Universidad Zaragoza
1542

Proyecto Fin de Carrera

Ingeniería de Telecomunicación

Sistema domótico *low-cost* con Android, Arduino y Raspberry Pi

Autor

Carlos Ceamanos Gaya

Director

Luis Manuel Ramos Martínez

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Año 2017

Agradecimientos:

A mi familia y en especial a mi mujer y a mi hija, por darme todo su apoyo y su cariño en todo momento, que me ha servido para llevar a buen puerto la tarea de finalizar este proyecto. Por su comprensión en los buenos y malos momentos, por los enfados que haya podido tener y por todos los momentos que les he robado del tiempo de estar con ellos y que sabré compensarles ahora que esto llega a su fin; mi más sincero agradecimiento.

Quiero también expresar mi agradecimiento a Luisma, por haberme dirigido en este proyecto de fin de carrera, y comprometerse conmigo desde el primer momento sin ni siquiera haberme conocido. Por su confianza en mí y por haberme animado a plasmar la idea que tenía en la cabeza, por todas las atenciones, por el tiempo dedicado, y sobre todo por el apoyo que me ha prestado, gracias.

Y en general a todos los amigos, compañeros y gente que me han ayudado a seguir adelante y formarme como persona e ingeniero durante todos estos años.

RESUMEN

Este proyecto trata de adentrarse en el mundo de la domótica y su principal objetivo es el de diseñar un sistema domótico de bajo coste que consiga adaptarse a la mayoría de los hogares de hoy con sencillez y facilidad, soportando la mayoría de las demandas que los usuarios de estos sistemas tienen en la actualidad.

Para este fin se han utilizado en el diseño plataformas de software abierto como son Arduino (microcontrolador) y Raspberry Pi (sistema Linux) así como el SO Android, ampliamente conocido. También se ha utilizado para la comunicación de los distintos componentes del sistema, la tecnología *XBee-Zigbee*, de forma que sean inalámbricos todos los dispositivos del sistema.

El sistema domótico desarrollado se fundamenta en las medidas recogidas por unos sensores que ofrecen datos al microcontrolador Arduino. Este, a través de XBee y programado en C++, remite la información que registran los sensores al cerebro central, en la Raspberry Pi (Servidor Linux con Python), donde el sistema almacena la información en una base de datos. Dicha base de datos (MySql), mantiene los estados de los dispositivos que en el sistema pueden ser activados (bombillas, electrodomésticos, persianas, etc.), y es accesible también desde el programa Android a través de un servidor Apache con PHP, enviando y recibiendo datos mediante JSON. Desde el sistema Android, podemos apagar y encender dispositivos, programarlos y configurar el sistema, a través de una aplicación diseñada y desarrollada para el control del mismo.

El proyecto se completó con un programa Android que maneja todo el sistema domótico diseñado, permitiendo al usuario interactuar con el mismo, así como configurarlo, de características similares a lo que el mercado real ofrece en la actualidad, salvando las distancias por los recursos disponibles para este proyecto.

Se ha dotado al prototipo de un sistema de alarmas configurable que avisa al usuario a través de un programa de notificaciones al móvil de las incidencias que pueda tener en su hogar, detectadas por el sistema domótico (escapes de gas, presencias indeseadas, etc.).

Asimismo, se implementa en la aplicación Android la adquisición de los datos del precio de la electricidad, aconsejando al usuario la programación de sus electrodomésticos en el momento óptimo por precio.

Contenido

1.- Introducción

En las secciones del presente apartado se van a detallar los motivos de la realización de este Proyecto Fin de Carrera, además de explicar el contexto y el entorno del mismo, para facilitar la comprensión del plan de proyecto que se ha seguido.

1.1.- Motivación

El **IoT** (*Internet of Things*) – en español, Internet de las Cosas – es uno de los términos más populares del entorno tecnológico de los últimos años, aunque parece que no termina de adoptarse dentro del ámbito comercial del gran público como ha ocurrido por ejemplo como los *Smart Phones*.

La **domótica** es el conjunto de las tecnologías que se aplican al control y la automatización inteligente de las viviendas, permitiendo una serie de prestaciones de eficiencia energética, seguridad y confort. Estas tecnologías no son tan recientes, pues uno de sus estándares más conocidos, X10, nació en 1975, pero a la domótica le ha ocurrido lo mismo que al IoT, que todavía no ha llegado a ser incluido dentro de nuestras vidas de una manera masiva.

Ambas tecnologías siempre han despertado mi curiosidad. En el momento actual, las grandes empresas tecnológicas han fijado su mirada en ambas (Google, Telefónica-Huawei, Apple), y parece que se están centrando en acercar a los consumidores de una manera sencilla y conjunta ambas tecnologías. Los primeros productos que se quieren conectar a Internet para ofrecer funcionalidades al consumidor, son los electrodomésticos y aparatos que todos tenemos en nuestras casas (sistemas de climatización, persianas, luces, cafeteras, frigoríficos, lavadoras, etc.).

Los *Smart Phones* han comenzado a formar parte activa de nuestras vidas, y prácticamente dependemos de ellos para una gran cantidad de funciones y labores sin las que nos sería impensable vivir, a pesar de que hace muy poco que disponemos de estas facilidades (volver atrás y pensar cómo nos comunicábamos sin WhatsApp, teníamos que ir al banco a realizar gestiones o hacíamos fotos entre otras actividades). De entre los sistemas operativos de los *Smart Phones*, el más implantado es **Android**, de Google.

Otro de los conceptos que ha aparecido recientemente en las TI ha sido el **DIY** (*Do It Yourself*) o traducido al español, "Hágalo Usted Mismo", al abrigo de una corriente de democratización del acceso a la tecnología y la creación de dispositivos de procesamiento "***Low Cost***" que permitan a todos acceder a los recursos que proporciona Internet. Uno de los obstáculos que ha hecho que los sistemas domóticos no penetrasen

en nuestras casas al ritmo de otras tecnologías, ha sido su elevado coste (el sistema dominante del mercado, KNX, tiene precios prohibitivos para la mayoría de los hogares).

El IoT no termina de extenderse debido a uno de los principales problemas que acosan a todos los sistemas actuales de información: la seguridad. Por ello, se utiliza como comunicación entre los componentes del sistema el estándar inalámbrico *ZigBee*, en lugar de optar por los que actualmente se implementan en casi todos los sistemas domóticos de bajo coste, que suelen ser Bluetooth o WiFi. Adoptar este sistema permite estudiar la constitución de **redes "*Mesh*"**, que están teniendo un gran desarrollo recientemente.

Uniendo todos estos conceptos (IoT, domótica, DIY, Android, Low Cost y redes Mesh), Arduino que permite manejo de sensores y actuadores, Raspberry Pi, los sistemas LAMP (Linux-Apache-MySql-PHP), y Android como SO de un *Smart Phone*, se implementará un prototipo de sistema domótico de bajo coste.

1.2.- Contexto.

Como se ha explicado, este proyecto se enmarca dentro de lo que se ha venido a llamar el "Internet de las cosas" (***Internet of things***) concepto que engloba la interconexión de todos los objetos cotidianos con Internet, y el **DIY** (*Do It Yourself*). En la actualidad todavía no existe la posibilidad de conectar directamente una gran cantidad de dispositivos de nuestro día a día con Internet, debido a que los fabricantes no han creído conveniente, bien por precio, bien por utilidad, implementar los sistemas necesarios para que elementos tales como una cafetera, un lavavajillas, nevera o lámpara, tengan esa facilidad de conexión.

Pero en el mundo de la tecnología todo evoluciona a gran velocidad: Samsung ya ha lanzado al mercado sus primeros electrodomésticos que se pueden manejar a través de un móvil Android, y otras muchas marcas tienen su estándar preparado para lanzar nuevos modelos de aparatos que contengan dicha facilidad (la diversidad de estándares puede ralentizar la evolución de estas tecnologías).

Vemos en la prensa indicadores de lo que va a ser el futuro inmediato, de lo que las compañías tecnológicas punteras pretenden, como la compra de la empresa Nest por parte de Google, empresa que dispone en su portfolio de productos de termostatos y detectores de humo inteligentes, configurables y manejables desde cualquier dispositivo conectado a la red, y más recientemente de la compañía Dropcam, dedicada a las cámaras de vigilancia. Google ha invadido casi todas las parcelas de la red, y ahora ha vuelto sus ojos hacia el "hogar inteligente". Telefónica también ha lanzado junto con Huawei un kit domótico adaptable para el hogar.

También ciertas empresas de electrónica están implementando el estándar DLNA en sus productos, y dotándolos de conectividad a internet, con lo que les dan acceso a datos y a la posibilidad de controlarlos desde el exterior: el caso más usual es el de los televisores.

Pero no todo son bondades. La domótica actual peca de varios defectos: precio muy alto, y dificultades para su instalación (necesidad de gran cantidad de obras para su

instalación). Pero mitigando estos inconvenientes, es un campo poco explotado dentro de la tecnología, y que reportará grandes beneficios en el momento que el gran público acceda a ella.

Con estas dos premisas principales se ha trabajado: lograr rebajar el precio del sistema domótico y hacer posible una instalación de los dispositivos en el hogar de la manera menos invasiva posible.

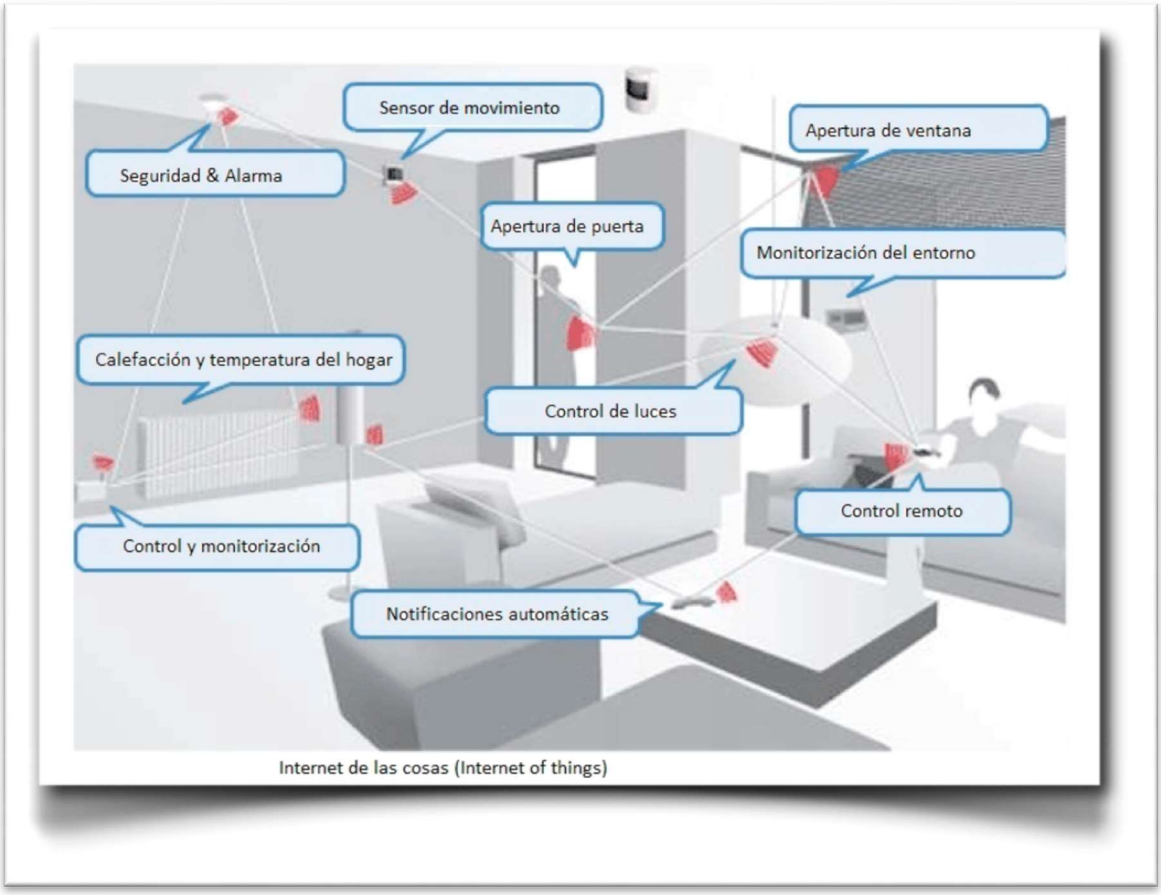


Figura 1. Internet de las cosas

1.3.- Objetivos del proyecto.

Una de las motivaciones iniciales del proyecto es el precio actual de los sistemas domóticos propietarios (sobre todo KNX). Por ello, el objetivo fundamental del mismo es conseguir un prototipo operativo que sirva como base para un sistema con un coste mínimo, ajustado y competitivo respecto al precio de los sistemas domóticos comerciales actuales, y equiparable en prestaciones.

Debido a esta premisa inicial, se han buscado dispositivos de bajo precio, desde los sensores a las placas controladoras, y por ello se ha vuelto la vista hacia dispositivos de *hardware* libre y no a sistemas propietarios. **Arduino** y los componentes que tiene éste asociados en el mercado (sensores y actuadores), como cliente del sistema; como servidor del sistema un micro-ordenador de bajo coste con distribución Linux como S.O. como es **Raspberry Pi**. Desde el comienzo del diseño de la arquitectura del prototipo se

descarta la utilización de componentes de domótica de sistemas propietarios, debido a su alto precio, e incluso se descartó que fuese el Gateway o cerebro del prototipo un PC convencional, ya que eso dispararía el precio.

Se plantearon varios requisitos indispensables que se pretendían alcanzar: debía ofrecerse información en tiempo cuasi-real de las variables del entorno (temperatura, humedad, gas en aire, agua en suelo, etc.); debía poder ser controlado y monitorizado desde el teléfono móvil, estuviese donde estuviese el usuario; tendría que emitir alarmas hacia el teléfono móvil del usuario acerca del cambio de estado de ciertos eventos, como presencia de gas en aire, o presencia de intrusos en el hogar.

También se fijó como requisito que los dispositivos que conformasen el prototipo, deberían ser inalámbricos para evitar tener que realizar obras y facilitar la instalación en el domicilio del usuario. Para este objetivo, se estudiaron varias soluciones: Arduino dispone de varios métodos de comunicación inalámbricos, como *shields* de comunicación radio a 433 Khz., *shields* de comunicación Wi-fi, *shields* de comunicación por GSM, etc. Finalmente se eligió la utilización de la tecnología *XBEE-Zigbee* por su versatilidad y características. Además, ésta era otra área que se deseaba investigar en este proyecto, debido al progreso de esta tecnología dentro de la domótica actual, y en otros campos (se utiliza para sistemas de muy diferentes características como monitorización forestal, *smart cities*, etc.). Se requeriría para el proyecto configurar redes *mesh* (redes jerarquizadas con configuración de los nodos). El estudio de la configuración de redes con dispositivos *XBEE-Zigbee* ha sido otro de los objetivos de este proyecto, aunque inicialmente no figuraba en los requisitos obligatorios, pero ha supuesto otra motivación más.

Otro de los objetivos que se ha pretendido alcanzar con el proyecto ha sido la de implementar un sistema que se adaptase a los sistemas de comunicación de datos más extendidos actualmente. Se requiere de un acceso a internet, ya sea vía wifi o móvil.

Finalmente, se fijó como requisito la utilización del sistema operativo Android para desarrollar una aplicación que interactuase con el sistema creado y que permitiese cómodamente al usuario de la misma, dirigir y controlar el mismo. Esta aplicación que podría instalarse en un teléfono o Tablet, debería tener la posibilidad de interactuar con los dispositivos conectados al sistema domótico (luces, persianas, válvulas, etc.) y configurar los mismos, así como contener ciertos parámetros de seguridad de acceso al mismo. El objetivo era conocer y aprender la programación y depuración de programas en Android.

1.4.- Resultados obtenidos

Los resultados más destacables de este PFC han sido los siguientes:

1. Se ha realizado un análisis de la potencia de aplicación de plataformas de hardware de bajo coste como son Arduino y Raspberry Pi, e integración y

adaptación a sistemas de uso cotidiano, como en este caso, un sistema de control domótico.

2. Se ha estudiado la utilización de los diferentes sensores de uso doméstico que se pueden adaptar a Arduino.
3. Se ha conseguido la reducción de costes del prototipo de un frontal de un sistema domótico real frente a los sistemas convencionales como KNX.
4. Se ha conseguido verificar la capacidad integración del sistema operativo Android con cualquier sistema real y ver su aplicabilidad sobre cualquier campo.
5. Se ha realizado la integración y estudio del sistema inalámbrico *XBEE-Zigbee* (redes *Mesh*) en aplicaciones reales, y en particular en redes de sensores, lo que ha abierto posibilidades a futuras implementaciones en diferentes usos del mundo real.
6. Se han estudiado diferentes tecnologías cliente-servidor y diferentes lenguajes de programación, necesarios para la implementación de este sistema.

1.5.- Planificación (diagrama de Gantt)

En la Figura 2, puede apreciarse el diagrama de Gantt con la planificación seguida durante la realización del proyecto, detallando las tareas seguidas y la duración de las mismas, a lo largo del año 2014.

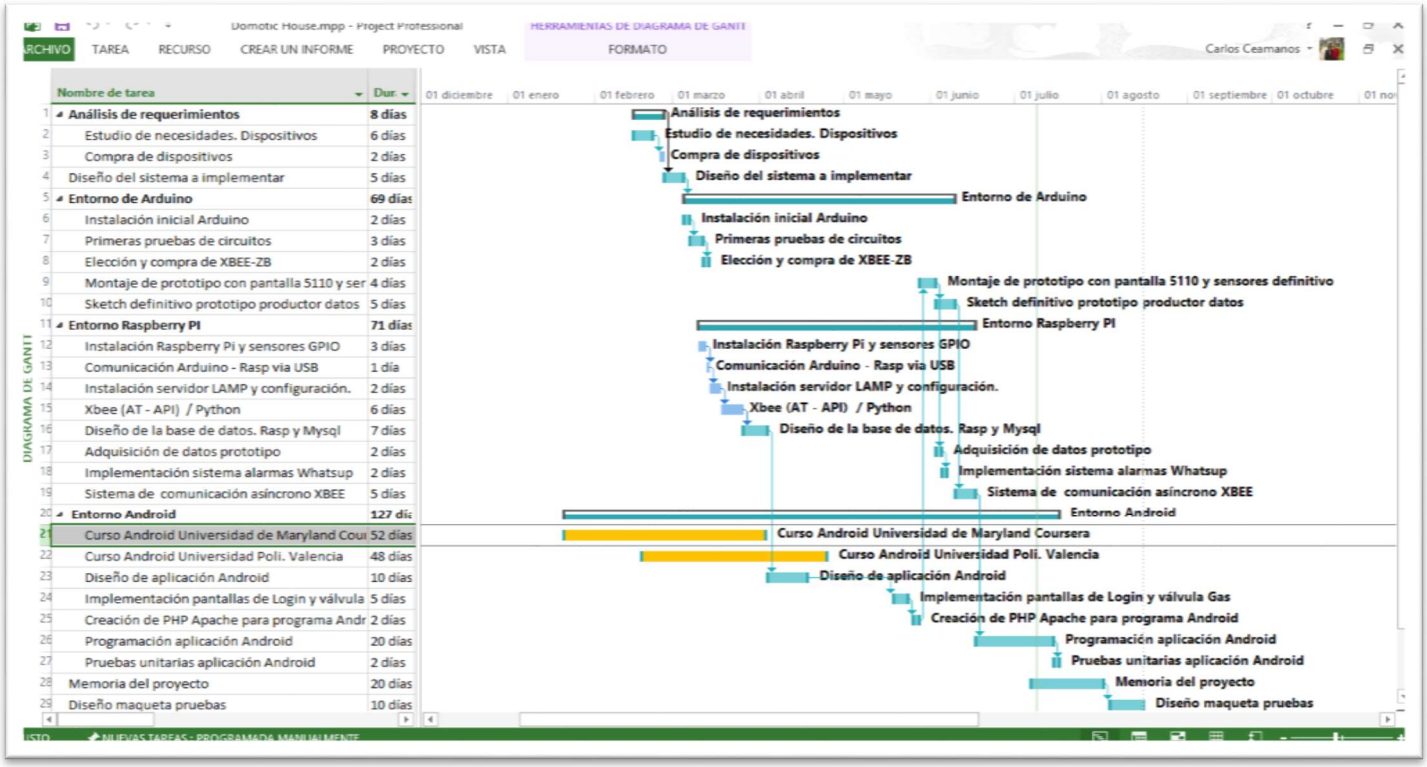
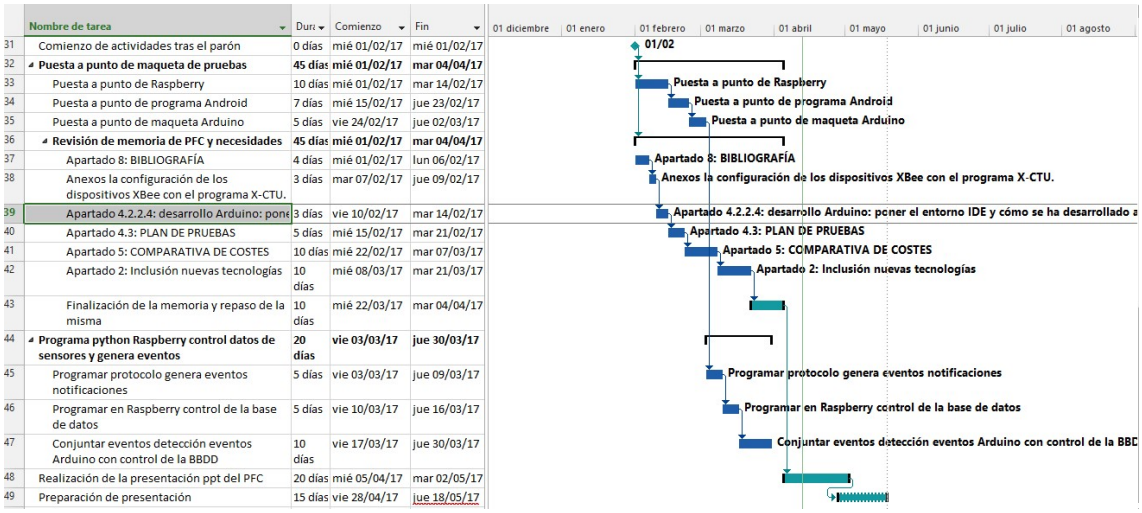


Figura 2. Diagrama de Gantt del proyecto 2014



	Nombre de tarea	Dura	Comienzo	Fin
31	Comienzo de actividades tras el parón	0 días	mié 01/02/17	mié 01/02/17
32	**Puesta a punto de maqueta de pruebas**	**45 días**	**mié 01/02/17**	**mar 04/04/17**
33	Puesta a punto de Raspberry	10 días	mié 01/02/17	mar 14/02/17
34	Puesta a punto de programa Android	7 días	mié 15/02/17	jue 23/02/17
35	Puesta a punto de maqueta Arduino	5 días	vie 24/02/17	jue 02/03/17
36	**Revisión de memoria de PFC y necesidades**	**45 días**	**mié 01/02/17**	**mar 04/04/17**
37	Apartado 8: BIBLIOGRAFÍA	4 días	mié 01/02/17	lun 06/02/17
38	Anexos la configuración de los dispositivos XBee con el programa X-CTU.	3 días	mar 07/02/17	jue 09/02/17
39	Apartado 4.2.2.4: desarrollo Arduino: pone	3 días	vie 10/02/17	mar 14/02/17
40	Apartado 4.3: PLAN DE PRUEBAS	5 días	mié 15/02/17	mar 21/02/17
41	Apartado 5: COMPARATIVA DE COSTES	10 días	mié 22/02/17	mar 07/03/17
42	Apartado 2: Inclusión nuevas tecnologías	10 días	mié 08/03/17	mar 21/03/17
43	Finalización de la memoria y repaso de la misma	10 días	mié 22/03/17	mar 04/04/17
44	**Programa python Raspberry control datos de sensores y genera eventos**	**20 días**	**vie 03/03/17**	**jue 30/03/17**
45	Programar protocolo genera eventos notificaciones	5 días	vie 03/03/17	jue 09/03/17
46	Programar en Raspberry control de la base de datos	5 días	vie 10/03/17	jue 16/03/17
47	Conjuntar eventos detección eventos Arduino con control de la BBDD	10 días	vie 17/03/17	jue 30/03/17
48	Realización de la presentación ppt del PFC	20 días	mié 05/04/17	mar 02/05/17
49	Preparación de presentación	15 días	vie 28/04/17	jue 18/05/17

Figura 3. Diagrama de Gantt del proyecto 2017

El proyecto tuvo una duración superior a la estimada, debido a que entre agosto de 2014 y febrero de 2017 hubo un parón en el mismo, por motivos laborales. Esto provocó que ciertos avances realizados durante el 2014, tuviesen que ser actualizados cuando se retomó el proyecto, y se tuviesen que repetir las pruebas de todo el sistema para comprobar que todo estaba vigente (la actualización del *kernel* de Linux por ejemplo fue uno de estos problemas, ya que algunos componentes tuvieron que ser reprogramados para que funcionasen con la nueva versión).

En la Figura 3 puede apreciarse el diagrama de Gantt con la planificación seguida una vez reanudado el proyecto para su entrega, en el año 2017.

2.- Tecnologías en la domótica actual

En este capítulo se pretende dar una visión básica de las tecnologías domóticas actuales, como marco de referencia al proyecto que se presenta.

2.1.- Estándares KNX y X10. *ZigBee*. Z-Wave. Webee y otros.

Existen varios sistemas domóticos comerciales, pero los fundamentales son KNX y X10, como sistemas propietarios (también tenemos LON). Los sistemas basados en *ZigBee* y *ZWave* y los dispositivos basados en estos estándares están aumentando su presencia en el mercado. Un ejemplo de estos sistemas es la plataforma Webee.

2.1.1. Estándar KNX

El estándar KNX o EIB KONNEX (European Installation Bus Konnex) es un sistema domótico descentralizado (no requiere de un controlador central de la instalación), en el que todos los dispositivos que se conectan al Bus del sistema poseen su propia electrónica de acceso al medio, por lo que normalmente se trata de un sistema con cableado dedicado.

Este sistema nació de la unión de varias empresas y de la necesidad de promover un sistema europeo de estas características. Se implicaron en él empresas como ABB, Tebis, etc.

Los componentes que posee un sistema KNX son módulos de alimentación de la red, acopladores de línea para interconectar diferentes segmentos de red y elementos actuadores. Este sistema tiene capacidad para más de 10.000 dispositivos interconectados.

Inicialmente se diseñó para realizar instalaciones de control industrial, y pasó más adelante a implantarse en edificios de oficinas y finalmente en viviendas.

Posee la ventaja de que con una única línea puede controlarse todo el sistema, y las futuras modificaciones o ampliaciones del sistema se realizan sin problema. Está orientado a la racionalización del uso de la energía, y el hecho de que esté estandarizado, hace que cualquier dispositivo fabricado para él, pueda integrarse en el sistema sin dificultad.

En la actualidad es el sistema domótico más demandado por su robustez, fiabilidad y modularidad.

Sin embargo, tiene varios inconvenientes:

- su precio: muy elevado. Esto ha ido cambiando en los últimos años, pero no tanto como para que se extienda al gran público. Sí que se ha mejorado sin embargo su módulo KNX/IP, permitiendo su conexión con cualquier dispositivo que comprenda el protocolo TCP/IP.

- requiere de una instalación dedicada, por lo que su montaje en hogares ya construidos, resulta muy invasiva, teniendo que realizarse grandes obras para ello, que incrementan aún más el precio final.

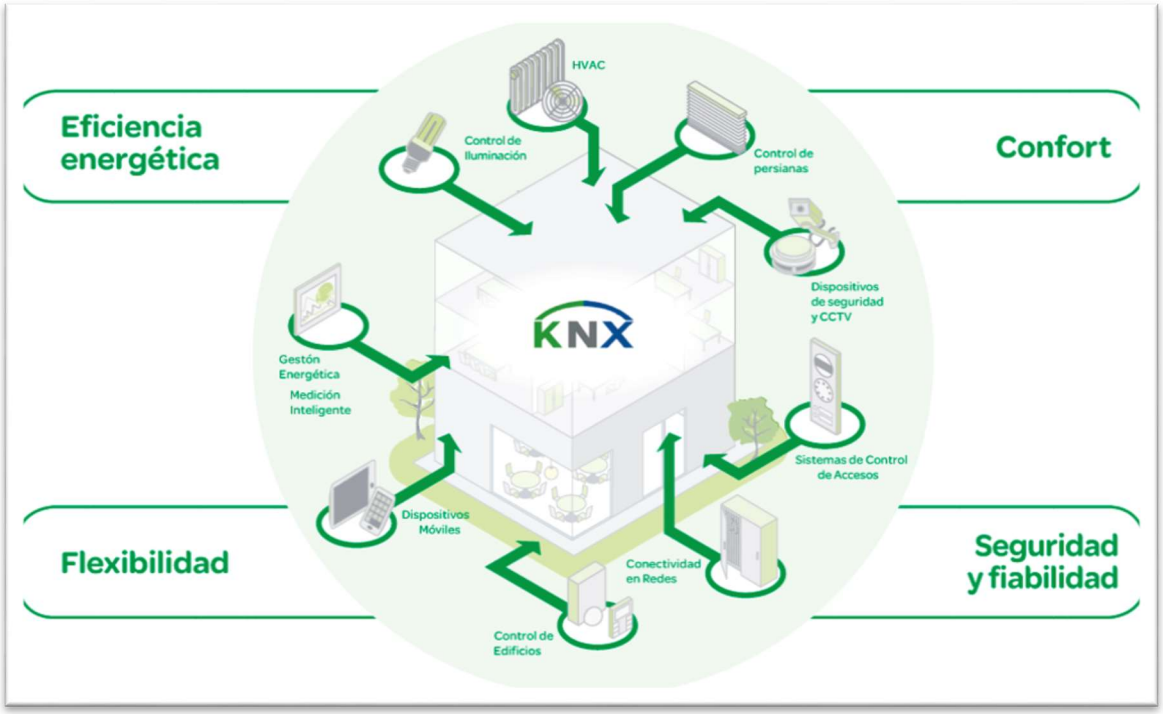


Figura 4. Sistema KNX

2.1.2. Estándar X10

El sistema X10 fue desarrollado en 1978 por Pico Electronics of Glenrothes, Escocia, para permitir el control remoto de los dispositivos domésticos. Fue la primera tecnología domótica en aparecer y continúa siendo la más ampliamente disponible, debido a su característica de autoinstalable, sin necesidad de cableado adicional, ya que las ordenes desde el sistema central hacia los dispositivos van a través de la corriente eléctrica, por la instalación de la casa (se trata de un sistema centralizado).

Su sencilla instalación y su reducido precio, lo hace un sistema muy extendido. Pero tiene ciertos inconvenientes, que con los dispositivos de última generación se están subsanando, como la bidireccionalidad de las comunicaciones: la mayoría de los componentes no permiten conocer el estado actual de un dispositivo, esto es, el sistema central no conoce si una luz está encendida o apagada.

Hay un sistema X10 que se comunica por radiofrecuencia, y permite a estos dispositivos inyectar tramas X10 en el sistema eléctrico.

Un sistema X10 puede tener tantos elementos direccionados como combinaciones puedan crearse con el grupo [A..P][0..16], 256 dispositivos, lo que da una amplio margen para cualquier hogar de control de dispositivos. Permite la emisión de órdenes grupales (p.ej. agrupamos todas las luces de una habitación y las apagamos todas a la vez).

Existen todo tipo de dispositivos X10 (las aplicaciones se pueden ver en la Figura 5):

- Transmisores: permiten inyectar la señal en la red eléctrica.
- Receptores: permiten recoger dicha señal y responder a la misma encendiéndose o apagándose.
- Bidireccionales: pueden actuar tanto de transmisor como de receptor. En caso de ser receptor, permitirían reportar su estado al control central.
- Inalámbrico: permiten emitir señales X10 por radio frecuencia, pero deben actuar sobre un trasmisor para traducir dicha señal (son mandos a distancia).

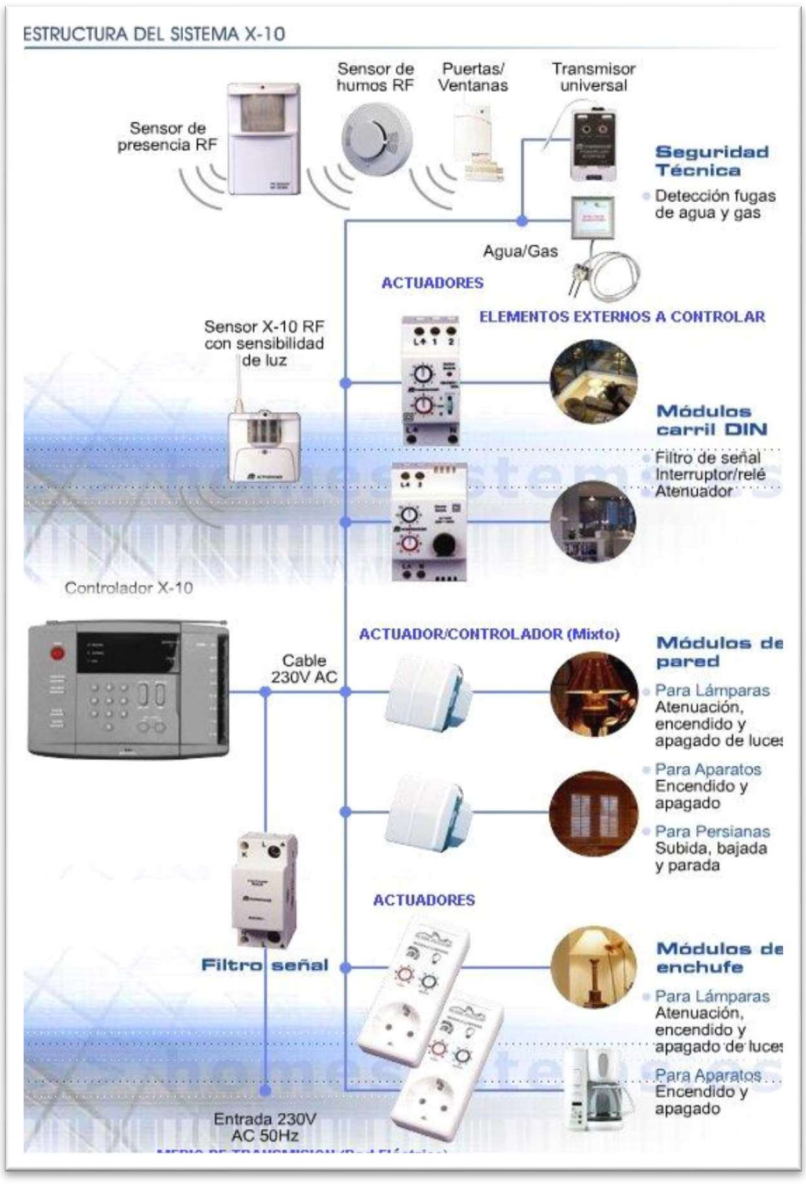


Figura 5: Sistema X10

2.1.3. *ZigBee*

La elección de este tipo de elementos para tener dispositivos inalámbricos en este proyecto no ha sido casualidad. *ZigBee* también conocido como "HomeRF Lite", es una

tecnología inalámbrica con velocidades comprendidas entre 20 kB/s y 250 kB/s y rangos de alcance de 10 m a 75 m.

Una red *ZigBee* puede estar formada por hasta 255 nodos los cuales tienen la mayor parte del tiempo el transceptor *ZigBee* dormido con objeto de consumir la mínima energía posible. El objetivo, es que un sensor equipado con un transceptor *ZigBee* pueda ser alimentado con dos pilas AA durante al menos 6 meses y hasta 2 años. Los dispositivos *ZigBee* poseen la capacidad de caer al estado *sleep* y pueden ser "despertados" de varios modos diferentes en el momento necesario. Esta autonomía los hace ser muy adecuados para la domótica y para la medición de sensores.

Otra de las características que poseen estos dispositivos es que permiten enviar la señal del voltaje actual del dispositivo, con lo que en caso de necesitar un cambio de batería, el mismo dispositivo informa al servidor central de ello.

Se espera, que los módulos *ZigBee* sean los transmisores inalámbricos más baratos jamás producidos de forma masiva, con un coste estimado alrededor de los 2 euros. Dispondrán de una antena integrada, control de frecuencia y una pequeña batería. *ZigBee* ofrece una solución tan económica porque la radio se puede fabricar con muchos menos circuitos analógicos de los que se necesitan habitualmente.

Podemos ver una muestra de los diferentes dispositivos en la web de *ZigBee* [1].

Existen también en el mercado muchas soluciones basadas en los módulos *ZigBee*, tales como bombillas controladas por bluetooth o a través del wi-fi mediante un Smat-host. Este host, se comunica mediante *ZigBee* con los dispositivos que tiene registrados, lo que evita las interferencias con la gran cantidad de sistemas de comunicación que existen en el hogar. Philips y Belkin son compañías que han ofrecido en los últimos años dispositivos de estas características.

En la Figura 6 puede observarse un esquema de control domótico con esta tecnología: el usuario desde su teléfono remite una señal a la nube, señal que se transmite al *gateway* a través de la red. El *gateway*, mediante el protocolo *ZigBee*, remite las órdenes pertinentes a los dispositivos que tiene conectados (on, off, dime, etc.).

En el apartado 3.4 se comentarán más a fondo las características de estos dispositivos, y en especial, se describirá el protocolo de seguridad que implementan, el AES-128 encryption, al mismo nivel de protección que el de algunos bancos, y las claves de red, para mantener seguros los nodos de la misma.

ZigBee fue desarrollado inicialmente para su uso en empresas de servicio público y de venta al por menor. Es muy popular también en el ámbito del uso doméstico, aunque requiere que los consumidores posean conocimientos técnicos para su puesta en marcha y mantenimiento.

Es perfecto para los DIYers (*Do It Yourself*) o para expertos en tecnología que quieren un sistema que puedan ajustar a sus preferencias e instalar ellos mismos.

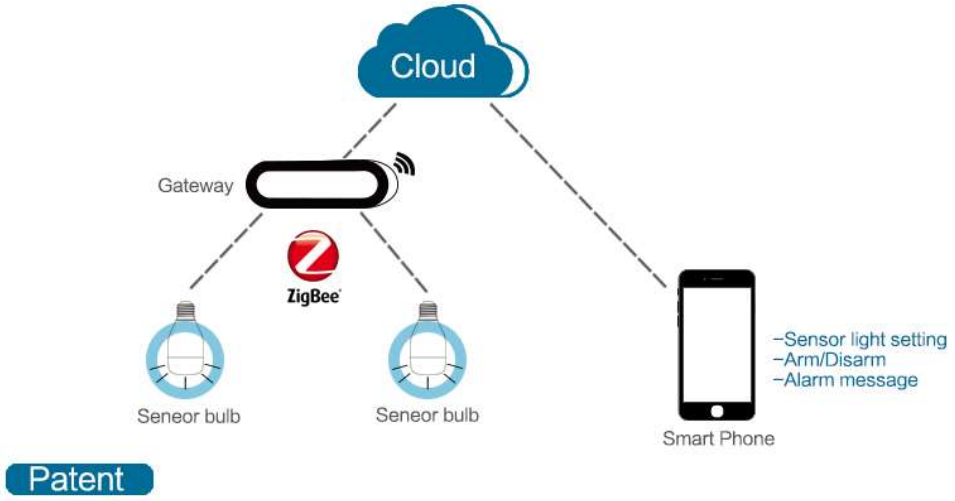


Figura 6: Esquema control bombillas con ZigBee

2.1.4. Z-Wave

Z-Wave es una de las tecnologías inalámbricas de red originales para la automatización del hogar. Su tecnología propietaria fue desarrollada específicamente para que los clientes monitorizaran y controlasen remotamente sus dispositivos de automatización del hogar como alumbrado, climatización, seguridad, etc.

Z-wave se basa en un servidor *hub* que actúa como controlador de la red doméstica, permitiendo que 232 dispositivos se conecten, comunicándose entre sí. Cuántos más dispositivos tienes conectados, mejor señal se obtiene. El sistema es similar al de las redes *mesh* de *ZigBee*, por lo que la señal de comunicación entre dos elementos de la red se puede retransmitir por cada uno de los otros elementos como si fueran repetidores.

The Z-Wave Alliance está formada por 375 compañías y unos 1500 productos que son interoperables entre sí. Nueve de cada diez compañías punteras en seguridad y comunicaciones de USA utilizan Z-Wave en sus soluciones de automatización del hogar. La tecnología Z-Wave tiene 325 fabricantes, lo cual proporciona una amplia gama de dispositivos que pueden comunicarse entre sí.

Acerca de la seguridad de los dispositivos, a cada uno se le asigna un ID único para comunicarse con el *hub*. Esto hace que ningún otro *hub* pueda controlar los dispositivos de otro. Para incrementar la seguridad, al igual que *ZigBee*, se utiliza además el protocolo de encriptación AES-128.

Sus ventajas son que provee un sistema sencillo para que los clientes puedan ponerlo en marcha ellos mismos y que una vez que se tiene un Z-Wave *hub* se le pueden conectar tantos dispositivos como se quiera.

Por el contrario, los dispositivos Z-Wave son caros, y van desde el rango de los 40 a 100 euros cada uno.

2.1.5. Webee

Existen también soluciones integradas, tales como la de la compañía Webee, en la web [2] (Figura 7).

Esta compañía fue fundada a través de crowdfunding. Ha desarrollado un sistema domótico basado en tecnología Wi-fi, *Zigbee* y *ZWave*, ofertando una solución adaptable a las necesidades domóticas de cualquier hogar (enchufes inteligentes controlados por el usuario, control de cámaras, de apertura de puertas, seguridad, etc.).

Figura 7: Sistemas Webee

Además, ha abierto su sistema a los desarrolladores para integrarlo en plataformas de *hardware* libre o de cualquier tipo (programas Android e iOS de control). Su solución básica tiene un precio muy asequible y una gran facilidad de instalación en cualquier hogar, con una modularidad impresionante. Ideas en esta dirección con un marketing adecuado son las que podrían terminar imponiéndose en la batalla que por la domótica se va a librar en los próximos años, y hacia las que se ha orientado este trabajo.

2.2.- Trabajos anteriores

Para este proyecto se han analizado varios trabajos y publicaciones anteriores del IEEE Transactions on Consumer Electronics. Podemos señalar los siguientes:

1. *"A ZigBee wireless domotic system with Bluetooth interface" [1]*, donde se utiliza una interfaz bluetooth con una consola central y comandos KNX. Aquí describe cada uno de los dispositivos como si se tratase de lenguaje de SNMP, de una MIB, interesante desde el punto de vista del autodescubrimiento de redes. Carece de la utilización de Android como sistema de control.
2. *"Design and implementation of smart home energy management systems based on zigbee" [2]*, donde se implementa un nuevo protocolo de routing en la red ZigBee,

con dos líneas de flujo de información y orientado a optimizar la energía utilizada en el hogar.

3. *"Smart home energy management system including renewable energy based on ZigBee and PLC" [3]*, que utiliza dispositivos ZigBee de medición de consumo de energía para optimizar el consumo de una casa. Implementa programación de escenarios dependiendo del precio de la energía. Se utiliza Zigbee además para la activación de los dispositivos.
4. *"Towards an Open Framework for Home Automation Development" [4],* donde se utiliza BLE (bluetooth) e IR para interactuar con los dispositivos finales. Implementa escenarios a partir de los que actúa. Prototipo muy similar a la solución de este PFC, pero sin Arduinos ni ZigBee. Se diseña *ad-hoc* una placa microcontroladora para comunicar entre la red IP y Bluetooth. El uso de esta placa hace que deban formatear todos los datos que reciben de los sensores para que se ajusten y puedan utilizarse (Arduino evita este paso).

Finalmente, haremos referencia aquí a un trabajo publicado acerca de la temática de éste proyecto, en el IEEE, "*A ZigBee-Based Home Automation System*" *[5]* .

En este artículo se analiza el motivo de la lenta penetración de los sistemas domóticos en nuestros hogares, y el potencial de los *ZigBee* como tecnología para implementar sistemas domóticos, exponiendo la arquitectura para un sistema de estas características apoyándose en *ZigBee.*

Se puede ver propuesta de los autores de este trabajo en la Figura 8, de similar diseño a la propuesta en este proyecto respecto a la arquitectura y a la tecnología utilizada para comunicar con los dispositivos (*ZigBee*), pero con diferencias notables.

Desde el PC, se generan eventos propios del sistema. El Home Gateway es una pasarela que comunica la red Wi-Fi con la red *ZigBee*, "traduciendo" el mensaje de una red a otra, pero sin procesar el mismo (se podría decir que es un intérprete).

Por otro lado, el Virtual Home es una capa que se ocupa de segurizar el sistema, autenticando los mensajes que le llegan del Home Gateway, y trasmitiendo al dispositivo adecuado la orden contenida en el evento codificado para *ZigBee* transmitido desde el PC. Descifra el mensaje que le llega del Gateway. Autentifica el mismo, verificando que se trata de un usuario autorizado; extrae el dispositivo al que va dirigido el evento, el mensaje a transmitir y los parámetros del mismo. Finalmente, cifra toda la información para la red *ZigBee* y transmite la orden a la red.

La red *mesh*, tiene un elemento central o coordinador que almacena el estado de los dispositivos de la misma y sus direcciones, realizando la función de distribuir las órdenes o eventos hacia los mismos (por ejemplo, encender o apagar una bombilla).

Hay diferencias notables con el caso de este PFC: el *Gateway* sería una Raspberry Pi con un servidor LAMP (Linux-Apache-MySql-PHP) que procesa todos los mensajes que llegan vía internet al servidor; seguriza y autentifica la orden llegada, verifica el estado del dispositivo al que va dirigido el evento, genera la orden correspondiente para comunicarla por la red *ZigBee*, y una vez ejecutada interpreta la orden y almacena el nuevo estado del dispositivo en la base de datos MySql. Este *Gateway* tiene capacidad de procesamiento y configuración.

Se añade además una nueva capa de procesamiento al sistema, introduciendo los controladores Arduino tras el conector *ZigBee*, lo que permite que se procesen los datos necesarios a enviar por la red antes de remitirlos al *Gateway*.

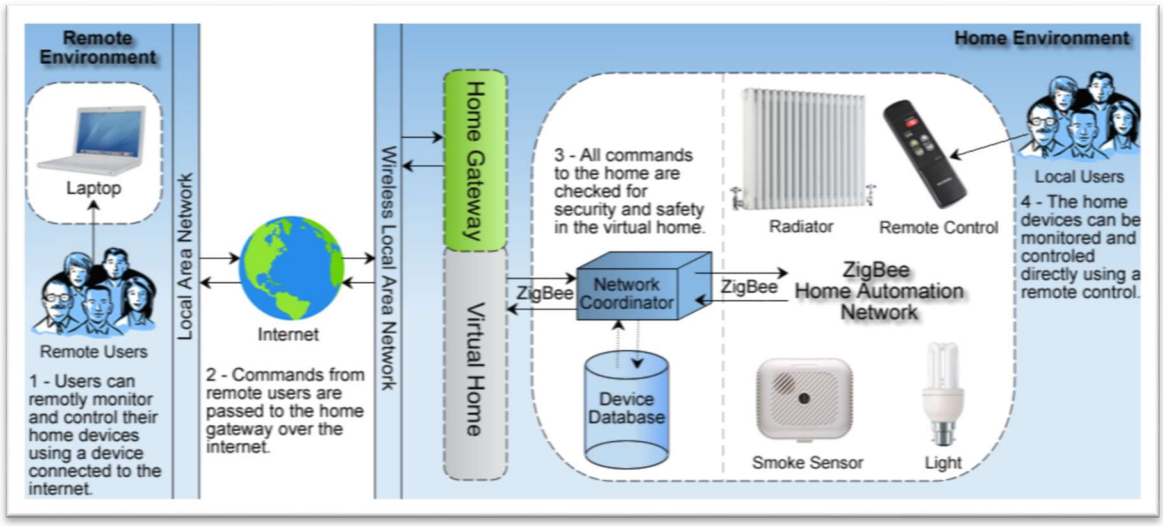


Figura 8: Arquitectura propuesta

El sistema propuesto en el artículo carece de la capacidad de procesamiento, por lo que es un simple intérprete de órdenes dirigidas a los dispositivos (a diferencia de lo propuesta en este PFC). No podrían configurarse en el sistema escenarios domésticos y la respuesta a los mismos (por ejemplo: en caso de que la temperatura baje 20 grados y se haya anochecido, enciende la calefacción, ilumina salón y dormitorios y cierra las persianas). Tanto el Gateway como el Virtual Home desconocen el estado de los dispositivos y no pueden realizar una interpretación del estado global del sistema para generar órdenes adicionales al evento recibido por parte del usuario.

En la conclusión del artículo se señala que existen varios puntos que impiden el desarrollo de la domótica: nivel de invasión de las instalaciones, complejidad y precio de las arquitecturas de los diferentes sistemas. La escasa compatibilidad entre los diferentes sistemas existentes sería otro.

En contraposición a estos problemas, se propone una arquitectura basada en comunicaciones *ZigBee*. Con ello se consigue el abaratamiento del coste del sistema, y la sencillez de instalación. Permite interactuar con el sistema mediante Internet, a través del *Gateway* lo que facilita la inter-operatividad entre la red Wi-Fi y la red *ZigBee*.

3.- Plataformas y tecnologías

En este apartado de la memoria se presentan las diferentes plataformas y tecnologías en las que se ha basado el diseño del prototipo de sistema domótico. Se realizará una breve introducción de las mismas, ya que no es el objetivo de esta memoria detallarlas en profundidad.

3.1.- Plataforma Android

En el proyecto, Android es una parte fundamental, ya que es el punto de acceso que el usuario tendrá a todo el sistema domótico. Al usuario se le presenta un interfaz con todas las posibles acciones a ejecutar y de configuración desde la pantalla de su dispositivo móvil. Se ha elegido la plataforma Android por tratarse de un sistema abierto, con facilidad de acceso a las herramientas necesarias para su desarrollo, al contrario de lo que ocurre con el sistema operativo iOS de Apple, de carácter más cerrado.

Una manera de explicar qué es Android, sería haciendo referencia a las palabras del ingeniero de Google Andy Rubin:

> *"La primera plataforma comprensiva para dispositivos móviles. Incluye un sistema operativo, interfaz de usuario y aplicaciones – todo el software necesario para hacer funcionar un dispositivo móvil pero sin los problemas de patentes que dificultan la innovación."*

Android forma parte de la **Open Handset Alliance** que es la unión de más de 80 empresas tecnológicas incluidas compañías de *hardware*, compañías de teléfono, desarrolladores de software como Samsung, Motorola, HTC, T-Mobile, Vodafone, ARM, y Qualcomm.

Android era un sistema operativo para dispositivos móviles que casi no se conocía hasta que en 2005 Google lo compró. En 2007 se lanzó la Open Handset Alliance y se proporcionó la primera versión de Android junto el SDK para que los programadores comenzaran a crear aplicaciones para este sistema.

Aunque los inicios fuesen lentos debido a que se lanzó antes el sistema operativo que el primer móvil Android, rápidamente se ha convertido en el sistema más extendido. En la actualidad, la versión más moderna es la 7.0 – 7.1 Nougat. Puede verse en la Figura 9 la cuota de mercado de la distribución de las versiones existentes de Android. Como se puede apreciar, se trata de un mercado muy fragmentado, y este es uno de los problemas a los que se enfrenta Google, y que Apple con iOS no tiene.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.2%
4.1.x	Jelly Bean	16	4.5%
4.2.x		17	6.4%
4.3		18	1.9%
4.4	KitKat	19	24.0%
5.0	Lollipop	21	10.8%
5.1		22	23.2%
6.0	Marshmallow	23	26.3%
7.0	Nougat	24	0.4%

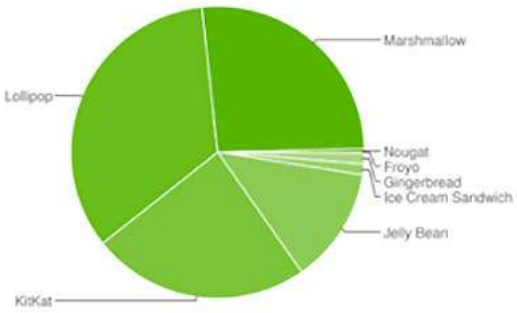


Figura 9: Versiones Android y cuota de distribución

Los dispositivos basados en Android suponen una cuota de mercado mundial del 86% y del 92% en España, lo que apunta a que este será el sistema operativo del futuro, aunque en el mundo de la tecnología, los cambios son muy rápidos e impredecibles.

En la Figura 10 se puede ver la arquitectura del sistema operativo Android.

Las características principales de este sistema son:

1. Plataforma realmente abierta.
2. Adaptabilidad a cualquier tipo de hardware.
3. Arquitectura basada en Internet (la interfaz de usuario está basada en Xml).
4. Tiene un aceptable nivel de seguridad (cada programa se ejecuta en una caja que hereda de Linux y el usuario debe facilitar a cada aplicación los permisos que requiera del terminal en caso de requerirlos).
5. Optimizado para baja potencia y pocos recursos de memoria y procesador.

Para desarrollar la aplicación en este PFC, se han utilizado los siguientes recursos:

1. Java Runtime Environment 6.0.
2. Eclipse (Eclipse IDE for Java Developers).
3. Android SDK (Google).
4. Eclipse Plug-in (Android Development Tools - ADT).

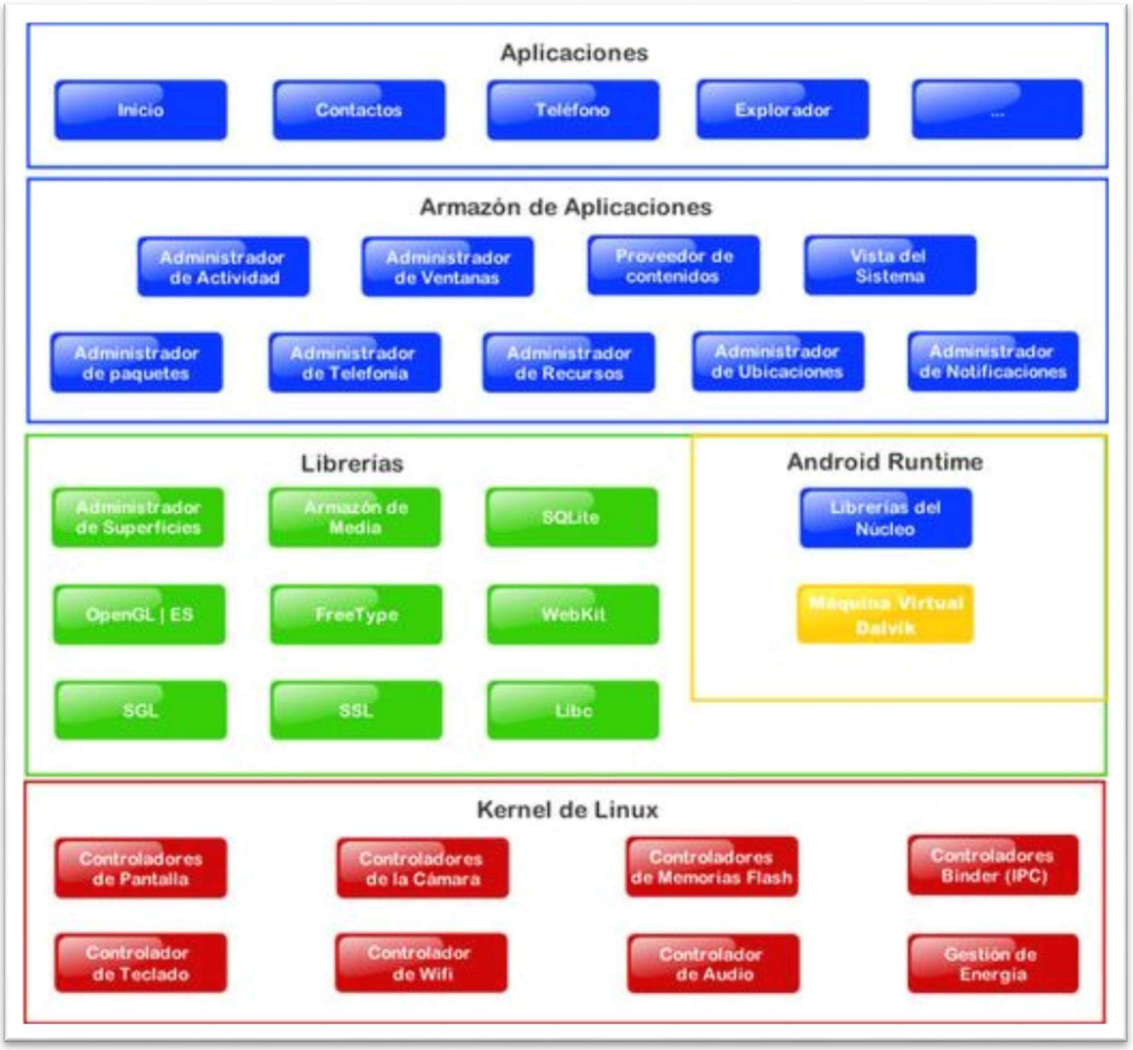


Figura 10: Arquitectura Android

3.2.- Raspberry Pi

Raspberry Pi es un microordenador de bajo coste desarrollado en Reino Unido; fue creado con la idea de ofertar un sistema barato y asequible para todo el mundo y facilitar la entrada al mundo del desarrollo de *software* a muchas personas que debido a los costes de la tecnología quedaban excluidas del mismo. El lanzamiento comercial del primer modelo se produjo en febrero de 2012.

Este dispositivo ha tenido una gran acogida en toda la comunidad educativa ya que uno de sus propósitos y objetivos fundamentales ha sido el de constituirse en un sistema de aprendizaje, orientado sobre todo a países en vías de desarrollo, pero la potencia del mismo ha hecho que hayan trascendido sus funciones y se haya convertido en base de proyectos tecnológicos.

Para la primera versión se utilizó un chip integrado Broadcom BCM2835, que contiene un procesador ARM11 con varias frecuencias de funcionamiento y la posibilidad de *overclocking* hasta 1 GHz además de un procesador gráfico VideoCore IV.

Raspberry Pi puede tener como SO un Linux para sistemas ARM. Este pequeño dispositivo viene equipado con unas altas prestaciones para su tamaño (8 cm. x 5 cm.), como son puerto Ethernet, salida de video HDMI, 2 puertos USB, slot para tarjetas SD y en las versiones más recientes, adaptador Wi-Fi y hasta 1 GB de RAM.

Además, posee una serie de entradas denominadas GPIO (*General Purpose Input Output*). Estas entradas/pines, permiten conectar el sistema con el mundo real a través de sensores analógicos, motores o actuadores con lo que permite que Raspberry obtenga datos del medio físico o actúe sobre él. Estos pines son tipos diferentes: I2C (*Inter Integrated Circuit*), UART (*Universal Asyncronous Receiver-Transmitter*) o PWM (*Pulse With Modulation*), lo que amplía el espectro de las posibilidades ofrecidas por este mini ordenador. En el anexo de esta memoria puede obtenerse más información acerca de este dispositivo.

En la Figura 11 se puede apreciar el tamaño del dispositivo utilizado para este PFC, que es el modelo B.

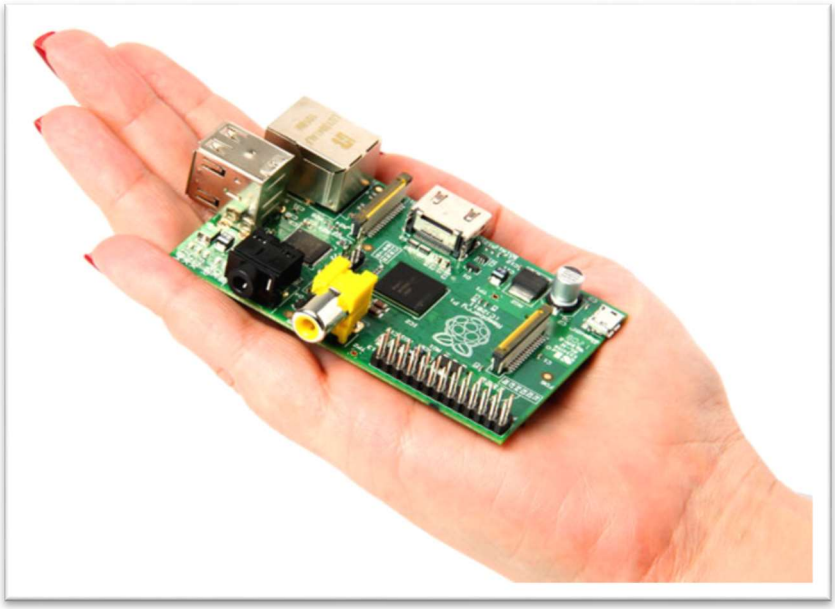


Figura 11: Raspberry Pi Modelo B

Aunque pueda parecer un ordenador normal, uno de los inconvenientes que tiene este dispositivo es que no posee disco duro. Puede ser una ventaja, ya que teniendo varias tarjetas SD se pueden tener varios ordenadores en uno sólo, cambiando la tarjeta y reiniciándolo, pero sin embargo, y por experiencia a lo largo de este proyecto, es un inconveniente muy alto, ya que debido a la gran cantidad de accesos realizados a la tarjeta SD, ocasiona que esta se deteriore, con la pérdida consiguiente de información si no se ha realizado una copia de seguridad adecuada. Recientemente ha salido un modelo al mercado, la Raspberry Compute Module 3, diez veces más potente que la Raspberry 3, y que lleva integrado un módulo de 4 GB de RAM, orientado a aplicaciones comerciales electrónicas.

En la Figura 12 se pueden observar todas las entradas y salidas del dispositivo Raspberry Pi Modelo B.

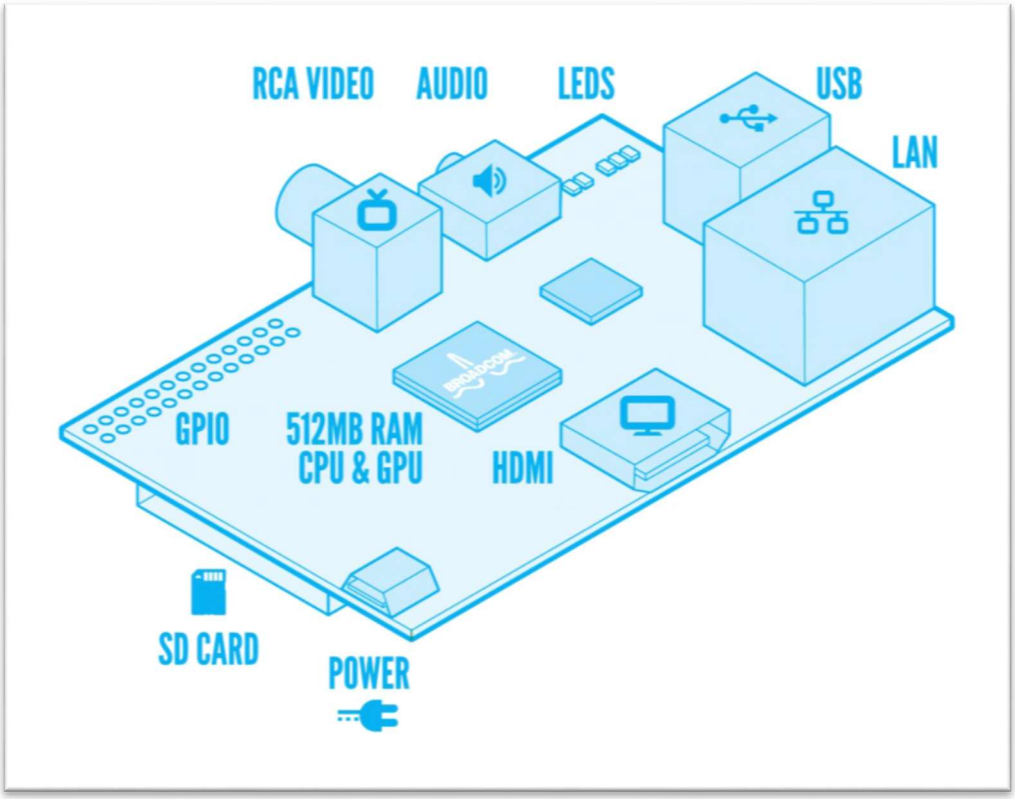


Figura 12: Esquema Raspberry Pi

3.3.- Arduino: sensores y dispositivos

Arduino es un dispositivo de *hardware* libre creado con la finalidad de poder realizar proyectos de electrónica de manera sencilla. Arduino se inició en el año 2005 como un proyecto para estudiantes de arte en el Instituto IVREA, en Ivrea (Italia). Pero poco a poco se fue transformando en un dispositivo al que investigadores e ingenieros han visto el potencial, y han surgido aplicaciones basadas en Arduino de muy diversa índole.

Existen muchos modelos de este dispositivo, con diferentes características. Recientemente incluso se ha creado un modelo denominado Yun LininoOs que admite una distribución de Linux y que lleva un SDK de Google para interaccionar con Android. Sin embargo, para este PFC se seleccionaron los modelos más baratos y básicos.

Inicialmente se compró una placa Arduino UNO R3, y más adelante una Arduino Mega 2560 R3 de muy similares características ambas, con la diferencia de que la Mega posee más memoria (de 32K a 256K de la Mega) y más pines IN/OUT (de 14 a 54). Puede obtenerse más información acerca de las especificaciones de la Arduino UNO R3 en: Arduino UNO [3] y de la Arduino Mega 2560 [4] (recientemente ambos modelos han sido renombrados como “Genuino” para los modelos comercializados fuera de USA).

Ambos dispositivos están basados en microcontrolador, en el ATmega328 la UNO R3 y en el ATmega2560 la Mega R3. Pueden verse los diferentes modelos de placas ofertados en la web de productos Arduino [5].

Este microcontrolador y la estructura que posee la placa, permiten a Arduino tomar toda la información de su entorno a través de una amplia gama de sensores que han sido adaptados a los protocolos de adquisición de esta placa. También pueden controlar actuadores (relés) y motores.

El microcontrolador de la placa es programable mediante un lenguaje propietario, basado en *Wiring*, muy similar a C++. Posee además un entorno de programación basado en *Processing* que permite con facilidad manejar el dispositivo.

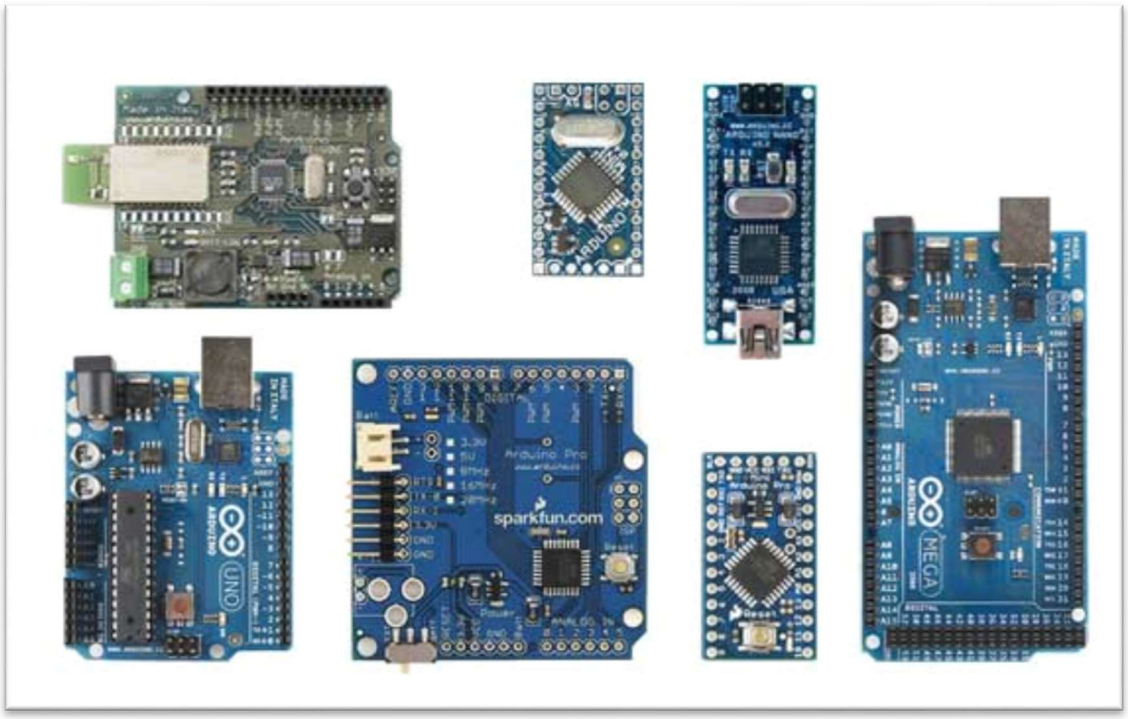


Figura 13 : Tipos de placas Arduino

Para facilitar el ensamblado de estas placas con otros dispositivos, Arduino ofrece una gran gama de "*shields*", que no son otra cosa que placas que se montan sobre el mismo Arduino para facilitar la interfaz con alguna clase de dispositivo. Existen *shields* para Ethernet (ofrecen conexión Ethernet a las placas que no lo llevan de serie, como las utilizadas en este proyecto, aunque hay un Arduino que la trae de serie, el Arduino Ethernet), para GSM (puede montarse una SIM de telefonía móvil), para Wifi (proporcionan conectividad Wifi), para puertos USB, para XBee (ofrece la posibilidad de montar un chip XBee de cualquier modelo con Arduino). Este último *shield*, fabricado por la empresa aragonesa Libelium, se ha utilizado en este proyecto para facultar a las placas Arduino utilizadas de conexión *XBee-Zigbee*. Puede verse este *shield* en la Figura 14: Shield XBee Arduino.

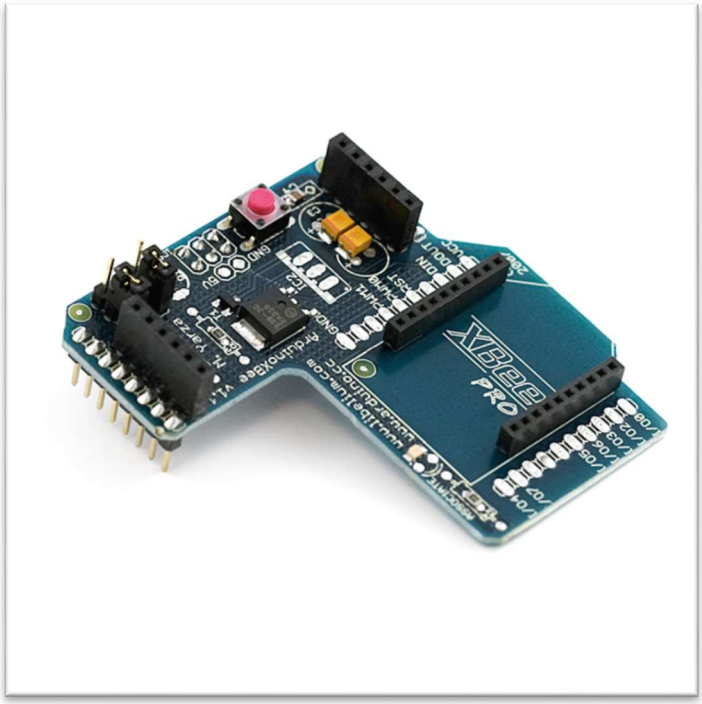


Figura 14: Shield XBee Arduino

El motivo fundamental de la elección de las placas Arduino UNO R3 para nuestro diseño fue su bajo precio, en torno a los 14 euros.

El papel fundamental de las placas Arduino en el proyecto es la actual como recolectores de información, esto es, detectar la información de los sensores que se les acoplan.

En los siguientes apartados detallamos los sensores que se han utilizado para este proyecto.

3.3.1.- Sensores de temperatura y humedad DHT11 y DHT21

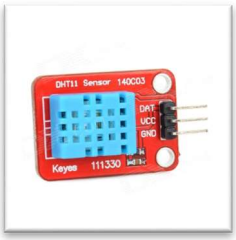


Figura 15: DHT-11

Estos dispositivos se caracterizan por tener una señal digital calibrada, lo que asegura una alta calidad y fiabilidad a lo largo del tiempo; poseen un microcontrolador de 8 bits integrado. Están integrados por dos sensores resistivos que les permiten ofrecer mediciones de temperatura (NTC - Negative Temperature Coefficient) y de humedad. Poseen una excelente calidad y una rápida respuesta a las medidas. Las diferencias entre uno y otro es el rango de precisión, siendo de mejor calidad el DHT21. Para este proyecto se han utilizado ambos, realizando pruebas con los dos tipos de dispositivos. También se ha utilizado el sensor analógico de temperatura TMP36, realizándose una prueba de mediante los GPIO de la Raspberry y otra constituyendo un sensor final sin Arduino y con emisor XBee (dispositivo durmiente que únicamente despierta para emitir la información del sensor cada cierto tiempo). En este proyecto, por condiciones de diseño, se decidió que Raspberry no tuviese sensores, y se descartó su utilización. Además, dado que en todos los dispositivos finales había más de dos

sensores, todos van montados sobre una placa Arduino por la facilidad de procesamiento de la información que este ofrece.

Para más información de las características:

- DTH11 [6]
- DHT21 [7]

3.3.2.- Sensor de gases MQ-2

El módulo sensor de gas analógico MQ2 (Figura 16) se utiliza en la detección de fugas de gas de equipos en los mercados de consumo y la industria; este sensor es adecuado para la detección de GLP, butano, propano, metano, alcohol e hidrógeno y posee una alta sensibilidad y un tiempo de respuesta rápido. Mide la concentración de estos gases en partes por millón. Su sensibilidad puede ser ajustada por el potenciómetro que lleva el mismo módulo. En sus características tiene una curva de respuesta para cada uno de los gases que detecta, dependiente de la temperatura y la humedad.

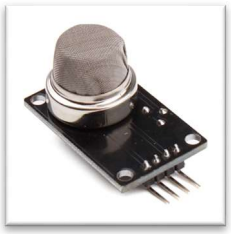


Figura 16: MQ-2

Para más información de sus características: MQ-2 [8]

3.3.3.- Sensor de presencia PIR HC-SR501

El módulo sensor de presencia PIR (*Proximity Infrared*), es un módulo con sensor digital infrarrojo (Figura 17), que detecta presencia por el cambio en el espectro de temperatura. Tiene posibilidad de ajuste del tiempo en alto de la señal de salida entre 5 segs. y 18 min. (lo que permite utilizarlo para el encendido y apagado luces, por poderse programar el tiempo de encendido)

Figura 17: HC-SR501

Su alcance de detección va de 3 a 7 metros (lo que lo hace muy adecuado para aplicaciones dentro del hogar).

Para más información: HC-SR501 [9]

3.3.4.- Sensor de luminosidad GY-30

El módulo de sensor de luminosidad GY-30 (Figura 18) utiliza el sensor BH1750FVI para medir la luminancia en luxes. Se utiliza como aplicación principal para medidas de luz en fotografía, pero se puede adaptar a las necesidades de la domótica para que con los datos que ofrece se puedan crear eventos programables sobre las luces, persianas, toldos, etc. de una casa. A estos se les denominan “escenarios”, disparo de eventos sobre actuadores en función de ciertos parámetros configurados previamente por el usuario.

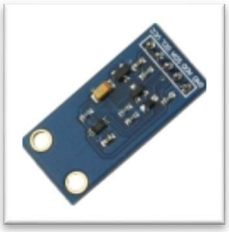


Figura 18: GY-30

El módulo posee un conversor analógico digital (ya que su base es una LDR-*Light Dependent Resistor*), y tiene la peculiaridad de que los datos se transmiten a Arduino por el protocolo I2C (*Inter Integrated Circuit*), protocolo digital muy utilizado en la industria. Este protocolo utiliza dos líneas para transmitir los

datos, una para los datos propiamente dichos (SDA) y otra para la señal de reloj (SCL). En Arduino, estos pines son los I/O analógicos A4 y A5 respectivamente (UNO R3).

Para más información: GY-30 (BH1750FVI) [10]

3.3.5.- Sensor de nivel de agua

Se trata de un sensor analógico (Figura 19) que permite detectar si el nivel del agua sube. Se utiliza para detectar si hay peligro de inundación en el hogar, aunque otro de sus usos determinar el nivel de humedad en tierra para plantas y jardines; esta detección la realiza a través de unas líneas de conducción paralelas al sensor, lo que hace que ante la presencia de humedad o agua, varíe la señal que devuelve. Debido a sus características y función, hace que este sensor tenga una vida útil de un año en uso intensivo. Su precio no llega a los 3 euros.

Figura 19: Sensor Agua Funduino

Para más información: Funduino [11]

3.3.5.- Otros dispositivos utilizados

Para este proyecto se han utilizado otros dispositivos conectados a Arduino, como son los relés. Se trata de unos módulos actuadores diseñados especialmente para Arduino, que llevan implementada la configuración necesaria de un relé (relevador), que es un dispositivo electromecánico, similar a un interruptor. Normalmente, para configurar un relé es necesario añadir al circuito un transistor y un diodo, de forma que nuestra placa no se fría con el voltaje que entra en el relé. Los módulos de relé Arduino ya vienen con toda la configuración. En particular para la maqueta se ha acoplado un módulo de 8 relés como el de la Figura 21, conectado a un Arduino UNO R3.

Figura 20: Sensor Hall

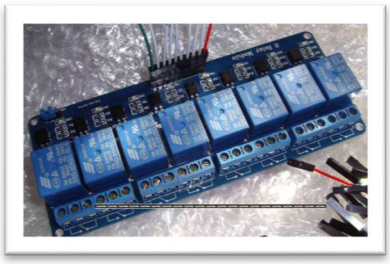


Figura 21: Placa 8 relés

Con esta placa de relés podemos dar órdenes en nuestro sistema a 8 dispositivos (ya sean válvulas, persianas, luces, etc.). El único problema que plantea este sistema es la situación en que se encuentra el dispositivo en caso de tener en el hogar también interruptores manuales conectados al sistema, lo que sería deseable. Para subsanar este problema se probó con un sensor de efecto Hall, que permite conocer si por un cable circula corriente o no, teniendo así la información de estado del dispositivo, y que podría conectarse a la placa Arduino también, como el que se puede ver en la Figura 20.

Para más información:

- Placa de 8 relés [12]
- Sensor Hall [13]

3.4.- *Zigbee* y *XBee*

ZigBee es un estándar de comunicaciones abierto para baja potencia, basado en la especificación *802.15.420* de redes inalámbricas WPAN21. Este protocolo de red posee servicios de seguridad y capa de aplicación que garantiza la comunicación entre los diferentes nodos con los que se conforme la red. Lo más destacable de este tipo de dispositivos es que permite la construcción de redes *mesh* (redes inalámbricas malladas, con comunicación entre todos sus miembros a través de una jerarquía establecida inicialmente en la configuración de los elementos, y con sofisticadas tablas de encaminamiento que proporcionan alta capacidad de adaptación en caso de caída de un nodo).

El protocolo que utiliza (*ZigBee*) trabaja a 250 KBps. sobre la banda de 2,4 GHz. Está publicado y administrado por un grupo de empresas denominado "*ZigBee Alliance*" por lo que el término *ZigBee* es una marca registrada y no un estándar técnico, aunque para fines no comerciales, la especificación *ZigBee* es abierta y libre al público.

Las principales características de *Zigbee* son su bajo coste (estándar abierto), bajo consumo de energía (posibilidad de dormir los dispositivos), alta confiabilidad, alta seguridad (hace uso para su comunicación del algoritmo de cifrado AES-128 sobre datos y sobre la autentificación), baja latencia (todas las operaciones de red son muy rápidas) y la posibilidad de montar más de 65K dispositivos en la misma red.

Dentro de una red mesh *ZigBee* se pueden configurar tres tipos de nodos:

1. ***Coordinador*:** se trata del nodo central de la red, el que la crea y el que detecta a los nuevos componentes que se van añadiendo, a los que asigna su dirección de red.
2. ***Router*:** se une a redes existentes y actúa de enlace entre un nodo final y el nodo coordinador. Posee capacidad de enrutamiento. Puede almacenar información y mensajes dirigidos hacia un nodo final hasta que este despierte.
3. ***Final*:** nodo básico para unirse a la red. En muchas ocasiones está formado por un *chip XBee* y los sensores de los que se pretende obtener la información. Este tipo de nodo puede configurarse para apagarse (*sleep mode*), reduciendo su consumo de energía al mínimo necesario. Para conectarse a la red necesita tener asignado un *router* o el coordinador.

En la Figura 22 puede apreciarse el modelo que se utilizaría para una red *mesh*.

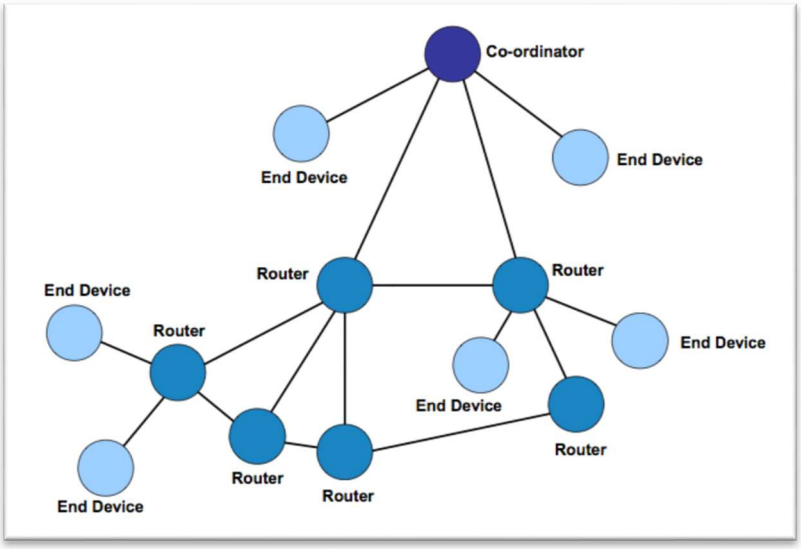


Figura 22: Modelo de red Mesh con protocolo Zigbee

Cada dispositivo *ZigBee* posee una dirección única, de 64 bits denominada dirección MAC que le distingue de cualquier otro dispositivo a nivel mundial, esta dirección viene impresa en el mismo chip *XBee* (Figura 23). Además, dispone de una dirección de red de 16 bits que le asigna el coordinador al asociarse a la red privada. Dentro de una PAN (*Personal Area Network*) todo dispositivo es identificado unívocamente por cualquiera de estas dos direcciones.

De igual modo, cada red *ZigBee* posee dos identificadores que permiten distinguirla de cualquier otra red de su entorno, incluso de las que operan en su mismo canal, permitiendo su coexistencia en un mismo espacio. El primero de estos identificadores se denomina PAN ID y es un valor de 16 bits que se establece al crearse la red. El segundo es el llamado Extended PAN ID, un valor de 64 bits asignado por el administrador de la red y que es añadido, junto al PAN ID, en los paquetes de señalización para permitir que los nodos que deseen asociarse puedan identificar de forma unívoca la red.

Figura 23: Chip Xbee-ZB

Además, dos redes *ZigBee* que compartan un mismo espacio se pueden diferenciar según la frecuencia del canal sobre el que trabajan. Cuando el nodo coordinador crea la red sondea los canales disponibles en la banda utilizada (2,4 GHz) y elige el más adecuado.

El protocolo *ZigBee* utiliza el AES128 (*Advanced Encryption Standard*) para mantener la seguridad de la red. Este algoritmo utiliza claves de 128 bits, lo que asegura la integridad de los datos intercambiados, así como la autenticación de los componentes del sistema, y le da cierta protección contra ataques al sistema (aunque como sabemos, no hay prácticamente sistemas totalmente seguros). Se utilizan dos tipos de claves, clave de red (se cifra en origen y se descifra en cada nodo, repitiendo la operación hacia el siguiente nodo), y clave de enlace (el paquete cifrado en origen permanece cifrado durante su enrutado, y sólo se descifra al llegar a destino).

Para este proyecto se han utilizado los chips de la Serie 2 de *ZigBee*, denominados *XBee*, y fabricados por Digi International. Estos dispositivos poseen un alcance de 100 metros en espacio abierto y de unos 40 en espacios cerrados, suficiente para poder diseñar el sistema que nos ocupa. Cabe destacar que estos chips pueden ser programados en dos modos, modo AT (modo transparente que únicamente permite comunicación punto a punto entre dos miembros de la red) y modo API (permite direccionamiento y *broadcast* de las tramas de datos remitidas entre los componentes de la red). En este proyecto se ha utilizado el modo API.

Estos dispositivos se programan mediante un software facilitado por el fabricante, denominado X-CTU, y transmiten las tramas a nivel de byte, por lo que es necesaria un tratamiento de los datos bastante dificultoso tanto para emitir como para recibir. En la Figura 24 se puede ver cómo sería una trama de este dispositivo, de la que hay que tener en cuenta todos sus bytes para poder establecer una comunicación correcta. Puede verse también cómo está configurada la trama de este dispositivo.

Para realizar su configuración a través del PC con este programa, los chips deben tener un adaptador denominado “explorer”, sobre el que se montan para poder comunicarlo con el PC y el programa en sí.

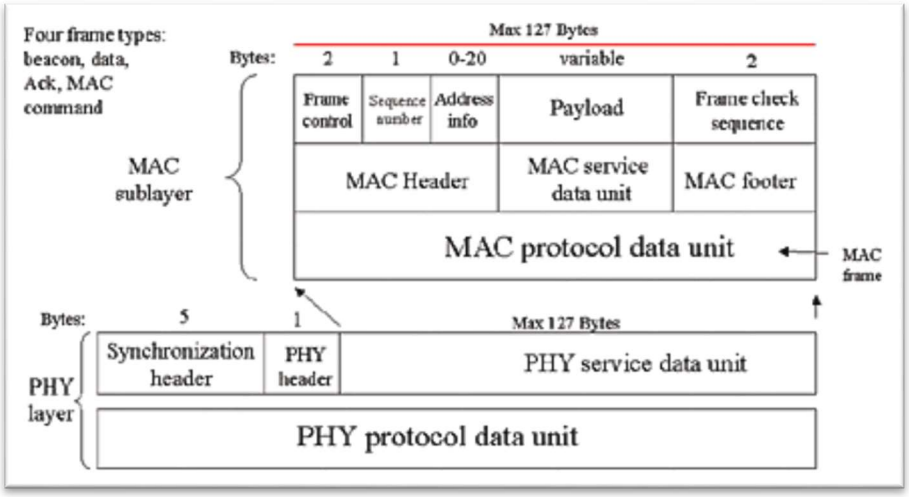


Figura 24: Trama protocolo Zigbee

4.- Requerimientos, análisis y diseño

En este capítulo se tratará de mostrar el proceso seguido para realizar el diseño final del sistema que es motivo de este PFC.

4.1.- Requisitos del proyecto. Alcance.

Requisitos básicos del sistema a implementar. Se enumeran y definen los mismos:

1. Realizar la implementación de un sistema de adquisición de datos basado en sensores con un dispositivo de *hardware* libre. El dispositivo elegido para este fin ha sido Arduino ya que una de las *constraints* del proyecto es que el sistema debe ser *low-cost*.
2. El sistema debe ser inalámbrico, por lo que la comunicación entre la adquisición de datos y el nodo central donde los mismos serán almacenados y procesados deberá utilizar tecnología inalámbrica, de forma que fuese lo menos invasiva para el hogar la instalación de los dispositivos, al no ser necesario el cableado.
3. El sistema a implementar debe poder monitorizar datos de temperatura, humedad, presencia, gas en aire (detector de gas) e inundación. Esto permitirá que sobre los datos proporcionados se puedan definir eventos hacia los actuadores (ej.: si sube la temperatura más de 25 grados, encender el aire acondicionado).
4. Implementar con *hardware* libre los sistemas actuadores sobre los dispositivos a controlar y automatizar (luces, electrodomésticos, etc.). Relés sobre Arduino ha sido la solución implementada.
5. El sistema debe tener un servidor central que contenga una base de datos donde se centralizará toda la información del mismo y que permita acceder a la misma. Esta implementación debe igualmente tener un precio reducido. Por ello, igualmente desde el principio se pensó en el microordenador Raspberry Pi para esta función, y en instalar en el mismo, un servidor LAMP (**L**inux **A**pache **M**ySql **P**hp).
6. El sistema debe proporcionar al usuario la capacidad de configurarlo desde un dispositivo móvil. Para ello se requiere la realización de una aplicación móvil que permita visualizar la información recogida por lo sensores, configurar el sistema domótico, y actuar sobre los dispositivos que el sistema posee (luces, electrodomésticos, válvulas, etc.). Para este desarrollo se ha elegido el sistema operativo móvil Android.
7. Se valora la necesidad de que el sistema a desarrollar pueda integrarse con alguno de los sistemas propietarios de domótica, esto es, que sustituyese parte de los elementos de estos sistemas, aunque este requisito es opcional en este alcance, y su implementación se condicionado a la evolución del mismo, o se dejará como posibilidad de evolución para una fase posterior.
8. Se establecerá algún método para alertar al usuario de eventos críticos en el sistema, como por ejemplo, la presencia de alguien en domicilio una vez activado

el detector de presencia, de un escape de gas (nivel gas por encima de umbral), etc. Se utilizará algún método de comunicación que soporte el móvil (mail, SMS, Line, Whatsapp, notificación, etc.).

9. Se estudiará el modo en que el sistema pueda ser modular, escalable y flexible, esto es, que permita la inclusión de nuevos componentes de una manera eficaz y sencilla. Este requisito puede formar parte de una fase posterior.

4.2.- Análisis y diseño

Una vez establecidos los requisitos, se realiza el análisis y diseño del sistema, para definir la solución técnica más adecuada a los mismos:

4.2.1.- Análisis y diseño hardware

1. El sistema requiere en su requisito 1 un sistema de adquisición de datos *low-cost.* La placa microcontroladora Arduino responde a la necesidad planteada, y dentro de los dispositivos Arduino en el mercado, la más barata es la Arduino UNO R3.

   A excepción de la placa Arduino Mega R3, que tiene la misma tipología que la UNO R3 pero con más pines de adquisición de datos, el resto de dispositivos Arduino están diseñados para cubrir nichos de mercado concretos (necesidades de tamaño reducido como la Arduino Nano, necesidades de potencia de software, como la Arduino Yun, que es una mezcla de Arduino y Raspberry, *wearables*, como la Arduino Lilypad o Gemma. Se decide realizar la implementación sobre una Arduino UNO R3 y una Arduino Mega R3.

   Arduino posee además una amplia gama de sensores que se adaptan a la funcionalidad del requisito 3 (sensores humedad, temperatura, presencia, etc.). De toda la oferta de mercado, se eligieron los descritos en el apartado 3.3 de esta memoria.

   Para cumplir con el requisito 4 (actuadores) se seleccionan módulos adaptados Arduino de relés individuales y una placa con ocho relés.

   Aprovechando la capacidad de los dispositivos *ZigBee* de permanecer en reposo, se construye un prototipo autónomo que permite leer la temperatura de un sensor analógico de temperatura LM35, enviar la lectura al servidor y apagarse, de forma que pueda estudiarse esta capacidad de los chips *ZigBee* y dada la gran cantidad de aplicaciones que pueden necesitar el bajo consumo como premisa de diseño. Se puede ver explicado en el anexo de “Pruebas” (7.4), prueba CP05.

2. Para el establecimiento de la comunicación inalámbrica (requisito 2) entre los diferentes componentes del sistema (esto es, placas actuadoras y placas de adquisición de datos colocadas en cada una de las estancias del hogar del usuario), con el sistema central, existen varias opciones, restringidas por la tecnología de Arduino (debe poder ser soportada por las placas).

   Existe la posibilidad de comunicación vía Wi-Fi, ya que Arduino tiene la capacidad de montar un *shield* Wi-Fi que le dota de esta posibilidad de comunicación. Sin embargo, el precio de este *shield* es casi prohibitivo, por lo

que es descartado, a pesar de la facilidad de programación que ofrece Wi-Fi, y de la facilidad de su implementación.

Otra posibilidad es la tecnología Bluetooth, pero esta tecnología dificulta la labor de formar redes, a pesar de que el coste no es problemático.

Finalmente, se selecciona para configurar la red de comunicación del sistema, la tecnología XBee (ZBee), que cumple la premisa de bajo coste y permite implementar redes *mesh*. Conocer este tipo de redes y su funcionamiento supone un añadido para formar parte del proyecto.

Para su implementación se utilizan los dispositivos XBee (ZBee). Se compra, como equipamiento, tres chips **Xbee 2mW con Antena chip - Serie 2 (ZB)**; para poder trabajar con ellos, dos **Arduino XBee Shield**, que permiten adaptar los chips a las placas Arduino, y un **XBee Explorer USB**, que permite conectar los XBee al PC para configurarlos con X-CTU. En el anexo se explicará cómo se ha realizado la configuración de estos dispositivos ya que merece una referencia especial debido a la complejidad que esto supone.

3. Para el servidor central del sistema (requisito 5) se selecciona una Raspberry Pi, que posee las características necesarias para instalar un servidor LAMP.

   En el momento de iniciar este proyecto existían varios modelos de Raspberry Pi. Se eligió la Raspberry Pi 2 Model B, entendiendo que era la más adecuada, cuyas características constan en el anexo, con una tarjeta SD de 8GB. Sobre este dispositivo se implementará todo el servidor. Se comunicará vía XBee con el resto de los dispositivos adquisidores de datos, y dentro de la red *mesh* será el elemento Coordinador de la red.

   Se le conectará vía USB el chip XBee a la Raspberry.

   El modelo más actual de Raspberry Pi es la 3 Model B, con características más potentes que la 2 (más RAM, 2 puertos USB adicionales, conectividad wifi, bluetooth, etc.).

   El problema más reseñable de Raspberry Pi proviene de su bajo precio: no tiene disco duro. Funciona con una tarjeta SD de memoria. Esto no sería un defecto si no fuese porque dicha tarjeta es propensa a estropearse cuando existen muchos accesos a ella (se han utilizado 3 tarjetas a lo largo de este PFC, debido a que 2 de ellas se estropearon mientras se hacían pruebas).

   Dado que existe una base de datos en la solución del sistema, albergada en la Raspberry, los accesos a la tarjeta son continuos. Para solucionar este inconveniente, se decide utilizar un hub USB que además sirviese de alimentador de la Raspberry Pi (este dispositivo se alimenta con un cargador microUSB muy similar a los utilizados por los móviles).

   Para dotar a la Raspberry Pi de conexión a Internet, se puede utilizar la conexión vía Ethernet (existe un puerto en el dispositivo). Para cumplir el mandato de que el sistema sea inalámbrico, se dota a la Raspberry de un mini adaptador Wi-Fi USB.

4. Para albergar la aplicación móvil, desarrollarla y probarla (requisito 6), se utilizan los teléfonos móviles que he poseído durante la elaboración de este proyecto, un Samsung Galaxy SII con el sistema operativo Android 4.0.3. y un

Motorola Moto G de 2ª Generación con sistemas operativos Android 4.4.4 *Kit Kat* y Android 5.1 *Lollipop.*

En la Figura 25: Esquema Hardware del proyecto puede observarse la arquitectura hardware utilizada para la implementación del sistema domótico.

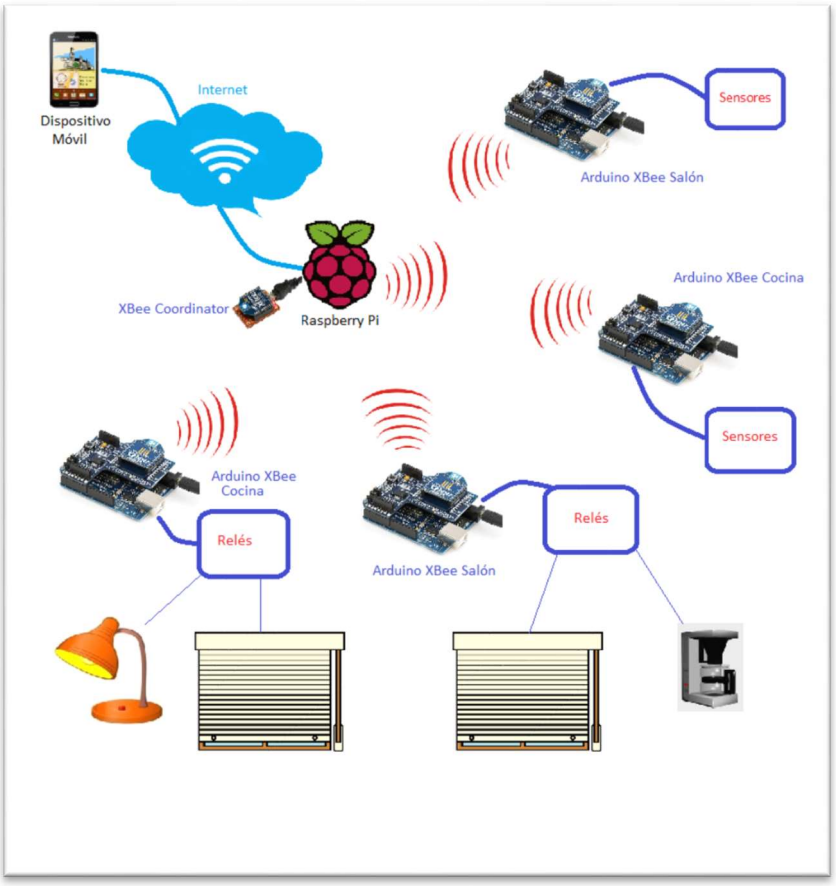


Figura 25: Esquema Hardware del proyecto

4.2.2.- Análisis y diseño software

En este apartado se detalla la arquitectura *software* implementada, y el desarrollo de los componentes *software* que integran el sistema. Esto dará respuesta a los requisitos 6 y 8.

La arquitectura *software* implementada es del tipo centralizada. Las decisiones del sistema se toman en el servidor, en la Raspberry Pi. Las placas Arduino son meros adquisidores de datos o recibidores de órdenes, esto es, toda la inteligencia del sistema reside en el servidor central. Sin embargo, para establecer la red *mesh* y que los datos viajen al servidor central, también ha sido necesario implementar *software* en las placas Arduino.

Se implementa una arquitectura cliente-servidor entre el dispositivo móvil Android y la Raspberry. Esta arquitectura cliente-servidor se fundamenta en un servidor web (Apache). Para ello se instala un servidor LAMP (Linux, Apache, MySql y Php) en la Raspberry Pi.

El dispositivo Android modifica y lee datos de la base de datos MySql a través del servidor web Apache, que ofrece un interfaz PHP permite realizar cualquier tipo de operación sobre la base de datos (borrado, actualización, inserción), modificando los parámetros referentes a los actuadores o a eventos programables.

El interfaz entre Android y el servidor se implementa así: Android realiza una llamada al servidor Apache y ejecuta un script PHP. Este script conecta con la base de datos, extrayendo o introduciendo información, y el intercambio de la información se realiza mediante el estándar ***JSON.*** *JSON* es el acrónimo de *JavaScript Object Notation*, y se trata de un formato ligero de intercambio de datos.

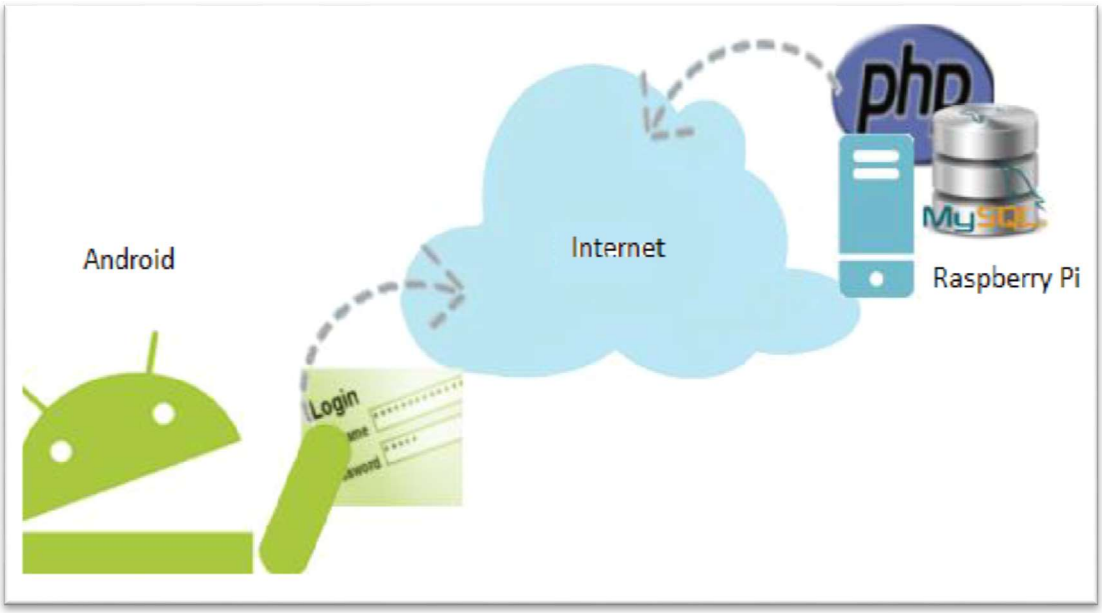


Figura 26: Esquema comunicación Android – Base de datos

Un ejemplo de lo que se comenta: encender luces del salón. Dentro de la base de datos del servidor, existe una tabla con las habitaciones, y otra con los eventos que pueden suceder en dicha habitación. Si se quieren encender las luces, desde Android, se ejecutará un script PHP que cambie el estado del parámetro EST_DISP (estado dispositivo) de la tabla de DISPOSITIVOS al valor 1 (el servidor detectará este cambio y enviará una orden al cliente, en este caso la placa Arduino, que interpretará la misma y apagará la luz correspondiente).

4.2.2.1.- Diseño de la base de datos

La base de datos se ha implementa utilizando una distribución de MySql para sistemas Linux.

Se realiza un diseño de tablas que permita albergar el modelo de datos que incluya toda la información necesaria para que el servidor tenga controlado el sistema. La estructura de la base de datos refleja el estado físico del sistema. Estas son las tablas principales diseñadas:

1. Tabla **USUARIOS**: datos referentes a los usuarios autorizados a entrar en el sistema.
2. Tabla **HABITACION**: datos referentes a las habitaciones que tienen sistemas domóticos instalados, esto es, las que tienen sensores o actuadores pertenecientes al sistema. En esta tabla se almacena el dato correspondiente al direccionamiento *ZigBee* (ADDR_ZBEE), que servirá al servidor para intercambiar información con el dispositivo adecuado.
3. Tabla **SENSORES_ACTUALES**: almacenan las medidas más actuales de los sensores instalados en el sistema.
4. Tabla **UMBRALES**: en esta tabla se almacenan los datos que hacen saltar ciertos automatismos, como el umbral del valor de la luz con el que las persianas se cierran, el valor de gas con el que salta la alarma de gas, etc. Estos valores serán configurables por el usuario, desde la app móvil.
5. Tabla **DISPOSITIVO**: en esta tabla se encuentra la información referente a cualquier dispositivo sobre el que se puede actuar desde el sistema, esto es, luces, persianas, válvulas de agua, gas o riego, electrodomésticos, etc. Está relacionado con el código de la habitación en el que se encuentra instalado, en la tabla HABITACION.
6. Tabla **EVENTOS_PROG**: esta tabla se utiliza para almacenar la programación en diferido de los DISPOSITIVOS. Contiene datos de inicio de evento y fin de evento, a fin de que sirva para introducir escenarios programables.
7. Tabla **HIST_SENSORES**: en esta tabla se almacenan datos de los sensores cada cierto tiempo para poder ver un historial de valores del mismo (por ejemplo, ver los últimos datos de temperatura de las 5 horas anteriores a la actual).

Para la elaboración de la base de datos y manejo de la misma, se ha utilizado el programa auxiliar ***phpMyAdmin*** que proporciona un API más amigable para el usuario de base de datos ***mySQL***.

En la Figura 27 se puede apreciar el esquema de tablas y sus relaciones.

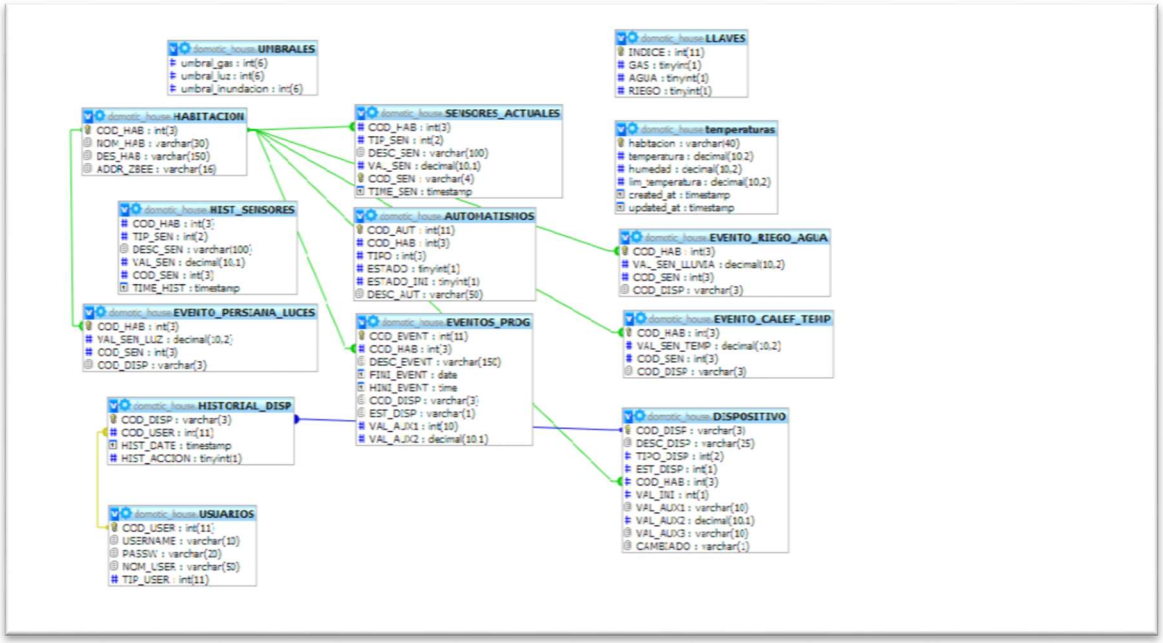


Figura 27: Diseño de la base de datos

4.2.2.2.- Desarrollo Android

En este apartado se va a detallar cuál ha sido el diseño a alto nivel del programa que permite al usuario configurar y controlar el sistema desde su dispositivo móvil.

El desarrollo se ha realizado utilizando Eclipse con el *plugin* de Android Development Tools (Android SDK), que era el utilizado cuando se comenzó el proyecto.

La *app* de control (DomoticHouse) se inicia con una pantalla de *Login* que solicita usuario y *password* del sistema. La aplicación lanza una petición al servidor vía PHP comprobando que el usuario está registrado en el sistema con dicha *password*. En caso de que no sea así, hace vibrar el dispositivo y lanza un mensaje de error. En caso favorable, lanza la pantalla de inicio del sistema que da acceso a un menú con todas las posibilidades del sistema domótico. En las Figura 28 se pueden ver las pantallas de "Login" y de "Menú principal" de la aplicación DomoticHouse.

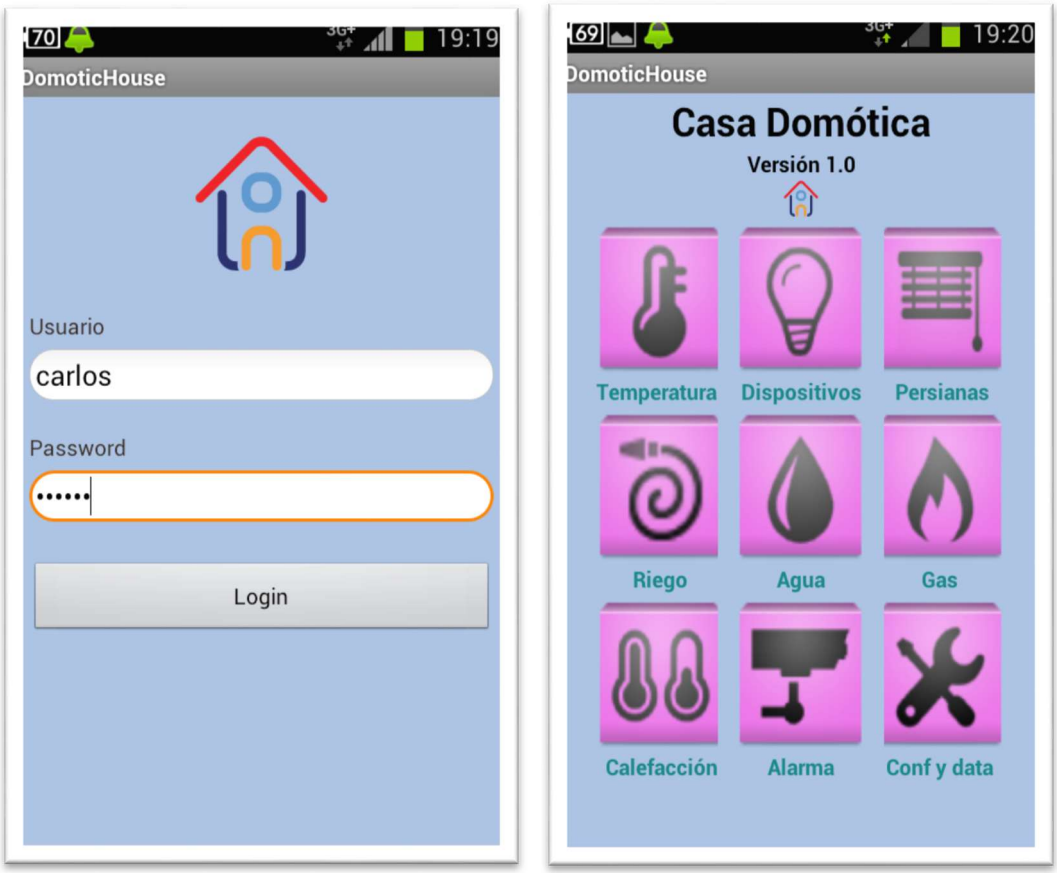


Figura 28: Pantalla de Login del Sistema y Página principal de la aplicación

Vamos a detallar cuáles son las funcionalidades que se han implementado en la aplicación:

1. ***Temperatura:*** esta función del programa permite al usuario acceder a los datos de temperatura y humedad de cada una de las habitaciones. Pulsando la opción, se accede al menú de habitaciones, y ahí permite elegir los datos de la habitación que se desean consultar. También ofrece el histórico de temperatura y humedad de las 5 últimas horas en la habitación seleccionada.
2. ***Dispositivos:*** esta función permite modificar el estado de todos los dispositivos que se encuentran en una habitación. Primero se accede al menú de habitaciones, permite seleccionar la deseada, y aparece el listado de todos los dispositivos modificables que existen en dicha estancia. Al seleccionar el dispositivo en cuestión, este puede encenderse o apagarse y también permite programar en diferido su encendido o apagado. Adicionalmente se ofrece la funcionalidad que **muestra los datos del precio de la luz del día actual**, señalando con color verde el periodo más barato y con color rojo el más caro. Igualmente, cuando se accede a la pantalla de programación del dispositivo, en el cuadrante izquierdo debajo de la pantalla, se muestra el dato de la hora con el precio de la luz más barato a partir de la hora actual, de forma que el usuario sepa cuándo le costaría menos que funcionase el dispositivo (por ejemplo, al programar una lavadora).

Cuando existe la ocasión de acceder a los datos del precio de la electricidad del día siguiente (los datos del día siguiente están disponibles a partir de las 20 horas del día actual), ofrece un botón para acceder a ellos y en la pantalla de programación, se muestra la hora óptima del día siguiente.

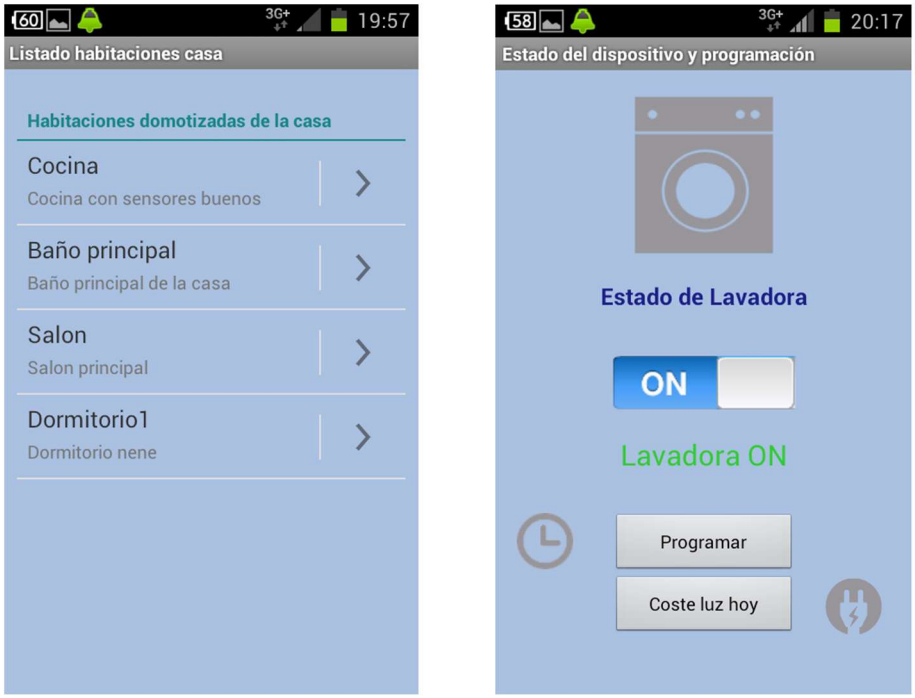


Figura 29: Pantallas de habitaciones y de dispositivo

En la Figura 29 puede observarse el menú donde aparece el listado de las habitaciones con domótica de la casa, y la pantalla de cualquiera de los dispositivos que pueden ser activados por el usuario pulsando el interruptor que aparece en la pantalla, así como programar su encendido o apagado accediendo a la pantalla de programación pulsando el botón “Programar”. En la Figura 30 se aprecia la pantalla de la aplicación donde aparece el precio de la luz por hora diaria (en euros por KW hora) y el menú de los dispositivos sobre los que actuar en una de las habitaciones.

Figura 30: Pantallas de dispositivos sobre los que actuar en una habitación y de precio de la luz cada hora en euros Kw/hora

3. ***Persianas:*** se ha habilitado un botón especial para los dispositivos de tipo persiana, ya que al usuario se le permite decidir cuál es el porcentaje de apertura de las mismas. Para ello se ha habilitado una barra de progreso para indicar cómo se quieren abrir o cerrar, al igual que en la pantalla de programación de las mismas. Para hacer efectivo el movimiento, una vez elegida el porcentaje de apertura, debe pulsarse el "Mover persiana".

   El sistema dispone adicionalmente de un automatismo para que la persiana se cierre automáticamente dependiendo del nivel de la luz en la habitación (dependerá del dato ofrecido por el sensor de luz instalado en la habitación). Este automatismo se tiene activo cuando está en "ON" el botón especificado. La pantalla de la persiana se detalla en la Figura 31.
4. ***Riego:*** la funcionalidad de riego, consiste además de su encendido o apagado (permite controlar una electroválvula de paso de agua), en dos automatismos. Uno establece la posibilidad de que se avise mediante notificación al móvil del usuario de si está lloviendo (si el automatismo está "ON", cuando el sensor de lluvia instalado pasa de cierto umbral, entonces se remite un mensaje al usuario avisándole de este hecho para que lo desactive para ahorrar agua).

   El otro apaga directamente el sistema de riego en caso de que se supere dicho umbral de lluvia si el riego está encendido. En la Figura 31 podemos ver la pantalla de riego con estos dos automatismos descritos.
5. ***Agua y gas:*** las funcionalidades de agua y gas, permiten controlar las válvulas de cierre del agua y del gas (se han considerado aparte dada la importancia de estos dispositivos). Se permite programar la apertura o cierre de la electroválvula, además de su cierre o apertura remota. La pantalla es igual a la de riego pero sin los automatismos, por lo que no vamos a colocar la figura de las mismas.

Figura 31: Pantallas de Persiana y de Riego con sus automatismos

6. ***Calefacción:*** esta funcionalidad da acceso a una pantalla diseñada como si se tratase de un programador de calefacción de los que actualmente se tienen en cualquier hogar. La distribución utilizada es muy similar a la que se tiene en estos. Primero aparece la temperatura actual (como hay que tomar una, se toma la de la habitación principal, en este caso la del salón, ya que suele ser habitual la colocación del programador en estas estancias; podría configurarse el sistema para elegir cualquier otro dato).

   Posteriormente aparece la temperatura que se desea que la calefacción alcance. Esta dispone de botones para modificarla. Igualmente aparece el dato de la histéresis: para que un sistema de calefacción sea eficiente, debe establecerse una ventana de funcionamiento, ya que no es óptimo que cada vez que el sistema baje de la temperatura deseada, se dispare, y que cuando la alcance se apague; esto haría que casi continuamente se encendiese y apagase el sistema. Para ello se establece este dato, que permite al usuario establecer el valor de la ventana de funcionamiento. Se puede fijar hasta el valor máximo de 4 grados de ventana con intervalos de medio grado. En los programadores normales se establece un valor de más menos 1.

   Además, se ha diseñado la pantalla para permitir que se apague o encienda la calefacción (interruptor del sistema), y que se permita que funcione manualmente o auto (mediante la programación establecida, o mediante los valores que el usuario proporciona en el momento).

   En la Figura 32 puede observarse la pantalla del sistema de calefacción.
7. ***Alarma:*** esta funcionalidad permite al usuario activar la alarma de presencia de la casa. Los sensores PIR instalados en cada habitación, si detectan movimiento envían “presencia detectada” al servidor. Si el automatismo de la alarma está

conectado, el servidor remite una notificación al móvil del usuario “Presencia detectada en casa”.

Figura 32 Pantallas de funcionalidad de control calefacción y configuración del sistema

8. ***Configuración:*** esta funcionalidad da la facilidad al usuario de acceder a las pantallas de configuración y de datos del sistema. Se accede a un menú que da las opciones de ver los valores de los sensores de cada habitación, de sus umbrales de operación, de añadir/modificar los usuarios del sistema (se ha colocado una restricción dependiendo del tipo de usuario) y de añadir/modificar/borrar habitaciones, sensores y dispositivos del sistema. Este menú permite realizar una instalación inicial del sistema domótico instalado, y poder modificarlo más adelante en caso de que haya algún cambio. Puede observarse esta pantalla en la Figura 32.

   Además, en el apartado “Valores de sensores”, permite visualizar el valor más actualizado de cada uno de los sensores instalados en cada estancia.

   Una mejora a esta aplicación sería incluir una funcionalidad de “Escenas configuradas”, que permitiese que eligiendo una de ellas, se realizasen varias acciones simultáneamente (por ejemplo, escena desayuno, en la que se abriesen las persianas, se encendiese la cafetera, y se activase el termostato de la calefacción a una temperatura determinada).

4.2.2.3.- Desarrollo Raspberry Pi

En este apartado vamos a analizar los desarrollos en el servidor instalado en la Raspberry en dos bloques:

- Desarrollos PHP (*Hypertext Pre-Processor*), necesarios para que la aplicación Android interactúe con la base de datos *MySq*l a través del servidor web Apache instalado en la Raspberry.
- Desarrollo del programa servidor en sí, implementado en Python, que interactúa con la base de datos *MySql*, detectando los cambios que se han producido en los parámetros de configuración del sistema, y que envía/recibe información a/de Arduino, parte esclava del sistema, así como los mensajes de alerta a la aplicación Android.

Sobre el desarrollo necesario en PHP, no realizaremos un análisis profundo, por su simplicidad: únicamente señalar que se han utilizado scripts cuya tarea única es conectarse a la base de datos, realizando la inserción, extracción o borrado del dato indicado y que el intercambio de datos entre el servidor web y la app Android se realiza mediante encriptación JSON.

Para el desarrollo software del programa servidor, se ha utilizado el lenguaje script nativo de la distribución de *Linux Raspbian*, **Python** (www.python.org). Se trata de un lenguaje de programación utilizado en gran variedad de entornos (se utiliza por ejemplo en la programación de Openstack, que es un entorno de virtualización de redes, para Big Data), y recomendado por Google.

Uno de los inconvenientes en Python es encontrar los errores. Existen *debuggers* tales como **pdb**, pero no tienen la potencia de los de C++ por ejemplo. Otro inconveniente es implementar programas con multiples hilos y es más lento que otros lenguajes como C (tanto que existe la necesidad de implementar partes críticas de los programas en C).

El software desarrollado en Python, utiliza librerías adaptadas a la comunicación con la base de datos y a la comunicación vía *XBee-Zigbee*. Se ha diseñado para que sea un programa que corre continuamente en bucle, y que revisa continuamente todos los eventos que pueden producirse comparando las modificaciones que en los parámetros de la base de datos se han podido efectuar.

La programación a alto nivel del bucle, muy relacionada con las pruebas de sistema implementadas, realiza las siguientes acciones:

1. Revisa la tabla de datos de los dispositivos, para ver si el campo "**CAMBIADO**" es verdadero (valor 1). En caso de que así sea, recoge la dirección de red de la estancia donde está el dispositivo, y remite una orden a dicho dispositivo con el cambio que se quiere realizar (encendido, apagado, subida o bajada de persiana, etc.). Una vez remitida la orden y recibido el NACK de que se ha realizado, cambia dicho campo a falso (valor 0).
2. Revisa la tabla de automatismos, y si la alarma está conectada y recibe una señal afirmativa de presencia de alguna habitación, entonces emite una notificación de alarma de presencia hacia el dispositivo móvil.
3. Si recibe una lectura del sensor de gas por encima del umbral establecido para la alarma, emite una notificación de alarma al móvil del usuario de "Alerta por gas".
4. Revisa igualmente si la lectura del sensor de luz de alguna de las habitaciones está por debajo del umbral establecido, y si el automatismo de la persiana de dicha habitación está activado, envía la orden de bajada de la misma.

5. Revisa la lectura del sensor de lluvia en caso de que los automatismos estén activados. Si el dato supera el umbral establecido, y el automatismo de avisar está activado, entonces remite una notificación al móvil del usuario de "Lloviendo en jardín". Si el automatismo de apagar está activado, y el dispositivo está encendido, entonces lo apaga automáticamente.

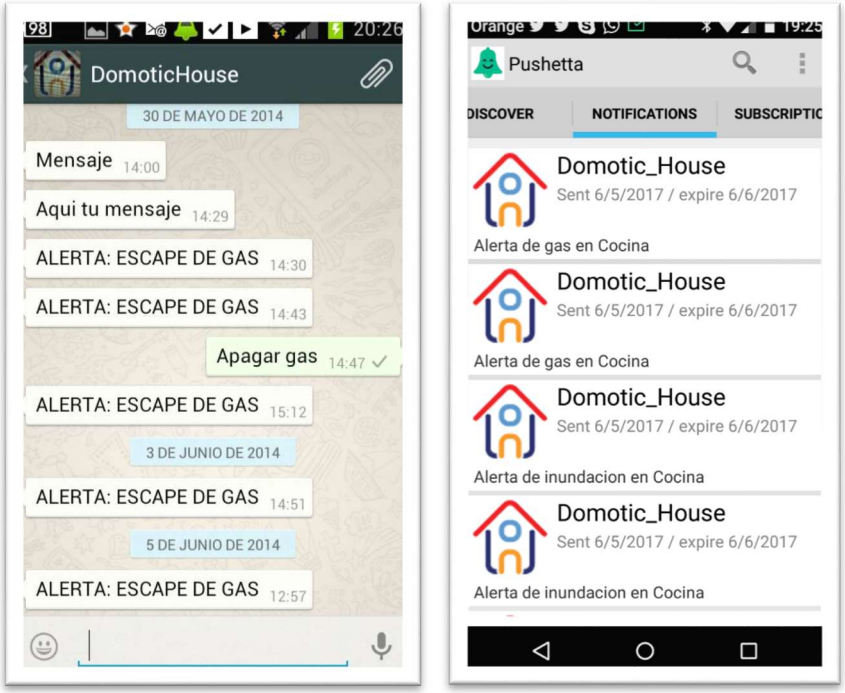


Figura 33 Detalle del envío de notificación en caso de escape de gas con Whatsapp y con Pushetta.

6. Revisa el funcionamiento de la calefacción con su automatismo. Revisa la temperatura actual y la compara con la deseada. En caso de estar por debajo, activa la calefacción, teniendo en cuenta para su funcionamiento la histéresis que esté establecida (si está programado, la que se programó, y si está manual, la que esté establecida).
7. Revisa la tabla de eventos programables, comparando cada minuto la hora con todos los que hay en la tabla. Si hora y fecha coinciden, ejecuta la programación y borra el evento de la tabla, para no sobrecargar la misma.
8. En caso de recibir información de los sensores de un dispositivo, entonces actualiza la información de dichos sensores en la base de datos.
9. Cada hora, guarda la información de datos de todos los sensores en la tabla de histórico. Se podría hacer que este valor fuese configurable (decidir cada cuánto se establece el histórico).

La recepción comunicación cliente-servidor (Arduino-Raspberry), es asíncrona. En un principio se implementó una comunicación síncrona, pero esta no disponía de una función de recepción de paquetes con *timeout*, lo que provocaba la detención en espera del programa. Este se reanudaba cuando el servidor recibía algún paquete de datos por parte de algún Arduino. El protocolo asíncrono permite enviar y recibir de manera continua información, minimizando el tiempo de ejecución de una orden y pareciendo inmediata.

Se ha establecido un código para cada orden de cada tipo de dispositivo (bombilla, electrodoméstico, etc.) instalado o conectado a Arduino en cada habitación. Los códigos de los dispositivos se han asignado utilizando la codificación del sistema domótico X10, lo que en un futuro podría permitir la integración del proyecto aquí desarrollado con los actuadores X10 estándar de manera sencilla, sin necesidad de realizar cambios en el sistema actual.

4.2.2.4.- Desarrollo Arduino

El desarrollo Arduino se ha realizado utilizando el entorno de desarrollo en Arduino (IDE). Este entorno es el encargado de gestionar la conexión entre el PC y el *hardware* de Arduino, con el fin de establecer una comunicación entre ellos. Esta se realiza a través de la carga de programas (*sketches*) desde el PC hacia el *hardware* de Arduino. Se explica más detalladamente todo lo referente a este entorno en el apartado 7.3 (Configuraciones). En la Figura 36 se observar el aspecto de este IDE.

Dependiendo del modelo de Arduino con el que se trabaja, se establecen una serie de parámetros que permiten la carga del software y la recepción del resultado de ejecución de los programas (velocidad de la línea serie, capacidad de la memoria, etc.).

El lenguaje de programación es muy similar al de C. En la Figura 34 puede apreciarse el montaje realizado en este proyecto con los sensores cableados, y en la Figura 35 puede verse un detalle del prototipo mostrando por pantalla la información referente a la humedad y temperatura ambiente.

Dado que se pretende realizar una implementación para todos los dispositivos conectados a Arduino, la base inicial del programa supone incluir software de configuración, donde se especifica qué sensores o actuadores y en qué número van conectados a la placa. Una vez establecida esta configuración, el programa global sabe qué subrutinas del programa de ejecutar (esto es, si tiene sensor de gas y de humedad pero no tiene sensor de presencia, en el bucle se ejecutarán las subrutinas correspondientes a los sensores de humedad y gas únicamente).

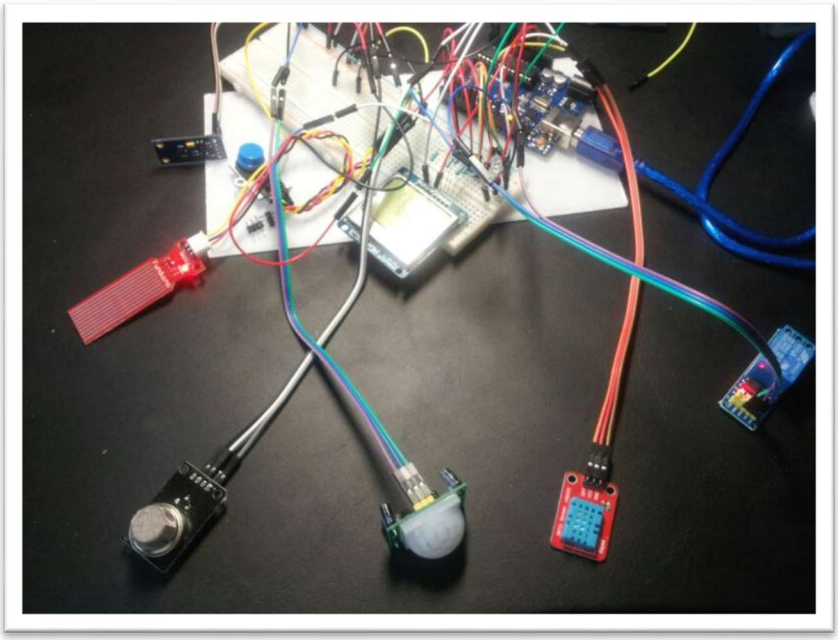


Figura 34 Montaje de sensores sobre Arduino Mega

Otra parte importante del programa es la de la comunicación *ZigBee* con el servidor. Se cargan para esta comunicación las librerías de comunicación "XBee.h" y las funciones "xbee.readPacket()" y "xbee.getResponse()" para enviar el ACK.

La subrutina de comunicación es un *case* con todos los valores posibles que pueden recibirse del servidor, cambiando el valor de la variable de estado correspondiente (por ejemplo, si se recibe un valor 1 correspondiente a apagado de gas; esto hace que se cambie el valor de la variable "estado_Gas" a 0, lo que quiere decir que queremos cerrar la válvula de paso del gas. En la función "cerrar_Gas", dentro del bucle que corre continuamente, se evaluará si ha cambiado dicha variable, y se enviará una orden "LOW" a la electroválvula que cierra el gas, lo que cerraría el paso del mismo.

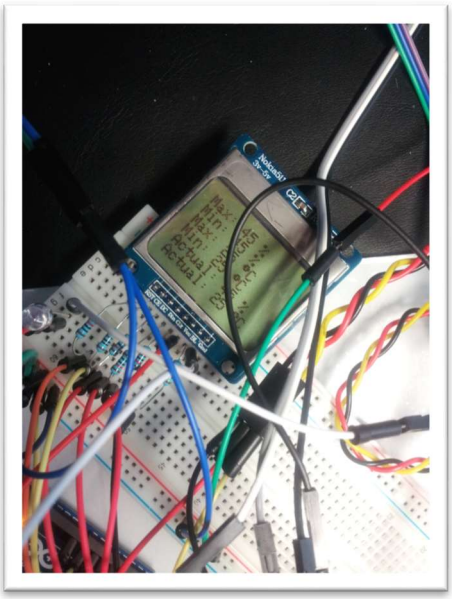


Figura 35 Detalle LCD Sistema con datos de humedad y temperatura

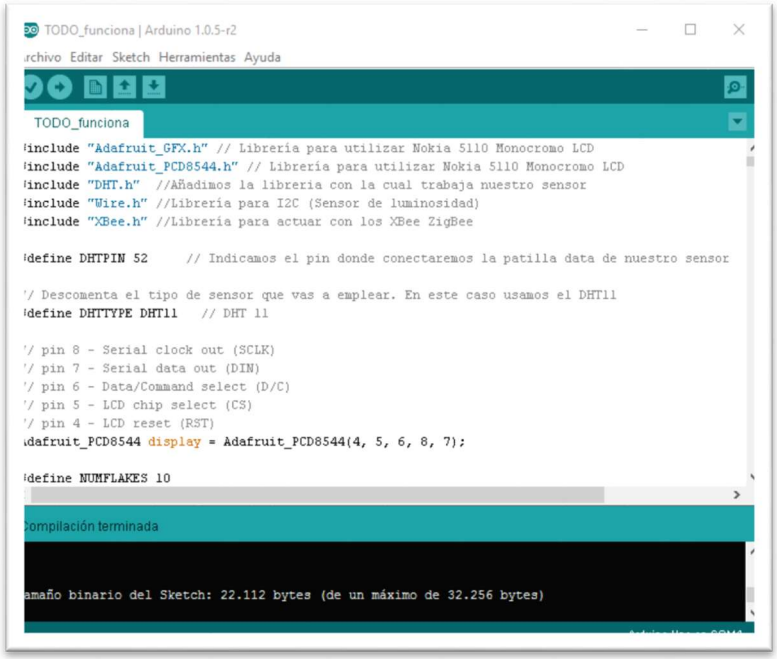


Figura 36 Entorno IDE de desarrollo Arduino

Esta misma rutina se repite para el resto de dispositivos de encendido y apagado.

Para el envío de los datos de los sensores, se utiliza la rutina "xbee.send (objeto xbee)", esperando el ACK del servidor como que se ha recibido la información correspondiente.

En el montaje de la placa Arduino principal, hubo que configurar los *sketchs* de lectura de datos de los sensores de gas, humedad y temperatura, inundación, presencia, una placa LCD para la presentación local de datos y un relé para simular el apagado y encendido de los dispositivos.

Cada uno de estos sensores tiene su propio método de conexión y su propia manera de captura de datos (los hay analógicos, digitales, con bus I2C). Las rutinas de todos ellos están integradas en el desarrollo. Primero se probaron por separado y finalmente se integraron todos en la misma rutina.

En el anexo dedicado al código (7.4) se describen las subrutinas utilizadas para la configuración de cada uno de ellos. No me extenderé más pues la descripción de todos los trabajos realizados para configurar los dispositivos daría para una memoria aparte.

En el diagrama de flujo de la Figura 37 se puede observar la rutina básica que se ha implementado para los dispositivos Arduino que soportan los sensores y el envío de información al servidor.

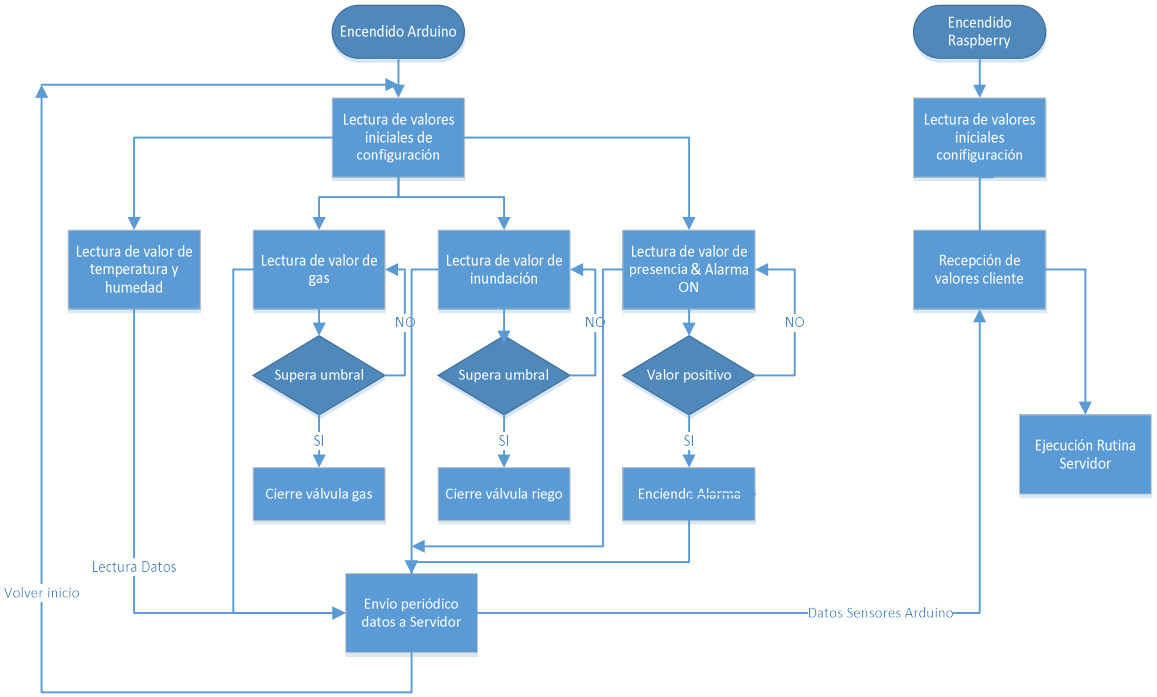


Figura 37 Diagrama de flujo básico del programa Arduino

4.3.- Pruebas de sistema y resultados

Una vez finalizada la instalación de todos los elementos del sistema (sensores de humedad y temperatura, sensor de movimiento, sensor de inundación, sensor de gas, relés, pantalla LCD) y que tenemos en funcionamiento la parte cliente y servidora, así como la aplicación Android del usuario, queda probar cada uno de los componentes el mismo y verificar que su funcionamiento es el adecuado.

Las pruebas podrían estructurarse atendiendo a la parte del sistema que corresponden:

1. Funcionamiento del entorno Android contra un servidor LAMP alojado en el servidor de Raspberry Pi (App Android de control del sistema vía móvil).
2. Funcionamiento de los sensores y cambio de la información que proporcionan al sistema cuando se detecta que su valor supera el umbral establecido en la configuración en el servidor.
3. Pruebas extremo a extremo.

Debido a la gran variedad de combinaciones existentes, las pruebas detallarán aspectos genéricos del sistema que deberán cumplirse, y permitirán testear la integridad del mismo. Estas pruebas se llevarán a cabo según el procedimiento establecido y descrito en la prueba; nos centraremos principalmente en pruebas extremo a extremo (*end2end)*. Se muestran aquí algunas y el resto en el anexo.

Los **casos de prueba** serán establecidos con la siguiente configuración de datos:

- Identificador de caso de prueba (**CPXX**): código alfanumérico que permite trazar la prueba, donde XX es el número de la prueba.

- Descripción del caso de prueba: una descripción de en qué consiste.
- Datos de entrada de la misma: ficheros, parámetros de entrada, configuración necesaria para realizar la prueba.
- Datos de salida de la misma: resultados que se esperan de la ejecución de la prueba.
- Requerimientos de la prueba: necesidades de la prueba

Identificador	CP01
Descripción	Ingreso desde la pantalla de login de la aplicación Android al sistema, mediante la introducción en el sistema de usuario y password. Comprobación de caso de error en la autenticación.
Entrada	Introducción de los parámetros en la pantalla de login, tanto correctos como fallidos.
Salida	Caso de ser correctos, permitirá acceder a la pantalla principal de menú de la aplicación. Caso de ser erróneos, la aplicación emite mensaje de error y hace vibrar el terminal.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente.
Resultado	Correcto: en caso de introducir mal el user/passwd, indica este hecho, y da paso si se introduce correctamente.

Identificador	CP02
Descripción	Obtención por parte del usuario de los datos de temperatura y humedad de cualquiera de las habitaciones configuradas en el sistema domótico.
Entrada	Se configura la placa Arduino correspondiente, y en la misma se tratan de modificar los datos de humedad y temperatura (mediante el aliento es una buena prueba) para que se envíen al sistema (se visualizarán en el IDE de Arduino que los valores han sido modificados).
Salida	Visualizar los datos que se han observado en el IDE de Arduino en la pantalla correspondiente a la temperatura y humedad de la app móvil.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino Mega con el sensor de temperatura y humedad configurado así como la conexión *ZigBee* para la placa Arduino y para Raspberry Pi, de modo que se comuniquen.
Resultado ejecución	Correcto (se visualiza que los valores modificados son presentados en la aplicación)

Identificador	CP03
Descripción	Verificación de apagado de válvula de gas desde la aplicación móvil.
Entrada	Se accede a la aplicación DomoticHouse, introduciendo login y password. Se pasa al menú principal. En este se accede a la funcionalidad "Gas". Desde esta pantalla, se pulsa sobre el interruptor central, para cambiar el estado. La válvula de gas estará emulada por un led que inicialmente estará encendido.

Salida	Debe verificarse que el relé conectado a la placa Arduino que emula la válvula de gas, apaga el led correspondiente respondiendo a la orden emitida desde la aplicación móvil.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose).
Resultado	Correcto (ante la orden el led es apagado).

Identificador	CP04
Descripción	Verificación de envío de alarma ante un escape de gas (simulado) a la aplicación móvil.
Entrada	Se simula un escape de gas para que lo detecte el sistema, a través de un mechero (se aproxima el mismo al sensor dejando escapar el gas).
Salida	Debe verificarse que en el móvil se recibe una alarma indicando que existe un nivel anormal en las concentraciones de gas en la habitación correspondiente.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose). Sensor de gas correctamente instalado en Arduino
Resultado	Correcto: ante el escape, se recibe una notificación en el móvil avisando de "Alerta de Gas en Cocina".

4.4.- Seguridad en Internet de las Cosas y *ZigBee*

Uno de los puntos críticos que tiene el IoT es la seguridad de sus dispositivos, tema que todavía no se ha solucionado por los fabricantes, y que va a ser crítico en pocos años, debido a la cantidad de dispositivos conectados a la red de estas características.

El viernes 21 de octubre de 2016 se produjo un ataque a gran escala de DDoS (*Distributed Denial of Service*) o ataque distribuido de denegación de servicio, que básicamente es que se ataca un servidor desde muchos puntos para que se bloquee y deje de funcionar, contra **Dyn**. Debido a esto, decenas de webs, servicios y varias redes sociales como Twitter, Reddit, Github, Amazon, Spotify se cayeron.

Hasta aquí no hay nada nuevo si no fuese porque este ataque fue realizado utilizando una vulnerabilidad existente en millones de accesorios, dispositivos y electrodomésticos conectados a la red, también conocido como el Internet de las Cosas. El ataque lo realizaron *botnets* utilizando Mirai (este *malware* infecta a estos dispositivos, explotando las vulnerabilidades de los *firmwares* y sistemas operativos de dispositivos sencillos del IoT y haciendo que focalicen sus ataques). En este caso, Mirai explota que la mayoría de estos dispositivos (cámaras IP, *routers* caseros, impresoras, etc.) mantienen sus *passwords* por defecto (del tipo 1234 o similares, o el mismo fabricante coloca la misma *password* para todos sus dispositivos, con lo cual si se conoce una se conocen todas).

La situación de inseguridad del internet de las cosas es tan grave que no hacen falta grandes conocimientos técnicos para coordinar a millones de dispositivos y crear un ataque a gran escala, como se pudo comprobar recientemente.

ZigBee Security

Como se ha comentado en el apartado 3.4, la capa de red de *ZigBee* asegura la integridad y encriptación de las tramas transmitidas aplicando el algoritmo de encriptación AES (AES modo CCM) con una longitud de clave de 128 bits, y asegura su integridad utilizando un bloque de mensaje cifrado con código de autenticación (CBC-MAC).

Para verificar la implementación de esta seguridad, varios estudios han sido realizados, y en la mayoría de ellos, se trata de encontrar las vulnerabilidades de los dispositivos comerciales, utilizados en sistemas de automatización, bombillas inteligentes, o cerraduras controladas por *ZigBee*.

Un estudio que podemos encontrar en la red, es el de Li Jun [14], en el que enseña cómo *hackear* los dispositivos, así como protegerse de estos hackeos. En este estudio, muestra cómo encontrar la clave de cifrado en el *firmware* de los dispositivos así como "olfatear" (*sniff)* la clave de red que es enviada como texto plano en las tramas de *ZigBee* cuando la conexión entre el dispositivo y el controlador se hace vía OTA (*On The Air*).
En la Figura 38 puede observarse el envío en texto plano de dicha clave.

Sin embargo, en el estudio *ZigBee* Exploited [15], el autor avisa de que los *hackers* pueden comprometer las redes *ZigBee* y tomar el control de todos los dispositivos conectados a ella. Las claves de cifrado son brevemente transmitidas sin cifrar cuando un nuevo dispositivo se une a la red. Algunos dispositivos utilizan la clave maestra por defecto, lo que significa que esta es transmitida cuando un nuevo dispositivo se añade a la red. La clave podría ser capturada por un atacante o ladrón que podría por ejemplo abrir una cerradura inteligente.

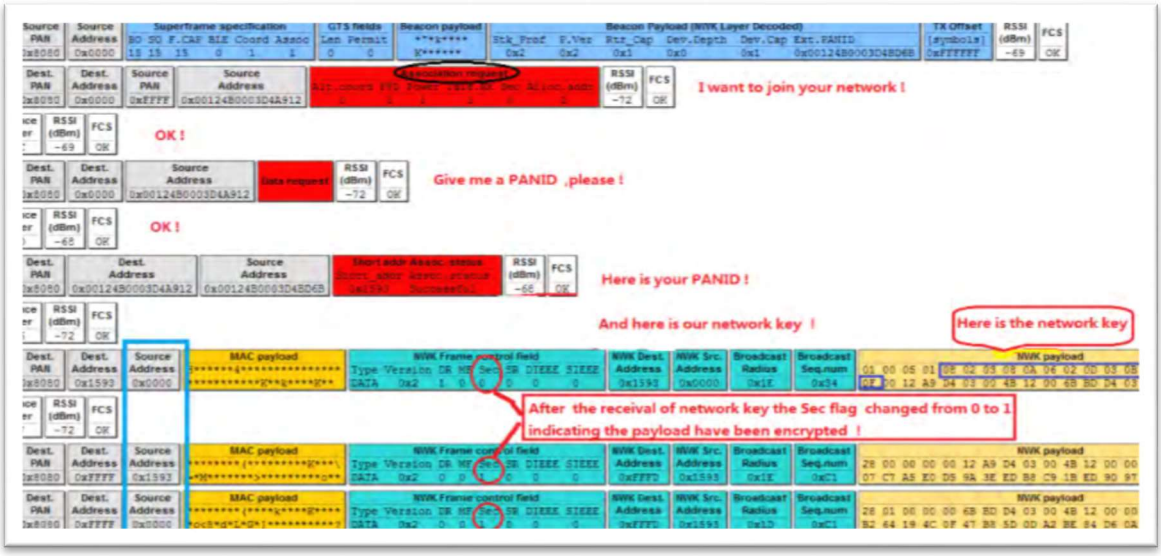


Figura 38 Tramas comunicación ZigBee en configuración

Todo esto no quiere decir que no se pueda utilizar *ZigBee*. El algoritmo de cifrado que tiene es muy robusto, pero es muy dependiente de que no se conozca la clave. El problema es que ciertos fabricantes únicamente llegan a los mínimos de seguridad exigidos para obtener una certificación. Se está tratando desde la *ZigBee* Alliance de que todos los dispositivos cumplan los estándares de *ZigBee* 3.0.

Algunos de los puntos que deberían cumplirse son:

- Borrado de información sensible en caso de manipulado de algún dispositivo (esto es, si un *hacker* manipula el *firmware* de un dispositivo, este lo detecta y borra toda su información sensible, incluida la *password* de red).
- El enlace de comunicación por defecto que trae *ZigBee* no debería utilizarse ya que envía información sin cifrar.
- En *link* que se realiza con el coordinador debería establecerse de un modo seguro al asociar un dispositivo a la red, y fuera de banda a ser posible para evitar los posibles *sniffers*.
- Como en todo sistema que pretenda tener seguridad, el renovado de las claves debería ser obligatorio cada cierto tiempo, haciendo desaparecer así las posibles rupturas de seguridad que ya se hubiesen realizado.

Otro documento de gran interés acerca de la seguridad en el Internet de las Cosas (IoT) es el publicado por Bitdefender, “Riesgos en el hogar conectado” [16], donde se realiza el análisis de varios dispositivos comerciales y sus debilidades ante amenazas.

Para más información acerca de la seguridad en las tramas de *ZigBee*, se puede consultar el manual “*Waspmote ZigBee: networking guide*” [21] en su capítulo 14.

5.- Comparativa de costes y prestaciones con otros sistemas domóticos. Resultados

Aunque la comparativa de las prestaciones y precios de varios sistemas domóticos sería bastante extensa (hay que diseñar y definir la funcionalidad básica sobre la que comparar los mismos, y sobre estas funcionalidades básicas, comparar los precios que tendrían estas diferentes configuraciones), sí se quiere en esta memoria realizar el análisis de costes de varias tecnologías domóticas.

Comenzaremos estableciendo el marco de referencia de la comparación. Se tratará de una vivienda en la que estableceremos un sistema domótico centralizado. Compararemos X10, KNX, ZWave y nuestro sistema.

La vivienda constará de unos 65 m2, altura de techo de 2,5 m. y será la vivienda básica de dos dormitorios (salón, cocina, baño, dos dormitorios, recibidor y pasillo).

Consideraremos igualmente que no se trata de una vivienda a construir, sino que se trata de una vivienda ya construida sobre la que tendremos que actuar y realizar obras.

Veamos qué funcionalidades pretendemos ofrecer en esta instalación simulada:

- Dos enchufes con control de consumo por estancia (incluso en recibidor y pasillo).
- Dos interruptores *dimmers* reguladores de la intensidad.
- Detector de presencia en todas estancias.
- Detector de humo e inundación en cocina.
- Detector de inundación en baño.
- Detectores de apertura en todas habitaciones (ventanas) + puerta exterior.
- Sensores de humedad y temperatura en todas habitaciones.
- Cuadros domóticos centrales (*Gateway* y controlador + pantallas control).
- Un termostato de pared en el salón para controlar la calefacción.

(Nota: los precios están basados en dispositivos ofrecidos por varias webs domóticas.)

Elemento	Sistema KNX (Precios varias fuentes)	Sistema ZWave (Precios [17])	Sistema X10 (Precios [19])	Proyecto low-cost Precios [18]
Canalizaciones	1.200 €	650 €	0 €	300 €
Cuadro domótico	3.800 €	0 €	0 €	0 €
Centro de control + Gateway	1.700 € Pantallas táctil	250€+350€=600€ Pantalla táctil + Gateway	129€+86€ = 215 € CM15 + Pantalla táctil	39,9€+24€x2=87,9€ (Raspberry+ adaptador USB XBEE + XBEE)

Interruptores X12	60€x12+86€x6=1.236€ Interruptores + acopladores bus	(45€+12€)x6=342€ Interruptores dos canales dimmers + bypass	53€x6=318€ Interruptores 2 canales	7€ Placa módulo 4 relés
Enchufes control consumo	150€x12=1.800€ (Enchufes con interruptor)	50€x12=600€	38€x12=456€ (Módulo X10 RF)	7€ Placa módulo 4 relés (esta funcionalidad debe implementarse con relés o N/A)
Detector de presencia	6x170€=1.020€	6x44€=264€	6x50€=300€	6x40€+6x3€+6x30€=438€ (Incluimos aquí las placas Arduino + sensor PIR+Chip *ZigBee*)
Detector humo	87€	60€	90€ **Marmitek SC9000**	5€
Detector inundación (x2)	120€x2=240€	60€x2=120€	30€x2=60€ **Marmitek SC9000**	5€x2=10€ (Funduino)
Detectores apertura	88€x7=616€	60€x7=420€	38€x7=266€ **Marmitek SC9000**	1,3€x7=9,1€ (Conectados a la placa de la habitación)
Sensor humedad y temperatura	370€x6=2.220€ Schneider Mtn60050001	Integrados con los de apertura	105€x6=330€	2,32€x6=13,92€ DHT22
TOTAL	**13.919€** (Posiblemente habría que añadir más acopladores de bus)	**3.056€**	**2.035€**	**872,92€**

Tabla 1: Presupuestos diferentes tecnologías domóticas

A tenor de los resultados obtenidos en la tabla, podemos ver que KNX es el sistema más caro con diferencia. Tiene la ventaja de que una vez que está instalado, su mantenimiento es mínimo. Se trata de un sistema con arquitectura centralizada, de gran robustez. Sin embargo, el precio de sus instalaciones es alto (habría que sumar al presupuesto realizado, el plan de proyecto que suele acompañar a todo proyecto domótico de estas características, y las líneas eléctricas que sería necesario tirar por el inmueble).

El segundo más caro es ZWave, pero muy alejado, al igual que X10. X10 tiene la inteligencia en los dispositivos que añade, aunque actualmente no es un sistema domótico muy utilizado por la limitación de sus funcionalidades. ZWave sin embargo está en alza, y existen una gran cantidad de sistemas basados en esta tecnología. Este sistema es el más similar al que se ha implementado en este proyecto.

Vemos que el coste de dotar a una casa de las mismas funcionalidades con nuestro sistema, es bastante más barato (se basa en un control vía app móvil y Raspberry, lo que evita tener centralitas de alto precio). Quizás habría que añadir las

cajas necesarias para incrustar en las paredes las placas, de forma que el cableado no estuviese a la vista (de bajo coste también, y por ello se ha añadido el precio de las canalizaciones).

En cuanto a funcionalidad, en esta implementación todos presentan similares. En la tabla siguiente se ve una comparativa de prestaciones a alto nivel en diferentes aspectos.

	KNX	ZWave	X10	Proyecto
Tamaño de la red	2E16	2E16	2E8	2E16
Velocidad de datos	9,6Kb/s.	20 – 250 Kb/s.	20 b/s.	20 – 250 Kb/s.
Coste	Alto	Medio	Bajo	Bajo
Interfaz	Gateway a nivel de aplicación	Gateway a nivel de aplicación	Solución adaptada	Gateway a nivel de aplicación
Coste instalación	Alto	Bajo	Bajo	Bajo
Conectividad	Media	Media	Baja	Media
Seguridad	Alta	Media	No	Media

Tabla 2: Comparativa tecnologías domóticas

6.- Conclusiones y vías abiertas

La elaboración de este proyecto ha sido muy satisfactoria, puesto que se han cumplido la mayor parte de los objetivos que en un principio se habían marcado.

Se ha conseguido realizar un prototipo de un sistema domótico modular, a un bajo precio y capaz de integrarse con sistemas domóticos de estándares comerciales como frontal del mismo: el sistema de adquisición de datos de los sensores, el programa Android de control del mismo y la parte servidora, podrían ser incluidos con facilidad en un sistema domótico comercial, sustituyendo los componentes que estos ofertan actualmente, y compitiendo en precio con ellos.

Esta integración sería una de las vías abiertas para futuras ampliaciones del alcance del proyecto: se ha instalado en la parte servidora la librería "Mochad" desarrollada por terceros como API para poder interactuar con el sistema X10 de una manera sencilla. Debería adquirirse un módulo controlador **CM15USB** de X10, que se conectaría vía USB a la Raspberry Pi, y cualquiera de los módulos que posee X10 como actuador (interruptores, *dimmers*, módulos para subir y bajar persianas). Una vez hecho esto, podríamos configurar la parte servidora para que se emitiesen órdenes hacia el módulo X10 que podrían ser directamente interpretadas por dicho controlador y enviadas por el sistema eléctrico de la casa al módulo correspondiente.

También sería muy sencillo el acoplamiento de este sistema con el bus KNX. El alto precio de los sistemas frontales de este estándar lo haría interesante. Existe para ello un API de KNX que traduce las órdenes de Python al bus del sistema KNX, aunque para realizar estas pruebas sería necesario disponer del bus del sistema. Esta alternativa podría dar lugar quizás a una ampliación del proyecto, ya que KNX es utilizado también en sistemas comerciales, y por lo tanto podría gobernarse desde una planta industrial hasta como he mencionado anteriormente cualquier hogar.

Como se ha comentado anteriormente, el hecho de que los dispositivos *ZigBee* estén proliferando, hace que se piense en la posibilidad de interactuar con dispositivos comerciales y tratar de que puedan gobernarse con el sistema implementado. Igualmente sería una buena opción integrar las tres tecnologías de comunicación más utilizadas, Wi-Fi, *ZigBee* y *Z-Wave* (estándar similar a *ZigBee* con gran cantidad de dispositivos en el mercado).

Una vía abierta y que hubiese sido muy interesante realizar ha sido la de que los dispositivos Arduino fuesen autoconfigurables, para facilitar la experiencia de usuario en caso de intentar realizar un sistema comercial. Para ello, debería profundizarse en la arquitectura del sistema, de modo que se pudiese facilitar que una vez se va a instalar la domótica de una habitación, con pulsar un interruptor del dispositivo a instalar en la misma, se actualizase todo el sistema y quedase configurado inmediatamente. Esta parte del proyecto no estaba contemplada y no se ha llevado a cabo, pero sería clave en el posible desarrollo de un sistema que tuviese como objetivo su comercialización. El

dispositivo que incluiría en una caja electrónica el *hardware* necesario de los sensores con Arduino y *XBee* con sus componentes, incluyendo el botón externo de autoconfiguración, el cual al pulsarse desencadenaría todo el proceso de autoinstalación en el sistema (actualización de la base de datos con la información nueva a incluir). Pero conseguir esto podría formar parte de un nuevo PFC por su complejidad, basado en el autodescubrimiento de redes e inventariado de sus componentes (tema de gran actualidad y complejidad).

Otra vía abierta y que daría para un estudio más profundo y con componentes prácticos, sería el de mejorar la seguridad de los dispositivos *ZigBee*, ya que son de aplicación en la mayoría de los dispositivos inteligentes del IoT. Uno de los quebraderos de cabeza de las grandes compañías en este momento, y que pretenden repartirse el pastel del IoT, es la seguridad de los dispositivos empleados con dicha tecnología.

Cabe resaltar las dificultades que se han encontrado realizando el interfaz gráfico del programa Android, debido a la poca facilidad que el IDE de Eclipse con las ADK Tools ofrece (quizás otro entorno de programación ofrezca mayores facilidades). Esta programación ha sido totalmente manual, mediante la programación XML de todos sus componentes.

También debo reseñar la gran dificultad que he encontrado al incorporar los dispositivos *XBee* al proyecto, pues su programación y configuración ha creado grandes problemas, y debido a ello, en varias ocasiones se ha tenido que cambiar la arquitectura del programa servidor alojado en la Raspberry y desarrollado en Python. Aun destacando lo interesante de su manejo, la curva de aprendizaje en estos dispositivos es más lenta que el de otras tecnologías utilizadas en el proyecto.

Sería interesante hacer un **análisis detallado del mercado**, para intentar encontrar un nicho en el mismo (se podría mezclar la domótica con la tele-asistencia a ancianos, por ejemplo) y realizar un ***caso de negocio*** de un posible producto que surgiese de este prototipo (habría que realizar un análisis detallado de costes y beneficios para establecer de inicio la viabilidad financiera). Si este caso de negocio resultase positivo, podría realizarse un proyecto para sacar el producto al mercado.

La mejor manera de realizar este proyecto sería mediante la metodología **Agile**, planteando para el primer *sprint* de este el mínimo producto necesario para salir al mercado, y en posteriores *sprints* del proyecto, ir añadiendo funcionalidades al *backlog* de producto hasta conformar una versión más o menos definitiva del mismo que pudiese comercializarse. Aquí, el primer paso sería obtener financiación y buscar un cliente con el que desarrollar el primer prototipo. Fundamental para ello la realización del estudio de mercado y de la competencia para conformar los requerimientos mínimos del producto y los precios actuales de equipamientos similares.

Como conclusión final, debo añadir que realizando este proyecto, desde el punto de vista personal, he disfrutado investigando todas y cada una de las necesidades que tenía el mismo, he adquirido una gran cantidad de conocimientos técnicos de muy variada índole y el resultado final del proyecto ha sido el esperado. El potencial de las redes de sensores inalámbricas es muy amplio, y ya se están aplicando sobre campos como el

control sobre de entornos físicos (control de incendios, control de niveles de contaminación, etc.) y para el desarrollo de lo que se ha denominado *Smart Cities* y el *IoT*. Se han abierto muchas posibilidades que desconocía que existían antes de realizar este trabajo. Seguiré investigando para realizar la implementación de más funcionalidades y la mejora del prototipo (reducción de tamaño vía circuito impreso PCB y auto-configuración y auto-descubrimiento de los dispositivos) que pudiese en un futuro suponer una oportunidad de negocio.

7.- Anexos

7.1.- Características técnicas de los dispositivos

7.1.1.- Raspberry Pi

	Model A	Model B
Target Price	20,00 €	28,00 €
SoC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)	
CPU	700 Mhz ARM1176JZF-S core	
GPU	VideoCore IV, OpenGL ES 2.0, 1080p30 Full HD HP H.264	
Memory	128 MiB SDRAM	256 MiB SDRAM
USB 2.0 ports	1	2 (via integrated USB hub)
Video outputs	Composite RCA, HDMI	
Audio outputs	3.5 mm jack, HDMI	
Onboard storage	SD / MMC / SDIO card slot	
Low-level peripherals	GPIO pins, SPI, I²C, UART	
Onboard network	none	10/100 wired Ethernet (RJ45)
Real-time clock	No clock or battery	
Power ratings	500 mA (2.5 Watt)	700 mA (3.5 Watt)
Power source	5 Volt via MicroUSB or GPIO header	
Size	85.60mm x 53.98mm	
Supported OS'es	Debian GNU/Linux, Fedora, Arch Linux	

Tabla 3: Características de Raspberry Pi

7.1.2.- Arduino

En la siguiente figura pueden observarse las entradas y salidas detalladas de la placa Arduino Uno R3.

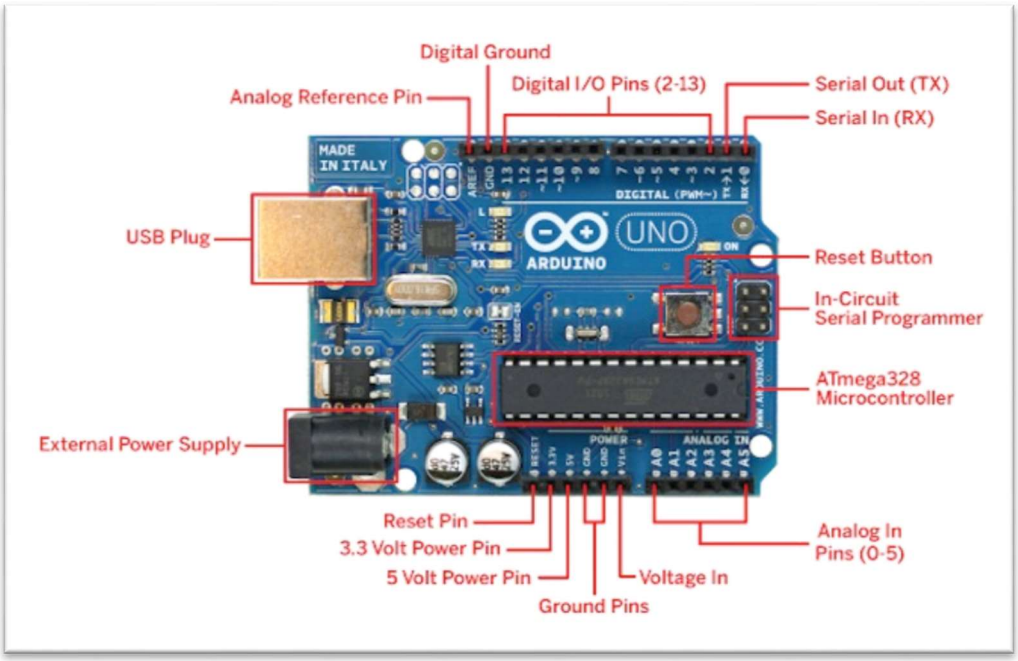


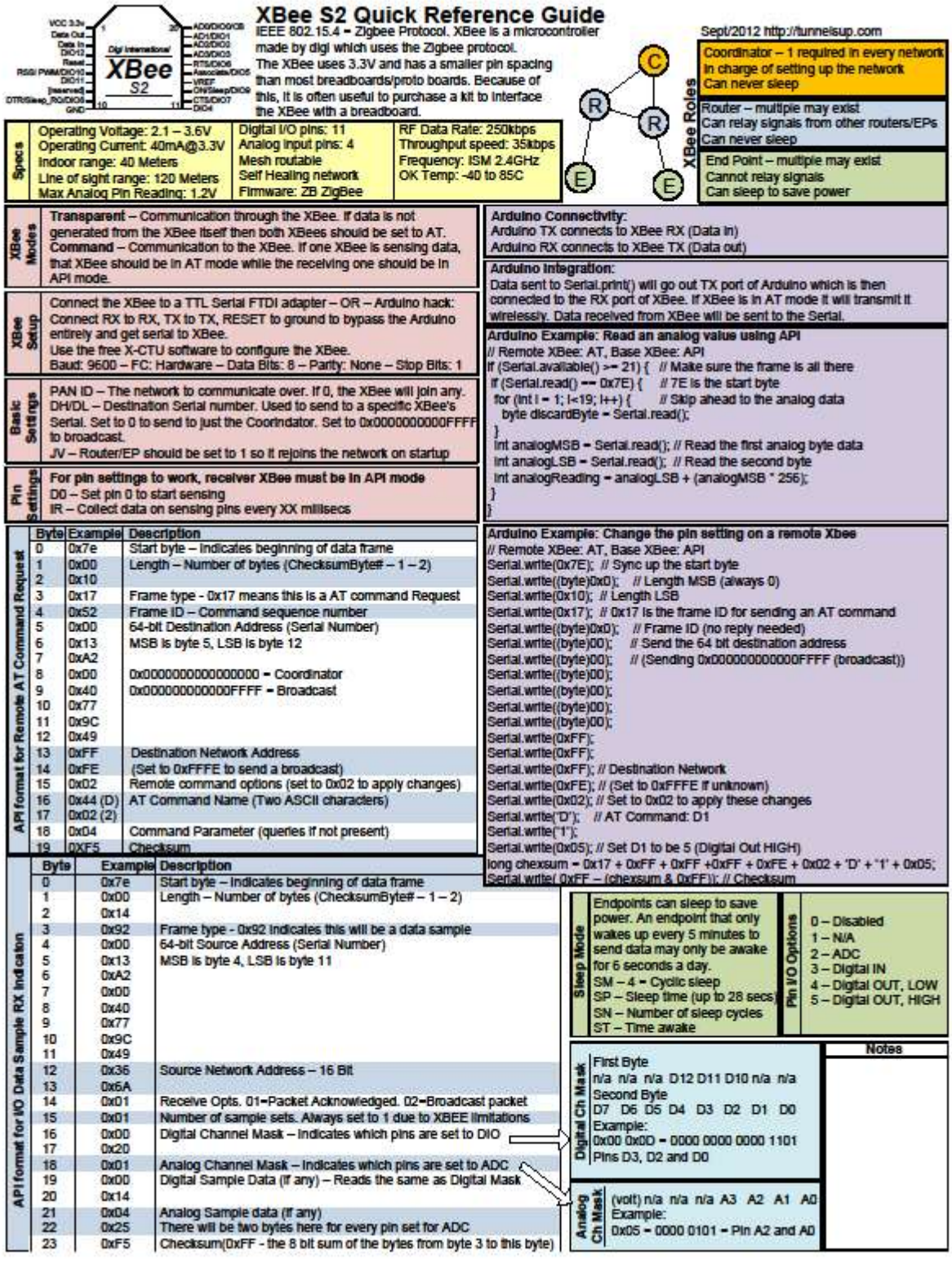
Figura 39: Esquema de pines de Arduino Uno R3

Veamos también las características generales de esta placa.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Tabla 4: Características Arduino Uno R3

7.1.3.- *ZigBee* XBee S2 Reference



7.2.- Configuraciones

7.2.1.- Configuración *ZigBee* mediante programa X-CTU

Para la configuración adecuada en modo API de los chips *ZigBee*, debe utilizarse el programa proporcionado por DIGI® X-CTU. Este puede observarse en la Figura 40. Para ello, se carga vía USB con el explorador de *ZigBee*, el *chip* correspondiente que se quiere configurar.

Lo primer es establecer el modo de comunicación en USB (pestaña "*PC settings*") y ahí seleccionamos el puerto donde tenemos conectado nuestro explorador.

En la pestaña "*Modem configuration*", para ver qué configuración tiene actualmente en dispositivo, se pulsa "*Read*" y carga la misma en pantalla. Sería conveniente actualizar las versiones ("*Download new versions*") para poder cargar el firmware más actual.

Los parámetros principales a configurar en caso de red estrella y modo API son:

- PAN ID: (*Personal Area Network*) todos los *chips* poseen un número de identificación único en la base del mismo. Esto conforma la *radio address.* La suma de este número de 16 bits + el PAN ID conforma una dirección de enrutamiento, similar a las IPs.
- Tipo de modelo de chip XBee: hay que seleccionar el modelo que se ha comprado.
- *Function set*: ahí se elige la función de este chip dentro de la red (ya hemos comentado que puede ser *Coordinator*, *Router* o *End-Device*. En el caso de la figura, lo vamos a configurar como coordinador.
- *Version*: debemos asegurarnos de que todos los dispositivos llevan el mismo *firmware*. En nuestro caso la versión 21A7.
- El código que deberemos emplear para nuestros programas y poder establecer la comunicación entre los diferentes dispositivos, es el que se forma entre los parámetros *Addresing*, *Serial Number High* y *Serial Number Low*. La dirección completa se da por la suma de ambas dos, en el caso nuestro: "13A20040ABBC0D".

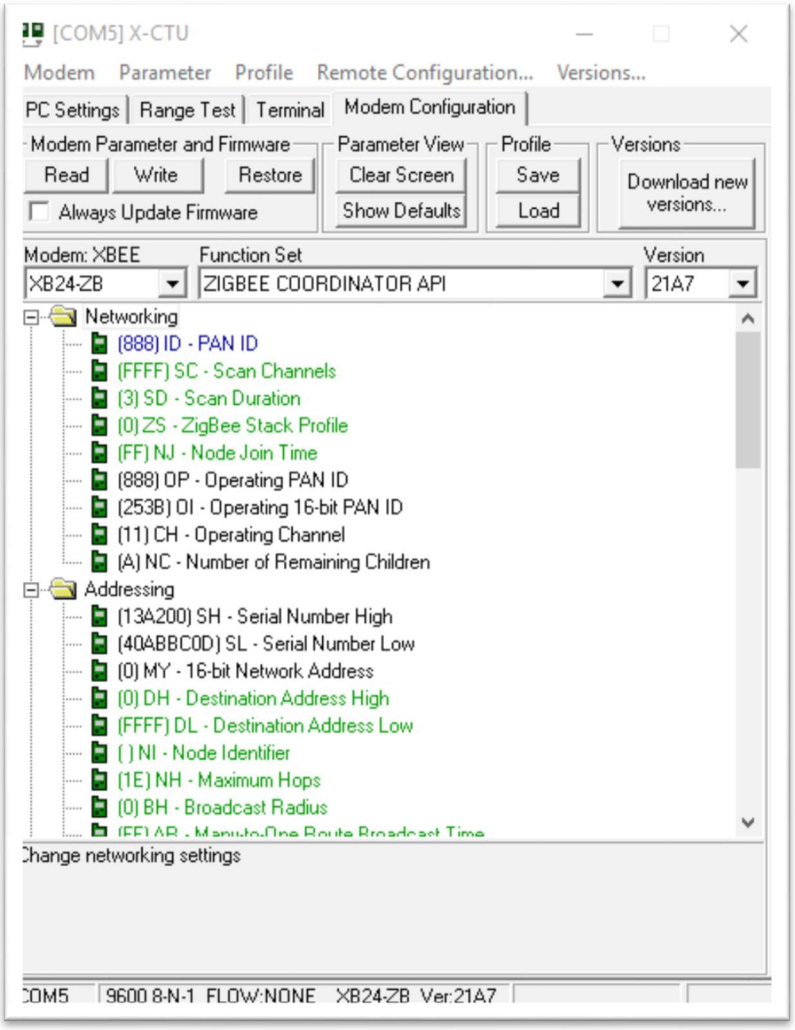


Figura 40: Programa de configuración ZigBee X-CTU

Para más información en este tema, pues esta configuración es larga y la explicación de los modos AT y API extensa, puede consultarse el capítulo 4 del libro "Building Wireless Sensor Networks" [6].

7.2.2.- Configuración de sensores en el prototipo Arduino Mega

Utilizando la aplicación Fritzing [20], se ha procedido a realizar un modelo esquemático del prototipo del proyecto en Arduino. Esta aplicación, una vez se tiene el diseño realizado, permite exportar un archivo PCB que se puede utilizar para realizar una placa con el diseño que se ha implementado.

Puede verse en la Figura 41 se puede observar el montaje de los sensores de luz (GY-30), de gas (MQ2), del PIR (detector de presencia), DHT11 (humedad y temperatura), de detección de nivel de humedad en suelo, la pantalla *LCD Nokia 5110* y un relé conectado a una bombilla. Asimismo podemos observar el *shield* para XBee y el chip XBee montados sobre la placa Arduino.

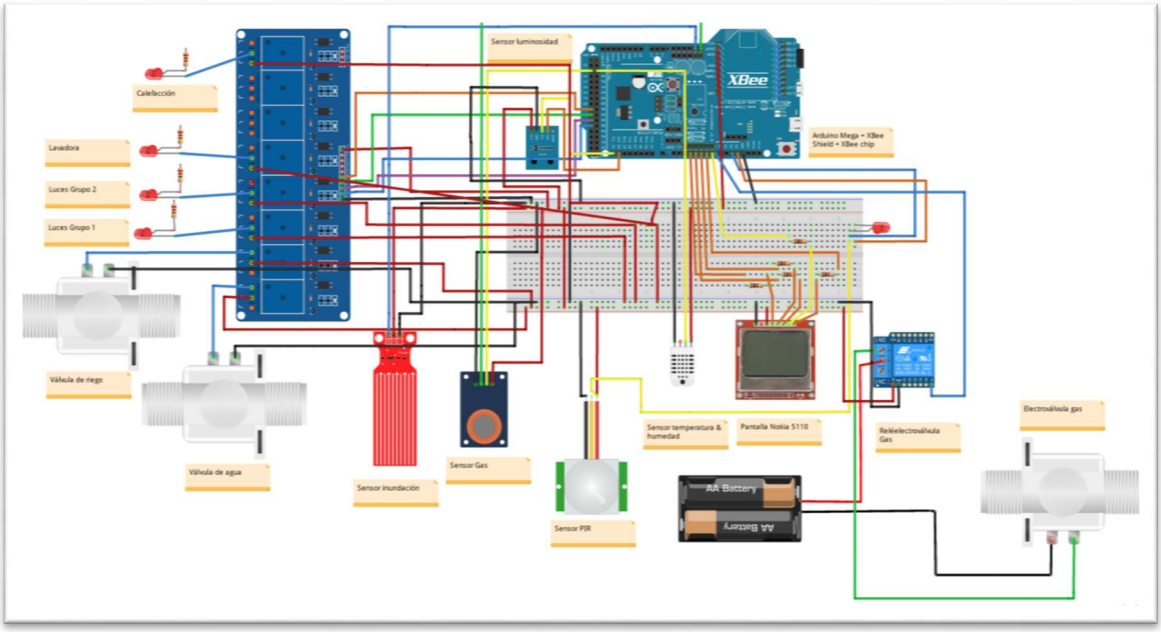


Figura 41: Esquema de montaje del prototipo Arduino

7.3.- Código

Dada la gran cantidad de código generada para la aplicación Android, no se va a incluir aquí, pero sí se considera relevante incluirlo para que pueda revisarse el trabajo realizado en el presente proyecto. Por ello se incluye el enlace al repositorio GitHub donde se han colgado el conjunto de los archivos fuente desarrollados, divididos en las carpetas de ficheros Android, PHP, servidor Python de Raspberry y código Arduino:

https://github.com/cceamanos/Domotic_House

7.4.- Pruebas

Dado el número de pruebas que se querían incluir, para no dilatar la memoria, se han incluido en este apartado

Identificador	CP05
Descripción	Montaje de un sensor inalámbrico de temperatura con un sensor LM35 analógico y un chip *ZigBee*, que tenga la capacidad de enviar la información cada cierto tiempo.
Entrada	La salida del sensor se redirige a la entrada A03 del *chip ZigBee*, y esta información es la que se enviará al programa del servidor central. Estos datos deberán ser tratados por el servidor para conseguir un valor en grados Celsius. La fórmula a utilizar será: Temperatura = ((valor decimal recibido * 1200/1024) – 500)/10
Salida	Debe verificarse que se recibe una temperatura adecuada en el servidor y esta es reflejada en el sistema.

Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose). Sensor de temperatura inalámbrico correctamente configurado.
Resultado	Correcto (se obtiene un valor coherente del sensor que es transmitido al servidor).

Dado lo especial del sensor que hemos configurado, explicaremos aquí como se ha realizado la configuración del mismo. Para ello, hay que comenzar utilizando un montaje como el de la figura Figura 42. Se alimenta con una fuente de alimentación para *protoboard* y una pila de 9v. En el montaje, la salida del sensor se comunica con el pin DI03 de *ZigBee*.

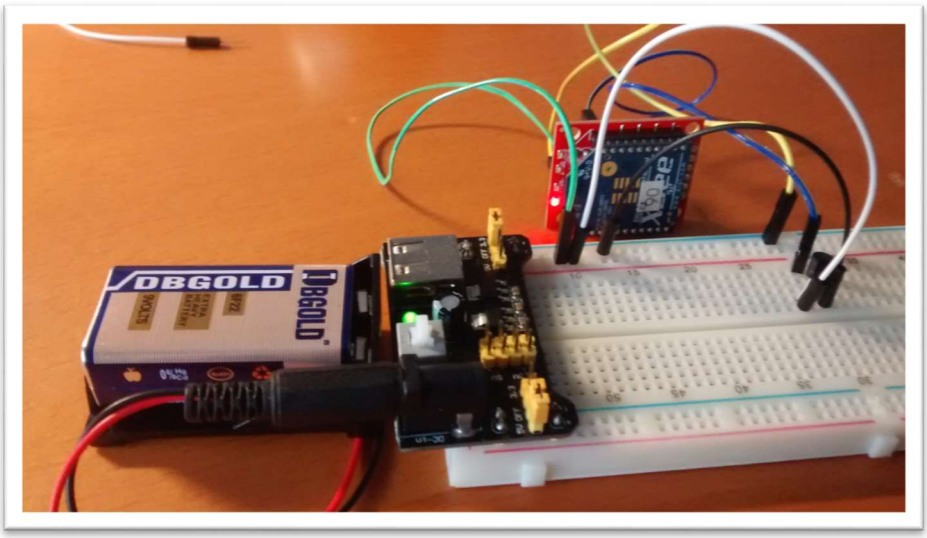


Figura 42 Sensor inalámbrico con ZigBee y LM35

Para configurar el chip con los datos necesarios, se ha utilizado la nueva versión del programa X-CTU (figura Figura *43*), asignando los valores siguientes:

- I/O Settings: D3:AD3/DIO3 Configuration ---------- 2 -ADC
- I/O Settings – I/O Sampling: IR: I/O Sampling Rate ---------- 3A98
- I/O Settings – I/O Sampling: V+:Supply Voltage Threshold ---- FFFF

Esta configuración indica que vamos a leer los datos de entrada por el pin D/A número 3, que será la información que vamos a remitir al servidor. Además, que vamos a enviarla cada 3A98 (15000 milisegundos, 15 segundos) y que enviamos el voltaje de referencia.

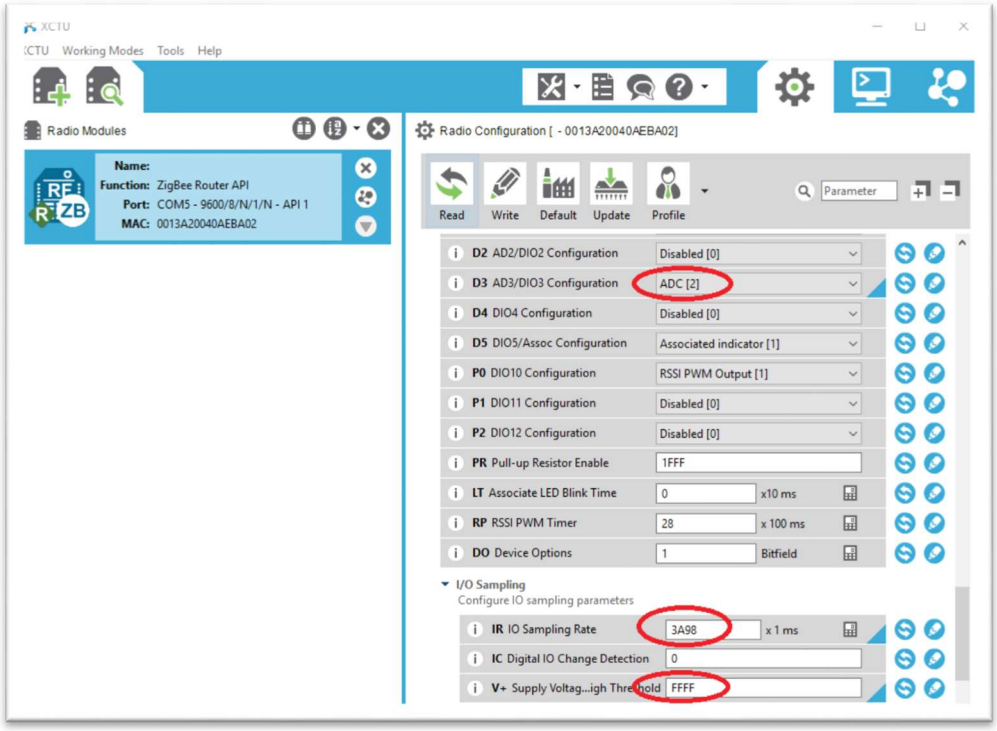


Figura 43: Parámetros configurados en X-CTU para sensor temperatura

Podemos ver en la consola de X-CTU el valor que nos está devolviendo este sensor (Figura 44). En este caso se está recibiendo el valor "0256" en hexadecimal, "598" en decimal. Utilizando la fórmula antes indicada en la prueba, obtenemos una temperatura de 20,07 °C, que es la que se tenía en la habitación en el momento de la prueba.

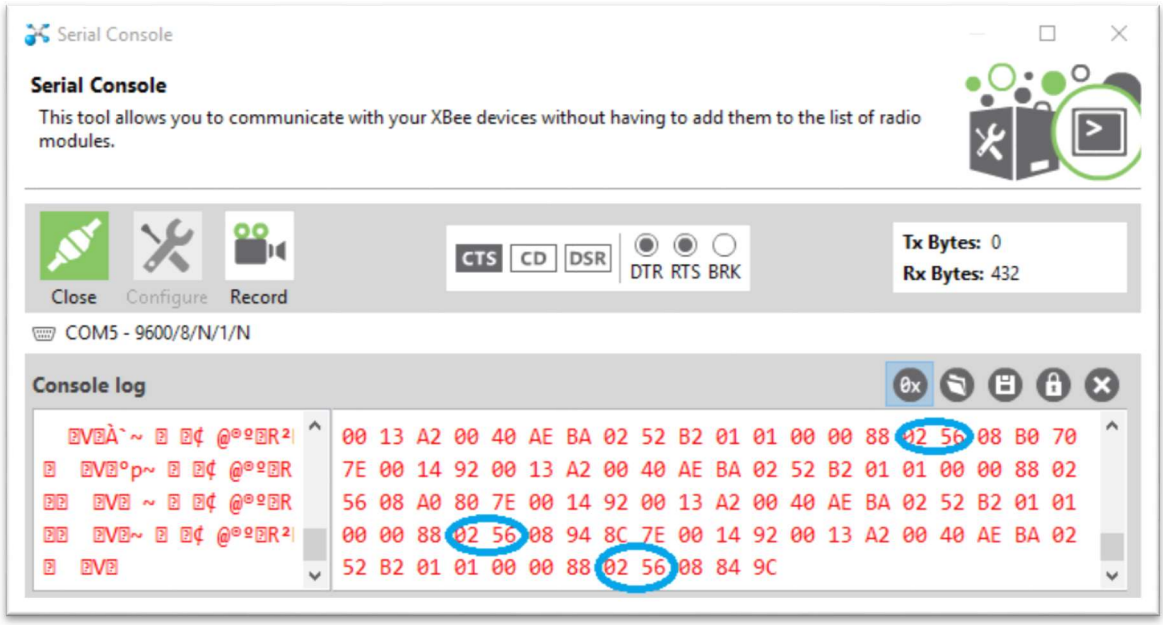


Figura 44: Consola del ZigBee servidor recibiendo las tramas

Identificador	CP06
Descripción	Verificación de envío de alarma ante un escape de agua (simulado) a la aplicación móvil.
Entrada	Se simula un escape de agua para que lo detecte el sistema, sumergiendo el sensor de inundación en un vaso.
Salida	Debe verificarse que en el móvil se recibe una alarma indica que existe un nivel anormal de agua en la habitación correspondiente.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose). Sensor de inundación correctamente instalado en Arduino
Resultado	Correcto: tras sumergir en el agua el sensor, se recibe una notificación en el móvil avisando de "Alerta de inundación en Cocina".

Identificador	CP07
Descripción	Verificación de envío de alerta de intruso una vez está la alarma de presencia conectada, a la aplicación móvil.
Entrada	Se simula la presencia de un intruso para que lo detecte el sistema, con y sin estar conectada la alarma, para verificar que el sistema envía la alarma cuando es preceptivo.
Salida	Debe verificarse que en el móvil se recibe una alarma indica que existe un intruso en la habitación correspondiente.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose). Sensor de presencia PIR correctamente instalado en Arduino
Resultado	Correcto: se hace pasar un objeto por delante con la alarma desconectada, no recibiendo nada, se conecta la alarma desde la aplicación móvil y se vuelve a repetir la operación, recibiendo una notificación en el móvil avisando de "Alerta de presencia en Cocina").

Identificador	CP08
Descripción	Verificación del funcionamiento del termostato de la calefacción desde la aplicación móvil: ver que se enciende y apaga cuando es preceptivo.
Entrada	Se simula el funcionamiento del termostato de la calefacción en la app móvil.
Salida	Debe verificarse que al cambiar en el móvil la temperatura y elevarla por encima de la temperatura actual de la habitación + histéresis, se activa la calefacción (estando en modo manual), y que una vez se baja la temperatura por debajo de la temperatura actual – histéresis, se apaga.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose). Sensor de temperatura correctamente instalado en Arduino
Resultado	Correcto: se sube la temperatura por encima de la t-actual + histéresis, y el sistema enciende el led que simula la caldera de la calefacción, e idénticamente cuando de baja la temperatura por debajo de t-actual menos la histéresis, apaga la caldera.

Identificador	CP09
Descripción	Verificación del funcionamiento de las luces de otra habitación diferente a la del prototipo principal.
Entrada	Se simula el funcionamiento de las luces del baño, conectando un led a un prototipo Arduino Uno diferente al del prototipo principal desde la app móvil.
Salida	Debe verificarse que al cambiar de dispositivo al que se envían las señales, estas llegan adecuadamente y la señal cumple su cometido.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino Uno conectada y en comunicación con servidor a través de *ZigBee*, en este caso otro chip (aplicaciones Arduino y Raspberry Pi ejecutándose).
Resultado	Correcto: se accede a habitación "Baño", se enciende y se apaga la luz que tiene configurada, y el led que las simula, se enciende y se apaga.

Identificador	CP10
Descripción	Verificación del funcionamiento de los eventos programables.
Entrada	Se programa el encendido de las luces de la cocina desde la app móvil para una hora concreta.
Salida	Debe verificarse que al llegar la hora del evento, se dispara la orden que está programada.
Requerimientos	Aplicación "DomoticHouse" instalada en un terminal móvil. Raspberry Pi conectada con el servidor LAMP levantado. IP de acceso a servidor compilada en la aplicación Android de forma que se pueda acceder al sistema correctamente. Placa Arduino del prototipo principal conectada y en comunicación con servidor a través de *ZigBee* (aplicaciones Arduino y Raspberry Pi ejecutándose).

Resultado	Correcto: se programa el evento para 5 minutos más adelante, y una vez se alcanza la hora, se ejecuta el switch del dispositivo.

8.- Bibliografía

Para la elaboración de este proyecto se ha contado con el apoyo de la siguiente bibliografía:

[1]. *A ZigBee wireless domotic system with Bluetooth interface. (Eurico Leite, Luis Varela, V. Fernão, A. J. Pires, João F. Martins)*

[2]. *Design and implementation of smart home energy management systems based on Zigbee. (Dae-Man Han and Jae-Hyun Lim, IEEE Members).*

[3]. *Smart home energy management system including renewable energy based on ZigBee and PLC. (Jinsoo Han, Chang-Sic Choi, Wan-Ki Park, Ilwoo Lee, and Sang-Ha Kim). Paper IEEE Transactions on Consumer Electronics, Vol. 60, No. 2, May 2014.*

[4]. *Towards an Open Framework for Home Automation Development. (Dang-Nhat Pham-Huu, Van-Hien Nguyen, Van-Anh Trinh, Van-Hieu Bui, and Hoang-Anh Pham) 2015 International Conference on Advanced Computing and Applications.*

[5]. *A ZigBee-Based Home Automation System. (Khusvinder Gill, Shuang-Hua Yang, Fang Yao and Xin Lu). Paper IEEE Transactions on Consumer Electronics, Vol. 55 nº 2. May 2009.*

[6]. *Building Wireless Sensor Networks. (Rober Faludi) A practical guide to the Zigbee mesh network protocol. Ed. O'Really Media 2011.*

[7]. Python para principiantes. http://librosweb.es/libro/python/

[8]. El gran libro de Android (3ª Edición) (Jesús Tomás Gironés) Ed. Marcombo Feb. 2013.

[9]. El gran libro de Android avanzado (Jesús Tomás, Vicente Carbonell, Carsten Vogt) Ed. Marcombo 2014

[10]. *Develop Android applications with Eclipse (IBM Paper Manual)* https://www.ibm.com/developerworks/opensource/tutorials/os-eclipse-android/

[11]. Arduino. Curso práctico de formación. (Óscar Torrente Artero). Editorial libros RC 2013.

[12]. Building automation. (Hermann Merz , Thomas Hansemann and Christof Hübner). Ed. Springer 2009.

[13]. Manual de programación Arduino (traducido José Manuel Ruiz Gutiérrez): http://lacajamakerspace.org/index.php/electronica/arduino/tutoriales?download=1:manual-de-programacion-de-arduino

[14]. *Programming your home. Automate with Arduino, Android and your Computer*

[15]. *Mile Riley 2012, The Pragmatic Programmers, LLC*.

[16]. *Great things, small packets. Your unofficial Raspberry Pi manual. (Christian Cawley):* http://www.makeuseof.com/tag/great-things-small-package-your-unofficial-raspberry-pi-manual/

[17]. *Raspberry Pi Automation with Arduino. Andrew K. Dennis. Ed. Packt Publishing 2013*

[18]. *Smart Home Automation with Linux and Raspberry Pi. Steven Goodwin. Ed. Apress 2013.*

9.- Referencias Web

Para la elaboración de este proyecto se ha contado con las siguientes referencias web:

[1]. Web de *ZigBee*: http://www.domodesk.com/zigbee.htm
[2]. Web WeeBee: http://webeelife.com
[3]. Información Arduino Uno: http://arduino.cc/en/Main/ArduinoBoardUno
[4]. Información Arduino Mega: http://arduino.cc/en/Main/arduinoBoardMega2560
[5]. Productos Arduino: http://arduino.cc/en/Main/Products
[6]. DTH11: http://issuu.com/rduinostar/docs/dht11_datasheet
[7]. DHT21: http://www.electrodragon.com/w/images/6/6f/DHT21.pdf
[8]. MQ-2: http://www.haoyuelectronics.com/Attachment/MQ-2/MQ-2.pdf
[9]. HC-SR501: http://electronilab.co/wp-content/uploads/2013/12/HC-SR501.pdf
[10]. GY-30: http://rohmfs.rohm.com/en/products/databook/datasheet/ic/sensor/light/bh1750fvi-e.pdf
[11]. Funduino: http://techamc.es/ARDUINO/SENSORES/SENSOR%20NIVEL%20AGUA/SENSOR_NIVEL_AGUA_ARDUINO.html
[12]. Placa de relés: http://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino/
[13]. Sensor efecto Hall: http://teslabem.com/modulo-sensor-magnetico-efecto-hall-ky.html
[14]. Li Jun, Yang Qing estudio vulnerabilidades y hackeo *ZigBee* https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Li-Jun-Yang-Qing-I-AM-A-NEWBIE-YET-I-CAN-HACK-ZIGBEE.pdf
[15]. *ZigBee* Exploited: The good, the bad and the ugly, Tobías Zillner http://www.sicherheitsforschung-magdeburg.de/uploads/journal/MJS_045_Zillner_ZigBee.pdf
[16]. Bitdefender: Risks in the connected Home: https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiqjOa44pzSAhVCvRoKHcnPAzQQFggaMAA&url=https%3A%2F%2Fdownload.bitdefender.com%2Fresources%2Ffiles%2FNews%2FCaseStudies%2Fstudy%2F87%2FBitdefender-2016-IoT-A4-en-EN-web.pdf&usg=AFQjCNHF890ZTZzzh91kPgEVP9VbVZBMsw&sig2=_UqFT_O_oyRUngZoMjiinw&bvm=bv.147448319,d.bGs
[17]. Precios ZWave: http://zwave.es/controllers/zwavegateways
[18]. Precios Arduino: http://tienda.bricogeek.com/
[19]. Precios X10: https://latiendadedomotica.com/100-equipos-especiales-x10
[20]. Fritzing: http://fritzing.org/home/
[21]. Waspmote *ZigBee*: Networking Guide: http://www.libelium.com/downloads/documentation/waspmote-zigbee-networking_guide.pdf

10.- Índice de figuras

11.- Índice de tablas