Grado Universitario en
Ingeniería en Tecnologías de la Telecomunicación

Escuela Técnica Superior de Ingeniería de las Telecomunicaciones

Trabajo Fin de Grado:

DESARROLLO DE UN SISTEMA DE SEGURIDAD DOMÓTICO CON JDEROBOT

Autor: Javier Fernández Hernando
Tutor: José María Cañas Plaza

Curso académico 2014-2015, Madrid

Una copia de este proyecto, las fuentes del programa y vídeos de los experimentos están disponibles en la siguiente dirección:

http://jderobot.org/Hernando-tfg
https://github.com/Javii91/Domotic

Agradecimientos

Me gustaría agradecer haber podido llegar hasta aquí...

A mis padres, que sin ellos no hubiera podido siquiera intentar conseguir un logro de este tamaño y que han depositado toda su confianza y apoyo en mi desde el principio.

A mi novia Laura, que me ha acompañado desde el primer día de universidad y hemos llegado hasta el final juntos superando todas las dificultades del camino.

A mi familia, por haber estado ahí apoyándome y ayudándome en los peores momentos.

A José María, por haber tenido que aguantar mis largos periodos de desaparición y haberme ayudado y aconsejado cuando llegaban las prisas del final.

Y, en general, a todo el mundo que se ha preocupado por mi en este periodo y ha tratado de ayudar en lo que ha podido.

Gracias.

Resumen

La domótica es un sector de gran atractivo y que está teniendo un gran crecimiento que se ha visto incrementado en los últimos años gracias al interés de grandes empresas tecnológicas y que gracias a esto ha obtenido un gran desarrollo tecnológico.

En este documento se explica cómo se ha desarrollado una aplicación domótica de seguridad que es capaz de conectarse a un sensor de temperatura, un sistema *X10*, cámaras y a sensores RGBD (Kinect, Asus Xtion..), creando un sistema domótico extremadamente configurable, sencillo y libre para el usuario en el que priorizan la obtención de ahorro energético, seguridad y confort totalmente adaptable a cualquier hogar independientemente de su tamaño y localización. Este sistema permite controlar el encendido y apagado de cualquier aparato electrónico conectado a un actuador X10, apagar o encender luces, radiadores o aparatos de aire acondicionado en función de la temperatura, detección de intrusiones y envío de alarmas entre otras opciones. Para obtener estas funcionalidades se ha tenido que analizar los datos crudos que provienen de estos sensores, en muchos casos, elaborarla para obtener información mas útil como por ejemplo detectar movimiento a partir de las imágenes de una cámara, y por último crear reglas de interoperación entre sensores y actuadores de los distintos módulos que conforman la aplicación. Estas reglas definen órdenes totalmente configurables y que automatizan acciones para ciertos estados en los sensores.

Para la creación de esta aplicación domótica también se ha tenido que desarrollar un componente que funcione como servidor y controlador de un sistema *X10*. Este servidor se conecta a una base central CM11A capaz de monitorizar un sistema *X10* y recibir órdenes de un cliente. Además, el control de temperatura también ha requerido el desarrollo de un driver para un termómetro USB que funciona como servidor y que es capaz de devolver la información bajo petición de un cliente.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se realizará una introducción sobre los conceptos específicos del área que se desarrolla. En especial se hará hincapié en la descripción y usos presentes de la domótica tanto en la sociedad como en la Universidad Rey Juan Carlos que realiza el grupo de robótica y que conciernen a la realización de este Trabajo de Fin de Grado.

1.1. Domótica

Un sistema domótico [15] es un sistema inteligente que es capaz de recoger información a través de sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas. Por lo tanto, hablamos de un sistema que puede ser automático y autosuficiente y que podemos encontrar generalmente en un ambiente doméstico y últimamente también en fábricas y edificios de oficina. Además el sistema puede acceder a redes exteriores de comunicación o información que le permite incrementar su capacidad.

El sector de la domótica ha evolucionado considerablemente en los últimos años, y en la actualidad ofrece una oferta más consolidada. En definitiva, la domótica de hoy contribuye a aumentar la calidad de vida, hace más versátil la distribución de la casa, cambia las condiciones ambientales creando diferentes escenas predefinidas, y consigue que la vivienda sea más funcional al permitir desarrollar facetas domésticas, profesionales, y de ocio bajo un mismo techo.

La domótica contribuye a mejorar la calidad de vida del usuario:

- Facilitando el *ahorro energético*: gestiona inteligentemente la iluminación, climatización, agua caliente sanitaria, el riego, los electrodomésticos, etc.,

aprovechando mejor los recursos naturales, utilizando las tarifas horarias de menor coste, y reduciendo así, la factura energética. Además, mediante la monitorización de consumos que nos ofrecen ciertos aparatos, se obtiene la información necesaria para modificar los hábitos y aumentar el ahorro y la eficiencia. Un ejemplo de este caso sería la luz de un rellano conectado a un sensor de movimiento y luminosidad, el cual solo mantendrá una luz encendida si por condiciones de visibilidad es necesario y únicamente si detecta movimiento en la sala.

- Aportando *seguridad* mediante la vigilancia automática de personas, animales y bienes, así como de incidencias y averías. Mediante controles de intrusión, cierre automático de todas las aberturas, simulación dinámica de presencia, fachadas dinámicas, cámaras de vigilancia, alarmas personales, y a través de alarmas técnicas que permiten detectar incendios, fugas de gas, inundaciones de agua, fallos del suministro eléctrico, etc. Un buen ejemplo práctico para este punto será el desarrollo de este proyecto, gracias al uso de cámaras, detectores de movimiento y distintas configuraciones que podremos realizar.

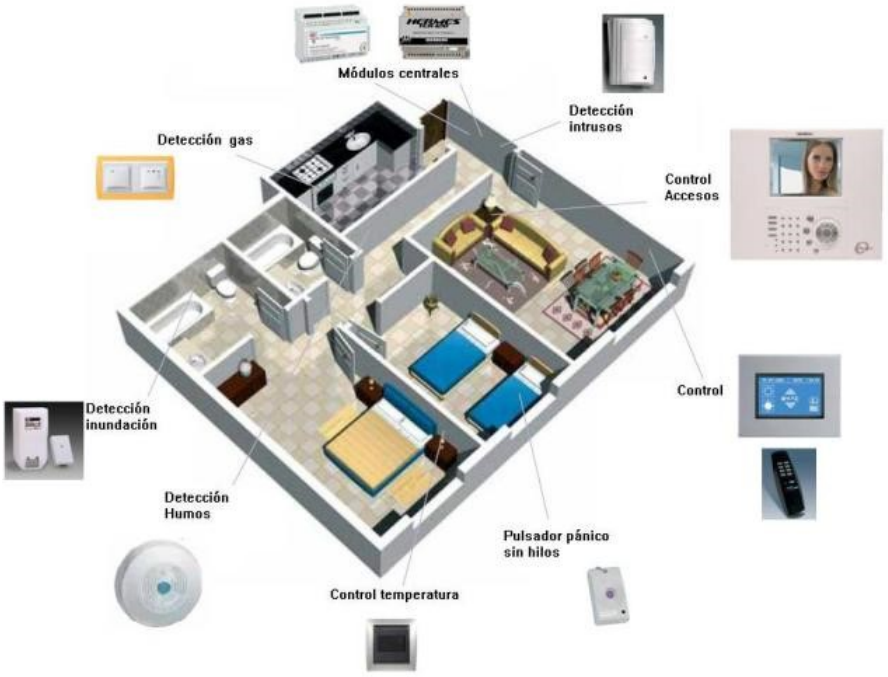

Figura 1.1: Ejemplo de sistema domótico de seguridad

- Convirtiendo la vivienda en un hogar más confortable a través de la gestión de dispositivos y actividades domésticas. La domótica permite abrir, cerrar, apagar, encender, regular... los electrodomésticos, la climatización, ventilación, iluminación

natural y artificial, persianas, toldos, puertas, cortinas, riego, suministro de agua, gas, electricidad... Todo ello pudiendo ser manejado desde una consola central que nos dará control total de todos los dispositivos conectados con los que podrá comunicarse y dar órdenes.

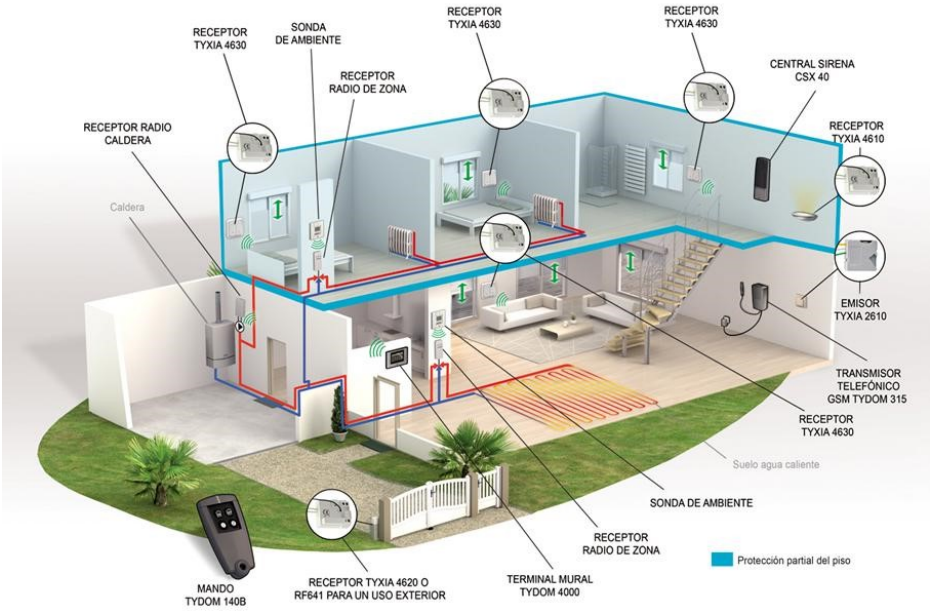

Figura 1.2: Ejemplo de sistema domótico de hogar confortable

1.2. Estándares domóticos

Los primeros sistemas comerciales fueron instalados, sobre todo, en Estados Unidos. Estos se limitaban a la regulación de la temperatura ambiente de los edificios de oficinas.

Debido a la necesidad de interoperar distintos sistemas domóticos, por ejemplo ampliar una red ya existente con nuevos componentes, se comenzaron a definir estándares.

Si bien el estándar inicial *X10* fue suficiente para soportar estas pequeñas aplicaciones, posteriormente los automatismos destinados a edificios de oficinas (junto con otros específicos) se han ido aplicando también a las viviendas de particulares u otro tipo de edificios, donde el número de necesidades a cubrir es mucho más amplio, dando lugar a nuevos estándares de comunicación [13].

- **X10**

  Protocolo estándar de comunicaciones para el control remoto de dispositivos eléctricos [38], desarrollado en 1975. Utiliza la red eléctrica para comunicar los distintos dispositivos del sistema a través de comandos predefinidos en el estándar.

  Fue la primera tecnología domótica en aparecer, dando solución específica a las comunicaciones domóticas, diferentes de las comunicaciones arbitrarias entre dispositivos. Este estándar sigue en uso hoy en día, aunque ha sido redefinido para otorgarle más funcionalidades y existen protocolos alternativos más recientes, pero con menor número de dispositivos comerciales (debido principalmente al extenso tiempo que *X10* lleva en el mercado). Se hablará de este protocolo más detenidamente en la sección 3.3 ya que concierne en un punto muy importante a este TFG.

- **LonWorks**

  Es un protocolo para sistemas inmóticos o de automatización de edificios, también conocido como *BMS* [26]. Basado en soluciones centralizadas de control punto-a-punto. Esto significa que existe un "maestro.º controlador principal similar a un ordenador, físicamente cableado a cada punto de control particular, como actuadores o sensores, denominados .ªesclavos". Su principal cualidad es su capacidad de crecimiento casi ilimitado sin necesidad de sustituir elementos en la red existente, solo se precisa añadir nuevos equipos (nodos) y su gran parecido con el protocolo TCP/IP.

- **CEBus**

  Consumer Electronics Bus [10] es un estandar vigente en los Estados Unidos que ha sido desarrollado por EIA-Electronic Industries Association. El estándar surgió en 1984 cuando la EIA se propuso unificar los protocolos de señalización infrarroja para el control de remoto de electrodomésticos. En 1992 el estándar se había extendido a todo el ámbito de control domótico.

  Los objetivos principales del estándar son: Facilitar el desarrollo de módulos integrados fácilmente en electrodomésticos, soportar la distribución de servicios de audio y vídeo, evitar la necesidad de un controlador central, distribuyendo la inteligencia de la red entre todos los dispositivos y permitir añadir y quitar componentes de la red sin que afecte al rendimiento del sistema y sin un gran esfuerzo la configuración por parte del usuario.

  Medios físicos permitidos: Red eléctrica, cable trenzado y coaxial, infrarrojos, radiofrecuencia, fibra óptica y bus de audio-video.

- **KNX**

  Estándar europeo para la automatización de las viviendas y oficinas [24]. Para la transmisión de datos puede utilizar cableado de baja tensión (el más utilizado actualmente), de alta tensión (red eléctrica), transmisión inalámbrica o por Internet.

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

- **SCP**

  Simple Control Protocol [29]. Estándar de *Microsoft y General Electric* y actualmente libre de royalties. Necesita conexiones punto a punto y cableado de baja tensión, por lo que es prácticamente inviable para sistemas con un gran número de sensores. En general, este estándar no está teniendo gran acogida, además, es incompatible con la legislación Europea (en concreto, con la norma EN-50065 [31]).

1.3. Mercado domótico

Recientemente ha crecido el número de empresas que ofrecen sistemas domóticos interoperables entre sí, permitiendo crear sistemas de forma flexible a un bajo precio. Normalmente, se necesitan conocimientos técnicos para poner a punto este tipo de redes. Con estos sistemas se pueden abarcar muchas soluciones: seguridad ante robos y averías domésticas, comodidad en el hogar y ahorro de energía. Generalmente, en estas soluciones comerciales, se pueden adquirir el número de nodos que se requieran.

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

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

Figura 1.3: Sensores domóticos disponibles en el mercado

A continuación, se ofrecen algunos ejemplos representativos del marco comercial de la domótica en España:

- **Z-Wave**

  Se autodefinen como estándar internacional [40]. Dispone de varios módulos de distintos fabricantes interoperables, con una gran variedad de sensores y actuadores. Permiten crear sistemas inalámbricos y gestión a través de teléfonos inteligentes. Al ser sistemas de distintos fabricantes, son sistemas propietarios, además aunque el sistema está en venta, algunos módulos siguen en desarrollo.

- **Verisure**

  Sistema de alarmas de la empresa *Securitas Direct* [37]. No se trata de un sistema domótico completo, pues únicamente dispone de sensores de alarma. La empresa dispone de una consola central y varios tipos de sensores con los que según un presupuesto, y dependiendo del hogar a cubrir, se realiza una combinación efectiva. La principal ventaja de este sistema consiste en el uso de sensores volumétricos adaptado a mascotas. Esto permite el uso de dichos sensores ante la presencia de animales sin provocar falsas alarmas.

- **Prosegur**

  Empresa similar a Securitas Direct y que ofrece servicios de vigilancia y domótica [33]. Disponen de un sistema domótico completo de seguridad, confort, ahorro energético

y telecomunicaciones, aunque su principal preocupación es la comercialización de sistemas domóticos de seguridad módulos de vídeo-vigilancia (cámaras IP), detectores de movimiento, contactos magnéticos y centralita con etiqueta NCF.

- **Delta Dore**

  Ejemplo de sistema domótico actualmente comercializado, genérico, multipropósito e inalámbrico [14]. Se compone de distintos módulos según la necesidad del cliente. El sistema puede tener capacidad para gestionar la climatización y la luz de la casa, alarmas y apertura de puertas automáticas. El sistema se controla a través de un mando a distancia, desde un cuadro de mando o desde una aplicación para el móvil.

- **Libelium**

  Red de sensores de fabricación española que implementa router multiprotocolo [25], el cual puede interactuar con las tecnologías Wi-Fi, ZigBee, GPRS, Bluetooth y GPS en una misma máquina. Es un sistema principalmente pensado para desarrollo (utiliza Arduino y XBee como base), aunque la propia empresa Libelium puede crear sistemas a medida a partir de sus placas y sensores.

- **Nest**

  Empresa comprada por *Google* [28] que actualmente comercializa dos tipos de productos, un termostato centrado en obtener la mejor eficiencia energética que es capaz de aprender nuestro comportamiento y preferencias mediante una combinación de sensores, algoritmos de aprendizaje y la computación en la nube. El segundo producto es un detector de humo y monóxido de carbono con alarma y envió de información a través de internet. En este momento *Nest* no comercializa sus productos en España pero su sencillez de uso e innovación están haciendo avanzar el mercado de la domótica.

1.4. Domótica en el grupo de robótica URJC

En este apartado se repasa varios aportes de la URJC [9] al mundo de la domótica y que encabezan el contexto inmediato de este TFG:

- **Proyecto Surveillance 1.0**

  La versión *Surveillance 1.0* es el producto del Trabajo Fin de Máster de *Roberto Calvo Palomino* [2]. Consiste en un sistema de video vigilancia distribuido integrado en

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

- **Proyecto Surveillance 2.0**

  Para la versión de *Surveillance 2.0* se constituye como una exploración aparte, realizada por *Roberto Calvo*, para visualizar el valor de diversos sensores a través de un teléfono inteligente con Android. Para ello, se implementó una aplicación compuesta por diversos sensores conectados a una placa Arduino y cuyos datos finales eran visualizados en un dispositivo móvil. Para lograr las comunicaciones, tanto de los sensores con la placa, como de la placa con el dispositivo móvil, se usaron cables. Se integraron sensores de infrarrojos (movimiento), temperatura y puerta abierta (sensor magnético). Para ello, los sensores se unieron a una placa Arduino mediante un cable USB, la placa procesa los datos y los envía a través de su interfaz Ethernet para poder visualizarlo en un tercer dispositivo. La comunicación entre el Arduino y el dispositivo final (la muestra se realizó con un dispositivo Android) se basa en los protocolos HTTP y JSON.

- **Proyecto Surveillance 4.0**

  Posteriormente, en la versión *Surveillance 4.0* [5] que fue desarrollado por *Daniel Castellanos* [4] como su Proyecto de Final de Carrera, la aplicación contaba con varios sensores de distinto tipo (humedad, temperatura, gas, etc.) que se conectaban inalámbricamente con un nodo central situado en una Raspberry Pi. La conexión inalámbrica se hacía mediante transmisores Zigbee con un protocolo propio llamado WHAP. El nodo central recibía los datos de los sensores y los mostraba mediante un servidor web que corría en la misma máquina. La aplicación web se desarrolló en Python usando el entorno de desarrollo Django. En *Surveillance 4.0*, los valores de los sensores se guardaban en una base de datos que la aplicación web consultaba cuando era necesario. Además, esta versión incluía un streaming de vídeo pero utilizando el software de código abierto M-JPEG Streamer.

- **Proyecto Surveillance 5.1**

  Una actualización de este proyecto fue llevada a cabo por *Edgar Barrero* [27] que realizo una aplicación web en *Surveillance 5.1* [7] basado en Ruby on Rails y que muestra una interfaz con HTML5 y WebGL que nos permite controlar los distintos

sensores, cámaras y Kinects.

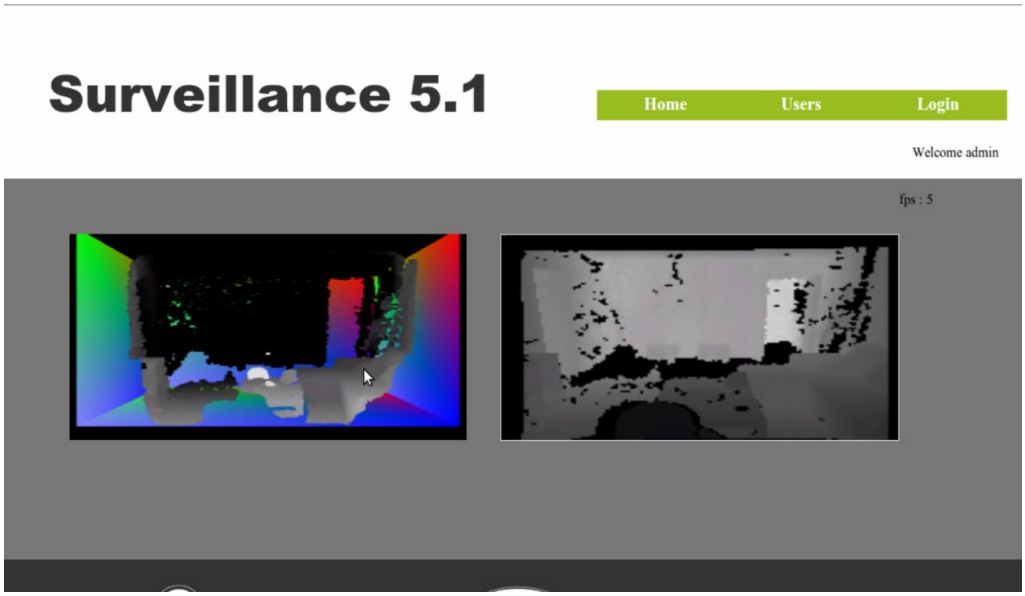

Figura 1.4: Funcionamiento del sensor Kinect en Surveillance 5.1

- **Proyecto ElderCare**

  ElderCare [3] es un sistema autónomo y no intrusivo, diseñado para detectar caídas mediante el uso de cámaras y Kinects (o Xtions). Su principal mercado es el cuidado de personas mayores, por lo que su uso principal se encuentra en residencias de ancianos o domicilios particulares donde vivan éstos. El sistema permite reducir drásticamente el tiempo de respuesta a la asistencia en caso de caída, estimando la posición del usuario caído y avisando a sus cuidadores o a la asistencia médica.

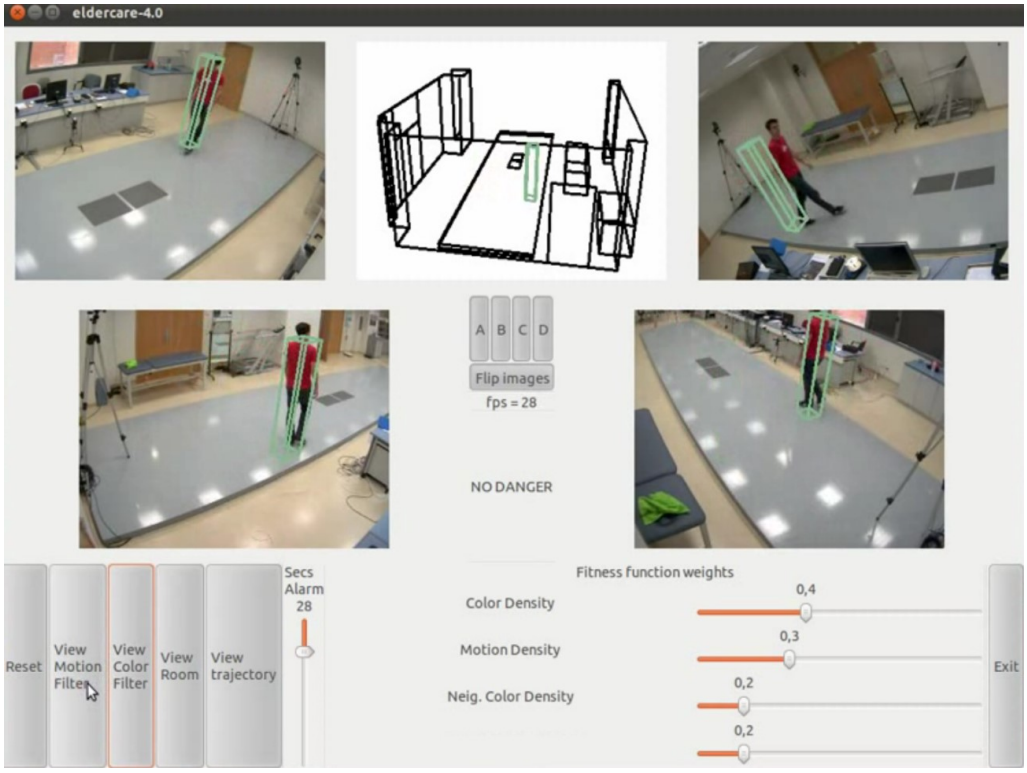

Figura 1.5: Funcionamiento aplicación ElderCare

El Trabajo de Fin de Grado presentado en esta memoria se puede dividir en varias partes que se descubrirán en los próximos capítulos, pero entre otras cosas hará uso de la plataforma *JdeRobot* [21] de la universidad en donde se actualizará un driver para el protocolo *X10* que sustituirá al existente que realizó *Sara Marugán Alonso* [1], se añadirá un driver de control de temperatura a través de un puerto USB y se diseñará combinando estos últimos proyectos mencionados una aplicación domótica de seguridad que tiene que ver en parte con los proyectos que ya se han mencionado en este punto.

En el apartado 2. Objetivos veremos como se ha dividido el proyecto en subobjetivos y la forma de trabajo que se ha realizado hasta el final. En el apartado 3. Infraestructura se explica el software y hardware básico en el que se ha apoyado el proyecto para obtener sus objetivos. Posteriormente se explica de forma detallada y con imágenes en el apartado 4. Sistema domótico todo el desarrollo del proyecto y funcionalidades, dividiéndose en función de los subobjetivos y características. Por ultimo, en la sección 5. Conclusiones analizaremos *a posteriori* que objetivos hemos cumplido y las posibles continuaciones que este proyecto podría tener.

Capítulo 2

Objetivos

Una vez presentado el contexto general sobre el que se asienta este proyecto, vamos a pasar a explicar una descripción detallada de los objetivos que pretendemos resolver con la realización de este trabajo así como la metodología utilizada.

2.1. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es crear una aplicación domótica de seguridad basándose en el control de los distintos sensores y actuadores que se conectan al sistema. Para ello, antes de comenzar con la aplicación, se necesita adaptar distintos tipos de sensores y actuadores con la creación de sus drivers específicos. Por lo que nuestro sistema domótico se debe dividir en los siguientes subobjetivos:

El primer subobjetivo englobaría los componentes que son necesarios crear o actualizar para la plataforma JdeRobot y que nos servirán para ampliar y mejorar las capacidades que tendrá la aplicación de seguridad. En primer lugar tendríamos que realizar la actualización del soporte de *X10* para la versión 5.3 de JdeRobot por uno nuevo que nos permita la comunicación desde la base central CM11A con los distintos sensores y actuadores a través de este protocolo. En segundo lugar vendría la creación de un componente encargado de obtener información de un sensor de temperatura conectado a un puerto USB.

El segundo subobjetivo sería el desarrollo de la aplicación de seguridad cuyo objetivo principal será la de ofrecer un sistema domótico extremadamente configurable y abierto para que cada usuario pueda adaptarlo fácilmente a sus necesidades. Estas posibilidades de adaptación deben ir encaminadas a la seguridad del hogar a través de la monitorización por cámaras, Kinects y sensores de movimiento, el ahorro energético fusionando sensores

de movimiento, sensores de temperatura y actuadores *X10* con el cual podremos controlar el encendido y apagado de luces, calefacción, caldera... y la comodidad de tener una casa totalmente configurable.

Como requisito para el sistema es necesario que funcione con la versión 5.3 de JdeRobot y que tenga un uso eficiente de recursos en su funcionamiento.

2.2. Metodología

Para este TFG se ha realizado utilizando una metodología Incremental. Con esta metodología se provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto software y reservando el resto de aspectos para el futuro.

Una serie de metodologías de mini-Cascadas se llevan a cabo, donde todas las fases del modelo de cascada se han completado para una pequeña parte de los sistemas antes de proceder a la próxima incremental.

Al igual que los otros métodos de modelado, el Modelo Incremental es de naturaleza interactiva pero se diferencia de aquellos en que al final de cada incremento se entrega un producto completamente operacional. De esta forma los riesgos se toman con cada incremento teniendo asegurados los riesgos pasados.

Con esto se mantiene al cliente en constante contacto con los resultados obtenidos en cada incremento. Es el mismo cliente el que incluye o desecha elementos al final de cada incremento a fin de que el software se adapte mejor a sus necesidades reales. El proceso se repite hasta que se elabore el producto completo. Estas reuniones con el cliente e incrementos se han realizado con las reuniones semanales que he tenido con el tutor de este TFG y han quedado grabadas con los respectivos incrementos y desarrollos en mi MediaWiki [12] que ha servido como cuaderno de bitácora. Además el proyecto se ha ido utilizando en GitHub [11] con sus respectivos comentarios de mejoras por lo que se puede llevar un buen seguimiento del trabajo realizado.

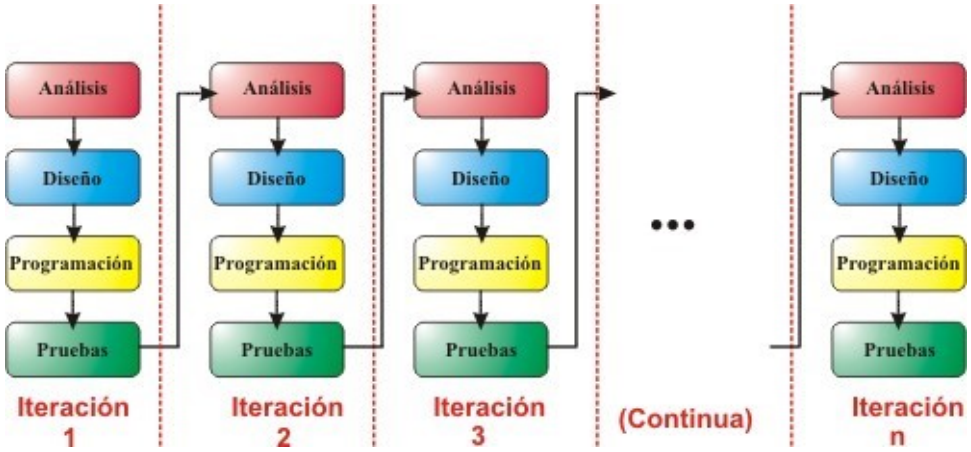

Figura 2.1: Esquema metodología incremental iteraciones

Esta metodología supone entre otras, las siguientes ventajas:

- Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial.
- El modelo proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento.
- Permite entregar al cliente un producto más rápido en comparación del modelo de cascada y resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos.

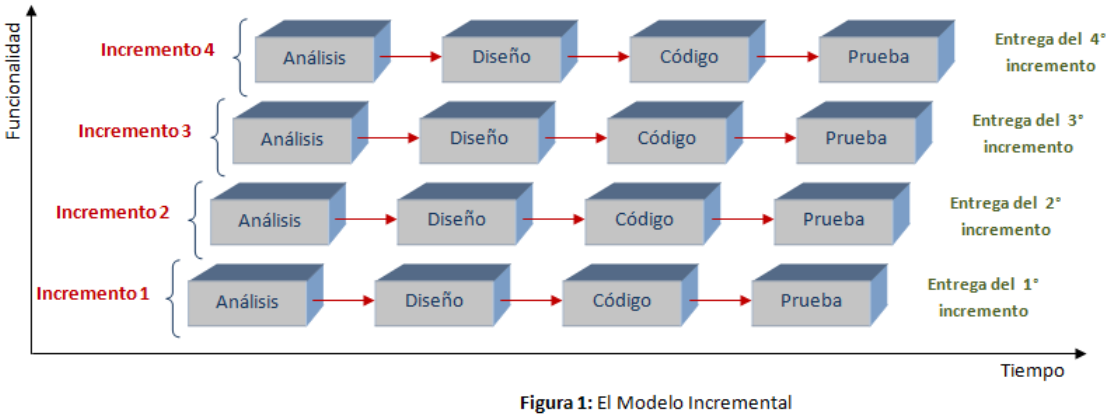

Figura 2.2: Esquema metodología incremental tiempo/funcionalidad

Como desventaja principal:

- Requiere de metas claras para conocer el estado del proyecto.

En resumen, el plan de trabajo ha sido:

- Familiarización con las herramientas de JdeRobot
- Estudio de la demótica en la URJC
- Diseño del driver para X10
- Diseño del driver para la temperatura
- Aplicación demótica donde poco a poco se añadían funcionalidades.
- Testing de la aplicación

Capítulo 3

Infraestructura

En este capítulo se describirá algunos de los componentes, plataformas o protocolos que ya existían con anterioridad al desarrollo de este TFG y que han servido como base y ayuda para un mejor desarrollo.

3.1. ICE

Internet Communications Engine (ICE) [20] es un conjunto de herramientas orientadas a objetos que permite construir aplicaciones distribuidas. ICE permite enfocar sus esfuerzos en su lógica de la aplicación, y se encarga de todas las interacciones con las interfaces de programación de red de bajo nivel. He utilizado la versión 3.5 para poder crear las comunicaciones cliente-servidor de los drivers creados y poder obtener datos de otros componentes de la plataforma JdeRobot.

3.2. JdeRobot

JdeRobot [21] es un entorno de desarrollo de software que ha sido creado por el departamento *GSYC de la Universidad Rey Juan Carlos*. El entorno de desarrollo se centra en aplicaciones de robótica, domótica y visión artificial. Estos dominios incluyen sensores, actuadores y software inteligente entre estos. Ha sido diseñado para ofrecer ayuda en el desarrollo de dicho software inteligente. En su mayoría, está escrito en lenguaje C++ y proporciona un entorno con distintos componentes distribuidos donde las aplicaciones pueden ser desarrolladas basándose en distintos componentes asíncronos. Cada componente puede ejecutarse en diferentes ordenadores y podemos conectar con ellos utilizando la capa

intermedia de comunicación que nos proporciona ICE. Por su parte, ICE nos permite que los componentes puedan ser desarrollados en distintos lenguajes como C++, Python, Java... y poder interoperar sin problemas.

Esta plataforma nos hace más simple el acceso a los distintos tipos de software que tiene ya preparados en sus componentes. De esta manera podemos obtener datos y medidas procedentes de sensores con tan solo una llamada a una función, y de la misma manera mandar una orden a un actuador de esta manera conseguimos abstraernos de la capa de Hardware. Estos actuadores pueden ser tantos reales como simulados, y nuestra aplicación podrá tener acceso a todas estas medidas y ordenes fácilmente.

A continuación nombraremos algunos componentes que nos ha proporcionado la plataforma.

3.2.1. CameraServer

Este componente [8] escrito en C++ funciona como un servidor el cual proporciona video en streaming de una o varias cámaras. Usa OpenCV internamente para manejar y procesar las fuentes de vídeo. Para su conexión con otros componentes y aplicaciones ofrece una interfaz ICE con la que comunicarse.

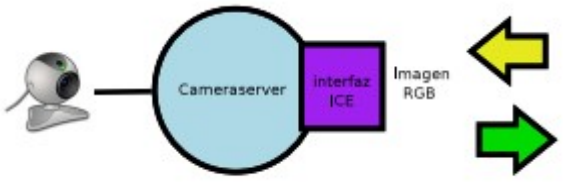

Figura 3.1: Esquema de funcionamiento de CameraServer

Cualquier programa que quiera conectarse con el componente CameraServer debe importar la interfaz camera.ice. Esta interfaz a su vez importa otra llamada image.ice. Esta última contiene un método llamado getImage() donde deberemos pasarle como parámetro el formato que queremos de la imagen. Al invocar este método se devuelve una estructura formada por una imagen en el formato deseado y las características de la misma. Para la ejecución de este componente, es necesario proporcionar la ruta de su archivo de configuración. En este fichero de configuración se define la IP y puerto donde escucha el servidor, el número y nombre de las cámaras que tiene y la configuración de las mismas

3.2.2. KinectServer

KinectServer [23] es un componente escrito en C++ que nos permite obtener imágenes en tiempo real de un sensor Kinect a través de una interfaz ICE. Gracias a este componente podremos obtener imágenes reales de un sensor Kinect para poder analizarlas, podemos recibir tanto la imagen RGB como su imagen de profundidad. Su funcionamiento es parecido al de *CameraServer* pero con un sensor Kinect.

3.2.3. Gazebo y FlyingKinect Plugin

Gazebo 5 es un software de simulación en el que *Juan Navarro Bosgos* [6] ha desarrollado un *plugin* que nos permite simular un sensor Kinect y que hemos utilizado para poder realizar la compatibilidad necesaria con nuestra aplicación de seguridad y un sistema de vigilancia basado en Kinect.

3.3. X10

X10 [38] es un protocolo de comunicaciones para el control remoto de dispositivos eléctricos que utiliza la línea eléctrica (220V o 110V) preexistente para transmitir señales de control entre equipos de automatización del hogar (domótica) en formato digital y la Radiofrecuencia (433MHz) para enviar las órdenes de control. No hace falta un cableado específico para la domótica ya que los comandos de control viajan a través del aire o del cable de la instalación eléctrica. Se trata de un sistema modular y fácilmente ampliable.

Se pueden direccionar hasta 256 módulos. A los módulos se les tiene que asignar una dirección de las 256 posibles compuestas por una letra (de la A a la P) y un número (del 1 al 16). En una misma instalación puede haber varios módulos configurados con la misma dirección, todos realizarán la función asignada cuando un controlador envíe una trama con esa dirección. Evidentemente cualquier módulo puede recibir órdenes de diferentes controladores.

Los dispositivos bidireccionales, tienen la capacidad de responder y confirmar la correcta realización de una orden, lo cual puede ser muy útil cuando el sistema X-10 está conectado a un programa de ordenador que muestre los estados en que se encuentra la instalación domótica de la vivienda.

3.3.1. Historia y Protocolo

X10 fue desarrollada en 1978 por Pico Electronics of Glenrothes, Escocia, para permitir el control remoto de los dispositivos domésticos. Fue la primera tecnología domótica en aparecer y sigue siendo la más ampliamente disponible, principalmente por su característica de autoinstalable, sin necesidad de cableado adicional.

Las señales de control de *X10* se basan en la transmisión de ráfagas de pulsos de RF (120 kHz) que representan información digital. Estos pulsos se sincronizan en el cruce por cero de la señal de red (50 Hz ó 60 Hz). Con la presencia de un pulso en un semiciclo y la ausencia del mismo en el semiciclo siguiente se representa un '1' lógico y a la inversa se representa un '0'. A su vez, cada orden se transmite 2 veces, con lo cual toda la información transmitida tiene cuádruple redundancia. Cada orden involucra 11 ciclos de red (220 ms para 50 Hz y 183,33, para 60Hz).

Los paquetes que se envían son, primero se transmite una orden con el Código de Casa y el Número de Módulo (4bits) que direccionan el módulo en cuestión. Luego se transmite otro orden con el código de función a realizar (Function Code también de 4 bits), por ejemplo, primero envía un "select code A3" seguido de un "turn on" que hará que se encienda el módulo A3 aunque también podríamos agrupar varias direcciones antes de enviar la función a realizar para así enviarlo a los módulos seleccionados.

Código	**Función**	**Descripción**
0 0 0 0	All units off	Apaga todos los módulos de la casa seleccionada.
0 0 0 1	All lights on	Enciende todos los módulos lámpara con control de brillo.
0 1 1 0	All lights off	Apaga todos los módulos lámpara.
0 0 1 0	On	Enciende un módulo
0 0 1 1	Off	Apaga un módulo
0 1 0 0	Dim	Reduce la intensidad de la luz.
0 1 0 1	Bright	Incrementa la intensidad de la luz.
0 1 1 1	Extended code	Código de extensión.
1 0 0 0	Hail request	Pide una respuesta con el código de la casa al módulo.
1 0 0 1	Hail acknowledge	Respuesta al comando anterior.
1 0 1 0	Pre-set dim	Permite predefinir 2 estados de intensidad
1 1 0 1	Status is on	Respuesta de Status Request indica si el módulo está activo
1 1 1 0	Status is off	Respuesta indicando si el módulo está inactivo
1 1 1 1	Status request	Pide el estado del módulo.

Cuadro 3.1: Mensajes del protocolo *X10*

Solo los módulos que poseen una comunicación bidireccional pueden responder a las

peticiones, lo que proporciona un sistema mucho más robusto.

3.3.2. Sensores

Entre otros sensores, en este TFG se ha utilizado únicamente el sensor de movimiento TM13, el cual consta de dos partes, un transceptor donde podremos seleccionar la Casa en la que se encuentra (no permite seleccionar el Número de Módulo) con comunicación bidireccional que se conecta a la red eléctrica por donde puede comunicarse con la base central y además posee un sistema de comunicación inalámbrica con la que puede comunicarse con la segunda parte del sensor que se trata de un sensor de movimiento.

Figura 3.2: TM13 Motion Detector

3.3.3. Actuadores

Entre otros actuadores que posee el sistema *X10*, se han utilizado 3 módulos de lámpara LM12. El cual consta de un único aparato conectado directamente a la red eléctrica donde nos permite seleccionar el código de la Casa y el Numero de Módulo para el dispositivo y cuenta con una comunicación unidireccional por lo que la comunicación con estos será a ciegas ya que podemos transmitirles ordenes desde la base central pero no obtendremos respuesta de si hay algo al otro lado o si ha funcionado correctamente.

Figura 3.3: LM12 Lamp Module

3.3.4. Base Central

Para la base central o controlador se ha utilizado el dispositivo CM11A, el cual nos permitirá realizar un enlace de comunicación entre nuestra aplicación y el sistema *X10* conectado a la red eléctrica. Este dispositivo se conecta directamente a la red eléctrica y también posee un cable en serie RJ11 donde nosotros utilizamos un conversor a USB para conectarlo con el ordenador. Este dispositivo se encarga de transformar las órdenes digitales que envía la aplicación en mensajes descritos anteriormente que se pueden transmitir por la red eléctrica y que el resto de módulos pueden entender.

Figura 3.4: CM11A Controller y adaptador Serial-USB

3.3.5. HEYU

HEYU [18] es un software discontinuado que nos permite hablar con dispositivos *X10*, para ello he usado su versión *2.11-rc1*. Este software se ha utilizado facilitar la comunicación entre el *driver X10* desarrollado por mí y la consola CM11A que hemos descrito. Este software ha sido fundamental para poder desarrollar la comunicación con los dispositivos *X10*. El problema principal de HEYU es que solo nos permite controlar un código casa al mismo tiempo establecido en un fichero por lo que siempre se utilizara el código House A.

3.4. Kinect

Kinect [22] es un sensor creado para la *Xbox 360* y en su versión 2.0 para la *Xbox One* compuesto por una cámara RGB, un sensor infrarrojo y un micrófono, el sensor de infrarrojo se encarga de detectar movimientos corporales, el micrófono captará nuestra voz y la cámara ayudara al sensor a la hora de interpretar los movimientos.

Figura 3.5: Sensor Kinect para Xbox 360

3.5. Python

Python [34] es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. He utilizado la versión 2.7.

Figura 3.6: Logo de Python

3.6. GTK+ 3.0

Se ha utilizado la librería *GTK+* versión 3 [17] para la creación de la interfaz gráfica de la aplicación tanto en forma de código para crear una interfaz dinámica, como con el editor *Glade* [16] para crear los primeros pasos de ventanas y diálogos.

3.7. OpenCV

Se ha utilizado esta librería en su versión *OpenCV 2* [30] para la edición de vídeo para obtener información importante procedente de la cámara y Kinect. Con esta edición se puede conseguir por ejemplo contar el número de personas en una habitación.

3.8. Termómetro

TEMPer2 [35] es un sensor de temperatura digital que se conecta directamente al USB. Posee dos sensores, uno interior y otro exterior extensible por un cable. Ambos sensores pueden medir la temperatura al mismo tiempo y el sensor exterior está protegido frente al agua por lo que es sumergible.

Para el control de este dispositivo me he ayudado del software *PCSensor v1.0.1* [31] creado por *Juan Carlos Pérez*, este software hace de puente entre el Hardware y el componente que he realizado para servir las temperaturas a través de ICE.

Figura 3.7: Dispositivo *TEMPer2*

Capítulo 4

Sistema Domótico

En esta sección se describe el sistema domótico programado, tanto en la parte de sus drivers como en la parte de la aplicación de seguridad. Todo el código fuente e información está disponible en GitHub [11] y MediaWiki [12]

4.1. Diseño

El sistema domótico realizado sigue un diseño en el que se centraliza en una aplicación, que actúa como cliente, el control de diferentes sensores y actuadores a los que se conecta como servidores.

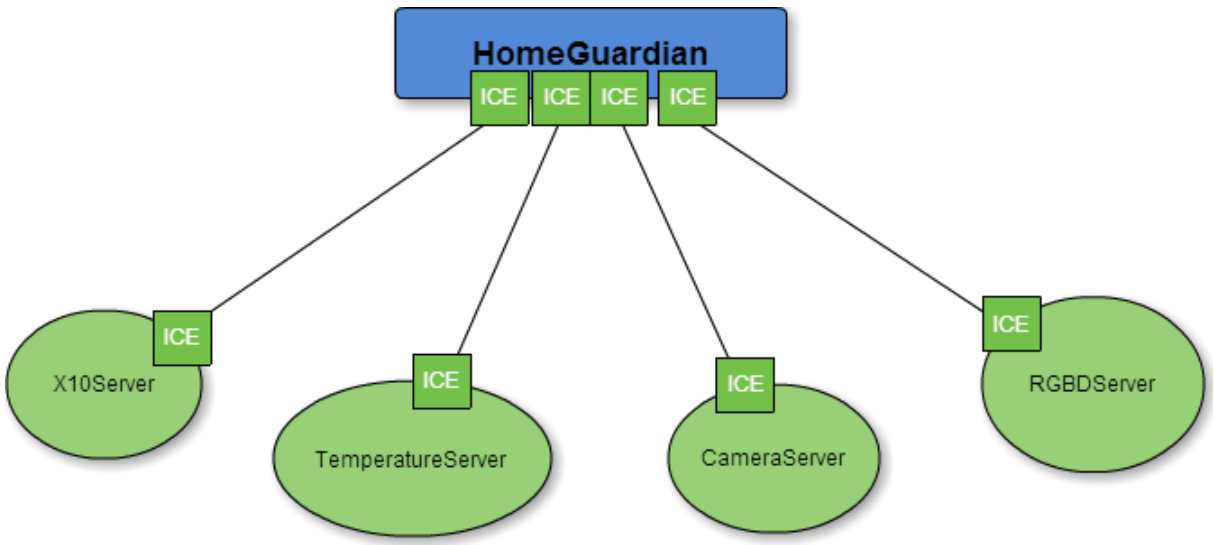

Figura 4.1: Diagrama de diseño del sistema domótico HomeGuardian

En la Figura 4.1 vemos el diseño del sistema domótico desarrollado, una aplicación llamada `HomeGuardian` que obtiene datos de sensores a través de sus interfaces ICE. Estos sensores son TemperatureServer que se encarga de la medición de datos de temperatura, CameraServer que se encarga de la obtención de imágenes de cámaras conectadas, RGBDServer que puede ser cualquier servidor que nos aporte imágenes RGBD, y por ultimo X10Server el cual nos añadirá tanto sensores de movimiento como actuadores y que conectará la aplicación con un sistema *X10*. `HomeGuardian` utiliza toda esta información recibida y permite al usuario crear rutinas y reglas configurables para crear su propio sistema domótico.

4.2. Drivers domóticos

Se han diseñado una serie de drivers necesarios para poder utilizar una mayor amplitud de sensores como el control de un sistema X10 o obtener información de un sensor de temperatura. Esto, nos permitirá ampliar las funcionalidades y características del sistema domótico.

4.2.1. X10Server y X10View

Este componente creado, escrito en Python, sustituye al que había previamente en la plataforma JdeRobot para la nueva versión 5.3. Este componente se encarga de hacer más sencilla la comunicación con los distintos dispositivos *X10* que tengamos conectados. Se ha dividido el componente en dos partes: un servidor (x10Server) que se encarga del control directo de los distintos dispositivos, y un cliente (x10View) que es una interfaz sencilla que nos permite diseñar y manejar nuestra red de *X10* fácilmente a través del servidor. Este diseño se muestra en el siguiente diagrama.

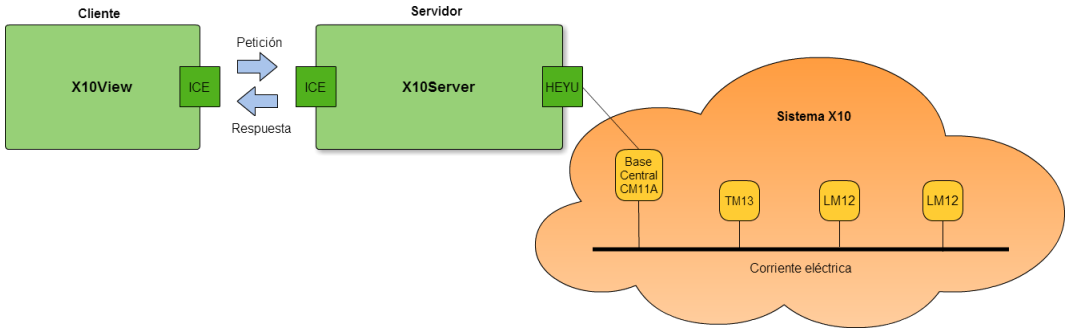

Figura 4.2: Diagrama de diseño driver control *X10*. X10View-X10Server

`X10Server` [39] es un componente que utiliza *HEYU* como base de comunicación con la base central CM11A. Se trata de un componente que corre en consola y que nos permite el control a través de una interfaz ICE de una red *X10* en la casa A. El funcionamiento de este componente se puede dividir en dos partes como se muestra en el siguiente esquema:

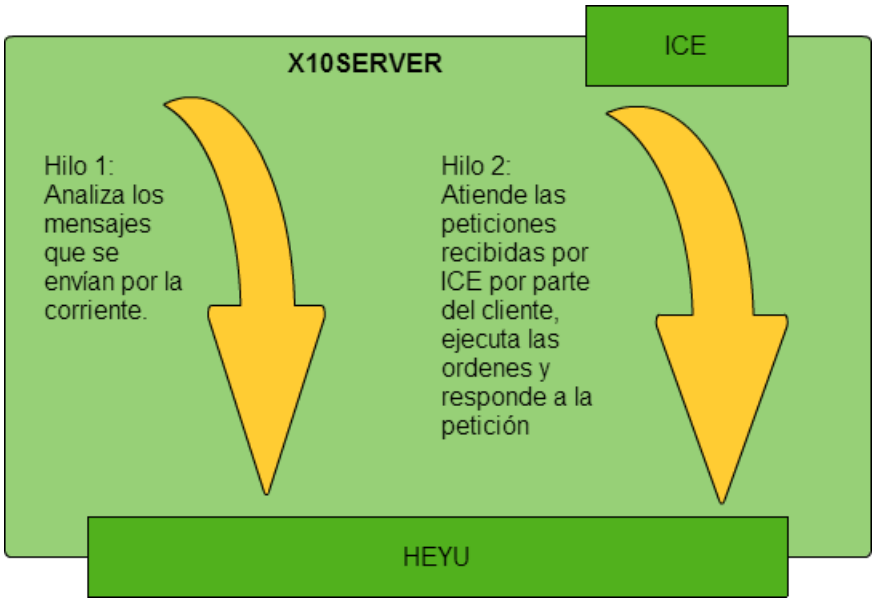

Figura 4.3: Diagrama de funcionamiento interno X10Server

- Un hilo (*Hilo 1* en Figura 4.3) de ejecución que se encarga de monitorizar los mensajes que se encuentran en la red eléctrica que los dispositivos transmisores puedan enviar como por ejemplo mensajes de sensores indicando su activación o desactivación y los muestra en la consola, en mi caso, al sólo disponer de un dispositivo que podía transmitir (sensor de movimiento TM13), este hilo sólo recoge mensajes si detecta movimiento o si deja de detectarlo. Además si descubre un mensaje de un sensor que

no había sido añadido a la red, este hilo lo añadirá junto al resto.

- Un hilo (*Hilo 2* en Figura 4.3) de ejecución que se encarga de leer las peticiones que el servidor puede recibir de los distintos clientes a través de ICE, mostrarlas en la consola y ejecutarlas. Entre estas peticiones podemos encontrar: petición estado de los dispositivos conectados, cambio de nombre a un dispositivo, añadir o eliminar un dispositivo de la red y activar o desactivar un dispositivo.

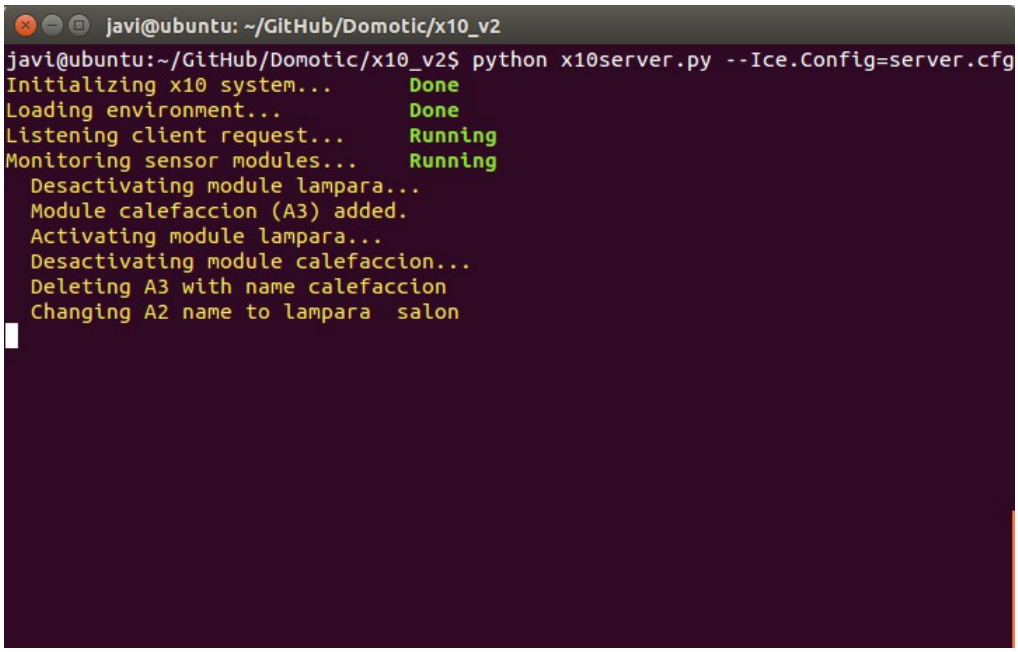

Figura 4.4: Ejemplo de ejecución x10Server

En la Figura 4.4 podemos ver un ejemplo de ejecución del servidor, en el que a través de un fichero de configuración le establezco que escuche en el puerto 9999 y también le digo que inicie con el dispositivo A1 como sensor de movimiento y con A2 como una lámpara. Además muestra las peticiones que reciben, entre otras las de activación y desactivación de dispositivos, cambio de nombre de un dispositivo y añadir y eliminar un dispositivo calefacción, por lo que tenemos un seguimiento de las acciones realizadas desde el servidor. X10View es el cliente que nos permite controlar la red fácilmente desde una interfaz gráfica. Este cliente nos da acceso a todas las funcionalidades que nos ofrece el servidor por lo que podremos controlar totalmente el sistema con este cliente.

En la Figura 4.5 vemos el componente `X10View` con la red que está definida en el servidor: contamos con un sensor de movimiento en A1 desactivado, si el sensor se activara el color bajo la columna 'Active' sería verde, además contamos con dos módulos 'Lamp'

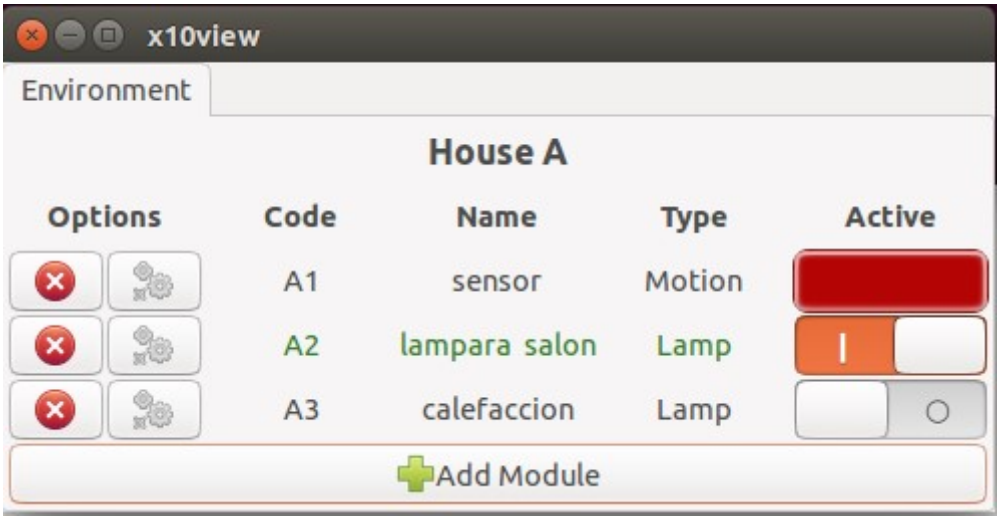

Figura 4.5: Ejemplo de ejecución x10View

que controlan una lámpara de salón en A2 y la calefacción en A3, estando este último desactivado y la lámpara de salón activada. Como podemos ver, los sensores sólo muestran la información de su estado mientras que los actuadores 'Lamp' nos permiten controlar su encendido y apagado a través del botón bajo la columna 'Active'. Además, todos los dispositivos bajo la columna 'Options' tienen dos botones, el primero que nos permite eliminar el dispositivo de la red, y el segundo que nos permite cambiarle el nombre a través de una ventana emergente. Es posible que la primera ejecución de un Activar/Desactivar se demore un par de segundos por temas ajenos al driver, pero después el tiempo de respuesta es muy pequeño.

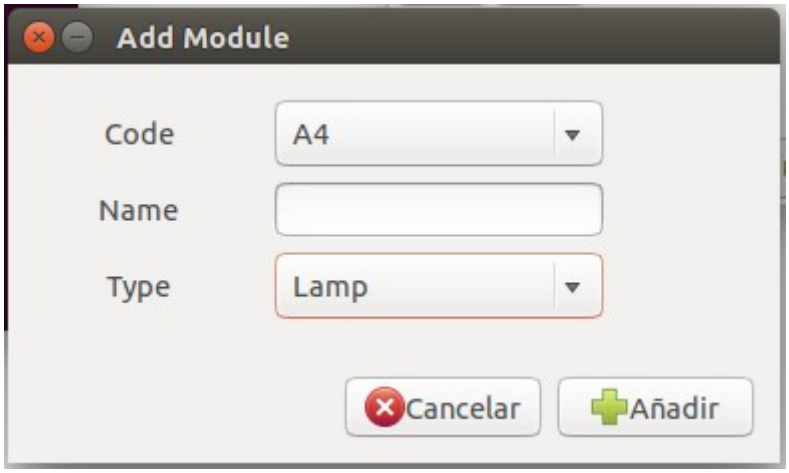

Figura 4.6: Ejemplo añadir dispositivo *X10*

En la Figura 4.6 vemos la ventana emergente que nos saldría al clicar en 'Add Module', en esta ventana podremos añadir un nuevo dispositivo a nuestra red otorgándole un nombre, el tipo de dispositivo 'Lamp' o 'Motion' dependiendo si es un dispositivo LM12 o TM13, y el código casa-módulo donde se encuentra conectado, esta lista sólo ofrecerá los códigos que aún no han sido cogidos desde A1 hasta A16.

Un pedazo de código de ejecución del servidor se muestra en la Figura 4.7.

```
## Initialize HEYU system
sys.stdout.write(txt.warning("Initializing x10 system...\t"))
sys.stdout.flush()
proc = subprocess.Popen("sudo heyu start", stdout=subprocess.PIPE, stderr=subprocess.PIPE, shell=True)
(out, err) = proc.communicate()

try:
    ic = Ice.initialize(sys.argv)
    adapter = ic.createObjectAdapterWithEndpoints("NetAdapter", ic.getProperties().getProperty("x10server.Endpoints"))
    if err != "":
      print txt.bold(txt.fail("Failed"))
      print txt.fail("Can't open tty line.  Check the permissions.")
      status = 1
      sys.exit(status)
    else:
      print txt.bold(txt.green("Done"))
    status = 0

    props = ic.getProperties().getPropertiesForPrefix("x10server")
    checkModules(props)

    object = NetI()
    adapter.add(object, ic.stringToIdentity("Net"))
    adapter.activate()

    sys.stdout.write(txt.warning("Listening client request...\t"))
    sys.stdout.flush()
    print txt.bold(txt.green("Running"))

    thread = Thread(target = object.checkSensor)
    thread.start()

    ic.waitForShutdown()
```

Figura 4.7: Pedazo de código de X10Server

En la Figura 4.7 se muestra el inicio del servidor, como inicializa el sistema HEYU, como inicia una interfaz ICE de comunicación, como lee el fichero de configuración para iniciar los módulos predefinidos, comienza a escuchar peticiones de clientes y por ultimo lanza un hilo para monitorizar mensajes en la red eléctrica.

4.2.2. TemperatureServer

Este componente escrito en Python se añade como nuevo a la plataforma *JdeRobot* y se encarga de obtener la información de temperatura que nos ofrece el sensor *TEMPer2* y ofrecerla a través de una interfaz ICE, para ello utiliza el software *PCSensor* como ayuda para la comunicación entre el componente y el dispositivo USB *TEMPer2*. Este diseño se muestra junto a su funcionamiento interno en el siguiente diagrama.

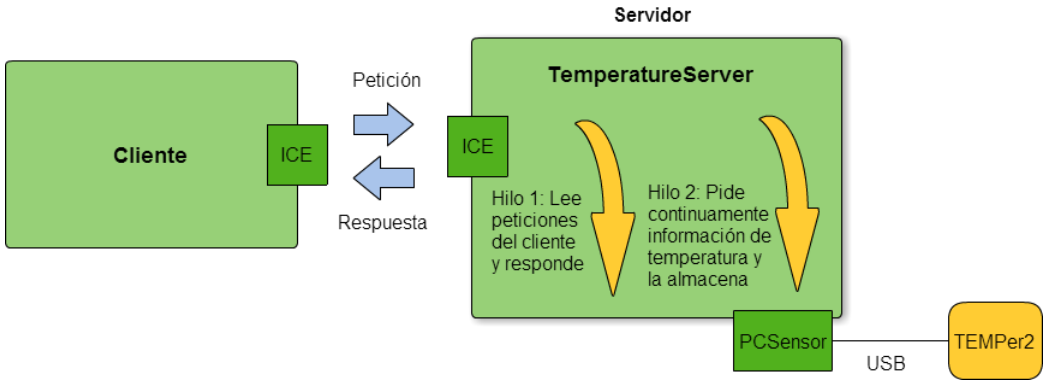

Figura 4.8: Diagrama de diseño y funcionamiento de TemperatureServer

El componente `TemperatureServer` [36] actualiza continuamente la información de temperatura que ofrece el termómetro y almacenando su valor a través del Hilo 2 de la Figura 4.8, por lo que el servidor está trabajando continuamente y listo para responder las peticiones que llegan desde el Hilo 1 de la Figura 4.8. En cuanto a la interfaz de este servidor, corre en la consola pero no muestra más información que su ejecución de inicio como se muestra en la siguiente figura.

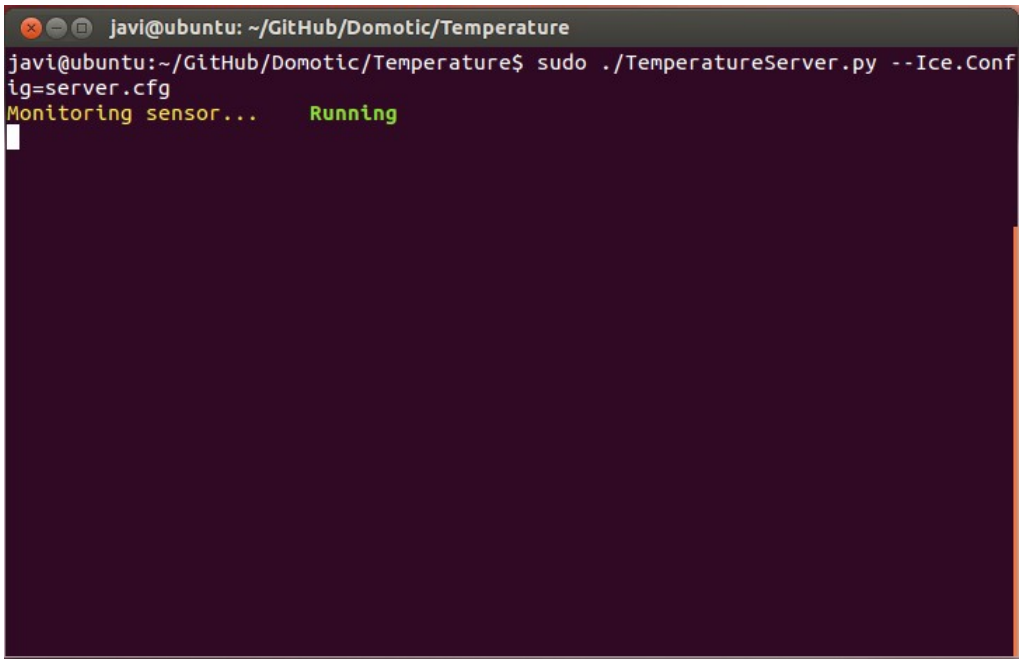

Figura 4.9: Ejemplo de ejecución de TemperatureServer

El componente permite pedir la temperatura de cualquiera de sus dos sensores de temperatura que posee y devuelve la temperatura como un 'Float' en grados Celsius.

Podemos configurarle a través de un fichero de configuración la dirección en la que el servidor se establecerá y leerá las peticiones de temperatura que vienen de ICE.

4.3. Aplicación domótica

`HomeGuardian` [19] es la aplicación domótica y de seguridad que se establece como principal y más importante desarrollo en este TFG. Se trata de un sistema sencillo y extremadamente configurable que lo hace diferente del resto de sistemas domóticos que se comercializan y que hemos nombrado con anterioridad. Estos sólo nos permiten unas configuraciones básicas previamente establecidas y que en su mayoría no son adaptables más allá de lo que viene ya configurado. La aplicación escrita en Python recoge datos procedentes de distintos sensores que muestra y permite tomar decisiones al usuario de cómo actuar pudiendo automatizar la toma de decisiones, además permite crear alarmas, controlar la temperatura de una estancia, ver remotamente las cámaras conectadas en el hogar o encender o apagar algún dispositivo remotamente entre otras funcionalidades.

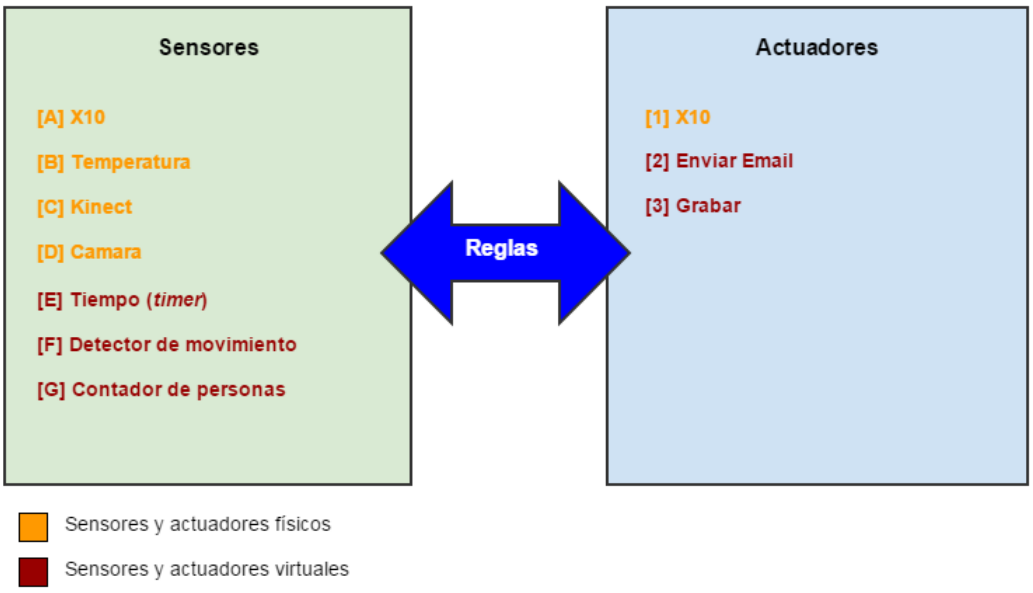

Figura 4.10: Diagrama de funcionalidades del sistema domótico HomeGuardian

En la Figura 4.10 vemos la división interna de funcionalidades que el sistema proporciona y también es la manera en la que se divide y se expone en esta memoria. Tenemos sensores o actuadores físicos o crudos y la aplicación muestra su información tal cual. Sobre los datos de los sensores crudos se puede elaborar cierta información

más sofisticada realizándoles algún procesamiento. Por ejemplo contar personas desde los datos del sensor RGBD. También hemos enriquecido las opciones de actuación con nuevas posibilidades como enviar un mensaje o grabar imágenes. A estos los llamamos sensores o actuadores virtuales. Por último tendríamos las reglas que son interacciones entre sensores y actuadores, con esto podremos enlazar acciones automáticas a ciertas condiciones de sensado. Por ejemplo, encender un actuador X10 cuando la cámara detecte movimiento.

4.3.1. Sensores y actuadores físicos

Esta aplicación está centrada en 4 módulos, cada uno de ellos ofrece una serie de sensores y actuadores que la aplicación une y comunica entre ellos permitiendo la configuración de reglas libres por parte del usuario para crear un sistema único, propio, y adaptado a cada hogar.

Estos módulos de sensores, en principio desconectados, se establecen en la aplicación en distintos paneles para tener una organización básica de los distintos servidores a los que la aplicación se conecta desde la siguiente ventana:

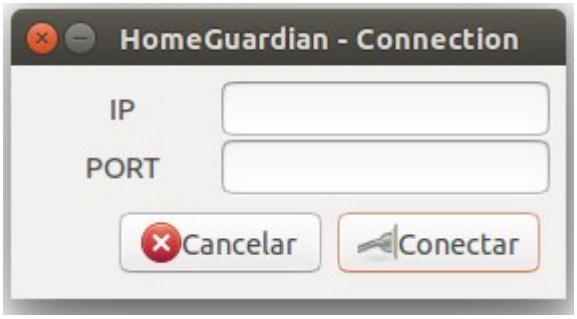

Figura 4.11: Ventana de conexión para cada servidor

Los distintos paneles de módulos que presenta la aplicación son:

- `Environment` (Entorno): En este panel se centraliza el cliente para visualizar la información de la red *X10* conectándose a un x10Server y también para visualizar la información de la temperatura conectándose a un TemperatureServer.

  En la Figura 4.12 vemos el panel Environment donde se muestra de forma estructurada la información que nos devuelve X10Server y TemperatureServer, a simple vista la parte de *X10* es muy similar a X10View ya que trae las opciones de activar, desactivar módulos, cambio de nombres y añadir o borrar dispositivos, así

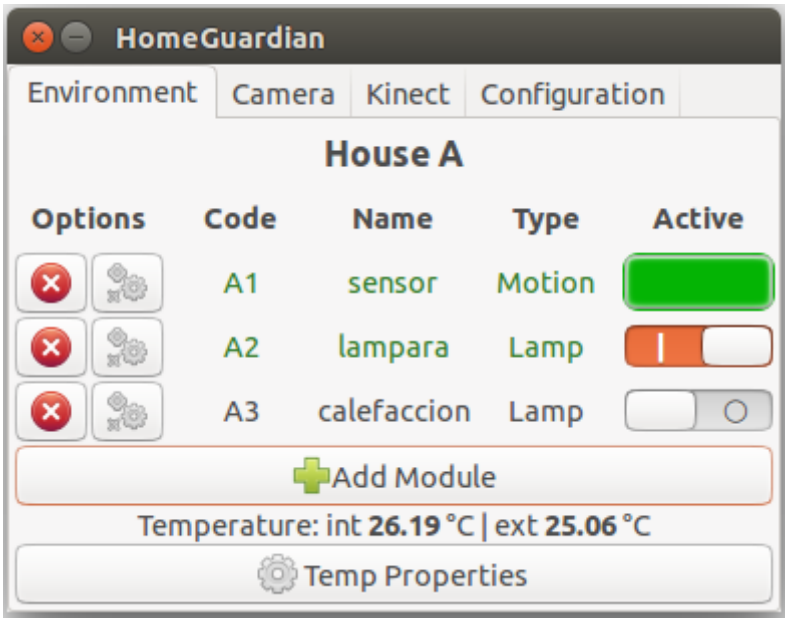

Figura 4.12: Panel de Environment

como visualizar los estados de cada uno, pero añade bastante más complementos y funcionalidades que hacen que sea una aplicación domótica bastante completa los cuales veremos en el siguiente apartado.

En la parte que nos da la información proveniente de TemperatureServer de la Figura 4.12 podemos ver la información de la temperatura de los dos sensores, un sensor interno y otro externo en grados Celsius pero esto se puede modificar en el panel de configuración de la temperatura para usar grados Fahrenheit.

- `Camera` (Cámara): En este panel podremos controlar diferentes cámaras a las cuales nos podremos conectar a través del componente CameraServer. Este panel está centrado en la seguridad del hogar. Podemos conectar todas las cámaras que queramos lo que nos permite controlar todos los puntos del hogar que queramos y visualizarlas en directo desde cualquier sitio.

  En la Figura 4.13 vemos el panel con dos cámaras conectadas cada una a un CameraServer distinto aunque dando imágenes de la misma habitación. Esto nos permite ver en directo que está sucediendo en el hogar.

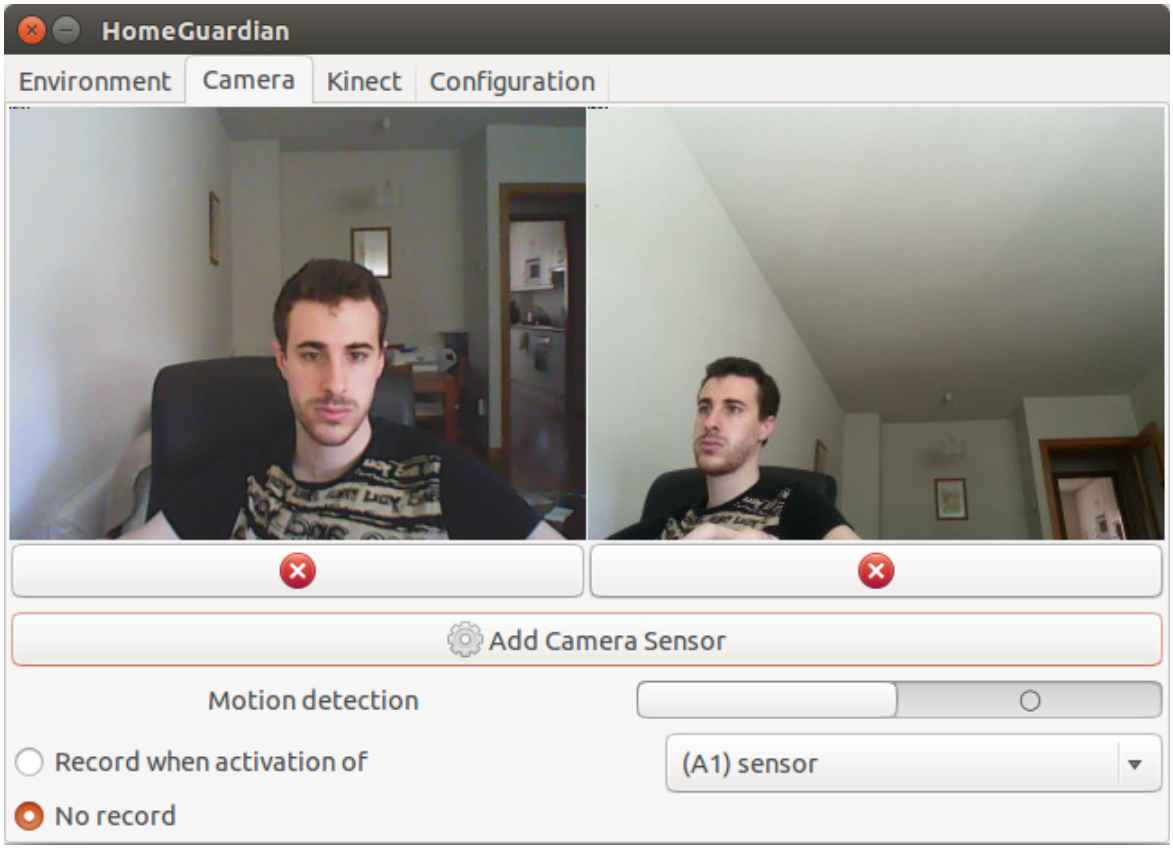

Figura 4.13: Panel para control de cámaras

- `Kinect`: En este panel tendremos una interfaz parecida al panel de Camera pero con sensores Kinect. En la siguiente imagen los sensores Kinect se tomaron sobre un mundo virtual que contiene un apartamento con 2 personas virtuales con movimientos establecidos y 2 sensores Kinect, uno en la puerta principal y otro en el salón del apartamento:

  Como podemos ver en la 4.14, tenemos conectadas 2 cámaras por cada sensor Kinect, unos mostrando la imagen real y otro mostrando una imagen de profundidad que se toma a partir de sensores infrarrojos. La ventaja que tenemos utilizando este sensor por encima de una cámara convencional del panel de Camera es que la imagen de profundidad sigue funcionando incluso en condiciones en que no existe luz.

  En la Figura 4.15 se muestra un pedazo de código de como visualiza la imagen una de las cámaras, en este caso se conecta la un servidor de sensor RGBD y pide la imagen RGB para mostrarla.

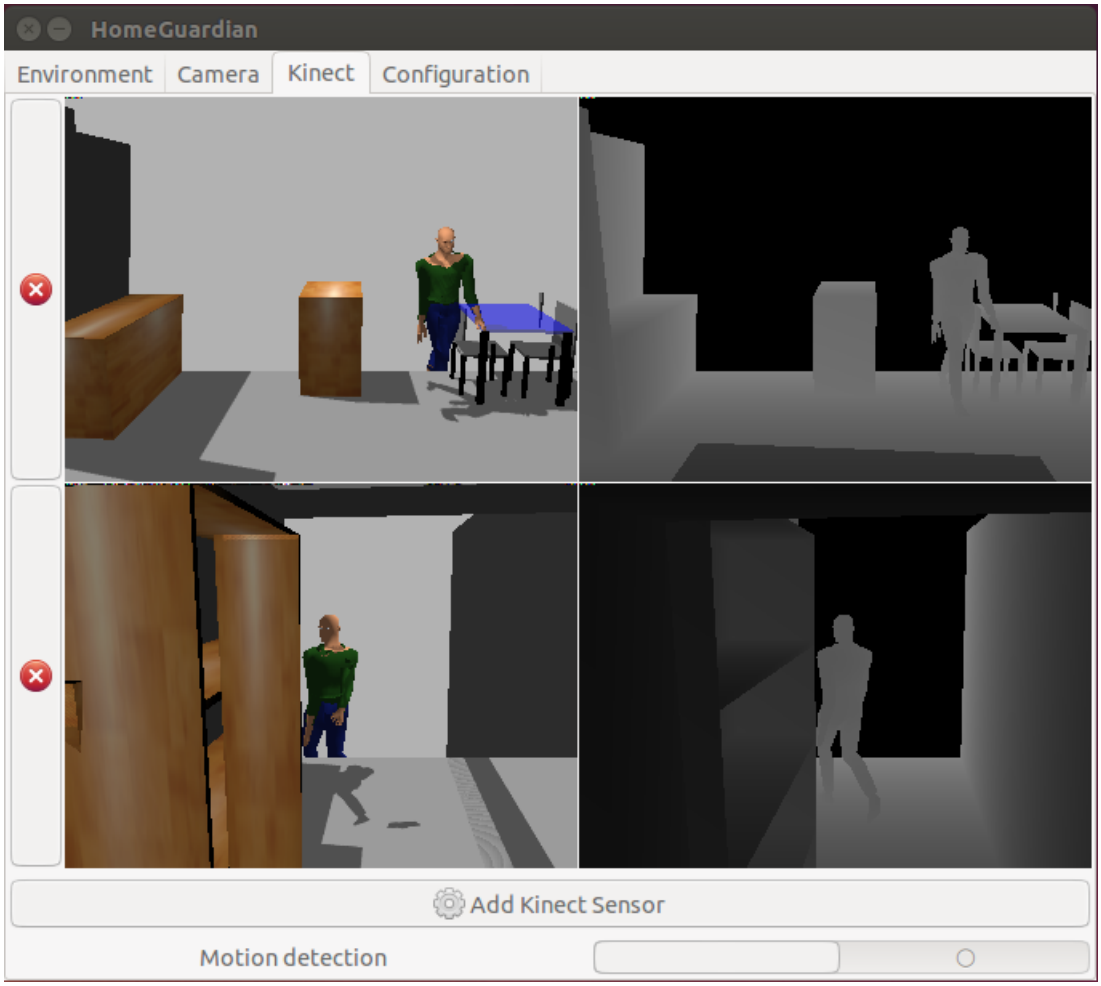

Figura 4.14: Panel para sensores Kinect en mundo virtual

```
try:
  obj2 = ic.stringToProxy('cameraRGB:default -h '+self.Kinect[self.nKinect][0]+' -p ' + self.Kinect[self.nKinect][1])
  kin = jderobot.CameraPrx.checkedCast(obj2)
except:
  print "KinectRGB: Connection Failed"
  return

formatRGB = kin.getImageFormat()[0]
while True:
  if threads == False or self.KinectRun == False or not hboxkinectcams in self.vboxkinectcams:
    break
  data = kin.getImageData(formatRGB)
  pixbuf = GdkPixbuf.Pixbuf.new_from_data(data.pixelData, GdkPixbuf.Colorspace.RGB, False, 8, data.description.width,
                                          data.description.height, data.description.width*3,None, None)


  if resolution(0) != data.description.width:
    pixbuf = pixbuf.scale_simple(resolution(0), resolution(1), GdkPixbuf.InterpType.BILINEAR);
  Gdk.threads_enter()
  kinectRGB.clear()
  kinectRGB.set_from_pixbuf(pixbuf)
  Gdk.threads_leave()
```

Figura 4.15: Pedazo de código de como se muestra la imagen en directo

En la Figura 4.16 se muestra una imagen capturada a través de un sensor Asus Xtion como ejemplo de funcionamiento con sensores reales.

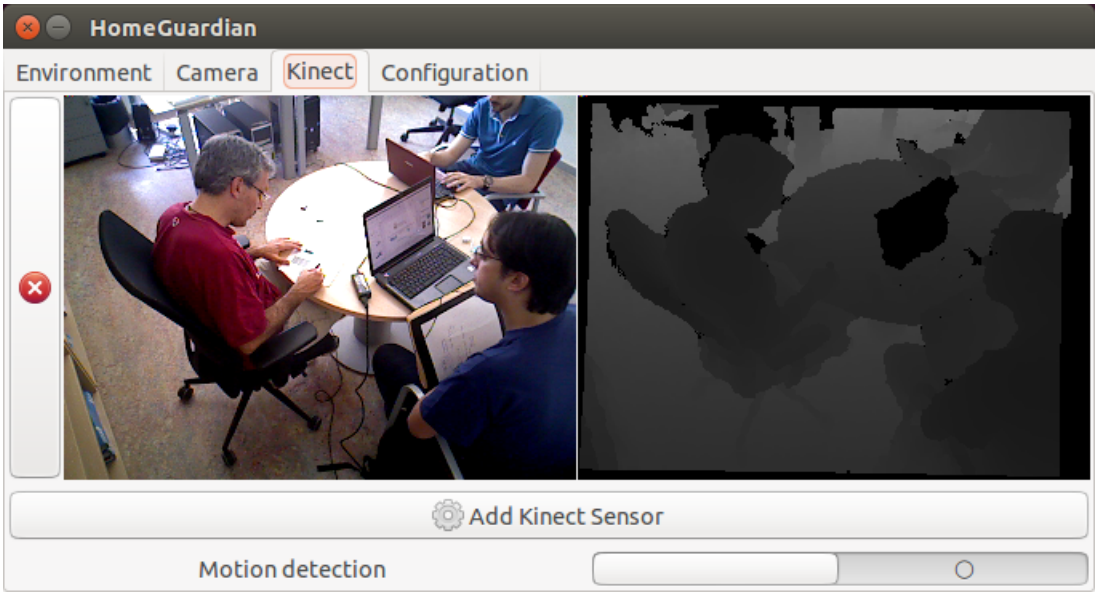

Figura 4.16: Panel para sensores Kinect con Asus Xtion

- `Configuration` (Configuración): Este panel se encarga de la configuración general de la aplicación, entre otras funcionalidades tenemos la de guardar en un fichero de configuración todas las opciones que hemos ido añadiendo tales como dispositivos *X10* y sus reglas y horarios de encendido y apagado, sensor de temperatura y sus reglas, cámaras y sensores Kinect conectados. De esta forma tendremos nuestra configuración guardada siempre que iniciemos la aplicación. Esto nos da la libertad de poder configurar el sistema de manera gráfica a partir de la aplicación y después guardar el fichero, o escribir el fichero a mano para después cargarlo al iniciar la aplicación.

  También nos permitirá elegir el correo o lista de correos si los separamos con ‘,’ a los cuales se les enviará notificaciones (si están activadas) de los sensores con sus respectivos contenidos. Y en este caso para evitar spam del correo podemos selección un intervalo de tiempo por el cual el mismo sensor no podrá enviar 2 correos consecutivos. Además, podremos seleccionar la resolución de las cámaras y Kinect con los que queramos que se muestre las imágenes ya que posiblemente al tener muchas cámaras conectadas podamos sufrir falta de espacio en el monitor. Todo lo mencionado se puede ver en la Figura 4.17.

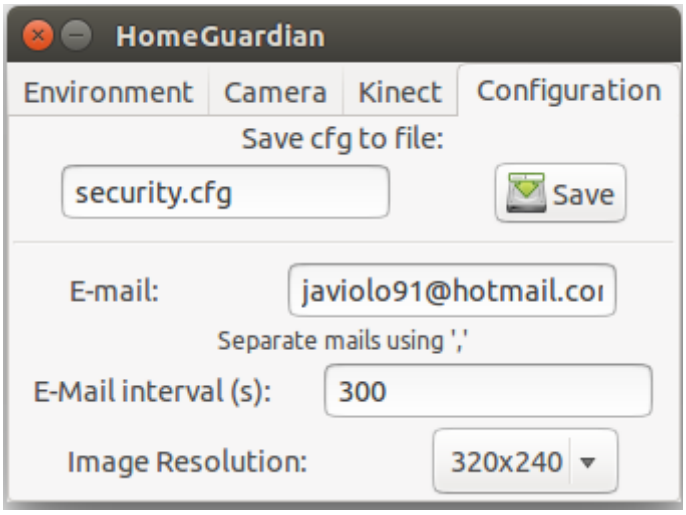

Figura 4.17: Panel de configuración general Configuration

4.3.2. Sensores y actuadores virtuales

Hasta ahora sólo hemos mostrado los datos que provienen directamente de los sensores físicos, estos datos crudos se pueden elaborar para obtener otro tipo información que pueda servirnos para otras funcionalidades. Para ello se ha diseñado un sistema que analiza los datos que provienen de los sensores físicos y ofrece información elaborada con la cual ampliar la funcionalidad y sobretodo, poder tener una información más especifica para la cual añadir reglas. Con los sensores *X10* y termómetro es más sencillo este trabajo dado que los datos que provienen ya son bastante completos, pero para la cámara y para Kinect sacar información útil requiere una mayor elaboración.

Si nos fijamos en los *sensores de cámara* podremos activar un `sensor virtual de movimiento sobre RGB` que funciona estudiando la imagen de cada cámara y que además desbloquea una serie de funcionalidades que no se mostraban y que podemos ver en la siguiente figura.

La Figura 4.18 muestra las funcionalidades añadidas cuando activamos el sensor de movimiento en la cámara, este movimiento se muestra con un rectángulo verde, y que para tratar de evitar falsos positivos el objeto que se detecta como movimiento debe cumplir un mínimo de tamaño suficientemente grande como para evitar la detección de animales de compañía como falsos positivos en el `sensor virtual de personas`. Podemos ver el número total de personas en movimiento en todas las cámaras bajo el medidor “Objects in motion”

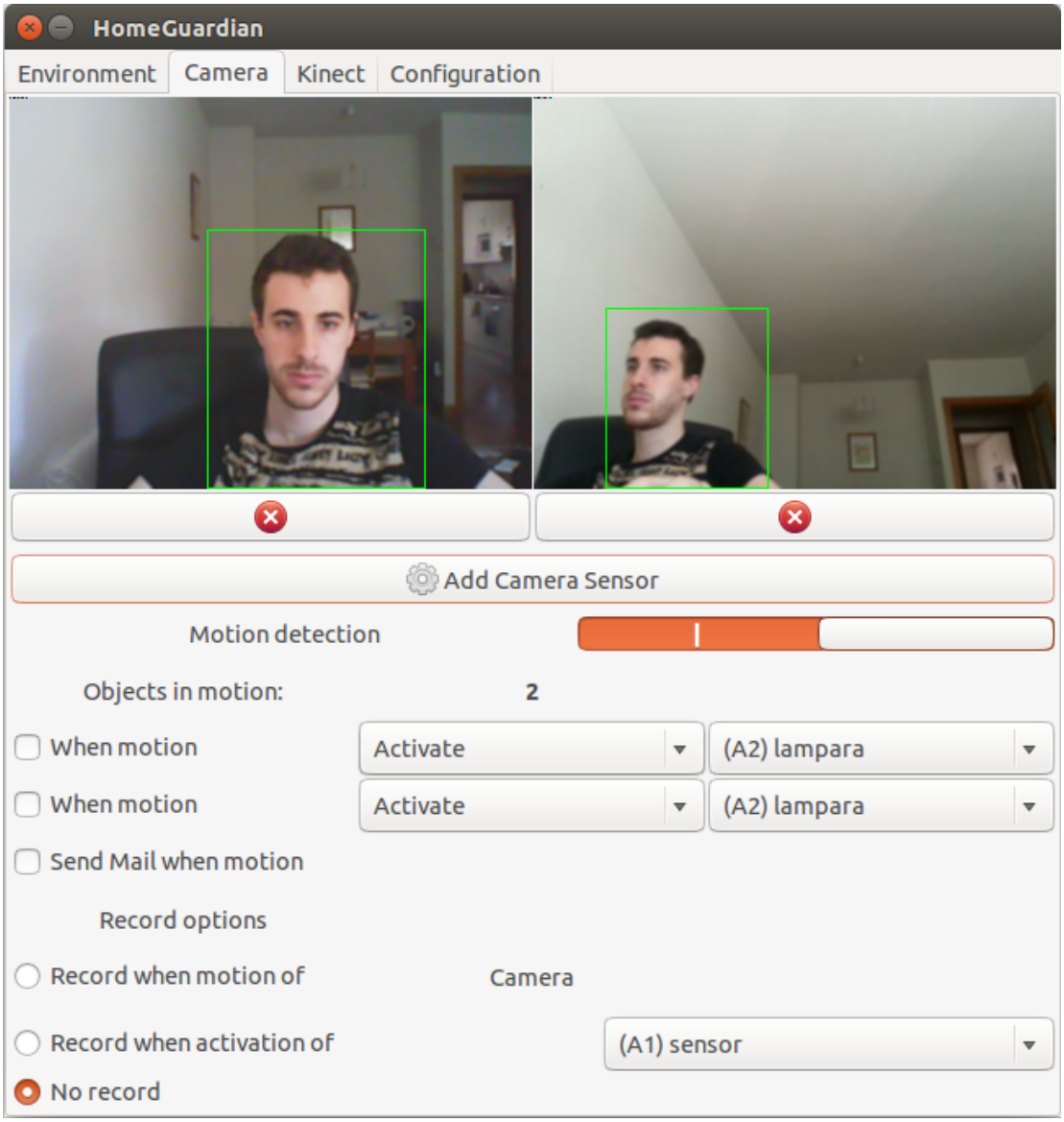

Figura 4.18: Panel de configuración para cámaras con movimiento activado

Un pedazo de código que se encarga de la detección de movimiento y su resaltado se muestra en la Figura 4.19.

```
if self.motion == True:
  # motion
  imagen = data_to_image (data, "RGB")
  color_image = cv.fromarray(imagen)

  # Smooth to get rid of false positives
  cv.Smooth(color_image, color_image, cv.CV_GAUSSIAN, 3, 0)

  if first:
    difference = color_image
    temp = color_image
    cv.ConvertScale(color_image, moving_average, 1.0, 0.0)
    first = False
  else:
    cv.RunningAvg(color_image, moving_average, 0.030, None)

  # Convert the scale of the moving average.
  cv.ConvertScale(moving_average, temp, 1.0, 0.0)

  # Minus the current frame from the moving average.
  cv.AbsDiff(color_image, temp, difference)

  # Convert the image to grayscale.
  cv.CvtColor(difference, grey_image, cv.CV_RGB2GRAY)

  # Convert the image to black and white.
  cv.Threshold(grey_image, grey_image, 70, 255, cv.CV_THRESH_BINARY)

  # Dilate and erode to get people blobs
  cv.Dilate(grey_image, grey_image, None, 18)
  cv.Erode(grey_image, grey_image, None, 10)

  storage = cv.CreateMemStorage(0)
  contour = cv.FindContours(grey_image, storage, cv.CV_RETR_CCOMP, cv.CV_CHAIN_APPROX_SIMPLE)
  points = []

  n = 0
  while contour:
    n = n +1
    bound_rect = cv.BoundingRect(list(contour))
    contour = contour.h_next()

    pt1 = (bound_rect[0], bound_rect[1])
    pt2 = (bound_rect[0] + bound_rect[2], bound_rect[1] + bound_rect[3])
    points.append(pt1)
    points.append(pt2)
    if bound_rect[2] < 50 or bound_rect[3] <50:
      n = n -1
```

Figura 4.19: Pedazo de código del sistema de detección de movimiento

El *sensor Kinect* tiene la opción de activar también un `sensor virtual de movimiento sobre RGBD`, en este caso, al ser la imagen de profundidad más fiable en cuanto a condiciones de baja luz, tendremos la detección de movimiento solo en esta imagen.

Las opciones que ahora nos muestra la Figura 4.20 cuando tenemos la detección de movimiento activada, es un contador de objetos en movimiento con un tamaño mínimo del objeto en movimiento para evitar falsos positivos con mascotas, en esta imagen en el primer sensor Kinect nos detecta una persona en movimiento que rodea con un rectángulo verde y nos indica el `número de personas` que está detectando el `sensor virtual de personas` en el apartado "Objects in motion".

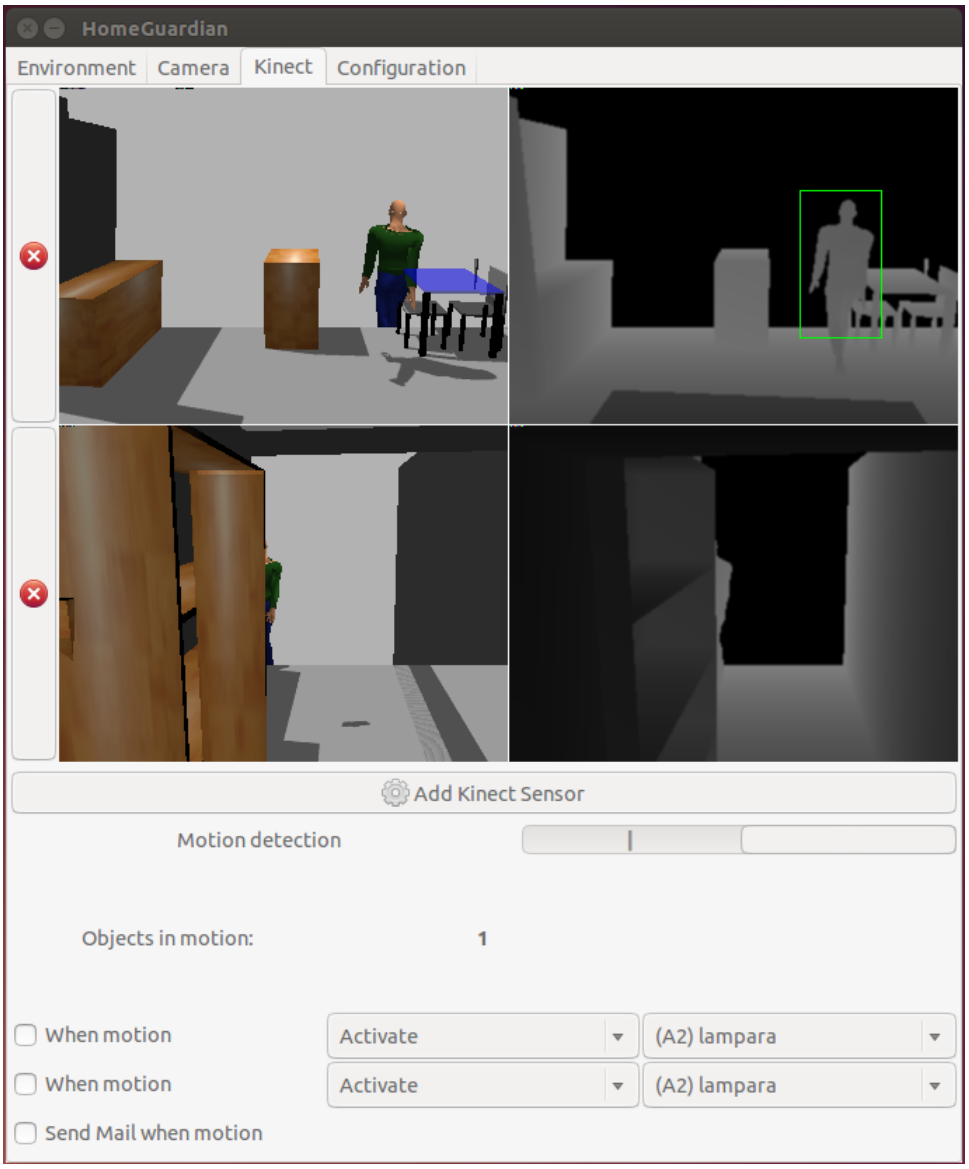

Figura 4.20: Panel Kinect extendido con control de movimiento

Si atendemos a los dispositivos *X10* y en especial a los *actuadores*, como se muestra en la Figura 4.22 de más adelante, vemos que nos permite más funciones aparte de cambiar el nombre, ahora nos permite tener un `sensor virtual de tiempo, timer`, siendo en este caso el sensor un reloj de tiempo que no necesita hardware y que ofrece como información la hora actual, con esto podemos automatizar el dispositivo donde podremos ponerle reglas para su control.

En cuanto al `control de la temperatura` tendremos la siguiente interfaz:

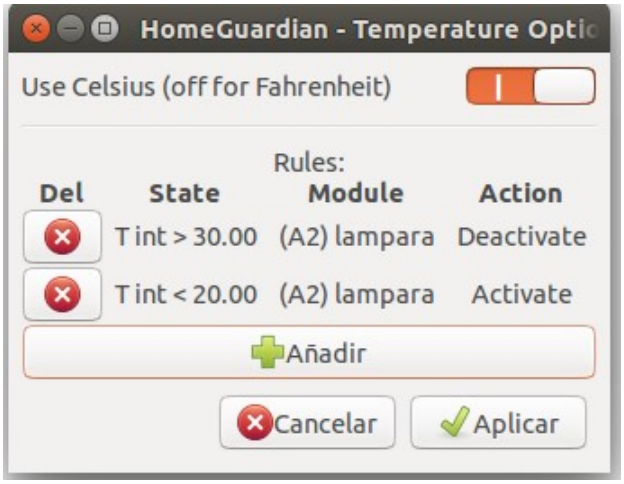

Figura 4.21: Panel de configuración para temperatura

La Figura 4.21 nos muestra como permite poner la temperatura en grados Fahrenheit o Celsius además se pueden observar reglas que se explicaran en el siguiente apartado.

También se han creado actuadores virtuales como por ejemplo `enviar un mensaje` de alarma, este actuador envía un email con información al correo o lista de correos que se puede establecer en el panel de configuración. Además, otro actuador que se añade es el de `grabar imágenes`, este, captura en un en un archivo de vídeo en formato 'xvid' con nombre la hora actual.

4.3.3. Reglas

Llamamos reglas a la conexión configurable entre sensores y actuadores, dicho de otro modo, la opción de configurar acciones en actuadores para cuando los sensores encuentren una situación determinada.

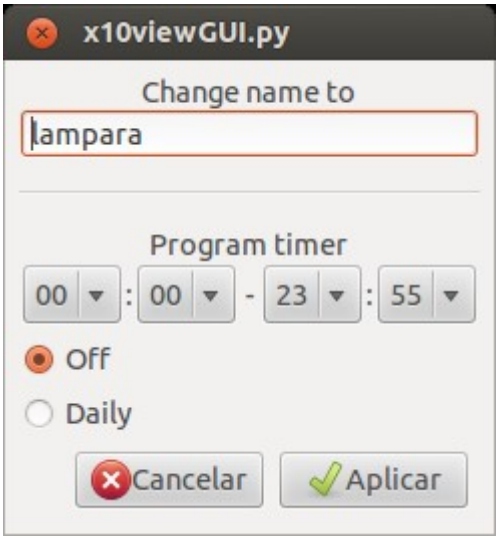

Figura 4.22: Panel configuración actuador *X10*

Para *X10* tenemos una serie de reglas, en cuanto a los `actuadores X10` si nos fijamos que la hora puede obtenerse también por un sensor virtual podríamos incorporar aquí la función `timer`, ya que en función del tiempo podemos decidir que se actúe de una manera u otra. Como vemos en la Figura 4.22.

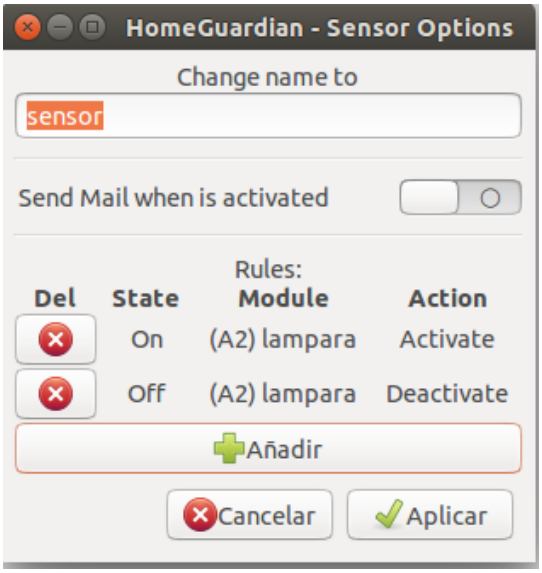

Figura 4.23: Panel configuración sensor *X10*

En cuanto a los `sensores X10` como vemos en la Figura 4.23 podremos activar una alerta por correo a la dirección deseada cuando el sensor se active para alertar al usuario de lo que sucede en ese momento en el hogar.

También nos permite añadir una serie de reglas con acciones personalizables para que se realicen cuando se active o desactive el sensor, podremos eliminarlas cuando ya no sean útiles y añadir tantas reglas como queramos, estas reglas estan dirigidas a actuadores X10. Se pueden añadir a partir de la interfaz que se muestra en la Figura 4.24

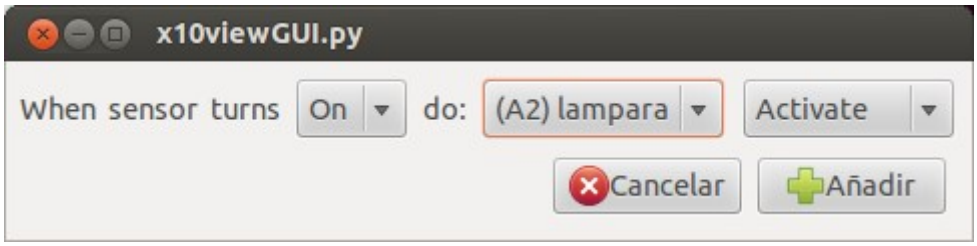

Figura 4.24: Panel para añadir reglas a sensor *X10*

En la Figura 4.24 vemos como añadir reglas de una forma sencilla: "When sensor turns On do: (A2) lámpara Activate" que vendría a ser "Cuando el sensor se encienda hacer: (A2) lámpara Activar". Es decir, cuando el sensor se active al detectar movimiento, mandara una orden para que se active el modulo (A2) lámpara.

Si observamos el `sensor de temperatura`, en la Figura 4.21 vemos una serie de reglas para actuadores *X10* que podremos añadir en función de la temperatura de cada uno de los sensores tanto interior como exterior, de este modo podremos activar un módulo *X10* si la temperatura interior baja de una cantidad de grados, o desactivarla si la temperatura sube de otra. Podremos añadir tantas reglas como queramos y eliminarlas a nuestro antojo gracias a la interfaz que se muestra a continuación.

Figura 4.25: Panel para añadir reglas de temperatura

En la Figura 4.25 nos muestra una interfaz sencilla donde podremos configurar las reglas que queramos para la temperatura y que intenta que sea como leer una frase "When Temperature T int ¿1 grado: (A2) lámpara Activate" que viene a ser "Cuando la Temperatura interior es mayor de 1º hacer: (A2) lámpara Activar". Es decir, cuando la temperatura del sensor interior sea mayor de 1, se mandará automáticamente una

orden para activar el modulo (A2) lámpara. Gracias a esto podemos conectar el sensor de temperatura con actuadores *X10* lo que nos abre un amplio abanico de posibilidades de configuración.

El `sensor Camera` tiene además entre las opciones desbloqueadas que se observan en la Figura 4.18 dos ranuras para añadir reglas relacionadas con módulos *X10*, de este modo cuando se detecte movimiento en alguna de las cámaras podremos activar o desactivar un módulo *X10*, así unimos este sensor con el entorno *X10* y aumentamos el número de posibilidades de seguridad.

Otra opción que nos permite este panel es enviar un correo a una dirección deseada de forma automática cuando exista movimiento, este correo muestra información de lo que está sucediendo en ese instante y contiene la imagen de la cámara que detecta el movimiento para ver de manera exacta lo que ocurre.

La última opción que se añade se trata de no grabar el vídeo de las cámaras, comenzar a grabar cuando un sensor de un módulo *X10* se active o grabar en un archivo la cámara que este detectando movimiento en ese momento, se establecen una serie de reglas para que si por ejemplo una persona entra en la habitación y se queda parada un momento y luego vuelve a caminar, evitar tener dos archivos distintos uno para el primer movimiento y otro para el segundo, sino que tendremos un solo archivo con todo lo sucedido.

En el `sensor Kinect`, como vemos en la Figura 4.20, tenemos la posibilidad de enlazar 2 reglas de activación o desactivación de módulos *X10* de la misma forma que cuando tenemos un sensor Cámara, cuando se detecte movimiento en alguno de los sensores Kinect podremos activar o desactivar cualquier actuador *X10* y así ampliar la funcionalidad de estos sensores.

Por último y al igual que en el sensor Cámara, nos permite crear una alerta de correo electrónico cuando en alguno de los sensores Kinect se detecte movimiento, en el correo viajara la información de lo que está ocurriendo en ese momento así como las dos imágenes del sensor que se ha activado para mostrar realmente lo que sucede y porque se ha activado.

Como resumen, las reglas que tenemos disponibles si atendemos a la Figura 4.10

- A1 La activación de un sensor X10 puede activar un módulo X10
- A2 La activación de un sensor X10 permite alertas por email
- A3 La activación de un sensor X10 permite activar la grabación de la cámara.
- B1 Alcanzar una cierta temperatura permite activar un módulo X10

- CF1 Si Kinect detecta movimiento permite encender un módulo X10
- CF2 Si Kinect detecta movimiento permite enviar una alerta por email con imágenes
- DF1 Si la cámara detecta movimiento permite encender un módulo X10
- DF2 Si la cámara detecta movimiento permite enviar una alerta por email con imágenes
- DF3 Si la cámara detecta movimiento permite grabar lo que esta sucediendo

4.3.4. Ejemplos prácticos

Estas son algunas de las funcionalidades de las que podremos sacar partido a esta aplicación dado su amplitud de opciones y configuraciones que hacen de esta aplicación un sistema domótico tan eficiente:

- Control calefacción por la temperatura:

  Gracias a este sistema podremos ahorrar energía ya que podemos crear nuestro propio termostato. Esto podremos realizarlo gracias al sensor de temperatura y las reglas de configuración que nos permite enlazar con sensores *X10*. De la misma manera como se muestra en la Figura 4.25, podemos contemplar que cuando el termómetro descienda de 20 grados, enviar una orden de encendido a un actuador *X10* que estará conectado a un sistema de calefacción, o radiadores. Del mismo modo, podemos poner otra regla para que se encienda un ventilador o aparato de refrigeración cuando la temperatura ascienda de 30 grados.

- Aviso por correo de intrusiones en casa:

  Como medida de seguridad, podremos dejar las cámaras o Kinect activados al salir de casa o ir de vacaciones, si activamos la casilla “Send Mail when Motion” podremos obtener un correo con las imágenes de que está sucediendo en casa cuando uno de estos sensores detecte movimiento. Gracias a la visualización en tiempo real, podremos ver las cámaras de forma remota tras el aviso si tenemos acceso a la aplicación.

- Simulación de gente en casa para ahuyentar intrusos.

  Gracias al `timer` que nos ofrecen los actuadores *X10* podremos dejar horarios de activación o desactivación para lámparas, televisiones, radios, etc. Que simularán que hay alguien dentro de la casa y así evitar posibles intrusiones en periodos vacacionales.

- Encendido de luces o alarmas en intrusiones

  Si alguna de las cámaras o sensores de movimiento detectase una intrusión, para intentar ahuyentar o asustar al intruso y que se vaya podemos conectar el sensor con reglas que vemos en la Figura 4.18 o Figura 4.20 que activen lámparas, sirenas, televisiones cuando detecte movimiento en el hogar.

- Encendido de luces cuando hay alguien en la sala y auto apagado cuando no.

  Para evitar dejar luces encendidas u otros aparatos podemos conectar el sensor de movimiento TM13 con reglas para activar la luz si entra alguien en la habitación y apagarla cuando la habitación este vacía para poder ahorrar energía.

- Bajar persianas en verano por temperatura.

  Si en la casa tenemos persianas motorizadas, podremos conectar nuestro sensor de temperatura tal como indica la Figura 4.25, para que en verano, si la temperatura de la casa asciende de cierta temperatura activar el motor de apertura o cierre de persianas.

- Grabación de intrusos o vigilancia de personas internas.

  Podremos tener vigilancia en directo de manera remota de lo que ocurre en casa a través de la aplicación, o dejar grabando las cámaras si en ese momento no podemos acceder a ellas pero queremos visualizarlas más adelante. De este mismo modo, podremos enviar una alarma por correo o enlazando el sensor con un actuador *X10* si una persona con sonambulismo se acerca a la puerta principal de entrada durante la noche.

- Control de aparatos electrónicos de la casa a partir de la aplicación.

  Como ahorro energético podremos apagar luces encendidas, encender la televisión si queremos verla desde la aplicación gracias al sistema *X10* que podemos controlar desde el panel de Environment.

Capítulo 5

Conclusiones

5.1. Conclusiones

Si tomamos conclusiones fijándonos en los objetivos se puede decir que se han cumplido todos ellos, ya que el objetivo final era crear un sistema domótico funcional.

Si nos referimos a los submódulos en que dividimos la aplicación, hemos creado un driver para controlar un sistema X10 configurable a nuestro antojo basándonos en el software *HEYU* que ha ayudado para la comunicación con la base central *X10*.

Además también se ha creado un servidor que actúa como sensor de temperatura con un dispositivo *TEMPer2* conectado al USB, aunque también acepte otros modelos de este tipo de sensor. Para esto se ha tenido que coger como base *PCSensor* que ayuda en la comunicación por USB con el dispositivo.

Por último, se ha diseñado un sistema domótico uniendo los sensores que se han utilizado y que reúnen todos los datos necesarios para crear una aplicación domótica de este estilo enfocada principalmente en seguridad pero también en ahorro energético y comodidad.

El sistema domótico se ha conseguido gracias a esta aplicación permite el completo control de un hogar generando comodidad con el encendido y apagado de aparatos electrónicos a distancia o utilizando horarios. Además se puede conseguir ahorro energético con un uso inteligente de encendido y apagado de aparatos gracias a sensores de presencia, o ahorro de energía en aparatos de control de temperatura que dependen de la temperatura. Por último, se han cumplido los objetivos de conseguir una aplicación de seguridad que contiene todas las funcionalidades que el mercado domótico ofrece y añade combinaciones adaptables en este aspecto que lo convierten en un sistema mas selecto.

Para conseguir esta funcionalidad de la que este proyecto presume se ha tenido que crear reglas que unieran en funcionamiento de cada uno de los sensores o módulos, por lo que se añade no ya la funcionalidad que se puede sacar de cada sensor, si no que se amplía a la funcionalidad que todos ellos en común pueden dar.

En total el trabajo ocupa 4505 líneas de código divididas en:

- X10Server: 237
- X10View: 485
- TemperatureServer: 82
- HomeGuardian: 1934
- Interfaces y Clases 1767

Por último, aunque el mundo del apartamento virtual para Gazebo utilizado como simulación para los dispositivos Kinect en su mayoría ya estaba hecho, por lo que no quiero contar como líneas de código de trabajo pero merece la pena comentarlo, ocupa un total de 5880 líneas de código.

5.2. Pasos futuros

Esta aplicación puede tomar bastantes pasos en un futuro. Para obtener un sistema de seguridad con sirena, podremos hacernos con el módulo *X10 PowerHorn* [32], de esta manera y sin tener que hacer ninguna actualización en la aplicación podremos contar con una sirena anti-intrusiones con solo activar el actuador *X10* al que está configurado cuando por ejemplo una cámara detecte movimiento.

Otro paso a conseguir sería mejorar el sistema de detección de personas para hacerlo más fiable contra falsos positivos que controlarlo solamente midiendo el tamaño en píxeles del objeto.

Otro posible paso sería adaptar la aplicación para que funcione en sistemas móviles, ya sea utilizando el navegador o con una aplicación similar adaptada a estos dispositivos.

Bibliografía

[1] Proyecto Sara Marugán. 2010.
http://jderobot.org/Salons-tfm.

[2] TFM Roberto Calvo Palomino. 2010.
http://svn.jderobot.org/users/rocapal/master/memoria/memoria.pdf.

[3] ElderCare. 2013.
http://jderobot.org/ElderCare.

[4] PFC Daniel Castellano Bonilla. 2013.
https://svn.jderobot.org/users/d.castellanob/pfc-teleco/texto/PFC.pdf.

[5] Surveillance 4.0. 2013.
http://jderobot.org/Surveillance.

[6] Juan Navarro Bosgos. FlyingKinet 2013-2015.
http://jderobot.org/Jnbosgos#Flying$_K$inect.

[7] Surveillance 5.1. 2014.
http://jderobot.org/Aerobeat-colab.

[8] CameraServer.
http://jderobot.org/index.php/Drivers#cameraserver.

[9] Universidad Rey Juan Carlos.
https://www.urjc.es/.

[10] Estándar CEBus.
http://en.wikipedia.org/wiki/CEBus.

[11] GitHub de este TFG.
https://github.com/Javii91/Domotic.

[12] MediaWiki de este TFG.
http://jderobot.org/Hernando-tfg.

[13] Estándares domóticos.
http://www.iecor.com/domotica-cordoba/informacion/domotica-estandares-protocolos.html.

[14] Delta Dore.
http://www.deltadore.com/spain/es/casa-domotica.html.

[15] ¿Qué es la domótica?
http://www.cedom.es/sobre-domotica/que-es-domotica.

[16] Glade.
https://glade.gnome.org/.

[17] GTK+.
http://www.gtk.org/.

[18] HEYU.
http://heyu.tanj.com/.

[19] HomeGuardian.
http://jderobot.org/Hernando-tfg#HomeGuardian.

[20] ICE.
https://zeroc.com/ice.html.

[21] JdeRobot.
http://jderobot.org/Main_Page.

[22] Kinect.
http://es.wikipedia.org/wiki/Kinect.

[23] KinectServer.
http://jderobot.org/index.php/Drivers#KinectServer.

[24] Estándar KNX.
http://es.wikipedia.org/wiki/KNX.

[25] Libelium.
http://www.libelium.com/es/.

[26] Estándar LonWorks.
http://en.wikipedia.org/wiki/LonWorks.

[27] TFG Edgar Alejandro Barrero Mateo.2014.
https://svn.jderobot.org/users/aerobeat/colab/trunk/TFG/memoria.pdf.

[28] Nest.
https://nest.com/.

[29] An Overview of the Simple Control Protocol (SCP). Abril 2005.
http://download.microsoft.com/download/c/6/2/c62fe596-74ef-4307-8eea-17d9835e28b1/scpwhitepaper.doc.

[30] OpenCV.
http://opencv.org/.

[31] PCSensor para TEMPer2.
https://github.com/peterfarsinsen/pcsensor.

[32] X10 PowerHorn.
http://www.amazon.com/PowerHorn-Decibel-Security-PH508-PSH01/dp/B0007PANF0.

[33] Prosegur.
http://www.prosegur.es/esp/hogares-y-personas/domotica/.

[34] Python.
https://www.python.org/.

[35] TEMPer2.
http://pcsensor.com/usb-thermometer/temper2.html.

[36] TemperatureServer.
http://jderobot.org/Hernando-tfg#TemperatureServer.

[37] Verisure.
http://www.securitasdirect.es/es/verisure.

[38] Estándar X10.
http://es.wikipedia.org/wiki/X10.

[39] X10Server.
http://jderobot.org/Hernando-tfg#x10server.

[40] Z-Wave.
http://www.z-wave.com/.