Gestión centralizada de dispositivos electrónicos *low-cost* para su uso en entornos domésticos

PFC-Administración de redes y sistemas operativos

Alumno: Angel Bueno Pardo

Consultor: José Manuel Castillo Pedrosa

Junio de 2014

Resumen

Hoy en día, en la mayoría de los hogares existe gran cantidad de elementos electrónicos de bajo coste que pueden ser gestionados de forma autónoma e incluso a distancia, o bien permiten su control a través de teléfonos inteligentes. Estos componentes no disponen de ningún elemento de gestión centralizada que permita agregarles funcionalidades adicionales, para que puedan trabajar de forma coordinada. Además, tampoco usan ningún protocolo de comunicaciones estandarizado que permita su integración con otros interfaces.

El presente proyecto pretende usar equipos ya existentes en entornos domésticos con funcionalidades limitadas (administración autónoma) para integrarlos y poder gestionarlos de forma centralizada con una interfaz amigable. El beneficio que aporta este entorno es poder usar la tecnología para mejorar la eficiencia energética de los hogares con un bajo coste, así como el confort.

Para lograr el objetivo marcado, será necesario analizar y comparar las diferentes opciones existentes en el mercado de herramientas hardware y software que permitan integrar los elementos existentes y a un bajo coste. Una vez evaluadas estos elementos, será el momento de diseñar la integración de los diferentes componentes y por tanto de conocer su medio de comunicación -descifrándolo si fuera necesario- y así poder interactuar con él y lograr un mejor acoplamiento.

Por tanto, para abordar este proyecto es necesario estudiar cómo interactuar con los elementos que se van a gestionar, analizar un equipo en el que instalar las herramientas necesarias para dicha comunicación, investigar un software que realice dicha comunicación, con capacidades de configuración y acceso a través de una interfaz amigable para el ser humano y finalmente llevar a cabo la integración de todos los elementos.

Índice de contenido

Índice de tablas

Índice de ilustraciones

1 Introducción

1.1 Justificación

En todos los hogares existen dispositivos electrónicos que en su mayoría funcionan de forma aislada y no permiten, salvo con accesorios del mismo fabricante, trabajar de forma coordinada. En caso de querer llevar a cabo una instalación de domótica, sí que existen estándares en el mercado como son KNX o X10, sobre los que hay desarrollados diversos elementos.

Estos estándares implican un rediseño de la vivienda, puesto que no permiten en la mayoría de los casos reaprovechar los elementos electrónicos existentes, además de que requieren de una instalación de cableado específica que implican un elevado coste y complejidad de instalación.

Con estos antecedentes, este proyecto justifica el desarrollo e investigación de los elementos existentes en una vivienda, que no disponen de un protocolo normalizado de comunicaciones, para su integración y control mediante un equipo que les dote de la inteligencia suficiente para mejorar su eficiencia.

Puesto que los elementos que se pretende integrar son de bajo coste, no resulta rentable (salvo para consumo propio o con una economía de escala) desarrollar interfaces de comunicación para estos dispositivos, por tanto, se considera que no es una funcionalidad sensible de ser implementada por ningún fabricante en el mercado y aquí es donde entra en juego el presente PFC, con el que poder abordar esta necesidad.

1.2 Objetivos del proyecto

El objetivo de este proyecto es la gestión centralizada de diversos elementos electrónicos domésticos, para la mejora de la eficiencia y funcionalidad de los mismos. De forma global, se puede indicar que la principal meta es la mejora de la eficiencia energética a través del control automático de los dispositivos eléctricos de una vivienda, haciendo uso de elementos carentes de una gestión integrada.

1.2.1 Entradas de información en el sistema

1. Dentro de la mejora de la eficiencia energética, es necesario conocer el **consumo eléctrico** que dispone la vivienda. A partir de los datos resultantes se podrá conocer y por tanto evaluar los hábitos de consumo y cómo mejorarlos para que reporten un menor gasto manteniendo un mismo estado de confort dentro de la vivienda. Por tanto el sistema de domótica ha de conocer el consumo eléctrico en cada momento.

2. Una vez que se conoce el gasto energético, se puede optimizar el gasto eléctrico derivando

consumos a franjas horarias donde el coste es inferior, siempre y cuando estas actividades no dependan de otros factores más importantes como la presencia de los inquilinos. Un ejemplo claro se encuentra en la lavadora, electrodoméstico con un alto consumo eléctrico que se puede activar en los slots de tiempo de menor coste. Para llevar a cabo esta tarea es necesario que el sistema de domótica conozca el **coste eléctrico** de cada franja horaria.

3. El sistema domótico también debe llevar un **control horario**, para poder coordinar la ejecución de las diferentes tareas acorde a la actividad humana.
4. La temperatura se ve afectada por varios elementos y es necesario que el sistema de domótica conozca la **temperatura** en cada momento para poder gestionarlos de forma adecuada. En consecuencia, conocer la temperatura será uno de los objetivos de este sistema.
5. Así mismo, es importante automatizar el deslastrado de elementos energéticos de confort que carecen de sentido en caso de ausencia de inquilinos, como las luces de la vivienda, o el sistema de calefacción, por tanto, en caso el sistema de domótica ha de controlar la **presencia** o ausencia de los inquilinos dentro de la vivienda, y desactivar aquellos elementos eléctricos que no resulten necesarios en ausencia de personas.

1.2.2 Dispositivos a controlar

Los elementos que gestionará el sistema de domótica afectan a los vectores energéticos de la vivienda (temperatura, luz) y de ocio (que requieran energía). Los que se podrán gestionar a partir de la puesta en marcha de este proyecto son:

1. **Temperatura**
    1. **Radiadores**. Uno de los objetivos es poder controlar el encendido y apagado de radiadores de bajo coste, carentes de todo tipo de control de temperatura o programación horaria. Por medio del sistema que se implante, es necesario poder encender bajo demanda los radiadores.
    2. **Persianas**. Otro elemento importante en la gestión térmica y lumínica dentro de una vivienda son las persianas, que mantienen la temperatura interior estable frente a las oscilaciones térmicas exteriores. En consecuencia, es otro objetivo de este proyecto poder subir o bajar las persianas de forma centralizada y así aprovechar la climatología externa en la gestión térmica de la vivienda.
    3. **Ventilador de techo.** Si bien este dispositivo no modifica en sí mismo la temperatura de la habitación, sí que altera la sensación térmica en ambientes calurosos, mejorando sensiblemente el ambiente.

2. **Luz**. Otro elemento energético ampliamente usado dentro de una vivienda es la luz. Este proyecto debe gestionar el encendido, apagado y cantidad lumínica de las principales estancias de una vivienda.
3. **Lavadora**. Como ya se ha indicado, la lavadora es uno de los electrodomésticos con mayor consumo eléctrico de la vivienda, por lo que se deberá controlar para que su actividad quede relegada a las horas de menor coste monetario de la electricidad.
4. **Ocio**. Como elemento dedicado al ocio mayoritario dentro de los hogares se encuentra la **televisión**, por tanto, el sistema de domótica propuesto deberá gestionarla de forma integrada con el resto de elementos (persianas, luz, etc.) para crear un área de confort en momentos de ocio.

1.2.3 Reglas de negocio

1. Se deben quedar registrados los eventos de cada elemento controlado. De esta forma se podrá llevar un control del sistema y también simular la presencia dentro del período estival
2. Los radiadores se deben poder encender de forma remota desde el sistema de domótica mediante la interacción del usuario, o bien atendiendo a los parámetros de temperatura y cercanía a la vivienda.
3. Las persianas se deben poder subir/bajar de forma centralizada mediante una intervención manual, o bien programada por el usuario atendiendo a criterios horarios.
4. La lavadora se debe poder encender de forma manual, o bien atendiendo a criterios de consumo energético, de forma que se active en la franja horaria de menor coste monetario.
5. Las luces se deben poder encender, apagar y regular de forma centralizada.
6. Se debe establecer una serie de parámetros para acondicionar el salón según la situación, de forma que la persiana se baje, la luz baje de intensidad y la televisión se encienda con sólo pulsar un botón de la herramienta de domótica, de igual forma se debe deshacer el escenario anterior con sólo un botón.
7. El entorno se debe poder usar desde dispositivos móviles y equipos con navegador.

1.3 Enfoque y metodología

Los pasos necesarios para abordar este proyecto son análisis y toma de decisión sobre la solución software que cubra las necesidades del proyecto; análisis y toma de decisión sobre la solución hardware; diseño, implantación e integración de los diversos elementos y finalmente la verificación de que los objetivos se han cumplido. Este resumen se detalla de la siguiente forma:

1. Investigación de soluciones existentes. Es necesario conocer las herramientas que ya existen en el mercado, para decidir si se usa alguna de ellas (en caso de existir y no tener coste) o bien se opta por un desarrollo a medida.
   1. Investigar los desarrollos de software libre sobre domótica.
   2. Investigar los gateways de protocolos como KNX o X10.
   3. Investigar si existen frameworks para desarrollos de domótica.

2. Investigación de los equipos a gestionar. En el punto de partida hay que conocer los elementos que van a integrar la solución y como se comunican entre ellos, para poder interactuar con ellos.
   1. Investigar equipos de bajo coste que hospeden la solución de domótica. Estudiar los equipos que permiten ejecutar código y disponen de una interfaz GPIO. De entre todos ellos hay que estudiar las opciones más económicas que se ajusten a las soluciones encontradas en la tarea 1.
   2. Investigar equipos que doten de un control telemático a aquellos dispositivos que sean gestionados de forma autónoma y local, y que además permitan una fácil integración. En el caso de elementos que no tengan control remoto como puede ser los radiadores, es necesario poder accionarlos de forma telemática y este control lo ofrecen por ejemplo los enchufes accionados por radiofrecuencia.
   3. Investigar y desencriptar los protocolos de comunicaciones de cada uno de los equipos que puedan ser gestionados de forma telemática. A modo de ejemplo, aquellos dispositivos que puedan ser accionados mediante mandos a distancia, se intentará descifrar la comunicación para que sean gestionados por “nuestro propio mando a distancia”.
   4. Investigar sondas de datos. Temperatura, geoposicionamiento y consumo eléctrico. Para que el sistema de domótica pueda llevar a cabo acciones en función estos datos. Por ejemplo, encender la lavadora en la franja horaria en la que el coste eléctrico es menor.

3. Implantar la solución.
   1. Instalar y configurar el equipo que alojará la solución de domótica.
   2. Instalar y configurar los periféricos necesarios en el servidor de domótica para poder gestionar los equipos de forma telemática.
   3. Desarrollo/adaptación de la interfaz de gestión de domótica.
   4. Fase de pruebas de los equipos gestionados. En este punto se hará un proceso de verificación del correcto funcionamiento de cada uno de los elementos gestionados por el sistema independientemente de la interfaz de gestión.
      1. Encendido y apagado de radiadores.
      2. Encendido y apagado de luces.
      3. Subida y bajada de persianas.
      4. Ventilador de techo.
      5. Televisión.
      6. Lavadora.
      7. Temperatura.
      8. Detección de presencia.

9. Control horario.
10. Posición solar.

5. Fase de pruebas de la lógica de negocio. En este apartado hay que llevar a cabo la verificación de que la lógica que aporta el sistema funciona de forma adecuada.
   1. Test de pruebas para encendido y apagado de radiadores (accionados con presencia y temperatura por debajo de un umbral).
   2. Test de pruebas para gestión de luces (que no se puedan accionar sin la detección de presencia).
   3. Test de pruebas para persianas (que se suban o bajen acorde a la posición solar).
   4. Test de pruebas para ventilador de techo (que se accione de forma remota sólo en verano).
   5. Test de pruebas para televisión (que no se pueda accionar sin la detección de presencia).
   6. Test de pruebas para lavadora (que se accione si se solicita en la hora de menor coste eléctrico).
6. Fase de pruebas de las interfaces de gestión. Test de la interfaz de gestión en la que será necesario probar su correcto funcionamiento para cada una de las plataforma (PC, dispositivo móvil, etc.).

1.4 Planificación

	Nombre	Duracion	Notas	Inicio	Terminado	Predecesores
1	**Investigación de soluciones existentes**	**4 days**		**17/03/14 8:00**	**20/03/14 17:00**	
2	Investigar los desarrollos de software libre sobre domótica	4 days		17/03/14 8:00	20/03/14 17:00	
3	Investigar los gateways de protocolos como KNX o X10	4 days		17/03/14 8:00	20/03/14 17:00	
4	Investigar si existen frameworks para desarrollos de domótica	4 days		17/03/14 8:00	20/03/14 17:00	
5	**Investigación de los equipos a gestionar**	**18 days**		**21/03/14 8:00**	**15/04/14 17:00**	**1**
6	Investigar equipos de bajo coste que hospeden la solución de domótica	4 days		21/03/14 8:00	26/03/14 17:00	
7	Investigar equipos para control telemático	4 days		27/03/14 8:00	1/04/14 17:00	6
8	Investigar y desencriptar los protocolos de comunicaciones	10 days		2/04/14 8:00	15/04/14 17:00	7
9	Investigar sondas de datos	10 days		2/04/14 8:00	15/04/14 17:00	7
10	**Implantar la solución**	**28 days**		**16/04/14 8:00**	**23/05/14 17:00**	**5**
11	Instalar y configurar el equipo que alojará la solución de domótica	8 days	Presentación PEC2 el 17/04/2014	16/04/14 8:00	25/04/14 17:00	
12	Instalar y configurar los periféricos	8 days		28/04/14 8:00	7/05/14 17:00	11
13	Desarrollo/Instalación de la interfaz de gestión	10 days		8/05/14 8:00	21/05/14 17:00	12
14	**Fase de pruebas de los equipos gestionados**	**2 days**	**Presentación PEC3 el 23/05/2014**	**22/05/14 8:00**	**23/05/14 17:00**	**13**
15	Encendido y apagado de radiadores.	2 days		22/05/14 8:00	23/05/14 17:00	
16	Encendido y apagado de luces.	2 days		22/05/14 8:00	23/05/14 17:00	
17	Subida y bajada de persianas.	2 days		22/05/14 8:00	23/05/14 17:00	
18	Ventilador de techo.	2 days		22/05/14 8:00	23/05/14 17:00	
19	Televisión.	2 days		22/05/14 8:00	23/05/14 17:00	
20	Lavadora.	2 days		22/05/14 8:00	23/05/14 17:00	
21	Temperatura.	2 days		22/05/14 8:00	23/05/14 17:00	
22	Detección de presencia.	2 days		22/05/14 8:00	23/05/14 17:00	
23	Control horario.	2 days		22/05/14 8:00	23/05/14 17:00	
24	Posición solar	2 days		22/05/14 8:00	23/05/14 17:00	
25	**Fase de pruebas de la lógica de la domótica**	**3 days**		**26/05/14 8:00**	**28/05/14 17:00**	**10**
26	Test de pruebas para encendido y apagado de radiadores	3 days		26/05/14 8:00	28/05/14 17:00	
27	Test de pruebas para gestión de luces	3 days		26/05/14 8:00	28/05/14 17:00	
28	Test de pruebas para persianas	3 days		26/05/14 8:00	28/05/14 17:00	
29	Test de pruebas para ventilador de techo	3 days		26/05/14 8:00	28/05/14 17:00	
30	Test de pruebas para televisión	3 days		26/05/14 8:00	28/05/14 17:00	
31	Test de pruebas para lavadora	3 days		26/05/14 8:00	28/05/14 17:00	
32	Fase de pruebas de las interfaces de gestión	3 days		29/05/14 8:00	2/06/14 17:00	25
33	Documentación y revisión del trabajo	7 days	Entrega final el 11/06/2014	3/06/14 8:00	11/06/14 17:00	32

1.5 Productos obtenidos.

El resultado de este proyecto es un sistema compuesto por un equipo que tiene instalado un software de domótica y que gestiona enchufes por radio-control, luces a través de ethernet, motores de persianas por radio-control, recibe señales de RF con el consumo energético de la vivienda en la que se encuentra y permite accionar un ventilador de techo por RC.

Este sistema tiene una interfaz de gestión cómoda de usar para un usuario sin conocimientos de informática, y será accesible desde cualquier navegador web con acceso a internet o bien desde una aplicación específica para *smartphone*.

Dado que hay varios elementos que usan señales de radio para su gestión y el fabricante de los mismos no ofrece elementos software para su gestión, será necesario desarrollar en lenguaje C pequeñas aplicaciones que permitan interactuar con ellos.

El destino de esta solución es un entorno doméstico que quiera incrementar su eficiencia energética y confort, en el que se disponga de elementos carentes de funcionalidades de domotización, y no se quiera abordar una inversión en la renovación de los elementos existentes.

1.6 Estructura del documento.

Los siguientes apartados de este documento abordan la solución técnica del proyecto.

Primero, en el capítulo 2 “Estudio de mercado y equipos a gestionar” se hace un estudio de las herramientas existentes en el mercado relacionadas con la domótica que carecen de coste de licenciamiento y se escoge la solución que mejor se adapta a las necesidades del proyecto. Posteriormente se analiza y escoge el dispositivo que puede cubrir las necesidades del sistema. También se estudia a bajo nivel los dispositivos de comunicaciones que usa cada elemento a gestionar, analizando los protocolos utilizados en cada caso. Por último, se estudia como obtener la posición del propietario de la vivienda para su posterior integración en el sistema.

El siguiente capítulo “Desarrollo e implementación”, aborda los pasos necesarios para la instalación del equipo escogido con el software seleccionado, así como su configuración y las aplicaciones necesarias para integrar estos elementos con los protocolos de comunicaciones diseccionados en el anterior capítulo. También se analizan los pasos requeridos para instalar los componentes hardware que hacen de enlace entre los diversos elementos a controlar y el equipo de gestión.

En apartado “Fase de pruebas” verifica que el control de cada uno de los elementos, y las reglas de negocio solicitadas en los objetivos se cumplen adecuadamente.

Por último, el apartado de conclusiones muestra un resumen del proyecto, donde se analizan los objetivos, que posibilidad de mejorar el proyecto existen, y qué problemas se han detectado.

2 Estudio de mercado y equipos a gestionar

2.1 Soluciones software de gestión

El proyecto requiere que la solución no tenga coste adicional de licencias, por tanto se hace necesario el uso de herramientas basadas en software libre. En la actualidad existen bastantes soluciones que sirven de base para abordar los requisitos de este proyecto. Las características de cada una de ellas son:

2.1.1 FreeDomotic[1]

- Multiplataforma
- Distribuido y escalable
- Modular y extensible
- APIs multilenguaje
- Múltiples interfaces de acceso
- Independiente del hardware
- Basado en eventos
- Editor gráfico
- Registro de eventos
- Multilenguaje

2.1.2 OpenHomeAutomation[2]

- Funciona sobre sistema operativo windows
- Personalización de la interfaz
- Lenguaje de scripting sencillo con Windows PowerShell
- Creciente lista de complementos que funcionan con una variedad de dispositivos y servicios
- Control de diferentes tecnologías de encendido como Z-Wave, X10 e Insteon
- Obtiene el tiempo de cualquier parte del mundo desde Weather Underground
- Envía un correo electrónico con alertas de eventos
- Controla el sistema a través de mensajería instantánea XMPP como GoogleTalk.

- Detecta la presencia de dispositivos Bluetooth como teléfonos para ayudar a determinar quién se encuentra en las instalaciones
- Monitoriza varios tipos de sensores como movimientos, contactos o fotocéculas de diversas tecnologías como X10.
- Se puede comunicar y controlar mediante técnicas como reconocimiento de voz y texto a voz
- Controla tu termostato con Z-Wave o RS-485
- Web service RESTful
- Ejecuta comandos de windows desde un intérprete de comandos
- Controla y recibe actualizaciones de XBMC y Windows MediaCenter
- Controla tu servidore Squeezebox
- Monitoriza tus dispositivos de red por lo que sabe cuando se encuentran operativos
- Monitoriza el consumo eléctrico de dispositivos individualescon un Kill-A-Watt modificado.
- Usa etiquetas RFID para identificar a los ocupantes
- Controla servos con Phidgets
- Envía comandos a un servidor SSH

2.1.3 OpenHab[3]

- Está diseñado para ser independiente del fabricante, del hardware y de los protocolos
- Se puede ejecutar en cualquier dispositivo que soporte aplicaciones JVM.
- Permite integrar diferentes tecnologías de domótica en una sola.
- Tiene un potente motor de reglas para las necesidades de automatización.
- Viene con diferentes interfaces basadas en web, así como clientes nativos para iOS y Android.
- Es totalmente open source.
- Está mantenido por una apasionada y creciente comunidad.
- Es fácil ampliar e integrar nuevos sistemas y dispositivos.
- Ofrece APIs de integración con otros sistemas.

Por otra parte, la lista de funcionalidades que incorpora es tan elevada, que desde aquí se remite a su web para la consulta detallada de las mismas en https://github.com/openhab/openhab/wiki/Release-Notes-1.4. Entre las más importantes que aplican a este proyecto se encuentran:

- MQTTitude binding for presence detection. (v. 1.4.0)
- MiLight Binding (v. 1.3.0)
- Add support for new RGB-W bulbs to MiLight binding. (v. 1.4.0)
- Make actions modular (v. 1.3.0)
- added new Samsung TV binding (v. 1.2.0)

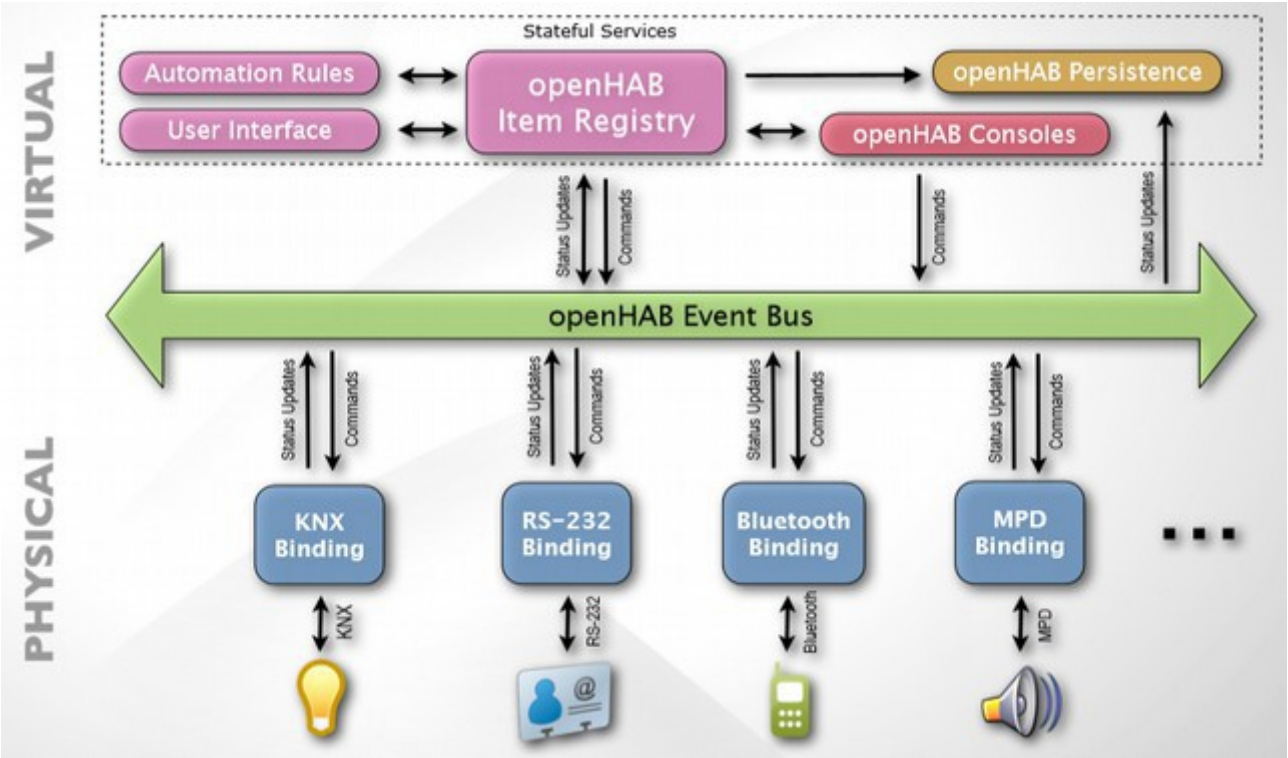

Ilustración 1: Arquitectura de OpenHAB

2.1.4 Solución escogida

De las tres opciones evaluadas, la que más se ajusta a los requerimientos deseados es **OpenHAB**, dado que puede correr en cualquier plataforma que soporte Java, es independiente de los fabricantes, del hardware y de los protocolos e incorpora conectores para los principales elementos que conforman el proyecto. Además, dispone de interfaces de gestión web, y aplicaciones nativas para los principales sistemas operativos de smartphone actuales.

Esta solución, dado que correo sobre plataforma java, permite un amplio espectro de dispositivos sobre los que se puede instalar. Por otro lado, al ser una plataforma enfocada a la independencia de fabricantes y protocolos, permitirá una mejor integración de los dispositivos que se pretende usar en este proyecto.

2.2 Dispositivos necesarios para implementar el sistema

Para ejecutar la aplicación de domótica, es necesario un dispositivo que permita la ejecución de código java, y se pretende que el coste sea el mínimo posible, por lo que se ha decidido optar por un equipo con procesador ARM y 512MB de memoria RAM, del fabricante Raspberry Pi[4], con sistema operativo linux (Arch linux) y máquina virtual java con hard float (permite la ejecución de instrucciones en coma flotante directamente en el procesador, sin necesidad de emularlas) que resulta ser mucho más rápido que otras máquinas virtuales, y dado que openHAB funciona en una máquina virtual java, esta solución será más eficiente que otras.

Por otro lado, es necesario disponer de elementos de comunicación con los radiadores, las persianas, los ventiladores y la lavadora. Por otro lado, es necesario un equipo medidor de consumo eléctrico y los elementos de comunicación con el mismo, y también es requerido contar con termómetros radiocontrolados.

En cuanto al resto de dispositivos; por un lado, las luces se gestionarán con una pasarela que permite su comunicación mediante una red ethernet; por otra parte, la televisión también se gestionará mediante ethernet; finalmente, la presencia en la vivienda se determinará mediante el GPS de los smartphone de los inquilinos y la aplicación owntrack, que enviarán su posición al equipo de domótica mediante tcp/ip y un broker de comunicaciones open source llamado mqtt. De esta forma se limita el uso de datos de posicionamiento por parte de terceros.

La comunicación entre los radiadores se hará mediante enchufes controlados por radiofrecuencia[5].

Ilustración 2: Enchufes controlados por RF

La señal de funcionamiento de estos equipos es de 433MHz, por lo que se colocará un emisor[6] y un receptor[7] de esta frecuencia en el equipo de domótica. De esta forma (emisor y receptor) se permitirá capturar las señales que se envíen a los diversos enchufes de forma autónoma, y poder registrarlas en el sistema de domótica, permitiendo que el sistema funcione de forma plenamente integrada. Este mismo sistema se utilizará para el control de la lavadora, pero con enchufes que soporten una carga más elevada.

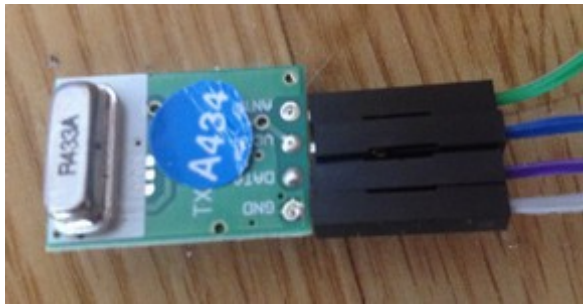

Ilustración 3: Emisor de 433.92MHz

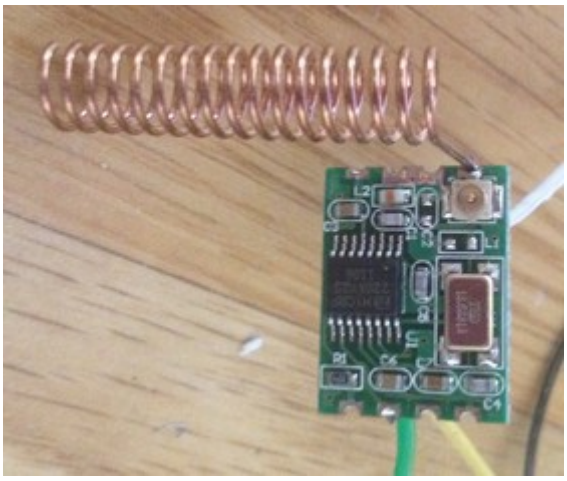

Ilustración 4: Receptor de 433.92MHz

En cuanto a los ventiladores, ahora mismo disponen de un mando a distancia por RF, que funciona a 315MHz, por lo que de igual forma que en el caso anterior, se colocará un emisor y un receptor de RF de 315MHz en el equipo de domótica.

El equipo de control de consumo eléctrico es del fabricante MIEO, modelo HA102[8], que envía los datos de consumo a un display mediante RF, a 433MHz, por lo que el mismo receptor usado para los enchufes radio-controlados servirá para obtener la señal de consumo eléctrico.

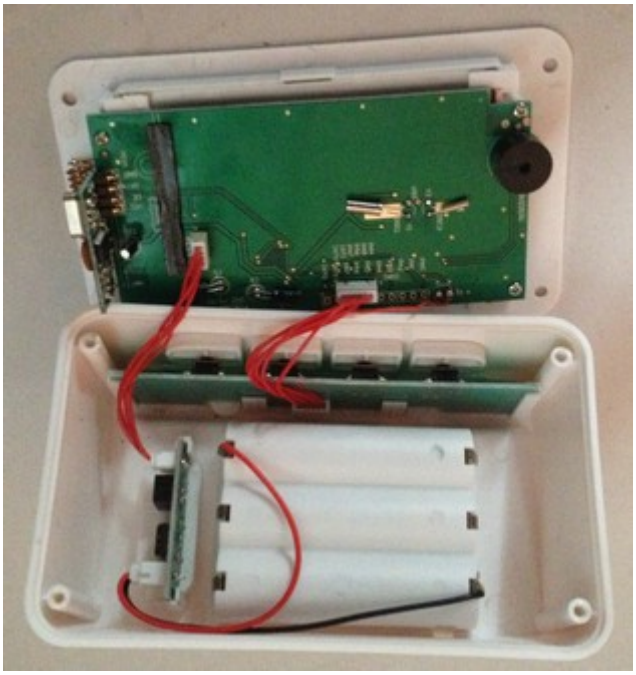

Ilustración 5: Interior de la consola del medidor de corriente

Para obtener la temperatura de las diferentes estancias se usará un controlador arduino y una sonda de temperatura DHT22, junto con una red Xbee, en la que transmitir la señal hacia el sistema de domótica.

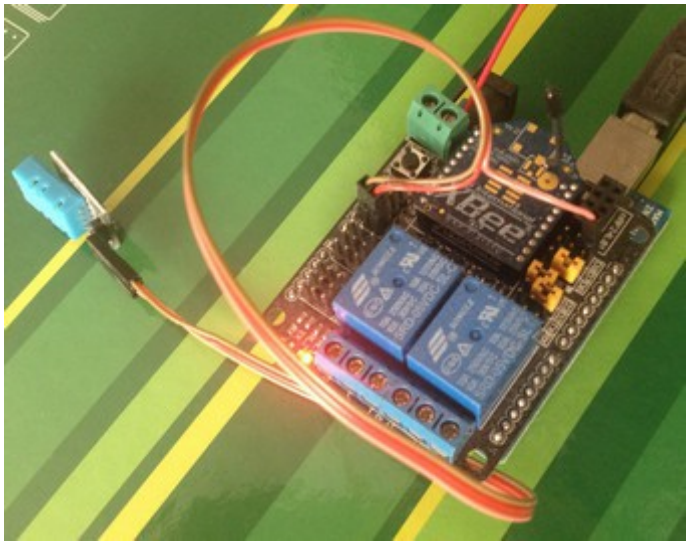

Ilustración 6: Detalle de placa arduino, sonda DHT22 y adaptador para XBEE y relés

2.3 Análisis de los protocolos de comunicaciones[9]

De los diferentes elementos que componen el proyecto, es necesario conocer el protocolo de comunicaciones de

1. El monitor de consumo eléctrico.
2. Los enchufes controlados por RF.
3. El ventilador controlado por RF.

Todos ellos se comunican por RF y usan una modulación ASK (OOK), para descifrar el protocolo de comunicaciones se requiere un PC con una tarjeta de sonido que disponga de una entrada de micrófono, una aplicación de sonido (en este proyecto se ha usado Audacity[10]) que muestre la gráfica del audio recibido por la entrada de micro, y un cable con un condensador en serie que conecte la señal de datos del receptor (o emisor) de los elementos a analizar a dicha entrada.

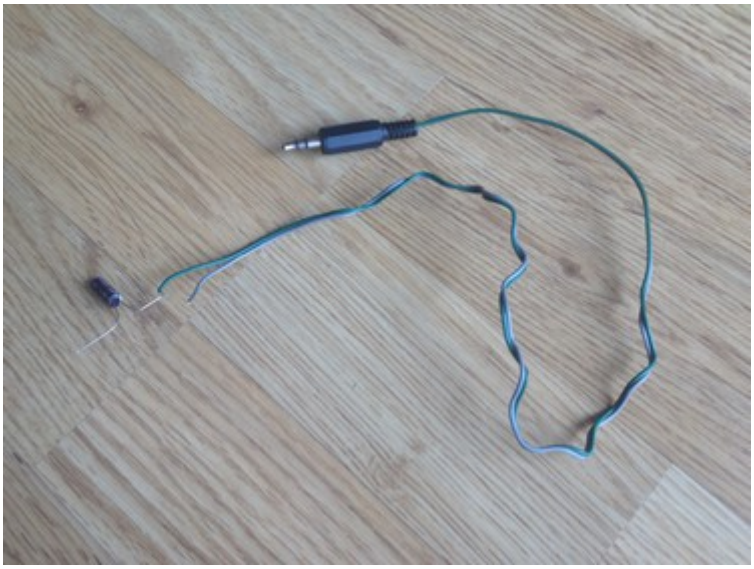

Ilustración 7: Conector minijack para decodificar señales RF

Una vez que se está en disposición de grabar la señal que se transmite/recibe, hay que almacenar cada una de las señales, y buscar la diferencias entre cada una de ellas para encontrar una correlación lógica en cada una de ellas. Hay que tener en cuenta que a pesar de la modulación ASK, cada fabricante usa un cifrado propio para determinar los estados lógicos. A modo de ejemplo, unos fabricantes emiten tres pulsos, uno alto y dos bajos para enviar un “0” lógico, y otros emiten un pulso alto y tres bajos para emitir un “0” lógico.

2.3.1 Monitor de consumo eléctrico

Una vez capturado con audacity la señal del monitor de consumo, hay que decodificarla, y averiguar el significado de cada bit. La trama en audacity se ve de la siguiente forma:

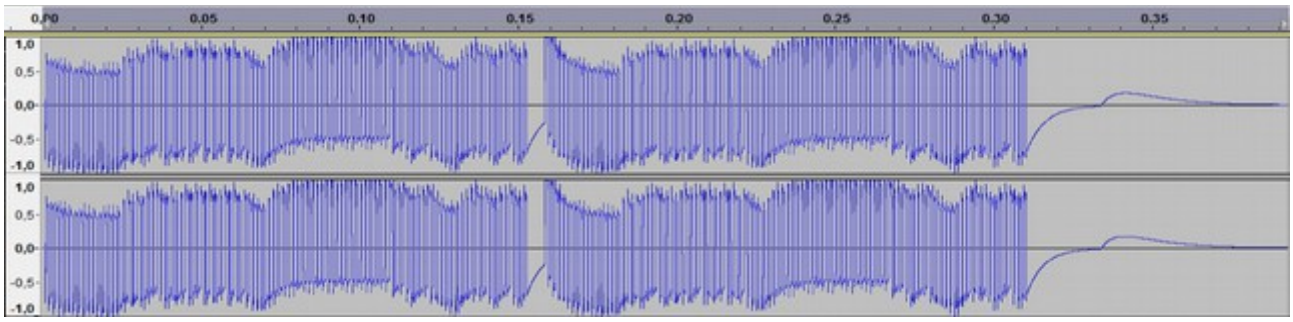

Ilustración 8: Trama del monitor de consumo eléctrico

Se aprecia claramente cómo existen dos tramas. Y si la ampliamos, se puede ver que el tamaño de los intervalos más pequeños es de 500ms:

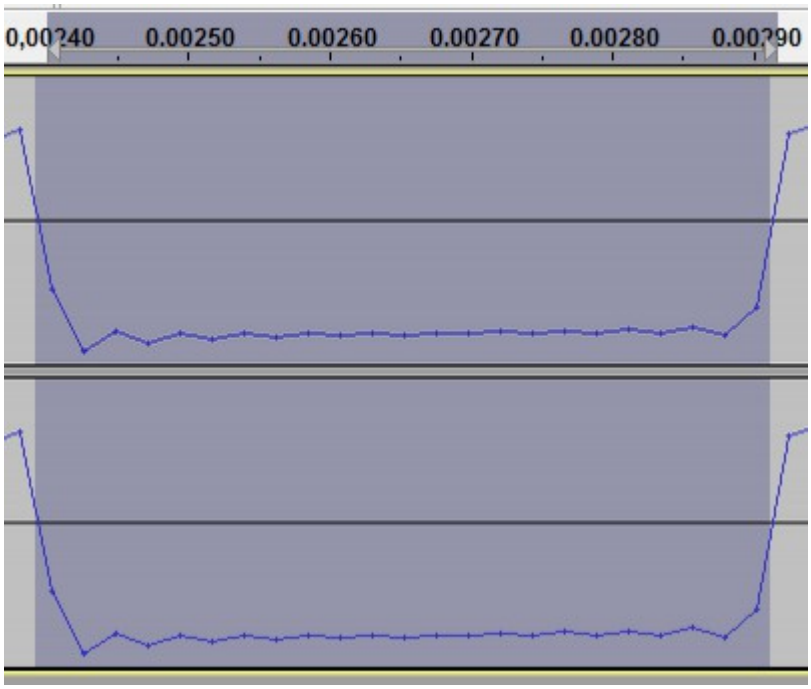

Ilustración 9: Intervalo más pequeño en la señal del control de corriente

Puesto que se trata de una señal ASK/OOK, se transforma dicha señal en una trama de ceros y unos, resultado, considerando una señal elevada como un uno, y una señal baja como un cero:

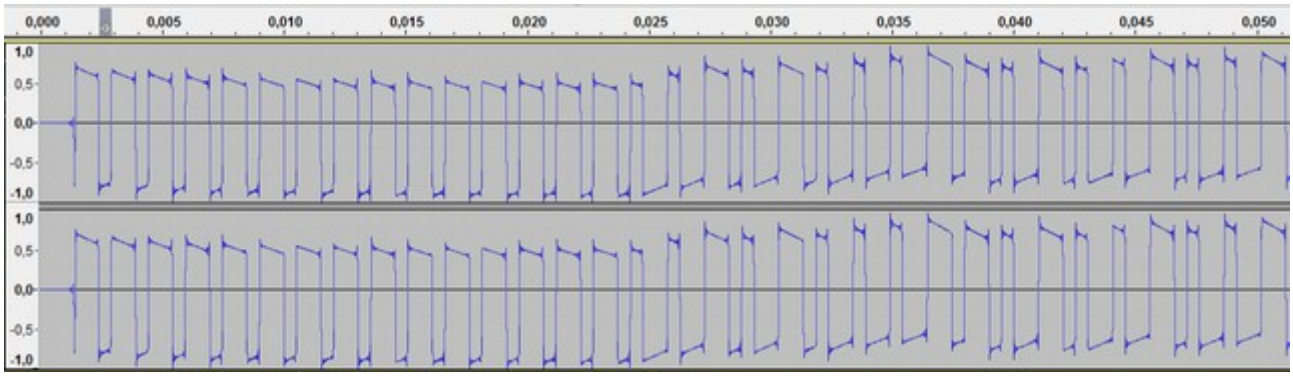

Ilustración 10: Detalle de pulsos

```
0010010010010010010010010010010010010010010010110110010110010110110110010010110010110110010110110010
0110010110110010110010110010010010010010110010110110110110110110110110110110110110110110110110110110
0110110110110110010110110010010110110110010010010010110010010110010110010110010010110110110110010010
0
```

Tabla 1: Pulsos de la señal, convertidos en ceros y unos

Con un poco más de detalle, se puede apreciar como hay secuencias de 3 bits, por lo que el protocolo de comunicaciones es similar al encontrado en otros aparatos electrónicos, cuya codificación es:

```
 _
|_|__ --> 1

 __
|  |_ --> 0
```

Tabla 2: Codificación de pulsos a binario

Es decir, un uno, seguido de dos ceros se transforma en un 1 lógico, y dos ceros seguidos de un uno equivale a un cero lógico.

Ahora se decodifica la trama original y se obtiene:

```
111111111111111001010001101001001010010101111101000000000000000000000000100110001111011010101100001
1
```

Tabla 3: Trama decodificada

Tras analizar varias tramas, con varios consumos (se asocia cada trama con el valor que refleja el display), se descartan los fragmentos de trama que son idénticos, y se intenta obtener un valor número que represente lo mostrado en el display.

Con esta dinámica se obtiene que el comienzo de la trama (44 bits) siempre es idéntico:

```
11111111111111100101000110100100101001010111
```

Tabla 4: Señal de sincronización. Comienzo de la trama idéntico

A continuación hay cuatro bits de sincronización (es un contador secuencial que puede servir para determinar si se han perdido tramas), y después 16 bits que indican el valor de los céntimos de amperaje medido (los amperios por la tensión indica el consumo en vatios). Se desconoce la aplicación del resto de la trama.

```
11111111111111100101000110100100101001010111 1111 0000000000101110 - 0,46 A
11111111111111100101000110100100101001010111 1010 0000000001000010 - 0,66 A
11111111111111100101000110100100101001010111 0110 0000000101001001 - 3,29 A
11111111111111100101000110100100101001010111 1010 0000001000110000 - 5,60 A
11111111111111100101000110100100101001010111 0101 0000001001001110 - 5,90 A
```

Tabla 5: Tramas con diversas medidas

Por tanto, de los primeros 64 bits, el valor de los últimos 16 indican la medida que interesa.

2.3.2 Los enchufes controlados por RF

Ya existe una librería de software para controlar estos dispositivos, por lo que no es necesario investigar su codificación. El código[11] original está escrito para los entrenadores digitales arduino, y existe un port para Raspberry Pi[12]. La codificación es como en el caso de los enchufes:

¯|¯|__ --> 1
¯¯|¯ |_ --> 0

Tabla 6: Codificación de pulsos a lenguaje binario

2.3.3 El ventilador controlado por RF

Este dispositivo tiene 8 pulsadores, trabaja en una frecuencia diferente a los enchufes (315Mhz), y de forma análoga a los enchufes, también tiene microrruptores para definir el identificador del equipo.

La codificación en este caso es triestado, y el código del apartado anterior también la puede gestionar.

¯|___|¯|___ → 0
___ ___ | |_| |_ → 1
¯|___| |_ → F

Tabla 7: Codificación de pulsos a lenguaje binario

Tras analizar las diferentes señales que emite el mando en cada uno de los pulsadores, el resumen es:

```
FAN 0FFF10000000
1H  0FFF01000000
2H  0FFF00100000
4H  0FFF00010000
LUZ 0FFF00001000
HI  0FFF00000100
MED 0FFF00000010
LOW 0FFF00000001
```

Tabla 8: Resumen de codificación de las diferentes señales

2.3.4 El motor de persiana radiocontrolado.

De forma análoga al resto de dispositivos, el control de las persianas se hace mediante RF en la frecuencia de 433MHz. La decodificación del protocolo para una ventana (un único canal) muestra que la frecuencia de los pulsos es de 340microsegundos, y la trama tiene un preámbulo de 14 ciclos con la señal arriba, y 4 ciclos con la señal abajo. Además, usa el protocolo visto con anterioridad en la que un 0 lógico viene dado por un ciclo activo y dos inactivos, y un 1 lógico se representa por medio de dos ciclos activos y uno inactivo. Quedando las tramas que hay que enviar de la siguiente forma:

Señal para subir: 000101100011000001101010111100010001111 00
Señal para bajar: 000101100011000001101010111100010011001 10
Señal para parar: 000101100011000001101010111100010101010 11

Tabla 9: Codificación de señales para el control de la persiana

A estas tramar hay que anteponerles la cabecera.

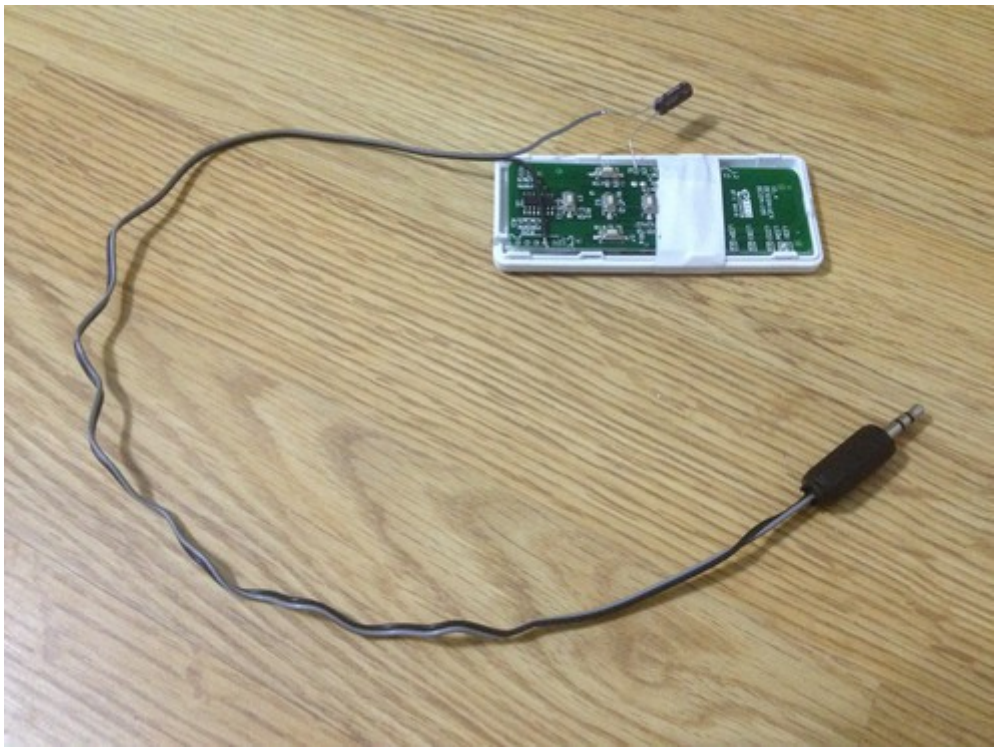

Ilustración 11: Interior del mando a distancia de las persianas con el conector jack para decodificar la señal

2.4 Geoposicionamiento

El sistema de gestión de domótica escogido (openHAB) dispone de complementos que permiten su integración con terceros, como es el caso del gestor de comunicaciones MQTT[13]. Este gestor de colas permite comunicar diversos elementos entre sí, entre los que se encuentra la aplicación para *smartphone* Owntracks[14].

Esta aplicación envía los datos de geoposicionamiento que dispone el terminal a un servidor de Mqtt. La idea es instalar un servidor de Mqtt en el equipo que ejecute openHAB, para que reciba la posición del *smartphone*, y a su vez, esta posición sea consultada por openHAB para realizar las acciones pertinentes.

3 Desarrollo e implementación

3.1 Instalación del dispositivo de control

El equipo que gestionará los diversos elementos es un ordenador Raspberry Pi modelo B; este dispositivo usa tarjetas de memoria SD como medio de almacenamiento y por tanto, el sistema operativo estará alojado en dicha tarjeta. Se ha optado por una tarjeta de memoria SD de 16GB de clase 10, para que el sistema operativo tenga un mejor ratio de acceso a disco, y el sistema operativo escogido es linux, distribución Arch, compilada para el procesador ARM.

Ilustración 12: Raspberry Pi Modelo B. Fuente: www.raspberrypi.org

La alimentación se hace a través de un cargador micro-usb. Se usará uno con 2A de corriente, para prevenir posibles caídas de tensión en la interfaz GPIO debido al uso de los elementos de control, como los receptores/emisores de radiofrecuencia.

Por otra parte, la conectividad ethernet del equipo se hará con un interfaz wifi conectado a uno de los dos puertos USB que dispone, también estará conectado mediante HDMI al televisor gestionado y finalmente, un teclado usb se usará como método de entrada para la configuración inicial.

Los pasos necesarios para la instalación[15] son:

1. Descargar la imagen del sistema operativo desde la web del fabricante del equipo en la dirección [http://downloads.raspberrypi.org/arch_latest].
2. Descargar el software necesario (Win32diskImager) para volcar la imagen a la tarjeta SD desde la estación de trabajo de la dirección [http://sourceforge.net/projects/win32diskimager/].

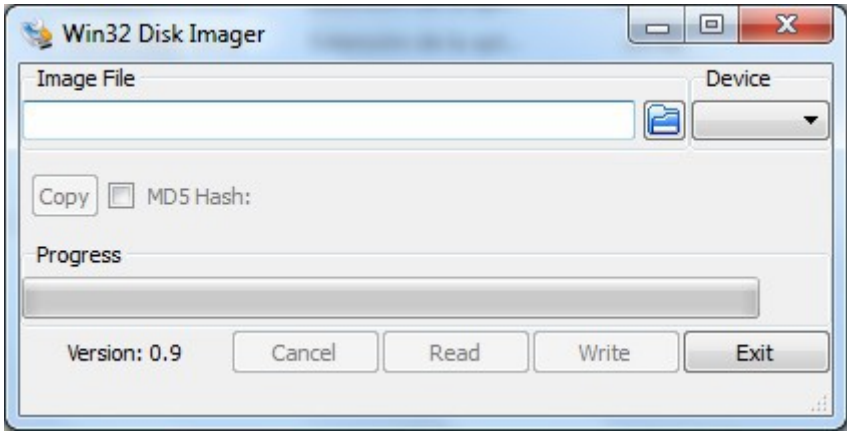

Ilustración 13: Herramienta para el volcado de imágenes

3. Volcar la imagen a la tarjeta SD.
4. Redimensionar la partición linux[16] para que ocupe todo el espacio de la tarjeta.
5. Configurar los datos de red.
6. Cambiar la clave y activar el servicio sshd.
7. Modificar la configuración regional.
8. Actualizar el sistema.

Una vez que el sistema se encuentra operativo y actualizado, se han llevado a cabo otras modificaciones:

- Instalar herramientas de desarrollo (gcc, make, etc.) necesarias posteriormente para el control de los elementos de radio frecuencia.
- Instalación y configuración de watchdog[17]. Este servicio usa un módulo del hardware para detectar si el sistema se ha vuelto inestable (o no responde) y provocar un reinicio del mismo en un intento de recuperar el control.
- Instalación y configuración de un cliente de dns dinámico (noip2[18]) para poder acceder al sistema desde cualquier equipo con conexión a internet usando una ip dinámica ofrecida por el ISP al que se encuentra conectado el equipo.
- Configuración de las reglas de nat en el router que ofrece acceso a internet al dispositivo, para que el demonio ssh y el resto de servicios necesarios para el entorno sean accesibles.

3.2 Instalación de los periféricos de control remoto

El ordenador de control dispone de una interfaz llamada GPIO (General Purpouse Input Output) que permite controlar señales digitales de 3,3v. A través de esta interfaz, se va a gestionar:

1. Un emisor de 433MHz
2. Un receptor de 433MHz
3. Un emisor de 315MHz
4. Un diodo led de control.

Dicha interfaz tiene diversas API[19] para su gestión, en este proyecto se usará una escrita en el lenguaje de programación C++[20].

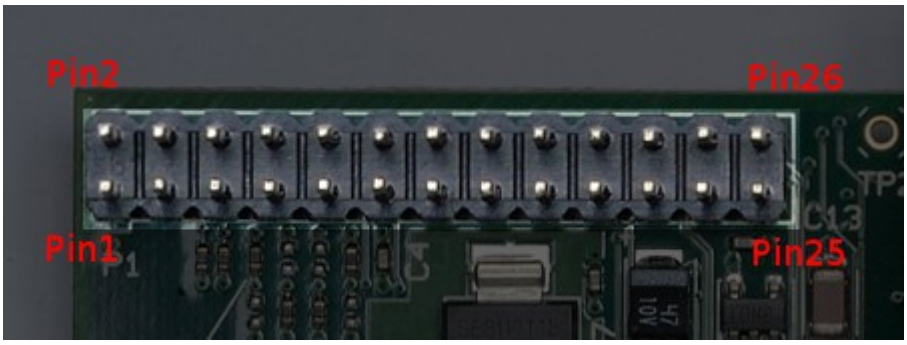

Ilustración 14: Interfaz GPIO Fuente: http://elinux.org/RPi_Low-level_peripherals

Dicha interfaz dispone de pines de alimentación para los diversos elementos, además de los pines de control (se pueden definir como entrada o como salida). En concreto, el emisor de 433MHz se ha conectado al pin1 en la nomenclatura WiringPi, que se corresponde con el pin 12. El pin 7 -la numeración en este caso es la misma tanto en WirinPi como en GPIO- contiene el diodo led de control (para verificar el correcto funcionamiento del entorno y partir de un caso sencillo). El pin 0 (11 en GPIO) tiene conectado el emisor de 315MHz (para el ventilador controlado por RF) y por último el pin 3 (15 en GPIO) tiene conectado el receptor de 433MHz para recibir las señales del medidor de consumo y de los mandos de los diversos elementos que operan en esa señal.

Una vez identificados los pines de señal, la conexión se hará mediante cables de cobre y conectores “dupont” para que su ensamblado sea fácil. Tanto los emisores como el receptor disponen de tres pines de conexión, más otro para la antena. Uno de los pines (Vcc) se conecta a una salida del GPIO que marque 3.3v ó 5v, según corresponda (los emisores/receptores adquiridos soportan un rango de tensión que tolera cualquiera de las dos), otro de los pines es el negativo (Gnd o ground) y se conecta a la señal de 0v. Por último, el pin marcado como data es el que se conectará conforme se

ha especificado en el párrafo anterior.

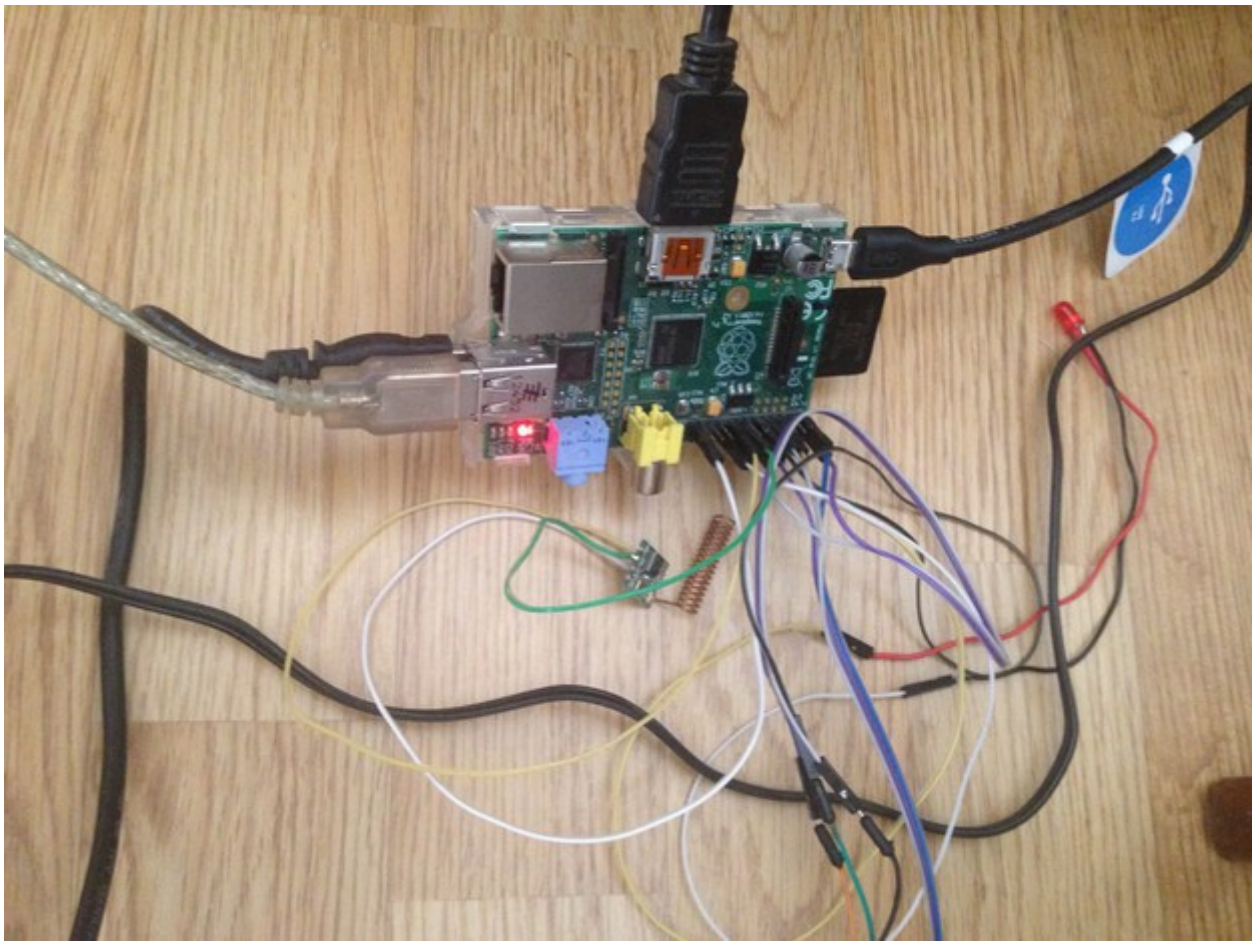

Ilustración 15: Vista parcial del prototipo

Una vez que el piloto se haya validado, los cables se podrán soldar para ofrecer una mejor conexión y prevenir posibles fallos de contacto.

Junto a estos elementos, también es necesario otro dispositivo de comunicación con las sondas de temperatura. Se trata de un adaptador de puerto USB para un chip Xbee que se conectará al dispositivo raspberry pi, y se gestionará desde el sistema operativo como si de un puerto serie se tratase. Con este elemento se podrá enviar señales a los equipos Xbee para activar relés que enciendan o apaguen dispositivos, y también recibirlas para obtener la información de temperatura y humedad de la sonda remota. En el otro extremo (inalámbrico) existe una placa arduino, una tarjeta de expansión que permite conectar el chip Xbee y que dispone de dos relés.

3.3 Instalación de las herramientas software necesarias

3.3.1 Máquina virtual Java

La herramienta de control escogida es openHAB y requiere de una máquina virtual java para su funcionamiento, por lo que es necesario instalar este componente en el sistema operativo. Dado que el equipo dispone de pocos recursos, y soporta la ejecución de instrucciones de coma flotante por hardware, se lleva a cabo la instalación del componente java para plataforma arm con dicho soporte (hard float). Oracle ha publicado un video[21] explicativo con las diferencias de ambas versiones.

La dirección de descarga es http://www.oracle.com/technetwork/java/embedded/downloads/javase/index.html?ssSourceSiteId=otncn y hay que escoger la versión "ARMv6/7 Linux - Headful EABI, VFP, HardFP ABI, Little Endian1". En el momento de redactar este documento, la versión disponible era la "ejre-7u55-fcs-b13-linux-arm-vfp-hflt-server_headless-17_mar_2014.tar.gz" que hay que descomprimir en /usr. Para poder modificar fácilmente la versión de java instalada sin necesidad de tocar ninguna configuración adicional, se creará un enace simbólico, apuntando al directorio ya descomprimido, en la ruta /usr/java. De esta forma si se necesita modificar la versión de java, sólo será necesario modificar dicho enlace simbólico.

Una vez descomprimido, y creado el enlace simbólico, se creará un script para que altere la configuración del entorno, y agregue en la variable PATH la ruta a los binarios de java. Para ello se crea el fichero de texto /etc/profile.d/java.sh con el contenido:

```
# Set path to java

PATH=$PATH:/usr/java/bin
JAVA_HOME=/usr/java

export PATH
export JAVA_HOME
```

Tabla 10: Configuración del entorno java

De esta forma, cada usuario que inicie sesión ya tendrá disponible en su entorno los binarios de java sin necesidad de teclear la ruta completa.

3.3.2 Software necesario para la comunicación con la sonda de temperatura

Hay dos elementos que intervienen en esta comunicación. Por un lado se encuentra el equipo con arduino[22], la sonda de temperatura y los relés. Este dispositivo tiene el siguiente código instalado en su memoria:

```
#include <SoftwareSerial.h>
#include <stdlib.h>

#include <TimerOne.h>

#include "DHT.h"
#define DHTPIN 8

#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define rxPin 2
#define txPin 3

int rele1 = 4;
int rele2 = 5;

char temp_str[13];

SoftwareSerial mySerial(rxPin, txPin);

void Callback()
{
   Serial.println("------->  Callback Send AT");
   mySerial.print("AT");
}

void setup()
{
   // define pin modes for tx, rx, led pins:
   pinMode(rxPin, INPUT);
   pinMode(txPin, OUTPUT);
   mySerial.begin(9600);
   Serial.begin(9600);

   pinMode(rele1, OUTPUT);
   pinMode(rele2, OUTPUT);

   dht.begin();

   Serial.println("-------> Start ");
}

void loop()
{
  int i = 0;
  char someChar[32] = {0};
  char tempF[10] = {0};
  String temperature;
  char buffer;
byte byteRead;
float h = dht.readHumidity();
float t = dht.readTemperature();

  // when characters arrive over the serial port...
  if(Serial.available()) {
    Serial.println();
    Serial.print("------->  ");
    do{
      someChar[i++] = Serial.read();        // Note HC-06 end not '/r/n', direct input "AT" command
from Serial port
    }while (Serial.available() > 0);

    mySerial.print(someChar);
    Serial.println(someChar);
  }
```

```
int tempInt = (int) t;
int tempDec = (int) (t - tempInt) * 100;
temperature = "<T1:" + String(tempInt,DEC) + "." + String(tempDec,DEC) + ">";

char tempChar[temperature.length()+1];
temperature.toCharArray(tempChar,temperature.length()+1);
for (int index=0; index < temperature.length()+1; index++) {
  Serial.print(tempChar[index]);
  mySerial.print(tempChar[index]);
}
  Serial.println("");
  mySerial.println("");

delay(1000);

  while(mySerial.available()) {
    buffer = (char)mySerial.read();
    if (buffer == 'H'){
      digitalWrite(rele1,HIGH);
    }else if (buffer == 'L'){
      digitalWrite(rele1,LOW);
    }else if (buffer == 'U'){
      digitalWrite(rele2,HIGH);
    }else if (buffer == 'D'){
      digitalWrite(rele2,LOW);
    }else if (buffer == '1'){
      mySerial.println(digitalRead(rele1));
      Serial.println(digitalRead(rele1));
    }else if (buffer == '2'){
      mySerial.println(digitalRead(rele2));
      Serial.println(digitalRead(rele2));
    }
    Serial.print(buffer);
  }
}
```

Tabla 11: Software de la primera sonda de temperatura con arduino

Se puede apreciar en el programa, como se envía una cadena de texto que empieza por <T1: para indicar la temperatura y por <H1: para la humedad. Estos caracteres serán los que determinen en el otro extremo de qué sonda se está recibiendo información. Además, si este equipo recibe un carácter H/L se activa/desactiva el primer relé, y si recibe U/D ocurre lo mismo con el segundo actuador.

El software para tratar la información recibida en Raspberry Pi permite parametrización, útil en el caso de disponer de varias sondas de temperatura. Los parámetros que recibe son el identificador de la sonda (1 pues sólo existe una sonda en la actualidad) y el nombre del elemento a actualizar en openHAB sin el prefijo Temperature o Humidity que se incluirá automáticamente en el script. El código es el siguiente:

```
#!/bin/bash

OH_URL=http://localhost:8080
OH_USER=XXXXXX
OH_PASS=XXXXXX
OH_ITEM=$2

TEMP_FILE=/tmp/temp1.log
DEVICE=/dev/xbee

dd if=$DEVICE count=10 status=none > $TEMP_FILE

TEMP=`grep -a T$1 $TEMP_FILE | cut -c 5- | tr -d ">"`
HUMIDITY=`grep -a H$1 $TEMP_FILE | cut -c 5- | tr -d ">"`

curl --user $OH_USER:$OH_PASS --max-time 2 --connect-timeout 2 --header "Content-Type: text/plain"
--request PUT --data "$TEMP" $OH_URL/rest/items/Temperature$OH_ITEM/state
curl --user $OH_USER:$OH_PASS --max-time 2 --connect-timeout 2 --header "Content-Type: text/plain"
--request PUT --data "$HUMIDITY" $OH_URL/rest/items/Humidity$OH_ITEM/state
```

Tabla 12: Software de monitorización de temperatura y humedad

Dicho script se lanzará automáticamente mediante una tarea de cron.

El otro componente que hay que definir en Raspberry Pi es el que permite activar o desactivar los relés que tiene conectados arduino. Hay un script por cada relé y el código es el siguiente:

```
#!/bin/bash
USBPORT=/dev/xbee

if [ $1 -eq 1 ]; then
        case $2 in
                ON) echo H > $USBPORT
                ;;
                OFF) echo L > $USBPORT
                ;;
        esac
else
        case $2 in
                ON) echo U > $USBPORT
                ;;
                OFF) echo D > $USBPORT
                ;;
        esac
fi
```

Tabla 13: Código que ejecuta openHAB para activar/desactivar los relés de arduino

Se puede ver como usa dos parámetros, el primero que indica el relé a usar y el segundo la acción a acometer.

3.3.3 Servidor MQTT (mosquitto)

La conectividad con la herramienta de geoposicionamiento se llevará a cabo mediante un protocolo de mensajería llamado MQTT. La aplicación owntracks[23] se instala en los smartphone de los inquilinos de la vivienda, y se configura para que envíe la posición del smartphone al servidor de mensajería. El servidor mqtt a su vez, será consultado por la herramienta de domótica sobre los cambios de posición de los inquilinos, detectando di la posición coincide con las definidas en su

listado.

La implementación del servidor mqtt que se ha escogido se llama mosquitto, y está disponible en la URL http://www.mosquitto.org. Los pasos a seguir para su instalación[24] son:

1. Obtener el software
2. Descomprimir el software
3. Compilar e instalar con make
4. Editar el fichero de configuración
5. Crear el servicio en el sistema operativo e iniciarlo

El contenido del fichero de configuración usado es:

```
pid_file /var/run/mosquitto/mosquitto.pid
user mosquitto
log_dest syslog
password_file /etc/mosquitto/mosquitto.pass
```

Tabla 14: Fichero de configuración de mosquitto

Se puede apreciar como el usuario del servicio es mosquitto, por lo que será necesario crear también dicho usuario en el sistema. Además, el fichero con las credenciales de los suscriptores del broker será necesario crearlo con el comando mosquitto_passwd y se almacenará en la ruta especificada en el fichero de configuración.

El contenido del fichero para crear el servicio "/etc/systemd/system/mosquitto.service" es:

```
[Unit]
Description=Mosquitto MQTT Broker daemon
ConditionPathExists=/etc/mosquitto/mosquitto.conf
Requires=network.target

[Service]
Type=simple
ExecStartPre=/usr/bin/rm -Rf /var/run/mosquitto
ExecStartPre=/usr/bin/mkdir /var/run/mosquitto
ExecStartPre=/usr/bin/chown mosquitto /var/run/mosquitto
ExecStartPre=/usr/bin/rm -f /var/run/mosquitto/mosquitto.pid
ExecStart=/usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf -d
ExecReload=/bin/kill -HUP $MAINPID
PIDFile=/var/run/mosquitto/mosquitto.pid
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Tabla 15: Fichero de configuración del servicio mosquitto

Esta es la configuración básica y no implementa mecanismos de seguridad en la comunicación. Se puede configurar que el protocolo vaya cifrado mediante SSL, pero se deja dicha configuración para una posterior fase del proyecto como mejora del mismo.

Una vez que este servicio se encuentra operativo, es momento de configurar la aplicación Owntracks en el smartphone.

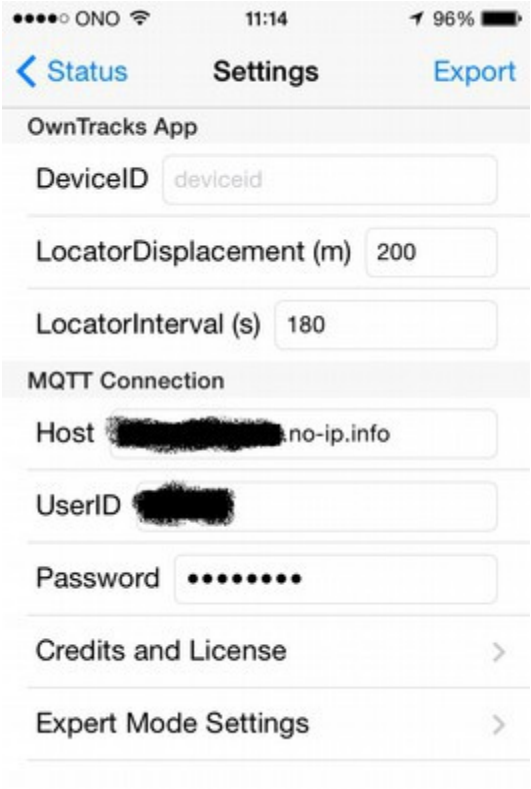

Ilustración 16: Configuración de owntracks en un dispositivo iOS

En el host se usa la dirección que se ha creado en el servicio de dns dinámico, para poder conectarse al equipo desde Internet, haciendo uso de la ip dinámica que ofrece el ISP usado por el equipo de control.

Las credenciales que solicita owntracks son las definidas con el comando mosquitto_passwd para los suscriptores de la mensajería.

Por otra parte, se crearán tantos “amigos” como ubicaciones se quieran gestionar dentro de openHAB. En este caso se han creado dos, una para detectar la ubicación de la vivienda, y otra para detectar el lugar de trabajo habitual.

3.3.4 Control de los enchufes

El control de los enchufes se hace con el emisor conectado en el pin wiringPi 1, y se basa en la librería rcswitch-pi, que gestiona el protocolo de comunicaciones de gran variedad de enchufes controlados por radiofrecuencia. El ejecutable creado para la gestión de los calefactores se encuentra en /usr/local/bin/send1 y el código fuente es:

```
/*
 Usage: ./send1 <systemCode> <unitCode> <command>
 Command is 0 for OFF and 1 for ON
 */

#include "RCSwitch.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {

    /*
     output PIN is hardcoded for testing purposes
     see https://projects.drogon.net/raspberry-pi/wiringpi/pins/
     for pin mapping of the raspberry pi GPIO connector
     */
    int PIN = 1;
    char* systemCode = argv[1];
    int unitCode = atoi(argv[2]);
    int command  = atoi(argv[3]);

    if (wiringPiSetup () == -1) return 1;
        printf("sending systemCode[%s] unitCode[%i] command[%i]\n", systemCode, unitCode, command);
        RCSwitch mySwitch = RCSwitch();
        mySwitch.enableTransmit(PIN);

    switch(command) {
        case 1:
            mySwitch.switchOn(systemCode, unitCode);
            break;
        case 0:
            mySwitch.switchOff(systemCode, unitCode);
            break;
        default:
            printf("command[%i] is unsupported\n", command);
            return -1;
    }
        return 0;
}
```

Tabla 16: Código fuente de la utilidad para controlar los enchufes de forma remota

3.3.5 Recepción de datos de consumo eléctrico

La recuperación de los datos de consumo eléctrico se hace por radiofrecuencia, y la utilidad creada a tal efecto también se basa en el proyecto rcswitch-pi, y consiste en crear una interrupción que detecte los pulsos del receptor de RF. Cada vez que dicha interrupción se lance, se activará un contador de tiempo, para conocer cuanto tiempo pasa entre pulso y pulso, para luego comprobar si la cadena de pulsos cumple con los patrones establecidos, y en caso afirmativo, almacenar la información recogida en openHAB.

El mismo ejecutable se usará para detectar señales de otros dispositivos. En concreto, se monitorizará que el mando de los enchufes radiocontrolados pulse active cualquiera de los radiadores, para que esa información se vea actualizada en openHAB.

```
bool RCSwitch::receiveProtocol3(unsigned int changeCount){

          unsigned long code = 0;
          unsigned long value = 0;
      unsigned long delay = 500;
      unsigned long delayTolerance = delay * RCSwitch::nReceiveTolerance * 0.01;
// El primer bit no se tiene en cuenta, el contador empieza en 1
      unsigned int
master[RCSWITCH_MAX_CHANGES]={0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,0,1,0,0,0,1,1,0,1,0,0,1,0,0,1,0
,1,0,0,1,0,1,0,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1
,1,0,1,0,1,0,0,0,0,0,0,0,1,0,1,1,0,0,1,0,1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,1,1,0,0};

      for (int i = 1; i<changeCount ; i=i+2) {
          if (RCSwitch::timings[i] > delay-delayTolerance && RCSwitch::timings[i] <
delay+delayTolerance && RCSwitch::timings[i+1] > delay*2-delayTolerance && RCSwitch:
:timings[i+1] < delay*2+delayTolerance) {
            RCSwitch::bin[i/2+1] = 0;
            code = code << 1;
            if ((i >= 96) && (i < 128)) {
                value = value << 1;
            }
          } else if (RCSwitch::timings[i] > delay*2-delayTolerance && RCSwitch::timings[i] <
delay*2+delayTolerance && RCSwitch::timings[i+1] > delay-delayTolerance &&
RCSwitch::timings[i+1] < delay+delayTolerance) {
            RCSwitch::bin[i/2+1] = 1;
            code+=1;
            code = code << 1;
            if ((i >= 96) && (i < 128)) {
                  value += 1;
                  value = value << 1;
            }
          } else {
            // Failed
            i = changeCount;
            code = 0;
          }
          if ((i < 88) && (RCSwitch::bin[i/2+1] != master[i/2+1])) {
                return false;
          }
      }
      value = value >> 1;
      code = code >> 1;

    if (changeCount > 65) {    // ignore < 32bit values
      RCSwitch::nReceivedValue = code;
      RCSwitch::nReceivedAmpValue = value;
      RCSwitch::nReceivedBitlength = changeCount / 2;
      RCSwitch::nReceivedDelay = delay;
          RCSwitch::nReceivedProtocol = 3;
    }
        if (code == 0){
                return false;
        }else if (code != 0){
                return true;
        }
}
```

Tabla 17: Función para la recepción del contador de corriente. Incorporada en la librería RCSwitch

```
/*
  RF_Sniffer

  Hacked from http://code.google.com/p/rc-switch/

  by @justy to provide a handy RF code sniffer
*/

#include "RCSwitch.h"
#include <stdlib.h>
#include <stdio.h>
#include <curl/curl.h>


RCSwitch mySwitch;



int main(int argc, char *argv[]) {

     // This pin is not the first pin on the RPi GPIO header!
     // Consult https://projects.drogon.net/raspberry-pi/wiringpi/pins/
     // for more information.
     // int PIN = 2;
     int PIN = 3;
     unsigned long bin = NULL;
     unsigned long amp = NULL;
     unsigned long watt = NULL;

     if(wiringPiSetup() == -1)
       return 0;

     mySwitch = RCSwitch();
     mySwitch.enableReceive(PIN);  // Receiver on inerrupt 0 => that is pin #2


     while(1) {

      if (mySwitch.available()) {

        int value = mySwitch.getReceivedValue();

        if (value == 0) {
          printf("Unknown encoding");
        } else {
          char command[254];
          if (mySwitch.getReceivedProtocol() == 3) {
                bin = mySwitch.getReceivedAmpValue();
                amp = bin * 0.01;
                watt = bin * 2.2;
                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2 --connect-timeout
2 --header \"Content-Type: text/plain\" --request PUT --data %i htt
p://XXXXXX:XXXXXX@localhost:8080/rest/items/Current_GF/state",watt);
                system(command);
                printf("Received %i - %i cent. A - %iW\n", mySwitch.getReceivedValue(), bin, watt);
}
          else {
                switch (mySwitch.getReceivedValue())
                {
                        case 1361:
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data ON http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Corridor/state");
                                system(command);
                                printf("Se ha encendido 1\n");
                                break;
                        case 1364:
```

```
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data OFF http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Corridor/state");
                                system(command);
                                printf("Se ha apagado 1\n");
                                break;
                        case 4433:
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data ON http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Boy/state");
                                system(command);
                                printf("Se ha encendido 2\n");
                                break;
                        case 4436:
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data OFF http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Boy/state");
                                system(command);
                                printf("Se ha apagado 2\n");
                                break;
                        case 5201:
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data ON http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Corridor2/state");
                                system(command);
                                printf("Se ha encendido 3\n");
                                break;
                        case 5204:
                                sprintf(command,"/usr/bin/curl --user XXXXXX:XXXXXX --max-time 2
--connect-timeout 2 --header \"Content-Type: text/plain\" --request P
UT --data OFF http://XXXXXX:XXXXXX@localhost:8080/rest/items/Heating_GF_Corridor2/state");
                                system(command);
                                printf("Se ha apagado 3\n");
                                break;
                        case 5393:
                                printf("ENCENDEMOS 4\n");
                                break;
                        case 5396:
                                printf("APAGAMOS 4\n");
                                break;
                        default:
                                printf("Received %i\n", mySwitch.getReceivedValue() );
                                break;
                }
              printf("Received %i\n", mySwitch.getReceivedValue());
          }
          mySwitch.setProtocol(1);
        }

        mySwitch.resetAvailable();

      }
  }
  exit(0);
}
```

Tabla 18: Aplicación que recibe y decodifica las señales del contador de corriente y los enchufes de RF

3.3.6 Control del ventilador

De la misma forma que el control de los enchufes, el ventilador usa la misma librería rcswitch-pi, y

dado que la trama de bits ya se conoce, el código fuente del ejecutable es el siguiente:

```
/*
 Usage: ./send <code>
 */

#include "RCSwitch.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {

    /*
     output PIN is hardcoded for testing purposes
     see https://projects.drogon.net/raspberry-pi/wiringpi/pins/
     for pin mapping of the raspberry pi GPIO connector
     */
    int PIN = 0;
    char* code = argv[1];

    if (wiringPiSetup () == -1) return 1;
        printf("sending code[%s]\n", code);
        RCSwitch mySwitch = RCSwitch();
        mySwitch.enableTransmit(PIN);

/*     switch(command) {
        case 1:
            mySwitch.switchOn(code, unitCode);
            break;
        case 0:
            mySwitch.switchOff(code, unitCode);
            break;
        default:
            printf("command[%i] is unsupported\n", command);
            return -1;
    } */
        mySwitch.sendTriState(code);
        return 0;
}
```

Tabla 19: Utilidad para controlar el ventilador de techo con luz incorporada

3.3.7 Control de la persiana

De la misma forma que en el resto de equipos gestionados por radio frecuencia, en este caso también se ha hecho uso del proyecto rcswitch-pi, aunque en este caso la trama de bits no la gestiona de forma directa la librería RCSwitch, sino que se envía en el ejecutable, dado que el preámbulo no sigue el mismo patrón que en los otros casos, y es necesario enviarlo mediante pulsos. El ejecutable ha quedado de la siguiente forma:

```
#include "RCSwitch.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[]) {

    /*
     output PIN is hardcoded for testing purposes
     see https://projects.drogon.net/raspberry-pi/wiringpi/pins/
     for pin mapping of the raspberry pi GPIO connector
```

```
     */
    int PIN = 1;
    char* upCode   = "000101100011000001101010111100010001111100";
    char* stopCode = "000101100011000001101010111100010101011";
    char* downCode = "000101100011000001101010111100010011001110";
    if (wiringPiSetup () == -1) return 1;
    int command = 5;
    if (argc == 2) command  = atoi(argv[1]);
        RCSwitch mySwitch = RCSwitch();
        mySwitch.setProtocol(2,340);
        mySwitch.enableTransmit(PIN);

    switch(command) {
        Case 1:
            printf("sending UP command\n");
            mySwitch.transmit(14,4);
            mySwitch.send(upCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(upCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(upCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(upCode);
            break;
        case 0:
            printf("sending DOWN command\n");
            mySwitch.transmit(14,4);
            mySwitch.send(downCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(downCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(downCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(downCode);
            break;
        case 2:
            printf("sending STOP command\n");
            mySwitch.transmit(14,4);
            mySwitch.send(stopCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(stopCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(stopCode);
            delayMicroseconds( 340 * 70);
            mySwitch.transmit(14,4);
            mySwitch.send(stopCode);
            break;
        default:
            printf("command[%i] is unsupported\n", command);
            printf("help: %s (0|1|2)\n", argv[0]);
            printf("0 is for Down the blind, 1 is for Up the blind and 2 is for stop the blind\n");
            return -1;
    }
        return 0;
}
```

Tabla 20: Utilidad para controlar las persianas por vía inalámbrica

3.3.8 Librería cliente de HDMI

Los dispositivos compatibles con HDMI pueden ser controlados a través de esta interfaz. La televisión que se integra en este proyecto dispone de dicho interfaz, por lo que se conectará al equipo controlador por esta vía, para que se pueda encender a través del sistema de domótica. Por esta razón es necesario instalar la librería libcec ejecutando el comando

```
pacman -S libcec
```

Tabla 21: Comando para instalar la utilidad de gestión HDMI

3.3.9 Control de la televisión

Las televisiones del fabricante Samsung en sus últimos modelos pueden ser controladas por tcp/ip. Gracias a esta funcionalidad, openHAB dispone de un complemento para su gestión. La limitación que tiene esta interfaz es que no permite el encendido, por lo que dicho control se ha relegado a un script externo a openHAB, para que vía HDMI se pueda encender. El comando a lanzar desde la línea de comandos de linux es:

echo 'on 0' | cec-client -s

Por otro lado, el televisor se puede encender sin necesidad de usar openHAB; para que openHAB conozca el estado en el que se encuentra, se chequea a través de HDMI de forma periódica con el siguiente script:

```
#!/bin/bash
OH_URL=http://localhost:8080
OH_USER=XXXXXX
OH_PASS=XXXXXX
OH_ITEM=TV_GF_Multimedia_TV_power
RESULT=`echo pow 0 | cec-client -d 1 -s | grep "power status:" | awk '{ print $3; }'`

case $RESULT in
        on)
                curl --user $OH_USER:$OH_PASS --max-time 2 --connect-timeout 2 --header "Content-
Type: text/plain" --request PUT --data "ON" $OH_URL/rest/items/$OH_ITEM
/state
                exit 0
                ;;
        *)
                curl --user $OH_USER:$OH_PASS --max-time 2 --connect-timeout 2 --header "Content-
Type: text/plain" --request PUT --data "OFF" $OH_URL/rest/items/$OH_ITE
M/state
                exit 1
esac
```

Tabla 22: Script para comprobar el estado del televisor (encendido/apagado)

3.3.10 Control de coste eléctrico por franja horaria

El mercado eléctrico español ha variado su facturación, y ahora se pasará a facturar por franjas horarias, conociendo el coste de cada hora el día anterior. Dicho coste se publica en la url

http://www.esios.ree.es/, y el siguiente script se descarga la tabla con los precios e indica el tramo consecutivo de dos horas en la que el coste es el más económico, para que la lavadora se active en ese momento. Finalmente, ese dato (la hora más económica) se publica en openHAB para que dicho dato se pueda utilizar en una regla que automatice el encendido de la lavadora.

```
#!/bin/bash
OH_URL=http://localhost:8080
OH_USER=XXXXXX
OH_PASS=XXXXXX
OH_ITEM=Current_GF_Time

if [ ! -d /tmp/PVPC ]; then
        mkdir /tmp/PVPC
fi

cd /tmp/PVPC

rm -Rf /tmp/PVPC/*

FECHA=`date +%Y%m%d`
HORA_ACTUAL=`date +%H`

if [ $HORA_ACTUAL -gt 19 ]; then
         let FECHA+=1
fi

wget -nv --user-agent="Mozilla/5.0 (Windows NT 5.2; rv:2.0.1) Gecko/20100101 Firefox/4.0.1" --post-
data "fileName=PVPC_DD_$FECHA&fileType=html&idioma=es" http://www.esi
os.ree.es/Solicitar -O PVPC.html

cat PVPC.html | sed -e "s/<tr>/\n<tr>/g"  > PVPC1.html

head -n -1 PVPC1.html | tail -n +5 > PVPC2.html

sed -e "s/<tr><td class=\"cabeceraTablaNoNegrita\" nowrap=\"nowrap\">/|/g" PVPC2.html > PVPC3.html
sed -e "s/<\/td><td class=\"filaTabla numero\" style=\"background-color:#CC0000;color:#FFFFFF;\"
data-colortramo=\"#FFF9CF\">/|/g" PVPC3.html > PVPC4.html
sed -e "s/<\/td><td class=\"filaTabla numero\" style=\"background-color:#FFF9CF;color:#005675;\"
data-colortramo=\"#FFF9CF\">/|/g" PVPC4.html > PVPC5.html
sed -e "s/<\/td><td class=\"filaTabla numero\" style=\"background-color:#3399FF;color:#FFFFFF;\"
data-colortramo=\"#FFF9CF\">/|/g" PVPC5.html > PVPC6.html
sed -e "s/<\/td><td class=\"filaTabla numero\" style=\"background-color:#3399FF;color:#FFFFFF;\"
data-colortramo=\"#D9EBFE\">/|/g" PVPC6.html > PVPC7.html
sed -e "s/<\/td><td class=\"filaTabla numero\" style=\"background-color:#D9EBFE;color:#005675;\"
data-colortramo=\"#D9EBFE\">/|/g" PVPC7.html > PVPC8.html
sed -e "s/<\/td><\/tr>/|/g" PVPC8.html > PVPC9.html

cat PVPC9.html | awk -F '|' '{ print $3 }' > listaPrecios.txt
cat PVPC9.html | awk -F '|' '{ print $4 }' > listaPreciosDH.txt

let HORA=0
let MINIMO_ACUMULADO=10
let MINIMO_TEMPORAL=10
let ANTERIOR=0
let ACTUAL=0
let MINIMO_HORA=0
for i in `cat listaPreciosDH.txt`; do
        echo $HORA $i
        if [ $HORA -gt 0 ]; then
                ANTERIOR=`echo $ANTERIOR | tr "," "."`
                ACTUAL=`echo $i | tr "," "."`
                MINIMO_TEMPORAL=`echo $ANTERIOR+$ACTUAL | bc`

#               echo --- $MINIMO_TEMPORAL --- $MINIMO_ACUMULADO
```

```
                ACTUAL=`echo ${MINIMO_ACUMULADO} \< ${MINIMO_TEMPORAL} | bc`
                if [ $ACTUAL -eq 0 ]; then
                        MINIMO_ACUMULADO=$MINIMO_TEMPORAL
                        MINIMO_HORA=$HORA
                        let MINIMO_HORA--
                fi
        fi
        ANTERIOR=$i
        let HORA+=1
done

echo "HORA DE COMIENZO: " $MINIMO_HORA "PRECIO DE DOS HORAS: " $MINIMO_ACUMULADO

curl --user $OH_USER:$OH_PASS --max-time 2 --connect-timeout 2 --header "Content-Type: text/plain"
--request PUT --data $MINIMO_HORA $OH_URL/rest/items/$OH_ITEM/state
```

Tabla 23: Script para conocer la franja horaria de consumo eléctrico más económica

3.4 Desarrollo/instalación de la interfaz de gestión

3.4.1 Instalación de openHAB

Una vez que el equipo ya tiene instalado y configurado tanto el sistema operativo, como la máquina virtual java, se puede proceder a la instalación de la herramienta principal de gestión openHAB. Se ha optado por descargar la última versión disponible (1.5.0), para aprovechar todas las funcionalidades implementadas, a pesar de que no es la versión estable según la web del proyecto. Durante la fase de pruebas no se han apreciado errores significativos que apoyen la idea de descartar esta versión frente a una inferior calificada como estable.

La descarga de la última compilación se hace desde la dirección *https://openhab.ci.cloudbees.com/job/openHAB*. Una vez descargado el software, los siguientes pasos son:

1. Descomprimir en /opt y renombrar a /opt/openhab-1.5.0-SNAPSHOT-FECHA sustituyendo FECHA por la fecha de descarga del software.

2. Crear un enlace simbólico /opt/openhab a /opt/openhab-1.5.0-SNAPSHOT-FECHA. Esto permitirá tener varias versiones de openhab sin necesidad de cambiar la configuración del servicio.

3. Modificar el script start.sh de openhab para que el puerto seguro de escucha de openhab sea el 443 como indica el RFC. Esto evitará posibles problemas de conectividad en redes externas. Además, hay que activar los PINs de la interfaz GPIO que se van a usar como salida (los emisores de RF y el diodo led). También se inicia una aplicación de escucha de RF que detallada con anterioridad. El script queda de la siguiente forma:

```
#!/bin/sh

cd `dirname $0`

# set path to eclipse folder. If local folder, use '.'; otherwise, use /path/to/eclipse/
eclipsehome="server";

# set ports for HTTP(S) server
HTTP_PORT=8080
HTTPS_PORT=443

PATH=$PATH:/usr/java/bin
JAVA_HOME=/usr/java

export PATH
export JAVA_HOME

/usr/bin/gpio mode 0 out
/usr/bin/gpio mode 1 out
/usr/bin/gpio mode 7 out

/usr/local/bin/receive433MHz &

/usr/bin/touch /opt/openhab/logs/events.log

# get path to equinox jar inside $eclipsehome folder
cp=$(find $eclipsehome -name "org.eclipse.equinox.launcher_*.jar" | sort | tail -1);

echo Launching the openHAB runtime...
java -Dosgi.clean=true -Declipse.ignoreApp=true -Dosgi.noShutdown=true -Djetty.port=$HTTP_PORT
-Djetty.port.ssl=$HTTPS_PORT -Djetty.home=. -Dlogback.configurationFile=configurations/logback.xml
-Dfelix.fileinstall.dir=addons -Djava.library.path=lib
-Djava.security.auth.login.config=./etc/login.conf -Dorg.quartz.properties=./etc/quartz
.properties -Dequinox.ds.block_timeout=240000 -Dequinox.scr.waitTimeOnBlock=60000
-Dfelix.fileinstall.active.level=4 -Djava.awt.headless=true -jar $cp $* -console
```

Tabla 24: Script de inicio de openHAB

4. El siguiente paso es configurar la aplicación. Para ello se utiliza la configuración de ejemplo que ofrece openHAB, que se colocará en el directorio /opt/openhab-configurations y se creará un enlace simbólico en /opt/openhab/configurations. De esta forma se puede cambiar la configuración con sólo reemplazar el enlace simbólico que se acaba de crear.

5. Copiar los addons necesarios al directorio /opt/openhab/addons. En este caso, el listado de los ficheros que contiene dicho directorio son:

```
org.openhab.action.mail-1.5.0-SNAPSHOT.jar
org.openhab.action.nma-1.5.0-SNAPSHOT.jar
org.openhab.action.prowl-1.5.0-SNAPSHOT.jar
org.openhab.action.squeezebox-1.5.0-SNAPSHOT.jar
org.openhab.action.twitter-1.5.0-SNAPSHOT.jar
org.openhab.action.xbmc-1.5.0-SNAPSHOT.jar
org.openhab.action.xmpp-1.5.0-SNAPSHOT.jar
org.openhab.binding.configadmin-1.5.0-SNAPSHOT.jar
org.openhab.binding.exec-1.5.0-SNAPSHOT.jar
org.openhab.binding.http-1.5.0-SNAPSHOT.jar
org.openhab.binding.milight-1.5.0-SNAPSHOT.jar
org.openhab.binding.mqtt-1.5.0-SNAPSHOT.jar
org.openhab.binding.mqttitude-1.5.0-SNAPSHOT.jar
org.openhab.binding.networkhealth-1.5.0-SNAPSHOT.jar
org.openhab.binding.ntp-1.5.0-SNAPSHOT.jar
org.openhab.binding.samsungtv-1.5.0-SNAPSHOT.jar
org.openhab.binding.tcp-1.5.0-SNAPSHOT.jar
org.openhab.io.dropbox-1.5.0-SNAPSHOT.jar
org.openhab.io.multimedia.tts.freetts-1.5.0-SNAPSHOT.jar
org.openhab.io.multimedia.tts.marytts-1.5.0-SNAPSHOT.jar
org.openhab.persistence.cosm-1.5.0-SNAPSHOT.jar
org.openhab.persistence.db4o-1.5.0-SNAPSHOT.jar
org.openhab.persistence.exec-1.5.0-SNAPSHOT.jar
org.openhab.persistence.gcal-1.5.0-SNAPSHOT.jar
org.openhab.persistence.logging-1.5.0-SNAPSHOT.jar
org.openhab.persistence.mqtt-1.5.0-SNAPSHOT.jar
org.openhab.persistence.rrd4j-1.5.0-SNAPSHOT.jar
```

Tabla 25: Listado de ficheros de complementos de openHAB

6. Instalar HABmin para la administración del entorno.
    1. Se descarga el software de *https://github.com/cdjackson/HABmin/archive/master.zip* y se descomprime en /opt/openhab/webapps/habmin.
    2. Se crean enlaces simbólicos en /opt/openhab/addons a los ficheros existentes en /opt/openhab/webapps/habmin/addons/
7. Se crea el fichero /etc/systemd/system/openhab.service para el servicio del sistema operativo con el contenido:

```
[Unit]
Description=OpenHAB Service
After=network.target

[Service]
ExecStart=/opt/openhab/start.sh

[Install]
WantedBy=multi-user.target
```

Tabla 26: Fichero de servicio de openHAB

8. Finalmente, se especifica que el servicio se inicie de forma automática con el comando systemctl enable openhab.service

La versión de HABmin probada ha sido la 0.1.2 y a pesar de que presenta funcionalidades interesantes como la visualización de gráficas mejorada frente a lo que ofrece openHAB, no dispone de la suficiente estabilidad como para dejar dicha herramienta funcionando más tiempo del necesario para configurar el entorno, por lo que una vez que se ha revisado la configuración de openHAB, se recomienda desinstalar HABmin.

La otra herramienta gráfica disponible para la configuración del entorno es el diseñador que trae openHAB. Es una aplicación multiplataforma basada en el editor de código Eclipse, y que permite modificar los diferentes ficheros de configuración necesarios para que openHAB funcione acorde a las necesidades de la vivienda.

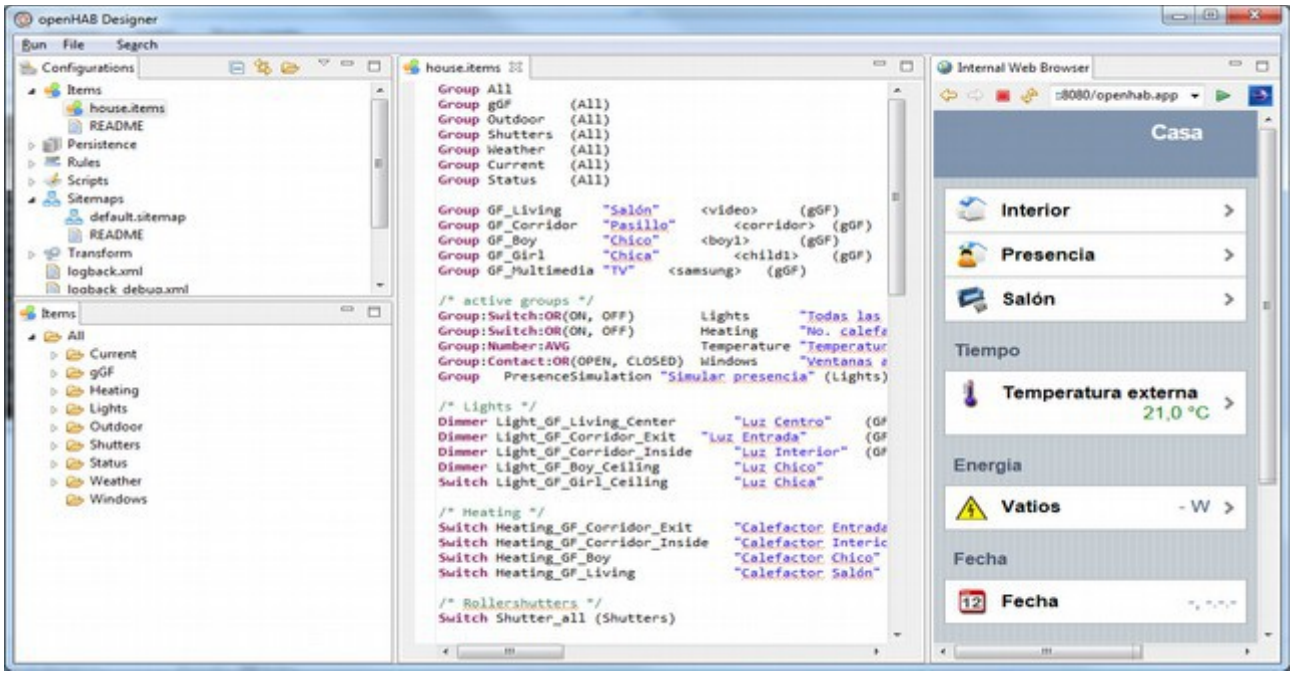

Ilustración 17: openHAB designer con los ficheros de configuración del proyecto y una muestra de la apariencia final

3.4.2 Configuración de openHAB

Todos los ficheros se encuentran dentro del directorio `configurations` ya mencionado con anterioridad. Los diferentes elementos que lo componen son:

3.4.2.1 Fichero openhab.cfg

El fichero openhab.cfg contiene la configuración de la persistencia, los datos de acceso a sistemas externos y, en general, cualquier dato de configuración del entorno. Cabe reseñar que en la configuración definitiva, algunos de estos parámetros se verán alterados, como por ejemplo, los relacionados al chequeo en los cambios de configuración; El parámetro "`folder:items=10,items`"

especifica que se chequee el fichero con la configuración de los elementos de la domótica cada 10 segundos en busca de cambios. Cuando el entorno se encuentre finalizado, este comportamiento se deshabilitará para mejorar el rendimiento. De forma análoga ocurre con el resto de ficheros de configuración definidos en openhab.cfg.

```
folder:items=10,items
folder:sitemaps=10,sitemap
folder:rules=10,rules
folder:scripts=10,script
folder:persistence=10,persist
security:option=ON
persistence:default=rrd4j
mainconfig:refresh=30
chart:provider=default
mail:hostname=smtp.gmail.com
mail:port=587
mail:username=XXXXXX@gmail.com
mail:password=XXXXXX
mail:from=XXXXXX@gmail.com
mail:tls=true
xmpp:servername=talk.google.com
xmpp:proxy=google.com
xmpp:port=5222
xmpp:username=XXXXXX@gmail.com
xmpp:password=XXXXXX
xmpp:consoleusers=angel.bueno@gmail.com,XXXXXX@gmail.com
gcal:username=XXXXXX@gmail.com
gcal:password=XXXXXX
gcal:url=https://www.google.com/calendar/feeds/XXXXXX@gmail.com/private/full
gcal:refresh=90000
logging:pattern=%date{ISO8601} - %-25logger: %msg%n
gcal-persistence:username=XXXXXX@gmail.com
gcal-persistence:password=XXXXXX
gcal-persistence:url=https://www.google.com/calendar/feeds/XXXXXX@gmail.com/private/full
ntp:hostname=es.pool.ntp.org
samsungtv:Livingroom.host=192.168.0.50
samsungtv:Livingroom.port=55000
milight:gateway.host=192.168.0.90
milight:gateway.port=8899
mqttitude:home.lat=XX.XXXXXX
mqttitude:home.lon=XX.XXXXXX
mqttitude:geofence=300
mqtt:broker.url=tcp://localhost:1883
mqtt:broker.clientId=openhab
mqtt:broker.user=XXXXXX
mqtt:broker.pwd=XXXXXX
mqtt:broker.qos=1
mqtt:broker.retain=true
```

Tabla 27: Fichero openhab.cfg

3.4.2.2 Fichero items/house.items

En este archivo se configuran los elementos que gestiona el sistema. Por tanto, aquí es donde hay que indicar los componentes (luces, persianas, calefactores, tv, etc.) y las acciones que gestionará el sistema.

```
Group All
Group gGF               (All)
Group Outdoor   (All)
Group Shutters  (All)
```

```
Group Weather   (All)
Group Current   (All)
Group Status    (All)

Group GF_Living         "SalÃ³n"        <video>         (gGF)
Group GF_Corridor       "Pasillo"               <corridor>      (gGF)
Group GF_Boy            "Chico"         <boy1>          (gGF)
Group GF_Girl           "Chica"                 <child1>        (gGF)
Group GF_Multimedia     "TV"    <samsung>       (gGF)
Group GF_Patio  "Patio"         <garage>        (gGF)

/* active groups */
Group:Switch:OR(ON, OFF)                Lights          "Todas las luces [(%d)]"
(All)
Group:Switch:OR(ON, OFF)                Heating         "No. calefactores encendidos [(%d)]"
<heating>               (All)
Group:Number:AVG                                Temperature     "Temperatura media [%.1f Â°C]"
<temperature>   (Status)
Group:Contact:OR(OPEN, CLOSED)  Windows         "Ventanas abiertas [(%d)]"
<contact>               (All)
Group   PresenceSimulationGroup "Simular presencia" (Lights)

/* Lights */
Dimmer Light_GF_Living_Center           "Luz Centro"    (GF_Living, Lights,
PresenceSimulationGroup)     { milight="gateway;7;brightness;27" }
Dimmer Light_GF_Corridor_Exit   "Luz Entrada"           (GF_Corridor, Lights,
PresenceSimulationGroup)   { milight="gateway;9;brightness;27" }
Dimmer Light_GF_Corridor_Inside         "Luz Interior"  (GF_Corridor, Lights,
PresenceSimulationGroup)   { milight="gateway;10;brightness;27" }
Dimmer Light_GF_Boy_Ceiling             "Luz Chico"             (GF_Boy, Lights,
PresenceSimulationGroup)                { milight="gateway;8;brightness;27" }
Switch Light_GF_Girl_Ceiling            "Luz Chica"             (GF_Girl, Lights,
PresenceSimulationGroup) { exec="OFF:/usr/local/bin/sendFan 0FFF00001000, ON:/usr/local/bin/sendFan
0FFF00001000" }

/* Fan */
Rollershutter Fan_GF_Girl_Ceiling       "Ventilador Chica"      (GF_Girl)
{ exec="UP:/usr/local/bin/sendFan 0FFF00000100, DOWN:/usr/local/bin/sendFan 0FFF00000001,
STOP:/usr/local/bin/sendFan 0FFF10000000" }

/* Wash Machine */
Number Wash_GF_Patio_Selection  "Lavadora" <garage>     (GF_Patio)
Switch Wash_GF_Patio            "Lavadora" <garage>     (GF_Patio)

/* Heating */
Switch Heating_GF_Corridor_Exit         "Calefactor Entrada"    <heating>       (GF_Corridor,
Heating)  { exec="OFF:/usr/local/bin/send1 1111100100 1 0, ON:/usr/local/bin/send1 1111100100 1
1" }
Switch Heating_GF_Corridor_Inside       "Calefactor Interior"   <heating>       (GF_Corridor,
Heating)  { exec="OFF:/usr/local/bin/send1 1111100100 3 0, ON:/usr/local/bin/send1 1111100100 3
1" }
Switch Heating_GF_Boy                           "Calefactor Chico"      <heating>       (GF_Boy,
Heating)              { exec="OFF:/usr/local/bin/send1 1111100100 2 0, ON:/usr/local/bin/send1
1111100100 2 1" }
Switch Heating_GF_Living                        "Calefactor SalÃ³n"     <heating>       (GF_Living,
Heating)    { exec="OFF:/usr/bin/gpio write 7 0, ON:/usr/bin/gpio write 7 1" }

/* Rollershutters */
Switch Shutter_all (Shutters)
Rollershutter Shutter_GF_Living                 "Persiana SalÃ³n"       <rollershutter> (GF_Living,
Shutters)   { exec="UP:/usr/local/bin/sendBlind 1, DOWN:/usr/local/bin/sendBlind 0,
STOP:/usr/local/bin/sendBlind 2" }
Rollershutter Shutter_GF_Boy                    "Persiana Chico"        <rollershutter> (GF_Boy,
Shutters)

/* Indoor Temperatures */
Number Temperature_GF_Corridor  "Temperatura [%.1f Â°C]"        <temperature>   (Temperature,
```

```
GF_Corridor)
Number Temperature_GF_Living    "Temperatura [%.1f Â°C]"        <temperature>   (Temperature,
GF_Living)
Number Temperature_GF_Boy       "Temperatura [%.1f Â°C]"        <temperature>   (Temperature,
GF_Boy)
Number Temperature_GF_Girl              "Temperatura [%.1f Â°C]"        <temperature> (Temperature,
GF_Girl)

/* Windows */
//Contact Window_GF_Frontdoor   "Frontdoor [MAP(en.map):%s]"            (GF_Corridor, Windows)
//Contact Window_GF_Living              "Terrace door [MAP(en.map):%s]"         (GF_Living,
Windows)
//Contact Window_GF_Girl                        "Bedroom [MAP(en.map):%s]"
(GF_Girl, Windows)
//Contact Garage_Door                   "Garage Door [MAP(en.map):%s]"          (Outdoor, Windows)

/* Weather */
Group Weather_Chart
(Weather)
Number Weather_Temperature              "Temperatura externa [%.1f Â°C]"        <temperature>
(Weather_Chart) { http="<[http://weather.yahooapis.com/forecastrss?
w=20079786&u=c:60000:XSLT(yahoo_weather_temperature.xsl)]" }
Number Weather_Temp_Max                 "MÃ¡ximo de hoy [%.1f Â°C]"     <temperature>
(Weather_Chart)
Number Weather_Temp_Min                 "MÃnimo de hoy [%.1f Â°C]"     <temperature>
(Weather_Chart)
Number Weather_Chart_Period             "GrÃ¡fico"
DateTime Weather_LastUpdate             "Ãltima actualizaciÃ³n [%1$ta %1$tR]"   <clock>

/* Current */
Group Current_GF_Chart  (Current)
Number  Current_GF      "Vatios [%.1f W]"       <energy>        (Current_GF_Chart)
Number  Current_GF_Time "Hora de menor gasto [%d]"      <clock>
Number  Current_GF_Max  "Vatios max. [%.1f W]"  <energy>        (Current_GF_Chart)
Number  Current_GF_Min  "Vatios min. [%.1f W]"  <energy>        (Current_GF_Chart)
Number  Current_GF_Chart_Period         "GrÃ¡fico"

/* TV */
Switch  TV_GF_Multimedia_TV_power       "Encendido"     (GF_Multimedia)
{ exec="ON:/usr/local/bin/samsungTvStart.sh, OFF:/bin/true",
samsungtv="OFF:Livingroom:KEY_POWEROFF, ON:Livingroom:KEY_POWERON" }
Rollershutter   TV_GF_Multimedia_TV_channel     "Canal" <video> (GF_Multimedia)
{ samsungtv="UP:Livingroom:KEY_CHUP, DOWN:Livingroom:KEY_CHDOWN, STOP:Livingroom:KEY_CH_LIST" }
Rollershutter   TV_GF_Multimedia_TV_Volume      "Volumen"       (GF_Multimedia)
{ samsungtv="UP:Livingroom:KEY_VOLUP, DOWN:Livingroom:KEY_VOLDOWN, STOP:Livingroom:KEY_MUTE" }

/* NTP binding demo item */
DateTime                Date                    "Fecha [%1$tA, %1$td.%1$tm.%1$tY]"      <calendar>
{ ntp="Europe/Madrid:es_ES" }

/* Outside Light items */
String strSunset "Atardecer [%s]" <clock> (Weather)
{ http="<[http://api.wunderground.com/api/XXXXXX/astronomy/q/zmw:00000.1.08433.xml:3600000:XSLT(wun
derground_sunset.xsl)]" }
String strSunrise "Amanecer [%s]" <clock> (Weather)
{ http="<[http://api.wunderground.com/api/XXXXXX/astronomy/q/zmw:00000.1.08433.xml:3600000:XSLT(wun
derground_sunrise.xsl)]" }
Switch  noche "Noche" <moon>

/* Automatic items */
Number Scene_General                    "SalÃ³n"                         <sofa>
Switch  Presence_ABP    "Angel @ Home"  <present> (Outdoor,Status)
{ mqttitude="broker:owntracks/XXXXXX:casa" }
Switch  Presence_ABP_work       "Angel @ Work"  <present> (Outdoor,Status)
{ mqttitude="broker:owntracks/XXXXXX:trabajo" }
Switch   PresenceSimulation      "Presencia siulada" <present> (PresenceSimulationGroup)
```

Tabla 28: Fichero house.items

Se puede observar como hay definidos grupos (un elemento puede pertenecer a varios grupos) para que la gestión se pueda llevar a cabo de una forma jerarquizada. Otro punto a reseñar en este apartado, son las acciones que se ejecutan con cada elemento. Algunas de ellas vienen definidas por el elemento de openHAB asociado (por ejemplo el que gestiona sistemas de luces milight) y otras son aplicaciones que se encuentran en la propia máquina. Dichos ejecutables han sido creados explícitamente para este entorno, y se han comentado en el apartado "Instalación de las herramientas software necesarias".

3.4.2.3 Fichero rules/house.rules

Este archivo contiene diferentes reglas de negocio como son:

```
/**
 * Automatic heat system
 */
rule "Heat for living room"
        when
                Item Heating_GF_Living_Automation received command or
                Item Heating_GF_Living_Automation received update or
                Item Presence_ABP received command or
                Item Presence_ABP received update or
                Item Temperature_GF_Living received update
        then
                if (Heating_GF_Living_Automation.state==ON) {
                        if (Presence_ABP.state==OFF || Temperature_GF_Living.state>23)
sendCommand(Heating_GF_Living,OFF)
                        if (Presence_ABP.state==ON && Temperature_GF_Living.state<=23)
sendCommand(Heating_GF_Living,ON)
                }
end
```

Tabla 29: Regla de negocio para activar el calefactor del salón de forma automática

Con esta regla, en caso de que se habilite la opción de calefacción automática en la sala de estar, si el propietario se encuentra en la vivienda y hay una temperatura igual o inferior a 23 grados se encenderá el radiador. De esta forma se cumple con el objetivo "Los radiadores se deben poder encender de forma remota desde el sistema de domótica mediante la interacción del usuario, o bien atendiendo a los parámetros de temperatura y cercanía a la vivienda."

```
/* Rule to act at sunrise and sunset time */
rule "React to sunset and sunrise"
when
        Time cron "0 0 16 * * ?"   // Every day 16:00 hours, evaluate sunset
then
    var year   = now.getYear
    var month  = now.getMonthOfYear
    var day    = now.getDayOfMonth
```

```
    var datum  = year+"-"+month+"-"+day+" "+strSunset.state
    var datumSunrise  = year+"-"+month+"-"+day+" "+strSunrise.state
    logInfo("Sunset","datum = " + datum)
    logInfo("Sunrise","datum = " + datumSunrise)
    var DateTime sunset = parse(year+"-"+month+"-"+day+"T"+strSunset.state)
    var DateTime sunrise = parse(year+"-"+month+"-"+day+"T"+strSunrise.state)

     // Cancel timer to avoid reschedule
    if(tIndoorLightsOn!=null) {
        logInfo("Sunset","Timer tIndoorLightsOn cancelled")
        tIndoorLightsOn.cancel()
    }
    if(tIndoorLightsOff!=null) {
        logInfo("Sunrise","Timer tIndoorLightsOff cancelled")
        tIndoorLightsOff.cancel()
    }
    logInfo("Sunset","Timer tIndoorLightsOn created")
    tIndoorLightsOn = createTimer(sunset.minusMinutes(15)) [|
        logInfo("Sunset","Timer tIndoorLightsOn executed")
        postUpdate(noche, ON)
     if (Scene_General!=0) postUpdate(Scene_General,Scene_General.state);
    ]
    logInfo("Sunrise","Timer tIndoorLightsOff created")
    tIndoorLightsOn = createTimer(sunrise.plusMinutes(15)) [|
        logInfo("Sunrise","Timer tIndoorLightsOff executed")
        postUpdate(noche, OFF)
     if (Scene_General!=0) {
         sendCommand(Light_GF_Living_Center,0)

     }
    ]
end
```

Tabla 30: Regla de negocio con acciones a realizar en la puesta y salida de sol

Se crea una tarea programada que se ejecuta todos los días a las 16:00 horas. Dicha tarea lanzará 15 minutos antes de la puesta de sol un proceso que activará el elemento noche (elemento con el que el sistema controlará si es de noche). Además, dependiendo de la escena seleccionada para el salón (apagado, TV, cena o lectura), actualizará el estado de la escena, para que se enciendan o se apaguen las luces según corresponda. Esta regla también permite subir o bajar las persianas según sea de día o de noche, simplemente agregando la sentencia ***sendCommand(Shutter_GF_Living,UP)*** para subirla, o ***sendCommand(Shutter_GF_Living,DOWN)*** para bajarla. De esta forma se cumple con el objetivo "Las persianas se deben poder subir/bajar de forma centralizada mediante una intervención manual, o bien programada por el usuario atendiendo a criterios horarios."

La regla definida a continuación sirve para cumplir el objetivo: "La lavadora se debe poder encender de forma manual, o bien atendiendo a criterios de consumo energético, de forma que se active en la franja horaria de menor coste monetario". Hay una selección, para desactivar el enchufe de la lavadora, activarlo y que la lavadora funcione en ese momento, y una última opción para programar el encendido de la lavadora en la franja horaria en la que el precio del Kwh sea el mínimo del día.

```
/* Wash Machine activation rule */
```

```
rule "Wash Machine"
        when
                Item Wash_GF_Patio_Selection received update
        then
                if (Wash_GF_Patio_Selection.state==2) {
                        var year   = now.getYear
                        var month  = now.getMonthOfYear
                        var day    = now.getDayOfMonth + 1
                        var datum  = year+"-"+month+"-"+day+" "+Current_GF_Time.state+":00:00"
                        var DateTime washTime =
parse(year+"-"+month+"-"+day+"T"+Current_GF_Time.state+":00:00")
                        var endTime = (Current_GF_Time.state as DecimalType).intValue + 2
                        var datumEnd  = year+"-"+month+"-"+day+" "+endTime+":00:00"
                        var DateTime washTimeOff =
parse(year+"-"+month+"-"+day+"T"+endTime+":00:00")
                        logInfo("Wash Machine switch on at ","datum = " + datum)
                        logInfo("Wash Machine switch off at ","datum = " + datumEnd)
                        if (tWashMachineOn!=null) {
                                logInfo("Wash Machine","Timer tWashMachineOn cancelled")
                                tWashMachineOn.cancel()
                        }
                        if (tWashMachineOff!=null) {
                                logInfo("Wash Machine","Timer tWashMachineOff cancelled")
                                tWashMachineOff.cancel()
                        }
                        tWashMachineOn = createTimer(washTime) [|
                                logInfo("Wash Machine","Timer tWashMachineOn executed")
                                postUpdate(Wash_GF_Patio, ON)
                                tWashMachineOff = createTimer(washTimeOff) [|
                                        logInfo("Wash Machine","Timer tWashMachineOff executed")
                                        postUpdate(Wash_GF_Patio, OFF)
                                        postUpdate(Wash_GF_Patio_Selection, 0)
                                ]
                        ]
                } else if (Wash_GF_Patio_Selection.state==1) {
                        postUpdate(Wash_GF_Patio, ON)
                } else if (Wash_GF_Patio_Selection.state==0) {
                        postUpdate(Wash_GF_Patio, OFF)
                        if (tWashMachineOn!=null) {
                                logInfo("Wash Machine","Timer tWashMachineOn cancelled by off
selection")
                                tWashMachineOn.cancel()
                        }
                        if (tWashMachineOff!=null) {
                                logInfo("Wash Machine","Timer tWashMachineOff cancelled by off
selection")
                                tWashMachineOff.cancel()
                        }
                }
end
```

```
/**
 * Rule to set different scenes for the living room
 */
 rule "Select Living Scene"
        when
                Item Scene_General received command
        then
                switch(receivedCommand) {
                        case 0 : {
                               sendCommand(Light_GF_Living_Center,0)
                               if (TV_GF_Multimedia_TV_power != OFF)
sendCommand(TV_GF_Multimedia_TV_power,OFF)
                          }
                        case 1 : {
```

```
                                if (TV_GF_Multimedia_TV_power != ON)
sendCommand(TV_GF_Multimedia_TV_power,ON)
                                if (noche == ON) sendCommand(Light_GF_Living_Center,10)
                            }
                        case 2 : {
                                if (TV_GF_Multimedia_TV_power != OFF)
sendCommand(TV_GF_Multimedia_TV_power,OFF)
                                if (noche == ON) sendCommand(Light_GF_Living_Center,100)
                            }
                        case 3 : {
                                if (TV_GF_Multimedia_TV_power != OFF)
sendCommand(TV_GF_Multimedia_TV_power,OFF)
                                if (noche == ON) sendCommand(Light_GF_Living_Center,100)
                            }
                    }
end
```

Tabla 31: Regla para configurar diferentes esquemas de ambiente en el salón

El elemento “Scene_General” puede tener cuatro estados. El primero de ellos es apagado. En ese estado se apaga la luz y la TV del salón. El segundo es TV. En este estado, si es de noche, la luz se enciende a un 10% de intensidad y se activa también la TV. En el tercer y cuarto caso, la televisión se apaga y la luz se enciende al 100% en caso de que sea de noche. Esta regla cumple con los criterios definidos para acondicionar el salón según la situación.

```
rule "Update max and min temperatures"
when
    Item Weather_Temperature changed or
    Time cron "0 0 0 * * ?" or
    System started
then
    postUpdate(Weather_Temp_Max, Weather_Temperature.maximumSince(now.toDateMidnight).state)
    postUpdate(Weather_Temp_Min, Weather_Temperature.minimumSince(now.toDateMidnight).state)
end

rule "Update max and min current"
when
    Item Current_GF changed or
    Time cron "0 0 0 * * ?" or
    System started
then
    postUpdate(Current_GF_Max, Current_GF.maximumSince(now.toDateMidnight).state)
    postUpdate(Current_GF_Min, Current_GF.minimumSince(now.toDateMidnight).state)
end
```

Tabla 32: Reglas de negocio para actualizar los valores máximo y mínimo de la temperatura y la corriente

Con estas dos reglas se actualizan los valores máximo y mínimo de la temperatura exterior y del consumo energético en el día actual.

3.4.2.4 Ficheros de transformación

Algunos elementos permiten obtener páginas de información mediante protocolo http, y aplicarle una hoja de estilos xsl para conseguir un determinado campo de la página visitada. Con este sistema

se permite conocer el tiempo exterior de la localidad, la hora de amanecer o la de anochecer. Los ficheros de transformación configurados son:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

        <xsl:output indent="yes" method="xml" encoding="UTF-8" omit-xml-declaration="yes" />

        <xsl:template match="/">
                <!-- format: hh:mm:ss -->
                <xsl:value-of select="//sunrise/hour/text()" /><xsl:text>:</xsl:text><xsl:value-of
select="//sunrise/minute/text()" />
        </xsl:template>

</xsl:stylesheet>
```

Tabla 33: wunderground_sunrise.xsl para conocer la hora de amanecer

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

        <xsl:output indent="yes" method="xml" encoding="UTF-8" omit-xml-declaration="yes" />

        <xsl:template match="/">
                <!-- format: hh:mm:ss -->
                <xsl:value-of select="//sunset/hour/text()" /><xsl:text>:</xsl:text><xsl:value-of
select="//sunset/minute/text()" />
        </xsl:template>

</xsl:stylesheet>
```

Tabla 34: wunderground_sunset.xsl para conocer la hora de anochecer

```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0" version="1.0">

    <xsl:output indent="yes" method="xml" encoding="UTF-8" omit-xml-declaration="yes" />

    <xsl:template match="/">
        <xsl:value-of select="//item/yweather:condition/@temp" />
    </xsl:template>

</xsl:stylesheet>
```

Tabla 35: yahoo_weather_temperature.xsl para conocer la temperatura externa

3.4.2.5 Configuración del sitio

La visualización de los diferentes elementos se determina en el fichero sitemaps/default.sitemap cuyo contenido es

```
sitemap Default label="Casa"
{
        Frame {
```

```
                Group item=gGF label="Interior" icon="house"
                Group item=Outdoor label="Presencia" icon="present"
                Selection item=Scene_General mappings=[0=Off, 1=TV, 2=Cena, 3=Lectura]
                Selection item=Wash_GF_Patio_Selection mappings=[0=Off, 1=Manual, 2=Programado]
        }
        Frame label="Tiempo" {
                Text item=Weather_Temperature
valuecolor=[Weather_LastUpdate=="Uninitialized"="lightgray",Weather_LastUpdate>90="lightgray",>25="
orange",>15="green",>5="orange",<=5="blue"] {
                        Frame {
                                Text item=Weather_Temp_Max
valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]
                                Text item=Weather_Temp_Min
valuecolor=[>25="orange",>15="green",>5="orange",<=5="blue"]
                                Text item=Weather_LastUpdate visibility=[Weather_LastUpdate>30]
valuecolor=[Weather_LastUpdate>120="orange", Weather_LastUpdate>300="red"]
                        }
                        Frame {
                                Switch item=Weather_Chart_Period label="GrÃ¡fico"
mappings=[0="Hora", 1="DÃa", 2="Semana"]
                                Chart item=Weather_Chart period=h refresh=600
visibility=[Weather_Chart_Period==0, Weather_Chart_Period=="Uninitialized"]
                                Chart item=Weather_Chart period=D refresh=3600
visibility=[Weather_Chart_Period==1]
                                Chart item=Weather_Chart period=W refresh=3600
visibility=[Weather_Chart_Period==2]
                        }
                }
        }
        Frame label="EnergÃa" {
                Text item=Current_GF {
                        Frame {
                                Text item=Current_GF_Time icon="energy"
                                Switch item=Current_GF_Chart_Period label="GrÃ¡fico"
mappings=[0="Hora", 1="DÃa", 2="Semana"]
                                Chart item=Current_GF_Chart period=h refresh=600
visibility=[Current_GF_Chart_Period==0, Current_GF_Chart_Period=="Uninitialized"]
                                Chart item=Current_GF_Chart period=D refresh=3600
visibility=[Current_GF_Chart_Period==1]
                                Chart item=Current_GF_Chart period=W refresh=3600
visibility=[Current_GF_Chart_Period==2]
                        }
                }
                Text item=Current_GF_Time
        }
        Frame label="Fecha" {
                Text item=Date
                Text item=strSunrise
                Text item=strSunset
        }
}
```

Tabla 36: Fichero default.sitemap

3.4.2.6 Configuración de la persistencia

Uno de los objetivos del proyecto era poder almacenar el comportamiento habitual de los elementos de la vivienda, como la luz o la televisión, para poder repetirlo en períodos de ausencia, y simular así que la vivienda no se encuentra vacía. Esta característica se llama *persistencia* en openHAB, y permite guardar las acciones en diferentes canales. Uno de ellos es el calendario de

google, con el que también se pueden ejecutar acciones programadas.

Además de la simulación de presencia, la persistencia también permite mostrar gráficos con la evolución de determinados datos, como la temperatura o el consumo de corriente. Existe un fichero de persistencia por cada sistema de almacenamiento.

```
// persistence strategies have a name and a definition and are referred to in the "Items" section
Strategies {
    // for rrd charts, we need a cron strategy
    everyMinute : "0 * * * * ?"
}

Items {
    NoOfLights,Heating* : strategy = everyChange, everyMinute, restoreOnStartup

    // let's only store temperature values in rrd
    Temperature*,Weather_Chart*,Current_GF_Chart* : strategy = everyMinute, restoreOnStartup
}
```

Tabla 37: Fichero persistence/rrd4j.persist

```
// Persistence strategies have a name and a definition and are referred to in the "Items" section
Strategies {
    // If no strategy is specified for an item entry below, the default list will be used.
    default = everyChange
}

Items {
    PresenceSimulationGroup* : strategy = everyChange
}
```

Tabla 38: Fichero persistence/gcal.persist

En el caso de la persistencia gcal, se almacenarán las acciones de los elementos que pertenezcan al grupo PresenceSimulationGroup.

3.4.3 Supervisión del sistema

El sistema operativo dispone de un *perro guardián* que detecta posibles incidencias en el equipo y lo reinicia en caso de que no responda. La limitación que tiene este elemento es que no detecta que los diferentes servicios se encuentren operativos y los reinicie en tal caso. Por tanto, se ha creado un script que desempeñe esta función y se ha programado con cron para que se ejecute de forma periódica. El script /usr/local/bin/watchdog-openhab.sh es

```
#!/bin/bash

SERVICE=`systemctl status openhab | grep "Active:" | grep running`
RESULT=$?

EVENTS=`find /opt/openhab/logs/events.log -mmin -10`

#date >> /var/log/watchdog-openhab.log

if [ $RESULT -ne 0 ]; then
        date | mail -s "Error en openhab" angel.bueno@gmail.com
        systemctl restart openhab
fi
```

```
SERVICE=`systemctl status mosquitto | grep "Active:" | grep running`
RESULT=$?
if [ $RESULT -ne 0 ]; then
        date | mail -s "Error en mosquitto" angel.bueno@gmail.com
        systemctl restart mosquitto
        systemctl restart openhab
fi

if [ -z $EVENTS ]; then
        date | mail -s "Error en los eventos openhab" angel.bueno@gmail.com
        systemctl restart openhab
```

Tabla 39: Script para controlar que openHAB se encuentre operativo

Se verifica que el servicio openHAB esté operativo, que se registren eventos como mínimo cada 10 minutos y que el servicio mosquitto también esté funcionando. Con cualquiera de las anteriores anomalías reiniciaría el servicio correspondiente y enviaría un correo electrónico de alerta.

4 Fase de pruebas

4.1 Control de dispositivos

El control de todos los dispositivos propuestos se hace adecuadamente. Durante la fase de estudio se ha observado que ocasionalmente el equipo, debido a sus reducidas capacidades de cpu y memoria, ha producido un elevado retardo que provoca una mala interpretación por parte del usuario, al esperar una reacción inmediata a su solicitud.

Se han comprobado de forma satisfactoria el control de cada uno de los elementos de forma manual, tanto los de encendido/apagado, como los numéricos y los de ajuste (intensidad de luz).

4.2 Detección correcta de las señales de entrada

Se ha verificado el correcto funcionamiento de las señales de entrada:

- Monitor de consumo eléctrico
- Temperatura
- Control horario
- Ubicación

Atendiendo a los parámetros requeridos tal y como se solicitaba en los objetivos del proyecto.

4.3 Chequeo de las reglas de negocio

Las reglas de negocio también han sido validadas:

- Registro de actividad de los elementos. Se almacena la actividad de los elementos en un calendario de google para poder activarlos posteriormente y simular la presencia en la vivienda.

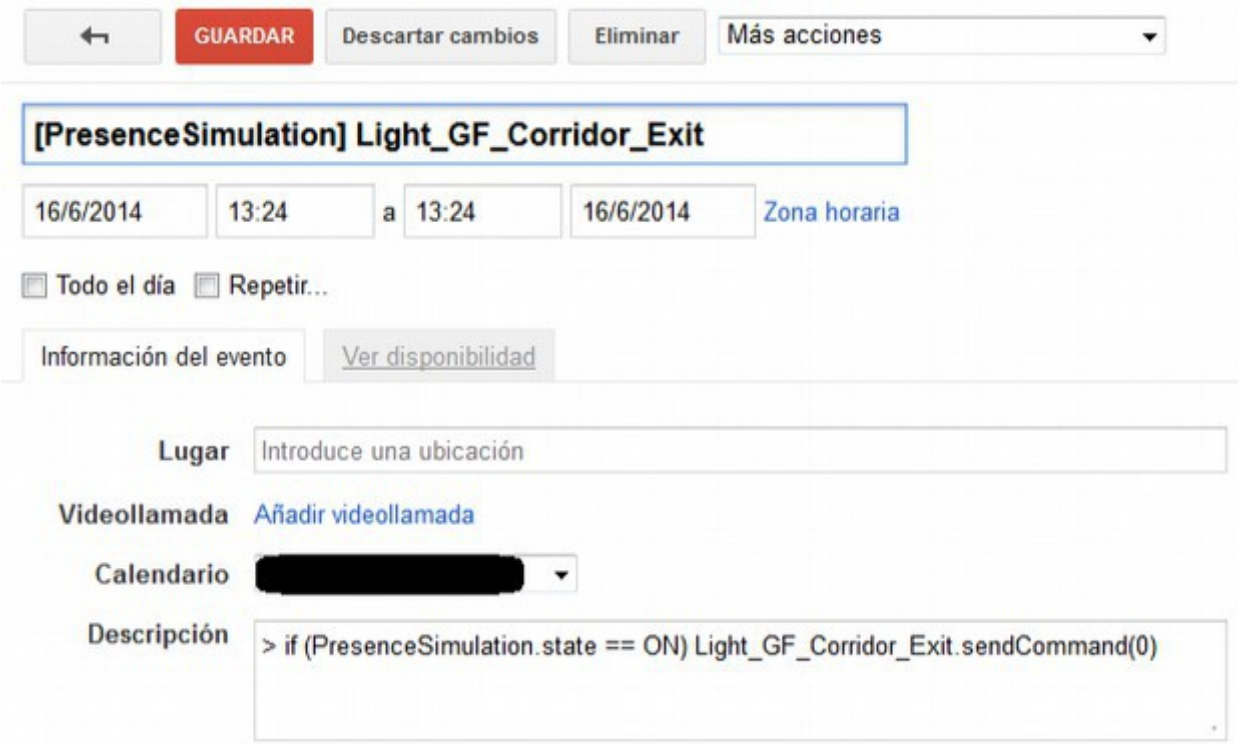

Ilustración 18: Evento creado automáticamente a partir de la actividad cotidiana para simular la presencia

- Control de temperatura automatizado. Se verifica que la calefacción se enciende cuando la ubicación del smartphone configurado se encuentra en la vivienda y la temperatura baja del umbral definido.
- Control de persiana horario. Se comprueba el correcto funcionamiento de la subida/bajada de persiana con respecto a la hora de amanecer y puesta de sol. Esta hora se obtiene de internet.
- Lavadora. Se comprueba que el encendido de la lavadora se produce en la franja horaria más económica (de dos horas correlativas) que ofrece la web de Red Eléctrica Española "http://www.esios.ree.es/" cuando se indica que se active de forma programada.
- Escenario. Se verifica que la televisión y las luces se controlan de forma automatizada en función del escenario seleccionado, acorde a las indicaciones definidas en la regla de negocio.
- Acceso remoto. Todo el entorno se ha comprobado que funciona desde una aplicación específica para smartphone, y desde un entorno web.

5 Valoración económica

Es necesario indicar que el objetivo de este proyecto es el de controlar elementos que ya se encuentren en la vivienda, por lo que sólo se van a valorar aquellos equipos/software que sea necesario para la integración de los sistemas.

Cantidad	Elemento	Coste
1	Raspberry Pi	38,95
1	Tarjeta SD Clase 10	6,95
1	Cargador de red microUSB	4,25
1	Cable HDMI macho-macho	2,75
1	Emisor 433Mhz	4,83
1	Receptor 433Mhz	6,17
1	Emisor 315Mhz	2,99
1	Arduino Uno	4,89
1	Adaptador con relé para Xbee	7,37
1	Adaptador Xbee para puerto USB	6,07
2	Módulo Xbee 1mW	62,38
1	Adaptador Wifi para puerto USB	12,48
1	Cables, conectores y otros	2,00
1	Juego de enchufes radio-controlados	15,20
	TOTAL	177,28

6 Conclusiones

6.1 Objetivos

El proyecto ha cumplido con los objetivos solicitados dado que el sistema permite:

- Controlar el consumo energético de la vivienda, almacenando dicha información para que pueda ser tratada; para este propósito se utiliza un medidor de consumo energético doméstico que envía la información por radio frecuencia y que es recopilada por el sistema instalado.
- Conocer en cada momento si es de día o de noche (según la hora de amanecer y atardecer); de esta forma se puede gestionar de forma inteligente la luz de las habitaciones, acorde a la situación de la luz solar.
- Medir la temperatura externa y de al menos una habitación. Con este dato se puede controlar el encendido automático de los calefactores que carecen de termostato.
- Controlar la ubicación del propietario de la vivienda para automatizar determinadas acciones, como por ejemplo el encendido de la calefacción en caso de que el usuario se encuentre en las inmediaciones de la vivienda, y desactivarla en caso contrario (teniendo en cuenta también la temperatura de la vivienda)
- Permite el encendido y apagado remoto de elementos eléctricos que usen un enchufe normal (calefactores, lavadora,etc.), con el uso de enchufes radio-controlados. Hay que tener en cuenta que el elemento a gestionar debe tener una potencia inferior a la que pueda gestionar el enchufe.
- También es capaz de controlar motores de persianas gestionados por radio frecuencia.
- De la misma forma, también permite interactuar con un ventilador de techo mediante radio-frecuencia; pudiendo enviar cualquiera de los comandos que tiene el mando a distancia de dicho elemento.
- Interactuar con lámparas led controladas vía ethernet; Esta funcionalidad permite que las luces se puedan encender/apagar, controlar la intensidad y el color de forma centralizada.
- También permite gestionar la televisión, conociendo en cada momento el estado de la misma, sin necesidad de que el encendido sea a través del sistema de domótica.
- Por último, la gestión del entorno se puede hacer desde cualquier equipo con conexión a internet y un navegador web, ofreciendo la posibilidad de utilizar una aplicación específica

para *smartphone*.

6.2 Mejoras

Una vez que se han cumplido los objetivos, se ha incluido las siguientes mejoras:

- Incorporar un receptor para detectar activaciones de los enchufes radio-controlados y conocer el estado de cada enchufe sin necesidad de que sea activado/desactivado a través del sistema de domótica.
- Controlar el sistema de televisión sin necesidad de utilizar mando de infrarrojos, conociendo en cada momento el estado de encendido/apagado a través del cable HDMI.
- Conocer la franja horaria de menor coste energético para poder activar equipos eléctricos de alto consumo energético y así reducir la factura eléctrica.
- Se ha incorporado un sistema de "autovigilancia" que reinicie el entorno de forma automática en caso de fallo.

6.3 Problemas detectados

Se han detectado los siguientes puntos débiles en el sistema:

- El equipo destinado al control de la domótica es de bajo coste, con unos recursos bastante limitados, para los servicios que tiene instalados. Esto provoca que ocasionalmente se vea desbordado y ralentice la gestión de los diversos elementos, principalmente los que dependen de una base de tiempo como son los radio-controlados (los pulsos se tienen que enviar en tiempo real con una base de tiempo específica).
- Si bien, se ha implementado un receptor para las señales de 433Mhz, no se ha llevado a cabo de la misma forma para las señales de 315Mhz, por lo que el sistema de domótica no tiene conocimiento del estado del ventilador radio-controlado en caso de que se gestione desde su mando autónomo.
- Los equipos de radiofrecuencia no envían un retorno para confirmar la recepción de la señal, por lo que no se puede asegurar que una señal enviada ha sido recibida. Esto puede provocar que el sistema piense que se ha encendido un radiador o se ha apagado cuando realmente esto no sea así.
- Los equipos de radiofrecuencia no disponen de un protocolo de acceso al medio. Este inconveniente puede provocar interferencias que provoquen una denegación de servicio en un momento dado. A pesar de que los diversos elementos usan pulsos de diferente longitud, al no existir un protocolo que gestione el acceso al medio, si hay dos emisores enviando una

señal al mismo tiempo, ninguna de las dos señales sea recibida correctamente.

- La sonda de temperatura está construida con elementos de comunicación más “inteligentes” que una simple señal de radiofrecuencia, pero en contra suponen un gasto desproporcionado para la funcionalidad que aportan, por lo que es conveniente estudiar un método alternativo de envío de dicha señal mediante elementos más sencillos de radiofrecuencia, de la misma forma como realizan las estaciones meteorológicas.
- La seguridad del sistema, en cuanto a las comunicaciones de los equipos que usan señales de radio frecuencia simples, es prácticamente nula; por lo que su uso no se aconseja en dependencias que puedan ser colindantes con otras propiedades. Esta baja seguridad permitiría en un momento dado que una persiana o un radiador se active por una persona ajena a la instalación, o que provoque una denegación de servicio. Efectos indeseados en cualquier caso.
- Copias de seguridad. Inicialmente no se ha contemplado, pero es conveniente llevar a cabo un volcado de la configuración cada vez que cambie. OpenHAB permite realizar esta tarea con servicios de almacenamiento en la nube como dropbox, aunque en este proyecto no se ha configurado.

6.4 Propuesta de mejoras

- Una mejora es la modificación del sistema de detección de temperatura. Actualmente sólo hay uno, por lo que no abarca de forma individual cada una de las habitaciones. Es recomendable controlar la temperatura en cada una de las habitaciones por separado, añadiendo más sondas.
- Uno de los aspectos negativos del proyecto actualmente es la inseguridad y poca fiabilidad del canal de comunicaciones usado por los elementos de radio-frecuencia. En consecuencia, otra mejora sustancial sería reemplazar esta comunicación por un protocolo seguro y fiable como zigbee basado en el estándar IEEE 802.15.4 (con elementos Xbee como los usados para la lectura de la temperatura y control de relés); aunque hay que destacar que esta medida sería muy intrusiva, al menos para elementos como el motor de las persianas que contiene el módulo de recepción de radio-frecuencia integrado. En caso de no optar por esta vía, en la que se replantearían todas las comunicaciones de los diversos elementos para usar un canal fiable y seguro, habría que tener en consideración las siguientes mejoras:
    - Optimizar el código encargado de la recepción de pulsos en la frecuencia de 433Mhz. Este código implementa una interrupción, provocando muchos cambios de contexto en momentos en los que se reciban muchas señales de esta frecuencia que deriva en una penalización en el uso de CPU.

- Incorporar un receptor de 315Mhz para detectar las señales que se envíen al ventilador de forma ajena al sistema de domótica, y poder conocer en cada momento el estado de dicho elemento.
- Crear nuevas sondas de temperatura con un canal de comunicaciones más económico, a pesar de que sea menos fiable.

6.5 Resumen

El sistema cumple los requisitos, por lo que queda demostrado que implantar un sistema de domótica reutilizando elementos electrónicos de bajo coste es posible, gracias a potentes herramientas de software que facilitan dicha tarea como es el caso del proyecto openHAB y de la misma forma a proyectos de hardware de bajo coste como Raspberry Pi. Ambos proyectos son el pilar del presente documento, que junto con el análisis de las señales de radiofrecuencia de los elementos a gestionar, han permitido implementar un pequeño entorno de domótica cuyo coste en un producto que ofrezca similares funcionalidades hubiera sido varias órdenes de mangintud superior.

Es posible por tanto, implementar un sistema de gestión centralizada que utilice elementos autónomos, eliminando de esta forma la criticidad del sistema de domótica, pues cada equipo se puede seguir controlando de forma independiente. Esta centralización permite dotar de “inteligencia” a los equipos, para que trabajen de forma coordinada, en base a las entradas que dispone el sistema como son el coste energético o la luz del día.

7 Bibliografía

1: Enrico Nicoletti; Proyecto Freedomotic; ; [2013?]; http://www.freedomotic.com/content/learn-more; Consultado el 17/03/2014

2: Rupp, Vaughn y Woodworth, Brian; Proyecto opensourceautomation; OSA; [2013?]; http://www.opensourceautomation.com/wiki/index.php?title=Features; Consultado el 18/03/2014

3: Kreuzer, Kai y Eichstädt-Engelen, Thomas; Características del proyecto openHAB; openHAB; [2012?]; http://www.openhab.org/features.html; Consultado el 18/03/2014

4: Varios; Proyecto Raspberry Pi; Fundación Raspberry Pi; [2012?]; http://www.raspberrypi.org/quick-start-guide; Consultado el 21/03/2014

5: Varios; Página de producto; Elro; [2010?]; http://www.elro.eu/es/productos/cat/home-automation/home-control/conjuntos1/3-interruptores-con-mando-a-distancia; Consultado el 22/03/2014

6: Varios; Especificaciones técnicas "AM SuperHeterodyne Receiver"; Quasar UK; [2011]; http://docs-europe.electrocomponents.com/webdocs/0fe7/0900766b80fe7d94.pdf; Consultado el 27/03/2014

7: Varios; Especificaciones Técnicas "AM TRANSMITTER MODULE QAM-TX1"; Quasar UK; [2005]; http://docs-europe.electrocomponents.com/webdocs/087d/0900766b8087d2df.pdf; Consultado el 27/03/2014

8: Varios; Babeitech; Babeitech; [2010?]; http://en.babeitech.com/Index/PostDetail?OID=19&parentID=5; Consultado el 28/03/2014

9: Artés Rodriguez, Antonio; Pérez González, Fernando y otros; Comunicaciones digitales; Universidad Carlos III de Madrid; [2012]; Consultado el 12/02/2014

10: Varios; Audacity; Audacity; [2009?]; http://audacity.sourceforge.net/?lang=es; Consultado el 29/03/2014

11: Heiner, Wolf; Proyecto rc-switch; ; [2013]; http://code.google.com/p/rc-switch/wiki/HowTo_OperateLowCostOutlets; Consultado el 29/03/2014

12: Jenster, Ruben; Proyecto rcswitch-pi; ; [2012?]; https://github.com/r10r/rcswitch-pi; Consultado el 29/03/2014

13: Varios; Proyecto MQTT; MQTT; [1999?]; http://mqtt.org/; Consultado el 6/04/2014

14: Varios; Proyecto Owntracks; OwnTracks; [2013?]; http://owntracks.org/; Consultado el 06/04/2014

15: Varios; Embedded Linux Wiki; Embedded Linux; [13/10/2006]; http://elinux.org/ArchLinux_Install_Guide; Consultado el 15/04/2014

16: Varios; Embedded Linux Wiki; Embedded Linux; [13/10/2006]; http://elinux.org/RPi_Resize_Flash_Partitions#Manually_resizing_the_SD_card_on_Raspberry_Pi; Consultado el 15/04/2014

17: Ricardo Arturo Cabral; Blog de Ricardo Arturo Cabral; Ricardo Arturo Cabral; [01/2013]; http://blog.ricardoarturocabral.com/2013/01/auto-reboot-hung-raspberry-pi-using-on.html; Consultado el 16/04/2014

18: Varios; Proyecto ArchWiki; Arch Linux; [2005?]; https://wiki.archlinux.org/index.php/Noip; Consultado el 17/04/2014

19: Varios; Embedded Linux Wiki; Embedded Linux Wiki; [13/10/2006]; http://elinux.org/RPi_Low-level_peripherals; Consultado el 18/04/2014

20: Gordon Henderson; Página personal de Gordon Henderson; Gordon Henderson; [2012?]; http://wiringpi.com/; Consultado el 18/04/2014

21: Oracle; Java SE Embedded HardFP ABI Support for ARM Processors in 7U40; Oracle; [10/09/2013]; https://www.youtube.com/watch?feature=player_embedded&v=adrZDsRXmq4; Consultado el 20/04/2014

22: Doukas, Charalampos; Building Internet of Things with the Arduino; ; [02/04/2012];

22: José María Díaz; Blog personal de José María Díaz; José María Díaz; [17/04/2014]; http://trasteandoarduino.com/2014/04/17/intregrando-owntracks-en-openhab/; Consultado el 21/04/2014

23: Alexander Rudde; Blog personal de Alexander Rudde; Alexander Rudde; [02/2014]; http://alexander-rudde.com/2014/02/install-mosquitto-mqtt-broker-on-raspberry-pi-running-arch-linux/; Consultado el 21/04/2014