

Control de una casa domótica para personas dependientes

Proyecto Final de Carrera
Ingeniería superior en Informática

Facultat d'Informàtica de Barcelona
Daniel Hernández Portugués

Tabla de contenido

1	Introducción	8
1.1	Centro de Vida Independiente (CVI)	8
1.1.1	¿Qué es?	8
1.1.2	Modelo de Investigación + Desarrollo	12
1.2	Objetivos	13
1.3	Motivación	13
2	Estado del arte	14
2.1	Domótica	14
2.1.1	Arquitecturas	15
2.1.2	Sistemas de comunicación	16
2.1.3	Protocolos y estándares	16
2.2	Tecnología Konnex	18
2.2.1	Historia del estándar	18
2.2.2	KNX en la actualidad	19
2.2.3	Principales ventajas	20
2.2.4	Modos de configuración	22
2.2.5	Medios de transmisión	23
3	Estudio de las tecnologías utilizadas en el CVI	24
3.1	Konnex	25
3.2	Infrarrojos	26
3.3	Radiofrecuencia	27
3.4	Domótica Fagor	28
3.4.1	Servidor Fagor	28
3.4.2	Electrodomésticos	29
3.4.3	Interfaces de control	30
3.5	Dispositivos independientes	31
4	Requisitos de diseño	32
5	Desarrollo de una API para interactuar con el sistema KNX	36
5.1	Elección de la tecnología a utilizar	36
5.2	Calimero	37
5.3	Implementación	38
6	Desarrollo de un extensor inalámbrico WiFi	40
6.1	Diseño del extensor	40

6.1.1	Pruebas de generación de señales IR con el PIC16F876	40
6.1.2	Pruebas con el dsPIC30F4011 y la TCP/IP Stack de Microchip	42
6.1.3	Pruebas con el PIC24FJ128GA010 y el módulo ZeroG ZG2100M PICtail Plus	43
6.2	Componentes definitivos a utilizar	45
6.2.1	Módulo WiFi MRF24WB0MA	45
6.2.2	Microcontrolador PIC24FJ256GA106	47
6.2.3	Módulo WiFi FlyPort	49
6.3	Firmware del Microcontrolador	52
6.3.1	TCP/IP Stack de Microchip	52
6.3.2	Aplicación servidor TCP	54
6.3.3	Protocolo de comunicación para el envío de las órdenes	55
6.3.4	Emisión de la señal de infrarrojos	56
6.4	Implementación del prototipo	59
6.4.1	Material utilizado	59
6.4.2	Esquema	59
6.4.3	Diseño de la etiqueta	60
6.4.4	Resultado final	61
7	Adaptación de la API para la interacción con el extensor inalámbrico	64
8	Desarrollo de la aplicación de escritorio	66
8.1	Elección del entorno de desarrollo	66
8.2	Estructura	67
8.3	La interfaz gráfica	68
9	Desarrollo de la aplicación móvil	74
9.1	Elección del entorno de desarrollo	74
9.2	Estructura	75
9.3	La interfaz Gráfica	76
10	Adaptación de la API para su funcionamiento con la librería DLA	78
10.1	Definición del DLA	78
10.2	Funcionamiento del DLA	79
10.3	Consultas implementadas	81
10.3.1	WriteDomoticControl	81
10.3.2	ReadStateDomoticControl	81
10.3.3	ReadDevicesXML	81
10.4	XML con los dispositivos accesibles	82

11	Evaluación de resultados	84
12	Valoración económica	86
12.1	Coste de la fabricación del prototipo	86
12.2	Coste de desarrollo del PFC	87
13	Planificación seguida	88
14	Posibles ampliaciones o mejoras	90
15	Anexos	92
	ANEXO I: Manual para la utilización de la API	92
	ANEXO II: Protocolos de comunicación IR implementados	94
	Philips RC-5	94
	Sony SIRC	96
	Panasonic	97
	Daewoo	98
	JVC	99
	ANEXO III: Código principal de la API	100
	ANEXO IV: Código del servidor TCP implementado en el PIC24	102
	ANEXO V: Código de generación de la señal IR desde el PIC24	108
	ANEXO VI: Funciones del DLA	112
16	Bibliografía	120
17	Agradecimientos	122

Índice de Figuras

FIGURA 1.1: CENTRE DE VIDA INDEPENDENT	8
FIGURA 1.2: ÁREAS DEL CENTRO	9
FIGURA 1.3: ÁREA DE VIVIENDA	10
FIGURA 1.4: ZONAS DEL ÁREA DE VIVIENDA	10
FIGURA 1.5: ÁREA DE I+D+I	11
FIGURA 1.6: ZONA DE SERVIDORES	11
FIGURA 1.7: ATENCIÓN PERSONALIZADA.....	11
FIGURA 1.8: ÁREA PERSONALIZADA	11
FIGURA 1.9: SALA DE REUNIONES	11
FIGURA 1.10: MARCO DE COLABORACIÓN	12
FIGURA 2.1: ARQUITECTURA CENTRALIZADA.....	15
FIGURA 2.2: ARQUITECTURA DISTRIBUIDA.....	15
FIGURA 2.3: ARQUITECTURA MIXTA	15
FIGURA 2.4: LOGO KNX.....	18
FIGURA 2.5: USO DE LAS DIFERENTES TECNOLOGÍAS DOMÓTICAS EN EL MUNDO	19
FIGURA 2.6: PROGRESO DEL USO DE DIFERENTES PROTOCOLOS EN LOS ÚLTIMOS CUATRO AÑOS	20
FIGURA 2.7: INTEROPERABILIDAD DE PRODUCTOS.....	20
FIGURA 2.8: TODAS LAS FUNCIONES Y APLICACIONES.....	21
FIGURA 2.9: KNX SE ADAPTA A TODO TIPO DE CONSTRUCCIONES.....	21
FIGURA 2.10: FUNCIONALIDAD / SOFISTICACIÓN DE LOS MODOS DE CONFIGURACIÓN.....	22
FIGURA 2.11: MEDIOS DE TRANSMISIÓN	23
FIGURA 3.1: ISLAS DE CONTROL PRESENTES EN EL CVI	24
FIGURA 3.2: DETECTOR DE PRESENCIA.....	25
FIGURA 3.3: BOTÓN DE ALARMA	25
FIGURA 3.4: BOTONERA PARA LAS LUCES	25
FIGURA 3.5: DETECTOR DE HUMO	25
FIGURA 3.8: PUERTA	26
FIGURA 3.8: VENTANA	26
FIGURA 3.8: TOLDOS.....	26
FIGURA 3.9: CAMA.....	26
FIGURA 3.10: DISPOSITIVOS MULTIMEDIA.....	26
FIGURA 3.11: MANDOS INFRARROJOS	26
FIGURA 3.12: BOTONES RF	27
FIGURA 3.13: MANDO UNIVERSAL HARMONY® 1100 ADVANCED DE LOGITECH	27
FIGURA 3.14: SERVIDOR FAGOR	28
FIGURA 3.15: MÓDULO MC-400	28
FIGURA 3.16: MÓDULO MIT-400.....	29
FIGURA 3.17: MÓDULO MB-300.....	29
FIGURA 3.18: FILTRO DE RED DOMÓTICA.....	29
FIGURA 3.19: ELECTRODOMÉSTICOS FAGOR.....	29
FIGURA 3.20: MAIOR VOCCE	30
FIGURA 3.21: GRÚA	31
FIGURA 3.22: SOFÁ.....	31
FIGURA 3.23: RETRETE	31
FIGURA 3.24: ARMARIO DE ALTURA AJUSTABLE	31
FIGURA 4.1: ESQUEMA BÁSICO DEL DISPOSITIVO SEGÚN LOS REQUISITOS	32
FIGURA 4.2: FUNCIONAMIENTO DEL EXTENSOR LOGITECH.....	33
FIGURA 4.3: ESQUEMA ILUSTRATIVO DEL FUNCIONAMIENTO DEL EXTENSOR INALÁMBRICO	33

FIGURA 4.4: ISLAS DE CONTROL EN EL CVI UNA VEZ REALIZADO EL PROYECTO.....	34
FIGURA 5.1: ARQUITECTURA WAIST-LINE Y EL SISTEMA NO ESTRICTO DE CAPAS EN CALIMERO.....	37
FIGURA 5.2: ESQUEMA DE LAS INTERFACES.....	38
FIGURA 5.3: ESQUEMA UML DE LA API.....	39
FIGURA 6.1: ESQUEMA DE LAS PRUEBAS CON EL PIC16F876.....	40
FIGURA 6.2: CAPTURA DEL OSCILOSCOPIO DE UNA SEÑAL DE 40 KHz	41
FIGURA 6.3: CAPTURA DEL OSCILOSCOPIO DE UNA SEÑAL DE CEROS Y UNOS MODULADA A 40 KHz	41
FIGURA 6.4: MANDO LOGITECH.....	42
FIGURA 6.5: IMAGEN DEL DSPICDEM 2.....	42
FIGURA 6.6: ADAPTADOR TQFP 100 v2.0.....	43
FIGURA 6.7: ESQUEMA DE INTERCONEXIÓN PIC24- ZG2100M	44
FIGURA 6.8: ZEROG ZG2100M PICTAIL PLUS.....	44
FIGURA 6.9: MÓDULO MRF24WB0MA.....	45
FIGURA 6.10: DIMENSIONES DEL MÓDULO MRF24WB0MA	45
FIGURA 6.11: DIAGRAMA DE BLOQUES.....	46
FIGURA 6.12: INTERFAZ MICROCONTROLADOR - MRF24WB0Mx	46
FIGURA 6.13: MICROCONTROLADOR PIC24FJ256GA106	48
FIGURA 6.14: FLYPORT DESDE ARRIBA	49
FIGURA 6.15: FLYPORT DESDE ABAJO	49
FIGURA 6.16: ESQUEMA DE BLOQUES DEL MÓDULO FLYPORT	49
FIGURA 6.17: ESQUEMA DEL MÓDULO FLYPORT	50
FIGURA 6.18: CAPAS DEL MODELO TCP/IP DE REFERENCIA	52
FIGURA 6.19: COMPARACIÓN DEL MODELO TCP/IP DE REFERENCIA Y LA PILA TCP/IP DE MICROCHIP	52
FIGURA 6.20: DIAGRAMA DE ESTADOS DEL SERVIDOR TCP	54
FIGURA 6.21: ESQUEMA DE CONEXIONES DEL PROTOTIPO	60
FIGURA 6.22: ETIQUETA DEL PROTOTIPO	61
FIGURA 6.23: PROTOTIPO FOTO 1	61
FIGURA 6.24: PROTOTIPO FOTO 2	62
FIGURA 6.25: PROTOTIPO CONECTADO Y CON EL CABLE IR	62
FIGURA 7.1: ESQUEMA UML DE LA API ACTUALIZADO.....	64
FIGURA 8.1: PARTES DE LA INTERFAZ.....	68
FIGURA 8.2: INTERFAZ GRÁFICA, MENÚ EDITAR	69
FIGURA 8.3: EDITAR ZONA	69
FIGURA 8.4: MENÚ DESPLEGABLE DE ZONAS	70
FIGURA 8.5: EDITAR DISPOSITIVO	70
FIGURA 8.6: ZONA CON DISPOSITIVOS EN EL MAPA	71
FIGURA 8.7: DRAG&DROP	72
FIGURA 8.8: MENÚ DESPLEGABLE	72
FIGURA 8.9: CONTROL DIMMER	73
FIGURA 8.10: CONTROL TV	73
FIGURA 9.1: ESQUEMA MÓVIL-SERVIDOR-API	75
FIGURA 9.2: PANTALLA PRINCIPAL DE LA APLICACIÓN ANDROID.....	76
FIGURA 9.3: MENÚ DESPLEGABLE PARA LA SELECCIÓN DEL TIPO DE DISPOSITIVO	76
FIGURA 9.4: MENÚ DESPLEGABLE PARA LA SELECCIÓN DEL DISPOSITIVO	77
FIGURA 9.5: ENVÍO DE LA ORDEN SOLICITADA	77
FIGURA 10.1: REPRESENTACIÓN DE LA INDEPENDENCIA ENTRE LA APLICACIÓN Y LA API CON EL DLA	78
FIGURA 10.2: ELEMENTOS QUE COMPONENTE LA ARQUITECTURA DE CONTROL DLA	79
FIGURA 11.1: PIRÁMIDE DEL CONTROL DEL CVI	84
FIGURA 11.2: EJEMPLO DE FUNCIONAMIENTO DEL CONTROL DE LA CAMA	84
FIGURA 11.3: EJEMPLO DE FUNCIONAMIENTO DEL CONTROL DE UNA LUZ DE TIPO DIMMER	85

FIGURA 11.4: EJEMPLO DE FUNCIONAMIENTO DEL CONTROL DE LA TELEVISIÓN	85
FIGURA 12.1: TABLA DE PRECIOS DE LOS COMPONENTES	86
FIGURA 12.2: TABLA DEL COSTE PARA LA REALIZACIÓN DE ESTE PROYECTO	87
FIGURA 13.1: DIAGRAMA DE GANTT DE LA PLANIFICACIÓN SEGUIDA	88

1 Introducción

Este proyecto final de carrera ha consistido en la elaboración de una API para el control de dispositivos en redes KNX, así como la implementación de una aplicación de escritorio y otra para móviles Android con el fin de controlar dichos dispositivos desde cualquier lugar vía TCP/IP. Además, se ha diseñado, implementado y construido un prototipo de extensor inalámbrico WiFi para el control de dispositivos multimedia, como el televisor o el reproductor de música, a través de esta misma API.

Este proyecto nace de la colaboración del departamento de ESAlI de la UPC con el Centro de Vida Independiente (CVI), empresa especializada en tecnología domótica para personas discapacitadas, con la intención de satisfacer una necesidad inminente como es la de controlar los diferentes elementos de una casa desde un mismo dispositivo apto para personas con diferentes discapacidades.

A lo largo de esta memoria he expuesto de forma detallada todo el proceso de la realización de este PFC. A continuación explico de forma más detallada que es el CVI, como los objetivos del proyecto y la motivación para realizarlo.

1.1 Centro de Vida Independiente (CVI)



Figura 1.1: Centre de Vida Independent

1.1.1 ¿Qué es?

El Centro de Vida Independiente (CVI) fue inaugurado el 2 de junio de 2008 por el conseller d'Innovació, Universitats i Empresa, Josep Huguet, y la consellera d'Acció Social i Ciutadania, Carme Capdevila. Está situado en el Centro Collserola, en el Paseo de la Vall d'Hebron, 159-169, de Barcelona.

El CVI es un espacio real de valoración, configurado como un hogar digital accesible, donde un grupo humano interdisciplinario da servicio de valoración a personas dependientes utilizando

todas las herramientas que las nuevas tecnologías permiten con la intención de asesorarles en que aspecto la tecnología les puede ayudar en función de su grado de necesidad.

Es un servicio creado para resolver las dificultades que la autonomía personal afronta en situaciones concretas de dependencia o discapacidad.

Sus objetivos fundacionales son:

- Fomentar la mejora de la autonomía de las personas dependientes o discapacitadas mediante la utilización de productos de apoyo y tecnología de asistencia.
- Valorar individualmente, mediante equipos interdisciplinares, los productos de apoyo que permitan una mejor calidad de vida a las personas dependientes o discapacitadas.
- Asistir y colaborar con entidades y profesionales en los campos de la accesibilidad en las viviendas o en los espacios públicos por parte de personas dependientes o discapacitadas.
- Colaborar con entidades públicas y privadas en la investigación y la innovación productos de apoyo y aplicaciones tecnológicas destinadas a personas dependientes o discapacitadas.
- Formar al personal para que implante y haga el seguimiento de las ayudas técnicas que utilicen las personas dependientes o discapacitadas.

El CVI se compone de tres áreas: Área de vivienda, Área de I + D + I y Área personalizada:

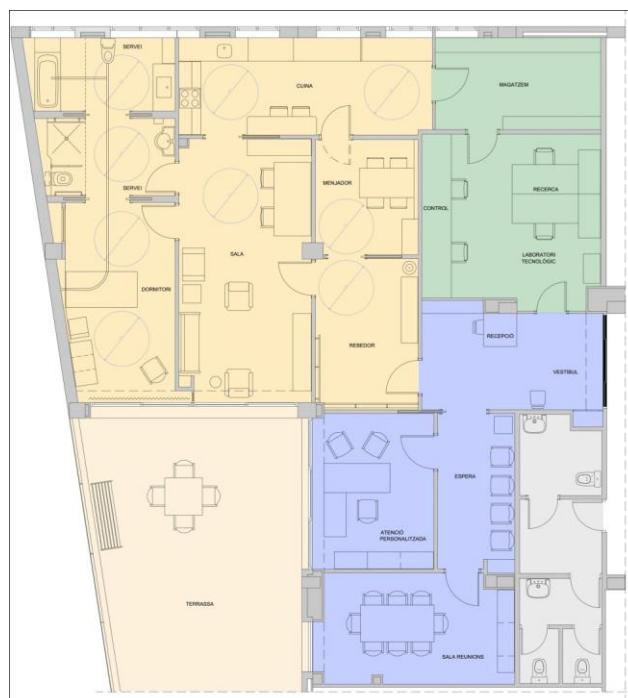


Figura 1.2: Áreas del centro

A continuación se explica con más detalle cada una de estas zonas.

1.1.1.1 Área de vivienda

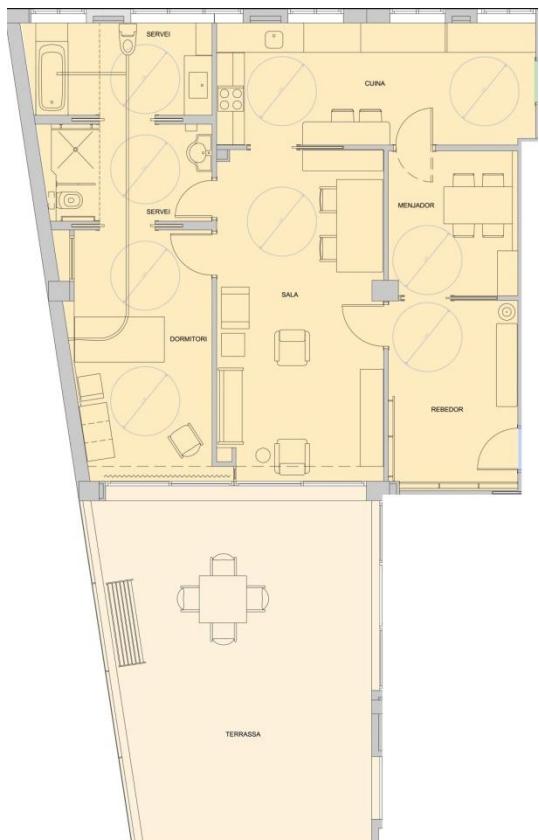


Figura 1.3: Área de vivienda

El CVI recrea, en un espacio real, las características y necesidades de una vivienda, con sus espacios y diferentes ambientes:

- Zonas de estar
- Zonas de descanso
- Zonas de higiene
- Zonas de servicio

Todas ellas equipadas con las últimas tecnologías y los avances técnicos más emergentes, para fomentar y potenciar al máximo la autonomía en el propio hogar.



Figura 1.4: Zonas del área de vivienda

1.1.1.2 ÁREA DE I + D + I

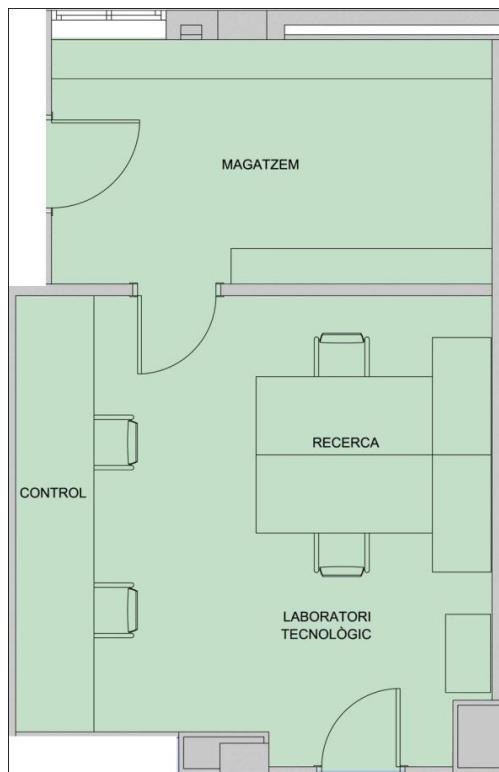


Figura 1.5: Área de I+D+I

En el CVI convive también un laboratorio destinado a la investigación, desarrollo e innovación (I+D+i). Es un espacio donde participa la Universidad Politécnica de Cataluña, que colabora en el desarrollo de proyectos tecnológicos pensados para personas dependientes.

Este laboratorio colabora con empresas privadas y usuarios diseñando nuevas ayudas, o mejorando las actuales, con el objetivo de dar respuestas a nuevas necesidades que puedan plantearse.

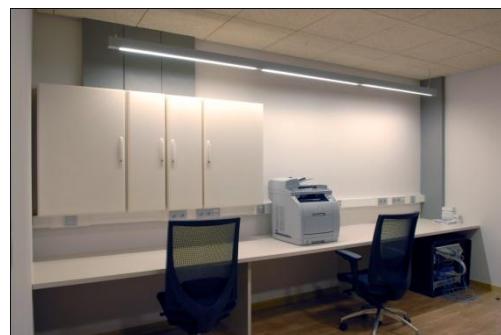


Figura 1.6: Zona de servidores

1.1.1.3 ÁREA PERSONALIZADA



Figura 1.8: Área personalizada

Es la zona reservada para que los distintos profesionales realicen las entrevistas personalizadas encaminadas al diagnóstico previo de las necesidades de la persona.



Figura 1.7: Atención personalizada



Figura 1.9: Sala de reuniones

1.1.2 Modelo de Investigación + Desarrollo

El CVI colabora con la UPC en la actividad de I + D + I para mejorar y aplicar las tecnologías activas en el entorno doméstico para que doten de autonomía a las personas con dependencia. El CVI ofrece sus instalaciones para valorar prototipos que están en desarrollo en la Universidad y traslada a ésta aquellas ideas y proyectos sobre los cuales es necesaria una investigación aplicada.

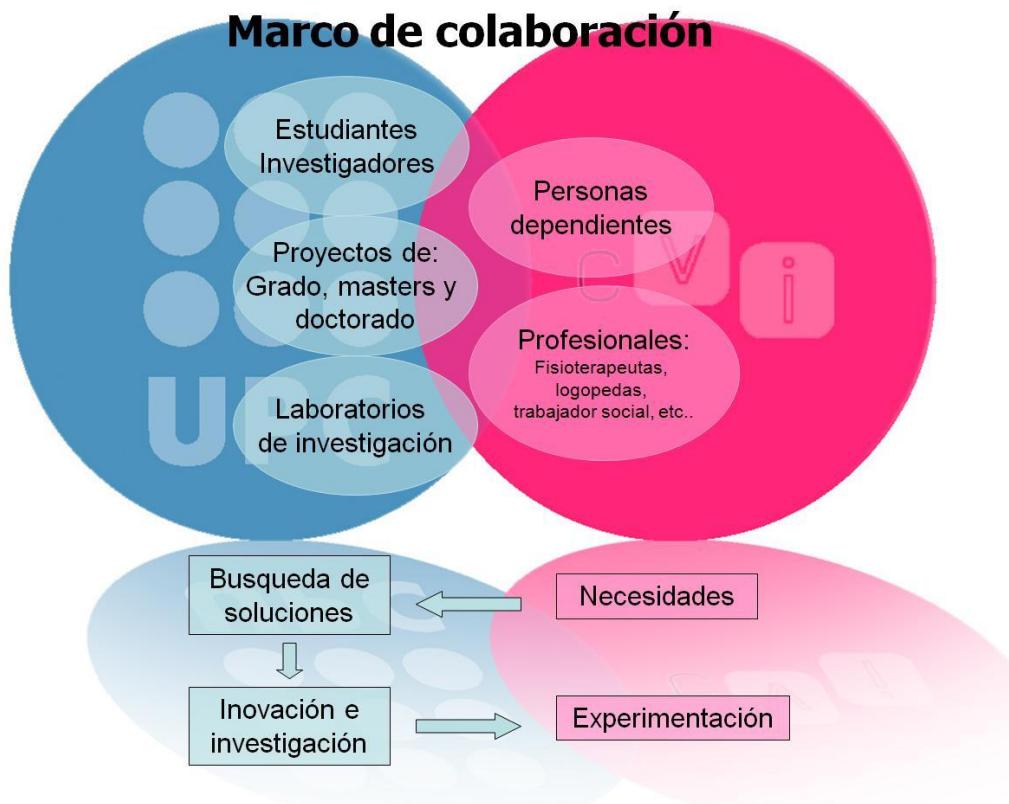


Figura 1.10: Marco de colaboración

Este marco promueve la puesta en común de la experiencia del investigador y de los profesionales expertos en la atención de personas con dificultades en su autonomía.

El CVI cuenta con la participación y soporte de los grupos de investigación Knowledge Engineering, Machine Learning Grup (KEMLG) y el Grupo de Robótica Inteligente i Sistemas (GRINS).

La colaboración se extiende, en especial, al ámbito de las tecnologías asistenciales para personas dependientes.

El CVI también ha establecido acuerdos con un conjunto de empresas con el objetivo de innovar en productos de mercado que asuman las necesidades de las personas dependientes.



El CVI ofrece a las empresas una plataforma donde éstas pueden validar sus productos con usuarios con dificultades de autonomía.

1.2 Objetivos

El objetivo primordial del proyecto final de carrera es la elaboración de una aplicación personalizable y orientada a personas con diferentes tipos de discapacidad, que permita controlar los diferentes elementos de la casa conectados a la red Konnex y, si es posible, añadir a este control aquellos elementos que no se pueden manejar actualmente a través de Konnex, tales como los dispositivos multimedia (televisión, reproductor de DVD, etc.).

En el CVI, como se explicará más adelante, posee controles a distancia muy sofisticados que permiten el control de la mayoría de elementos de la casa controlados por Konnex, incluyendo incluso, gracias a un extensor inalámbrico, algunos elementos multimedia. Aunque esto pueda parecer exactamente lo que nos están pidiendo no es así. Estos controles remotos son bastante complejos de utilizar y programar, sobre todo para personas con serias discapacidades que no pueden apretar un botón de mando convencional. Además el extensor que poseen funciona con radiofrecuencia y tiene un alcance reducido, sin mencionar que solo funciona con un mando a distancia de la misma marca.

Por eso ha sido necesaria la realización de este proyecto, para acercar más a las personas con cualquier tipo de discapacidad a tecnologías domóticas que pueden hacer su vida mucho más fácil.

En el capítulo 4, Requisitos de diseño, se explican con más detalle los objetivos a realizar, según los requisitos del proyecto, después de analizar las diferentes tecnologías presentes en el CVI y en el mundo de la domótica.

1.3 Motivación

La domótica es un tema en constante evolución y cada vez con una mayor presencia en los hogares de clase media. Por eso he encontrado este proyecto muy interesante, aún más teniendo en cuenta que desde mi infancia me han interesado estos temas. Un claro ejemplo de esto es el trabajo de investigación de Bachillerato que realicé sobre inteligencia artificial, robótica y sus posibles aplicaciones en el futuro, un futuro que es ahora mismo.

Durante la realización en la carrera de asignaturas como *Robótica* (ROB), *Visió per Computador* (VC) o *Sistemes Informàtics per a l'Automatització* (SIA), mis ganas de aprender sobre el tema han crecido, y cada vez veo más claro que quiero dedicarme profesionalmente a esta rama de la ingeniería. Clara está esta decisión cuando actualmente estoy cursando paralelamente a la realización de este proyecto el máster de Automática y Robótica que imparte el departamento de ESAII en la UPC.

Vistas las necesidades y los objetivos pedidos por la empresa, explicados anteriormente, ha quedado demostrado que éste es un PFC muy completo, con una parte importante de investigación y análisis de requisitos y una parte de diseño que culminaría en la implementación de un prototipo, además de abarcar áreas de conocimiento muy diversas. Todos estos motivos expuestos han sido los que me han hecho considerar este proyecto como una aportación importante al gran conocimiento en este campo que tanto me apasiona.

2 Estado del arte

En este capítulo explico cual es actualmente el estado de la domótica y las diferentes tecnologías que existen, poniendo especial énfasis en la tecnología Konnex, ya que gran parte del proyecto está orientado a la utilización de esta tecnología.

2.1 Domótica

La palabra domótica proviene de la unión de dos palabras: *domus* (casa en latín) y *robotica*, entonces el significado de domótica es: casa robotizada. Se entiende por lo tanto como domótica el conjunto de los sistemas capaces de automatizar y controlar una vivienda, aportando servicios como la gestión energética, seguridad, bienestar y comunicación.

Algunos de los elementos que se pueden controlar en una casa domótica son los siguientes: la iluminación, la climatización, puertas, ventanas, persianas y toldos, electrodomésticos o el suministro de agua, gas y electricidad. En general, puede llegar a controlarse todo aquello que se proponga, nadie ha puesto un límite aún a la domótica.

La domótica está cada vez más presente en los hogares actuales, pero ha sido en los últimos años que se ha dado un gran impulso en este sector, gracias al abaratamiento de los componentes y el hecho que los edificios de obra nueva empiezan a estar preparados para soportar esta tecnología. Esto último ha fomentado que la domótica ya no sea un producto sólo para ricos, cualquier persona de clase media puede optar a domotizar su casa por lo menos parcialmente.

Pero en este proyecto se ha estudiado, evaluado e implementado tecnología para un sector minoritario y específico de la población, el de las personas discapacitadas. El objetivo de la domótica para estas personas no es simplemente el de facilitar ciertas tareas, sino directamente el de permitirlas sin ayuda de nadie.

En el capítulo 3, Estudio de las tecnologías utilizadas en el CVI, se habla de las diferentes tecnologías que existen especialmente orientadas a personas discapacitadas y están disponibles actualmente en el CVI.

En un sistema domótico se pueden encontrar elementos de los siguientes tipos:

- **Controladores:** dispositivos encargados de gestionar toda la información sobre los otros elementos del sistema. También incluye las interfaces necesarias para interactuar con el usuario u otra aplicación.
- **Actuadores:** dispositivos capaces de recibir una orden del controlador y realizar la acción pertinente. Pueden ser el motor de una puerta o el interruptor de una luz, por ejemplo.
- **Sensores:** Su función es la de obtener información del entorno para enviársela al controlador para que este realice las acciones necesarias. Pueden ser sensores de presencia o temperatura, interruptores o un mando de control remoto.

Puede darse el caso que un mismo dispositivo sea un actuador y sensor e incluso controlador, lo convertiría en un elemento domótico autónomo del resto de dispositivos existentes en la vivienda.

2.1.1 Arquitecturas

Dependiendo de la distribución de los elementos domóticos y los controladores de estos, se pueden distinguir tres tipos de arquitecturas diferentes:

- **Arquitectura centralizada:** existe un único dispositivo controlador que recibe la información de múltiples sensores y una vez procesada la información se generan las órdenes oportunas y se envían a los actuadores.

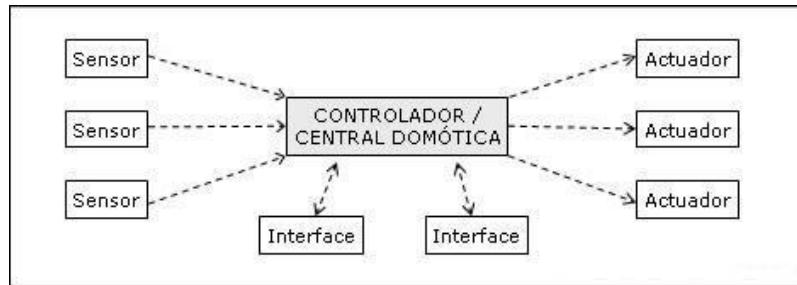


Figura 2.1: Arquitectura centralizada

- **Arquitectura distribuida:** no existe ningún controlador centralizado, cada elemento posee su propio controlador y es capaz de realizar las órdenes oportunas sobre los actuadores del sistema.

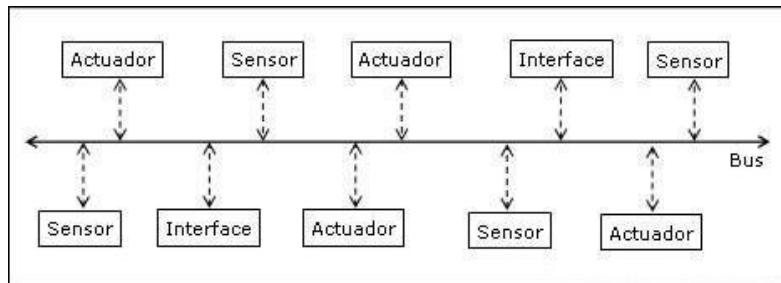


Figura 2.2: Arquitectura distribuida

- **Arquitectura mixta:** Sería un término medio entre las dos arquitecturas, existen sistemas que utilizan una arquitectura descentralizada pero que poseen módulos o islas de control centralizadas.

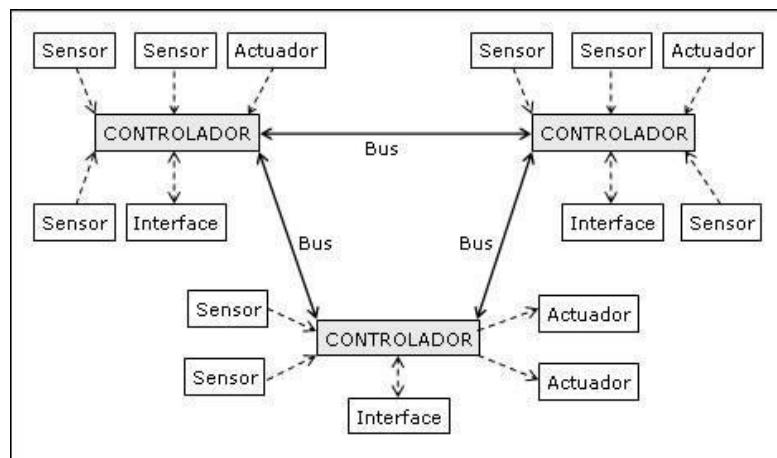


Figura 2.3: Arquitectura mixta

Las tres soluciones presentan sus ventajas y desventajas. En función de las necesidades que se quieran satisfacer puede ser mejor una u otra.

2.1.2 Sistemas de comunicación

Dependiendo del tipo de enlaces que se utilicen en el sistema domótico se pueden clasificar en dos categorías:

- **Cableados:** Sistemas en los que todos los sensores, actuadores y controladores están conectados entre sí con cables. Se pueden catalogar los sistemas cableados en función de si el cableado que se utiliza es exclusivo o compartido con otro sistema:
 - **Cableado exclusivo:** en los hogares normalmente no existe ningún otro cableado que no sea el de la tensión eléctrica a 220V o el telefónico, excepto en obras nuevas que pueden disponer de cableado para transmisión de música. Cuando hablo de cableado exclusivo me refiero a tener un sistema de cableado solo para el sistema domótico, sin compartirlo con ningún otro sistema. Esto implica cablear toda la vivienda para añadir un nuevo bus de comunicaciones que permita el envío de datos con los dispositivos domóticos. Esto resulta práctico cuando se aplican a viviendas de obra nueva mientras se construyen, ya que no supone un gran coste adicional añadir tres cableados en vez de dos.
 - **Cableado compartido:** Cuando no hay posibilidad de cablear la vivienda de nuevo se puede optar por esta solución, que consiste en utilizar un cableado ya existente y compartirlo. Concretamente se suele utilizar la línea eléctrica, la tecnología se llama *Power Line Carrier* (PLC), funciona mediante la modulación de una onda portadora cuya frecuencia oscila entre los 20 y 200 kHz inyectada directamente en el cableado eléctrico.
- **Inalámbricos:** Cuando no es posible cablear la vivienda y tampoco utilizar la tecnología PLC se puede optar por utilizar tecnologías inalámbricas como WiFi, Bluetooth, infrarrojos o radiofrecuencia. Estas tecnologías permiten que el dispositivo domótico no necesite estar en un lugar fijo, ya que puede comunicarse con el sistema desde cualquier lugar dentro del alcance del receptor.

2.1.3 Protocolos y estándares

2.1.3.1 Lonworks

Es una tecnología de control desarrollado por la compañía americana *Echelon Corp*. Puede utilizar una gran variedad de medios de transmisión. Aire, par trenzado coaxial, fibra o red eléctrica. Necesita la instalación de una serie de nodos a lo largo de la red, estos gestionan los diferentes sensores y actuadores. La configuración de estos nodos se tiene que realizar utilizando la herramienta que ofrece la propia empresa, *Lonmaker*.

Se trata de una tecnología muy robusta y fiable, especialmente indicada para la automatización industrial, que es el ámbito de donde proviene. Aunque su uso se ha extendido a la domótica.

2.1.3.2 ZigBee

Se trata de un protocolo de comunicaciones inalámbrico, similar a Bluetooth y basado en el estándar para redes inalámbricas de área personal *WPAN IEE_802.15.4*. Surgió de una alianza sin ánimo de lucro entre más de 200 empresas, con el objetivo de conseguir el desarrollo de una tecnología inalámbrica de bajo coste.

Se pensó especialmente en la utilización en domótica debido a su bajo consumo, su sistema de comunicación vía radio y su fácil integración, ya que se pueden integrar con poquísimas electrónicas.

Se le pueden conectar un máximo de 64000 nodos, consume 30mA transmitiendo, 3mA en reposo y transmite a una velocidad de 250kbps. Estas características hacen que este protocolo sea ideal para domótica, ya que ésta necesita enviar pequeños paquetes de datos. El hecho que ZigBee pueda situarse en modo “reposo” resulta muy útil ya que puede pasar mucho rato entre cada acción del usuario.

2.1.3.3 X10

X10 es un protocolo de comunicaciones para el control remoto de dispositivos eléctricos. Utiliza la línea eléctrica (220V o 110V) para transmitir señales de control entre equipos de automatización del hogar.

X10 fue desarrollada en 1975 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.

Las señales de control de X10 se basan en la transmisión de ráfagas de pulsos a 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).

Primero se transmite una orden con el Código de Casa y el Número de Módulo que direccionan el módulo en cuestión. Luego se transmite otra orden con el código de función a realizar. Hay 256 direcciones soportadas por el protocolo.

2.1.3.4 KNX

Es el estándar KNX es uno de los más utilizados actualmente para uso domótico, ya que se trata del único estándar abierto a nivel internacional para este tipo de aplicaciones. También es el utilizado en el CVI y, por lo tanto, el que me he visto obligado a estudiar de forma mucho más exhaustiva. A continuación describo de forma extensa esta tecnología.

2.2 Tecnología Konnex

Konnex (o KNX) es la principal tecnología usada en el CVI e involucra una gran parte de este proyecto, por lo tanto cabe hacer una destacada mención de ella antes de profundizar en esta memoria.

KNX es un estándar abierto para aplicaciones de control de la vivienda o cualquier tipo de edificio. A continuación paso a explicar cómo surgió este estándar, cual es el estado actual de esta tecnología en el mundo, qué ventajas tiene y otros detalles técnicos que deberían conocerse.

2.2.1 Historia del estándar

Las especificaciones anteriores a KNX aparecieron a principios de los 90 de la mano de Batibus, EIB y EHS. Estas tres importantes soluciones avanzadas para el control de viviendas y edificios en Europa intentaron primeramente desarrollar sus mercados separadamente, tratando de hacerse un lugar en la estandarización europea. Batibus lo hizo especialmente bien en Francia, Italia y España, mientras que EIB lo hizo en los países de lengua germana y norte de Europa. Por su parte, EHS fue la solución preferida para fabricantes de productos de gama blanca (Frigoríficos, lavadoras, microondas, lavavajillas, secadoras, etc.) y marrón (Informática e hifi).

En 1997 estos tres consorcios decidieron unirse con el fin de desarrollar conjuntamente el mercado del hogar inteligente, acordando estándares industriales comunes que también podrían ser propuestos como estándar internacional. La especificación KNX fue publicada en primavera de 2002 por la recién establecida KNX Association. La especificación está basada en la especificación de EIB completada con los mecanismos de configuración y medios físicos nuevos originalmente desarrollados por Batibus y EHS.

En diciembre de 2003 el protocolo KNX así como los dos medios de transmisión TP (par trenzado) y PL (línea eléctrica) fueron aprobados por los comités nacionales europeos y ratificados por el CENELEC Bureau Technique, como estándar europeo EN 50090. KNX en radiofrecuencia fue aprobado en mayo de 2006.

KNX, además de ofrecer especificaciones para la automatización de equipos de instalación eléctrica, ofrece soluciones para aplicaciones HVAC (Calefacción, Ventilación y Aire Acondicionado). Por este motivo la Asociación KNX también propuso sus especificaciones al CEN (Comité Europeo de Estandarización), para su publicación como estándar europeo de sistemas de control y automatización de edificios. CEN aceptó la propuesta y las especificaciones de KNX fueron publicadas por el CEN como Norma Europea EN 13321-1 y EN13321-2.

En vista del gran interés fuera de los países europeos por productos compatibles KNX y su sólida tecnología, KNX Association también dio los pasos necesarios que su estándar fuera aprobado a nivel internacional. De esta manera, a finales de 2004 los países activos en CENELEC TC 205 propusieron la norma europea para estandarización EN 50090, para su



Figura 2.4: Logo KNX

estandarización internacional, por la Organización ISO/IEC. En noviembre de 2006 el protocolo KNX ha sido aceptado, incluyendo todos los medios de transmisión (TP, PL, RF, IP) como ISO/IEC 14543-3-x para publicarse como Estándar Internacional. De esta manera KNX es el único estándar abierto de gestión técnica de viviendas y edificios a nivel mundial.

El gran interés en China acerca de los productos compatibles de KNX y de su tecnología fue el motivo por el cual la Asociación KNX hizo traducir al idioma chino la norma ISO/IEC 14543, la cual está reconocida como estándar internacional por el ISO/IEC. El comité de estandarización de China, SAC TC 124, presentó el estándar KNX al sistema de normalización en China y en Julio de 2007 fue adoptado como estándar chino GB/Z 20965.

También está estandarizado internacionalmente el acoplamiento de KNX a otros sistemas de automatización: tanto el estándar americano ANSI/ASHRAE 135, como el documentos de ISO 16484-5 para el “mapping” entre KNX y BACnet.

2.2.2 KNX en la actualidad

En la actualidad KNX es una de las tecnologías más utilizadas en el mundo y la que posee mayor apoyo internacional de organizaciones de todo tipo. A continuación se muestran algunas cifras (datos de septiembre de 2010) que corroboran lo dicho:

- 215 KNX miembros en 30 países.
- 21.030 KNX partners 106 países.
- 159 centros de formación en 31 países.
- 68 socios científicos en 17 países.
- 8 userclubs en 8 países.
- 6 miembros asociados.
- 22 grupos nacionales.
- 33.813 licencias ETS vendidas en 96 países.
- 6.553 grupos de productos certificados.

En la gráfica siguiente se puede apreciar el dominio de la tecnología KNX frente a otros protocolos y tecnologías diferentes:

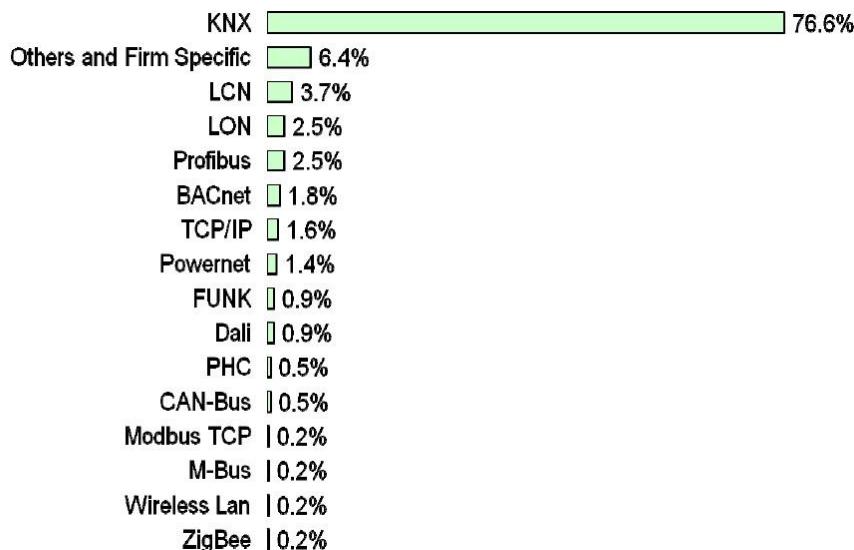


Figura 2.5: Uso de las diferentes tecnologías domóticas en el mundo

Si hacemos un zoom sobre cuatro de las tecnologías más usadas en el 2005 y vemos su progreso hasta la actualidad se puede apreciar cómo la presencia de KNX aumenta hasta situarse en el 76.6% del mercado mundial. Destaca considerablemente el hecho que el uso del protocolo TCP/IP para fines domóticos ha disminuido drásticamente hasta situarse en un triste 1.6 del mercado.

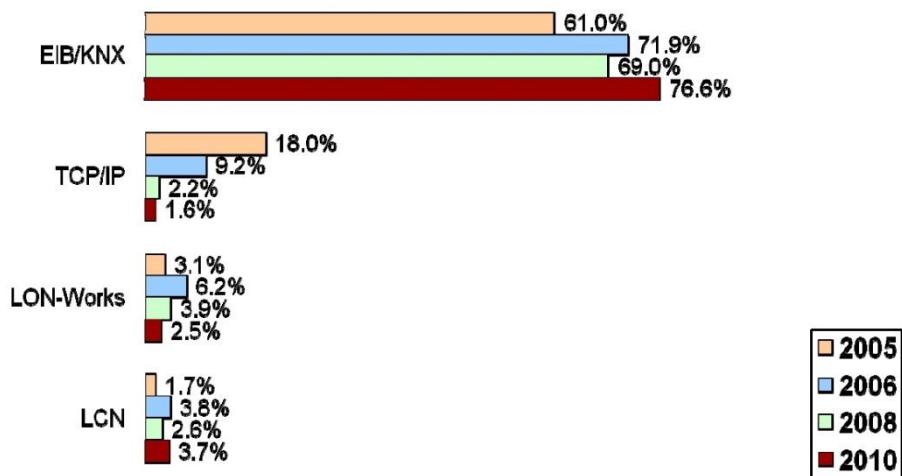


Figura 2.6: Progreso del uso de diferentes protocolos en los últimos cuatro años

2.2.3 Principales ventajas

A continuación se nombran las principales ventajas que ofrece la tecnología KNX frente a otros sistemas anteriormente nombrados:

- Estándar Internacional que garantiza su continuidad en el futuro. Es el único estándar abierto para el control de casas y edificios.
- Gracias a la certificación de producto, KNX garantiza Interoperabilidad de productos. El proceso de certificación KNX asegura que funcionarán y se comunicarán diferentes productos de diferentes fabricantes usados en diferentes aplicaciones. Esto asegura un alto grado de flexibilidad en la extensión y modificación de las instalaciones. Laboratorios neutrales (terceras compañías) son quienes analizan la conformidad del producto. KNX lleva a cabo un plan de certificación para productos, centros de formación (instituciones profesionales y privadas) e incluso personas.
- KNX representa alta calidad de producto: La KNX Association exige un alto nivel de producción y control de calidad durante todas las etapas de la vida del producto. Por lo que todos los miembros fabricantes tienen que mostrar conformidad a la norma ISO 9001 incluso antes de que soliciten la certificación para productos KNX. Además de la conformidad del fabricante a la norma ISO 9001, los productos tienen que cumplir con

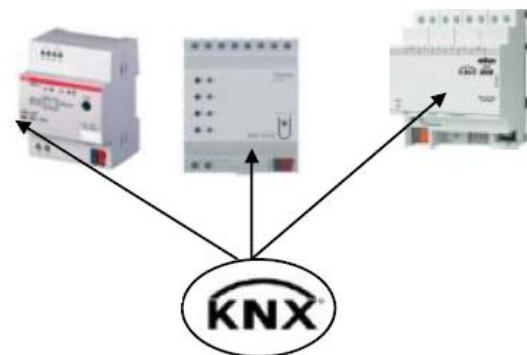


Figura 2.7: Interoperabilidad de productos

los estándares tanto Europeos como Internacionales para el control de Casas y Edificios. En caso de duda, la KNX Association tiene el derecho de volver a analizar el producto o puede exigir al fabricante el informe de conformidad de dicho hardware.

- Un único software independiente del fabricante ETS® (Engineering Tool Software). La herramienta software ETS permite proyectar, diseñar y configurar todos los productos certificados KXN. Dicha herramienta es además independiente del fabricante: quien integre el sistema podrá combinar los productos de varios fabricantes en una instalación.
- KNX puede ser usado para todas las aplicaciones en el control de casas y edificios desde iluminación, contraventanas, control de seguridad y alarmas, calefacción, ventilación, aire acondicionado, control de agua y dirección de energía, medición, hasta aplicaciones para el hogar, audio y mucho más.



Figura 2.8: Todas las funciones y aplicaciones

- KNX mejora el confort y la seguridad con sus instalaciones así como contribuye al ahorro energético y la protección del clima (se puede conseguir hasta un 50% de ahorro en iluminación y calefacción).
- KNX se adapta a diferentes tipos de construcciones. Puede ser usado tanto en nuevas construcciones como en las ya existentes. Por lo que las instalaciones KNX pueden ser fácilmente extendidas y adaptadas a las nuevas necesidades con una pequeña inversión de tiempo y dinero. También puede ser instalado tanto en pequeñas casas como en grandes edificios (oficinas, hoteles, palacios de congresos, hospitales, escuelas, grandes almacenes, aeropuertos, etc.).



Figura 2.9: KNX se adapta a todo tipo de construcciones

- Soporta diferentes modos de configuración. Estos son explicados en el siguiente punto.
- Soporta diferentes medios de comunicación, como es explicado más adelante con detalle.
- Puede ser acoplado a otros sistemas. Distintos fabricantes ofrecen pasarelas a otros sistemas, es decir a otros sistemas de automatización de edificios, redes de telefonía, redes multimedia, redes IP, etc.

- KNX es independiente de cualquier plataforma hardware o software. Puede ser llevada a cabo bajo cualquier plataforma de microcontrolador. Puede ser implementada desde el principio, pero para una entrada más sencilla en el mercado, los fabricantes de KNX también pueden recurrir a los proveedores de componentes KNX. Para los miembros de KNX, esto es completamente gratis.

2.2.4 Modos de configuración

El estándar KNX permite a cada fabricante seleccionar su modo de configuración idóneo, que le permita elegir su combinación ideal según el segmento del mercado que desee orientarse. El estándar KNX permite 3 modos de configuración:

- S-Mode (System Mode):** Este mecanismo de configuración está dirigido a los instaladores KNX formados que llevan a cabo funciones de control sofisticadas en sus instalaciones. Una instalación realizada con componentes S-Mode puede ser configurada a través de la herramienta de software ETS® 3 Professional. S-Mode ofrece el mayor grado de flexibilidad para la configuración de las funciones de control de edificios.
- E-Mode (Easy Mode):** Este mecanismo de configuración está dirigido para instaladores con conocimientos básicos sobre KNX. Los productos compatibles con E-Mode ofrecen funciones más limitadas si lo comparamos con el modo S-Mode. Los componentes E-Mode ya vienen pre programados y cargados con parámetros por defecto. Con un simple configurador, cada componente puede ser parcialmente reconfigurado (principalmente sus parámetros de configuración y los enlaces de comunicación).
- A-Mode (Automatic Mode):** Con este mecanismo de configuración los dispositivos se configuran ellos mismos automáticamente. Este modo está destinado al usuario final, para que éste sea capaz de instalarlos él mismo.

En la gráfica Figura 2.10 se puede apreciar mejor la funcionalidad y la sofisticación que aporta cada modo, excepto el A-Mode, que puede adoptar cualquier tipo de configuración según el fabricante.

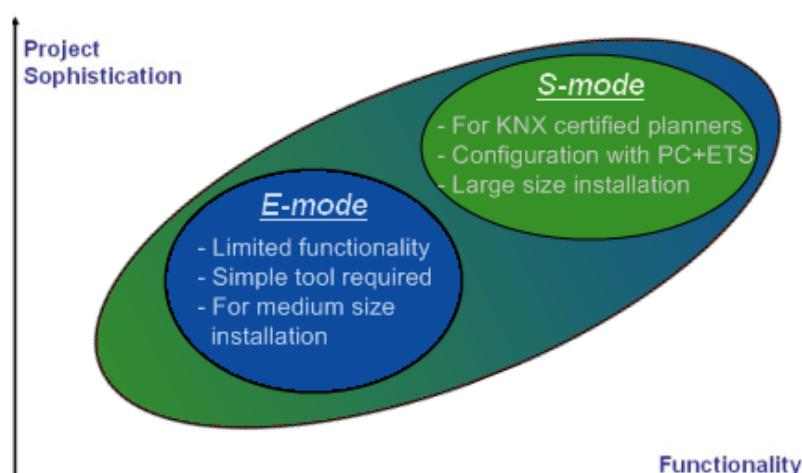


Figura 2.10: Funcionalidad / sofisticación de los modos de

2.2.5 Medios de transmisión

Además de los 3 modos de configuración, el estándar KNX incluye distintos medios de transmisión. Cada uno de éstos puede ser usado con uno o más modos de configuración, de tal forma que posibilita a cada fabricante elegir la combinación adecuada para el segmento del mercado que desea.

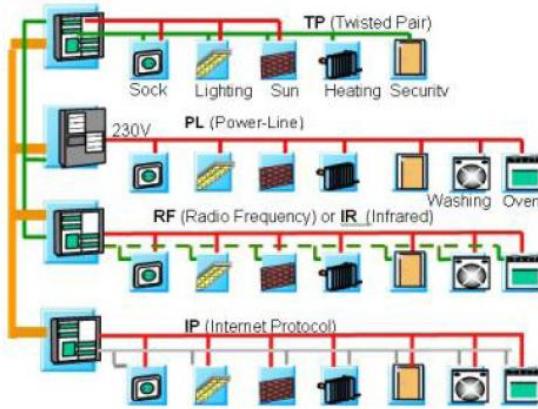


Figura 2.11: Medios de transmisión

- **TP (Twisted Pair):** Este medio de comunicación basado en un cable par trenzado tiene una velocidad de transmisión de 9600 bits/s, y ha sido tomado de EIB. Todos los productos certificados EIB TP1 y KNX TP1 operan en el mismo bus.
- **PL (Power line):** Este medio de transmisión basado en *Power Line Carrier* (PLC), de velocidad de transmisión 1200 bits/s, también ha sido tomado de EIB. Los productos certificados EIB y KNX PL110 operan y se comunican los unos con los otros bajo la misma red de distribución eléctrica.
- **RF (Radio frequency):** Los dispositivos KNX admiten este medio de transmisión que emplea señales de radio para transmitir telegramas KNX. Dichos telegramas son transmitidos en la banda de frecuencia 868 MHz (dispositivos de corto alcance), con una fuerza máxima irradiada de 25 mW y velocidad de transmisión de 16.384 kBit/sec. El medio de transmisión KNX RF puede ser fabricado con componentes que se encuentran disponibles. Otras características es que permite implementaciones tanto unidireccionales como bidireccionales. Se caracteriza por su bajo nivel de consumo energético y está destinado a pequeñas y medianas instalaciones que sólo necesitan transmisores en casos excepcionales.
- **IP (Ethernet):** la especificación KNXnet/IP permite que los telegramas KNX puedan ser también encapsulados en telegramas IP. De esta forma podemos enviar dichos telegramas KNX por redes LAN e Internet.

3 Estudio de las tecnologías utilizadas en el CVI

Durante los primeros días de mi estancia en el CVI me dediqué a estudiar los diferentes dispositivos instalados en la casa domótica y las diferentes tecnologías que se utilizaban. El objetivo principal fue determinar con qué dispositivos o familia de dispositivos se podía interactuar remotamente.

En este capítulo muestro el resultado de dicho estudio, centrándome principalmente en las tecnologías utilizadas, organizando los dispositivos por islas de control. Entendiendo por isla de control todos aquellos dispositivos que se controlan desde un mismo sistema o con una misma tecnología.

El estudio ha consistido en la identificación de los diferentes dispositivos que se pueden encontrar en la casa domótica del CVI y las tecnologías que éstos utilizan para comunicarse e interactuar con el usuario o con un servidor, si se da el caso.

En función de las tecnologías utilizadas por éstos dispositivos podemos agruparlos en lo que se llaman islas de control. En el siguiente esquema muestro gráficamente que dispositivos forman parte de las diferentes islas de control estudiadas y la interacción entre ellas.

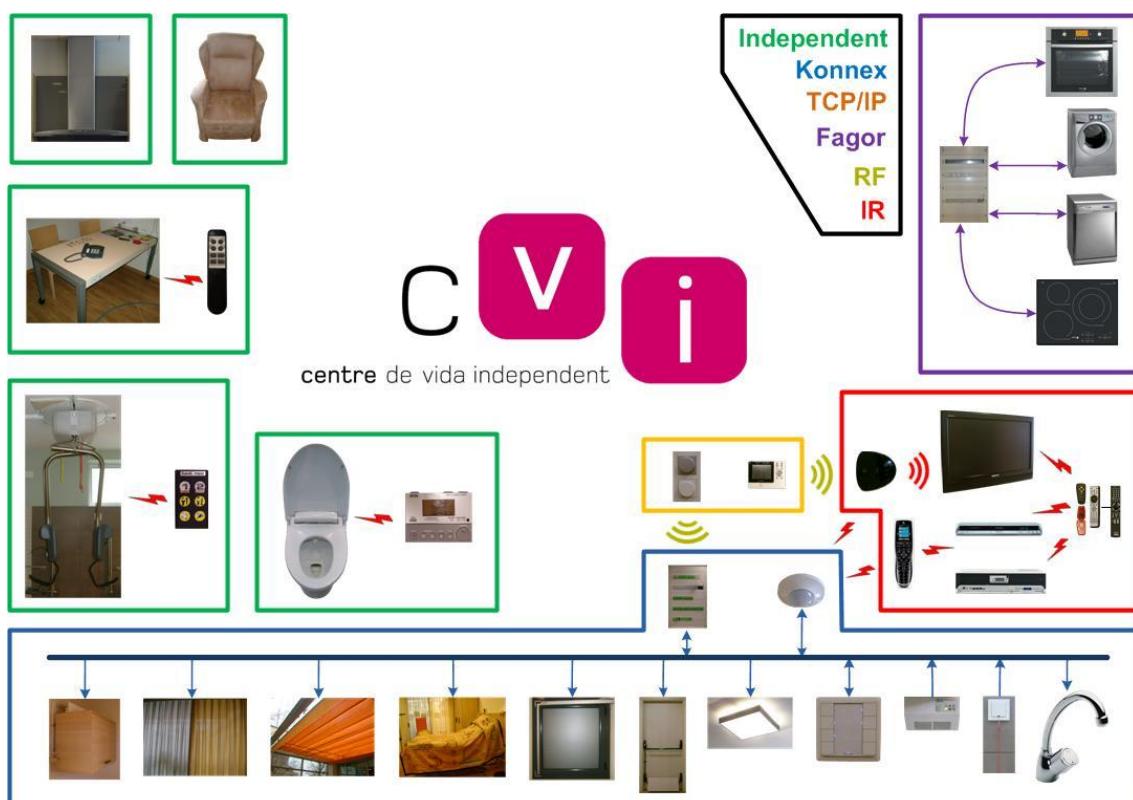


Figura 3.1: Islas de control presentes en el CVI

A continuación explico con detalle cada isla de control y los dispositivos que engloban.

3.1 Konnex

Esta es la isla de control principal de la casa, formando parte la mayoría de dispositivos domóticos del CVI. Los dispositivos se comunican con un servidor central a través de un bus de datos.

Podríamos distinguir dos grupos de dispositivos dentro de esta isla de control, los destinados a interactuar directamente con el usuario y los que únicamente esperan órdenes del servidor.

Dentro del primer grupo entrarían únicamente los que envían su nuevo estado al servidor para que éste realice los cambios oportunos. Algunos ejemplos serían los siguientes:

- **Detector de presencia / Extensor IR:** Este aparato tiene dos funcionalidades muy diferentes: A través de LEDs infrarrojos es capaz de detectar movimiento cuando alguien atraviesa el haz infrarrojo. Se utiliza, por ejemplo, para encender las luces de una sala cuando alguien entra. Por otro lado también puede utilizarse como receptor de señales de infrarrojos extendiendo la funcionalidad de algunos mandos de control remoto permitiendo controlar algunos elementos de la red KNX.
Figura 3.2: Detector de presencia
- **Botones de alarma:** Están situados en lugares donde la persona discapacitada puede necesitar ayuda, como por ejemplo los baños. Tienen una cuerda roja para que tirando de ella se active el botón.
Figura 3.3: Botón de alarma
- **Botoneras para las luces:** En cada sala de la casa se puede encontrar una de estas botoneras. Con ellas se pueden encender manualmente cada una de las luces de la sala e incluso programar lo que se denomina escena, que consiste en programar una serie de luces para que se enciendan de una sola vez.
Figura 3.4: Botonera para las luces
- **Detector de humo:** Está situado en la cocina y envía una señal al servidor KNX cuando detecta humo. Cuando eso ocurre suena la alarma y el servidor KNX avisa al exterior del riesgo de incendio.
Figura 3.5: Detector de humo

Dentro del segundo grupo entrarían todos los demás dispositivos, tales como armarios, puertas, ventanas, cortinas, etc. Estos únicamente reciben órdenes del servidor y actúan de alguna forma. En las figuras siguientes se pueden ver algunos de estos dispositivos.



Figura 3.8: Ventana



Figura 3.8: Puerta



Figura 3.8: Toldo

Tanto las ventanas como las puertas y el toldo están robotizados para que la persona discapacitada no necesite hacer esfuerzo alguno para moverlos. Lo mismo ocurre con la cama, ésta posee motores para adaptar la altura, la inclinación de la cabeza y de los pies. Se controla desde un mando anexo a la cama, pero puede controlarse también desde la red KNX.



Figura 3.9: Cama

3.2 Infrarrojos

La segunda tecnología que más presencia tiene en el CVI es la Infrarroja, o mejor dicho el control de dispositivos a través de señales de infrarrojos. Los dispositivos que encontramos en el CVI que usan esta tecnología son los llamados dispositivos multimedia (Figura 3.10), como el televisor, el reproductor de DVD o el media center.



Figura 3.10: Dispositivos multimedia

Éstos se pueden controlar a través de diversos mandos de infrarrojos convencionales o a través de un extensor inalámbrico, tal como se explica en el siguiente punto.

En el CVI se utiliza una gran diversidad de mandos infrarrojos, ya que dependiendo del tipo de discapacidad que tenga el paciente le puede ser más cómodo uno u otro. En la Figura 3.11 aparecen algunos de los mandos utilizados. Se puede apreciar como algunos son muy simples y con botones bien diferenciados y otros mucho más complejos (uno de ellos incluso posee una pantalla táctil configurable).



Figura 3.11: Mandos Infrarrojos

Como se menciona en el punto 0, algunos mandos son capaces de comunicarse por infrarrojos con el detector de presencia / Extensor IR. Este es el caso del mando situado a la derecha de la Figura 3.11, que a través de la pantalla táctil es capaz de controlar las luces y las persianas, entre otros elementos.

Cabe destacar que existen algunos mandos que además son capaces de interactuar con otros dispositivos sin utilizar infrarrojos. El funcionamiento de estos se explica en el siguiente apartado.

3.3 Radiofrecuencia

Otra tecnología inalámbrica presente en el CVI para la comunicación entre dispositivos es la radiofrecuencia (siglas RF). Esta tecnología se basa en la emisión de ondas de radiofrecuencia para el envío de información, tiene un alcance limitado de entre 20 y 100 metros dependiendo de la potencia del emisor y de los obstáculos entre él y el receptor.

Esta tecnología se utiliza cuando, por motivos técnicos, no se puede pasar ningún cable directo entre dos dispositivos y la utilización de infrarrojos es imposible. También se puede usar para conseguir libertad de movimiento para ese dispositivo dentro de la vivienda.

En el CVI se utiliza esta tecnología por ambos motivos. En el primer caso, cuando es imposible utilizar un cable o tecnología infrarroja, podemos encontrar interruptores como los de la **¡Error! No se encuentra el origen de la referencia.** Estos pueden desengancharse de la pared y transportarlos a otra habitación fácilmente. Las casas no suelen ser construidas pensando en personas discapacitadas, además cada discapacidad es diferente y debe ser tratada de forma única, por ese motivo en ocasiones no resulta fácil añadir dos interruptores más a una habitación allá donde se necesitan. Con este dispositivo, que se comunica con el servidor KNX vía RF se puede controlar cualquier otro dispositivo conectado a la red KNX.



Figura 3.12:
Botones RF



Figura 3.13: Mando universal Harmony® 1100 Advanced de Logitech

Como se comentó en el apartado anterior, existen mandos de control a distancia que pueden controlar también dispositivos conectados a la red KNX. Estos utilizan la tecnología RF para comunicarse con el servidor KNX a través de lo que llamamos extensor inalámbrico. Es el caso del mando Logitech de la Figura 3.13: Mando universal Harmony® 1100 Advanced de Logitech, que gracias al extensor inalámbrico que se muestra junto a él permite controlar una gran variedad de dispositivos de la red KNX, sin perder la funcionalidad de un mando de infrarrojos convencional.

3.4 Domótica Fagor

Fagor ha desarrollado una tecnología para redes domóticas muy potente y con una gran variedad de productos compatibles, su tecnología se llama *@Net Compatible*. Esta tecnología se basa en el protocolo BDF (Bus Domótico Fagor), que transmite información utilizando ondas portadoras sobre la red eléctrica, lo que permite una fácil instalación sin necesidad de cablear la vivienda.

3.4.1 Servidor Fagor

El Maior-Domo® Fagor es el centro del sistema. Es el encargado de hacer llegar las órdenes a los diferentes sistemas eléctricos como electrodomésticos, sistemas de seguridad y automatismos que se deseen controlar. En la Figura 3.14: Servidor Fagor puede verse el aspecto que tiene el armario donde se aloja el servidor de Fagor. Los diferentes módulos que pueden verse se explican más abajo.



Figura 3.14:
Servidor Fagor

El servidor tiene un límite de nodos por instalación:

- 8 Detectores de agua.
- 8 Detectores de gas.
- 3 Controles de electroválvula de agua.
- 3 Controles de electroválvula de gas.
- 8 Automatismos para el control de motores con doble sentido de giro.
- 8 Automatismos para el control de cargas o relés libres de tensión.
- 6 electrodomésticos NET Compatible.

El servidor se divide en una serie de módulos, cada uno con funciones muy específicas:

- **Módulo de control:** el módulo MC-400 posee un microprocesador y es la unidad central del sistema. El procesador analiza las señales que le llegan desde los diversos nodos que forman la red y ordena las acciones oportunas a los nodos cuya actuación sea requerida. Sus funciones son:



Figura 3.15:
Módulo MC-400

- Gestión de los sistemas de seguridad domóticos de protección frente a inundaciones y fugas de gas.
- Simulación de presencia automática.
- Detección de fallos en el suministro eléctrico.
- Activación y desactivación de funciones por comunicación telefónica.
- Programación horaria de funciones por comunicación telefónica.
- Consulta de estado.
- Gestión de tarifas eléctricas.
- Funcionamiento por prioridades de los electrodomésticos.
- Telediagnosis: gestión de alarmas y avisos de avería en los electrodomésticos.

- **Módulo telefónico:** El Módulo de Interface Telefónico MIT-400 incorpora la interfaz de usuario y la conexión a la línea telefónica RTC para el control telefónico del sistema

tanto en local como en remoto. Se utiliza para configurar los diversos parámetros personalizables en el módulo de control. También permite transmitir a dos números de teléfono las alarmas activadas por el módulo de control, así como para activaciones, programaciones o consultas de estado de los automatismos y electrodomésticos a través del teléfono. Las funciones del MIT-400 están protegidas por un código de acceso, de tal modo que sólo las personas autorizadas pueden controlar el módulo de control.



Figura 3.16:
Módulo MIT-400

- **Módulo batería:** El módulo de baterías MB-300 se usa para mantener el funcionamiento del módulo de control durante caídas de tensión o cortes de corriente. En el caso de producirse cortes de corriente de más de dos horas de duración continua proporciona una señal al módulo de control que le permite iniciar la gestión de la alarma telefónica por ausencia de suministro eléctrico.
- **Módulo Maior Vocce:** Este módulo es encargado de gestionar las órdenes que se envían a través de la interfaz de voz con el mismo nombre explicado en el apartado de interfaces.
- **Filtro de red domótica:** este módulo bloquea las tramas domóticas dentro de los límites de la instalación. El Filtro de la red domótica es un elemento que impide que las señales domóticas que circulan por la red eléctrica salgan de una vivienda y puedan afectar a los sistemas domóticos de viviendas cercanas.



Figura 3.17:
Módulo MB-300



Figura 3.18: Filtro
de red Domótica

3.4.2 Electrodomésticos

Fagor ofrece una amplia variedad de electrodomésticos, todos ellos compatibles con la tecnología domótica *@Net Compatible*. En el CVI se dispone de cinco de estos electrodomésticos: un frigorífico, un horno, una lavadora, un lavavajillas y una vitrocerámica (Figura 3.19). Cada uno, a través de las interfaces de control explicadas a continuación, permite configurar una gran variedad de parámetros de forma remota. De esta forma podemos, por ejemplo, calentar el horno antes de llegar a casa o comprobar que lo hemos apagado.

Para cierto tipo de personas discapacitadas esto puede resultar muy útil, ya que para algunas acciones no se necesita para nada la presencia de una persona, pueden realizarse remotamente desde cualquier punto del hogar o incluso del mundo.



Figura 3.19: Electrodomésticos Fagor

3.4.3 Interfaces de control

Fagor pone a disposición de sus clientes varias formas de interactuar con sus electrodomésticos domóticos. En el CVI se utilizan dos interfaces para ese fin:

- **Interfaz de voz Maior Vocce:** Maior Vocce permite al usuario el control de la Red Domótica mediante comandos hablados. Se Coloca como si fuese una pulsera, de esta forma se puede llevar con él donde se quiera. Puede ser compartido por varios usuarios sin necesidad de entrenamiento previo ni ningún tipo de programación. Es un complemento para Mayor-Domo y debe ser instalado conjuntamente con él. Se comunica a través de un módulo carril DIN situado junto al servidor de Fagor vía Radiofrecuencia a 868 MHz.
- **Teléfono:** Desde cualquier teléfono, móvil o fijo, y desde cualquier punto del planeta se puede acceder a la vivienda y controlar diversos parámetros de su instalación, navegando por sus opciones a través del teclado del teléfono. También se pueden recibir avisos de cualquier incidencia o problema que ocurra. Se trata de otra forma de acceder a los servicios del módulo Maior-Domo.
- **Internet:** la domótica de Fagor también puede programarse por la red a través del módulo de Maior-Domo. Esto permite un control total de todos los elementos Fagor instalados en el hogar desde cualquier parte del mundo de forma fácil y segura.



Figura 3.20: Maior Vocce

Fagor dispone de otra interfaz que consiste en una pantalla táctil, pero no está presente en el CVI, así que no he considerado necesario explicarla.

3.5 Dispositivos independientes

El CVI posee una gran variedad de dispositivos domóticos independientes, es decir, que no están unidos a ningún sistema como KNX o Fagor. Estos son perfectamente capaces de cumplir su función por sí mismos. A continuación procedo a explicar algunos de los elementos más destacados que se pueden encontrar:

- **Grúa:** Este dispositivo es muy útil para que personas inválidas puedan moverse por la casa sin ninguna ayuda. Consiste en una grúa capaz de desplazarse por la vivienda a través de unos carriles situados en el techo, se puede desplazar verticalmente para ayudar a superar obstáculos y para que sea más fácil subirse a ella. Se controla a través de su propio mando de control.

En el CVI ha sido instalada para trasladar a una persona de la cama al retrete o la ducha y viceversa.



Figura 3.21: Grúa

- **Sofá:** El CVI posee un sofá que, aparentemente, parece un sofá normal como el que podemos tener en nuestra casa. Es capaz, a través de un mando de control, inclinar la espalda y los pies. Hasta aquí nada que no hayamos visto hasta ahora, pero también es capaz de inclinar el asiento lo suficiente para dejarte casi de pie en el suelo. Esto resulta muy útil para personas mayores que les cuesta mucho levantarse por sí mismos.
- **Retrete:** Este elemento aparentemente no parezca que tenga nada de especial, pero es capaz de darte auténticos masajes. De entrada, cuando te acercas la tapa se sube automáticamente y la taza se calienta para no notar el frío tacto de la cerámica. Luego, una vez se ha terminado, a través de un mando de control incorporado, dispara un chorro de agua caliente y seguido un ventilador nos secará con aire caliente. Al levantarnos el retrete detecta que la persona ya ha finalizado y tira automáticamente de la cadena y baja la tapa. Para personas inválidas, que no son capaces de hacer algo tan simple como limpiarse o levantar la tapa resulta muy útil y realmente les hace la vida diaria más fácil.



Figura 3.22:



Figura 3.23: Retrete

- **Armario de altura ajustable:** El siguiente elemento es muy práctico para personas que no puede acceder a posiciones muy elevadas, como es el caso de personas con silla de ruedas. Este armario, a través de un control remoto, es capaz de bajar literalmente de la pared a través de un mecanismo motorizado situado detrás de él.



Figura 3.24: Armario de altura ajustable

4 Requisitos de diseño

El CVI ha dejado muy clara la necesidad que tienen para controlar de forma accesible el máximo número de elementos de la casa. Esto se ha traducido en los siguientes requisitos de diseño:

- Tiene que permitir controlar la mayor parte de aplicaciones posible de la casa.
- Tiene que tener una interfaz gráfica fácil de usar por personas discapacitadas.
- Tiene que tener una interfaz gráfica adaptable según la deficiencia y necesidades del paciente.
- Debería ser posible controlar la aplicación con comandos de voz, quizás con el propio sistema de reconocimiento que incorpora Windows 7.

Para satisfacer estos requisitos ha sido necesario elaborar un dispositivo hardware para el control de aparatos multimedia desde la misma aplicación y así aumentar el abanico de dispositivos que se puedan controlar. La necesidad de este dispositivo ha añadido, por parte de la UPC, nuevos requisitos técnicos que debería cumplir, estos son:

- Debe poder enviar por infrarrojos las órdenes a cada dispositivo directamente al receptor del aparato.
- Debe poderse controlar de forma inalámbrica vía IP.
- Debe de tener cuatro salidas de infrarrojos como mínimo.
- Dada la existencia de un extensor Logitech, que cumple una función parecida pero únicamente con sus mandos a distancia y que también posee cuatro salidas de infrarrojos que van directamente al receptor, es necesario que nuestro dispositivo pueda multiplexar las señales del extensor Logitech y las de nuestro dispositivo para que salgan por la misma salida.
- Debe de indicar, a través de LEDs por ejemplo, información del estado del dispositivo en cada momento.

Cumplidos estos requisitos, un esquema sencillo de lo que podría ser el dispositivo requerido sería el de la siguiente figura:

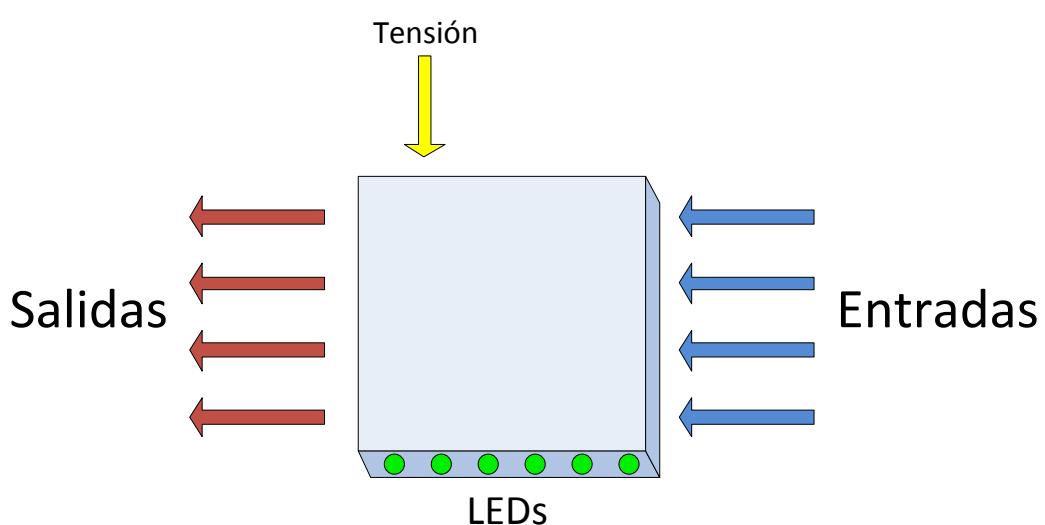


Figura 4.1: Esquema básico del dispositivo según los requisitos

Para dar una visión más global, la función del extensor Logitech que existe en el CVI actualmente queda perfectamente reflejado en el siguiente esquema:

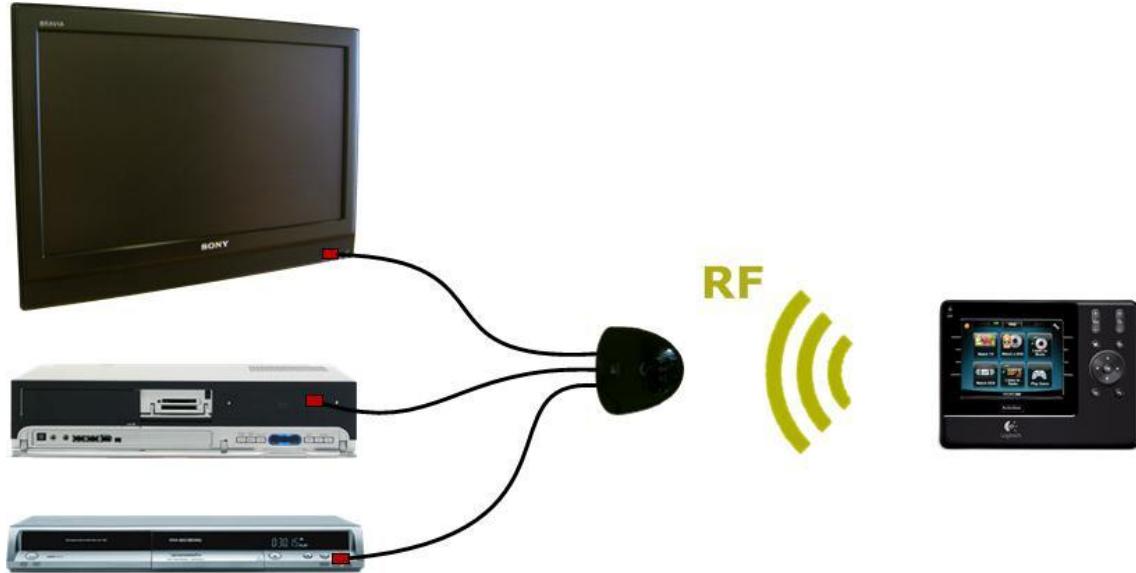


Figura 4.2: Funcionamiento del Extensor Logitech

En el esquema siguiente se representan los mismos dispositivos que en la figura anterior más los que participarían en el funcionamiento del extensor a realizar. Se puede apreciar pues como desde un móvil convencional se puede mantener una comunicación con el extensor mediante el enrutador WiFi presente en cualquier casa, y como este envía las órdenes a los dispositivos multimedia a través de un cable con un LED infrarrojo en la punta directamente situado en el receptor infrarrojo:

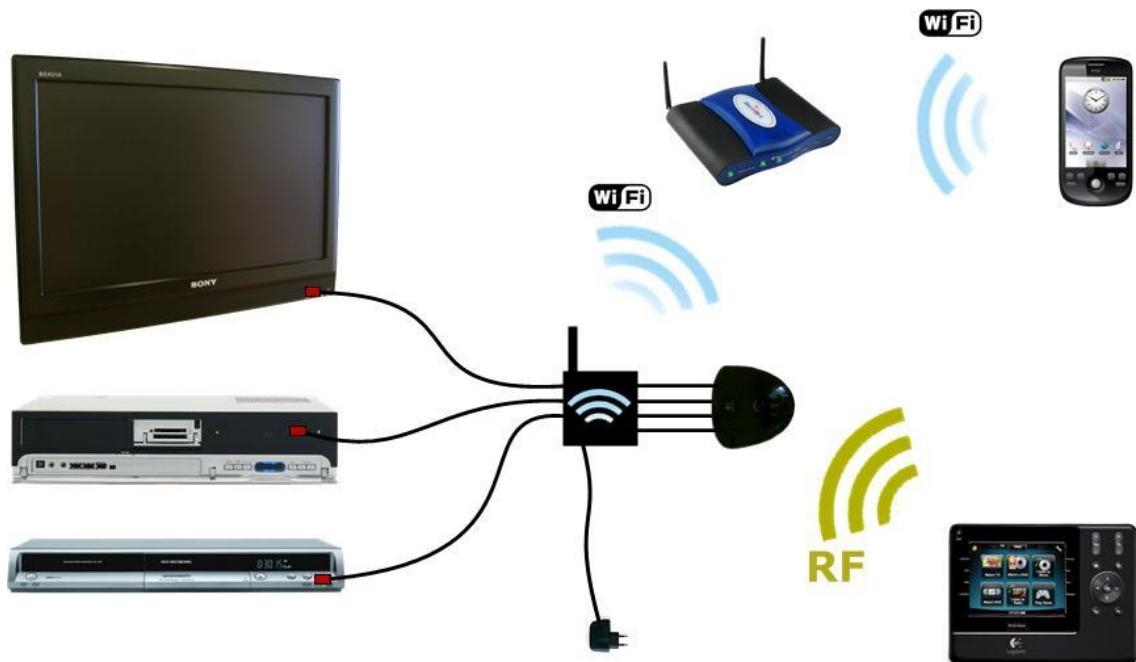


Figura 4.3: Esquema ilustrativo del funcionamiento del Extensor inalámbrico

De esta forma, el esquema de las islas de control presentes en el CVI mostrado en el capítulo anterior quedaría de la siguiente manera:

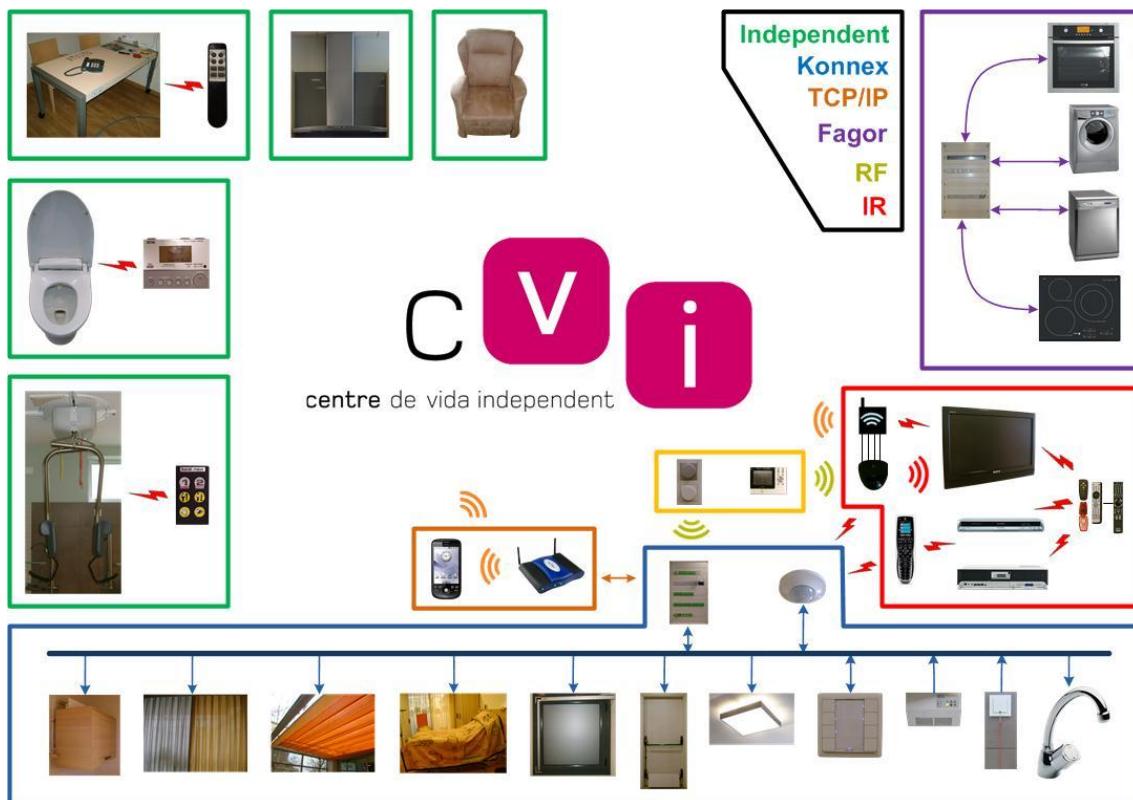


Figura 4.4: Islas de control en el CVI una vez realizado el proyecto

Estos requisitos son muchos para un solo proyecto final de carrera, además que engloba áreas de conocimiento muy diversas. Por ese motivo el siguiente proyecto se ha centrado en la elaboración de una API para el control de los elementos de la casa, así como el dispositivo para el control de los elementos multimedia. También he elaborado – a modo de ejemplo un par de aplicaciones bastante completas. Pero la interfaz que cumple los requisitos antes señalados ha sido, o está siendo, desarrollada por el departamento LSI de la UPC a través de otro proyectista. En el capítulo 10 se explica cómo se comunica la API desarrollada con su aplicación con el objetivo de independizar completamente las tecnologías utilizadas en cada proyecto.

5 Desarrollo de una API para interactuar con el sistema KNX

En el CVI la principal isla de control está formada por dispositivos conectados por una red Konnex (también conocida como KNX). Así que el primer paso y el más lógico fue la de crear una API para poder acceder al servidor KNX y poder controlar dichos dispositivos.

El objetivo de esta API no es solo poder controlar los dispositivos conectados al KNX, sino poder controlar cualquier tipo de dispositivo, simplemente añadiendo nuevos módulos para atacar las diferentes tecnologías que se quieran abordar.

En este capítulo se puede ver el proceso de creación de esta API, su estructura y ejemplos de su funcionamiento.

La intención es desarrollar una API capaz de comunicarse con todos los dispositivos de la casa, que permitiera a una aplicación cualquiera controlarlos sin necesidad de conocer la tecnología utilizada, es decir que añadiese un nivel de abstracción entre la aplicación y la tecnología.

El desarrollo de esta API ha tenido dos etapas, la primera consistió solo en crear la estructura de la API e implementar una interfaz para interactuar con el sistema KNX.

Como expliqué en el capítulo anterior, en el CVI existen muchas tecnologías diferentes pero hay una que predomina por encima de todas: es la tecnología Konnex o KNX. Dado el poco tiempo que tengo para realizar mi proyecto, no me podía permitir implementar la API para todas las tecnologías así que me centré en la tecnología KNX. Eso sí, la API debería permitir que en un futuro se le pudieran añadir soporte a otras tecnologías.

La segunda fase consistió en la integración con sistema DLA para permitir una total independencia de la tecnología. Esta fase se explica en el capítulo 10 de este documento.

5.1 Elección de la tecnología a utilizar

Después de una reflexión sobre nuestras necesidades los requisitos que debe tener la API, ambos se pueden resumir en esta lista:

- Fácilmente modificable para dar soporte a otro tipo de dispositivos.
- Independencia de la tecnología que utilicen los dispositivos soportados.
- Independencia de la tecnología que se vaya a usar en una aplicación final.
- Fácil de utilizar desde cualquier tipo de plataforma.

Para satisfacer el primer requisito considero necesario utilizar una plataforma orientada a objetos. De esta forma se podría utilizar una estructura jerárquica de clases, donde habría una clase abstracta “padre” y una clase “hija” por cada tecnología que se quiera dar soporte. Así añadir otra tecnología se podría reducir a añadir una clase más que sea hija de la clase padre.

Dados los otros tres requisitos mi decisión fue la de utilizar la tecnología J2SE de java, ya que esta está presente en la mayoría de plataformas utilizadas (Windows, Linux, Mac OS, Android, etc.) y una aplicación podría ejecutarse en cualquiera de estas plataformas sin necesidad de cambios en el código.

5.2 Calimero

En mi búsqueda de información sobre la tecnología KNX la mayoría de mis fuentes hacían referencia a la librería Calimero.

Se trata de una colección de APIs de código abierto escritas en Java que permite una fácil comunicación sobre sistemas EIB/KNX, incluyendo acceso remoto y control. Fue presentada por primera vez al público en la KNX Scientific Conference 2005 como parte del proyecto KNXLive!.

Dado que esta librería está escrita también en la Java, era todo lo que necesitaba para poderme comunicar con el servidor KNX y controlar los dispositivos conectados a él de forma fácil.

La última versión liberada y la que yo he utilizado es la Calimero 2.0 alfa 4.

El software Calimero cierra la brecha que hay entre la aplicación y el medio KNX. Para conseguirlo se puede hacer con una variedad de protocolos, como a través de una interfaz de red IP o mediante una conexión por cable serie a un acoplador de bus. Además, la red KNX puede utilizar diferentes medios de comunicación, que requieren formatos específicos adecuados medio.

Calimero añade una interfaz abstracta bien definida que encapsula y oculta este tipo de diferencias. Como consecuencia, todo el diseño de Calimero sigue una arquitectura conocida como la arquitectura waist-line. La Figura 5.1 muestra cómo, en esta arquitectura, la funcionalidad se divide en tres capas. Las implementaciones de los servicios básicos, en nuestro caso la gama de protocolos de acceso a la red, están ubicados en la parte inferior. En el medio, el enlace de red ofrece una interfaz homogénea y estándar para la comunicación con las redes KNX, ocultando el servicio básico elegido. Este enlace es considerado la “cintura” de la arquitectura. En la capa superior encontramos los servicios de alto nivel para comunicarse con el medio KNX.

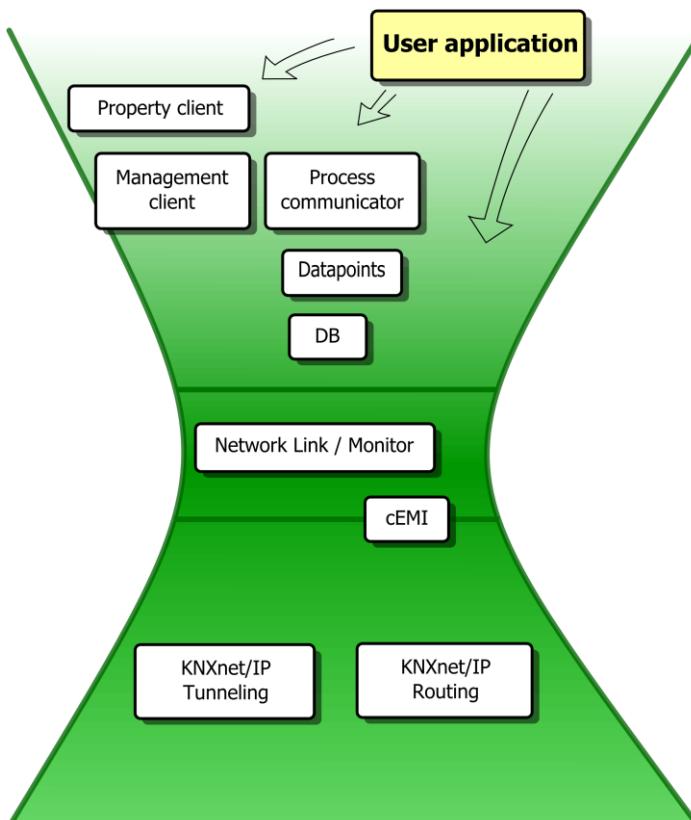


Figura 5.1: Arquitectura Waist-line y el sistema no estricto de capas en Calimero

Calimero sigue un enfoque de capas no estricta. El usuario no está obligado a interactuar con las clases del nivel superior. Así, un usuario puede elegir el nivel deseado de la abstracción de la API.

5.3 Implementación

La API tiene que permitir la fácil adición de interfaces para la comunicación con otras tecnologías. Para satisfacer este requisito he optado por utilizar, para las interfaces, la siguiente estructura de clases:

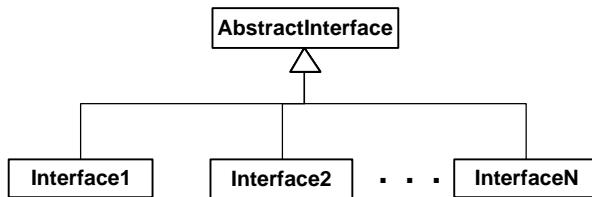


Figura 5.2: Esquema de las interfaces

Se dispone de una clase Abstracta que define la estructura que debe tener cada interfaz a implementar y de esta hereda cada una de estas interfaces. Una estructura simple pero que nos permite fácilmente crear una nueva interfaz y añadirla a nuestro sistema.

La estructura interna de la clase abstracta debe ser lo suficientemente genérica como para permitir la adición de cualquier nueva tecnología a la API. Por este motivo una interfaz sólo implementará las siguientes cuatro funciones:

- **Connect:** Permite iniciar la conexión con el medio específico. Puede no ser necesario dependiendo del protocolo con el que se comunique.
- **Disconnect:** Permite acabar la conexión con el medio. Igual que la función *Connect*, puede no ser necesaria dependiendo del protocolo con el que se comunique.
- **Read:** Permite la lectura de cualquier dato del medio, siempre que éste permita la lectura,
- **Write:** Permite la escritura de cualquier dato del medio, siempre que éste permita la escritura.

Tanto la función “Read” como “Write” necesitarán de parámetros específicos en función del protocolo que haya detrás. Así que he optado por declarar dichas funciones de la siguiente forma:

```
Read(entrada address : int, entrada type : int) : Object
```

```
Write(entrada address : int, entrada value : Object, entrada type : int)
```

Cada dispositivo, sea cual sea el medio, va a tener un identificador o dirección, ese será el parámetro “address”. Puede hacer falta indicar el tipo de datos que se intenta leer o escribir, este será el parámetro “type”. Cada interfaz deberá indicar, de alguna forma los tipos de datos soportados. Como no se sabe qué tipo de datos se van a enviar o recibir se ha optado por utilizar un objeto “Object” para ese fin, que la aplicación final deberá convertir al tipo de datos deseado según la información que le provea la interfaz en cuestión.

Aquí se puede apreciar como la aplicación no es del todo independiente de la tecnología que hay detrás de cada interfaz. En el capítulo 10, que habla de la integración con el DLA, explico cómo se consigue ese objetivo.

Como dije con anterioridad, me centré en la tecnología KNX a la hora de implementar las interfaces, así que hasta éste punto sólo esta implementada la interfaz para este medio. El siguiente esquema es el de las clases que forman la API hasta este punto:

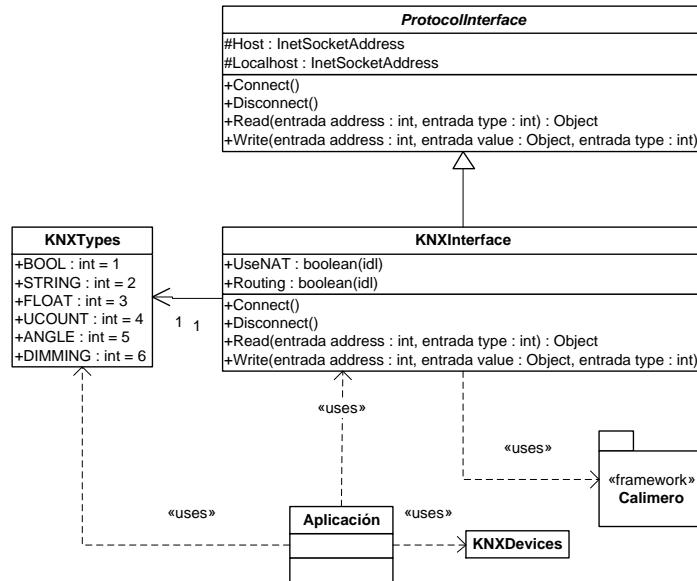


Figura 5.3: Esquema UML de la API

La clase **KNXdevices** contiene constantes con todas las direcciones de los dispositivos en el medio KNX del CVI y no forma parte de la API, sino de la aplicación que la utiliza. Su propósito es permitir un fácil acceso a dichas direcciones y evitar confusiones. Esta clase es exclusiva para el CVI, pero como allá es donde se hicieron las pruebas y las aplicaciones que se explicarán en siguientes capítulos, la añado al esquema para que el lector sepa de su existencia.

La clase **KNXTypes** contiene constantes que representan los diferentes tipos de datos con las que la interfaz KNX permite interactuar. Como dije con anterioridad, cada interfaz debería de alguna forma indicar los diferentes tipos que permite.

En el ANEXO III: Código principal de la API se adjunta el código principal comentado de la API, por si se quiere más información.

6 Desarrollo de un extensor inalámbrico WiFi

Con el fin de poder controlar otra isla de control muy presente en cualquier hogar, como los dispositivos multimedia controlados por mandos infrarrojos (como la Televisión, reproductor de DVD, equipos de música, etc.), se decidió diseñar un aparato, a modo de extensor inalámbrico, que permitiera el control de dichos dispositivos a través de una conexión TCP/IP, permitiendo el control de éstos a través de cualquier PC o teléfono móvil con WiFi o conexión de datos.

En este capítulo podréis ver cómo ha sido el proceso de desarrollo de este dispositivo, así como su diseño y funcionamiento.

6.1 Diseño del extensor

El diseño del extensor supuso dos retos: encontrar la forma de generar una señal IR válida y encontrar la forma de comunicarme vía WiFi desde el microcontrolador. Las pruebas realizadas, que se explican a continuación, son las que me ayudaron a determinar las mejores técnicas y los mejores componentes a utilizar para conseguir mi propósito.

6.1.1 Pruebas de generación de señales IR con el PIC16F876

Las primeras pruebas que realicé consistieron en la generación de señales IR desde un PIC16 convencional. La elección de este microcontrolador fue simplemente por haber trabajado con él en otras ocasiones, lo que me permitió avanzar relativamente deprisa. Sabía de antemano que este no iba a ser el microcontrolador elegido para la realización del prototipo ya que es demasiado simple, pero mi objetivo era solo encontrar la forma de poder generar la señal. El esquema del circuito es muy simple, consistía únicamente en un cristal de cuarzo de 20 MHz, un LED convencional y una resistencia para reducir la intensidad (en el esquema no se muestran las conexiones de corriente y masa):

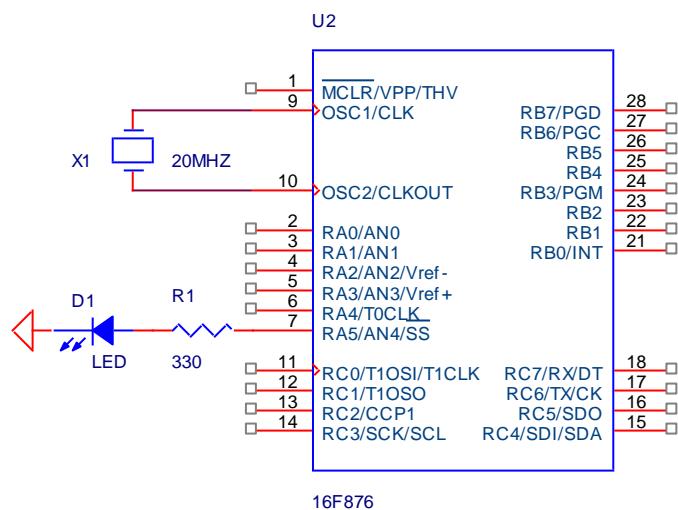


Figura 6.1: esquema de las pruebas con el PIC16F876

Lo primero fue conseguir generar una señal cuadrada de 40 KHz a través de un pin de salida del PIC. Para conseguirlo primero hay que calcular el período para esa frecuencia:

$$\frac{1 \text{ s}}{40 \text{ KHz}} = 25 \mu\text{s}$$

Luego hacer un bucle infinito que en cada iteración cambie de valor el pin y tarde 25 μ s en pasar a la siguiente iteración. Hay que tener en cuenta que las instrucciones de salto y de asignación tardan en ejecutarse 200 ns con una frecuencia de 20 MHz.

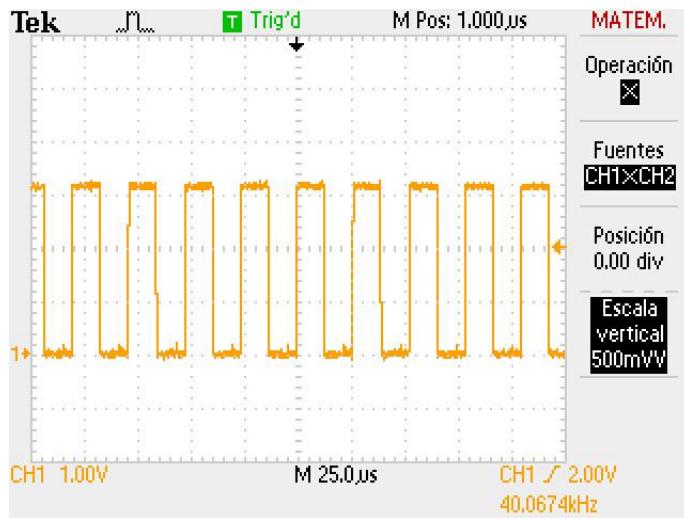


Figura 6.2: Captura del osciloscopio de una señal de 40 KHz

Una vez generada la onda portadora hay que enviar información. Dependiendo del protocolo se envía de una forma u otra, para más detalles se puede consultar el ANEXO II: Protocolos de comunicación IR implementados. A continuación se adjunta una imagen de cómo queda la señal codificando una cadena infinita de ceros y unos en protocolo SIRC:

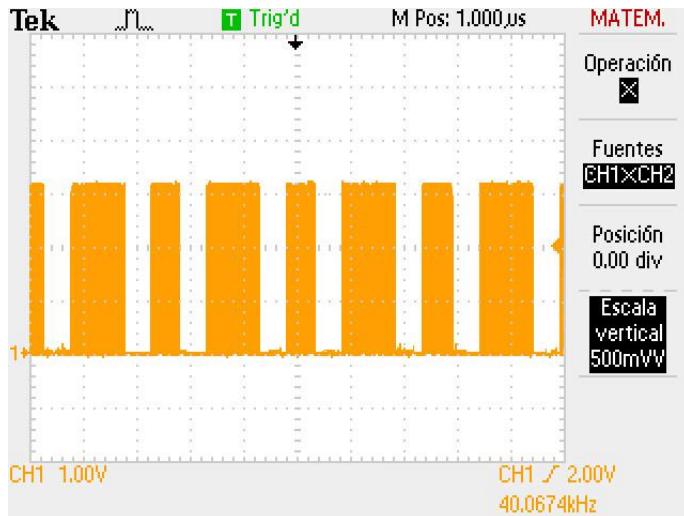


Figura 6.3: Captura del osciloscopio de una señal de ceros y unos modulada a 40 KHz

Para comprobar que las tramas que generaba era correctas, a falta de un televisor en el laboratorio, utilicé el mando Logitech de la Figura 6.4: Mando Logitech, ya que éste posee la capacidad de aprender un comando IR a través de un receptor que tiene en la parte trasera. Si la trama es correcta el mando la identifica y la memoriza, en caso de que la trama no sea correcta el mando no entiende la señal y da un mensaje de error.



Figura 6.4: Mando Logitech

Una vez ya era capaz de conseguir generar tramas SIRC perfectamente desde un PIC16 quise probar con otros protocolos como el RC5 y el JVC, que no explicaré aquí ya que en el ANEXO II: Protocolos de comunicación IR implementados se explican más detalladamente.

6.1.2 Pruebas con el dsPIC30F4011 y la TCP/IP Stack de Microchip

Estas pruebas se realizaron una vez decidido que el módulo WiFi a utilizar era el ZG2100M de Microchip, ya que tiene todas las características que necesitaba para la realización de mi proyecto. Dicho módulo está preparado para ser utilizado junto a un PIC. El PIC en cuestión debe utilizar la TCP/IP stack de microchip, la cual es gratuita y existen muchos ejemplos de su uso. En el punto 6.3.1 se explica de forma detallada su funcionamiento.

Antes siquiera que el módulo WiFi llegara empecé a hacer pruebas con la TCP/IP stack y el microcontrolador dsPIC30F4011. El motivo por el que elegí este PIC fue porque disponíamos de muchos en el laboratorio, y para hacer las primeras pruebas ya me servía.

Para las pruebas utilicé una placa de desarrollo llamada dsPICDEM 2, que permite la utilización de este PIC y da mucha más comodidad a la hora de trabajar.

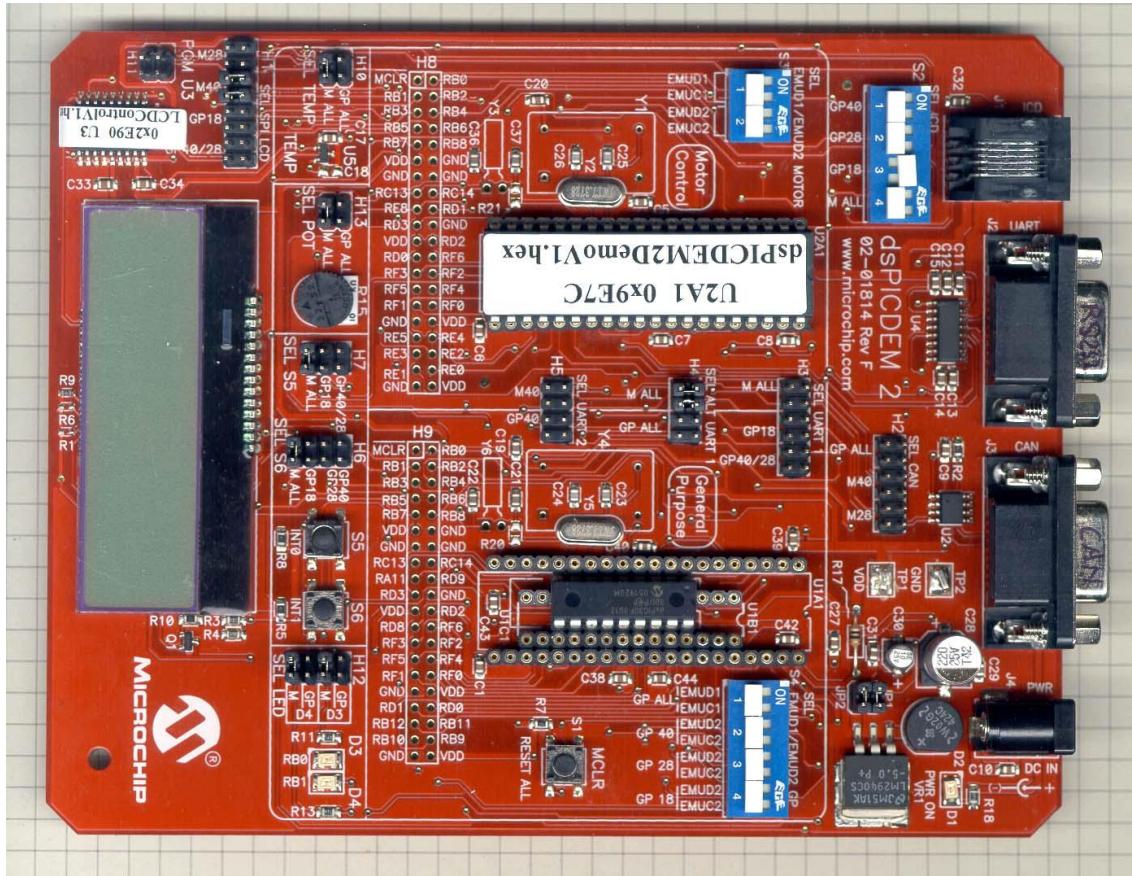


Figura 6.5: Imagen del dsPICDEM 2

Las pruebas consistieron en la compilación de un programa que fuera capaz de crear un servidor TCP para la comunicación con un computador cualquiera.

Cabe decir que perdí muchos días intentando comprender la pila TCP/IP y otros más intentando compilar el programa. La causa de que todo resultó ser el tamaño descomunal que tiene la pila TCP/IP. Como el dsPIC30F4011 posee una memoria de programa de 48 Kbytes, un tamaño ridículo para una aplicación tan compleja como el que pretendía hacer. Aun así fui capaz de compilar el programa, eso sí, eliminando una gran cantidad de funcionalidades que sabía que en un futuro próximo iba a utilizar. Por eso mismo se abandonó la posibilidad de usar este PIC.

6.1.3 Pruebas con el PIC24FJ128GA010 y el módulo ZeroG ZG2100M PICtail Plus

Después de las experiencias con el dsPIC anterior me informé mejor de los requisitos que necesitaba el módulo WiFi para funcionar. Uno de los microcontroladores que recomendaban era el PIC24FJ128GA010, principalmente por su memoria de 128 Kbytes. Además disponía de una gran cantidad de código de ejemplo para este PIC, lo que fue decisivo en mi elección.

El programa compilaba perfectamente para este procesador, pero tenía un problema, el chip no estaba disponible en encapsulado DIP, es decir compatible con las llamadas protoboards (la distancia entre pines es muy amplia y no requiere soldar para usarlo). En consecuencia venía en formato TQFP de 100 pines, esto quiere decir que el chip tiene 100 pines en un tamaño de un centímetro cuadrado. Lo que me obligó a diseñar un adaptador a formato DIP.

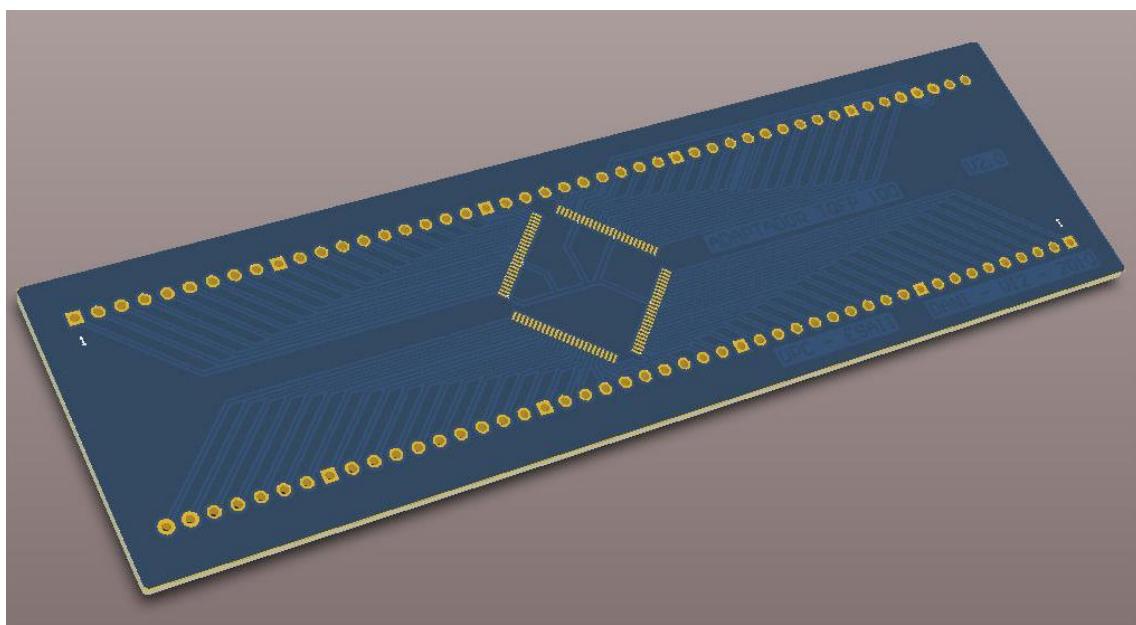


Figura 6.6: Adaptador TQFP 100 v2.0

Este adaptador ha sido desarrollado con la aplicación Altium, y permite acceder a cada uno de los 100 pines.

El siguiente paso fue conectar los pines necesarios del PIC con el módulo WiFi y el resto de componentes, tal como se muestra en el esquema siguiente:

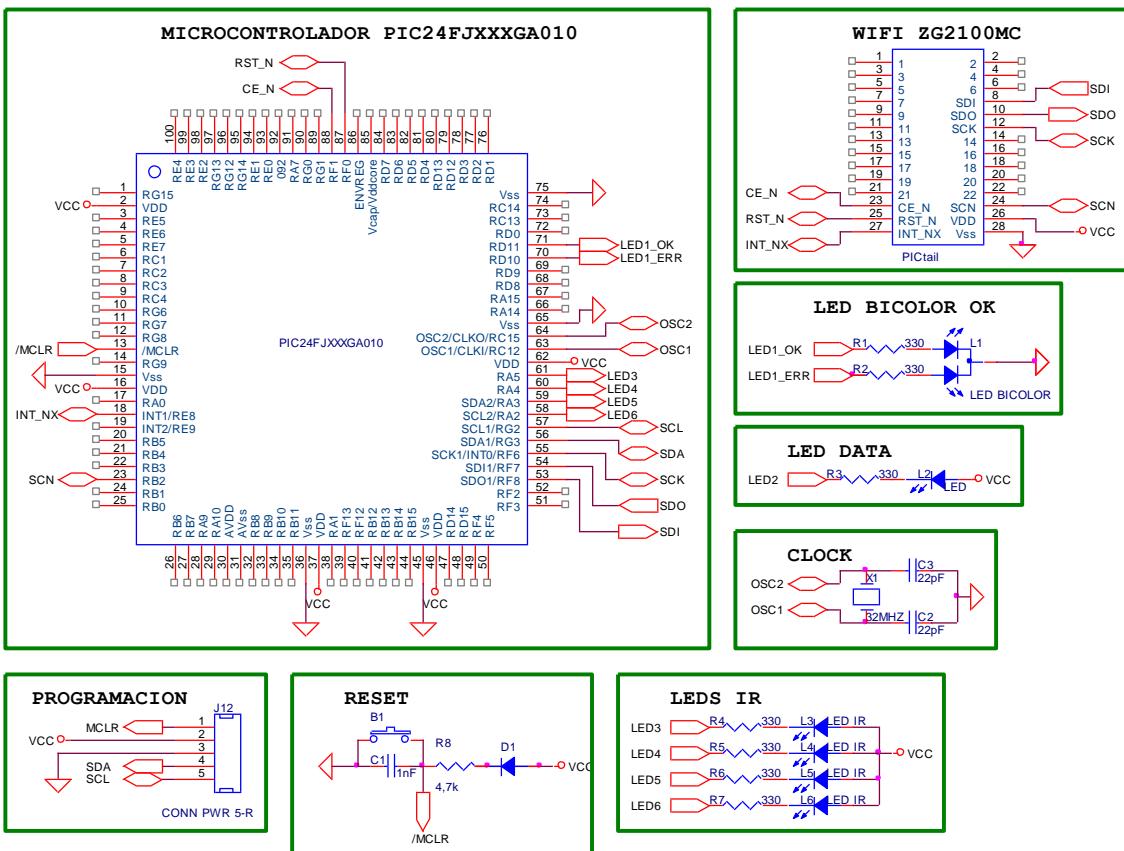


Figura 6.7: Esquema de interconexión PIC24- ZG2100M

Para hacer más fácil las pruebas se compró el módulo ZG2100M encapsulado en una pequeña placa de desarrollo pensada para ser utilizada conjuntamente con una placa de desarrollo que no tenemos. Aun así me permitió una mayor facilidad para acceder a los pines del chip.



Figura 6.8: ZeroG ZG2100M PICtail Plus

Para ser sinceros, nunca llegué a comunicarme correctamente con éste módulo, la causa sigue siendo un misterio. Puede que llegara defectuosa a nuestras manos o que haciendo pruebas se haya fastidiado, quizás por una subida de tensión.

Incluso llegué a pedir un dsPIC33 de muestra pero daba el mismo problema. No era culpa del microcontrolador ni del programa, ya que ni los ejemplos proporcionados por Microchip

funcionaban. Este contratiempo me hizo perder como un mes de trabajo y retrasó mi proyecto considerablemente.

Al final se optó por comprar un módulo WiFi con un PIC24 de la misma familia incluido en la misma placa y la nueva versión del chip ZG2100M. Esta placa, la llamada FlyPort, ha sido creada y está siendo producida por un grupo Italiano de desarrollo de hardware abierto llamado OpenPicus. Con la suerte que justo lo sacaron a la venta cuando tuve este contratiempo, en septiembre.

A continuación se explica de forma más detallada los elementos definitivos utilizados para el desarrollo del prototipo.

6.2 Componentes definitivos a utilizar

Después de realizar pruebas con una gran serie de componentes se llegó a la conclusión de utilizar los elementos descritos a continuación para componer el prototipo del extensor inalámbrico.

6.2.1 Módulo WiFi MRF24WB0MA

MRF24WB0MA es un módulo transceptor de WiFi IEEE 802.11. Posee una antena integrada de PCB y es compatible con el protocolo TCP/IP. Es compatible con cientos de microcontroladores PIC y utiliza una interfaz SPI de 4 hilos. Además posee soporte por hardware para AES y TKIP (WEP, WPA, WPA2).

Este módulo está diseñado para ser utilizado con la versión 5.25 o posterior de la TCP/IP Stack de Microchip, que es un conjunto de programas que presta servicios a aplicaciones basadas en TCP/IP (HTTP Server, cliente de correo, etc.) o en aplicaciones personalizadas.

Con la aplicación HTTP Server se pueden ejecutar scripts Ajax para la utilización de gráficos avanzados y análisis de datos para poder visualizarlos desde cualquier navegador web.

El MRF24WB0MA es un módulo de montaje en superficie. Las dimensiones del módulo se muestran en la Figura 6.10. El módulo de circuito impreso (PCB) es de 1 mm de espesor con puntos de montaje en dos lados. Su tamaño es suficientemente pequeño para ser incorporado en nuestra placa sin causar problemas de espacio.

Existe el modelo MRF24WB0MB sin antena integrada, pero por temas prácticos no se ha elegido ese modelo.



Figura 6.9: Módulo MRF24WB0MA

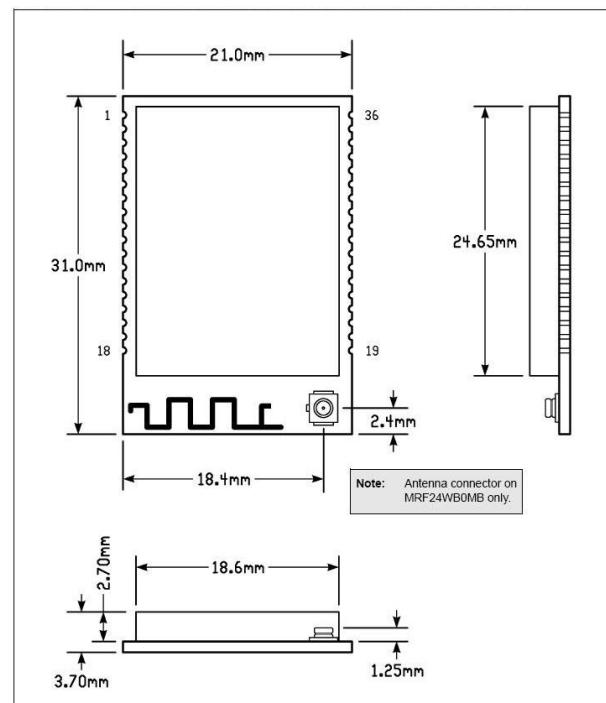


Figura 6.10: Dimensiones del módulo MRF24WB0MA

El diagrama de bloques en la Figura 6.11: Diagrama de bloques representa el módulo MRF24WB0Mx y la comunicación con un PIC18, PIC24, dsPIC33 o PIC32 de Microchip a través de cuatro pines de interfaz serie SPI más 5 pins: interrupción, encendido, hibernar, Reset y masa.

El módulo se ejecuta con una tensión de alimentación de 3.3V. También es compatible con JTAG y depuración en serie para testear su funcionamiento, pero no es necesario para su correcto funcionamiento.

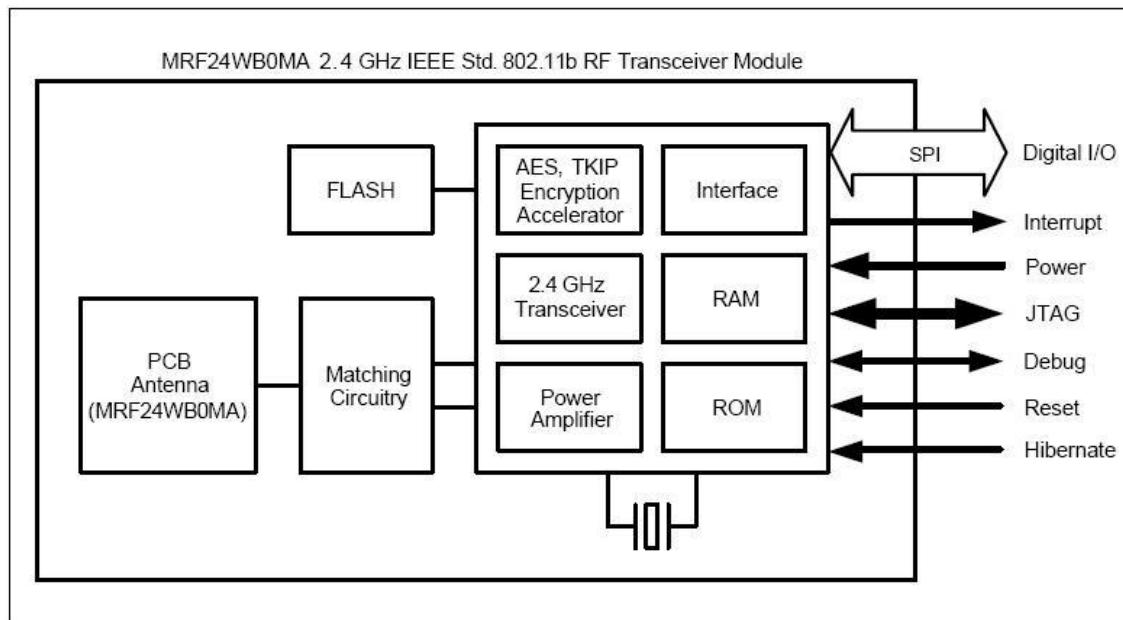


Figura 6.11: Diagrama de bloques

La Figura 6.12: Interfaz Microcontrolador - MRF24WB0Mx muestra un ejemplo simplificado de conexión entre un microcontrolador PIC de Microchip y el módulo MRF24WB0Mx.

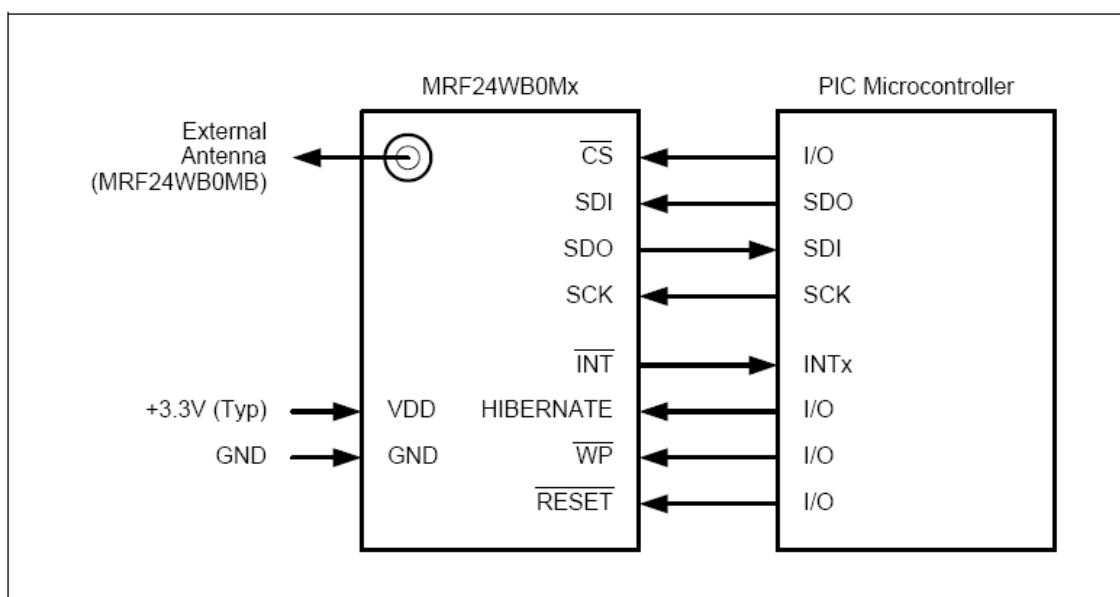


Figura 6.12: Interfaz Microcontrolador - MRF24WB0Mx

6.2.2 Microcontrolador PIC24FJ256GA106

El microcontrolador PIC24FJ256GA106 ha sido el elegido para la realización del prototipo debido principalmente a su gran capacidad de memoria, que permitirá, en un futuro, alojar sin problemas una página web de configuración que está previsto incorporar. También al hecho de incorporar más de un módulo SPI y UART, el primero utilizado para la comunicación con el módulo WiFi y el segundo para la comunicación en serie con el ordenador para debuggar la aplicación. Cabe destacar que, aunque existe una gran variedad de microcontroladores compatibles con el módulo WiFi anteriormente mencionados, éste es uno de los que Microchip recomienda para su uso, junto con su TCP/IP Stack, explicada en el punto 6.3.1.

Las principales características de este chip son:

- Rango de voltaje operativo de 2.0V a 3.6V.
- Tolera un voltaje de entrada de 5.5V en los pines digitales.
- 16 MIPS a una frecuencia de 32 MHz.
- Memoria de 256 KB.
- Compatibilidad con In-Circuit Serial Programming™ (ICSP™) y In-Circuit Debug (ICD) a través de 2 Pins.
- Tres módulos SPI.
- Cuatro módulos UART.
- Peripheral Pin Select.

La característica Peripheral Pin Select (PPS) permite elegir los pines que van a utilizar los diferentes periféricos, tales como SPI, UART, interrupciones, etc. programándolos por software. Con ésto se consigue dar una mayor flexibilidad al diseñador en la forma de disponer y conectar los periféricos. Gracias al PPS es posible reducir al mínimo el tamaño del chip y el número de pines.

A continuación se adjunta un esquema del chip en cuestión y sus respectivos pines:

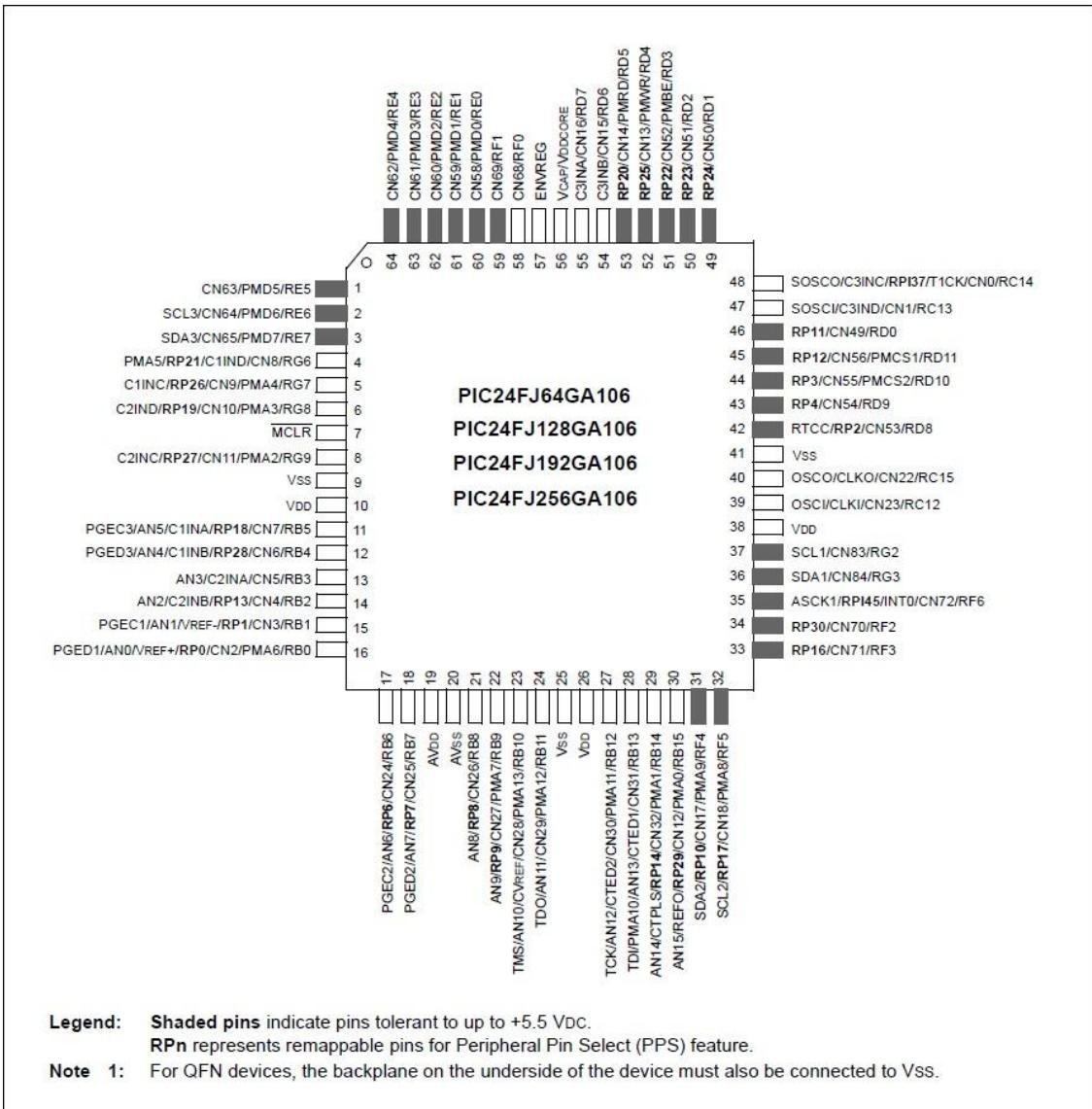


Figura 6.13: Microcontrolador PIC24FJ256GA106

6.2.3 Módulo WiFi FlyPort

FlyPort es un módulo profesional pequeño y delgado con conectividad WiFi, diseñado para la integración con otros productos electrónicos existentes.

Está basado en el procesador de 16 bits PIC 24F256FJ106GA de Microchip y módulo WiFi MRF24WB0MA/RM compatible con la norma 802.11 b/g/n.

Se ha optado por utilizar este módulo por el hecho de incorporar el microcontrolador y módulo WiFi deseados en una misma placa, lo que ha facilitado muchísimo el desarrollo de un primer prototipo y la realización de las pruebas.

El modulo incorpora una aplicación bootloader para facilitar su programación a través de un puerto serie, así como una aplicación webserver de ejemplo para tomar como punto de partida en posibles aplicaciones.

En el esquema de la Figura 6.16: Esquema de bloques del módulo FlyPort se puede apreciar los diferentes componentes del módulo FlyPort y las conexiones para interactuar con él. Dichas conexiones son las siguientes:

- 5V o 3.3V de corriente continua de entrada.
- Hasta 400m de alcance WiFi (en campo abierto).
- Puerto serie UART.
- Reloj en tiempo real RTC.
- Bus I2C.
- Bus SPI.
- Puertos de E/S Digital.
- Puertos PWM.
- Puertos Analógicos.



Figura 6.14: FlyPort desde arriba

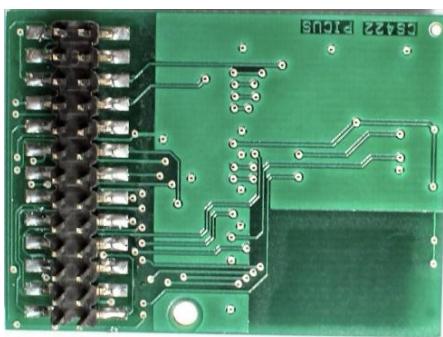


Figura 6.15: FlyPort desde abajo

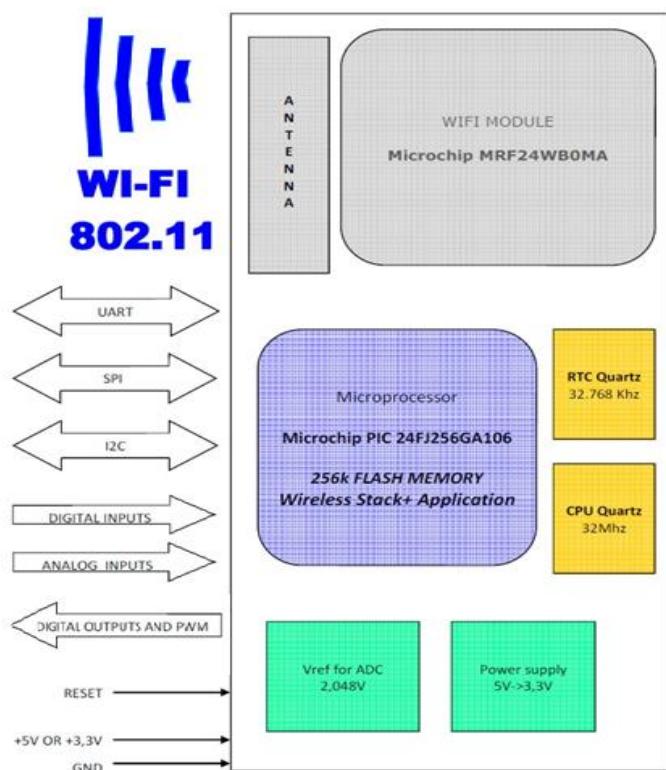


Figura 6.16: Esquema de bloques del módulo FlyPort

La Figura 6.17 Corresponde al esquema eléctrico del módulo FlyPort, con las interconexiones de cada componente.

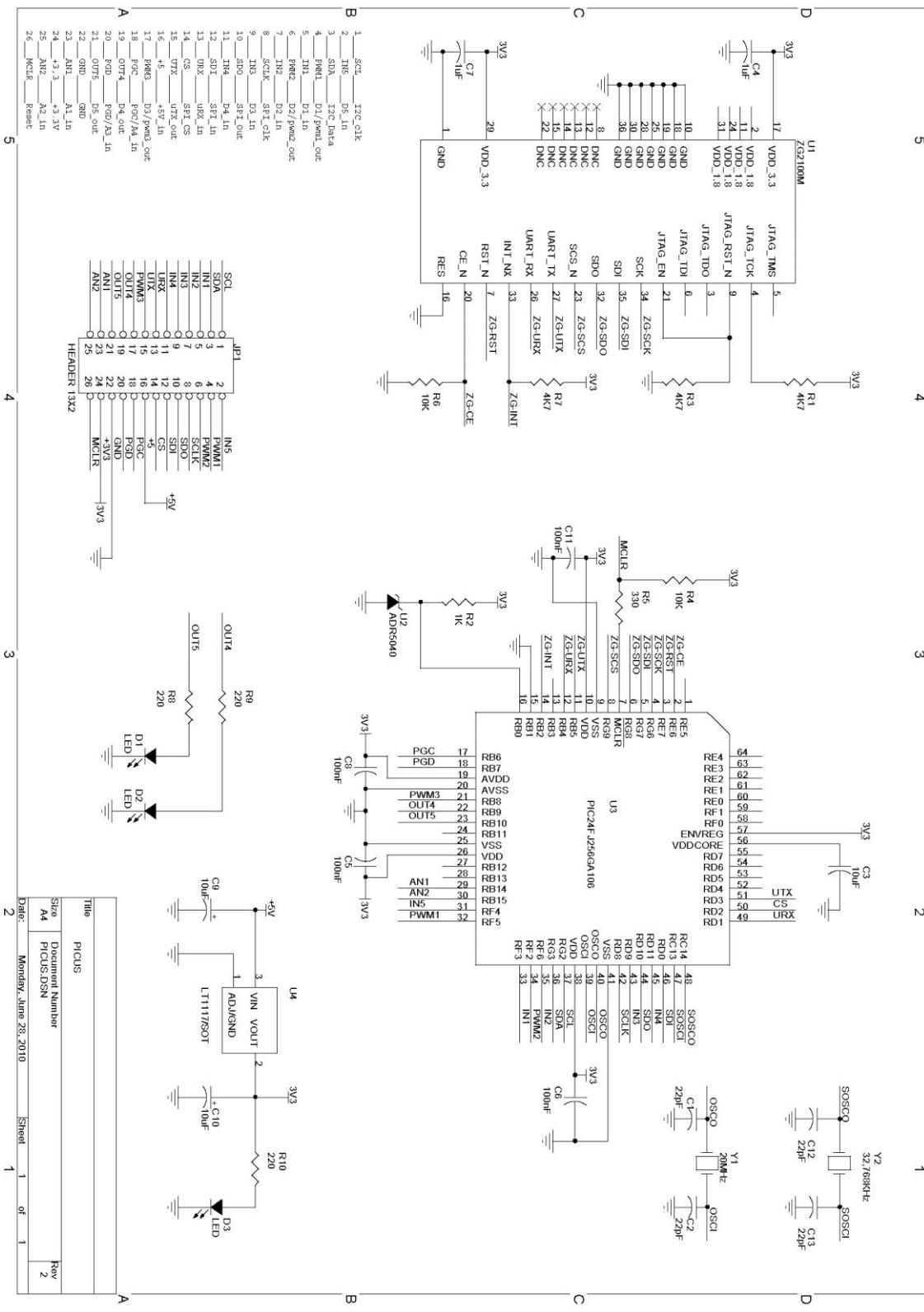


Figura 6.17: Esquema del módulo FlyPort

6.3 Firmware del Microcontrolador

El firmware que se ha elaborado para este microcontrolador se compone de tres partes:

- La TCP/IP Stack de Microchip.
- El servidor TCP para la recepción de las órdenes.
- El envío de la señal infrarroja.

A continuación se explica de forma detallada cada una de estas partes del código, así como el protocolo de comunicación que se ha optado por utilizar entre el usuario y el servidor TCP.

6.3.1 TCP/IP Stack de Microchip

La TCP/IP Stack de Microchip es un conjunto de programas que presta servicios a aplicaciones basadas en TCP/IP (HTTP Server, cliente de correo, etc.) o para aplicaciones personalizadas. La TCP/IP Stack de Microchip se implementa en un sistema modular, siendo cada uno de sus servicios altamente abstractos. Esto permite que los programadores no necesiten saber todas las complejidades de la red TCP/IP para su uso.

La mayoría de las implementaciones de la pila TCP/IP siguen una arquitectura conocida como el “modelo TCP/IP de referencia”. El software basado en este modelo se divide en varias capas, cada capa se apila una encima de otra (de ahí el nombre de “pila TCP/IP”) y cada capa accede a los servicios de una o más capas directamente debajo de él, tal como se puede ver en la Figura 6.18: Capas del modelo TCP/IP de referencia.

Al igual que el modelo TCP/IP de referencia, el modelo de Microchip divide la pila TCP/IP en múltiples capas, como se muestra en la Figura 6.19. El código de la aplicación de cada capa se encuentra en un archivo de código fuente por separado, mientras que los servicios y las API (Application Programming Interfaces) se definen a través de una cabecera.

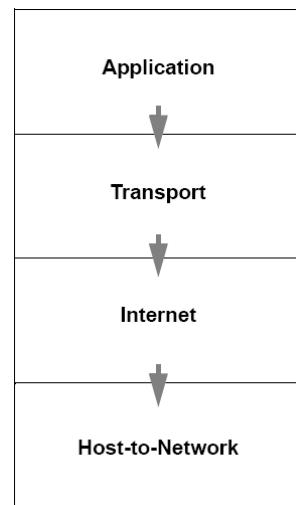


Figura 6.18: Capas del modelo TCP/IP de referencia

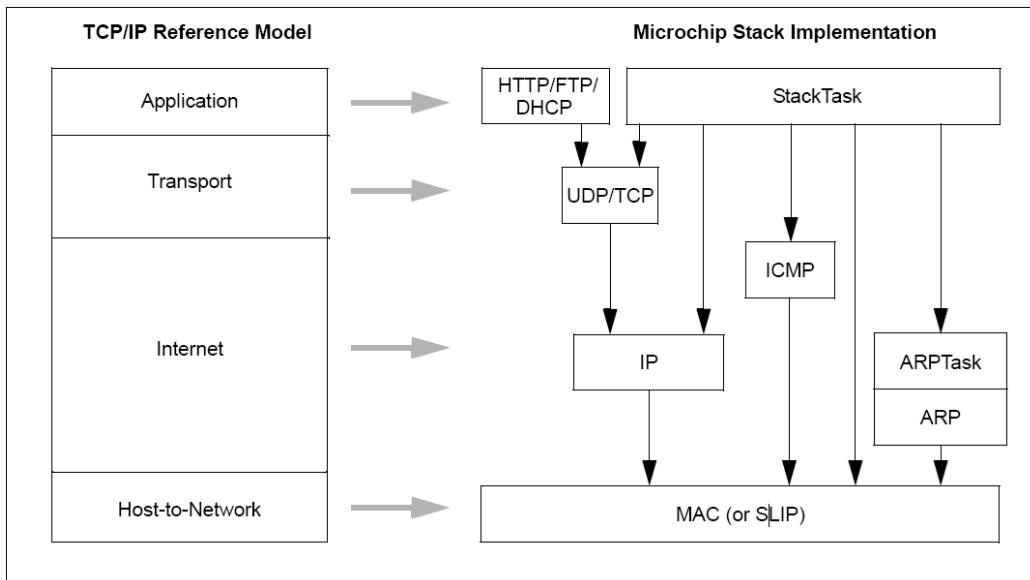


Figura 6.19: Comparación del modelo TCP/IP de referencia y la pila TCP/IP de Microchip

A diferencia del modelo TCP/IP de referencia, algunas de las capas de la pila TCP/IP de microchip acceden directamente a una o más capas las cuales no están directamente debajo de ellas. Esta decisión fue tomada para ofrecer aplicaciones más eficientes, no hay que olvidar que estos programas están pensados para dispositivos con recursos de memoria y calculo muy limitados.

Un cambio importante respecto a una aplicación tradicional TCP/IP es la adición de dos nuevos módulos: “StackTask” y “ARPTask”. StackTask administra las operaciones de la pila y todos sus módulos, mientras que ARPTask gestiona los servicios de la capa ARP (Protocolo de resolución de direcciones).

La TCP/IP Stack de Microchip no implementa todos los módulos que están normalmente presentes en la pila TCP/IP. En nuestro caso los que faltan no los necesitamos, pero siempre se pueden implementar como una aplicación o módulo independiente.

Para esta aplicación los módulos de aplicaciones que se han activado son los siguientes:

- **ICMP_SERVER:** Permite la capacidad para responder a una petición ping.
- **DHCP_SERVER:** Servidor DHCP para un único host. Útil cuando nos conectamos en modo Access-Point.
- **DNS:** Un servidor DNS (Domain Name Service) para resolver las direcciones IP de los clientes.

6.3.2 Aplicación servidor TCP

A parte de estas aplicaciones que incorpora la propia pila TCP/IP se ha implementado un servidor TCP para la comunicación con el usuario. Esta aplicación servidor es la encargada de procesar la orden procedente del usuario y ejecutar las operaciones correspondientes.

Los diferentes estados por los que puede pasar el servidor se pueden ver en el siguiente diagrama de estados:

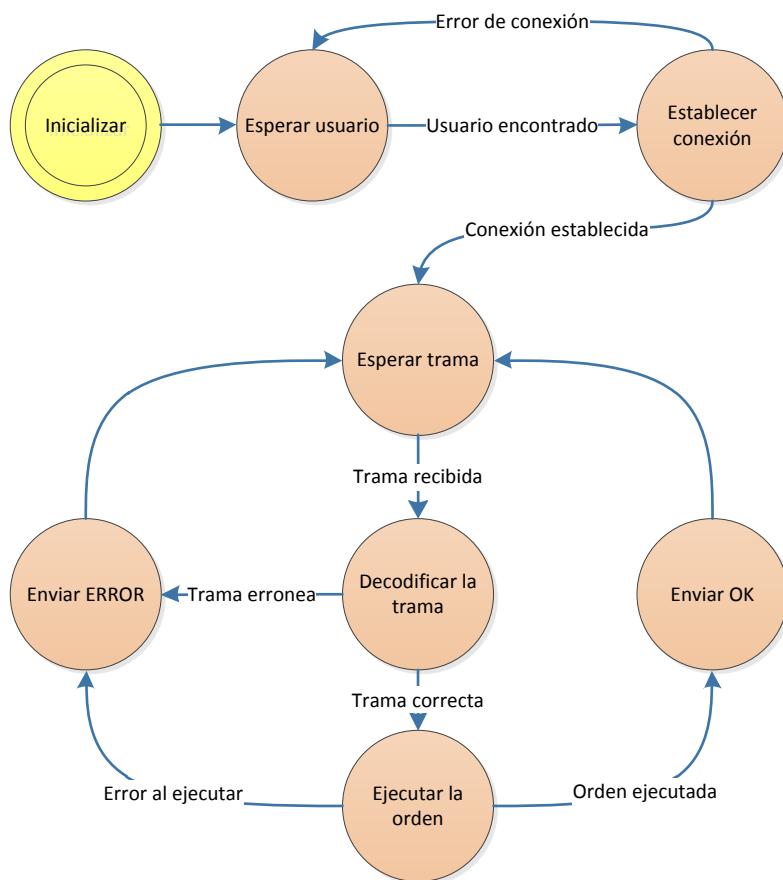


Figura 6.20: Diagrama de estados del servidor TCP

Para más detalle consultad el ANEXO IV: Código del servidor TCP implementado en el PIC24.

6.3.3 Protocolo de comunicación para el envío de las órdenes

Las órdenes que se quieran enviar al extensor WiFi deben cumplir un protocolo que yo mismo he definido. Para la elaboración de este protocolo he tenido en cuenta las siguientes necesidades:

- Es necesario indicar el protocolo IR con el que se enviará la señal.
- Es necesario indicar la salida o puerto del extensor a utilizar. Por defecto se envían por todas las salidas.
- Es necesario indicar el toda la trama IR, ya que el extensor no conoce ni los dispositivos ni los comandos que existen, únicamente como se envía una trama en un protocolo determinado.

Para indicar el protocolo IR a utilizar he optado por enumerar del 0 al 11 los diferentes protocolos implementados hasta ahora. A continuación el trozo de código que asocia cada protocolo a un número:

```
#define IR_PROTOCOL_ITT          0
#define IR_PROTOCOL_JVC           1
#define IR_PROTOCOL_NEC          2
#define IR_PROTOCOL_NRC17         3
#define IR_PROTOCOL_SHARP         4
#define IR_PROTOCOL_SIRC          5
#define IR_PROTOCOL_RC5           6
#define IR_PROTOCOL_RC6           7
#define IR_PROTOCOL_RCMM          8
#define IR_PROTOCOL_RECS80        9
#define IR_PROTOCOL_RCA          10
#define IR_PROTOCOL_XSAT         11
```

El extensor tiene cuatro salidas o puertos, así que para indicar la salida por la que se debe enviar la señal he utilizado una idea similar a la anterior:

- Todos los puertos: 0
- Puerto 1: 1
- Puerto 2: 2
- Puerto 3: 3
- Puerto 4: 4

Ahora solo falta definir cómo será toda la trama. He optado por enviar la trama como una cadena de caracteres, por comodidad.

- Inicio de trama: Carácter 'I'.
- Protocolo: Dos caracteres numéricos (de 00 a 11).
- Puerto: Un carácter numérico (0 a 4).
- Trama IR: cadena de caracteres de 0 a 1 sin longitud máxima.
- Final de trama: Carácter 'E'.

Por ejemplo, si queremos enviar una trama en protocolo SIRC por el puerto 3 para bajar el volumen de la televisión sería como la siguiente: B53110010010000E

6.3.4 Emisión de la señal de infrarrojos

Como se comentó en el punto anterior, existen cuatro puertos IR de salida. Cada uno de ellos puede estar conectado a uno o más dispositivos, si el cable lo permite. El código realizado permite el envío de una misma señal en modo broadcast, es decir, enviar la señal por todos los puertos existentes.

A priori se puede pensar que esto es un problema ya que enviaremos la misma orden a dispositivos diferentes y estos podrían actuar de diferente forma, pero no ocurre así. Aunque existan dos dispositivos conectados que utilicen el mismo protocolo si estos son dispositivos diferentes (como un televisor y un reproductor de DVD) sólo uno entenderá la señal, ya que dentro del mensaje enviado, sea cual sea el protocolo, siempre se indica el tipo de dispositivo al que va dirigido. Sólo supondría un problema en el caso que se conecten dos dispositivos del mismo tipo y del mismo fabricante, aunque esto es improbable ya que, por lo general, nadie tiene en el mismo lugar dos dispositivos iguales (hay que recordar que la idea de este extensor es la de controlar dispositivos multimedia situados cerca entre ellos, como suele pasar en el salón de casa).

Para enviar la trama se tiene que generar una onda portadora que suele ir de los 35KHz hasta los 40KHz en función del protocolo. La señal se modula para aumentar el alcance y calidad, ya que una señal modulada es más resistente ante posibles ruidos e interferencias.

Dada la diferencia que existe entre protocolos, a continuación se explicará de forma genérica como se transmite la trama, sin entrar en detalle en la información que se envía. Para más información, en el ANEXO II: Protocolos de comunicación IR implementados se explica de forma detallada algunos de los protocolos implementados más utilizados en la actualidad.

6.3.4.1 Generación de la onda portadora

La trama debe codificarse en una señal portadora, idealmente sinusoidal. Por motivos prácticos y técnicos en este proyecto se han utilizado señales cuadradas, mucho más fáciles de generar desde un microcontrolador. A efectos prácticos, el resultado es el mismo. Cualquier receptor es capaz de discriminar la señal de igual forma.

Se ha conseguido generar esta señal cuadrada oscilando de 0 a 1, el bit correspondiente al puerto por el que se quiere enviar.

Para generar una señal de 36KHz deben enviarse 36.000 flancos ascendentes por segundo, lo que equivale a enviar flancos ascendentes con un periodo de 27.78125 μ s.

En el

ANEXO V: Código de generación de la señal IR desde el PIC24 se adjunta el código donde se genera la onda portadora y se envía la señal IR utilizando el protocolo SIRC de Sony.

6.3.4.2 Codificación de la trama

La trama puede ser de diferentes tamaños e incluir información diversa en función del protocolo utilizado, incluso dentro del mismo protocolo pueden existir tamaños variables. Sin embargo, todos los protocolos suelen enviar tramas que se componen de una cabecera para indicar que se va a empezar a emitir una trama (dependiendo del protocolo puede contener más información), una serie de bits para identificar el tipo de dispositivo (para que los demás receptores no hagan caso de la señal) y otra serie de bits para identificar la orden dentro de las posibles para ese dispositivo.

Para más información se puede consultar el ANEXO II: Protocolos de comunicación IR implementados, donde se explica detalladamente el funcionamiento de los principales protocolos IR.

6.4 Implementación del prototipo

Para la implementación del primer prototipo se ha optado por utilizar una caja de plástico duro convencional y encapsular el módulo FlyPort, con toda la circuitería adicional. A continuación se hace una explicación de los componentes utilizados, el esquema eléctrico de la caja, el diseño de la etiqueta y unas fotos del prototipo ya acabado.

6.4.1 Material utilizado

A continuación se enumera el material utilizado para el desarrollo de éste prototipo:

- Un módulo FlyPort.
- 4 conectores mini Jack de 3.5 mm macho.
- 8 conectores mini Jack de 3.5 mm hembra.
- 3 LEDs verdes de 3 mm.
- 2 LEDs multicolor (verde-rojo) de 3 mm.
- Interruptor de dos posiciones de panel.
- Conector de alimentación.
- 4 resistencias de 50 ohm ¼ W.
- Caja de plástico negra de dos piezas.
- 4 resistencias de 10k ohm ¼ W.
- 7 resistencias de 220 ohm ¼ W.
- Un conector de 5 pines macho.
- Cableado diverso.

En el capítulo 12, Valoración económica, aparecen reflejados estos mismos componentes con la referencia y el precio correspondiente.

6.4.2 Esquema

La siguiente imagen representa el cableado del prototipo. Se pueden diferenciar cuatro partes:

- **LEDs IR:** los JAGs del 1 al 4 son las entradas por donde vendrá la señal IR del aparato Logitech, la señal se mezcla con la del pin de salida IRLED(n) del FlyPort. La señal de ambas fuentes sale por los JAGs del 5 al 8.
- **LEDs informativos:** Se han colocado 5 LEDs, dos de los cuales son multicolor. La función de cada uno de ellos es la siguiente:
 - **LED_ON:** Se enciende siempre que el dispositivo este encendido.
 - **LED_CONN:** Se enciende cuando se ha establecido una conexión con una aplicación externa.
 - **LED_SIGNAL:** Este LED se enciende a la vez que se emite una señal de infrarrojos.

- **BLED_COM:** Se trata de un LED multicolor rojo-verde para indicar que se ha recibido una orden del exterior, cuando está verde significa que la orden es correcta y se va a procesar, si está rojo significa que ha habido un error.
- **BLED_WF:** Otro LED multicolor para indicar el estado de la conexión WiFi. Cuando se enciende el dispositivo el LED es de color naranja (verde y rojo a la vez), cuando el dispositivo está listo para recibir conexiones por WiFi se pone de color verde, en caso de algún error se pone de color rojo.

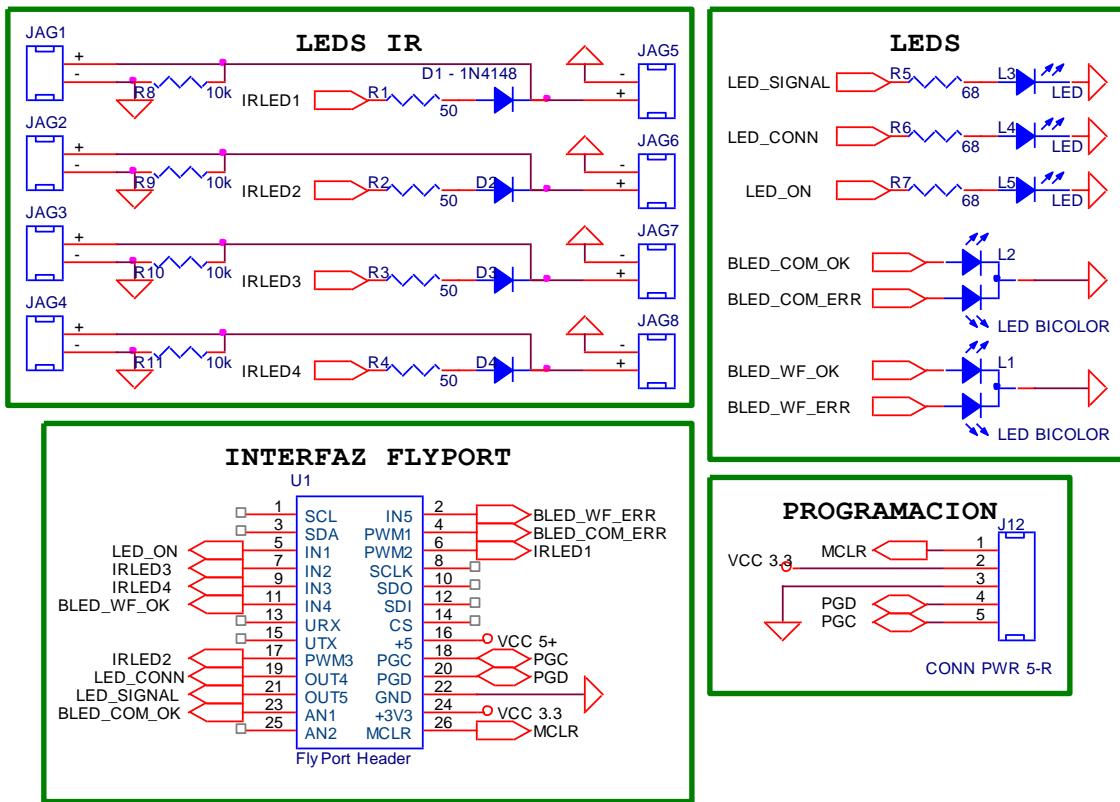


Figura 6.21: Esquema de conexiones del prototipo

6.4.3 Diseño de la etiqueta

Dado que la caja, con tantos conectores, LEDs y el interruptor tiene todo el lateral ocupado se optó por colocar una pegatina en la cubierta superior de la caja, posibilitando la indicación de todos los elementos exteriores e incluso el logo del CVI y de la UPC. Se ha dejado un espacio en blanco central para colocar posiblemente el logo del CVI, pero aún no se ha decidido. En la siguiente imagen se muestra la etiqueta mencionada:



Figura 6.22: Etiqueta del prototipo

6.4.4 Resultado final

A continuación se muestran una serie de imágenes del prototipo resultante de todo el trabajo realizado. En la primera imagen se puede apreciar el lado de la caja donde está el interruptor y donde se conecta a la corriente, así como los conectores mini Jack de entrada.



Figura 6.23: Prototipo Foto 1

La siguiente imagen se enseña el otro lado de la caja, donde se encuentran los LEDs (en la foto apagados) y los mini Jack de salida.



Figura 6.24: Prototipo Foto 2

Y por último una foto del prototipo conectado a la corriente y con un cable LED enchufado. Se puede apreciar como los LEDs que se encienden son los de Encendido y del estado de la WiFi. Estos son los LEDs que deberían encenderse nada más conectar el dispositivo, si ambos están verdes significa que ya está preparado para recibir conexiones externas.



Figura 6.25: prototipo conectado y con el cable IR

7 Adaptación de la API para la interacción con el extensor inalámbrico

Una vez finalizado el extensor inalámbrico, el siguiente paso fue la adaptación de la API explicada anteriormente, para que sea capaz de controlar los dispositivos multimedia a través de él. En este capítulo explico el resultado de ampliación de la API.

La ampliación de la API ha sido tremadamente sencilla. Toda la estructura ya estaba diseñada, solo hacía falta añadir la nueva interfaz para comunicarse con el extensor WiFi. El único inconveniente es que tuve que modificar la clase abstracta *ProtocolInterface* para hacerla que no implementase ciertas funcionalidades que no necesitaba la nueva clase. A continuación el esquema UML actualizado:

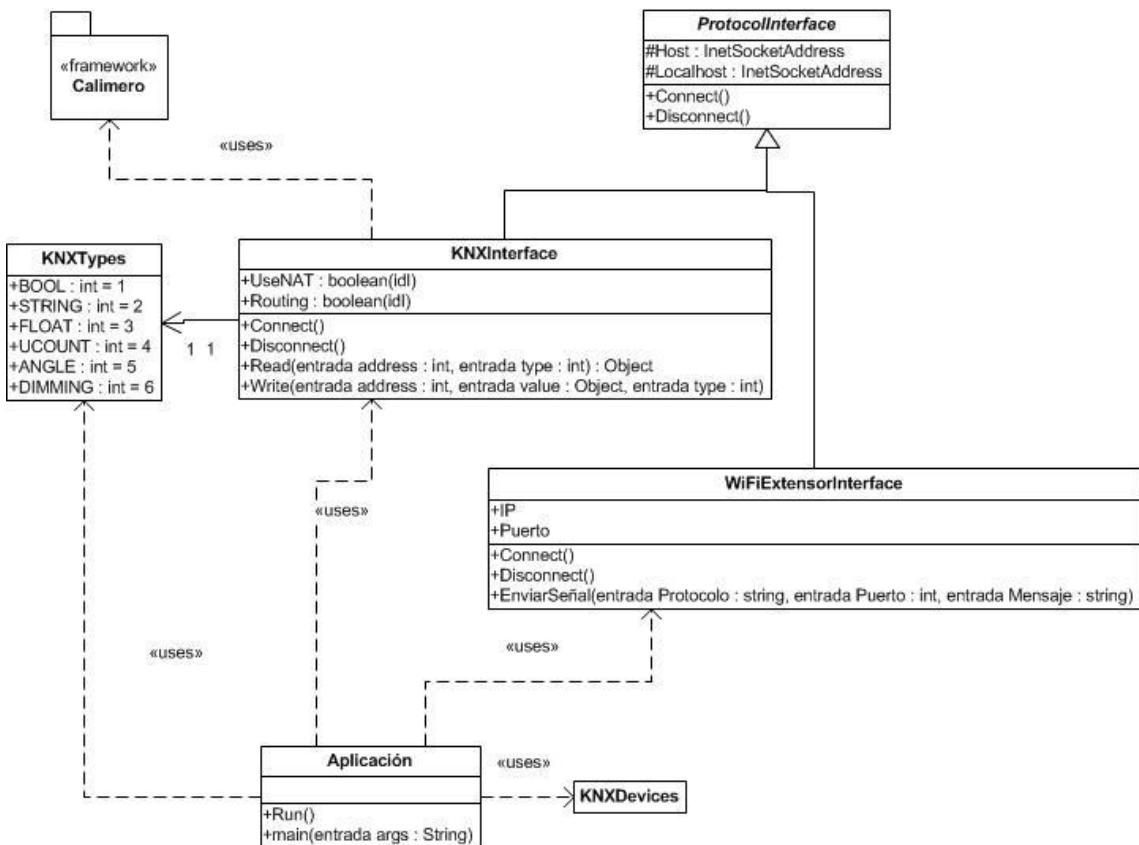


Figura 7.1: Esquema UML de la API actualizado

Para comunicarse con el extensor basta con abrir un servidor TCP a través de la función *Connect*, pero especificando antes la IP y el puerto con el que se ha configurado el extensor.

Una vez conectados tenemos la función *EnviarSeñal* que, especificando el protocolo, el puerto (o salida del extensor) y el mensaje a enviar, manda la señal al dispositivo.

8 Desarrollo de la aplicación de escritorio

A modo de ejemplo se desarrolló una aplicación de escritorio que, utilizando la API implementada, permite el control de los diferentes dispositivos de la casa de forma gráfica e intuitiva.

En este capítulo hago una breve explicación de su estructura y diseño, así como ejemplos de su funcionamiento.

8.1 Elección del entorno de desarrollo

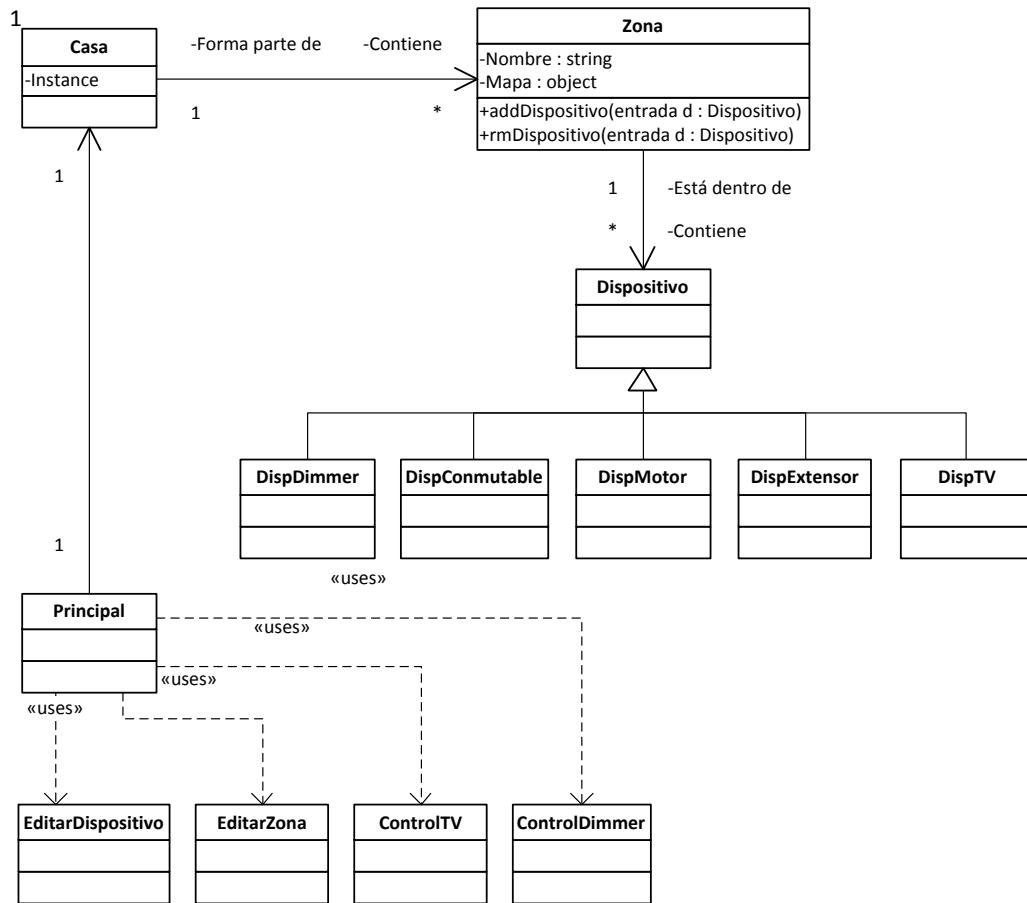
Barajé diferentes entornos de desarrollo para esta aplicación, uno de las primeras opciones fue utilizar Eclipse programando en Java, ya que la API la desarrollé en esa misma plataforma. Pero al final me decanté por utilizar Visual Studio 2010 de Microsoft programando en C#. Los motivos por los que elegí este entorno son:

- El CVI tiene un convenio con Microsoft para el desarrollo de aplicaciones con el reconocimiento de voz de Windows 7, por lo tanto iban a utilizar este sistema operativo en sus computadoras. Considero que Visual Studio es la mejor forma de crear aplicaciones para Windows de forma rápida
- Visual Studio permite desarrollar interfaces gráficas muy agradables para entornos de Windows. Dado que la aplicación, aunque esté desarrollada a modo de ejemplo, debe ser bonita y fácil de usar como para poder enseñarla a posibles clientes.
- C# es un lenguaje de programación parecido a Java y C++ y ya había trabajado antes con él, de esta manera no necesitaría aprender otro entorno de desarrollo y, en mi opinión, ofrece una gran libertad y comodidad para programar bajo Windows.

Dado que la API la programé en Java, podría parecer que esto sería un problema. Pero existe una fácil solución ya que, gracias a la API OpenJDK IKVM pude compilar mi API en una librería de Windows dll. Una vez hecho esto se puede importar cualquier dll al proyecto de Visual Studio junto con la dll de la API IKVM sin problemas y sin perder eficiencia.

8.2 Estructura

La estructura de esta aplicación ha sido pensada para permitir la posibilidad de añadir nuevos tipos de dispositivos actualmente no soportados.



8.3 La interfaz gráfica

Procedo a explicar la interfaz dividiéndola en zonas, tal y como se muestra en la Figura 8.1:

- **Barra de herramientas:** aquí se encuentran tres elementos:
 - **Archivo:** nos permite crear un nuevo proyecto, guardarlo o abrir un proyecto previamente guardado.
 - **Editar:** desde aquí es donde se pueden añadir o eliminar zonas de la casa a nuestro proyecto.
 - **Ayuda:** Como el nombre indica, aquí se puede consultar un pequeño manual de cómo usar el programa, así como información sobre quien lo ha desarrollado.
- **Selección de Zona:** En esta lista desplegable se puede elegir la zona de la casa que se desea ver en el mapa, así como todos los dispositivos que están presentes.
- **Mapa:** aquí se muestra la zona de la casa que se ha seleccionado en la lista de selección de zona.
- **Botón de bloqueo/desbloqueo del drag&drop:** el sistema drag&drop (arrastrar y soltar en español) consiste en mover los objetos del mapa con el ratón para situarlos allá donde se quiera. Con este botón se puede bloquear y desbloquear esta funcionalidad.



Figura 8.1: Partes de la interfaz

A continuación se explica cómo es el funcionamiento de esta aplicación utilizando como ejemplo la misma casa domótica del CVI.

En el menú editar se puede acceder a Zonas, donde se puede añadir, modificar o eliminar una zona.

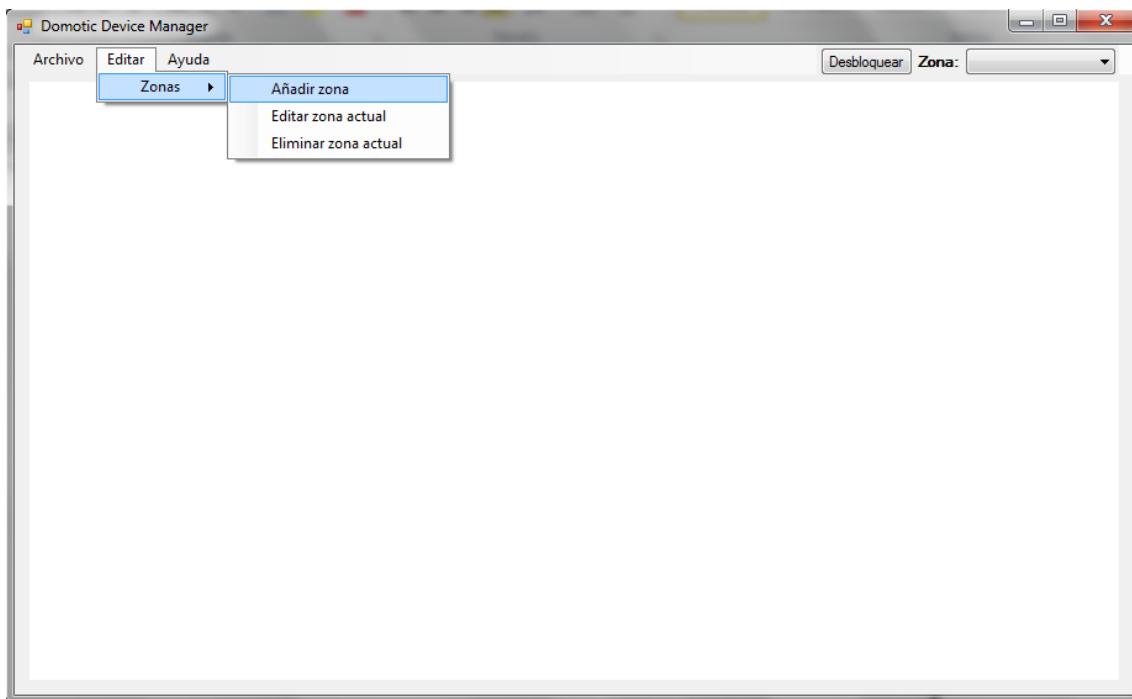


Figura 8.2: Interfaz gráfica, menú editar

Si seleccionamos *añadir una zona*, nos saldrá la siguiente pantalla donde deberemos introducir un nombre y una imagen correspondiente al plano de la misma.

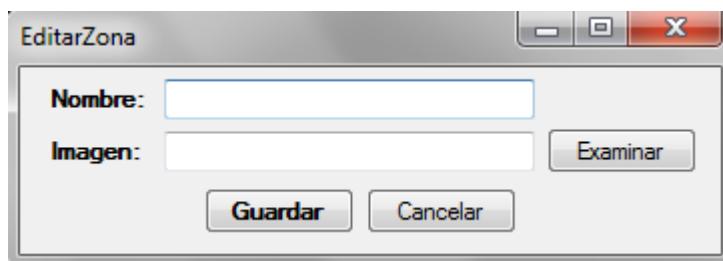


Figura 8.3: Editar zona

Una vez añadidas todas las zonas que queramos, éstas nos aparecerán en la pestaña de la esquina superior derecha, donde seleccionando sobre una de ellas nos saldrá en el mapa.

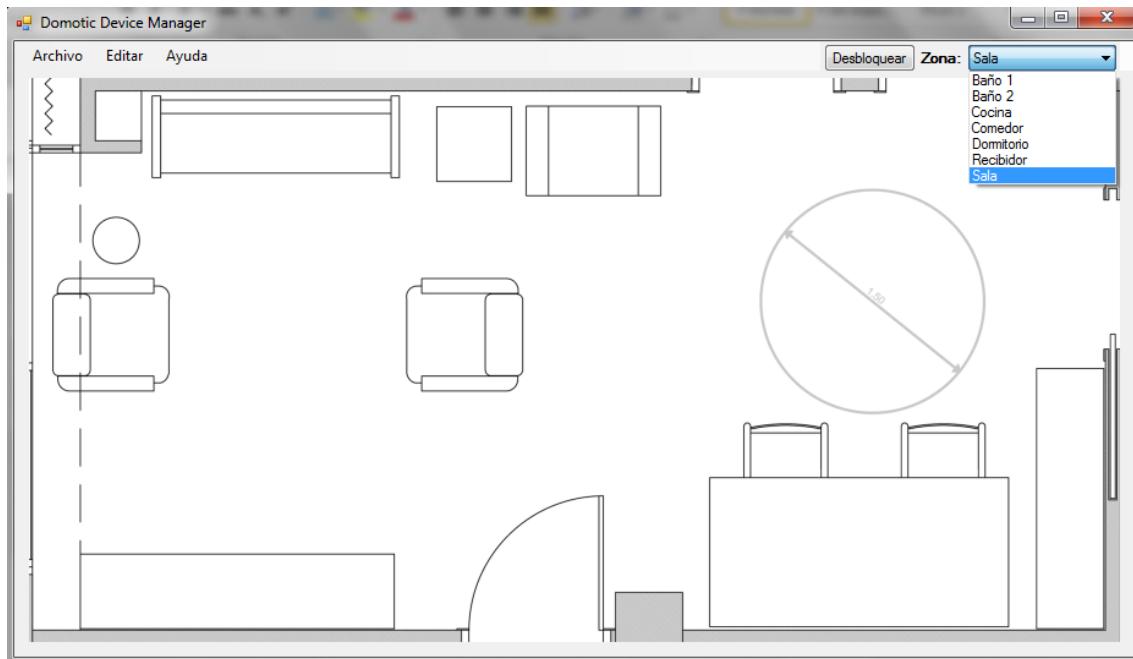


Figura 8.4: Menú desplegable de zonas

Ahora ya podemos añadir dispositivos en el mapa, para ello clicamos con el botón derecho -> Añadir dispositivo para que nos aparezca un formulario como el siguiente:

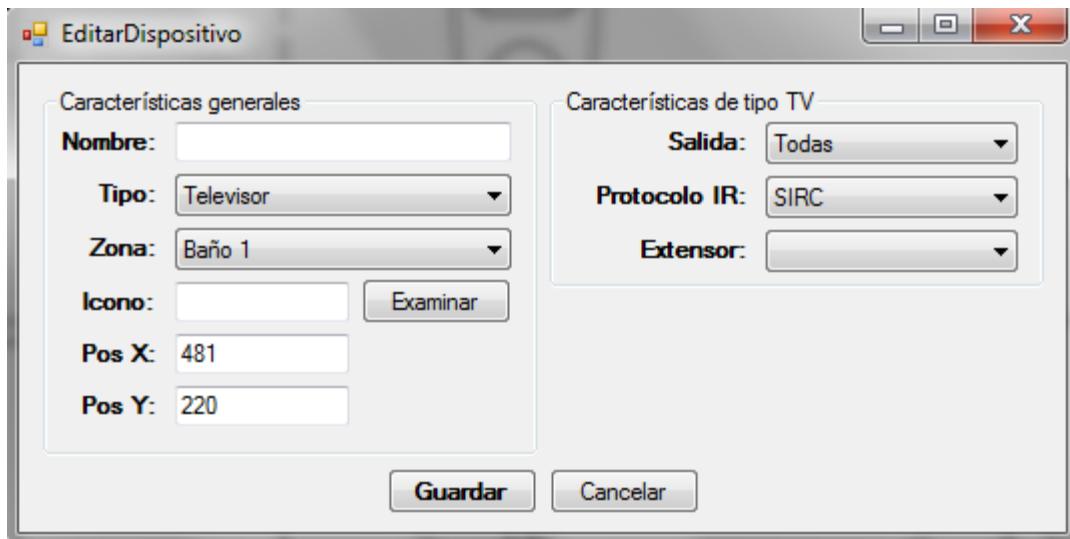


Figura 8.5: Editar dispositivo

Es obligatorio indicar un nombre al dispositivo, así como el tipo, la zona la imagen a mostrar y las coordenadas X e Y (estas se ponen automáticamente en función de donde se ha hecho clic en el mapa). En el desplegable Tipo seleccionamos el tipo de dispositivo que queremos añadir, por ahora existen los siguientes tipos:

- **Conmutable:** dispositivos, como ahora las luces, que pueden activarse y desactivarse con una única orden a modo interruptor.

- **Motor:** Los motores funcionan de forma diferente, tienen una orden para moverse o pararse y otra para elegir el sentido. Existen otras formas de controlar un motor, pero en el CVI solo tienen esta implementada.
- **Dimmer:** Se trata de dispositivos capaces de regularse en intensidad. Se parecen a los de tipo conmutable pero poseen una orden para elegir un valor entre 0 y 255. Un ejemplo serían las luces regulables, el CVI posee una en el salón.
- **Televisor:** No hace falta mucha explicación el respecto. Para añadir uno hay que indicar que extensor se va a utilizar (tiene que existir como mínimo uno), la salida por donde está conectado y el protocolo IR que utiliza.

Existen otros dispositivos multimedia que no se han añadido a la lista, como reproductores de DVD, cadenas de música, etc. Esto es así por falta de tiempo, ya que cada dispositivo multimedia posee su propia pantalla de control, como se verá al final del capítulo.

En la siguiente imagen se puede apreciar cómo queda una zona con diferentes dispositivos añadidos.

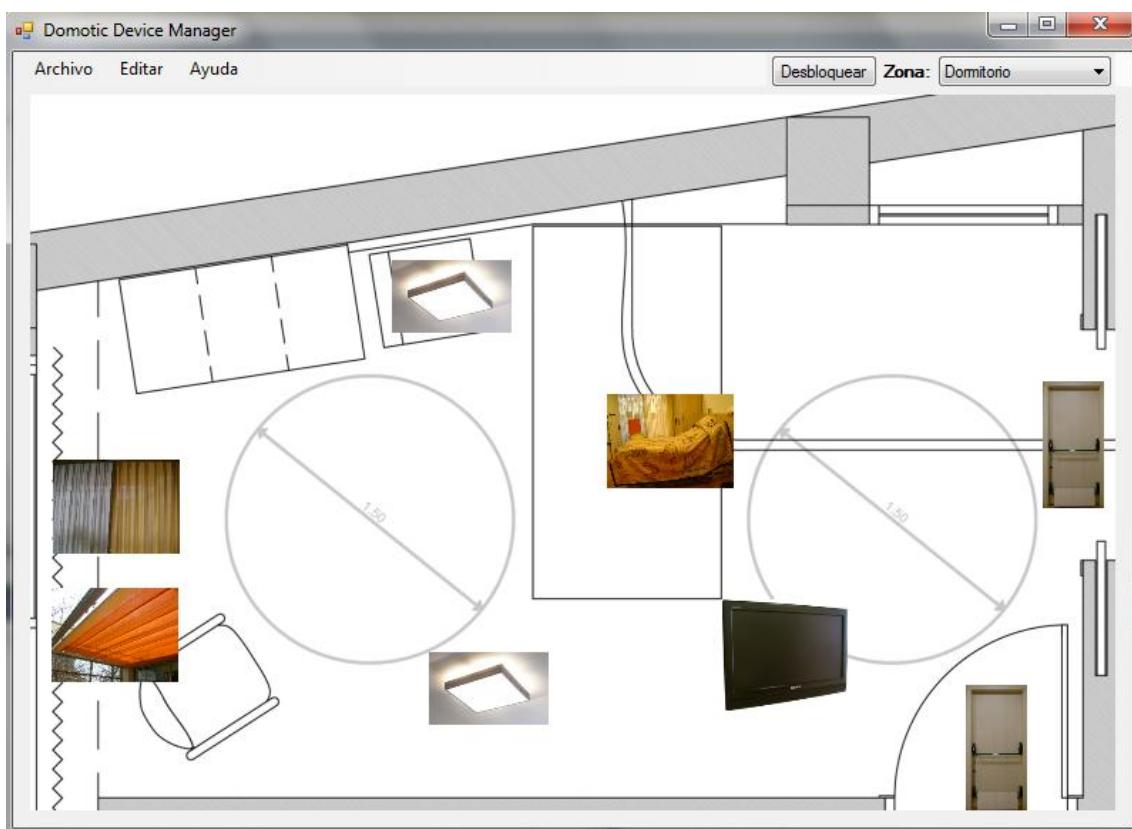


Figura 8.6: Zona con dispositivos en el mapa

Otra funcionalidad que he implementado en el programa es la posibilidad de mover los dispositivos por el mapa en modo drag&drop. Para activar esta funcionalidad hay que presionar el botón “desbloquear” situado al lado del desplegable de zonas.

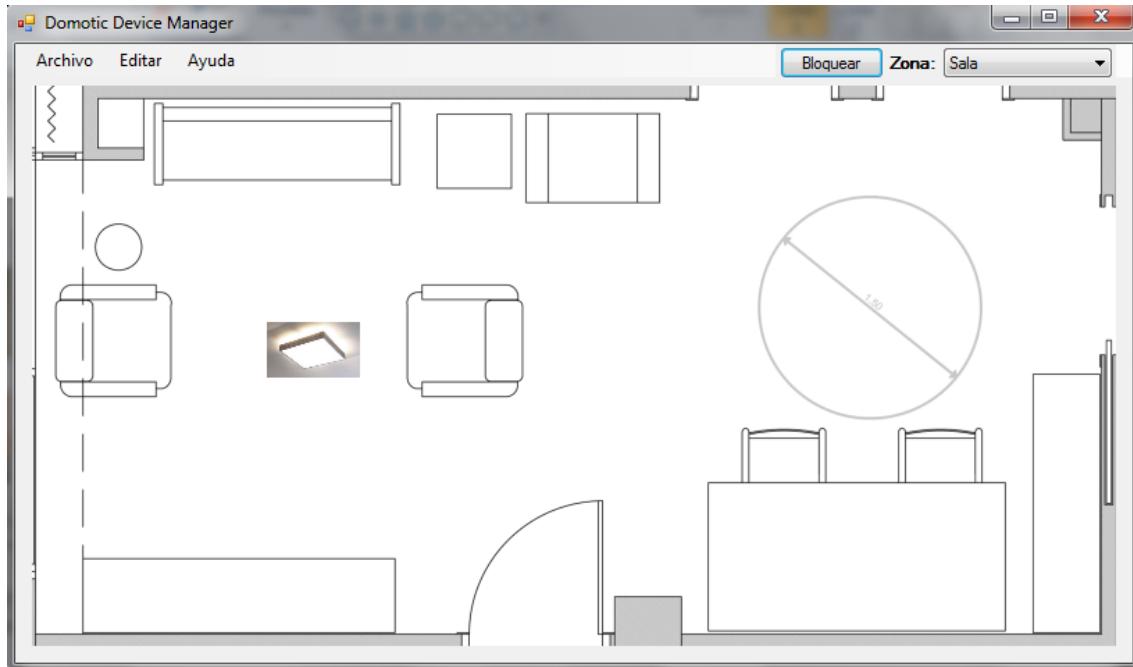


Figura 8.7: Drag&drop

Una vez presionado el botón cambiará de nombre y se llamará “bloquear”. Ahora podemos arrastrar cualquier dispositivo por el mapa con toda libertad (desgraciadamente en la imagen no puede apreciarse el movimiento, pero sí como cambia el botón). Si queremos bloquear el mapa de nuevo le damos otra vez al mismo botón.

Para encender una luz o abrir una puerta basta con hacer clic sobre la imagen para que se ejecute la acción.

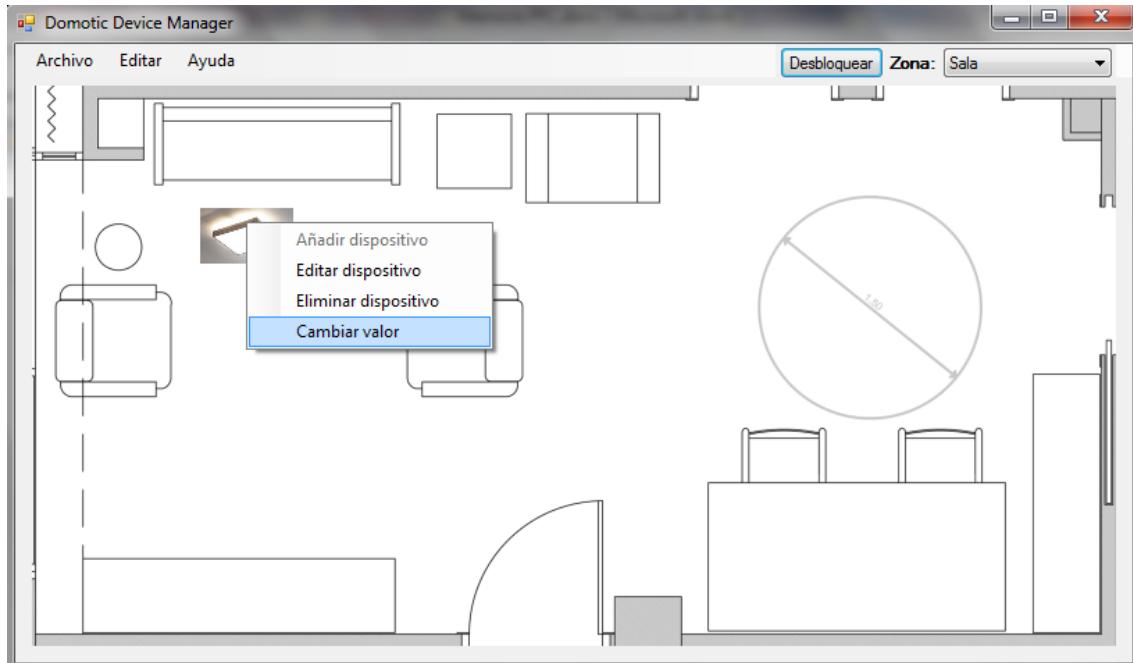


Figura 8.8: Menú desplegable

Para dispositivos de tipo dimmer, además de encenderse y apagarse, como se pueden regular disponen de un control auxiliar que se activa haciendo clic derecho sobre el dispositivo y seleccionando “Cambiar valor”. Haciendo esto aparecerá el control mostrado en la imagen siguiente:

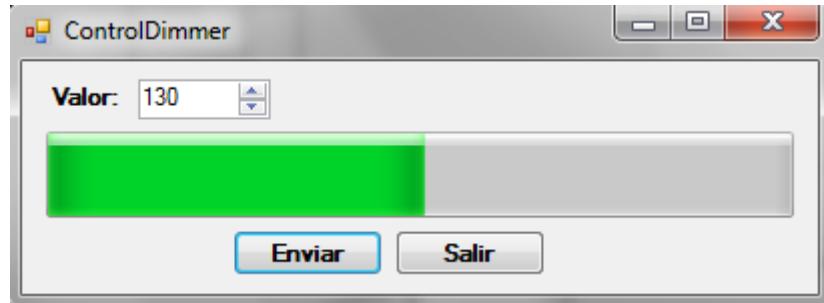


Figura 8.9: Control Dimmer

Para los dispositivos de tipo Televisor, si hacemos clic normal sobre ellos nos aparecerá directamente un control como el de la figura:

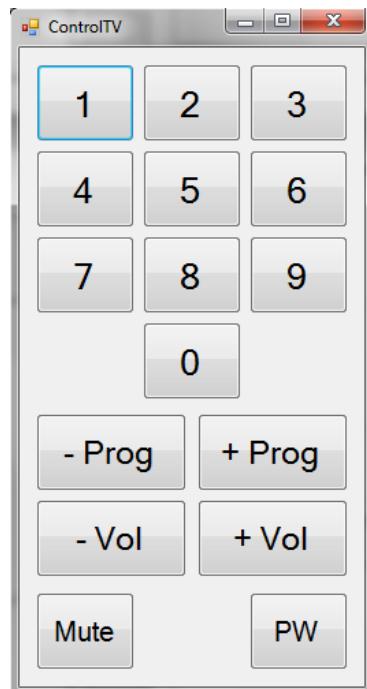


Figura 8.10: Control TV

Este control nos permite ejecutar las acciones básicas que permite cualquier televisor normal.

9 Desarrollo de la aplicación móvil

También -a modo de ejemplo- se desarrolló una aplicación móvil para dispositivos Android que permitiera, igual que la aplicación de escritorio, el control de los dispositivos KNX de la casa, pero con una interfaz mucho más simple.

Igual que en el capítulo anterior, en éste explico por qué se eligió el entorno de desarrollo, cual es la estructura de la aplicación y como es la interfaz gráfica.

9.1 Elección del entorno de desarrollo

Primero de todo se eligió desarrollar una aplicación para Android y no para cualquier otro teléfono inteligente por los siguientes motivos:

- Contrariamente a iPhone, Android permite crear y distribuir libremente cualquier aplicación sin necesidad de usar el Android Market. Lo que nos ahorra pedir permisos a Google y da mucha facilidad a la hora de desarrollar aplicaciones.
- El lenguaje de programación es Java, lo que me permitió utilizar la API sin necesidad de hacer ningún cambio.
- Ya había programado con anterioridad aplicaciones para Android, lo que me permitió ahorrarme el aprendizaje del nuevo sistema.
- El entorno de desarrollo para aplicaciones Android es Eclipse, el mismo que he utilizado para el desarrollo de la API.

9.2 Estructura

Durante el desarrollo me encontré con un problema que me impidió continuar con la implementación de la aplicación móvil. El problema fue que algunos dispositivos Android, especialmente los que utilizan la versión 1.6 del sistema operativo, no son capaces de crear la conexión con el servidor de KNX. Este problema me llevó de cabeza durante bastante tiempo, pero conseguí encontrar una solución práctica: Crear un servidor TCP en una máquina local y que fuera este el que mantuviera la comunicación con KNX. De esta forma independizaba completamente la aplicación móvil de la API.

Como en un futuro, y como se explicará en el próximo capítulo, la aplicación funcionará a través del servidor DLA la idea de colocar un servidor intermedio no supone un inconveniente más.

A continuación se muestra el esquema móvil-servidor-API:

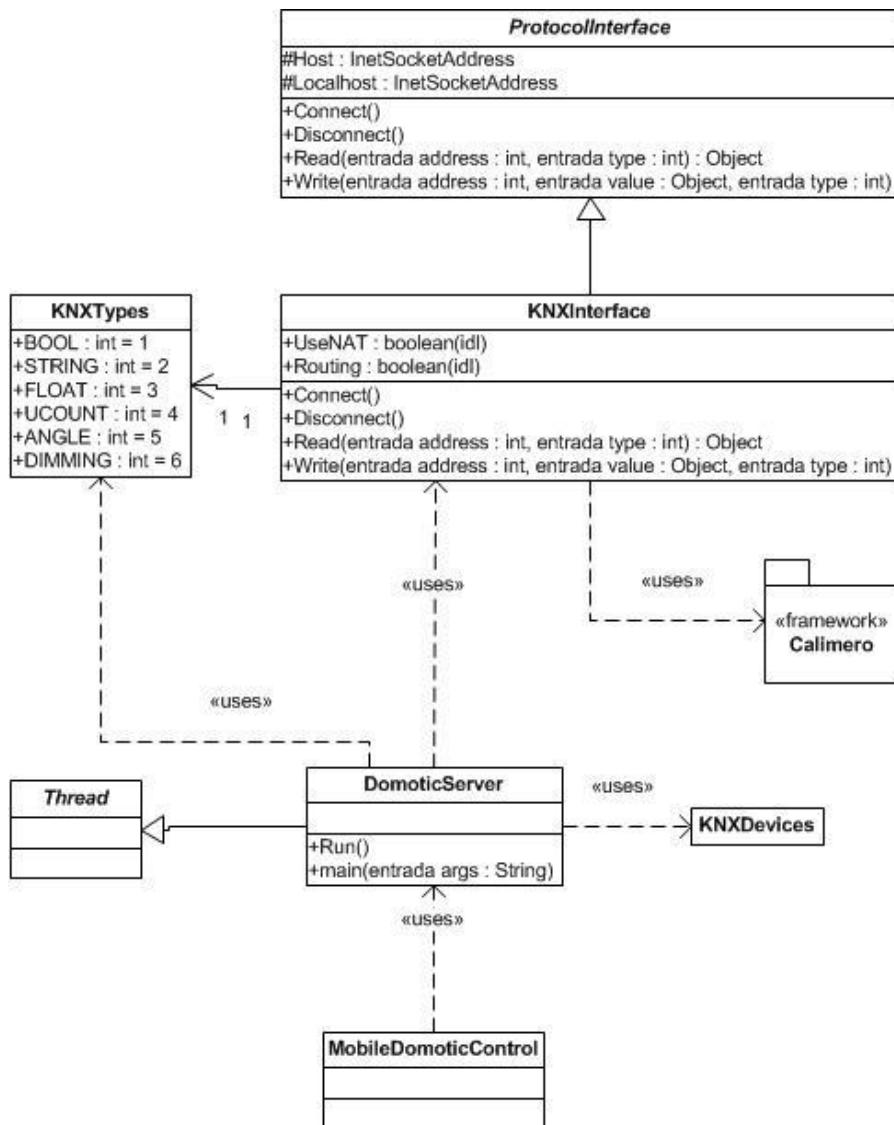


Figura 9.1: Esquema Móvil-Servidor-API

La aplicación móvil es muy sencilla y no ha sido necesaria la creación de una estructura de clases tan compleja como la aplicación de escritorio explicada anteriormente.

9.3 La interfaz Gráfica

La interfaz gráfica, como ya se ha comentado, es muy sencilla y diseñada para poner a prueba todo el sistema. Dado que esta aplicación es para uso de pruebas y no ha sido desarrollada para un uso comercial, no he pretendido que la interfaz gráfica sea bonita, sino funcional. La siguiente figura corresponde a la pantalla principal de la aplicación:



Figura 9.2: Pantalla principal de la aplicación Android

En el primer menú desplegable que encontramos arriba de todo podemos elegir el tipo de dispositivo que queremos controlar, tales como luces, persianas, puertas, etc. Esto se ha hecho así para no abrumar al usuario con tantos dispositivos diferentes, ya que en el CVI existe una gran cantidad de ellos.



Figura 9.3: Menú desplegable para la selección del tipo de dispositivo

En el segundo menú desplegable es donde se elige el dispositivo, dentro del grupo seleccionado, que queremos controlar. Los nombres de los dispositivos son los que la empresa que montó el sistema puso en la base de datos, y para no hacer más difícil la identificación de éstos se han dejado tal cual.



Figura 9.4: Menú desplegable para la selección del dispositivo

Una vez elegido el dispositivo elegimos la acción que queremos ejecutar. Para esta aplicación solo se han implementado dos acciones: activar y desactivar. En caso de las luces sería encender/apagar y en caso de las puertas sería abrir/cerrar, pero por simplicidad y generalización se ha puesto este texto. En la siguiente imagen se puede ver la aplicación preparada para apagar las luces LED del suelo del receptor:



Figura 9.5: Envío de la orden solicitada

10 Adaptación de la API para su funcionamiento con la librería DLA

El departamento de LSI está desarrollando una aplicación con interfaces inteligentes para el control de los dispositivos del CVI utilizando la API implementada. El problema es que quieren que su aplicación sea completamente independiente de la tecnología con la que he desarrollado la API. Por eso mismo se ha optado por utilizar una librería llamada DLA que permiten ese nivel de independencia entre sistemas.

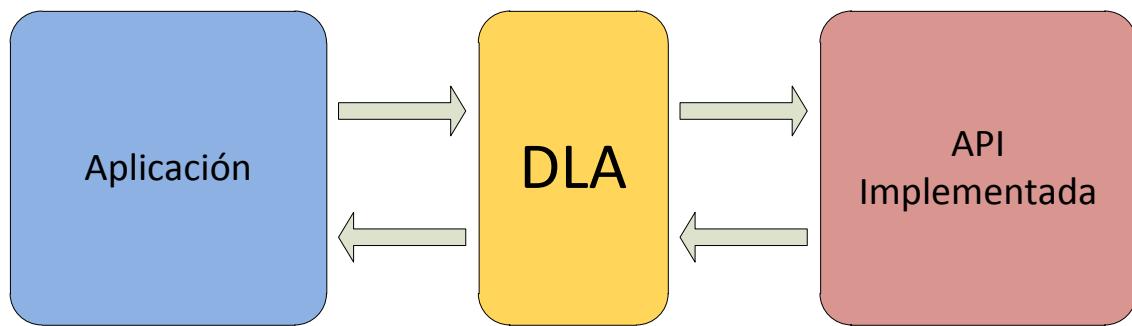


Figura 10.1: Representación de la independencia entre la aplicación y la API con el DLA

En este capítulo expongo en qué consiste dicha librería y cómo funcionan conjuntamente los dos sistemas.

10.1 Definición del DLA

La arquitectura *DLA* se utiliza para implementar sistemas cooperativos que necesiten la interacción de distintos módulos distribuidos libremente en distintas máquinas. Para ello, implementa un modelo de memoria compartida distribuida, donde existe un elemento central que gestiona el almacenamiento y acceso a todos los datos que se quieren compartir en el sistema.

Dispone de dos elementos básicos para facilitar la interacción entre módulos:

- **Conexiones:** permiten almacenar todos aquellos datos que los distintos módulos quieren compartir entre sí.
- **Buzones:** permiten sincronizar la actividad de los distintos módulos entre sí.

Existen así mismo dos esquemas distintos de la arquitectura *DLA*:

- **Esquema distribuido:** permite la interacción entre módulos ubicados en distintas máquinas, a través de un servidor central conocido como *DLAServer*.
- **Esquema local:** permite la interacción entre módulos ubicados en la misma máquina, a través de mecanismos de memoria compartida.

Ambos esquemas pueden por supuesto ser combinados, utilizando el esquema distribuido para gestionar la interacción entre los módulos ubicados en distintas máquinas y el esquema local para gestionar la interacción entre los módulos ubicados en la misma máquina.

10.2 Funcionamiento del DLA

En la figura que se muestra a continuación se pueden apreciar los distintos elementos que componen la arquitectura de control *DLA*.

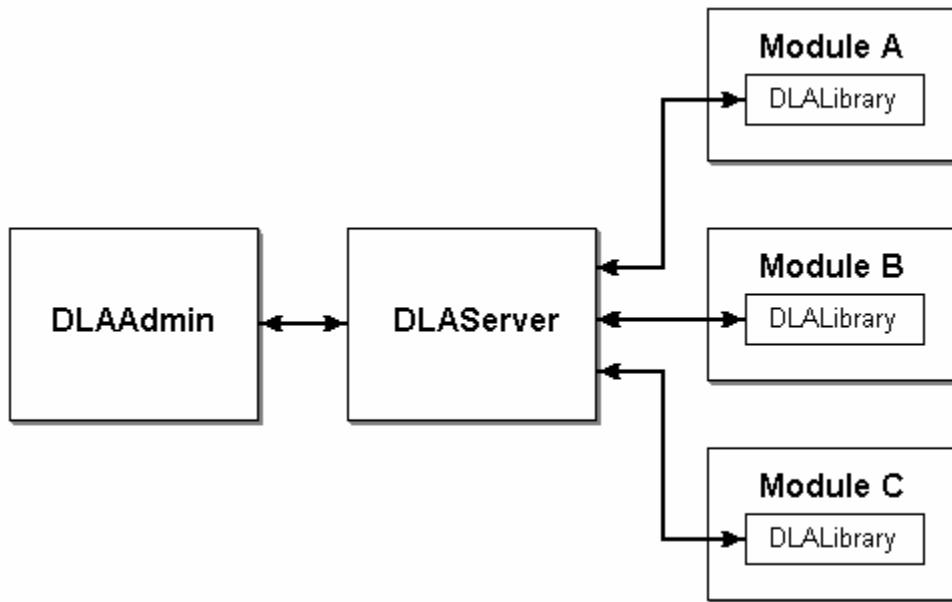


Figura 10.2: Elementos que componen la arquitectura de control DLA

La utilización de estos elementos es la que se describe a continuación:

- **DLAServer:** el servidor central *DLAServer* gestiona la creación, lectura y escritura de las conexiones y buzones necesarios en el sistema.
- **DLALibrary:** la librería *DLALibrary* implementa todas las funciones necesarias para que cualquier módulo se conecte con el servidor central *DLAServer* y utilice las conexiones y buzones que precise.
- **DLAAdmin:** el administrador remoto *DLAAdmin* permite monitorizar y modificar en tiempo real la operación del servidor central *DLAServer*, constituyendo una herramienta muy potente para analizar el flujo de intercambio de datos existente en el sistema y facilitar así la depuración del mismo.

Para pasar datos entre el *DLAServer* y los diferentes módulos que existan deben especificarse los diferentes tipos de conexión que van a existir y que datos se enviarán por cada uno. A continuación se especifican las diferentes consultas implementadas. Para más información sobre cómo funciona el DLA consultad el

ANEXO VI: Funciones del DLA.

10.3 Consultas implementadas

Solo se ha considerado oportuno crear tres consultas. Una para ejecutar acciones sobre los dispositivos, otra para realizar una consulta del estado actual del dispositivo y otra para descargar el archivo XML con la lista de dispositivos y sus posibles acciones.

10.3.1 WriteDomoticControl

Esta estructura define el protocolo para ejecutar una orden de escritura específica en el dispositivo indicado.

```
struct WriteDomoticControl {  
    int id_mecanisme;  
    int id_comanda;  
    int res;  
    byte ack;  
}
```

- **id_mecanisme:** Identificador del dispositivo al que se quiere acceder.
- **Id_comanda:** Identificador de la orden que se quiere ejecutar.
- **res:** Resultado de la operación. Cero si es todo correcto, menor que cero si ha habido algún error (indicando cual con el número).
- **ack:** Cero si todavía no se ha acabado de ejecutar la orden, uno cuando ya se ha ejecutado y el resultado está disponible.

10.3.2 ReadStateDomoticControl

Esta estructura define el protocolo para ejecutar una orden de lectura del estado o valor actual del dispositivo indicado.

```
struct ReadStateDomoticControl {  
    int id_mecanisme;  
    int res;  
    byte ack;  
}
```

- **id_mecanisme:** Identificador del dispositivo que se quiere consultar.
- **res:** Resultado de la consulta. Si es menor que cero si ha habido algún error (indicando cual con el número).
- **ack:** Cero si todavía no se ha acabado de ejecutar la consulta, uno cuando ya se ha ejecutado y el resultado está disponible.

10.3.3 ReadDevicesXML

Esta estructura define el protocolo para recuperar el archivo XML donde se especifican todos los dispositivos accesibles y las diferentes órdenes posibles para cada uno. Dado que el tamaño del XML no está delimitado hay que hacer tantas consultas como requiera el fichero.

```

struct ReadDevicesXML {
    byte[1000] xml;
    byte begin;
    byte next;
    byte end;
}

```

- **xml:** 1000 bytes siguientes del archivo XML.
- **begin:** indica que queremos iniciar la consulta del XML desde el principio.
- **next:** Cero si no quiero más información, uno si quiero que me prepare el siguiente fragmento.
- **end:** indica si hemos acabado de leer el XML, si no es así se vuelve a llamar para obtener el siguiente fragmento.

A continuación se explica cómo es el formato de este XML.

10.4 XML con los dispositivos accesibles

Conjuntamente con el departamento de LSI, que se encargan de implementar la interfaz de la aplicación, definimos como debería ser el XML con los dispositivos de la casa y sus propiedades y operaciones permitidas. El código comentado que se muestra aquí abajo es el esquema que debe seguir el XML.

```

<devices>
    <errors> // Errores generales
        <error code="código error" name="Nombre error" />
    </errors>
    // Por cada tipo de dispositivo que exista se añade un tag devices
    <device type="Tipo de dispositivo" readable="true/false">
        // A continuación se especifican las propiedades de este tipo de
        // dispositivos
        <errors> // Errores específicos para este tipo de dispositivo
            <error code="código error" name="Nombre error" />
        </errors>
        <commands>
            // Comandos u órdenes que aceptan este tipo de dispositivos
            <command code="código comando" name="nombre comando"/>
        </commands>
        <mechanisms>
            //AQUÍ se introducen todos los dispositivos de este tipo
            <mechanism code="código dispositivo" name="nombre disp." />
        </mechanisms>
    </device>
</device>

```

Para tener una mejor idea de cómo quedaría este XML con datos reales se adjunta a continuación un ejemplo con un par de tipos de dispositivos y algunas propiedades reales:

```
<devices>
    <errors>
        <error code="1" name="Dispositivo inexistente" />
        <error code="2" name="Comando inexistente" />
    </errors>
    <device type="Llums" readable="true">
        <errors>
            <error code="3" name="Error servidor" />
        </errors>
        <commands>
            <command code="0" name="Apagar"/>
            <command code="1" name="Encender"/>

        </commands>
        <mechanisms>
            <mechanism code="0" name="comedor"/>
            <mechanism code="1" name="cocina"/>
            <mechanism code="2" name="cuarto de baño"/>
            <mechanism code="3" name="dormitorio"/>
            <mechanism code="4" name="recibidor"/>
            <mechanism code="5" name="sala de estar"/>
        </mechanisms>
    </device>
    <device type="Televisor" readable="false">
        <errors>
            <error code="4" name="No se encuentra extensor WiFi" />
        </errors>
        <commands>
            <command code="0" name="Apagar"/>
            <command code="1" name="Encender"/>
            <command code="2" name="Aumentar canal"/>
            <command code="3" name="Disminuir canal"/>
            <command code="4" name="Aumentar volumen"/>
            <command code="5" name="Disminuir volumen"/>
        </commands>
        <mechanisms>
            <mechanism code="6" name="cocina"/>
            <mechanism code="7" name="dormitorio"/>
            <mechanism code="8" name="sala de estar"/>
        </mechanisms>
    </device>
</devices>
```

11 Evaluación de resultados

Gracias a mi trabajo se ha podido aumentar el control de los dispositivos presentes en el CVI desde una misma aplicación. La siguiente pirámide representa de una forma clara las diferentes capas de control desde los dispositivos hardware hasta la aplicación final:

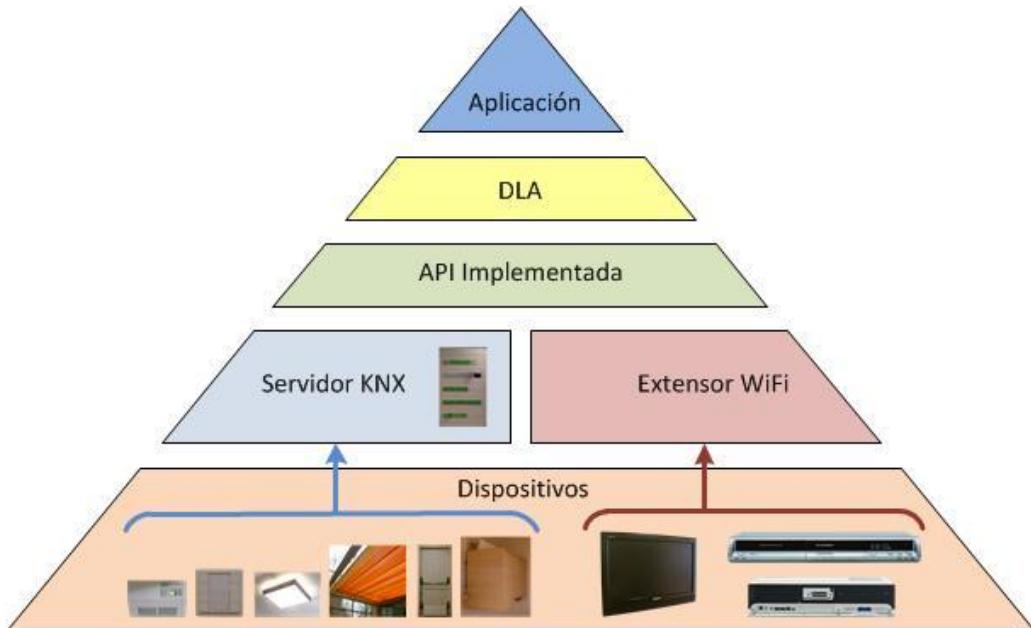


Figura 11.1: Pirámide del control del CVI

Las pruebas realizadas en el CVI han sido un éxito y han demostrado el correcto funcionamiento, tanto de la API como de los programas que he implementado. A continuación se muestran una serie de imágenes del funcionamiento del programa de escritorio desde un Tablet PC por la casa domótica del CVI.

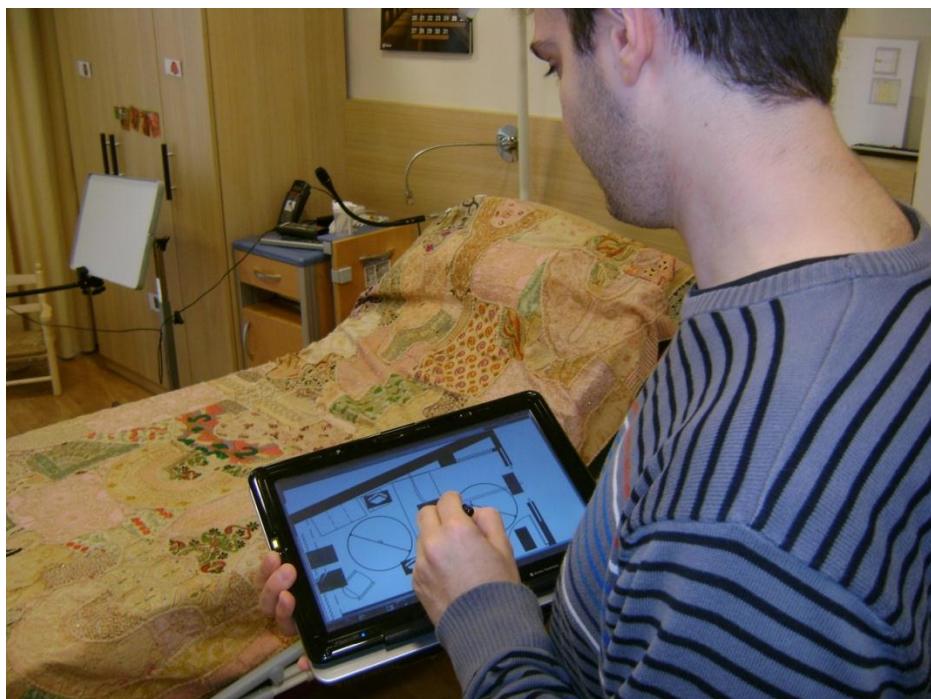


Figura 11.2: Ejemplo de funcionamiento del control de la cama

En la imagen anterior estoy controlando, mediante un simple clic en la pantalla, la inclinación del reposacabezas de la cama. En la imagen siguiente aparezco controlando la intensidad de una lámpara de tipo Dimmer. En ambos casos se trata de dispositivos KNX.

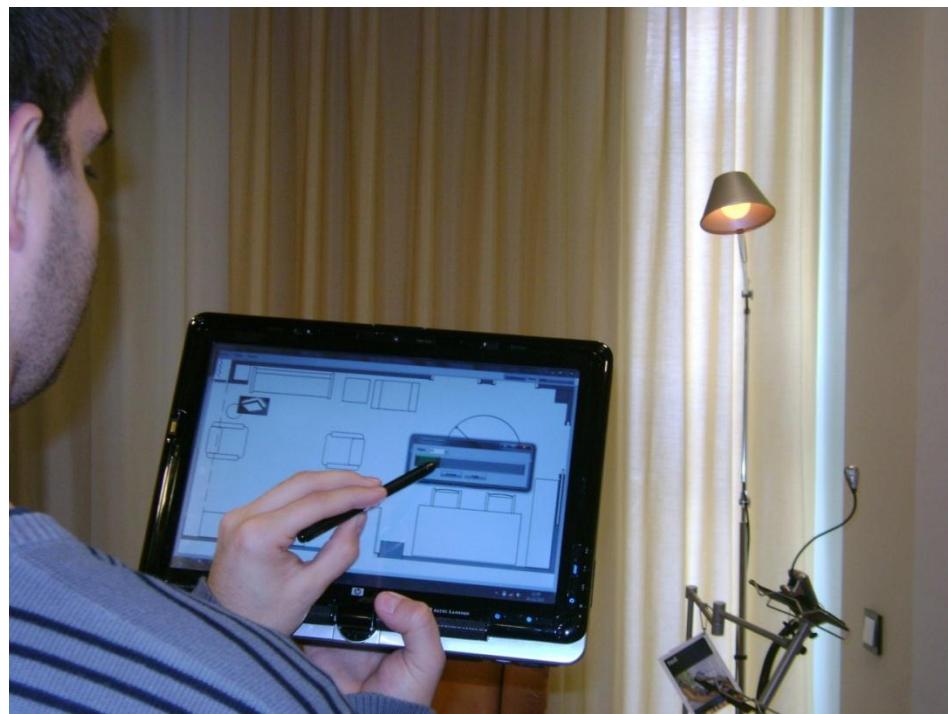


Figura 11.3: Ejemplo de funcionamiento del control de una luz de tipo Dimmer

En la siguiente imagen aparezco aumentando el volumen del televisor. Puede apreciarse como en el televisor aparece la barra de volumen, a la vez que se enciende los LEDs de conexión establecida y señal enviada del extensor.



Figura 11.4: Ejemplo de funcionamiento del control de la televisión

12 Valoración económica

En este capítulo hago la valoración económica de este PFC, incluyendo tanto el coste de fabricación del prototipo como el coste de desarrollo de todo el proyecto.

12.1 Coste de la fabricación del prototipo

La siguiente tabla contiene una lista de todos los componentes utilizados para el montaje del prototipo, así como información del proveedor y el precio por unidad.

Nombre	Proveedor	Código	Descripción	Unidades	Precio un.	Total
CONECTOR MINIJACK 3.5MM MACHO	Farnell	1243261	LUMBERG - KLS 2 - PLUG, 3.5MM JACK, MONO	4	0,54 €	0,00 €
CONECTOR MINIJACK 3.5MM HEMBRA	Farnell	1270966	LUMBERG - 1502 03 - SOCKET, 3.5MM JACK, CHASSIS	8	0,77 €	6,16 €
LED VERDE 3mm	Farnell	1581130	MULTICOMP - MCL514GD - LED, 3MM, 70°, GREEN	3	0,17	0,51 €
LED MULTICOLOR 3mm	Farnell	1581465		2	0,40	0,80 €
INTERRUPTOR 2 POSICIONES DE PANEL	Farnell	9473378	MULTICOMP - 1MS1T1B5M1QE - SWITCH, SPDT	1	1,22 €	1,22 €
CONECTOR D'ALIMENTACIÓN	RS	705-1538	IP68 2.5mm PCB DC power socket	1	8,90 €	8,90 €
RESISTENCIA 50 OHMS 1/4W	Farnell	770760	MULTICOMP - MCF RESISTENCIA, 0,25W 5% 16K	4	0,00 €	0,01 €
CAJA DE PLÁSTICO				1	3 €	3,00 €
RESISTÈNCIA 10K OHMS 1/4W	Farnell	1652662	VISHAY DALE - RN60C1002B - METAL FILM RESISTOR	4	0,30 €	1,20 €
CONECTOR 5 PINS MACHO	Farnell	588600	TYCO ELECTRONICS / AMP - 640456-5 - 0.1", 5WAY	1	0,56 €	0,56 €
RESISTENCIA 220 OHMS 1/4W	Farnell	9341528	MULTICOMP - MF25 220R - RESISTENCIA, 0,25W 1%	7	0,04 €	0,25 €
FlyPort WI-FI	EIKON			1	49 €	49,00 €
Gastos de envío	Farnell					9,00 €
Gastos de envío	EIKON					24,00 €
TOTAL = 104,61 €						
TOTAL + IVA 18% = 123,44 €						

Figura 12.1: Tabla de precios de los componentes

El precio total con IVA ha resultado ser de 123,44 €, un precio elevado para ser comercializado pero teniendo en cuenta que es un prototipo a resultado bastante económico.

12.2 Coste de desarrollo del PFC

Para calcular el coste de este proyecto se han estimado las siguientes horas trabajadas, dividiéndolas en horas dedicadas en el estudio de las tecnologías existentes en el CVI, en la creación de la API, en la creación del extensor y en la documentación.

	Horas	Precio €/Hora	Total €
Estudio previo sobre las tecnologías en el CVI	50	30	1500
Análisis, Diseño e Implementación de la API	240	30	7200
Análisis, Diseño e Implementación del extensor	400	30	12000
Documentación	150	20	3000
Total			23700

Figura 12.2: Tabla del coste para la realización de este proyecto

Entonces, el coste de mi tiempo invertido en este proyecto ha sido de 23.700€.

13 Planificación seguida

El proyecto ha sido llevado a cabo en una gran variedad de etapas consecutivas pero, dada la naturaleza del proyecto (estudio de tecnologías y desarrollo de diferentes elementos) ha sido más que necesaria la realización de ciertas tareas de forma paralela, ya que el desarrollo de una puede condicionar a las anteriores. A continuación se muestra el diagrama de Gantt:

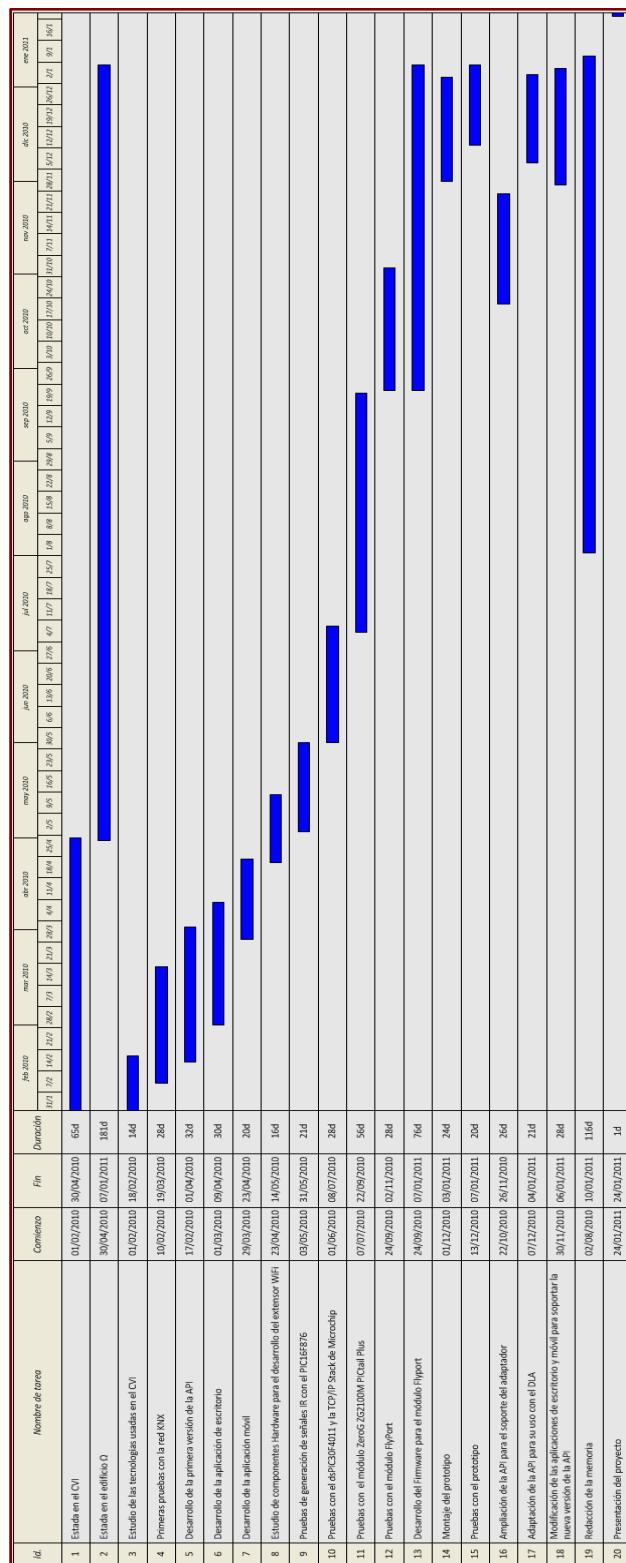


Figura 13.1: Diagrama de Gantt de la planificación seguida

14 Posibles ampliaciones o mejoras

A lo largo del desarrollo del proyecto han ido saliendo algunas ideas y necesidades de cara a comercializar el resultado de mi trabajo. A continuación se enumeran algunas ampliaciones o mejoras propuestas:

- De cara a convertir el extensor inalámbrico WiFi en un producto comercial ya se está preparando el diseño de una placa para reemplazar el módulo FlyPort por uno propio específico para su función, lo que permitiría reducir considerablemente el tamaño de la caja y abaratar costes.
- Actualmente la configuración para redes inalámbricas del extensor se debe introducir junto con el firmware del controlador, lo que supone un obstáculo a la hora de convertirlo en un producto comercial, ya que cada casa tendrá su propia configuración WiFi. Para solucionar este inconveniente, se tiene pensado crear una página web de configuración que alojará el propio extensor y permitirá al usuario configurar la red a su gusto con facilidad.
- También, para dar una mayor facilidad al usuario a la hora de configurar las aplicaciones que accedan a la API, se pretende añadir una base de datos actual de los diferentes modelos de aparatos multimedia que existen y el protocolo que utiliza cada uno. Esto me enfrenta al reto de encontrar dicha información, que hoy por hoy no he podido localizar.
- Dado que la tecnología domótica de Fagor dispone de un módulo IP con salida a internet sería posible ampliar la API para que pueda funcionar con toda la gama de dispositivos Fagor. Esto ampliaría notablemente los dispositivos que se podrían controlar desde una misma aplicación.

15 Anexos

ANEXO I: Manual para la utilización de la API

Para interactuar con la red KNX utilizando la API implementada hay que hacer los siguientes pasos:

Iniciar una instancia de la clase KNXAdapter indicándole un nombre identificativo (opcional) y la dirección IP del servidor KNX. Los demás parámetros son opcionales, el puerto por defecto de KNX es 3671.

```
KNXAdapter cvi = new KNXAdapter("CVI", "192.168.1.150", 3671, null, null);
```

Si se quiere utilizar network address translation (NAT) hay que poner la siguiente variable a true. El parámetro será ignorado en modo Routing.

```
cvi.UseNAT = true;
```

La siguiente variable determina el modo de comunicación a abrir. Si su valor es true se utilizará Routing, en caso contrario Tunneling. Por defecto su valor es false.

```
cvi.Routing = false;
```

Si se quiere establecer un tiempo de espera máximo específico para la respuesta se puede especificar con el siguiente método:

```
cvi.setTimeout(1);
```

KNX permite crear redes con diferentes tipos de tecnologías o medios. Estos son TP0, TP1, P110, P132 y RF (radiofrecuencia). Por defecto el medio siempre es TP1, pero si se quiere especificar otro se puede hacer con el siguiente método:

```
cvi.setMedium("TP0");
```

Ahora ya podemos establecer la conexión con el servidor KNX.

```
cvi.Connect();
```

Supongamos que queremos leer un entero positivo (o su equivalente en KNX, ucount), debemos especificar la dirección de la variable y el tipo que se espera leer. Este sería un ejemplo:

```
int res = (Integer) cvi.Read(1040,KNXAdapter.KNXTypes.UCOUNT);
```

El resultado es un objeto de la clase Object, en función del tipo especificado se debe hacer un cast u otro.

Estos son los tipos KNX y sus equivalentes en Java:

```
KNXTypes.BOOL: Boolean.  
KNXTypes.FLOAT: Float.  
KNXTypes.STRING: String.  
KNXTypes.UCOUNT: Integer.  
KNXTypes.ANGLE: Integer.  
KNXTypes.SWITCH: Integer.
```

En caso de querer escribir en una dirección la forma de hacerlo es similar, se indica la dirección, el nuevo valor que se quiere escribir y el tipo del valor. En este caso Java hace el cast del valor automáticamente.

```
cvi.Write(dir,50,KNXAdapter.KNXTypes.UCOUNT);
```

Finalmente, si no queremos seguir comunicándonos con el servidor, hay que llamar al siguiente método:

```
cvi.Disconnect();
```

Si cerramos el programa sin desconectar, el servidor no cerrará la conexión hasta pasado un tiempo, y algunos servidores KNX no permiten más de una conexión simultánea. Esto quiere decir que no podríamos conectarnos de nuevo hasta que el servidor decidiera dar por cerrada la conexión antigua.

ANEXO II: Protocolos de comunicación IR implementados

Dado que no existe ningún protocolo estándar de comunicaciones para dispositivos IR ha habido que implementar varios protocolos de diferentes fabricantes. Los protocolos actualmente implementados se explican a continuación.

Philips RC-5

Esta codificación es probablemente la más popular ya que son varios los fabricantes que lo implementan. El protocolo está bien definido para diferentes tipos de dispositivos con el fin de garantizar la compatibilidad con un sistema de entretenimiento completo.

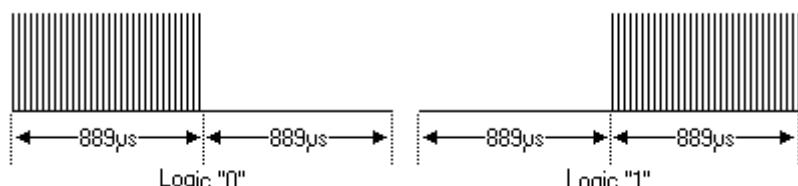
Actualmente Philips está utilizando el protocolo RC-6 en sus nuevos modelos, sucesor del RC-5, que añade más características. Aun así sigue dando soporte y compatibilidad al RC-5.

Características

- 5 bits de dirección y 6 bits de comando (7 bits de comando para el RC-5X).
- Codificación bifase (Manchester).
- Onda portadora de 36 KHz.
- Tiempo de bit constante de 1.778 ms (64 ciclos de 36 KHz).
- Creado por Philips.

Modulación

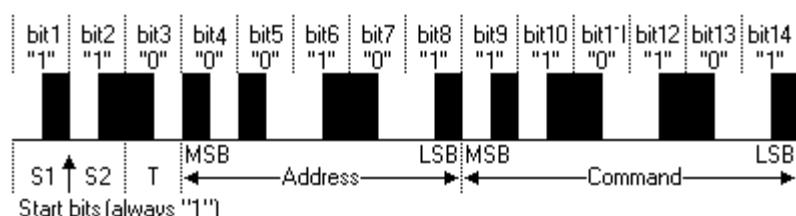
El protocolo usa una modulación bifase (o también llamado codificación Manchester) con una onda portadora IR de 36 KHz. La principal característica de esta codificación es que todos los bits tienen la misma longitud, en este caso de 1.778 ms, con la mitad del tiempo en pausa y la otra con una ráfaga de 36 KHz.



Como se aprecia en la imagen superior, el cero lógico se representa por una ráfaga de 889 μs, que equivale a 64 ciclos de 36 KHz, y 889 μs de pausa. El uno lógico se representa igual pero de forma invertida, con 889 μs de pausa en la primera parte del bit y por una ráfaga de 889 μs en la segunda parte.

Trama

La imagen de abajo muestra la trama de un mensaje RC-5 típico. En este ejemplo se transmite el comando 0x35 a la dirección 0x05.



Toda trama comienza con dos bits a 1 seguido del llamado bit *Toggle* o comutador. Este bit se invierte cada vez que el mismo botón es apretado y liberado. De esta forma el receptor puede distinguir cuando un botón permanece apretado o si se aprieta de forma repetida.

Los siguientes 5 bits representan la dirección IR del dispositivo, cada tipo de dispositivo posee una dirección única, en el ejemplo se trataría de un dispositivo VCR. Le seguirían 6 bits del comando a ejecutar, en el ejemplo se trata de la acción *Play*.

El mensaje consiste en un total de 14 bits, cuya duración total es de 25 ms. En caso de que el mismo botón se mantenga apretado el mensaje se repetirá cada 114 ms, durante todas esas repeticiones el bit *Toggle* permanece al mismo nivel lógico.

Existe el llamado Extended RC-5 o RC-5X. Este consiste también en una trama de 14 bits pero posee solo un bit de *Start* a 1, con ello se consigue aumentar a 7 el número de bits de comando.

Sony SIRC

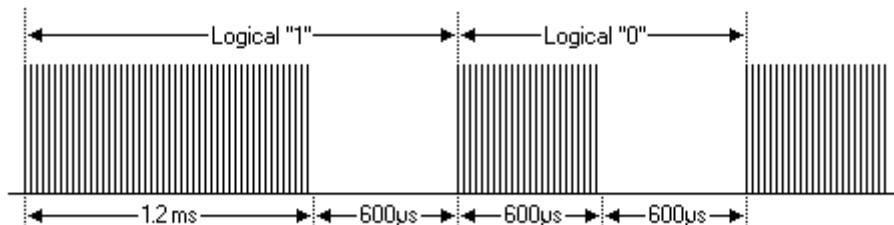
El protocolo de Sony está basado en la modulación por ancho de pulsos (MAP o PWM, siglas del inglés *Pulse-Width Modulation*). Existen tres versiones de este protocolo, de 12, 15 y 20 bits.

Características

- Versiones de 12, 15 y 20 bits.
- 7 bits de comando, 5 bits de dirección (8 en la versión de 15 bits).
- 8 bits extras para la versión extendida de 20 bits.
- Modulación por ancho de pulsos.
- Onda portadora de 40 KHz.
- Creado y usado solo por Sony.

Modulación

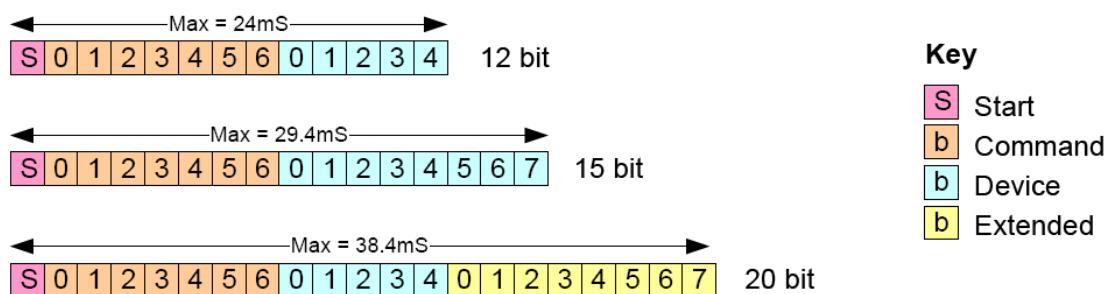
El protocolo SIRC usa una modulación por ancho de pulsos para codificar los bits. Cada bit se representa por una ráfaga 40 KHz de tamaño variable en función del bit y seguido de 600 µs de pausa.



Para representar un 1 lógico se envía una ráfaga de 1.2 ms y para representar un 0 lógico solo se envía una ráfaga de 600 µs, seguidas ambas de 600 µs de pausa.

Trama

En la imagen de abajo se pueden apreciar las tramas de las tres diferentes versiones del protocolo SIRC que existen.



El bit de Start es un bit especial que se representa enviando una ráfaga de 2.4 ms seguida de 600 µs de pausa. Luego siguen 7 bits de comando y 5 u 8 bits con la dirección del dispositivo. En caso de la versión extendida se disponen de 8 bits extras para dar información específica al dispositivo.

El tiempo mínimo entre dos tramas no puede ser inferior a 45 ms. Los controles remotos de Sony por defecto repiten cada trama un mínimo de 3 veces.

Panasonic

Este protocolo está inspirado en el RECS-80, pero usa más bits que éste. Igual que los otros protocolos explicados, éste también modula la señal para enviar la trama. Para la comunicación se utiliza un pulso con una longitud fija, seguida de una pausa que representa el estado lógico del bit.

Modulación

- Frecuencia de 38KHz.
- $T = 420 \mu s$ hasta aproximadamente $424 \mu s$ en los EE.UU y Canadá.
- $T=454 \mu s$ hasta aproximadamente $460 \mu s$ en Europa y otros países.
- La cabecera es de $8T$ a alto nivel y $8T$ a bajo nivel.
- Un 1 se codifica con $2T$ a alto nivel y $6T$ a bajo nivel.
- Un 0 se codifica con $2T$ a alto nivel y $2T$ a bajo nivel.

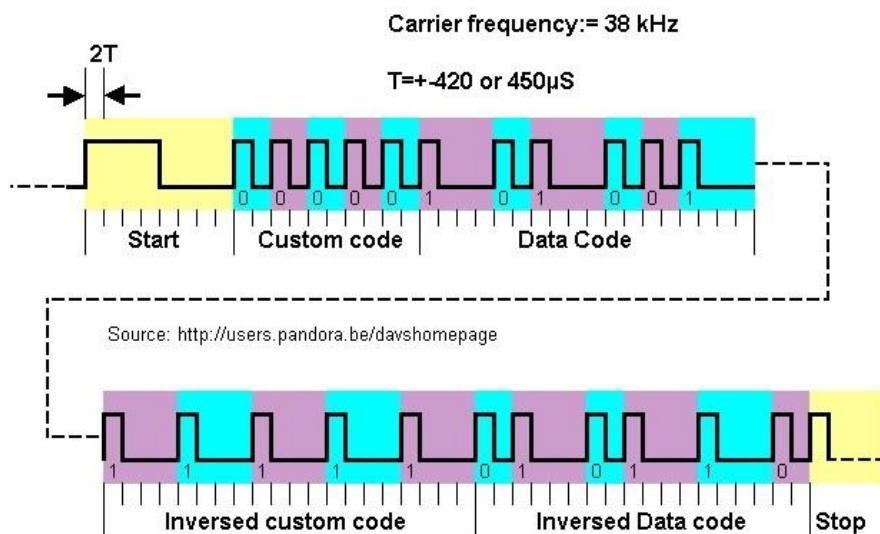
Cada una primera parte de un bit es siempre a alto nivel con un tiempo fijo y es seguido por un nivel bajo donde el tiempo se define si el bit es un 1 lógico o un 0 lógico.

Trama

En este protocolo se definen 2048 códigos, dividido en 5 trozos de código personalizado y 6 trozos de código de datos. El código personalizado es un valor que representa el código del fabricante y el código de datos es un valor que representa el botón pulsado del mando a distancia.

El código de transmisión total es de 22 bits: En primer lugar una cabecera se envía a continuación el código personalizado (5 bits), luego el código de datos, seguido por la inversa del código personalizado y la inversa del código de datos, seguido de un bit de fin de trama. Los bits de transmisión inversa son muy útiles para la detección de errores.

Old Panasonic IR remote control code



Daewoo

Modulación

Un 1 lógico está formado por 550 µs a alto nivel seguido por la señal de bajo nivel durante 1450 µs, esto significa que el tiempo de duración total de un 1 lógico es 2 ms. Un 0 lógico está formado por 550 µs a alto nivel seguido por una señal de bajo nivel durante 450 µs, esto significa que el tiempo de duración total de un 0 lógico es 1 ms.

La frecuencia de la onda portadora es de 38 kHz.

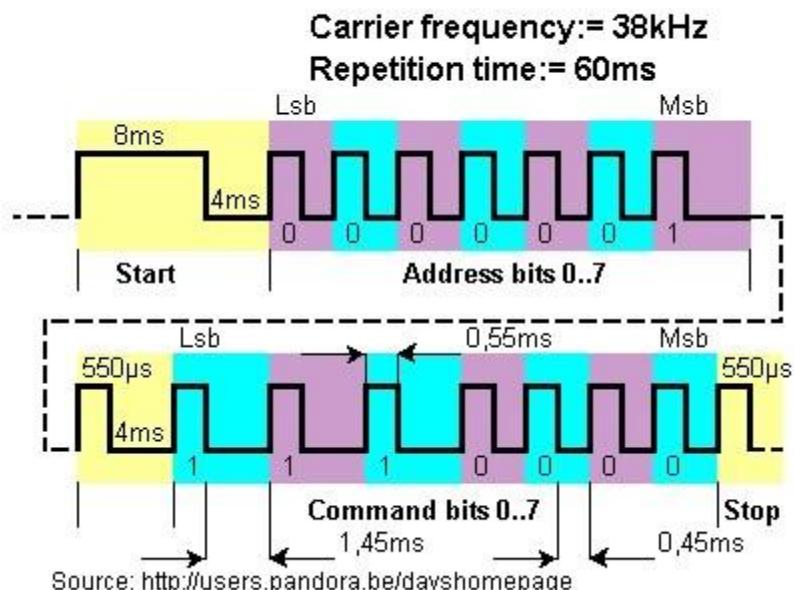
Trama

La trama se inicia con un bit de inicio de 8 ms a alto nivel y un pulso de 4 ms a bajo nivel. Después de esto se transmiten los 7 bits de dirección y los 7 bits de comandos. Cuando la trama acaba se envía un pulso de 600µs a alto nivel.

El protocolo se inicia con un bit de inicio de 8 ms a alto nivel y un pulso de 4 ms a bajo nivel. Después de esto se envían los 7 bits de dirección y los 7 bits de comandos. Pero las dos palabras se dividen por una señal de 550µs a alto nivel y seguido de una señal de 4ms a bajo nivel. Cuando finaliza la trama se envía un pulso de alto nivel de 550µs.

El tiempo entre dos tramas cuando el botón del mando a distancia está siendo presionado constantemente es de 60 ms.

Daewoo infrared remote protocol:



JVC

El protocolo remoto JVC de infrarrojos es casi igual al protocolo Daewoo. Sólo los tiempos y el espacio entre la dirección y los bits de datos difieren un poco.

Modulación

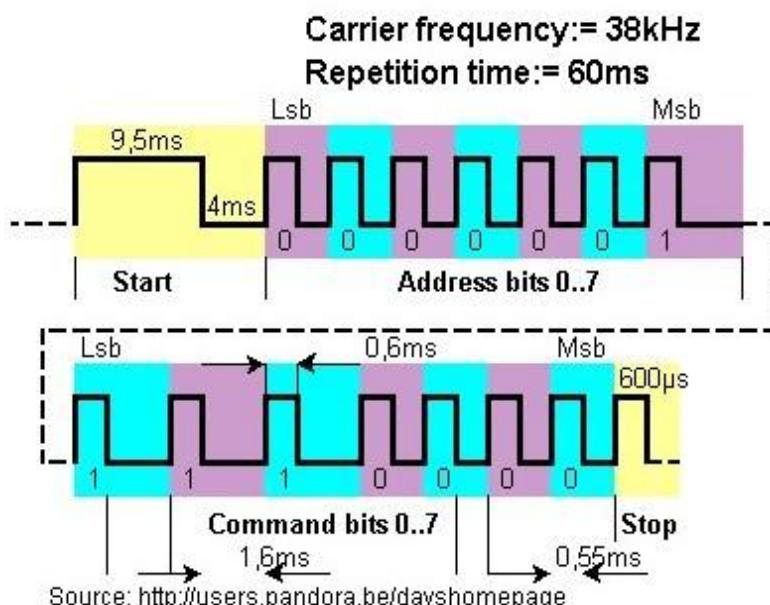
Un 1 lógico está formado por 600 µs a alto nivel seguido por la señal de bajo nivel durante 1600 µs. Un 0 lógico está formado por 600 µs a alto nivel seguido por una señal de bajo nivel durante 500 µs

La frecuencia de la onda portadora es de 38 kHz.

Trama

La trama se inicia con un bit de inicio de 8 ms a alto nivel y un pulso de 4 ms a bajo nivel. Después de esto se transmiten los 7 bits de dirección y los 7 bits de comandos. Cuando la trama acaba se envía un pulso de 600µs a alto nivel.

JVC infrared remote protocol:



ANEXO III: Código principal de la API

A continuación se muestra trozos de código de la API que he considerado importante.

Clase KNXInterface

```
@Override
public void Connect() throws KNXException {

    // create the network link to the KNX network
    final KNXNetworkLink lnk = createLink();

    // create process communicator with the established link
    pc = new ProcessCommunicatorImpl(lnk);

    registerShutdownHandler();
    // user might specify a response timeout for KNX message
    // answers from the KNX network
    if (this.Timeout > 0) pc.setResponseTimeout(this.Timeout);
}

@Override
public void Disconnect() {

    if (pc != null) {
        final KNXNetworkLink lnk = pc.detach();
        if (lnk != null) lnk.close();
        Runtime.getRuntime().removeShutdownHook(shutdownHandler);
    }
}

public Object Read(int address, String knxType) throws KNXException {

    final Datapoint dpt = new StateDP(new GroupAddress(address), "DP"
" "+address,1,knxType);

    String res = pc.read(dpt);

    if (knxType.equals(KNXType.BOOL)) {

        if (res.equals("true")) return true;
        else if (res.equals("false")) return false;
        else throw new KNXException("No se ha podido leer la
dirección indicada.");
    }
    else if (knxType.equals(KNXType.FLOAT)) {

        return Float.valueOf(res);
    }
    else if (knxType.equals(KNXType.STRING)) {

        return res;
    }
    throw new KNXException("No se reconoce el tipo de datos
indicado.");
}

public void Write(int address, String Type, Object value) throws
KNXException {

    final Datapoint dpt = new StateDP(new GroupAddress(address), "DP"
" "+address,1,value);
}
```

```

"+address,1,Type);

    pc.write(dpt, value.toString());
}

```

Clase WiFiExtensorInterface

```

@Override
public void Connect() throws Exception {

    ExtensorSocket = new Socket(ExtensorIP,ExtensorPort);

    out = new PrintWriter(ExtensorSocket.getOutputStream(), true);
    in = new BufferedReader(new InputStreamReader
(ExtensorSocket.getInputStream()));
}

@Override
public void Disconnect() {

    try {
        in.close();
        out.close();
        ExtensorSocket.close();
    } catch (IOException e) {e.printStackTrace();}
}

public int EnviarSeñal(int protocolo, int salida, String mens) throws
Exception {

    String protocol = String.valueOf(protocolo);
    if (protocolo < 10) protocol = "0" + protocol;

    out.println("B"+protocol+String.valueOf(salida)+mens+E");

    return Integer.valueOf(in.readLine());
}

```

ANEXO IV: Código del servidor TCP implementado en el PIC24

A continuación adjunto el código del servidor TCP implementado en el microcontrolador:

```
// Define el puerto que se utilizará para la comunicación
#define SERVER_PORT 85

// Cabecera de la función que gestiona el envío de la señal IR
int send_signal (BYTE buffer []);

/***********************/

Function:
void TCPServer(void)

Summary:
Implementa el servidor TCP encargado de gestionar las órdenes
procedentes del usuario.

Description:
Esta función implementa el servidor TCP encargado de gestionar
Las órdenes procedentes del usuario para el envío de comandos por
infrarrojos.

Precondition:
TCP ha sido inicializado.

Parameters:
None

Returns:
None

*******/
void TCPServer(void) {

WORD w;
BYTE AppBuffer[32];
WORD wMaxGet, wMaxPut, wCurrentChunk;
static TCP_SOCKET MySocket;
static enum _TCPServerState {
    SM_HOME = 0,
    SM_LISTENING,
```

```

    } TCPServerState = SM_HOME;

    switch(TCPServerState) {

        case SM_HOME:

            // Se asigna un zócalo para escuchar y aceptar conexiones

            MySocket = TCPOpen(0, TCP_OPEN_SERVER, SERVER_PORT,
TCP_PURPOSE_GENERIC_TCP_SERVER);

            if(MySocket == INVALID_SOCKET) return;

            TCPServerState = SM_LISTENING;

            break;

        case SM_LISTENING:

            // Se mira si hay alguien conectado al servidor

            if(!TCPIsConnected(MySocket)) {

                LED_CONN_IO = 0;

                return;

            }

            // De ser así se enciende el led CONN

            LED_CONN_IO = 1;

            // Averiguar cuantos bytes hemos recibido y cuantos podemos transmitir.

            wMaxGet = TCPIsGetReady(MySocket);      // Get TCP RX FIFO byte count

            wMaxPut = TCPIsPutReady(MySocket);      // Get TCP TX FIFO free space

            // Nos aseguramos que no cogemos más bytes de RX FIFO de los que

            // podemos meter en TX FIFO

            if(wMaxPut < wMaxGet) wMaxGet = wMaxPut;

            // Procesamos todos los bytes que podemos

            // La trama no debería ser mayor que sizeof(AppBuffer), por lo tanto no se

            tratan

            // más bytes. En caso de que se envíen demasiadas órdenes seguidas las

            ultimas se

            // perderán. No se pueden acumular las órdenes ya que esto podría ocasionar

            Lags,

```

```

// es preferible que no se ejecuten.

wCurrentChunk = sizeof(AppBuffer);

// Se transfieren los datos de la FIFO TCP RX a nuestro buffer de
procesamiento

// local.

TCPGetArray(MySocket, AppBuffer, wCurrentChunk);

// Se pone el led bicolor COM a 0

BLED_COM_OK_IO = 0;

BLED_COM_ERR_IO = 0;

// Se envia la señal IR

byte res = send_signal(AppBuffer);

// Si el resultado es menor que 0, algo ha ido mal. Se envia el código de
error y

// se pone el led bicolor COM en rojo

if (res<0) {

    AppBuffer[0] = res;

    AppBuffer[1] = '\0';

    BLED_COM_OK_IO = 0;

    BLED_COM_ERR_IO = 1;

} // Si ha ido bien se envia un 0 y se pone el led bicolor a verde

else {

    AppBuffer[0] = '0';

    AppBuffer[1] = '\0';

    BLED_COM_OK_IO = 1;

    BLED_COM_ERR_IO = 0;

}

// Se transfieren los datos del buffer local al TCP TX FIFO.

TCPPutArray(MySocket, AppBuffer, 2);

```

```

// Bucle de espera, para mantener el LED encendido un rato

int i,j;

for (i=0; i<5; i++) {

    for (j=0; j<16383; j++) {

        asm("nop");

    }

}

// Se apaga el led bicolor

BLED_COM_OK_IO = 0;

BLED_COM_ERR_IO = 0;

// No hay necesidad de hacer ningún flush. Los datos en TX FIFO se transmiten

// automáticamente ellos solos después de que se acumulen por un tiempo.

break;

}

}

/*****
```

Function:

```
byte send_signal (BYTE buffer [])
```

Summary:

Implementa la función encargada de enviar los comandos IR.

Description:

Esta función se encarga de tratar la trama que le envia el servidor TCP,
comprueba que los datos sean correctos y envia la señal IR correspondiente.

Precondition:

Parameters:

Buffer con la trama

Returns:

```
0 si ok, <0 si error.  
*****  
byte send_signal (BYTE buffer []) {  
    unsigned char protocol = 0;  
    unsigned char port = 0;  
  
    // Los tres primeros bytes son el protocolo y el puerto  
    if(buffer[1] >= '0' && buffer[1] <= '9') protocol = (buffer[1]-48)*10;  
    else return -1;  
    if(buffer[2] >= '0' && buffer[2] <= '9') protocol += (buffer[2]-48);  
    else return -1;  
    if(buffer[3] >= '0' && buffer[3] <= '4') port = (buffer[3]-48);  
    else return -1;  
  
    int i=4;  
    // Mientras no se encuentre el carácter E se sigue enviando la trama  
    while(buffer[i]!='E') {  
        if (buffer[i]!='0' && buffer[i]!='1') return -1;  
        i++;  
    }  
    return send_IR(protocol,port,&buffer[4]);  
}
```


ANEXO V: Código de generación de la señal IR desde el PIC24

A continuación se añade el código encargado de generar la señal IR desde el microcontrolador. Existe un fichero .c por cada protocolo implementado, aquí solo se muestra el código de protocolo SIRC.

```
#include "HardwareProfile.h"

void led_apagado_SIRC () { //off 600us

    int i;

    IRLED1_IO = 0;
    IRLED2_IO = 0;
    IRLED3_IO = 0;
    IRLED4_IO = 0;

    for (i=0; i<1580; i++) asm("nop");

}

void led_encendido_SIRC (unsigned char puertos) { //on 40 KHz

    int i,j;

    for (i=0; i<24; i++) {

        IRLED1_IO = 1&puertos;
        IRLED2_IO = 1&(puertos>>1);
        IRLED3_IO = 1&(puertos>>2);
        IRLED4_IO = 1&(puertos>>3);

        for (j=0; j<13; j++) asm("nop"); //10

        IRLED1_IO = 0;
        IRLED2_IO = 0;
        IRLED3_IO = 0;
        IRLED4_IO = 0;

        for (j=0; j<40; j++) asm("nop"); //11
    }
}
```

```

    }

}

void enviar_0_SIRC (unsigned char puertos) { //on-off

    led_encendido_SIRC(puertos);

    led_apagado_SIRC();

}

void enviar_1_SIRC (unsigned char puertos) { //on-on-off

    led_encendido_SIRC(puertos);

    led_encendido_SIRC(puertos);

    led_apagado_SIRC();

}

void enviar_cabecera_SIRC (unsigned char puertos) {

    led_encendido_SIRC(puertos);

    led_encendido_SIRC(puertos);

    led_encendido_SIRC(puertos);

    led_encendido_SIRC(puertos);

    led_apagado_SIRC();

}

int send_SIRC(unsigned char port, unsigned char mens []) {

    LED_SIGNAL_IO = 1;

    int i=0,j;

    unsigned char puertos;

    switch (port) {

        case 0:

            puertos = 15;

```

```

        break;

    case 1:

        puertos = 1;

        break;

    case 2:

        puertos = 2;

        break;

    case 3:

        puertos = 4;

        break;

    case 4:

        puertos = 8;

        break;

    default:

        puertos = 0;

}

enviar_cabecera_SIRC(puertos);

while(mens[i]!='E') {

    if (mens[i]=='0') enviar_0_SIRC(puertos);

    else if(mens[i]=='1') enviar_1_SIRC(puertos);

    else return -1;

    i++;

}

for (i=0; i<5; i++) {

    for (j=0; j<16383; j++) {

        asm("nop");

    }

}

LED_SIGNAL_IO = 0;

return 0;
}

```


ANEXO VI: Funciones del DLA

Conexiones

Las conexiones permiten el intercambio de datos entre los módulos del sistema. El tipo de datos a intercambiar no ha sido predefinido, siendo tratados por la arquitectura *DLA* como un flujo de bytes que puede contener cualquier tipo de datos simple o abstracto que requiera un módulo. Las funciones de la librería *DLALibrary* que permiten el uso de las conexiones son:

NewConnection

Función

Connection *NewConnection(char *host, int port, char *name, int size)

Descripción

Crea una conexión con su zona de memoria asociada para intercambiar datos en el sistema

Parámetros

- host: dirección IP del servidor central (o la palabra *local* para el esquema local).
- port: puerto de acceso del servidor central (ignorado para el esquema local).
- name: nombre de la conexión a ser creada
- size: tamaño en bytes de la conexión a ser creada

Valor devuelto

- Correcto: puntero a una variable de tipo *Connection* para posteriores Referencias
- Error: *NULL*

Las distintas conexiones del sistema están identificadas por su nombre, por lo que módulos que tienen que intercambiar datos entre sí sólo tienen que crear conexiones con el mismo nombre. Estas conexiones serán transparentemente enlazadas entre sí por la arquitectura *DLA*, independientemente de si utilizan un esquema distribuido o un esquema local. Las conexiones creadas inicializan automáticamente todos sus datos a cero.

El tamaño de una conexión indica el tamaño de los datos a intercambiar. Es importante notar que las conexiones no pueden ser redimensionadas una vez han sido creadas, por lo que el primer módulo que cree una conexión será el que establezca su tamaño. Todas las llamadas a la función *NewConnection* sobre una conexión ya existente simplemente ignoran el parámetro *size*.

InputConnection

Función

```
int InputConnection(Connection *conn, void *data, int pos, int size)
```

Descripción

Realiza la lectura de los datos almacenados en una conexión

Parámetros

- conn: puntero a la conexión que se quiere leer.
- data: puntero a la zona de memoria donde se quieren almacenar los datos leídos.
- pos: posición inicial de los datos que se quieren leer.
- size: tamaño en bytes de los datos que se quieren leer.

Valor devuelto

- Correcto: número de bytes leídos
- Error: -1

Mediante los parámetros *pos* y *size* se permite la lectura selectiva de una zona de datos determinada de la conexión sin necesidad de realizar la lectura completa de todos los datos almacenados en la misma. El parámetro *pos* indica la primera posición que desea ser leída, correspondiendo el valor cero a la primera posición disponible. El parámetro *size* indica la cantidad de datos en bytes que se desea leer, a partir de la posición *pos* especificada. Si el parámetro *size* contiene el valor cero, se interpreta que se quieren leer todos los datos disponibles desde la posición *pos* hasta el final de la conexión. Así pues, indicando cero en ambos parámetros se procede a realizar la lectura completa de todos los datos almacenados en la conexión.

OutputConnection

Función

```
int OutputConnection(Connection *conn, void *data, int pos, int size)
```

Descripción

Realiza la escritura de los datos almacenados en una conexión

Parámetros

- conn: puntero a la conexión que se quiere escribir
- data: puntero a la zona de memoria que contiene los datos que se quieren escribir
- pos: posición inicial de los datos que se quieren escribir
- size: tamaño en bytes de los datos que se quieren escribir

Valor devuelto

- Correcto: número de bytes escritos
- Error: -1

Mediante los parámetros *pos* y *size* se permite la escritura selectiva de una zona de datos determinada de la conexión sin necesidad de realizar la escritura completa de todos los datos almacenados en la misma. El parámetro *pos* indica la primera posición que desea ser escrita, correspondiendo el valor cero a la primera posición disponible. El parámetro *size* indica la cantidad de datos en bytes que se desea escribir, a partir de la posición *pos* especificada. Si el parámetro *size* contiene el valor cero, se interpreta que se quieren escribir todos los datos disponibles desde la posición *pos* hasta el final de la conexión. Así pues, indicando cero en ambos parámetros se procede a realizar la escritura completa de todos los datos almacenados en la conexión.

SizeConnection

Función

```
int SizeConnection(Connection *conn)
```

Descripción

Determina el tamaño en bytes de una conexión

Parámetros

- conn: puntero a la conexión cuyo tamaño se quiere determinar

Valor devuelto

- Correcto: tamaño en bytes de la conexión
- Error: -1

Esta función es útil para determinar el tamaño real que tiene una conexión recién creada en el sistema, pues como se ha comentado en la función *NewConnection* las conexiones no pueden ser redimensionadas una vez han sido creadas, por lo que el primer módulo que cree una conexión será el que establezca su tamaño.

CloseConnection

Función

```
int CloseConnection(Connection *conn)
```

Descripción

Destruye una conexión y libera los recursos reservados por la misma

Parámetros

- conn: puntero a la conexión que se quiere destruir

Valor devuelto

- Correcto: 0
- Error: -1

Esta función permite liberar los recursos utilizados por una conexión que no se va a utilizar más. Si se utiliza el esquema distribuido, la desconexión de los módulos del servidor central es automáticamente detectada y se liberan los recursos utilizados aunque no se utilice esta función. Si se utiliza el esquema local no se puede detectar automáticamente la desconexión de los módulos, por lo que es necesario utilizar esta función para liberar los recursos utilizados.

Buzones

Los buzones permiten ampliar la funcionalidad de la arquitectura de control *DLA* incorporando mecanismos de sincronización entre los distintos módulos del sistema. El tipo de datos a intercambiar ha sido predefinido al tipo entero, pues su utilidad principal no está asociada al dato intercambiado sino al instante de tiempo en el que se intercambia. Las funciones de la librería *DЛАLibrary* que permiten el uso de los buzones son:

NewMailbox

Función

```
Mailbox *NewMailbox(char *host, int port, char *name)
```

Descripción

Crea un buzón para realizar operaciones de sincronización en el sistema

Parámetros

- host: dirección IP del servidor central (o la palabra *local* para el esquema local)
- port: puerto de acceso del servidor central (ignorado para el esquema local)
- name: nombre del buzón a ser creado

Valor devuelto

- Correcto: puntero a una variable de tipo *Mailbox* para posteriores referencias
- Error: *NULL*

Los distintos buzones del sistema están identificados por su nombre, por lo que módulos que tienen que utilizar un mismo buzón para sincronizarse entre sí sólo tienen que crear buzones con el mismo nombre. Estos buzones serán transparentemente enlazados entre sí por la arquitectura *DLA*, independientemente de si utilizan un esquema distribuido o un esquema local. Los buzones pueden almacenar un valor de tipo entero, y al crearse inicializan automáticamente su valor a cero.

ReadMailbox

Función

```
int ReadMailbox(Mailbox *mail, int *data)
```

Descripción

Realiza la lectura del valor almacenado en un buzón

Parámetros

- mail: puntero al buzón que se quiere leer
- data: puntero a la variable de tipo entero donde se quiere almacenar el valor leído

Valor devuelto

- Correcto: 0
- Error: -1

Esta función permite leer el valor almacenado en un buzón. La función es no bloqueante, por lo que tras su ejecución el valor almacenado en el buzón es instantáneamente leído y almacenado en la variable indicada.

WaitMailbox

Función

```
int WaitMailbox(Mailbox *mail, int *data)
```

Descripción

Espera la llegada de nuevos valores a un buzón desde la última vez que se accedió

Parámetros

- mail: puntero al buzón que se quiere esperar
- data: puntero a la variable de tipo entero donde se quiere almacenar el valor leído

Valor devuelto

- Correcto: 0
- Error: -1

Esta función espera a que se escriban nuevos valores en un buzón desde la última vez que se accedió al mismo. Si no se ha escrito ningún nuevo valor en el buzón desde la última vez que se accedió la función es bloqueante, por lo que el módulo que realiza la llamada a la función pasa a un estado inactivo hasta que se escriba un nuevo valor en el buzón, momento en el cual será reactivado y se producirá el retorno de la función. Si algún nuevo valor ha sido escrito en el buzón desde la última vez que se accedió la función es no bloqueante, por lo que tras su ejecución el valor almacenado en el buzón es instantáneamente leído y almacenado en la variable indicada, siendo totalmente equivalente a la función *ReadMailbox*.

TestMailbox

Función

```
int TestMailbox(Mailbox *mail)
```

Descripción

Comprueba la llegada de nuevos valores a un buzón desde la última vez que se accedió

Parámetros

- mail: puntero al buzón que se quiere comprobar

Valor devuelto

- Correcto: 0 si no se han escrito nuevos valores, ó 1 si se han escrito nuevos valores
- Error: -1

Esta función es útil para determinar si una posterior llamada a la función *WaitMailbox* será o no bloqueante.

WriteMailbox

Función

```
int WriteMailbox(Mailbox *mail, int *data)
```

Descripción

Comprueba la llegada de nuevos valores a un buzón desde la última vez que se accedió

Parámetros

- mail: puntero al buzón que se quiere escribir
- data: puntero a la variable de tipo entero que contiene el valor que se quiere escribir

Valor devuelto

- Correcto: 0
- Error: -1

Esta función permite escribir un nuevo valor en un buzón. Tras escribir el valor correspondiente en el buzón, todos aquellos módulos bloqueados tras la llamada a la función *WaitMailbox* son automáticamente reactivados, produciéndose el retorno de dicha función.

CloseMailbox

Función

```
int CloseMailbox(Mailbox *mail)
```

Descripción

Destruye un buzón y libera los recursos reservados por el mismo

Parámetros

- mail: puntero al buzón que se quiere destruir

Valor devuelto

- Correcto: 0
- Error: -1

Esta función permite liberar los recursos utilizados por un buzón que no se va a utilizar más. Si se utiliza el esquema distribuido, la desconexión de los módulos del servidor central es automáticamente detectada y se liberan los recursos utilizados aunque no se utilice esta función. Si se utiliza el esquema local no se puede detectar automáticamente la desconexión de los módulos, por lo que es necesario utilizar esta función para liberar los recursos utilizados.

16 Bibliografía

Las fuentes de información que he necesitado para la realización de mi proyecto han sido principalmente extraídas de Internet. A continuación se adjuntan para ampliar toda la información dada en este PFC.

16.1 PFCs

- PFC de Miquel Jesús Tejero Rodríguez, *Identificació d'ordres provinents de comandaments a distància domèstics*.
- PFC de Jordi Divins Pujols, *Supervisió i control d'una vivenda Intel·ligent*.

16.2 Konnex

- Librería Calimero: <http://calimero.sourceforge.net/>
- Konnex Association: <http://www.knx.org/>

16.3 Firmware

- TCP/IP stack de microchip: ww1.microchip.com/downloads/en/AppNotes/00833b.pdf
- MRF24WB0MA/MRF24WB0MB Data Sheet:
ww1.microchip.com/downloads/en/DeviceDoc/70632A.pdf
- PIC24FJ256GA106 Data Sheet:
<http://ww1.microchip.com/downloads/en/DeviceDoc/39905e.pdf>
- FlyPort: <http://www.openpicus.com/>
- <http://picprojects.org.uk/>

16.4 IR

- <http://jlirc.sourceforge.net/>
- <http://www.lirc.org/>
- <http://www.sbprojects.com/knowledge/ir/ir.htm>
- http://www.uicontrol.com.ar/Articulos/protocolo_de_los_controles_remotos_philips_RC5/protocolo_de_los_controles_remotos_philips_RC5.htm
- <http://users.telenet.be/davshomepage/>
-

16.5 Otros

- Librerías IKVM: <http://www.ikvm.net/>
- Domótica Fagor: http://www2.fagor.com/domotica/_bin/cast/index.php

16.6 General

- <http://es.wikipedia.org>
- <http://www.google.es>

17 Agradecimientos

Este proyecto no hubiera sido posible sin el soporte recibido de muchas personas y organizaciones. Es por ello que les he querido dedicar un espacio en la memoria.

Primero de todo me gustaría agradecer a Carlos, Xevi, Quim, Laureano, Toni, José y Guillem, compañeros del departamento de ESAII, por conseguir que mis horas de dedicación al proyecto hayan sido más amenas, pero sobre todo por la ayuda desinteresada que me han ofrecido. Así como Guiem, Aurora, Berta y Georgina por lo mismo durante mi estancia en el CVI.

Agradezco todo el soporte que me ha ofrecido por todo el departamento de ESAII y el CVI, en especial a Antonio B. Martínez por ofrecerme la posibilidad de realizar este proyecto y a Joan Vidos por su infinita paciencia ayudándome en los problemas que me he ido encontrando.

También agradecer a mis padres, no solo por el apoyo económico que he necesitado a lo largo de la carrera, sino por todo el interés que han demostrado en mis estudios y en este proyecto, y en especial a mi madre por sus sabios consejos literarios.

A todos ellos, muchas gracias por vuestra ayuda.

