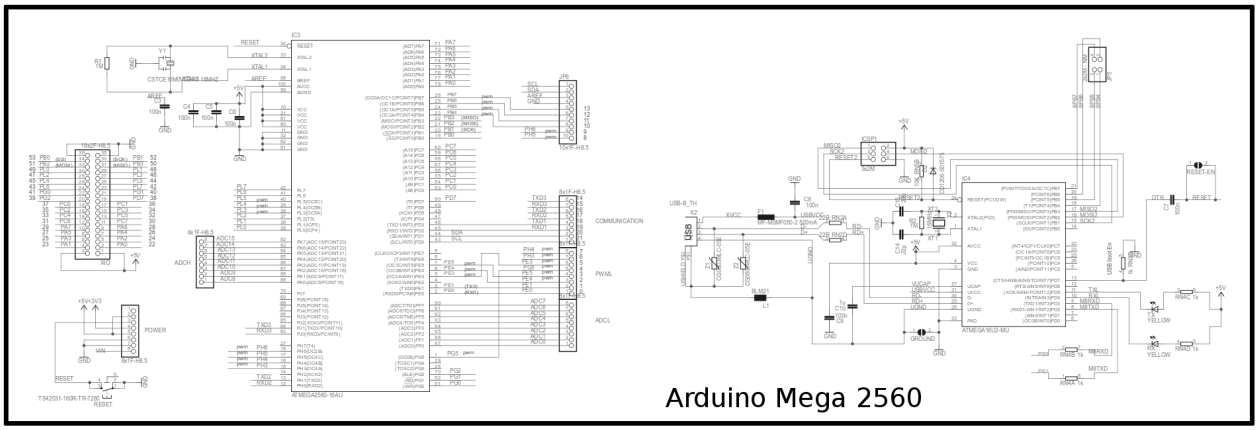
Para tomar la decisión de qué tecnología utilizar en el envío de datos, se tuvo en cuenta principalmente el factor de consumo eléctrico, siempre y cuando la tasa de transmisión fuese suficiente para nuestros propósitos. Se analizaron las tecnologías *ZigBee*, *Bluetooth* y Wi-Fi, como se muestra en la siguiente tabla:

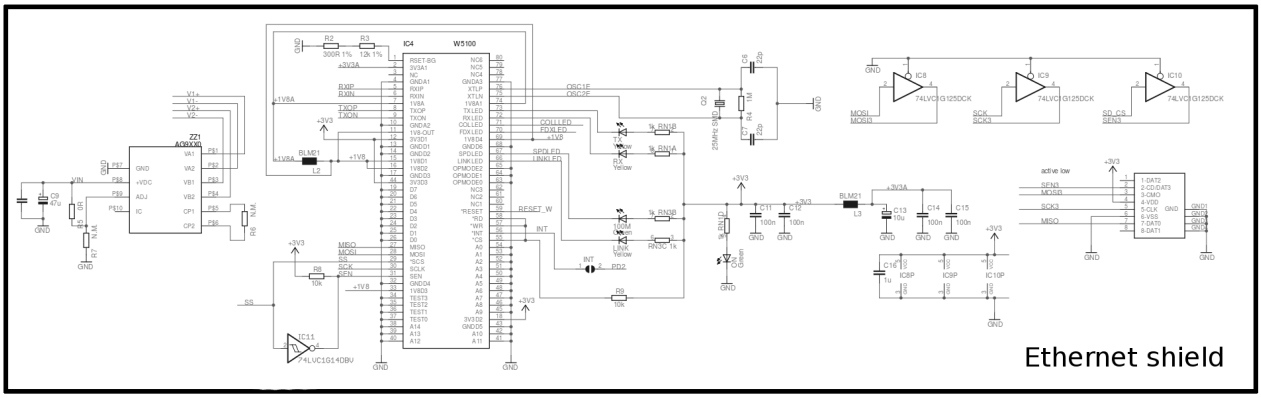
Tecnología	Velocidad	Consumo	Alcance	Precio
Wi-Fi	300 MBps	85 mA	100 m	Medio
Bluetooth	1 MBps	40 mA (0.2mA en reposo)	10 m	Bajo
ZigBee	250 KBps	30 mA (3uA en reposo)	50 m	Medio

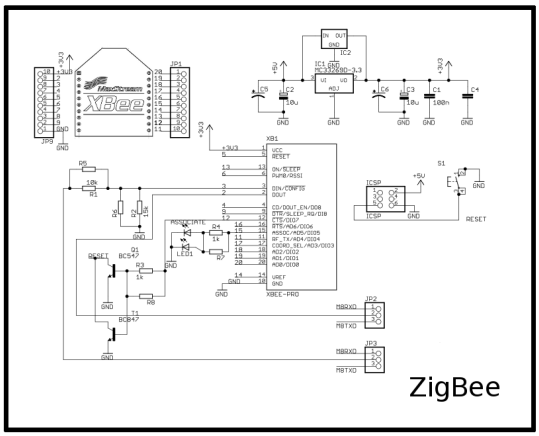
Dado que *ZigBee* ofrece suficiente ancho de banda para realizar el envío de los nodos sensores (se envían sólo unos pocos bytes, en el protocolo WHAP), ofrecía mucho menor consumo que sus competidores y, aunque se precio es más alto que *Bluetooth* (dado que la tecnología está menos madura y hay menos fabricantes), se optó por esta tecnología.

4.2.2. Procesamiento de datos, aplicaciones

Cada nodo satélite envía los datos del sensor y la batería (o sólo al batería en el caso de un nodo de alarma) cada 2 minutos. A la llegada de los datos enviados por el satélite, el nodo central comprueba si se trata de una alarma, así como el nivel de batería del nodo correspondiente; cada 5 minutos (un poco más del doble de tiempo entre envíos), el nodo central lleva a cabo unas







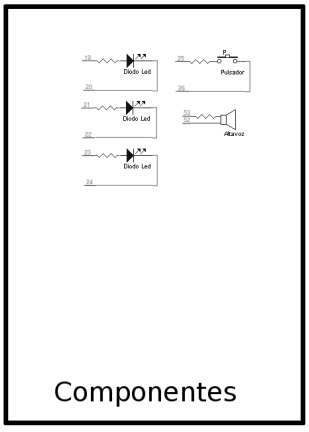


Figura 4.3: Esquema hardware del nodo central para Surveillance 3.

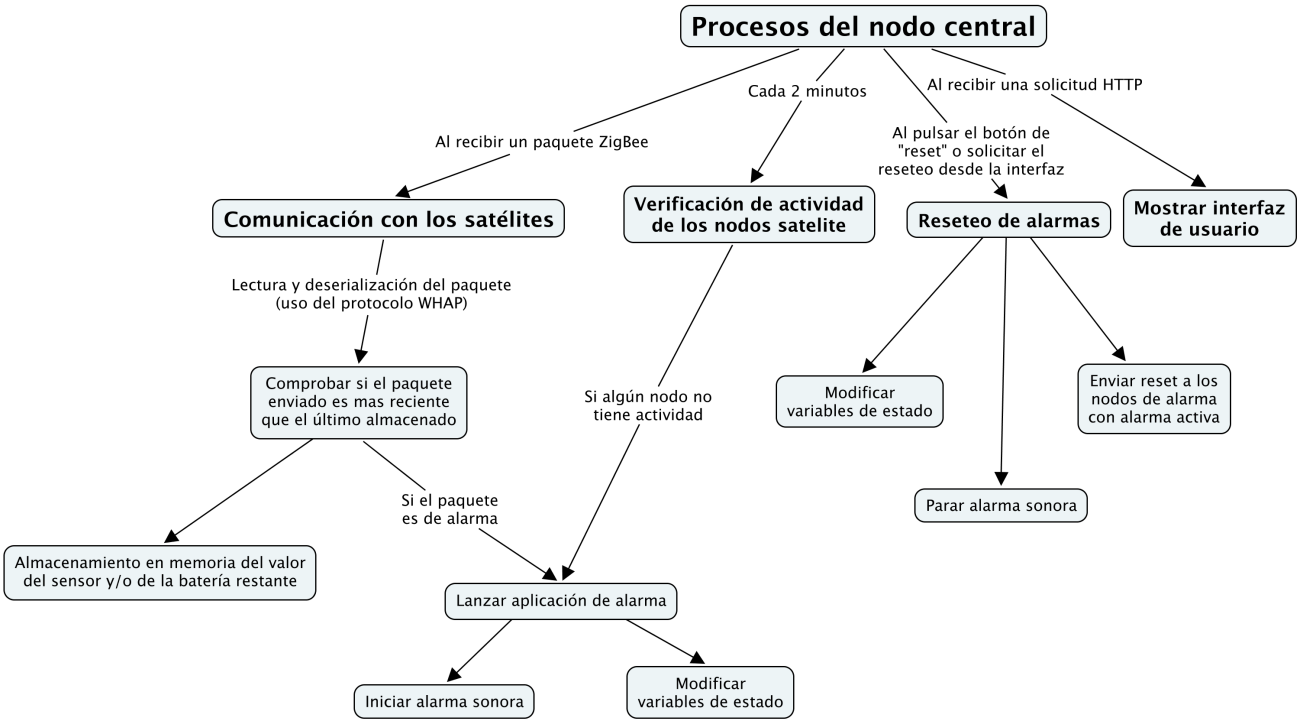


Figura 4.4: Esquema lógico del software programado en Surveillance 3.

operaciones para comprobar que los datos que ha recibido son actuales y no hay ningún nodo que no se pueda comunicar con el satélite o no esté funcionando correctamente. En caso de que algunas de estas comprobaciones no fuese satisfactoria, se iniciaría el modo de alarma o de aviso correspondiente.

El flujo concreto para el intercambio de información por *ZigBee* es el siguiente:

Captura de datos Cada satélite manda la información de su sensor y su batería restante cada 2 minutos; cuando esta información llega al nodo central, éste comprueba que el dato es más reciente que el que tiene almacenado (se utiliza el campo *data timestap* del protocolo WHAP). Si no fuese así, el paquete es descartado. Si cumple con este requisito, en caso de tratarse de un paquete de alarma, activará el protocolo de alarma. Si es un paquete de datos, almacena esta información sobrescribiendo la anterior y comprueba si la tensión de la batería del nodo remoto es superior a 6'5V. Si no es así inicia la función de aviso por batería baja.

Adicionalmente, cada dos minutos, el nodo central revisa que tiene información reciente de todos los nodos activos (la carga de esta distribución se reparte a lo largo de los dos minutos), si encontrase que un nodo remoto no le ha enviado información en más de cinco minutos, inicia la función de aviso de nodo inactivo.

Aplicación de alarma Si durante el procesamiento inicial del paquete, el nodo central detecta que el dato enviado es una alarma, se inicia el protocolo de alarma. Se han programado alarmas visuales tanto en el nodo central como en los satélites y una alarma sonora en el

nodo central. Cuando se inicie el protocolo, los leds rojos de ambos nodos se iluminarán permanentemente (no así los de otros nodos que no hayan activado su alarma) y el altavoz del nodo central comenzará a emitir un "pitido" intermitente.

La correspondencia entre la alarma/alerta y qué nodo la provocó puede consultarse a través de la interfaz web o visualmente localizando el satélite con el led encendido. Una vez detectada y eliminada si procede la causa de la alarma, el sistema puede volver a su estado normal bien mediante la pulsación del botón hardware de reseteo de alarmas o bien utilizando la opción de reseteo de alarmas en la interface web. Este acto provocará la desactivación del led y del altavoz hasta que se produzca una nueva alarma y el envío mediante *ZigBee* de la orden de parada de alarma a los nodos satélites, lo cual permite que el nodo satélite desactive su led de alarma y vuelva a funcionar con normalidad.

Si se limpia la alarma sin haber eliminado la fuente de la misma, el nodo satélite volverá a enviar el paquete de datos con alarma, lo cual provocará que se vuelva a activar el protocolo de alarma.

4.2.3. Interfaz web de usuario

El usuario puede consultar el valor de la última información enviada por el satélite, así como si el sistema tiene alguna alarma activa, a través de la interfaz de usuario, explicada a continuación.

El nodo central es el único nodo capaz de comunicarse directamente con el usuario, todos los demás nodos que requieran comunicación con el usuario tienen que utilizar el nodo central como punto de acceso; de esta forma, el usuario ha de dirigirse siempre al nodo central para realizar cualquier acción.

El usuario humano tiene habilitada la comunicación mediante una interfaz web a través del conector Ethernet, únicamente se utilizan los protocolos HTTP 1.1, HTML 4.1 y CSS 2.1, siendo por ello compatible con todos los navegadores web que están en el mercado actualmente (InternetExplorer 6 en adelante). Estas páginas son almacenadas en la tarjeta SD acoplada a la "*shield* Ethernet". Por ser una interfaz web, podemos acceder tanto desde la red LAN donde resida el nodo central (para ello utilizaremos la IP del nodo central y el puerto 80) como desde internet (para ello, tendremos que permitir el tráfico a través del router desde el exterior hasta el nodo central y conectarnos a la dirección IP del router y el puerto que se haya configurado para dicha conexión).

El servidor en esta versión comprende los métodos GET y POST, pudiendo utilizarse el método GET para recuperar cualquiera de los ficheros existentes en la tarjeta SD y el método POST únicamente para aquellas páginas que requieren de información adicional por parte del usuario (cambio en la configuración, autenticación...).

Para poder utilizar el servidor web, es necesario autenticarse. Las credenciales (usuario y contraseña) se encuentran como variables en el código del Arduino, pudiendo modificarse la contraseña a posteriori sin necesidad de recompilar el programa, para ello dichas variables se almacenan en la memoria EEPROM, recargándose de ésta si se reinicia la placa. Si el usuario no está autenticado, únicamente tiene acceso a la página de autenticación, al fichero CSS y a las imágenes necesarias para visualizar correctamente dicha página.

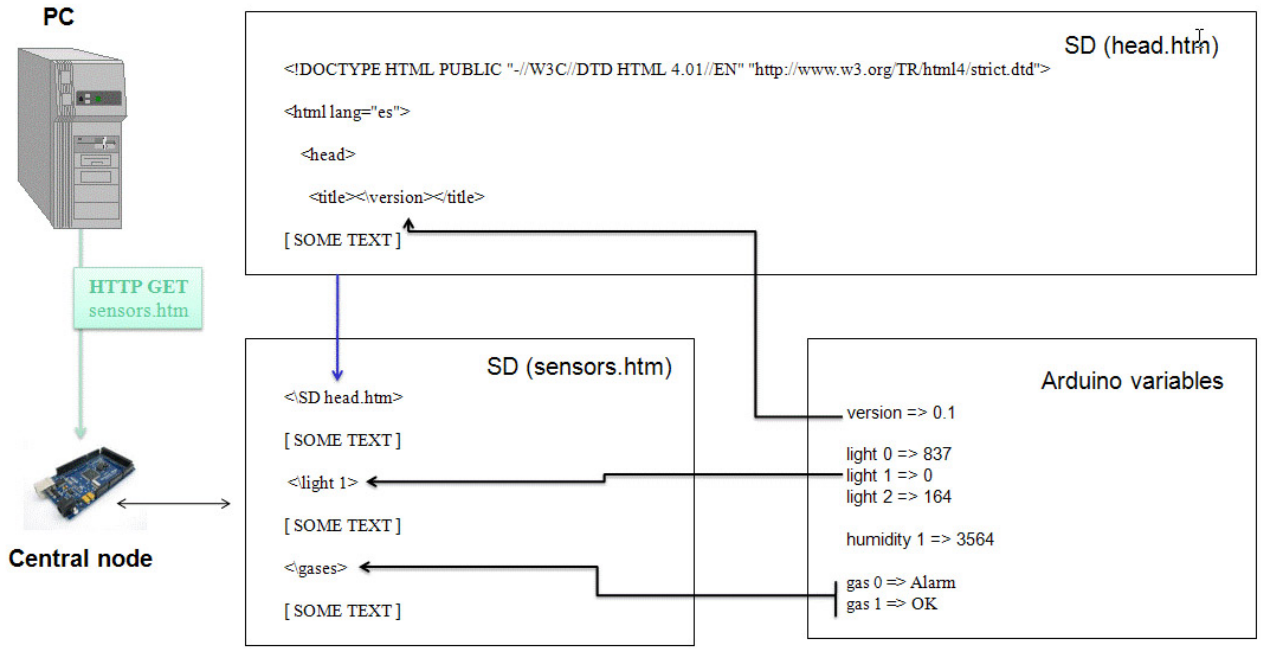


Figura 4.5: Servidor Web. Composición dinámica de una página HTML mediante tags.

Para almacenar el código HTML de las páginas web del proyecto, se utiliza una tarjeta SD introducida en Arduino en el conector que proporciona la "*shield* Ethernet", ya que este código ocupa bastante espacio. Esta actuación tiene tres implicaciones fundamentales, por un lado, la página tardará más en procesarse y mostrarse al tener que depender de un soporte externo, lo cual es más lento que tenerlo cargado en memoria, por otro lado, ahorramos memoria RAM o EEPROM y además la página puede ser modificada variando el fichero contenido en la SD, sin necesidad de recompilar el programa.

El código HTML correspondiente a las páginas de error se escribe en el código de la aplicación, en concreto, se aloja en la memoria EEPROM del microprocesador para no saturar la memoria del programa. Esto implica que para modificar las páginas de errores que se muestra al usuario, hay que recompilar y subir el programa entero, por otro lado, si hay algún tipo de problema al acceder a la tarjeta SD, la página de error se muestra correctamente.

Las páginas web pueden componerse de forma dinámica utilizando distintas cadenas de caracteres específicos o tags, la relación de tags se puede encontrar en `http://www.jderobot.org/index.php/D.castellanob-pfc`

Para distinguir los tags de este proyecto de cualquier otro tipo de cadena de caracteres, los tags han de comenzar siempre con los caracteres " < " y " \ " y finalizar con el carácter " > " (ej. <*tag*>); en la *figura 4.3* se puede observa un ejemplo de sustitución de dichos tags por su valor correspondiente en una parte del código HTML. Se ha creado un esquema ilustrativo, plasmado en la figura 4.5.

El número de conexiones simultáneas a la interfaz web es de 4, siendo este número determinado por las restricciones impuestas en la implementación por parte de Arduino del driver de la conexión Ethernet. Este número puede aumentar o disminuir si fuese necesario modificando

Figura 4.6: Interfaz de usuario en Surveillance 3. A la izquierda, pantalla de login (autenticación previa fallida), a la derecha, panel de configuración.

el código, aunque el fabricante recomienda no modificarlo.

Se ha tratado de componer páginas simples y ligeras para facilitar el uso y a su vez maximizar el rendimiento; dichas páginas se almacenan en una tarjeta SD, que es gestionada a través de la *shield* Ethernet y sus bibliotecas correspondientes. Debido a restricciones impuestas por estas librerías, los nombres de los ficheros deben ajustarse al formato DOS, es decir, como máximo han de tener un nombre de 8 caracteres y una extensión de 3 caracteres, siendo indiferente el uso de mayúsculas o minúsculas.

Hay dos páginas especiales que no están contenidas en la tarjeta si no en el propio programa, una para mostrar el error 404 (fichero encontrado) y otra para mostrar el error 500 (error interno), pues podría requerirse mostrar estos errores cuando no haya tarjeta o si la tarjeta se extrae.

Cómo se ha comentado anteriormente, hay poca RAM sobrante en el chipset (300 bytes aprox.). Para intentar paliar este problema se ha intentado maximizar el uso de la memoria de programa y de la EEPROM, en las cuales se han incluido la mayoría de las variables estáticas (especialmente los *strings* necesarios para la interfaz de usuario, puesto que ocupan bastantes bytes) y todas las opciones configurables por el usuario. Además de salvar memoria, esto nos ha permitido preservar las últimas opciones del usuario aunque el nodo se reinicie.

En cuanto a la interfaz en sí, se exponen las siguientes páginas para interactuar con el usuario:

Login Esta es la primera página que visualizaremos al entrar al sistema. Si no hemos iniciado sesión con anterioridad, el sistema muestra un formulario pidiendo el nombre y la contraseña del usuario. En caso de no ser correctos, se muestra de nuevo la misma página. En caso de ser correctos, se genera una cookie con un identificador de sesión (número entero,

Figura 4.7: Interfaz de usuario en Surveillance 3. A la izquierda, vista de los nodos sensores, a la derecha, vista de los nodos alarma, con alarma activada.

con valores comprendidos entre 1 y 2147483647) que es devuelta al navegador para mostrar en las sucesivas interacciones y se muestra la página "Index". Imagen 4.6

Index (/) Esta página contiene un mensaje de bienvenida y otro indicando que todos los sistemas están correctos o que hay alguna alarma activa (depende del estado del sistema), así como los enlaces en un menú lateral al resto de las páginas. Este menú está ubicado a la izquierda y está contenido en todas las páginas, excepto en la de login.

Alarms Se muestran los sensores alarma, el tipo de sensor acoplado, identificador (número asociado) y su estado ("OK" o "ALARM"), así como la batería restante de los mismos. Imagen 4.7

Sensors Esta página muestra el último valor recogido de todos los sensores configurados en el sistema y la batería restante. Imagen 4.7

SD Files Muestra una lista de los archivos almacenados en la tarjeta SD, así como la opción de editar (sólo para ficheros de texto) o eliminar cada archivo. También permite crear ficheros de texto plano nuevos.

Esta página permite al usuario modificar la interfaz de usuario (las páginas web) sin necesidad de parar el sistema y sin necesidad de un ordenador u otro elemento externo al sistema. También permite crear listas. Por ejemplo, podemos crear el fichero "'COMPRAR.txt", añadir elementos a través del editor que ofrece la interfaz y consultarla desde el supermercado.

Configuration Aquí podemos visualizar y modificar los parámetros del sistema, en concreto, podemos modificar la IP donde presentamos el servicio (el nodo no dispone de DHCP, por lo que este parámetro tiene que estar informado y ser único en la red) [1]. También permite modificar la hora y fecha del sistema, el número de nodos instalados de cada tipo (si se presentan más nodos de los configurados, los paquetes "sobrantes" son descartados) y por último, permite modificar la contraseña del usuario "*admin*". Imagen 4.6

Reset alarms Este enlace está disponible únicamente cuando hay alguna alarma activa. Cuando se le llama, introduce el estado *OK* tanto en el nodo central como en todos los nodos de tipo alarma y les envía un "reset" para que vuelvan al estado de reposo. Posteriormente envía al navegador la página "index".

Logout Envía al navegador la orden de eliminar la cookies referentes a la página web y elimina el identificador de sesión del servidor, de este modo, el usuario no puede volver a entrar si no se autentica nuevamente. Después envía al navegador la página "index".

Si el usuario intenta acceder a cualquier página sin entregar la cookie con el ID de sesión, se muestra la página de login con el formulario de entrada para autenticarse. Los únicos elementos que pueden obtenerse sin estar autenticado son la página de login y los elementos estáticos necesarios (css, png, gif y jpg).

4.3. Nodos satélite

Los nodos satélite están compuestos por un *Arduino UNO*, un *XBee shield* con su módulo Z24-B correspondiente[21], un led, una batería (una pila 6lr6), un sensor y los pequeños componentes necesarios par su funcionamiento. Tal cómo se ha expuesto anteriormente, los nodos satélite han de ser lo más simple posible, pues, entre otras consideraciones, son nodos inalámbricos que necesitan consumir lo mínimo posible.

Debido a los diferente tipos de sensores implementados, podemos distinguir tres tipos de nodos satélite; de alarma (generan o dejan de emitir tensión eléctrica cuando ocurre un evento), analógicos (la cantidad de tensión variará conforme varíe la magnitud que estamos midiendo) y digitales (es necesario el uso de un protocolo para obtener los datos).

Los sensores soportados en esta versión son los siguientes: Temperatura y humedad, humedad (para macetas), gas (o humo), intensidad lumínica, agua, vibración y switch magnético (puerta abierta). Debido a restricciones del protocolo, existe un limite de 256 nodos satélites de cada tipo para cada sistema desplegado.

Podemos dividir el procesamiento de estos nodos en dos segmentos, la lectura y procesado de los datos y la comunicación con el nodo central.

[1] En la página "Configuration" se permite modificar la máscara de red y el gateway/router de salida a Internet (estos dos últimos parámetros se pueden obviar, pues estaban pensados para utilizar los protocolos NTP y DHCP, los cuales finalmente no fueron implementados en esta versión de Surveillance)

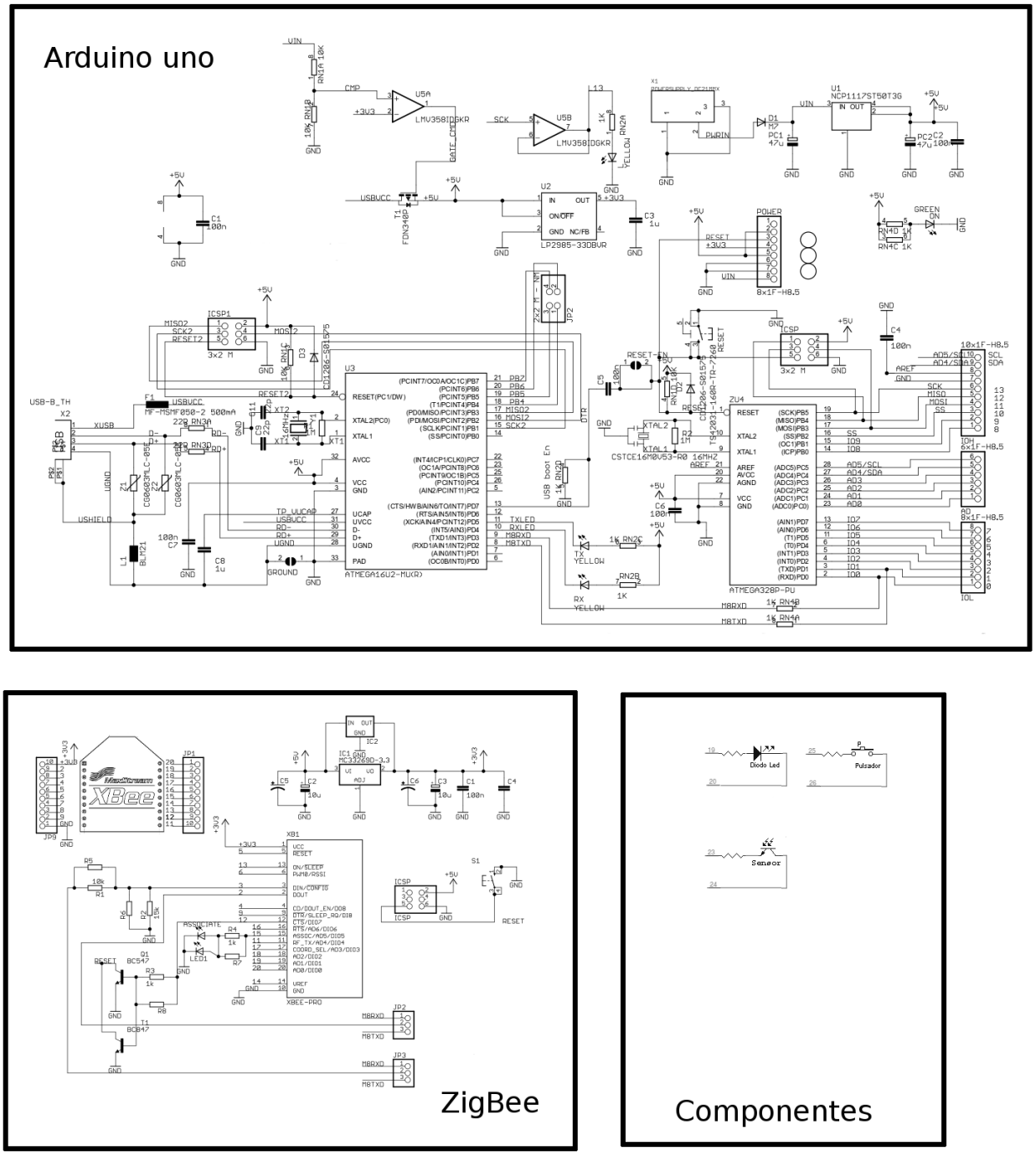


Figura 4.8: Esquema lógico del software programado en Surveillance 3.

4.3.1. Obtención de datos

La lectura de datos en los sensores *analógicos* es simple (utiliza el conversor A/D para leer el valor del sensor y de la batería). Para los sensores de alarma, el sensor lee y envía de forma periódica el nivel de batería, pero la generación de alarmas se realiza de forma síncrona (el sensor al generar la alarma despertará al micro y éste la envía inmediatamente al nodo central).

Para lograr minimizar el consumo, los nodos inalámbricos están "dormidos" el mayor tiempo posible, despertándose para realizar la medición del sensor, el envío de los datos al nodo central y la posible recepción de mensajes del nodo central, en ese orden. Los nodos satélite únicamente permanecen "despiertos" y con el módulo *ZigBee* escuchando activamente si se encuentran en estado de alarma o aviso, de esta manera, pueden recibir de manera síncrona los mensajes del nodo central (por ejemplo, de que la alarma ya ha sido solucionada por el usuario).

Aunque es posible agrupar varios elementos de sensado (alarmas o sensores) en un único nodo, por simplicidad del modelo, en este proyecto sólo se permite un único sensor o alarma por dispositivo, a excepción del sensor de humedad y temperatura que, si bien físicamente es un único sensor, nos otorga información de dos variables diferentes. Esta "doble" información de un único sensor está contemplada en el software únicamente para el sensor de temperatura y humedad [2].

4.3.2. Envío de datos

Los nodos satélite envían la lectura de la batería restante cada 2 minutos, si la tensión de la batería es inferior a 6'5V, se enciende el led de alarma. Además, si dicho nodo tiene un sensor que no sea de alarma, envía la información del sensor en el mismo paquete hacia el nodo central, donde ambas medidas son procesadas. Los satélites con sensor de tipo alarma envían un paquete inmediatamente y encienden el led de alarma, toda vez que la alarma se active.

Los sensores Surveillance 3.0 presentan dos puertos de entrada (referencia de 5V y tierra, que son proporcionados directamente por la placa) y un puerto de salida (la señal que proporciona el sensor), adaptada al voltaje y amperaje requerido por el Arduino, por lo que no necesitan ningún tipo de hardware adicional.

La implementación del código para obtención de los datos depende del tipo de sensor utilizado:

Temperatura y humedad Para leer los valores devueltos por el sensor de temperatura y humedad, se utiliza el protocolo DHT. El fabricante proporciona una implementación del protocolo en C para Arduino; posteriormente modifiqué dicho código para adaptarse mejor a nuestro sistema, ya que sólo nos interesaba la parte entera de los valores que proporciona el sensor.

[2]Si se desea implementar la posibilidad de tener un nodo satélite con varios elementos diferenciados, sería posible si "engañamos" al sistema emulando dos nodos satélites totalmente diferenciados, o lo que es lo mismo, que un mismo nodo satélite tenga tantos ID's como elementos de sensado desee utilizar.

Hay que tener en cuenta que en ese caso, el nodo satélite necesitará invertir más tiempo en comprobar el estado del sensor y en enviárselo al nodo central, con lo que su consumo de batería aumentará bastante.

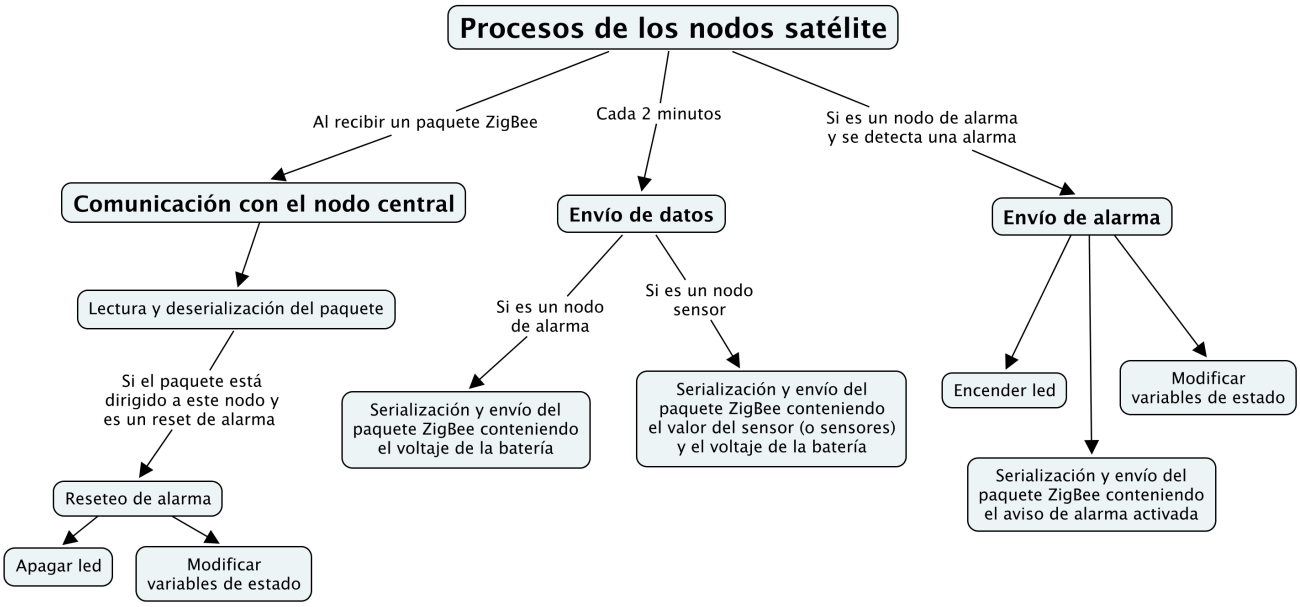


Figura 4.9: Esquema lógico del software programado en Surveillance 3.

Analógicos Para estos sensores, el código lee el voltaje que devuelve el sensor a través del conversor A/D del chip del Arduino, esta lectura se traduce en un bit (0-255, para un voltaje de 0-5V aprox.).

Los sensores no están calibrados, por lo que no hay una correspondencia lineal entre el valor obtenido en la lectura y el valor real de la magnitud física que representa.

Alarmas En este caso, si el sensor está en estado de alerta (depende del sensor, a nivel lógico alto o bajo), se iniciará el protocolo de alarma.

Este proceso se podría optimizar utilizando interrupciones por hardware, pero en este caso no ha sido posible utilizarlas, ya que la librería para la comunicación con los módulos *XBee* (*ZigBee*) hace uso de estas mismas interrupciones, entrando en conflicto a la hora de ejecutar; en concreto, cada vez que se realizaba un envío, por ejemplo, con el nivel de batería, el satélite interpretaba que se ha activo la alarma, aparte de otros comportamientos anómalos resueltos al comprobar el sensor mediante polling.

El sensor de temperatura y humedad inicia su modo de bajo consumo cuando no se utiliza, por ello requiere el uso de un protocolo propio para poder extraer los datos del sensor, lo cual obliga a que el código de este satélite sea diferente al de los demás satélites; en este caso, cada 2 minutos se envía al nodo central la información de la batería restante (1 byte), de la temperatura (2 bytes) y de la humedad (2 bytes), en ese orden.

Los sensores de tipo "analógicos", como el sensor de intensidad lumínica, el sensor de humedad y el sensor de agua envían la información de la batería restante (1 byte), la información correspondiente al sensor (2 bytes) cada 2 minutos. Los sensores tipo "alarma", como el sensor vibración, el sensor de agua y el sensor de puerta abierta envían únicamente la información de su batería (1 byte) cada 2 minutos; si sucediese una alarma, esta seria enviada inmediatamente.

Adicionalmente, el sensor de gas se puede utilizar como sensor analógico (para controlar la calidad del aire) o a modo de alarma; el código distribuido contempla a este sensor como analógico, aunque tiene comentado el código necesario para hacerlo funcionar como alarma.

Todos los nodos encenderán su led de alarma si su nivel de batería es bajo o, en el caso de nodos alarma, si sucede una alarma. Vuelven a operar normalmente (con el led apagado) una vez que el nodo central le indique a través de *ZigBee* que el usuario da por resuelto el problema (si no estuviese resuelto, el nodo satélite volverá a iniciar la alarma o alerta).

4.3.3. *ZigBee* y WHAP

La comunicación entre el nodo central y los satélites se realiza a través de *ZigBee*; dicho protocolo es implementado por el módulo *XBee*, por lo que no ha sido necesario implementarlo en el código. Se utiliza el protocolo propio WHAP4.3.3 (el cual explicaremos en este apartadp) para el envío de alarmas, intercambio de información, etc. entre los nodos.

Hay que tener en cuenta que debido a la baja tasa de envío de bits que nos puede proporcionar *ZigBee*, este protocolo no es adecuado para el envío masivo de datos (por ejemplo, streaming de vídeo), por lo que para realizar estas tareas se ha de utilizar otros métodos.

Para configurar los módulos *ZigBee* (*XBee*), utilizaremos el programa *X-CTU* proporcionado por el fabricante con los parámetros mostrados en la figura 4.10. A continuación se comentan los parámetros más importantes:

Los parámetros con letra negra se autoconfiguran al arranque y no pueden ser modificados por el usuario.

El parámetro señalado con la flecha verde rayada (Function set) de la figura 4.10 indica cómo se programará el nodo. Necesitamos un nodo programado como “coordinator” (nodo central), mientras que el resto se programarán como “End device” (nodos satélites).

El parámetro señalado con la flecha roja con círculosde la figura 4.10 indica el número identificador de la red. Este número tiene que ser idéntico para todos los nodos de la misma red ya su vez debe ser unívoco entre todas las redes que tengan alcance entre sí.

El parámetro señalado con la flecha azul con cuadrados de la figura 4.10 (sólo en los nodos satélites) debe configurarse a ”1”para que el nodo se busque y se asocie con el nodo central (coordinador) en cada arranque (por defecto este parámetro es 0).

Los parámetros señalados con las flechas amarillas con aspas indican la dirección *ZigBee* de destino (análogo a una dirección MAC). Se ha de utilizar DH=0 y DH=FFFF en el nodo central (estos parámetros indican que se envía el paquete a toda la red) y DH=0 y DL=0 en los nodos satélites (estos parámetros indican que se envía sólo al nodo central).

El protocolo WHAP (Wireless Home Automation Protocol) es un protocolo diseñado en el marco de este proyecto para resolver las necesidades específicas del mismo.

Está especialmente pensado para trabajar sobre *ZigBee* (las características del protocolo encajan en sus restricciones), aunque, por supuesto, se puede utilizar sobre otros estándares menos restrictivos en cuanto a número de bytes por paquete como X10, KNX, Ethernet, etc.

Dependiendo del contexto, puede ser utilizado como un protocolo de transporte (similar a UDP, pues no contempla una garantía de entrega) o de aplicación (el caso de este proyecto, pues el protocolo *ZigBee* de los *XBee* implementa la capa de red y transporte y nos da garantía de

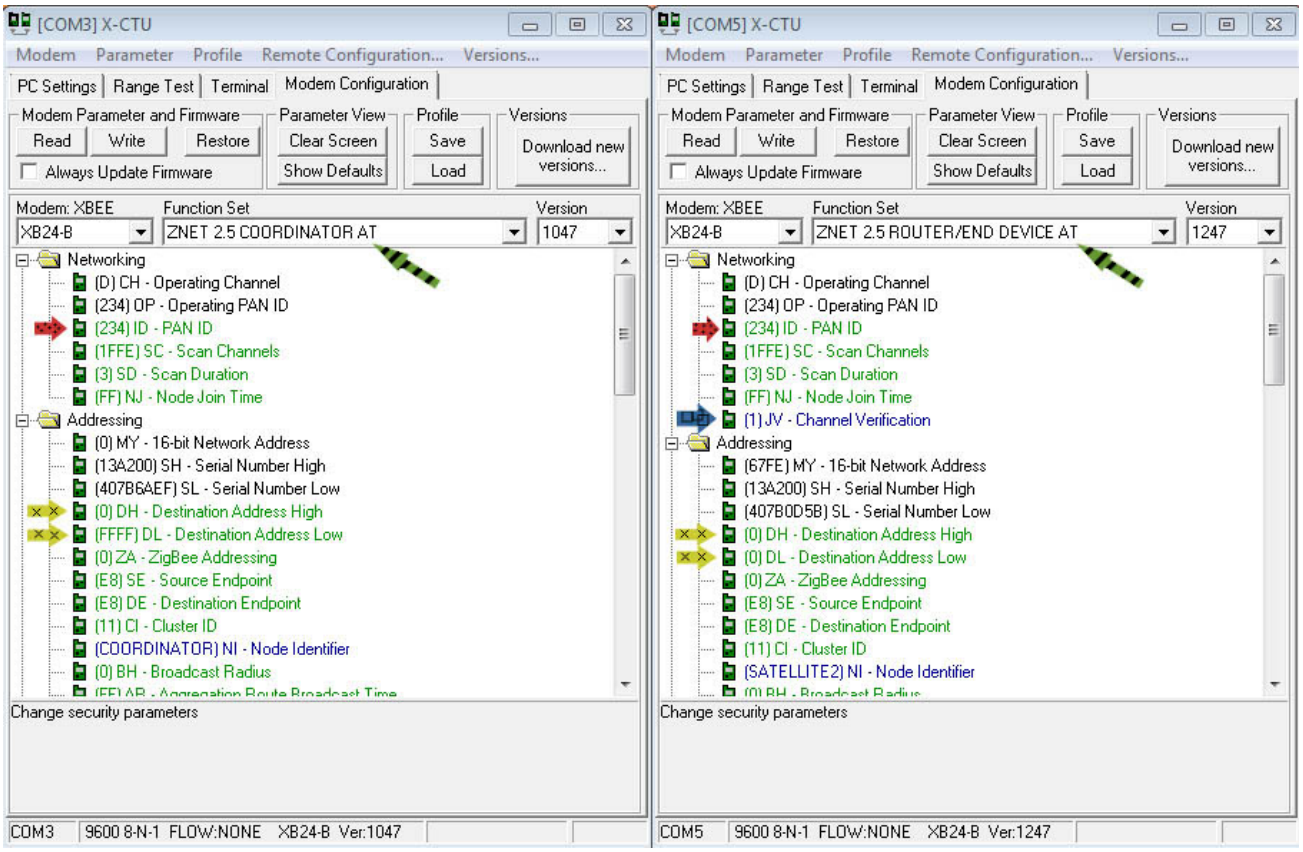


Figura 4.10: Configuración de los módulos *ZigBee*.

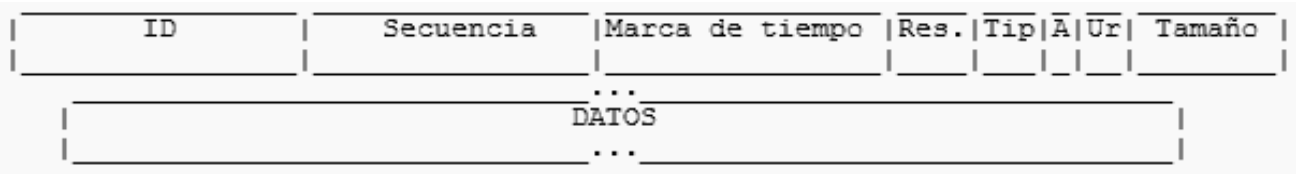


Figura 4.11: Esquema del protocolo WHAP.

entrega).

El protocolo WHAP permite hasta 255 sensores/actuadores del mismo tipo en una misma red (con margen para 255 tipos diferentes de sensores o actuadores), así como datos de hasta 6MB (hasta 96 bytes en un único paquete).

Los campos de la trama son los especificados a continuación:

ID 2 bytes para identificar unívocamente el nodo final al que va dirigido el paquete; dentro de este campo, el primer byte hará referencia al tipo de sensor/actuador y el segundo será el identificador del nodo, dentro de su tipo.

Actualmente, los diferentes tipos de nodos son:

0. Nodo central.
1. Sensor de temperatura (del sensor de temperatura y humedad ambiente).
2. Sensor de humedad ambiente (del sensor de temperatura y humedad ambiente).
3. Sensor de intensidad lumínica.
4. Sensor de humedad para plantas.
5. Sensor de gas y humo.
6. Sensor de vibración.
7. Sensor de inundación.
8. Sensor de puerta/ventana abierta.
9. Sensor PIR (movimiento por infrarrojos).
10. Cámara.
11. Actuador.

Si el paquete lo recibe un nodo satélite, el campo ID indica el nodo destino (un ID 0 significa que el mensaje es para todos). Si el paquete lo recibe el nodo central, el campo ID indica el origen del paquete.

Secuencia 2 bytes para el número de secuencia. Este campo irá incrementándose en uno por cada paquete con el mismo dato fragmentado que necesitemos enviar (para datos menores de 96 bytes, este campo es siempre 0).

Marca de tiempo 4 bytes con la fecha y hora (segundos transcurridos desde el 1 de enero de 1970) del momento en que se envió el dato (si es una secuencia de datos, cada paquete ha de llevar la misma marca de tiempo, que será la del momento en que se envió el primer paquete).

Byte con múltiples usos 1 byte "especial", con los campos especificados a continuación:

- 2 bits reservados (han de ser 0).
- 3 bits para determinar el tipo de datos contenidos:
  0. Datos desde un sensor al nodo central.
  1. Datos desde el nodo central a un actuador.
  2. Petición de datos (desde el nodo central a un sensor/actuador).
  3. Respuesta con los datos pedidos (desde un sensor/actuador al nodo central).
  4. Reseteo de alarma (indica que el usuario ha dado por finalizada la alarma).
  5. Petición de actuación (sólo para mensajes hacia actuadores).
  6. y siguientes. Reservados.
- 1 bit para indicar si se trata de una alarma desde un sensor o no.
- 2 bits indicando la prioridad del dato (0 =¿no urgente, 3 =¿muy urgente).

Tamaño del dato 1 byte indicando el tamaño total del dato en el paquete. Máximo 96 bytes.

4.4. Pruebas y limitaciones

El sistema se han probado con hasta dos nodos satélite simultáneos, pues la cantidad de elementos hardware estuvo limitada por el presupuesto. Se puede ver en la figura 4.7 el sistema funcionando con dos nodos alarma o con dos nodos sensor.

El nodo central se mantuvo encendido, como máximo, durante dos días, mientras que los nodos satélite estuvieron encendidos aproximadamente dos horas por cada sensor (algunos sensores se testearon seguidos y otros en días diferentes). A través de la interfaz mostrada en la figura 4.6, se dió de baja un nodo, volviendolo a habilitar posteriormente.

Se realizaron varios vídeos explicativos y demostrativos del sistema Surveillance 3.0 [31]. En dichos vídeos, podemos ver, por ejemplo, cómo una gota de agua activa la alarma de un nodo con sensor de inundación (figura 4.12).

Todos los sensores mencionados en este capítulo fueron probados, funcionando todos correctamente excepto el sensor de movimiento por infrarrojos (PIR), para el que no se ha conseguido extraer la información del sensor y, por lo tanto, no se ha podido incluir entre los sensores funcionales del sistema. Tanto en la documentación como en el código, se hace referencia a este sensor, aunque en el código de Surveillance 3 se han comentado las líneas necesarias para que no aparezca como sensor utilizable por el usuario, de esta forma, si se consigue extraer los datos, no resultaría complicado implementarlo en el sistema nuevamente (sólo habría que buscar las líneas correspondientes y descomentarlas).

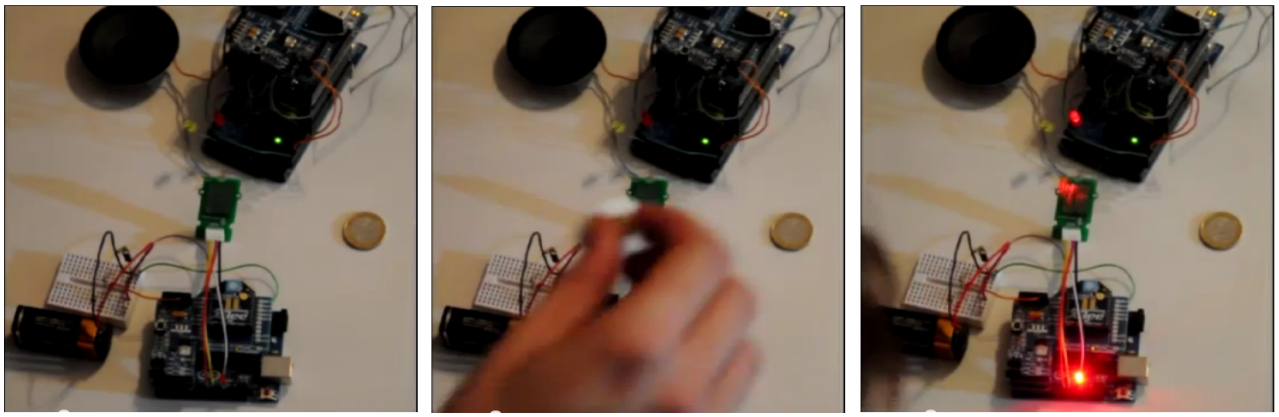


Figura 4.12: Fotogramas del vídeo demostrativo de Surveillance 3.0

Se probó a desenchufar el nodo central una vez estaba encendido y se comprobó que se cargan las últimas opciones de configuración introducidas por el usuario y no las programadas inicialmente; además, se vio que los datos anteriores al reinicio se perdían (no se pueden visualizar los datos anteriores al apagado, al guardarse en memoria RAM), pero los nuevos datos eran tratados correctamente.

Hay que destacar que toda la configuración está grabada en el código o se guardan en la memoria EEPROM del nodo central (las opciones configurables por el usuario, como IP, puerto, etc.) y se cargan al iniciar, por lo que si cualquier nodo se reinicia por cualquier motivo (ya sea un nodo satélite o el central), este se configurará correctamente de forma automática y seguirá funcionando igual que antes del reinicio, sin necesidad de ser reconfigurado por el usuario.

Surveillance 3, al ser la primera versión del sistema distribuido, tiene varias limitaciones:

1. No dispone de posibilidad de programar la habilitación o deshabilitación de las alarmas para que sólo estén activas durante un horario determinado.

2. No permite la posibilidad de que un sensor analógico dispare una alarma al alcanzar o bajar de un umbral.

3. Debido a la capacidad de cómputo de los nodos arduino, no es posible incluir sensores complejos, como cámaras.

4. El código final de Surveillance 3.0 está limitado por la memoria RAM del procesador de esta placa, pues el código final casi ocupa por completo esta memoria, a pesar de utilizar técnicas para minimizar este uso. Esto limita el número de sensores activos simultáneamente e impide incluir más código de programa.

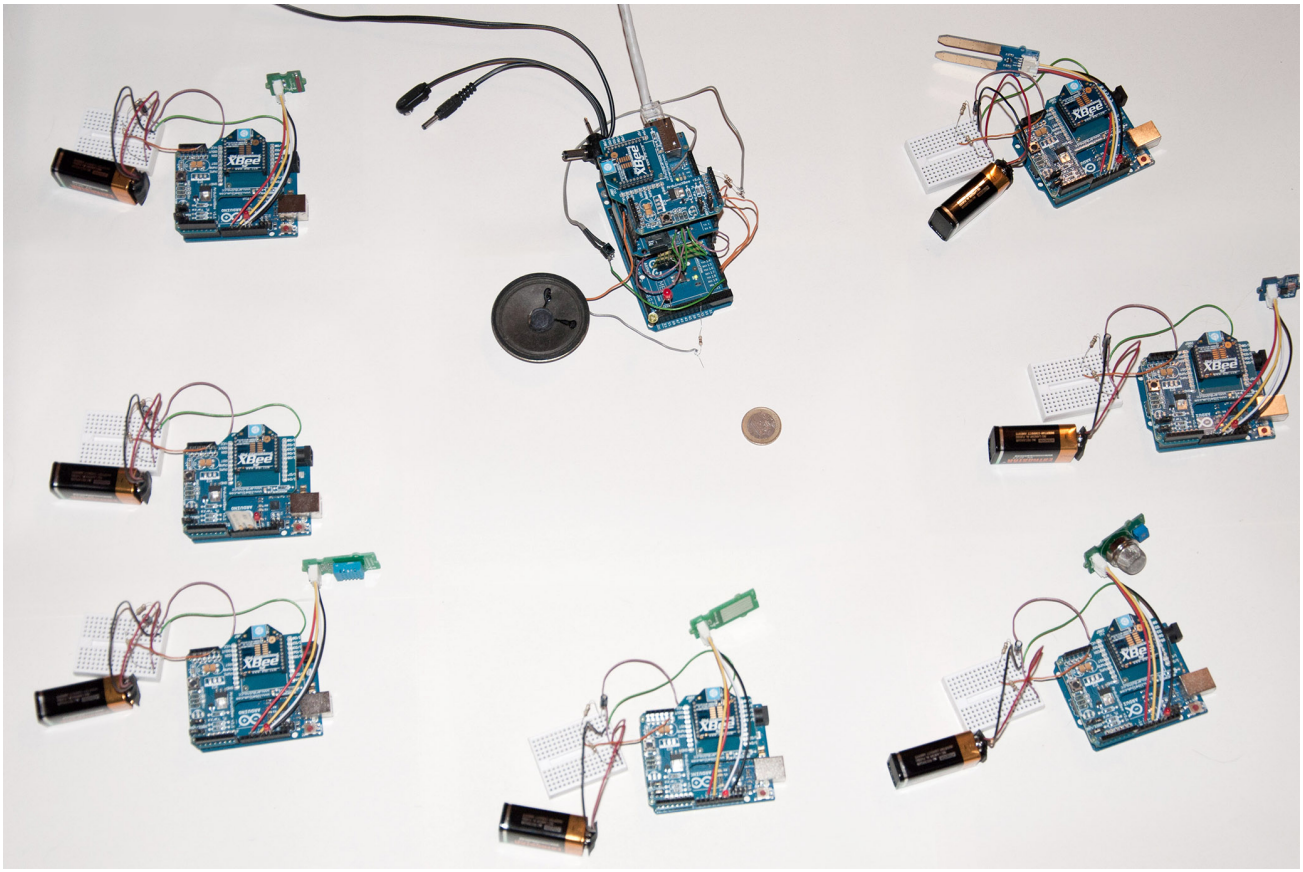


Figura 4.13: Nodo central y sensores disponibles en Surveillance 3.0

Capítulo 5

Surveillance 4.0

Surveillance 4 es la evolución natural de la versión 3.0, propiciada por la salida al mercado de ordenadores de placa única (*SBC*) de bajo coste. La anterior versión cumplía con los requisitos propuestos al inicio de este PFC, pero lo hacía de manera pobre, con un bajo nivel de detalle y a través de una interfaz poco atractiva a la vista. Es por ello que, con el ánimo de mejorar y ofrecer un sistema de calidad, se comenzó a trabajar en la versión 4. En esta versión se añade una interfaz de usuario más elegante, gracias a la sustitución del nodo central por una *Raspberry Pi* y el uso del servidor web *Django*. También se añaden nodos satélites que permiten el *streaming* de vídeo y nodos actuadores.

Habiendo llegado con Surveillance 3.0 prácticamente al límite del *Arduino Mega 2560* (placa Arduino más potente en la fecha de inicio de esta versión), se hizo necesario su reemplazo por una *SBC* más potente.

5.1. Diseño

Si bien el esquema parecido al de la versión anterior, hay cambios significativos. En el plano hardware, los cambios principales son el uso de Wi-Fi para interconectar ciertos satélites que necesitan usar un gran flujo de datos o el uso de la *Raspberry Pi* como nodo central.

Debido a la necesidad de gran capacidad de cómputo por parte del nodo satélite para poder analizar las imágenes, se ha decidido utilizar placas *Raspberry Pi* con conexión a la red eléctrica (sin baterías) en los nodos satélite que utilicen estas tecnologías complejas como sensores. De esta forma todo el procesamiento de la imagen se puede realizar en la placa y se libea al nodo central de ese trabajo.

En cuanto al software, si bien en los nodos satélites con sensores básicos no ha habido modificación de la arquitectura, el nodo central cambia por completo su esquema. Aunque conceptualmente es muy parecido, entre otras mejoras, se utiliza software de terceros que nos permitirá realizar más tareas con menor esfuerzo. Además, se utiliza *Python* como lenguaje de programación orientado a objetos para poder desarrollar aplicaciones más complejas en menos tiempo.

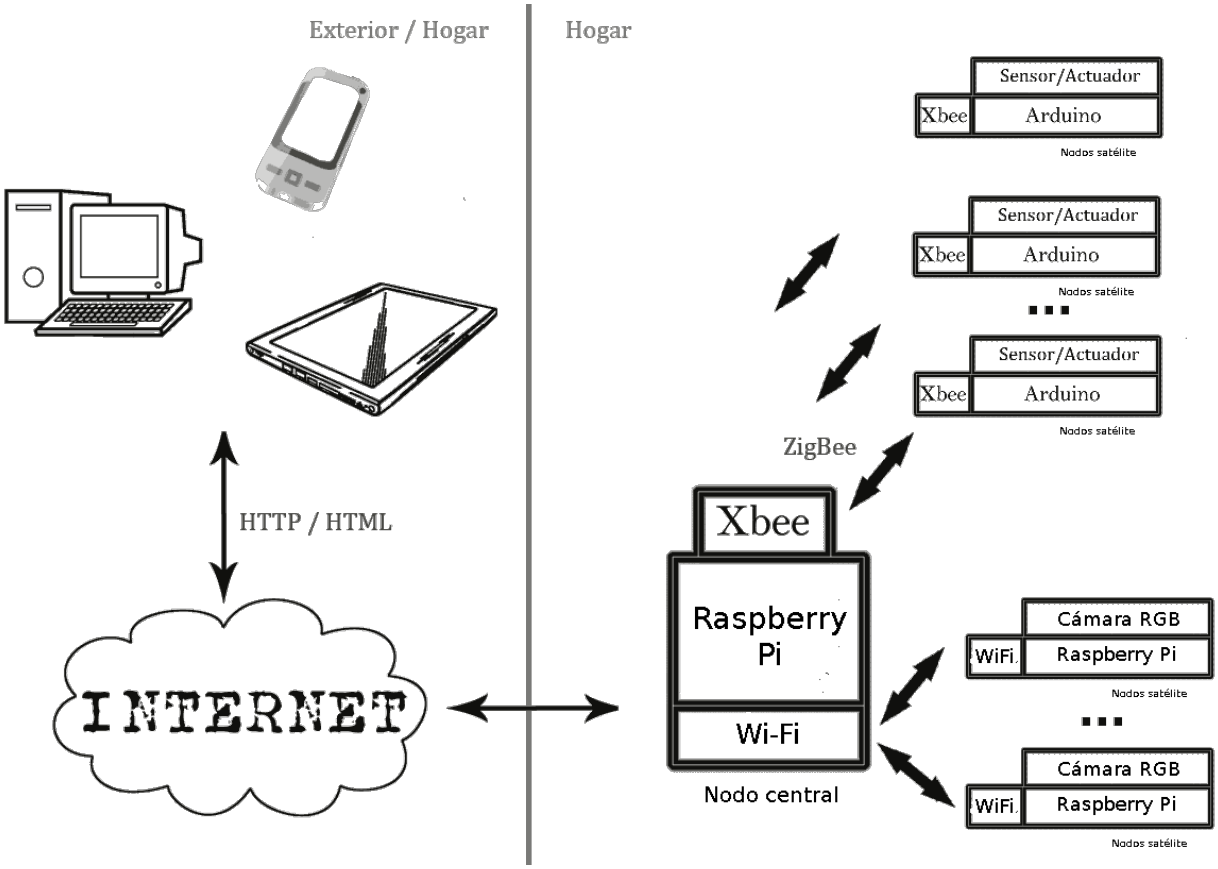


Figura 5.1: Esquema sobre la arquitectura de Surveillance 4.0.

5.2. Nodo central

Surveillance 4 se comenzó a desarrollar unos meses después de la salida al mercado de la *Raspberry Pi* y, por ende, de las *SBCs* de bajo coste. Cuando se tuvo que decidir que placa utilizar como nodo central, *Raspberry Pi* ya había obtenido una gran acogida, además, mientras que otras placas nos ofrecían mayor capacidad de cómputo u otras mejoras, siempre tenían un coste monetario algo superior. Finalmente se decidió usar la *Raspberry Pi* por ofrecer capacidad suficiente para el proyecto, además de contar con mayor documentación (por su mejor acogida entre los desarrolladores) y, sobre todo, por ofrecer una plataforma con un sistema operativo *Unix*.

El hardware utilizado en el nodo central cambia radicalmente, pues utilizaremos una placa *Raspberry Pi "B"*[23] como núcleo principal del nodo. Este dispositivo es de *hardware* abierto[1], lo que coincide perfectamente con la política de código abierto del departamento y principalmente, nos aporta con respecto a Arduino mayor velocidad de procesamiento, más capacidad de almacenamiento y un sistema operativo *Unix* (basado en *Debian*), lo cual nos permite utilizar software ya creado para la plataforma *Jderobot*[5] y resolviendo de paso parte de los problemas que teníamos en la versión anterior con Arduino4.4 al haber llegado prácticamente al límite de uso de la memoria RAM. Por contra, estas placas tienen un consumo muy superior al Arduino, lo que obliga a utilizarla mediante una conexión permanente a la red eléctrica y un transformador.

Adicionalmente se incorpora un "pincho" Wi-Fi por USB para conectar tanto los satélites que requieran de un gran flujo de datos (cámaras) cómo el router con acceso a Internet (si se quiere tener acceso al sistema desde el exterior de la LAN). Si el usuario quiere conectarse con el router mediante cable Ethernet, el funcionamiento no se verá alterado, pero es necesario tener el "pincho" y un router Wi-Fi si queremos poder acceder a los satélites que utilicen esta tecnología (el nodo central no puede hacer de router Wi-Fi).

Se añade un módulo *XBee* (que se conecta directamente con la *Raspberry*, ahorrando el coste de la *shield* respecto a Surveillance 3) para poder conectar con los satélites que se comuniquen mediante ZigBee (todos excepto las cámaras).

5.2.1. Servidor web

En esta ocasión, el servidor web se implementa a través del entorno de desarrollo web *Django*[25] (concretamente, se ha implementado la versión 1.6), lo que nos va a permitir desarrollar nuestra aplicación más rápidamente, además de poder utilizar funciones creadas y probadas por terceros.

Al tratarse de una interfaz HTTP y HTML, ésta puede ser accedida desde cualquier dispositivo con navegador web. Esto se puede encontrar hoy en día en dispositivos móviles, ordenadores, tabletas o incluso en algunos televisores, lo que proporciona una gran flexibilidad de acceso al usuario.

Si el router al que estamos conectados lo permite, podemos abrir los puertos que utiliza el sistema (con la configuración actual, el puerto 80 del nodo central y el 8080 del nodo de la cámara) para que el sistema sea accesibles desde el exterior. También utilizar el router para

[1]El esquema de la *Raspberry* ha sido liberado recientemente y es accesible al público desde el enlace [24].

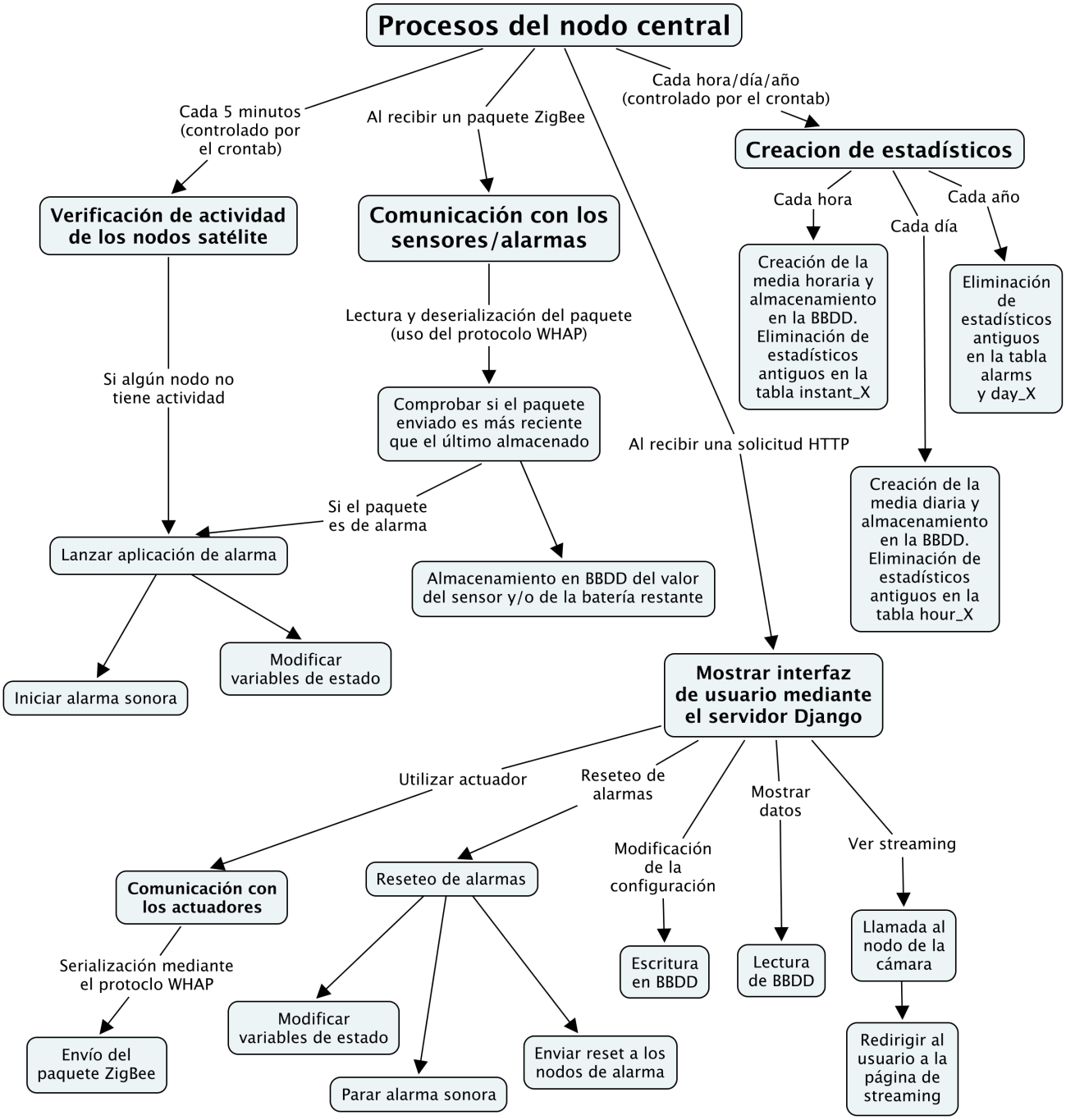


Figura 5.2: Esquema lógico del software programado en Surveillance 4.

permitir o denegar el acceso a a alguna o todas las cámaras desde el exterior, aunque esto también es posible realizarlo a nivel de configuración del sistema, como veremos más adelante.

Diseño web

Las páginas de Surveillance 4 utiliza las API's en *JavaScript* de *Bootstrap*[36] y *JQuery*[35] para obtener un resultado gráfico más llamativo.

La página web ha tenido que ser modificada casi por completo para adaptarse a las diferentes secciones de la interfaz de usuario, así cómo a las características del uso de plantillas para poder crear un sistema modular. Las plantillas se rellenan dinámicamente con los datos del nodo central a través del sistema de vistas y plantillas del servidor *Django*.

Se ha hecho énfasis en la simplicidad y facilidad de manejo por parte del usuario sin conocimientos técnicos, con textos e imágenes auto explicativas, de tamaño adecuado que mejoran la visualización sin llegar a entorpecer la navegación.

Se han implementado hojas de estilo adicionales, necesarias para que en caso de haber una alerta activa, el servidor *Django* indique estos nuevos ficheros en lugar de los CSS iniciales y que la interfaz de usuario adquiera una tonalidad rojiza, indicando visualmente de forma rápida el estado del sistema.

Secciones de la interfaz

Como se ha comentado, la interacción con el usuario se realiza mediante el servidor web *Django*, en el cual se han implementado las siguientes páginas:

LogIn/LogOut Imagen 5.3. Al igual que en la versión anterior, únicamente se permite el acceso sin autenticación previa a la página de login, en la cual aparece un formulario que nos permite introducir las credenciales (usuario y contraseña). La gestión de la sesión de usuarios se realiza mediante el uso de los módulos "django.contrib.auth" y "django.contrib.sessions".

Inicio Esta es la primera página que ve el usuario al entrar al sistema. Se presenta la página de Surveillance únicamente con los elementos comunes (menú de acceso al resto de las secciones, estado del sistema -con o sin alarma activa-, botón de logout y pié de página).

Admin Imagen 5.4. Para entrar en esta sección es condición necesaria ser administrador del sistema (al crear un nuevo usuario, se puede elegir si otorgarle ese rol o no). Si no se está autenticado como administrador, se mostrará otra página de login (distinta de la página de login inicial, pues al estar autenticado, esta página de login contiene el menú para navegar por el resto del sistema) para poder reautenticarse como administrador.

Esta página provee mediante el panel de administración de *Django* una herramienta para acceder a la base de datos. Mediante la manipulación de base de datos se puede dar de alta o de baja nuevos nodos, así como nuevos usuarios. Se ha creado la sección "Configuración" para poder realizar estas tareas de forma más intuitiva, pero se deja visible este panel de administración para su uso por usuarios avanzados.

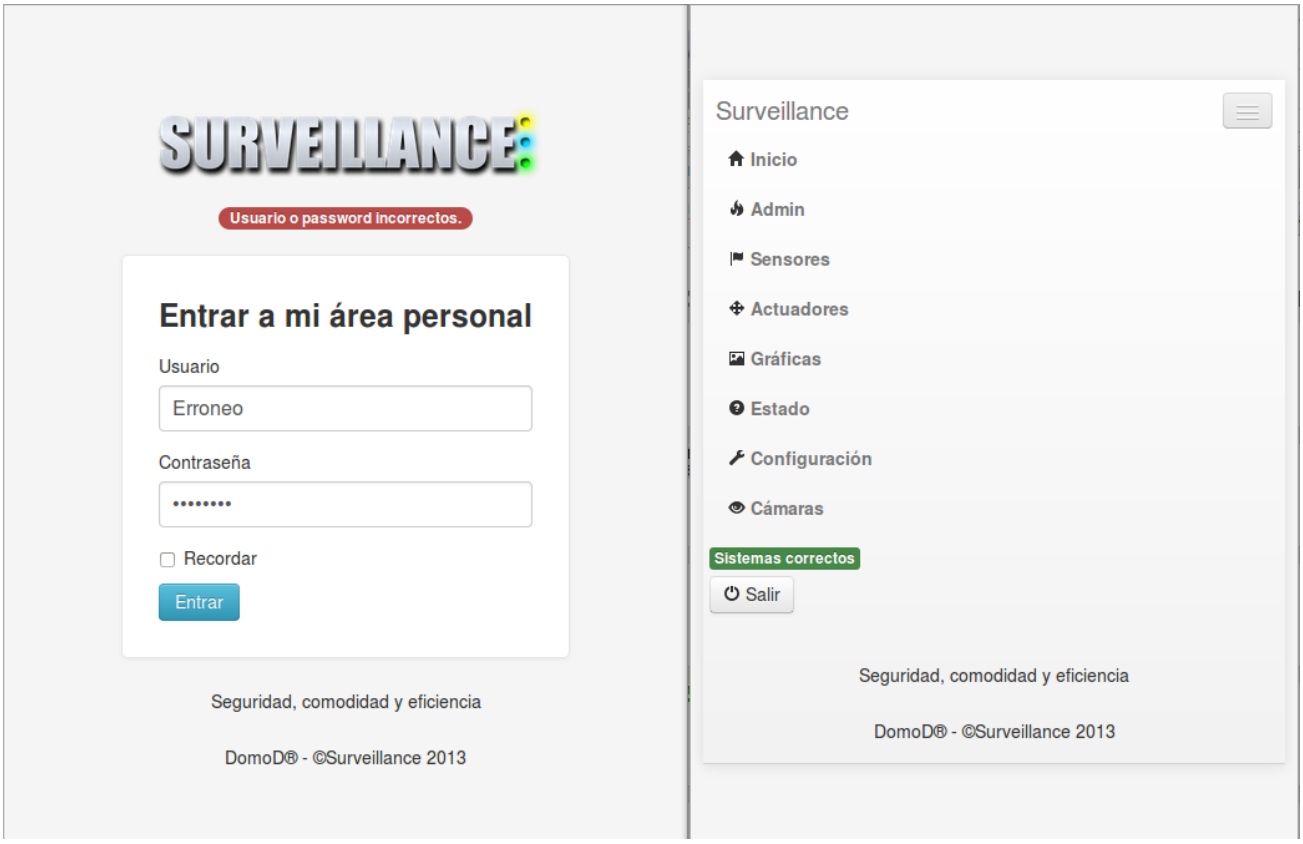


Figura 5.3: Interfaz de Surveillance 4. A la izquierda, módulo de login. A la derecha, menú de la aplicación (visualización con una pantalla estrecha)

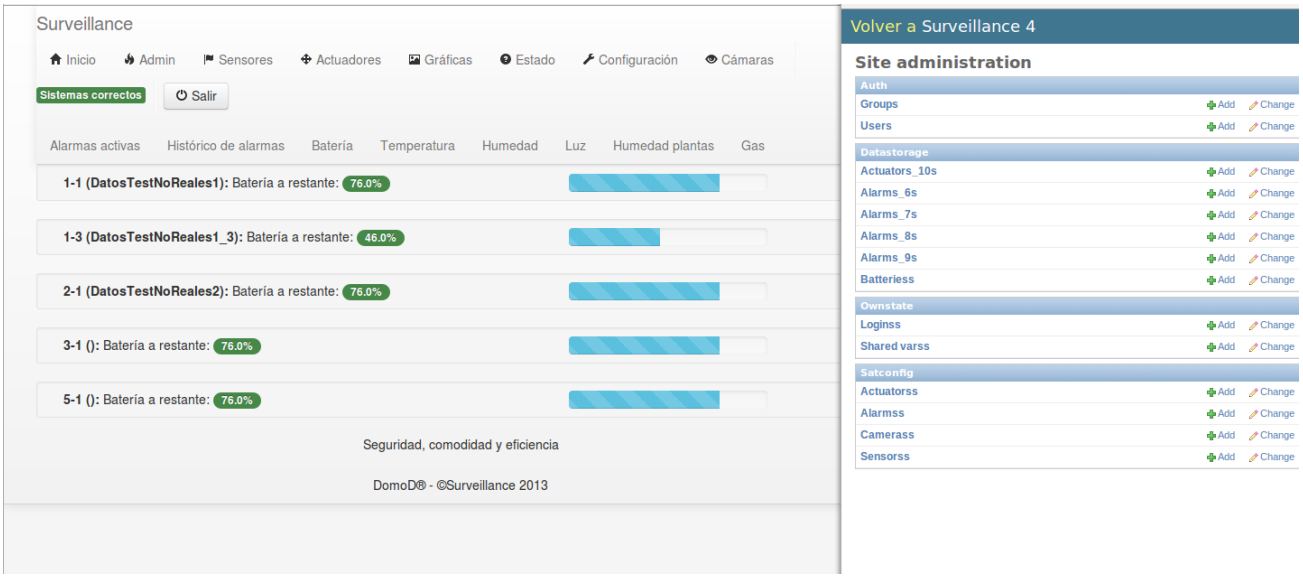


Figura 5.4: A la izquierda, información de la batería restante. A la derecha, panel de administración de base de datos.

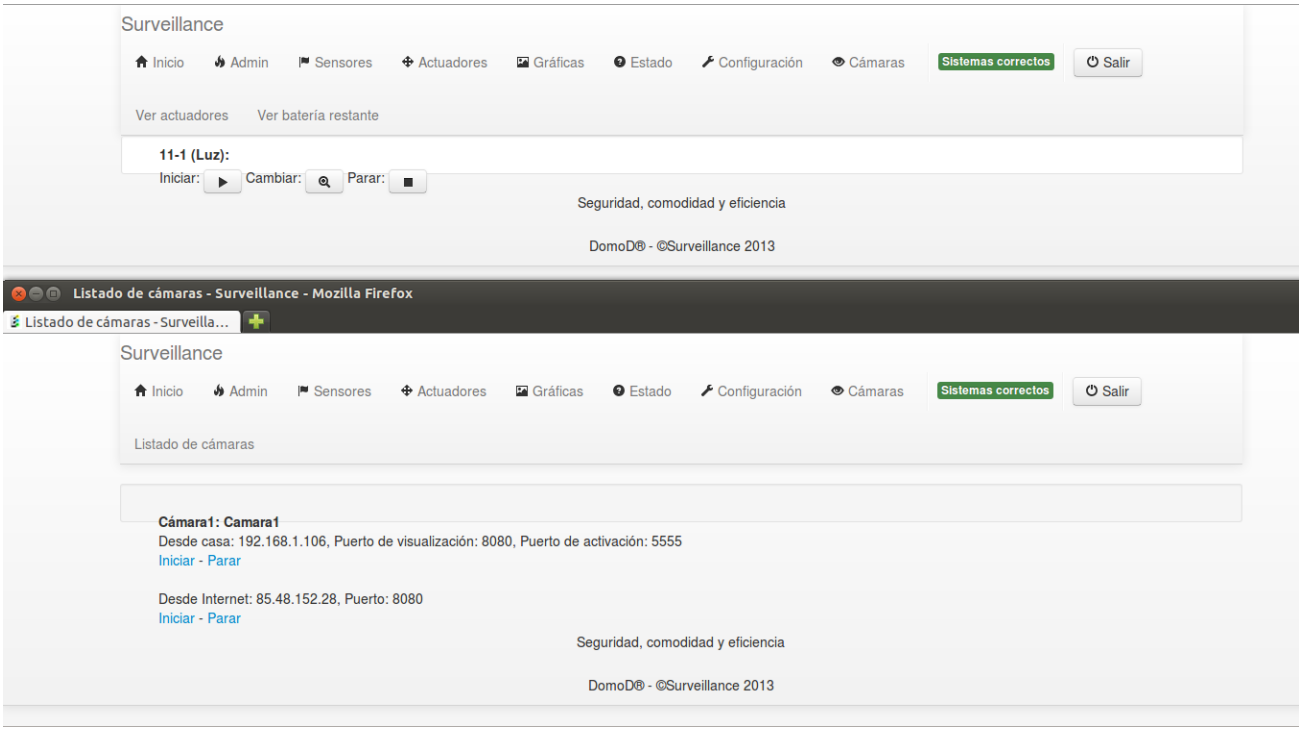


Figura 5.5: Interfaz de Surveillance 4. Arriba, control de actuadores. Abajo, control de cámaras.

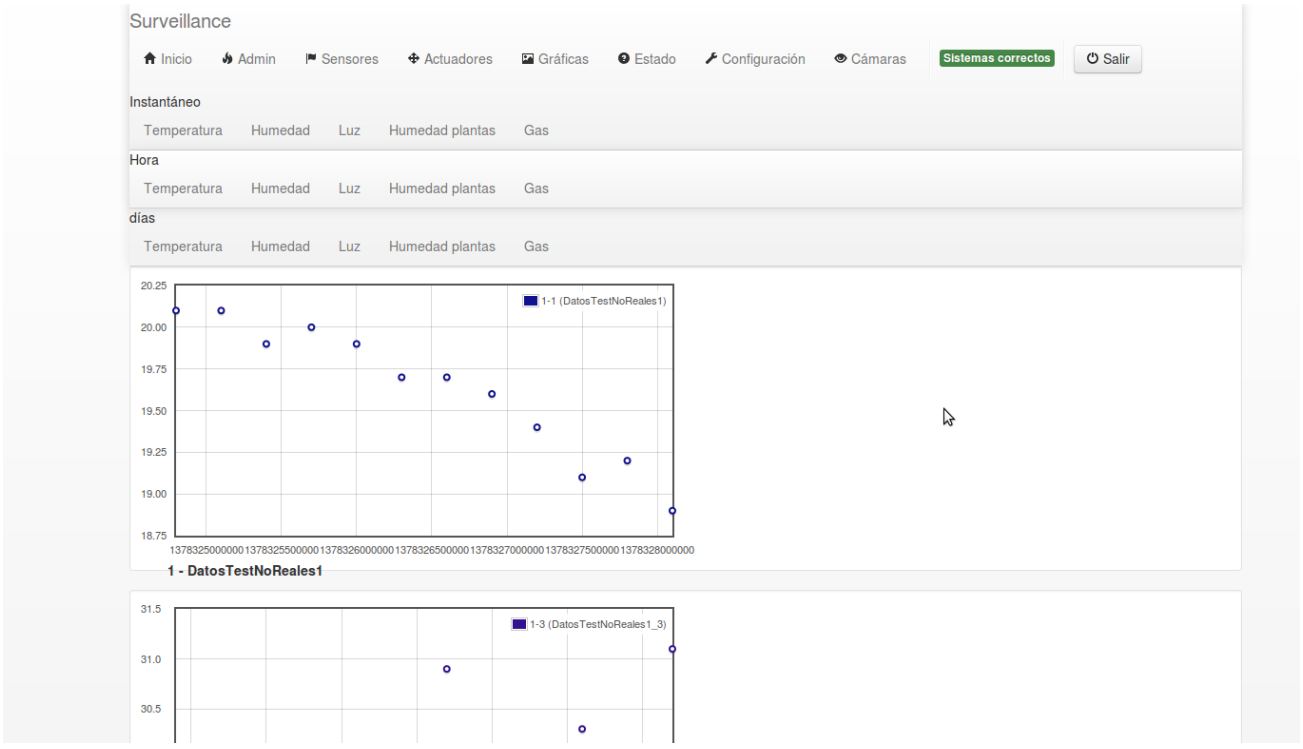


Figura 5.6: Interfaz de Surveillance 4. Visualización de gráficas de sensores.

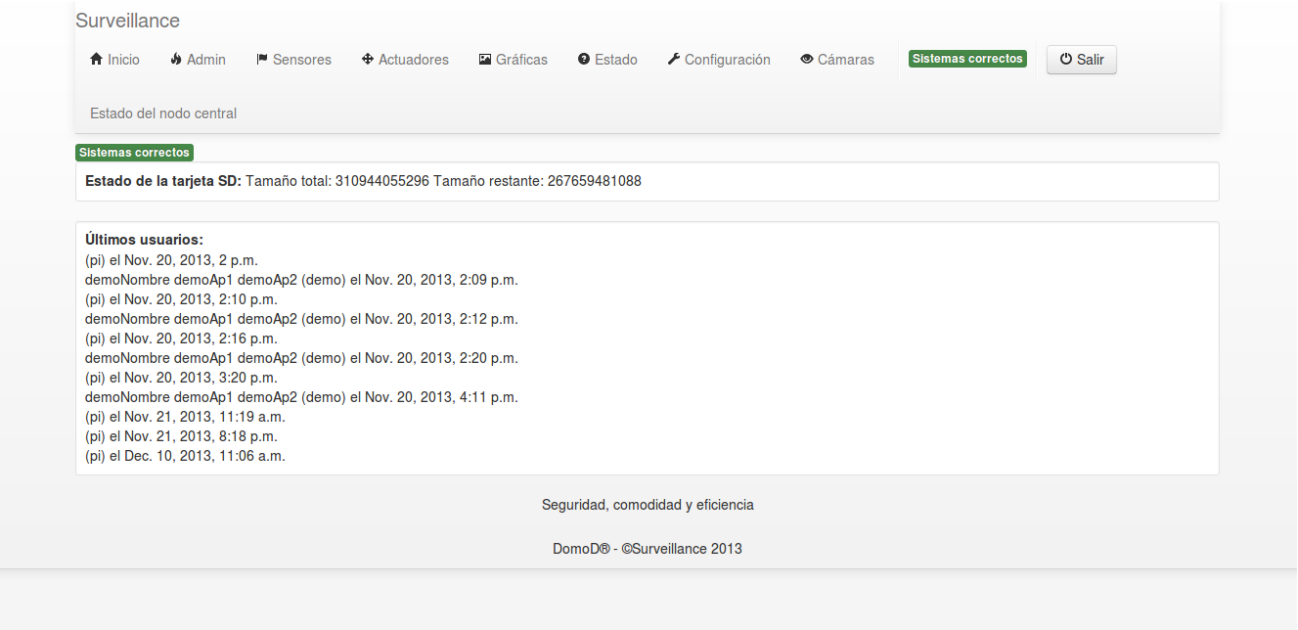


Figura 5.7: Interfaz de Surveillance 4. Página de estado del sistema.

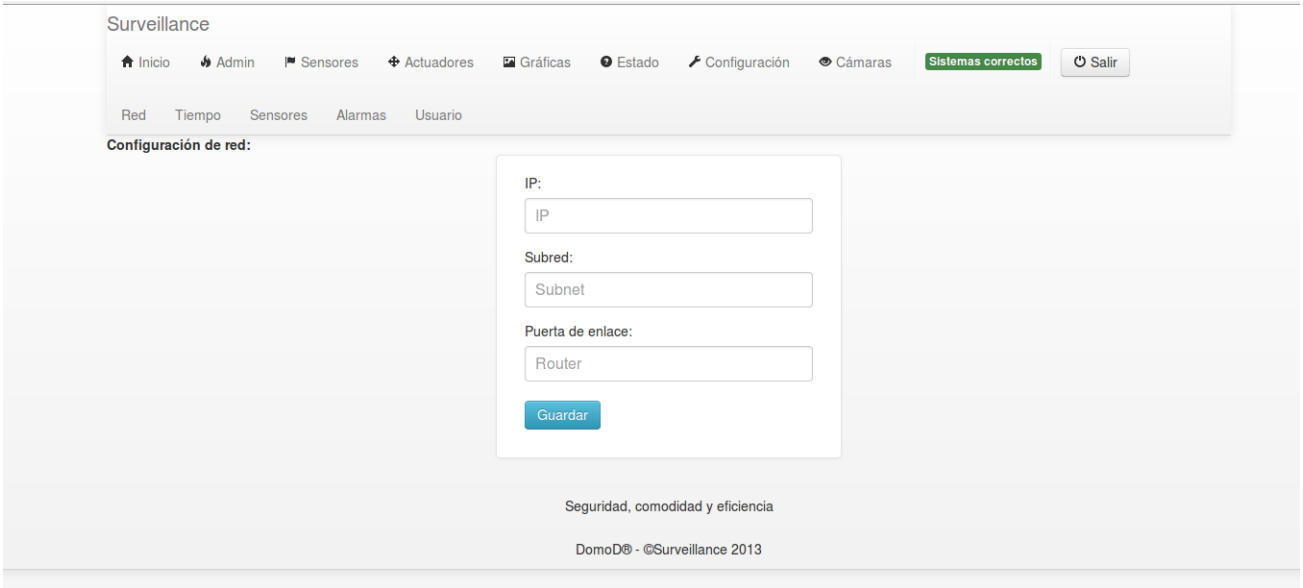


Figura 5.8: Interfaz de Surveillance 4. Página de configuración, sección de configuración de red

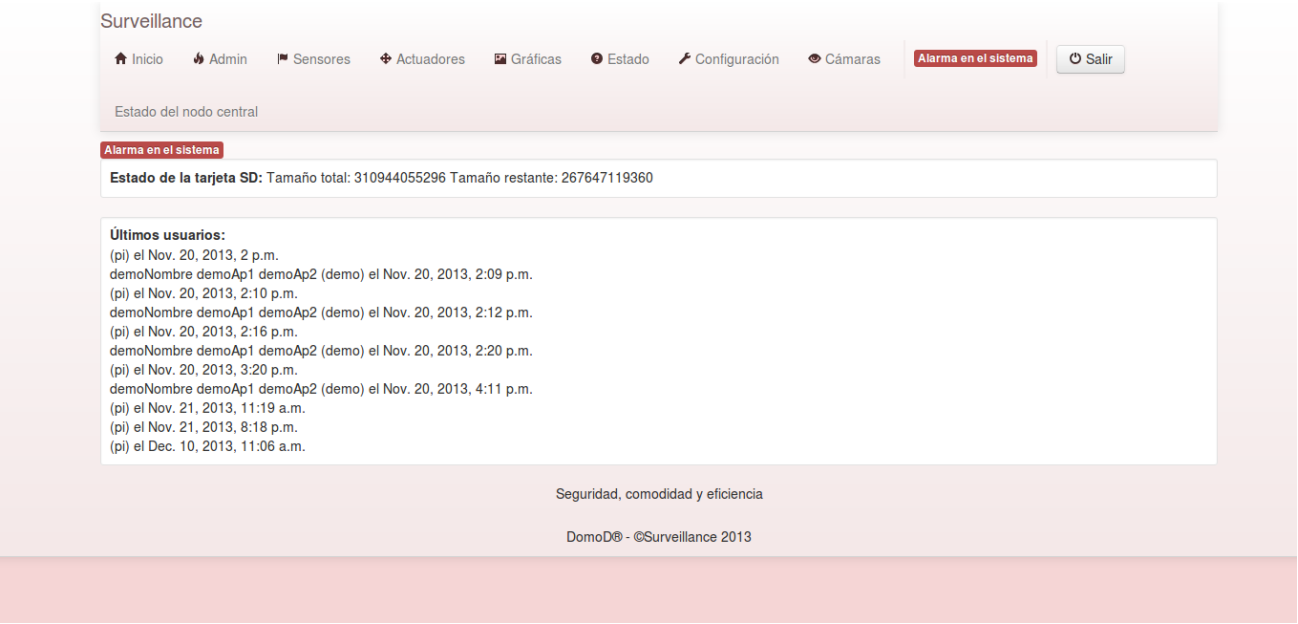


Figura 5.9: Interfaz cuando hay una alarma activa.

Sensores Esta sección contiene varias subsecciones relacionadas con los sensores y las alarmas:

- **Alarmas activas:** Aquí se muestran las alarmas que aún no han sido resueltas (si hubiera), la fecha en la que ha ocurrido la alarma y el botón de reinicio de alarmas, para poder volver a restaurar el sistema (quitar el estado de alarma) una vez arreglado el problema que originó la alarma.
- **Histórico de alarmas:** En esta subsección podremos ver las alarmas que han sucedido en el sistema (se muestran un máximo de 99 entradas, por orden cronológico inverso) y ya han sido resueltas. Se incluye la información del nodo que provocó la alarma, así como la fecha en la que sucedió.
- **Batería:** Imagen 5.4. Muestra el último valor recibido de las baterías de todos los nodos satélites con ZigBee (sensores, alarmas y actuadores), no así el de las cámaras, pues están están conectadas a la red eléctrica.

Sensores: Cada tipo de nodo satélite sensor (temperatura, humedad, luz, humedad de plantas y gas) hay una subsección, en ella se muestra el último valor recibido del satélite y la fecha en la que se obtuvo dicho valor.

Actuadores Imagen 5.5. En esta página se pueden visualizar los nodos actuadores configurados, así como tres botones para cada uno que nos permiten darle la orden de "encendido", "apagado" o "cambiar estado".

Gráficas Imagen 5.6. Esta sección contiene las gráficas generadas a partir de los datos históricos de los sensores. Está organizada en subsecciones, pues podemos observar las gráficas de los últimos valores enviados en una hora (sección *Instantáneo*), la media creada cada hora

(últimas 24 horas, sección *Hora*) o la media creada cada día (últimos 265 días, sección *Días*) de cada sensor, siempre que se disponga de datos (al menos un valor) para crear estas gráficas. Los datos usados en las gráficas "instantáneas" son los provenientes directamente del sensor, mientras que el script "meanmaker.py"5.2.3 se encarga de la creación de los datos para el resto de las gráficas.

Estado Imagen 5.7. Aquí se muestra la información relativa al nodo central y, en general, al sistema. Se muestra el espacio restante de la SD, así como el usuario y la hora de los últimos 10 accesos al sistema.

Configuración Imagen 5.8. En esta sección se pueden modificar las variables del sistema de forma sencilla. La principal diferencia con el panel de administración de *Django* (sección *Admin*5.2.1), consiste en que en esta página se permite modificar diferentes variables. Por ejemplo, se puede modificar en caliente los datos de red del nodo central (IP, gateway), cambiar la hora actual, etc. Ofrece también parte de la administración que se puede realizar desde la sección *Admin*, pero de de forma más sencilla [2].

Cámaras Imagen 5.5. Esta página permite la visualización del streaming de los nodos satélite con cámara, para ello utilizamos el servidor web del nodo central y otro en el nodo satélite, como se describe a continuación:

- **Control:** Para cada cámara configurada se muestra en la interfaz de usuario del nodo central cuatro enlaces, uno para iniciar y visualizar el streaming del vídeo accediendo desde la red LAN donde está ubicado el nodo satélite, otro para detener el streaming de vídeo y sus homólogos para accesos desde Internet (desde el exterior de la red LAN). Esta duplicidad es necesaria, puesto que la dirección IP/puerto en la que estará escuchando el servidor del nodo satélite cambia; si se quiere conectar desde la LAN, habrá de utilizarse la IP física del nodo satélite, pero si se accede desde el exterior, hay que utilizar la IP pública del router y el puerto que se haya configurado en el mismo.
- ***Streaming*:** En esta página, servida por el nodo satélite, se visualiza el streaming de la cámara. Se puede observar que desaparece el menú superior, quedando únicamente la opción de "volver atrás", esto es así debido a que tanto el streaming como la página web visualizada es servida por el nodo satélite, por lo que no se puede obtener acceso a los datos o variables del nodo central para componer el menú, por lo que únicamente se ofrece la opción de volver atrás (acción ejecutada mediante JavaScript).

5.2.2. Recepción de datos

Los valores enviados por los nodos satélites han de ser recepcionados por el nodo central para poder operar con ellos. Aquí explicaremos cómo se reciben estos datos y su tratamiento primario.

[2] Al término de este PFC, no se ha realizado la lógica para realizar las acciones descritas en la sección "Configuración", únicamente se han creado las páginas web subyacentes.

ZigBee

El nodo central contiene también software realizado en este PFC para leer los datos enviados desde el nodo satélite y almacenarlos en la base de datos de Django. Se ha creado un programa en Python que recoge la información "en bruto" desde el XBee (módulo ZigBee) a través del puerto serie de la GPIO de la Raspberry Pi, para ello se vale del protocolo propio WHAP (descrito en la sección 4.3.3).

El flujo de recepción de los datos por ZigBee es similar al usado en Surveillance 3, el sensor envía los datos utilizando el protocolo propio WHAP mediante el módulo ZigBee, el nodo central recibe esa información y la deserializa con el mismo protocolo. La diferencia es que los datos, después de deserializarse, se guardan en una base de datos. Si se trata de un paquete de alarma o si el valor recibido excede del límite configurado, se activar el protocolo de alarma (se puede ver el sistema en modo alarma en la imagen 5.9). Esta base de datos nos permitirá mostrar estadísticos de los valores al usuario.

Hay ciertas particularidades que afectan a este script: dado que utiliza el puerto serie, hay que deshabilitar los componentes del sistema que utilizan este mismo puerto para que no haya conflictos. Esto es posible dado que los componentes conflictivos no son utilizados en este PFC. Además, dado que es necesario acceder a la base de datos de Django para guardar el valor recibido, hay que indicarle al scripts los parámetros del servidor de Django que vamos a utilizar. Por otro lado, para que este programa se lance con el arranque de la placa, se ha incluido su ejecución en el fichero "/etc/rc.local".

5.2.3. Tratamiento de los datos, estadísticas

Una vez introducidos los datos recogidos del dispositivo ZigBee en la base de datos, se necesitan ciertos procesos estadísticos y de mantenimiento para minimizar el tamaño de la base de datos. Por disponer de un tamaño de almacenamiento finito, es imposible guardar todos los datos que envían los sensores sin modificarlos. Por ello se ha creado el programa "meanmaker.py". Este proceso se tiene que realizar periódicamente, por lo se ejecutará mediante el *crontab* del sistema cada hora.

El programa crea la media de los datos que van a quedar obsoletos para poder eliminarlos y guardar únicamente la media, utilizando así mucho menor espacio de almacenamiento, aunque perdamos granularidad.

- Cada hora, el programa recopila todos los datos enviados por el satélite en la ultima hora (de la tabla instant_X), realiza su media aritmética y guarda el resultado en la tabla de "medias horarias" (hour_X). Posteriormente se eliminan los valores de la tabla instant_X que tengan una antigüedad mayor de una hora.

- Cada día, el programa recopila todos las medias horarias de las ultimas 24 horas (de la tabla hour_X), realiza su media aritmética y guarda el resultado en la tabla de "medias diarias" (day_X). Posteriormente se eliminan los valores de la tabla hour_X que tengan una antigüedad mayor de un día.

- Cada mes, el programa elimina de la tabla day_X aquellas entradas que superen el año natural de antigüedad (365,25 días).

En resumen, se mantienen tres tablas (valores instantáneos, media de cada hora y media de cada día) para poder mantener los datos durante todo un año.

A través del *crontab* del sistema, cada 5 minutos se ejecuta el programa "activity_verifier.py", el cual verifica que los nodos configurados tienen actividad reciente. Se comprueba que los satélites tienen como dato más reciente uno con atiguedad inferior a media hora. Si esto no fuese así, se lanza una alarma en el sistema.

5.3. Nodo satélite

En Surveillance 4.0, además de los sensores utilizados en la versión anterior (humedad, temperatura, luz, vibración, etc.), añadimos actuadores (altavoces, controladores de luz), así como sensores más complejos (cámaras RGB) gracias a la mayor potencia de procesamiento que podemos alcanzar con el uso de *Raspberry Pi* en los nuevos nodos satélite.

Las dos versiones de Surveillance tratadas en este PFC se comunican por ZigBee con sus satélites, mediante el uso del protocolo propio WHAP4.3.3. Utilizaremos este método para flujo de datos pequeños (alarmas, sensores simples y actuadores), pero no es posible utilizarlo para grandes flujos de datos, como el streaming de vídeo. En esta versión también se han incluido los actuadores accionados por relé.

5.3.1. Sensores y alarmas

Los nodos sensores utilizan la misma arquitectura que en Surveillance 3 (ver figuras 4.8 y 4.9), de hecho, los sensores y alarmas son compatibles con los ya utilizados en la versión anterior (ver 4) sin realizar modificación alguna. Todo escrito en la sección *Comunicación con los nodos satélite*4.3.3 de Surveilance 3 es válido para los nodos satélites de alarma y sensores contenidos en Surveillance 4.0.

Cada 5 minutos el nodo envía la información de la batería restante y, en caso de ser un sensor, el valor actual del sensor. Esta operación se realiza mediante la salida serie del Arduino y el módulo ZigBee, para serializar los datos se utiliza el protocolo propio WHAP. En caso de producirse una alarma en los nodos de este tipo, la alarma es enviada inmediatamente hacia el nodo central.

5.3.2. Actuadores

En esta versión se ha añadido la posibilidad de utilizar actuadores. La arquitectura software y hardware es la misma que para los nodos sensores/alarma, pero en lugar de tener conectado el sensor, tiene un relé que permite o deniega el paso de electricidad hacia el elemento actuador. Se ha probado con una lámpara de 40W, pero se podría aplicar a cualquier elemento que funcione mediante conexión a la red eléctrica, por ejemplo, motores para persianas.

Estos nodos tienen una estructura muy similar a los sensores, utilizan los mismos elementos hardware (contienen un relé en lugar del sensor), siendo por tanto inalámbricos, se comunican con el nodo central mediante ZigBee y pueden recibir del nodo central las órdenes de on y off (relé abierto o cerrado).

El software programado es prácticamente idéntico al usado en los nodos sensores, con la salvedad de que si le llega un paquete ZigBee dirigido a ese nodo, ejecutará la orden emitida. Dicha orden se obtiene del primer elemento del campo *datos* del protocolo WHAP4.3.3, cuyos valores actualmente pueden ser: 1 (permitir corriente), 2 (denegar corriente) o 3 (cambiar estado). Esta orden se adecúa a la plataforma (se verifican qué pines de salida del Arduino deben establecerse a 5 voltios y cuales a tierra para ejecutar la orden deseada en el relé) y se transmite a un relé. El relé usado actúa a través de voltajes de 5V y permite el paso de hasta 130 voltios, lo que lo hace muy adecuado para nuestro nodo actuador.

Al igual que los nodos alarmas, los actuadores enviarán la información de la batería restante cada 5 minutos, para poder monitorizar su estado.

5.3.3. Vídeo

Surveillance 4.0 permite la visualización de vídeo en directo (*streaming*) a través de nodos satélite *Raspberry* con cámaras web estándar (ver imagen 5.12). Para ello nos valemos de una conexión Wi-Fi, principalmente debido a que una conexión mediante ZigBee no soporta un flujo de datos tan pesado. Debido a que el consumo eléctrico del conjunto del dispositivo (cámara, Wi-Fi y placa *Raspberry*) es demasiado grande como para ser alimentado por una batería, el nodo ha de estar conectado a la red eléctrica.

Para enviar el *streaming* de imágenes, utilizamos el programa de código abierto *M-JPEG Streamer*[20], el cual nos permite capturar imágenes en formato JPEG de una cámara conectada al USB y exponerlas como flujo de imágenes en una página web, sin necesidad de instalar ningún componente adicional en el navegador donde se visualice. Se han modificado las páginas que proporciona el proyecto *M-JPEG Streamer* para que se asemejen al resto de páginas de Surveillance 4, tal como se explica en 5.2.1.

El nodo satélite ejecuta desde su arranque el programa "listener.py". Este programa permite la comunicación con el nodo central para poder iniciar el flujo de vídeo únicamente cuando vaya a ser visualizado y detenerlo cuando finalice la visualización. De esta forma la CPU no está pidiendo imágenes de forma constante a la cámara, reposando cuando el usuario no requiere el flujo de vídeo.

El programa abre de forma continua un socket TCP, cuya dirección sólo se tiene acceso desde la LAN, donde se ubica el nodo central. Cuando le llega una petición en este socket, el nodo recoge el parámetro enviado ("init" para iniciar el flujo de vídeo, "stop" para detenerlo o "swit" para cambiar el estado). El nodo verifica su estado actual (transmitiendo o parado) y efectúa las acciones según el parámetro recogido (inicia o para el flujo de vídeo). Para iniciar el flujo, se creó otro socket diferente, donde escucha peticiones del navegador. Una vez le llegue el *bind TCP* del navegador, el nodo comenzará el *streaming* del vídeo. Para parar el flujo de vídeo, se mata el proceso que recoge las imágenes de la cámara y el socket TCP.

Actualmente hay un problema detectado con el USB por parte de la *Raspberry Pi*. Si se

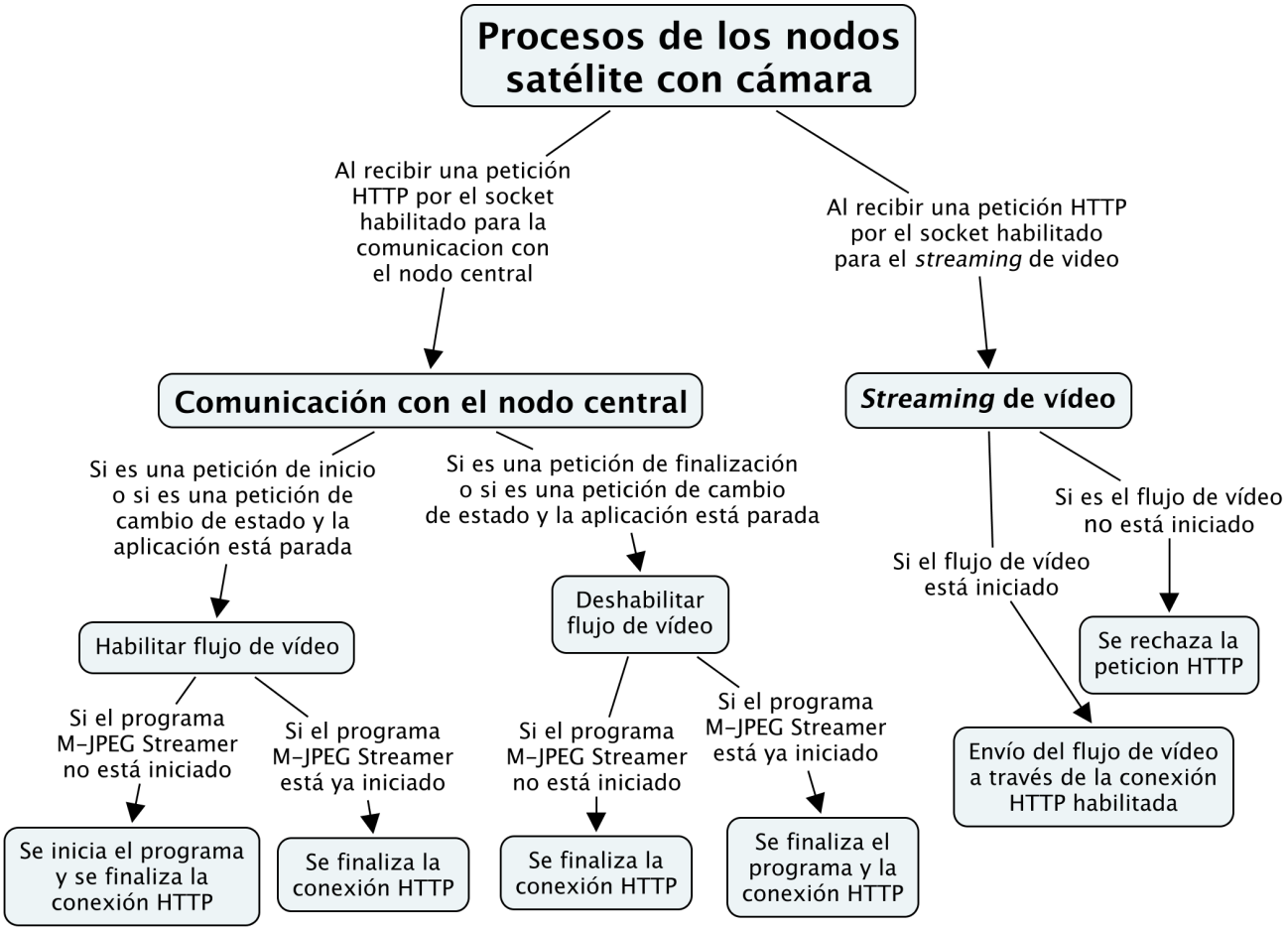


Figura 5.10: Esquema lógico del software programado en los nodos satélite con cámara.

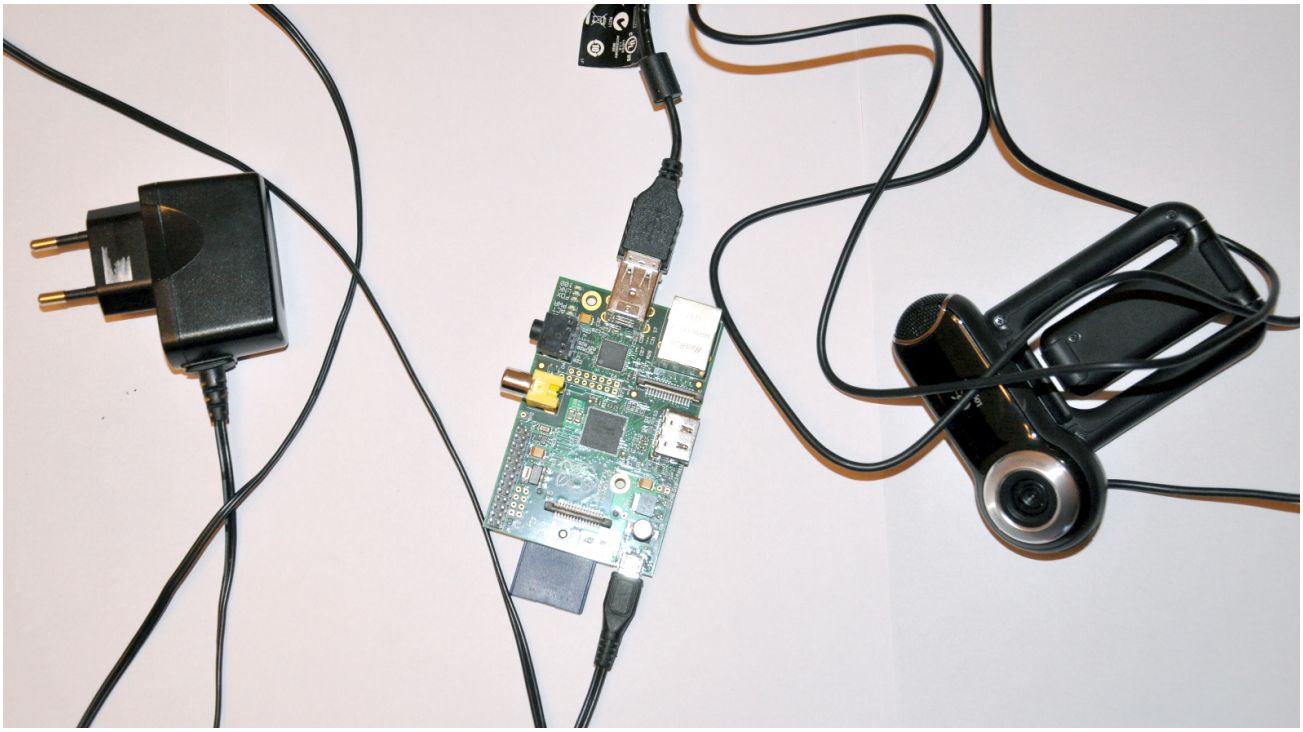


Figura 5.11: Nodo satélite con cámara.

requiere un acceso a grandes cantidades de datos a través del USB, la CPU pide datos aunque el USB aún no los tenga listos (la CPU es mucho más rápida que el puerto USB), por lo que la ocupación de la CPU queda cercana al 100 % (en nuestro caso, al realizar el *streaming*, la ocupación de la CPU está al 90 %). En el sistema actual no supone un gran problema (más allá del consumo y calentamiento extra que supone), pues no habrá otro componente activo simultáneamente con el servidor de streaming en la misma Raspberry Pi que necesite usar la CPU. Gracias al uso del script "listener.py" podemos paliar en gran medida el problema de la *Raspberry Pi*, parando el *streaming* cuando no se necesita.

5.4. Instalación

Uno de los requisitos es una instalación y puesta a punto sencilla. La instalación del hardware se ha conseguido que sea muy sencilla y sin necesidad de obra alguna: simplemente, se coloca el nodo en el lugar deseado según las pretensiones del usuario y, en el caso de los nodos de cámara o el nodo central, se enchufa a la red eléctrica.

La instalación del software, sin embargo necesita actualmente de participación por parte del usuario, aunque se ha conseguido minimizar este esfuerzo. Para efectuar la instalación hay que seguir los siguientes pasos:

1. Primero hay que comprar y ensamblar el hardware tal como se ha indicado en este escrito.

2. Posteriormente el hardware hay que programarlo con el software creado en este PFC, para ello hay que distinguir entre nodos Arduino o Raspberry.

   - Para nodos Raspberry (el nodo central o nodos satélite con cámara), el usuario puede instalar todos los programas necesarios para cada nodo (Django, M-Jpeg, módulos Python, etc. según el caso), añadir los ficheros de este proyecto y programar la placa *Raspberry* para que se ejecuten los programas del sistema al arrancarse, tal como se ha ido describiendo a lo largo de esta memoria.

   Otra opción más sencilla es restaurar una imagen de una tarjeta SD con el sistema configurado y con todos los programas ya instalados. Así el software se instala automáticamente en sólo dos pasos (hay que restaurar la copia y expandirla para que ocupe toda la tarjeta). Hay múltiples páginas que explican este proceso (ej. `http://faunageek.com/blog/?p=197`), sólo habría que utilizar el volcado de la tarjeta SD que se encuentra en el SVN del proyecto [1], en lugar de la imagen con el sistema operativo estándar.

   Una vez iniciemos la placa con la imagen proporcionada, restaría modificar el fichero "/etc/wpa_supplicant/wpa_supplicant.conf" para introducir los datos de la conexión Wi-Fi a la que queramos conectar los nodos y reiniciar. Si se quieren mantener los puertos estándar del proyecto (puerto 80 para el servidor web, 8080 para visualización del *streaming* y 5555 para recibir las órdenes en el nodo cámara), no es necesaria ninguna otra modificación, pues los scripts de arranque recogen dinámicamente la dirección IP que se le asigne al nodo.

   - Para los nodos Arduino, el usuario tiene que cargar el programa creado en este PFC. Según el tipo de sensor, habrá de utilizar un fichero de programa ".ino" u otro. Todos los programas están publicados en el svn del proyecto[1]). Únicamente se ha de modificar la variable "id" del programa para que cada nodo del mismo tipo tenga un *id* diferente (no es necesario que sean consecutivos).

   Los módulos ZigBee, si bien se pueden utilizar tal como vienen de fábrica, conviene modificar su parámetro "PAN ID" para evitar colisiones con otros sistemas[3]. Se puede ver un ejemplo de configuración en la figura 4.10.

3. Con todo el software instalado y tras reiniciar las máquinas, conviene comprobar que tienen conexión entre sí. Por último, hay que introducirse en el panel de administración de *Django* para modificar la base de datos. Para ello hay que introducir los datos de los satélites que se vayan a utilizar (indicar el tipo e identificador que le hemos otorgado), así como los de las cámaras (IP y puertos de escucha).

4. Una vez configurados los datos de los satélites, el sistema es completamente autónomo. Sólo se deberán modificar estos datos en caso de querer eliminar o introducir un nuevo satélite, en cuyo caso hay que repetir los pasos únicamente para el nodo satélite y registrarlo en la base de datos.

[3]No puede haber dos coordinadores con el mismo identificador de red. Si esto ocurre y están en canales distintos (no se solapan en frecuencia), el nodo satélite se conectará el primero que encuentre, que puede no ser el del sistema en el que pretende ser instalado. Si además los coordinadores se encuentran en la misma frecuencia, habrá colisiones a nivel de protocolo que impedirán las comunicaciones

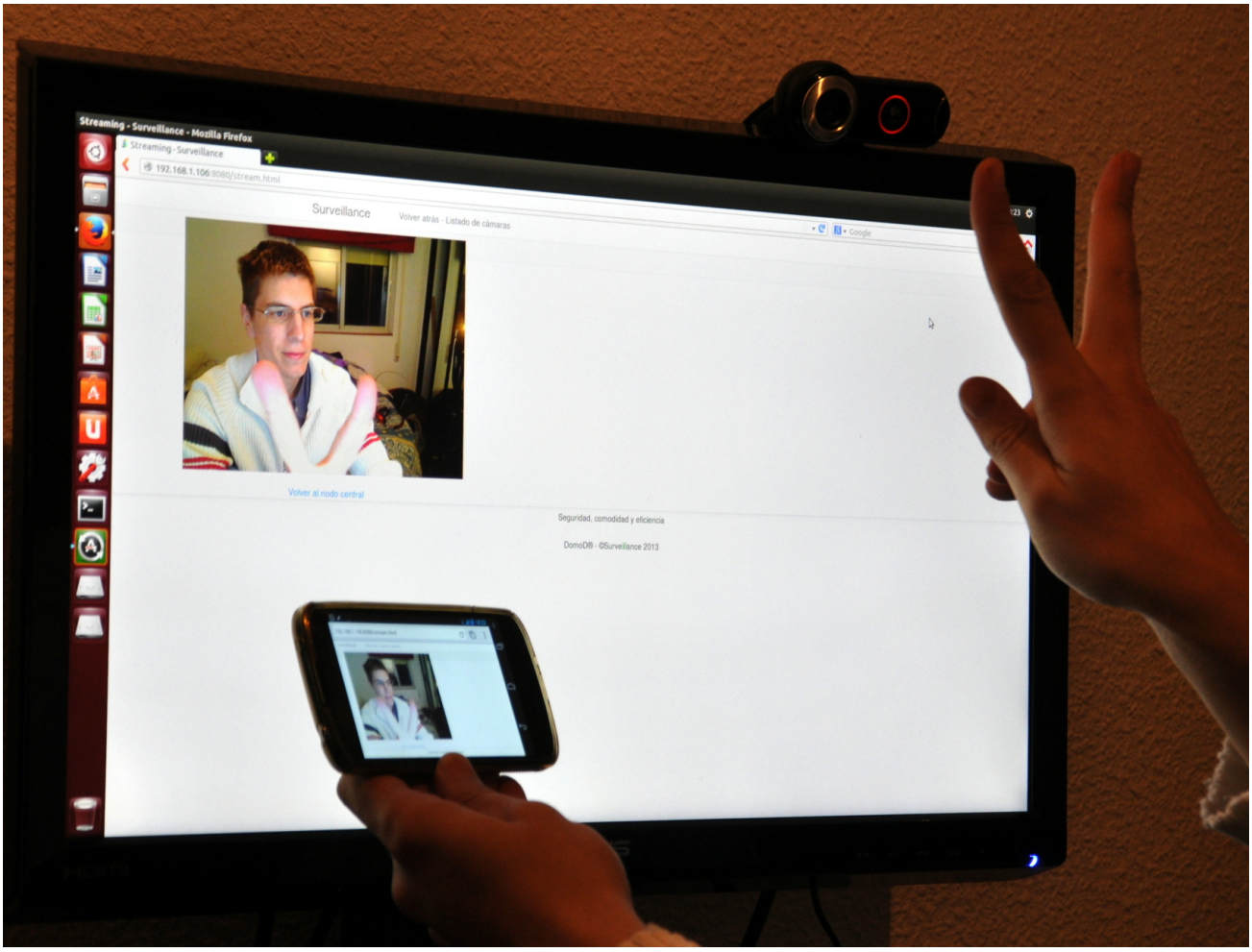


Figura 5.12: Prueba del streamin de vídeo simultáneo en *Ubuntu* y *Android*

5.5. Experimentos

Al igual que la versión anterior, el sistema se ha probado con dos nodos satélite ZigBee simultáneamente. Los nodos ZigBee se han ubicado en una habitación, aproximadamente a 16 metros del nodo central, que se encontraba en el comedor, con tres paredes entre ambos. En este caso se han podido añadir datos de prueba en la base de datos, a través del script creado "data_test.py", para simular la presencia simultánea de varios satélites.

El nodo central ha permanecido encendido de forma continua durante 30 días, sin mostrar pérdida de rendimiento ni calentamiento excesivo. Se ha probado a acceder tanto desde un ordenador como desde un dispositivo Android, desde la misma LAN y desde una conexión exterior, así como a realizar accesos simultáneos. Estas mismas pruebas han sido exitosas para el *streaming* de vídeo.

Se ha probado a cortar la alimentación a todos los nodos, sin realizar ninguna acción previa, para simular un apagón. Todos los nodos se volvieron a encender sin incidencias, quedando todo cunfigurado igual que estaba antes de apagar los nodos. Por ello podemos concluir que en

Surveillance 4.0 no hay ningún tipo de pérdida de estabilidad respecto a Surveillance 3.0.

Se han grabado vídeos demostrativos del sistema en funcionamiento, los cuales se pueden encontrar en los enlaces [32], [33] y [34].

5.5.1. Limitaciones

Se han encontrado una serie de limitaciones en esta versión que conviene tener en cuenta:

Max. satélites Por limitaciones propias del módulo XBEE, sólo se pueden conectar simultáneamente 10 satélites a un mismo nodo central.

Pérdida de enlace Wi-Fi El sistema operativo *Raspbian* permite que los nodos se conecten a una red Wi-Fi concreta, pero si por cualquier motivo en el arranque de la placa no puede conectarse a dicha red, se intentará conectar a cualquier red que no requiera seguridad y, por lo tanto, no se conectaría al router deseado. Esto provocaría que los satélites no se pudiesen comunicar con el nodo central, aunque en cualquier caso la información de los satélites ZigBee se seguiría recogiendo y procesando sin ningún inconveniente. Si una vez solventado el problema que impedía la conexión con el router deseado se reinicia el nodo, este se asociará al router correcto.

Capítulo 6

Conclusiones y discusión

El sistema domótico conseguido es funcional y muy práctico que puede ser ampliado en muchos ámbitos. Se ha construido un sistema complejo, diseñando tanto la parte hardware como software, e incluso se ha diseñado un protocolo de comunicaciones para adaptarlo a las necesidades del sistema y aprovechar lo mejor posible los recursos disponibles. Se han explorado e integrado varias tecnologías de diversa índole en un mismo sistema (ZigBee, servidor web, diferentes sensores, actuadores y cámaras).

Los objetivos propuestos se han alcanzado adecuadamente. Se ha obtenido un sistema domótico completo (sensores, alarmas, actuadores y cámaras) como se pretendía. El actuador implementado (encendido y apagado de luces) actúa a través de un relé, el cual podría usarse para activar otro tipo de elementos eléctricos sin necesidad de ninguna modificación en el sistema.

La mayoría de los sistemas domóticos que podemos encontrar actualmente en el mercado se centran únicamente en una de las tras partes de la domótica (generalmente en tele-vigilancia). Es difícil encontrar un sistema domótico completo, de hecho, el presente proyecto es el que tiene mayor diversidad de sensores de entre los revisados. Esto constituye una gran ventaja respecto a los sistemas mencionados en el apartado 1.2, pues no consiste únicamente en un sistema anti-intrusos, como son la mayoría de sistemas domóticos actuales, si no que también nos puede avisar inmediatamente de posibles averías para que podamos actuar con celeridad, reduciendo tanto los daños materiales como la posibilidad de sufrir un accidente en el hogar.

El sistema final es altamente escalable, pues se ha puesto énfasis en utilizar el menor número de recursos posibles, manteniendo la funcionalidades y siempre de forma modular. Los protocolos utilizados permiten añadir nuevos nodos, ya sean alguno de los mencionados en este PFC o nodos con nuevas funcionalidades (el protocolo propio WHAP4.3.3 permite definir hasta 255 tipos de nodos diferentes). Todo esto permite que el sistema sea viable y extensible de cara al futuro, pues los protocolos y el código utilizado es suficientemente laxo para prever posibles modificaciones posteriores.

Se ha conseguido un sistema que, una vez instalado, puede ser usado fácilmente por cualquier persona, aunque carezca de conocimientos técnicos. Por otro lado, al ser áltamente modular, el usuario puede diseñar su propio sistema de forma fácil, adecuándolo a sus necesidades.

Los objetivos propuestos en el PFC fueron casi todos cubiertos con Surveillance 3, mejorándolos y cubriendo el resto con Surveillance 4. A continuación se muestran con más detalle:

Bajo coste A continuación se detalla el coste de producción de cada componente, teniendo en cuenta los precios al por menor a los que fueron adquiridos. No incluye los elementos utilizados en sus primeras versiones que después fueron sustituidos o eliminados. Tampoco se incluye, en los nodos actuadores, el actuador (lámpara, motor de persianas, etc), debido a la gran variedad de precios entre los componentes.

Nodo central		Nodo cámara		Nodo sensor/actuador
Raspberry Pi: 27,40 euros Pincho Wi-Fi: 4 euros Módulo XBee: 20,45 euros Fuente a.: 6,21 euros		Raspberry Pi: 27,40 euros Pincho Wi-Fi: 4 euros Cámara web: 30 euros F. alimentación: 6,21 euros		Placa Arduino 20,00 euros ATMega328p: 2,00 euros Módulo XBee: 20,45 euros Sensor/relé: 2-6 euros Batería: 4 euros
Coste total: 59,46 euros		Coste total: 69,01 euros		Coste total: 52,45 euros

Como podemos ver, el precio de cada nodo es bajo si lo comparamos con los sistemas comerciales actuales. Además, todo el software de terceros utilizado en este PFC es *freeware*, de libre uso (la mayoría, de código abierto), por lo que no supone un coste adicional a lo mencionado.

Inalámbrico Se ha utilizado la tecnología ZigBee para ofrecer esta funcionalidad, de forma satisfactoria para los nodos sensores, alarma y actuadores. Si bien el nodo central y los satélites con cámara se conectan de forma inalámbrica mediante ZigBee, necesitan conexión a la red eléctrica.

Bajo consumo eléctrico El resultado en este apartado no ha sido totalmente satisfactorio, en total, el sistema consume 90mA mientras envía/recibe información y 48mA en reposo (agota una batería en aprox. 40 horas). El regulador de potencia de la placa Arduino consume 22mA aprox., por lo que sólo podríamos conseguir un consumo realmente bajo sise prescinde de dicha placa y se realizan algunos ajustes extra, como, por ejemplo, bajar la velocidad del procesador a 1 MHz (en lugar de los 8 que utiliza por defecto).

Acceso desde Internet Este requisito está conseguido, siempre que el router al que esté conectado el sistema permita el intercambio de información con Internet. Se puede acceder al sistema desde cualquier dispositivo con navegador web incorporado (ordenadores, teléfonos móviles inteligentes, etc.).

Facilidad de uso Aunque no ha sido probado por personas sin conocimientos técnicos, la interfaz de usuario es bastante simple, por lo que se entiende que su uso será asequible para cualquier persona.

Interoperabilidad Se ha conseguido interoperar distintos dispositivos (Arduino, Raspberry, PC's), así como integrar en el mismo sistema sensores de distinta índole (luz, gas, inundación, cámaras, etc.) y actuadores en el mismo sistema.

Espacio reducido Los modelos conseguidos tienen las siguientes medidas: 130mm x 56mm x 21mm los nodos *Raspberry* más el transformador (y la webcam en el nodo cámara) y 64mm x 53mm x 24mm más el sensor/actuador los nodos *Arduino*. Son nodos de tamaño muy reducido, especialmente comparadas con las de otros sistemas domóticos.

Comparándolo con los sistemas comerciales actuales (*Securitas Direct*, *Prosegur*...) vemos que el rango de sensores que podemos implementar es mucho más amplio. Además, hemos conseguido el resto de objetivos que nos propusimos: bajo precio, sin cables, arquitectura de software libre, etc. lo cual otorga bastantes ventajas frente al resto de sistemas.

Comparando las dos versiones creadas en este PFC, las diferencias entre Surveillance 4.0 con la versión 3.0 son: sustitución el nodo central por un sistema embebido de mayor capacidad de procesamiento, lo que nos permite mucha mayor funcionalidad. Se ha desarrollado módulos para la gestión de estadísticas (ej. generación de gráficas), mayor flexibilidad en la configuración del sistema, así como una interfaz visualmente más atractiva y funcional. Además, al estar basado el nuevo nodo central en un sistema *Unix*, nos permite reutilizar componentes desarrollados para esta plataforma *Jderobot* como *ElderCare*, etc. pudiendo otorgar una mayor funcionalidad y versatilidad. En Surveillance 4.0 se ha implementado el uso de actuadores, pudiendo así interactuar el sistema con la casa. También se ha creado el satélites con cámara para realizar *streaming* de vídeo y los satélites actuadores.

Al tratarse esta memoria de un proyecto fin de carrera, no pueden dejarse de lado el aprendizaje conseguido. El haber tenido que lidiar con distintos lenguajes de programación (Python, C/C++, HTML, JavaScript y CSS) y tecnologías tan diversas (*Django*, *Arduino*, *Raspberry*, diferentes sensores, ...) ha dado pié a no pocos problemas, los cuales han sido enfrentados con los conocimientos adquiridos previamente o con otros nuevos adquiridos en la realización de este PFC. También hay que recordar que se han realizado todas las fases posibles en un proyecto de I+D+I+D, desde el planteamiento de la arquitectura inicial hasta la creación de un producto funcional, cuidando costes, diseño e interfaces sencillas, etc.

6.1. Mejoras propuestas

Hay varias mejoras que se plantearon en algún momento de la realización de este proyecto y que, por no alargar más el proyecto, no han sido implementadas. Estas propuestas mejorarían el rendimiento u otros aspectos del sistema, pero en ningún caso son necesarias para el funcionamiento del mismo.

Las mejoras propuestas al término de este PFC son:

Instalación real Por falta de presupuesto, no ha podido ser probado simultáneamente el sistema completo, con muchos nodos de varios tipos a la vez (sólo ha sido posible trabajar con dos nodos ZigBee simultáneamente). Se recuerda que se ha probado la funcionalidad de todos los elementos citados en este documento por partes, es decir, varios nodos satélite simultáneamente, pero no todos a la vez, lo que debería ser suficiente y no diferir en comportamiento con respecto a una instalación real en la que haya muchos sensores simultáneos.

Xtion El siguiente paso para este proyecto será el de utilizar "cámaras" *Xtion* para poder crear alarmas volumétricas. El *Xtion* puede actuar como un sensor de movimiento, pero capaz de discernir si lo que se mueve es una persona, un animal, sólo son sombras, etc. evitando falsos positivos. Además, este dispositivo se puede utilizar de igual manera aún a falta de luz. Aparte de crear la alarma pertinente, puede enviar la señal de vídeo, tanto la imagen RGB (como una cámara normal) cómo el nivel de profundidad.

Aplicación *Android* Aunque actualmente se puede utilizar el sistema a través de un navegador web instalado en un dispositivo *Android*, se plantea como mejora la creación de una aplicación exclusiva para este tipo de plataforma. De este modo, la disponibilidad y visualización no dependería del navegador web.

Auto-configuración (satélites) Actualmente, los satélites han de estar configurados con ciertos parámetros del sistema antes de conectarse al mismo (ver sección de instalación 5.4). Todos los parámetros podrían ser proporcionados, al acoplar un nuevo nodo satélite, por el nodo central mediante un protocolo de auto-descubrimiento.

Abaratamiento de costes y menor consumo de nodos *Arduino* Con el hardware adecuado, se puede prescindir de la placa *Arduino* y utilizar directamente el procesador *ATMega328p*. Con ello, el precio de los nodos satélite con *Arduino* descendería sustancialmente. También se aprovecharía para utilizar un conversor de voltaje más eficiente, aumentando la vida útil de las baterías. Por último, se puede poner a dormir a los nodos satélite (dejarlos en un estado en el cual casi no consumen electricidad, pero no pueden realizar acciones) y despertarlos únicamente durante el envío de datos para minimizar el consumo de batería.

Por otro lado, se analizaron otros métodos para el envío del *streaming* de vídeo que permitiesen utilizar la GPU y el chipset conversor a H.264 integrados en la *Raspberry*, junto al uso de HTML5 para enviar el flujo de vídeo en lugar de imágenes. El principal problema que se encuentra con el envío de vídeo en lugar de imágenes es el retardo, pues, además de que se utilizaría más recursos (las operaciones son más complejas), tenemos que esperar a tener un "paquete" de imágenes para poder comprimirlas. Por ejemplo, para un fps de 20 (suponiendo que se pudiese enviar a esa velocidad), si formamos un grupo de 4 imágenes (4 "frames"), tendremos que esperar 4(imágenes)/20(imágenes/seg.)=0'2seg., a lo que habría que añadir el propio retardo generado por el procesamiento, red, decodificación, etc.

Se barajaron dos posibilidades para el envío de imágenes desde el nodo satélite al usuario, las cuales se detallan a continuación:

Directo El nodo satélite se comunica con el cliente final (navegador web) sin pasar el flujo de vídeo por el nodo central. La ventaja de este método consiste en que las imágenes sólo se transmiten una vez (dentro de la red Wi-Fi de la casa), lo cual se traduce en una menor latencia y menor uso del ancho de banda del canal Wi-Fi.

Por otro lado, tiene una desventaja bastante acuciante: una vez abierto el flujo de vídeo, cualquier usuario podrá visualizar el mismo capturando el tráfico, pues el satélite no cifra el contenido (no hará uso del túnel SSH que crea el nodo central).

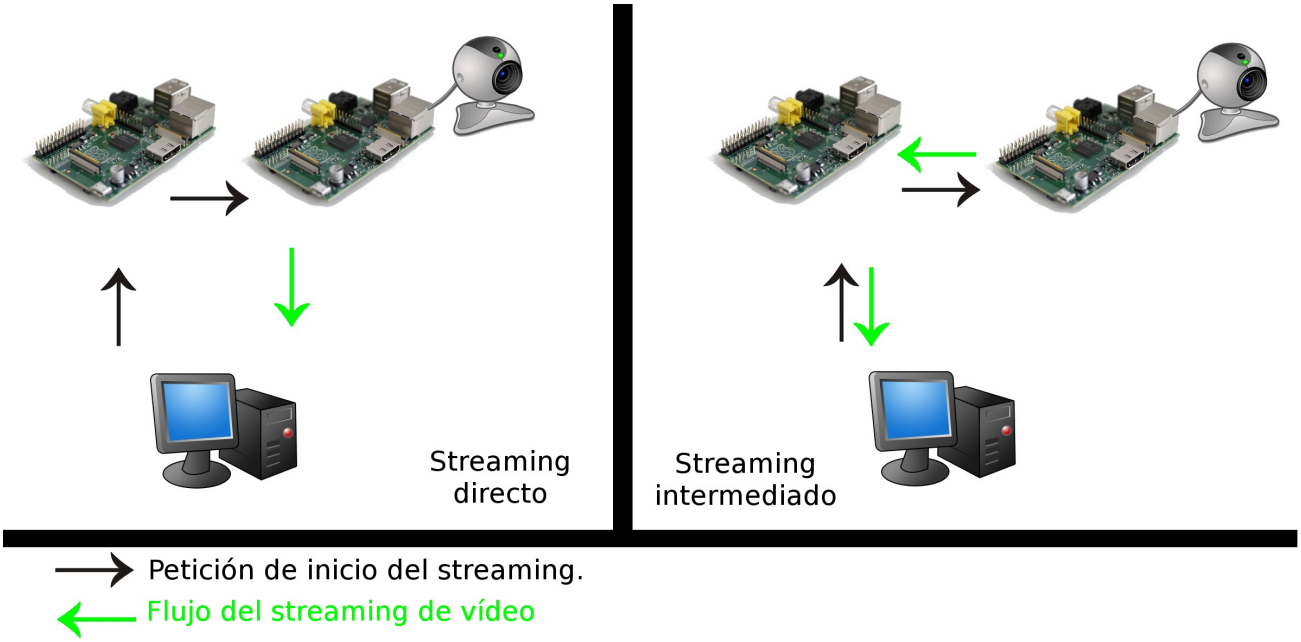


Figura 6.1: Esquema sobre las opciones barajadas para el flujo de vídeo.

Este método tiene asociada otra desventaja que puede ser especialmente molesta para los usuarios que no tengan conocimientos técnicos, pues si quieren acceder a la visualización de imágenes fuera de la LAN, habrán de abrir los puertos correspondientes al streaming de vídeo de los satélites en el router correspondiente.

Intermediado El nodo satélite envía las imágenes al nodo central, el cual se encarga de hacer llegar las imágenes al usuario.

Las ventajas de éste método son varias. Primero, Tenemos el control siempre en el nodo central, pudiendo usar exactamente el mismo formato de páginas web que en el resto de páginas. Segundo, la dirección IP y puerto a la que apunta el usuario podrían ser los mismos para la visualización del streaming que para el resto de páginas del sistema. Existe la posibilidad de que, si se cifra la conexión entre el nodo central y el usuario (mediante SSH), el flujo de vídeo también quedaría securizado.

Las desventajas en éste caso también son múltiples: utilizaríamos el doble de ancho de banda (transmitimos desde el satélite al nodo central y desde éste al router), lo cual perjudica sobremanera a la tasa de refresco (fps) que vamos a poder conseguir. Además, estaremos cargando el nodo central, pues este ha de modificar las cabeceras HTTP que le lleguen desde el satélite y, aunque no modifique los datos de las imágenes, ocupará procesador en retransmitirlos.

Por otro lado, se ha pensado en otras funcionalidades que podrían complementar el proyecto, pero que a priori serían más costosas de implementar:

Módulo GSM/GPRS Dado que las redes Wi-Fi/ADSL implantadas en los hogares españoles no son especialmente estables, y en muchos casos ni siquiera están disponibles (poblaciones

aisladas, casas de campo o en la montaña...), sería conveniente tener la opción de añadir un módulo que pueda interactuar con la red de telefonía inalámbrica, de esta forma, el usuario podría ser avisado mediante mensajes de texto plano (SMS) de cualquier alerta o anomalía para la cual se hubiese configurado este servicio, o incluso se podría utilizar esta conexión como punto de acceso desde Internet. Por supuesto, habría que añadir al presupuesto el mantenimiento de la línea y el coste los SMS.

Exploración de la transmisión de vídeo La placa *Raspberry Pi* tiene un codificador H.264 hardware integrado. Convendría que analizar más a fondo la posibilidad de codificar las imágenes de *streaming* en dicho formato, ocupando así menor ancho de banda, pudiendo alcanzar un mayor número de FPS y prescindiendo del uso de software de terceros para esta tarea. Aún con el codificador, esta configuración aumentaría el retardo por las circunstancias comentadas en la sección 5.3.3.

Securización del canal mediante SSL (HTTPS) Para poder acceder al sistema desde Internet mediante un navegador web de forma segura, sería necesario el cifrado de la información mediante SSL (con el protocolo HTTPS).

Bibliografía

[1] Repositorio SVN habilitado en *Jderobot* para este proyecto. `https://svn.jderobot.org/users/d.castellanob/pfc-teleco/trunk/`

[2] Bitácora alojada en los servidores de *Jderobot* utilizada para informar del estado del proyecto. `http://jderobot.org/D.castellanob-pfc`

[3] Distributed Video surveillance System based on Android, Roberto Calvo Palomino, Master project, 2010. `http://jderobot.org/index.php/Rocapal_Enabling_Technologies_LS`

[4] Orellana Galloso, Rubén,*Sistema de monitorización y control de riego e iluminación basado en tecnologías Arduino, Zigbee, Android y Bluetooth para un emplazamiento en Cádiz*, España, 2013.

[5] Proyecto jderobot. Middlelware para el desarrollo de componentes robóticos y domóticos. `http://jderobot.org/index.php/Main_Page`

[6] Detección de caídas en hogar basado en cámaras y kinects. `http://jderobot.org/index.php/ElderCare`

[7] Monográfico sobre el estándar X10. Historia, protocolo, implementaciones, etc. de este estándar. `https://forja.rediris.es/docman/view.php/414/761/X10.pdf`

[8] Página web del estándar KNX. `http://www.knx.org/es/`

[9] Norma UNE-EN 50065-1:2012 para la transimisión de sañales por la red eléctrica. `http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0048827`

[10] Página web del la alianza empresarial para la difusión de *Z-wave*. `http://www.z-wavealliance.org/`

[11] Sistema de seguridad domótica de la empresa *Securitas Direct*. `http://www.securitasdirect.es/myverisure.html`

[12] Alarma para el hogar *Prosegur Proview +*. `http://www.alarmahogarprosegur.com/?o=web`

[13] Sistema domótico textitDelta Dore. `http://www.deltadore.com/`

[14] Página web de la compañía *Libelium*. `http://www.libelium.com/company/`

[15] Estándar de la última especificación (2011-04-12) del lenguaje de programación C. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf

[16] Página oficial del lenguaje de programación Java SE. http://www.oracle.com/technetwork/java/javase/overview/index.html

[17] Documentación de las funciones del lenguaje Java SE versión 7 (*javadoc*). http://docs.oracle.com/javase/7/docs/api/

[18] Página oficial del lenguaje de programación Python. http://www.python.org

[19] Fielding, Roy T.,*Architectural Styles and the Design of Network-based Software Architectures*, EE.UU., 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[20] Página oficial del programa *M-JPEG Streamer*. http://sourceforge.net/projects/mjpg-streamer/

[21] *XBee* Multipoint RF Modules, *Digi International Inc*, 2011. http://www.digi.com/pdf/ds_xbeemultipointmodules.pdf

[22] Página oficial del proyecto Arduino. http://arduino.cc/

[23] Página ofical de la *Raspberry Pi*. http://www.raspberrypi.org/faqs

[24] Esquema hardware de la *Raspberry Pi*. http://www.raspberrypi.org/wp-content/uploads/2012/10/Raspberry-Pi-R2.0-Schematics-Issue2.2_027.pdf

[25] API de programación en Python para Django. https://docs.djangoproject.com/en/1.6/py-modindex/

[26] Wiki para el sistema de sensores de *Seeedstudio* utilizados en Surveillance 3 y 4. http://seeedstudio.com/wiki/GROVE_System

[27] Página oficial de OpenGL ES. http://www.khronos.org/opengles/

[28] Especificaciónes de los estándares *IEEE 802.11* (*Wi-Fi*). http://standards.ieee.org/about/get/802/802.11.html

[29] Especificación del estándar *IEEE 802.15.4* para la creación de redes inalámbricas. http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf

[30] Página web de la ZigBee Alliance, con información sorbe el consorcio y la especificación ZigBee. http://www.zigbee.org/Specifications/ZigBee/Overview.aspx

[31] Videos demostrativos de Surveillance 3.0, en el portal de vídeos *Youtube*. http://www.youtube.com/watch?v=2hWt0g0jHb8&list=PL0y2HkZZUAoz1xh27lU_Elu0clUYXRiEZ8

[32] Video demostrativo de la interfaz de Surveillance 4.0, en el portal de vídeos *Youtube*. http://www.youtube.com/watch?v=KA8Njb6a31k

[33] Video demostrativo sobre los sensores y alarmas en Surveillance 4.0, en el portal de vídeos *Youtube*. http://www.youtube.com/watch?v=sKqnJDti49A

[34] Video demostrativo sobre los actuadores y nodos cámara de Surveillance 4.0, en el portal de vídeos *Youtube*. http://www.youtube.com/watch?v=96yfZmr4UiU

[35] Librerías de javascript "*JQuery*", utilizadas en la interfaz web de Surveillance 4. http://jquery.com/

[36] Librerías de javascript "*Bootstrap*", utilizadas en la interfaz web de Surveillance 4. http://getbootstrap.com/

[37] Plugin de JQuery utilizado para la visualización de las gráficas en la interfaz web. http://www.jqplot.com