Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño y desarrollo de la iluminación de un camino de jardín programable inteligente

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Miguel Juan Monter

Tutor: José Luis Poza Luján

Grado de Ingeniería Informática

Resumen

Este proyecto surge de la necesidad de reducir el impacto ambiental que implica el uso de iluminación en entornos exteriores.

Se relata el análisis, el diseño y la elaboración de un sistema de gestión inteligente de iluminación, donde cada luminaria dispondrá de un arduino con distintos sensores y actuadores además de un sistema de interconexión, para el uso eficiente de la energía.

El proyecto está planteado para un jardín aunque perfectamente puede ser migrado a entornos de mayor envergadura como grandes espacios o ciudades.

Palabras clave: Arduino, control de iluminación, sensores y actuadores, sistema inteligente, Sistema distribuido.

Abstract

This project emerges from the need of reduce the environmental impact that involves the use of lighting in outdoor environments.

Analysis, design and development of an intelligent lighting management, where each fixture will have an Arduino with various sensors and actuators as well as a networking system for efficient use of energy are reported.

The project is proposed for a garden but can be fully migrated to larger environments such as large areas or cities.

Keywords: Arduino, lights control, sensors and triggers, intelligent system, distributed system.

Agradecimientos

Quiero mostrar mi agradecimiento a todas las personas que han hecho posible este proyecto. En primer lugar a mi director de proyecto, José Luis Poza Luján por su apoyo y energías en todas las fases de este proyecto.

En segundo lugar a todos mis compañeros que forman parte de la unión de proyectos que es ArduEntorno, por su colaboración y apoyo. También quiero agradecer a José Luis Jiménez García por su ayuda en todo momento en que la he necesitado.

Por último y más importante, a mi actual pareja, mi familia y amigos por haberme apoyado en todo momento.

Tabla de contenidos

Índice de Ilustraciones

Índice de tablas

Índice de códigos

1. Introducción

1.1. Contextualización

En los entornos abiertos como jardines, parques, zonas recreativas o grandes extensiones deben ser iluminados para poder tenerlos disponibles con una visibilidad adecuada en todo momento. Esta iluminación implica un coste energético que, en ocasiones, puede ser elevado para el nivel de uso de la instalación, por ejemplo, un jardín particular no tiene porqué encender todas sus luces en todo momento, puede iluminarse sólo la zona en la que haya gente o bien iluminar con la intensidad de luz adaptada a la luz ambiental en ese momento.

Actualmente los sistemas basados en micro controladores o micro procesadores, tales como Arduino [1], Intel Galileo [2] o Raspberry PI [3], permiten controlar una gran cantidad de sensores y actuadores. Emplear estos elementos en el control de luces de entornos abiertos permite gestionar, por medio de los microcontroladores o los microprocesadores, el gasto energético en función de las políticas (iluminación, consumo, etc.) que el usuario considere adecuadas.

1.2. Objetivos

Son múltiples los objetivos que se plantean con este proyecto. El objetivo principal del proyecto es conseguir una reducción significativa de los costes de iluminación mediante el uso de un microcontrolador Arduino y diversos sensores y actuadores basándose en políticas de gestión de la iluminación. Como consecuencia de este objetivo, se consigue también reducir el impacto medioambiental que este consumo genera.

Este objetivo principal genera una serie de objetivos procedimentales o sub-objetivos, que son los siguientes.

- Revisar y analizar los sistemas de gestión de iluminación existentes en la actualidad para conocer las tecnologías empleadas y poder tomar decisiones técnicas en el proyecto.
- Especificar, a partir del objetivo anterior, las características técnicas que deberá tener el proyecto, para cubrir el objetivo principal.
- Diseñar un sistema automático basado en Arduino con las características especificadas en el objetivo anterior que permita implementar el sistema eficientemente.
- Implementar un prototipo funcional del sistema diseñado que valide los objetivos anteriores.

A parte, se plantea que este proyecto sea una alternativa a otros competidores pero con unos costes de material mucho más reducidos, haciendo el precio del sistema más barato y por lo tanto más atractivo para el usuario final.

1.3. Entorno de realización

El proyecto se enmarca dentro de un sistema llamado ArduEntorno. El sistema ArduEntorno es un sistema modular. En el sistema existe un tipo de nodo llamado nodo de comunicación. La función de este nodo es la de hacer de intermediario entre el servidor y los nodos de control específicos de cada módulo. Con esta funcionalidad, cada usuario podrá decidir qué características del sistema ArduEntorno quiere implementar en su caso, haciendo que el sistema sea 100% modulable.

Adicionalmente, gracias a estos nodos de comunicación, obtenemos un sistema donde la escalabilidad está garantizada, ya se les puede conectar los nodos de control de cada

módulo, pudiendo tener varios módulos del mismo tipo pero en distintas superficies, dentro de la zona de instalación.

Gracias a los nodos de comunicación, podemos tener un sistema distribuido en el cual distintos módulos estén en distintas redes, ya que como estos módulos se conectan al servidor, no tienen ningún problema de conexión.

También garantizaremos la seguridad del sistema. Los nodos de comunicación son capaces de almacenar datos de los nodos asociados a él, evitando la perdida de datos en caso de una caída de la red o el servidor que une con el nodo de comunicación. Los nodos de control de cada módulo también serán capaces de almacenar sus propios datos en el caso de que detectase un fallo en la conexión. Además de garantizar que no se pierdan datos, los nodos de comunicación también desempeñan un papel de capa de seguridad separando los nodos de cada módulo con internet.

El sistema ArduEntorno está compuesto por una serie de módulos, que actualmente se están desarrollando en otros proyectos paralelos al actual. El listado de módulos es:

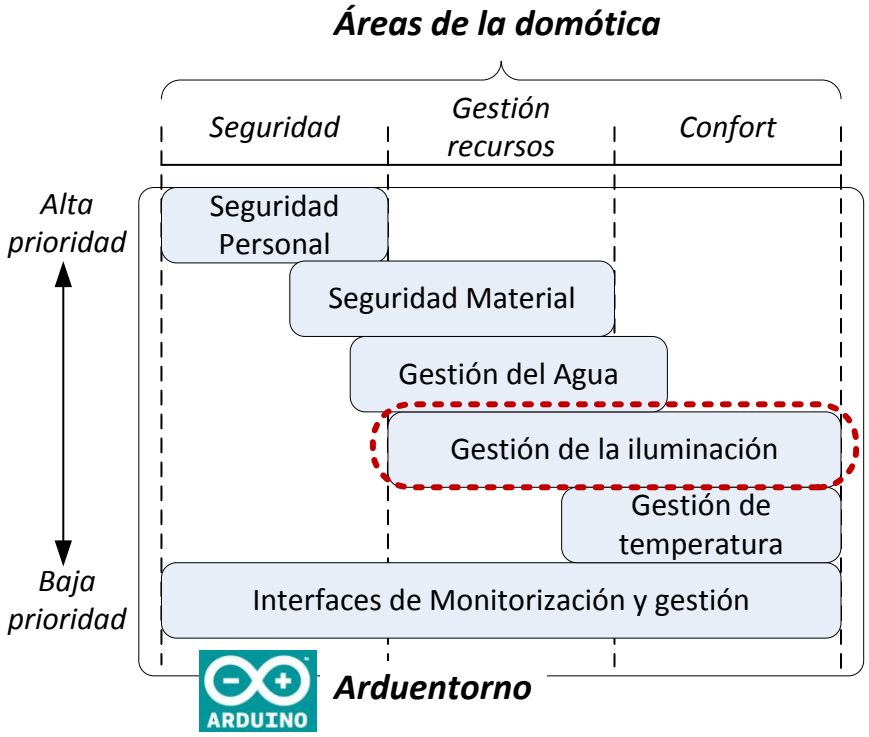

Ilustración 1. Módulos del sistema Arduentorno y ubicación del proyecto dentro del sistema (línea roja discontinua).

El módulo de ArduEntorno Android está centrado en la gestión y control del resto de módulos conectados a nuestro sistema.

1.4. Descripción del documento

El siguiente capítulo (capítulo 2) muestra una revisión detallada de los sistemas similares al objetivo del proyecto. El capítulo comienza con un apartado llamado "Estado de la cuestión" donde se hará un análisis de distintas compañías que compiten en el mismo mercado que el resultado de este proyecto. Después de la revisión, se realiza un análisis cuantitativo y cualitativo que permite obtener características comunes y comparar los distintos sistemas entre sí y con el sistema implementado. El capítulo concluye con una descripción de la tecnología que se empleará en la implementación del proyecto.

A continuación, en el capítulo 3 se especifican los requisitos del proyecto por medio del estándar IEEE 830, en esta especificación se explicarán las funcionalidades del sistema desarrollado en este proyecto.

Seguidamente, en el capítulo 4, se especificara las especificaciones software y hardware del sistema además de su topología. Se realizara una explicación completa de la topología.

Posteriormente, en el capítulo 5, se detallara en el proceso de implementación del sistema y los distintos sistemas que se emplearan en el proyecto. El capítulo concluirá con el producto final obtenido de este proyecto.

Finalmente, el capítulo 6, muestra las conclusiones, aportaciones y posibles aplicaciones del proyecto.

2. Estado de la cuestión

2.1. Introducción

En este apartado se analiza todos los sistemas similares que se pueden encontrar en el mercado. Además se realiza una comparación y una síntesis de que puntos podemos atacar para destacar el producto del resto de la competencia. Finalmente, se describen las características del sistema inteligente [4] que se deberá desarrollar.

2.2. Control de entornos abiertos

En el presente proyecto, estaremos trabajando en entornos abiertos. Podemos considerar un entorno abierto como un espacio acotado al aire libre donde todos los elementos de nuestro sistema no se encontraran en un interior, salvo el servidor y algún componente de comunicación. Como ejemplos prácticos de entornos abiertos podríamos considerar jardines, espacios lúdicos o incluso amplias extensiones de naturaleza donde cohabitan personas y naturaleza.

Al trabajar en entornos abiertos, estamos teniendo en cuenta de espacios donde la extensión de terreno a cubrir con nuestro sistema, sugiere una cantidad considerable de sensores, actuadores y nodos. A consecuencia de estar al aire libre el sistema, nos encontramos a un entorno más agresivo al que nos podríamos encontrar en un espacio cerrado [5]. Tendremos problemas de tipo ambiental y de tipo que nuestro sistema puede recibir daño físico tanto de animales como de personas.

Los sensores y actuadores se encontraran ante un entorno donde muchas de las variables que captemos sean difíciles de controlar [6]. Por ejemplo un sensor de temperatura en un espacio cerrado siempre tendrá una temperatura aproximadamente linear, mientras que en un espacio abierto, esa linealidad desaparece y los valores recibidos varían según luminosidad, condiciones ambientales o presencia de animales o personas. A consecuencia de esto, los valores que reciban nuestros sensores no son posibles de predecir.

Debido a estos problemas que podremos encontrarnos en un entorno abierto, nuestro sistema tendrá implícitamente las siguientes características:

Las distancias entre sensores, actuadores y nodos pueden llegar a ser considerablemente grande, por ello dispondremos de una red de comunicación alámbrica e inalámbrica para que todo elemento esté conectado al sistema.

Debido a la propia naturaleza del entorno, deberemos de disponer de distintos tipos se de sensores y actuadores. Como ejemplo, un sensor PIR (Passive InfraRed sensor, sensor de presencia por infrarrojos) puede percibir la presencia de personas su espacio de interacción, pero una vez llegamos a primavera y verano, la presencia del follaje de los árboles, sea imposible detectar presencia y por lo tanto necesite de un segundo sensor para respaldar al primer PIR y que se siga detectando la presencia en ese espacio.

Debido a grandes distancias entre elementos y a grandes extensiones de terreno, dispondremos de una amplia red de nodos de comunicación que se encargaran de enviar toda la información al servidor y que no quede ningún espacio sin cubrir, haciendo posible a su vez un sistema distribuido y escalable.

Por lo tanto, es un reto considerable el proceso de poder gestionar un entorno abierto el cual, en comparación con uno cerrado, presenta tantas dificultades.

A continuación se analizan diversos sistemas similares (en funcionalidad o en aplicación) al proyecto planteado, para poder contextualizar el entorno de trabajo y comprobar cómo resuelven, si lo hacen, los problemas anteriores y posibles decisiones estratégicas a tomar.

2.3. Sistemas similares

2.3.1. Tvlight BV

Tvlight BV [7] fue creada en colaboración con la Delf University of Tecnology, Holanda [8]. Este sistema puede suponer un ahorro energético de un 80% en comparación a un sistema de iluminación tradicional y una reducción de los costes de mantenimiento de un 50%.

Dispone de sensor de detección de presencia capaz de detectar personas y vehículos. Los niveles de luz se ajustan acorde a las condiciones climáticas, siendo mayor la intensidad con lluvias torrenciales o más débiles en presencia de nieve e incorpora un sistema a prueba de errores, en el cual, si el sistema falla, la luz se quedara encendida.

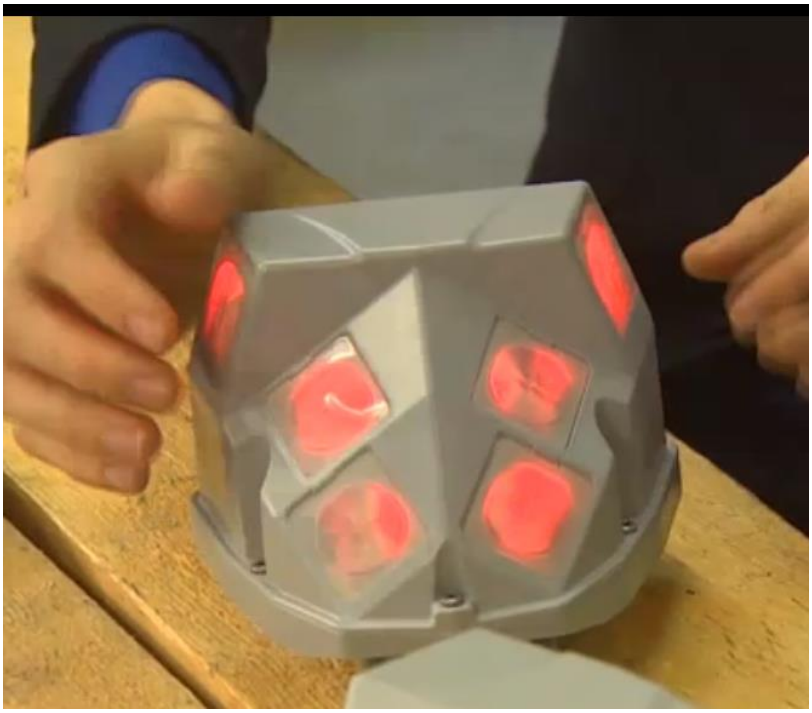

Ilustración 2 Sensor de presencia Tvlight BV.

Con el sistema de luces inteligentes, se incluye una aplicación para la gestión de las luces, pudiendo recibir notificaciones SMS o e-mails. Además almacena estadísticas de ahorro energético. Este software es una plataforma abierta que puede interaccionar con hardware de otros proveedores, habilitando servicios adicionales a la misma plataforma.

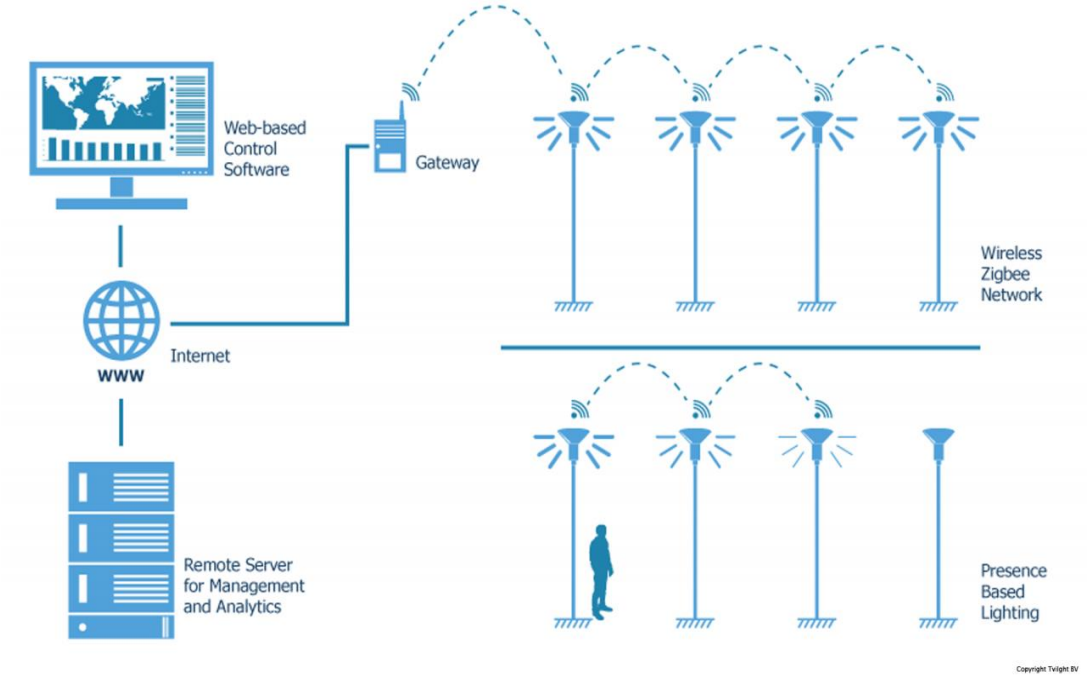

Ilustración 3: Arquitectura tvlight

Actualmente ofrecen 2 productos distintos: CitySense, Skylite.

El producto de CitySense está más pensado para espacios de gran envergadura como una ciudad. Además de la detección de presencia, dispone de un control de fenómenos atmosféricos, SMS de control en emergencias, notificaciones e-mail, estadísticas de ahorro energético y a prueba de errores. Además recomiendan el uso de la herramienta software, CityManager, donde puedes cambiar la configuración de una farola o incluso de un grupo de ellas. Este software es una plataforma abierta que permite interactuar con hardware de otros proveedores (como ejemplo un monitor ambiental y de tráfico) habilitando servicios adicionales con la misma plataforma.

Skylite está más pensado para particulares ya que es más económico que el anterior y de fácil instalación. Además del sistema de detección de presencia, se dispone de notificaciones SMS o por e-mail y estadísticas de ahorro energético. El sistema remoto y gestión se pueden realizar via la aplicación CityManager.

2.3.2. E-Street

E-Street [9] es un proyecto a nivel europeo en el cual participan múltiples compañías, ciudades y universidades.

En primer lugar, una luminaria equipada con fuentes de luz de intensidad regulable. La configuración básica es una lámpara y el balasto regulable en combinación con un controlador. Llamamos a este controlador “controlador de luminaria exterior” (OLC).

La iluminación se adapta al volumen de tráfico, condiciones medioambientales y la propia luminosidad de la superficie de la carretera.

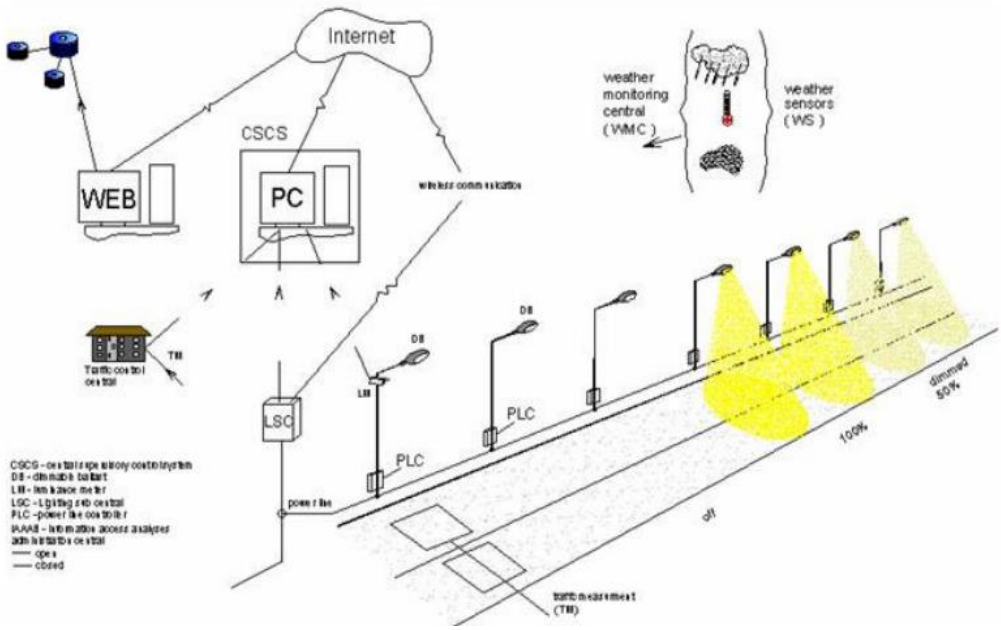

Ilustración 4. Estructura E-Street.

Cada farola dispone de un sistema local (PLC) que se encarga del control de la lámpara y sus sensores así como la transmisión al controlador de sector (LSC). El LSC realiza la tarea de hacer de intermediario entre una serie de PLCs y la central, conocida como CSCS, realizando tareas de intercomunicación y respaldo de datos. Este es capaz de almacenar cierto volumen de datos y dispone de un UPS capaz de aguantar 3 horas en caso de corte del suministro eléctrico para evitar la pérdida de información.

El centro de control del sistema (CSCS) es usado para controlar y gestionar la información recibida por los controladores de sector (LSC). Este es capaz de gestionar distintos LSC incluso si estos se encuentran en distintos proveedores de servicios y diferentes direcciones WAN.

2.3.3. LUIX

Luix [10] es una compañía que está asentada desde 2009 en Donostia (Guipúzcoa), perteneciente al grupo ACR.

Las farolas disponen de un nodo que puede ser desde un controlador simple, hasta un ordenador embebido con conector para modem 3G. La conexión puede ser Ethernet o 3G. Aparte se les puede añadir dispositivos adicionales como un altavoz o un hotspot wifi. El sistema Luix permite la transmisión de datos, voz y vídeo a través de una infraestructura que puede ser 100% PLC o híbrida (PLC + wifi). La sonorización se realiza mediante el uso de cámaras y/o radares.

Ilustración 5. Luminaria y sensor LUIX

En la aplicación SmartLuix, los puntos de luz están geo posicionados, facilitando su uso a los usuarios. Su funcionamiento es vía web y la conexión se asegura mediante conexión ADSL, 3G, GPRS o similar situada en el armario de control de la instalación de alumbrado. Esta nos permite: visualizar componentes instalados, monitorización del sistema a tiempo real, estimación de consumo, alarmas, información de vida útil, generación de informes y programación de tareas de mantenimiento.

2.4. Análisis sistemas similares

A continuación se muestran tres tablas con el análisis de las características de los principales sistemas similares al sistema objetivo del trabajo final de grado.

	Nº De luces	Tipo luz	Sensor	Detección 360º	Arquitectura
TVILIGHT VB	N/A	LED y Convencional	Proximidad	SI	En bus
E-STREET	N/A	LED y Convencional	Proximidad	SI	Jerárquica
LUIX	N/A	LED	Radar y cámara	SI	N/A

Tabla 1: Análisis cuantitativo 1

	Tolerancia a fallos	Uso Ornamental	Escalabilidad	Ahorro energético	Valor añadido
TVILIGHT VB	Si	N/A	Si	80,00%	N/A
E-STREET	Si	N/A	Si	N/A	N/A
LUIX	N/A	N/A	Si	80,00%	Nodos expansible con servicios adicionales

Tabla 2: Análisis cuantitativo 2

	Buses de comunicacion	Uso movil	Software	Hardware	Control Meteorologico
TVILIGHT VB	3G / GPRS	No	Si	Propietario	Si
E-STREET	N/A	No	Si	Propietario	Si
LUIX	Cableado eléctrico	SI	Si	Propietario	N/A

Tabla 3: Análisis cuantitativo 3

Podemos observar que podemos mejorar en comparación con la competencia en el uso de hardware y el uso ornamental.

Con motivo de que toda la competencia hace uso de hardware propietario, el hardware libre como es Arduino será una reducción considerable en los costes materiales. Además, la competencia no usa sus luminarias para dar un toque ornamental, pudiendo realizar espectáculos de luces y colores. Este puede ser otro flanco el cual afrontar para dar la posibilidad al cliente de realizar sus propios espectáculos.

Basándonos en la revisión anterior, se puede observar que:

- Los sistemas son escalables de forma que se puedan controlar tantas luminarias como se desee.
- Las luminarias deben de ser LED para la reducción de costes de consumo y mantenimiento.
- Los sistemas de detección deben de ser de 360º
- La tolerancia a fallos es un factor muy importante para el bienestar de los usuarios.
- El uso de un software de control le añade un valor extra al sistema ya que permite de forma sencilla la monitorización y gestión del sistema.

Por tanto, un sistema completo e integral, debería de cubrir como mínimo:

- Uso de luminarias LED.
- Múltiples sensores para asegurar la tolerancia a fallos y la detección a 360º.
- Sistema escalable.
- Software de gestión.

Y debería cubrir, opcionalmente:

- Gestión y puesta en marcha de espectáculos lumínicos.
- Gestión de luminarias por zonas y franjas horarias.
- Control de la iluminación conforme a las condiciones meteorológicas.

2.5. Tecnología implicada

Todos los sistemas analizados en este capítulo hacen uso de un sistema hardware propietario; lo que supone un mayor coste ya solo en hardware. Este es el motivo por el cual hacemos uso de la tecnología Arduino. Ya que como esta es de código abierto, es más económico en comparación. Otro beneficio de un sistema de código abierto, es el hecho de que hay más sensores y actuadores preparados para funcionar correctamente en Arduino, y no necesitan configuraciones complicadas o elementos hardware adicional.

Todos los sistemas hacen uso de sistemas de detección PIR, ya que por muy poco consumo eléctrico, tienes un sistema con alta fiabilidad que además dispone de un buen ángulo de detección y un alcance considerable. Este es el motivo por el cual también usaremos nosotros un sistema de detección PIR.

Aparte del sistema de detección PIR, dispondremos también de medidores de distancia de ultrasonidos. Estos serán usados como detectores de presencia. La diferencia con el PIR, es que este nos puede decir la distancia desde que se produce la detección.

2.6. Síntesis: Sistema a desarrollar

El proyecto se enmarcará, como se ha dicho previamente en la introducción, en un sistema llamado ArduEntorno. El sistema es uno de los múltiples módulos que hay disponible para ArduEntorno, pero adicionalmente el propio sistema es ya modulable por sí mismo. Como el sistema esta compuestos por nodos que pueden funcionar independientemente del estado en que se encuentren el resto de nodos, permitiendo que el sistema sea escalable.

Disponemos también de nodos de comunicación cuya única tarea es la de hacer de intermediario entre los nodos de control y el sistema. Este nodo permite al sistema ser distribuido ya que podemos tener nodos en distintas redes.

La seguridad del sistema está garantizada a través del servidor, que desempeña el papel de capa de seguridad entre el exterior y la red de arduinos evitando así intrusiones en el sistema. Adicionalmente los nodos de comunicación y los nodos de control son capaces de almacenar información que no ha sido posible enviar por el fallo de un nodo de comunicación intermedio o el servidor.

2.7. Conclusiones

Podemos sacar en conclusión que el uso de un sistema hardware de código abierto, nos hará más económicos los costes del sistema y por lo tanto el coste para el usuario final.

Indistintamente del hardware que se use, siempre que se use la misma jerarquía, podremos garantizar las características de modularidad, escalabilidad, distribución y seguridad de nuestro sistema.

Adicionalmente, el usuario final podrá decidir que módulos quiere disponer en su caso e incluso diseñar sus propios módulos o los módulos de otras empresas para complementar a su gusto el sistema.

Como consecuencia de todo lo analizado en este capítulo, se extrae todo lo necesario para el siguiente capítulo que es la especificación de requisitos.

3. Especificación de requisitos

3.1. Introducción

En este capítulo de especificación de requisitos, de primeras definiremos el personal involucrado, las definiciones que se emplearan a lo largo de todo el texto y el alcance. A continuación se hará un análisis detallado siguiendo el IEEE 830 donde especificaremos todos los requisitos funcionales.

3.1.1. Propósito

Esta especificación tiene como objetivo analizar y documentar las necesidades funcionales que deberán ser soportadas por el sistema a desarrollar. Para ello, se identificarán los requisitos que ha de satisfacer el nuevo sistema mediante entrevistas con el cliente, estudio de los problemas de las unidades afectadas y sus necesidades actuales.

El propósito del proyecto es la elaboración de un sistema de iluminación inteligente que permita el ahorro eléctrico y la reducción del impacto medioambiental.

3.1.2. Alcance

Este Trabajo Final de Grado está dentro del proyecto general “ArduEntorno” que consta de los siguientes sub-proyectos:

- Seguridad
- Energía
- Iluminación
- Confort
- Aplicación móvil

3.1.3. Personal involucrado

Nombre	Miguel Juan Monter
Rol	Jefe de proyecto
Categoría profesional	Ingeniero técnico informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno iluminación
Información de contacto	mijuamon@inf.upv.es
Aprobación	
Nombre	José Luis Poza Luján
Rol	Director del proyecto
Categoría profesional	Profesor Contratado Doctor de la UPV
Responsabilidades	Coordinación de sub-proyectos de ArduEntorno. Dirección de los TFG o PFC correspondientes
Información de contacto	jopolu@disca.upv.es
Aprobación	
Nombre	Alberto Pedrera Ros
Rol	Responsable de otro componente del sistema
Categoría profesional	Ingeniero informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno energía
Información de contacto	alpedro@inf.upv.es
Aprobación	

Nombre	Jesús Brun Conejos
Rol	Responsable de otro componente del sistema
Categoría profesional	Ingeniero informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno seguridad
Información de contacto	jesbruco@ei.upv.es
Aprobación	
Nombre	Alberto Ramírez
Rol	Responsable de otro componente del sistema
Categoría profesional	Ingeniero técnico informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno Android
Información de contacto	
Aprobación	
Nombre	Carlos Gil
Rol	Responsable de otro componente del sistema
Categoría profesional	Ingeniero técnico informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno ambiente
Información de contacto	
Aprobación	
Nombre	Carlos Quer
Rol	Responsable de otro componente del sistema
Categoría profesional	Ingeniero técnico informático
Responsabilidades	Diseño, desarrollo e implementación de ArduEntorno fuente
Información de contacto	
Aprobación	

Tabla 4: Personal involucrado

3.1.4. Definiciones, acrónimos y abreviaturas

Actuador	Componente electrónico y/o mecánico que ante una señal, es capaz de interactuar con el entorno
Aplicación	Programa en lenguaje JAVA con interfaz para la gestión de todo el entramado de nodos y áreas.
Arduino	Arduino UNO rev.3
Área	Reparto u organización del espacio determinada por los requisitos del cliente. Por ejemplo el área de entrada al jardín o el área de descanso.
Base de datos	Aplicación en MYSQL para almacenamiento de la topología del sistema así como de eventos detectados.
Código de Control	Código programado en el lenguaje de Arduino [11]
Detección parcial	Activación de uno o varios sensores, pero no todos, que forman parte de un nodo.
Detección total	Activación de todos los sensores que forman parte de un nodo, como motivo del paso de una persona.
Elemento de control	Cualquier dispositivo final (sensor, actuador o similar) que está conectado a un nodo. Por ejemplo, un sensor de luz o un relé que controla una farola.
Elemento virtual de control	Elemento que emplea un nodo, sin estar conectado a él. Por ejemplo, un sensor que, desde el nodo al que está conectado, envía información a otro nodo.

Entorno	Todo espacio en el que se aplica el sistema de control, aunque no esté controlado (se podría definir también como el conjunto de áreas que el cliente desea automatizar)
Entorno abierto	Espacio a automatizar, por nuestro sistema, el cual se encuentra a la intemperie.
Espacio de control	Conjunto de espacios de interacción con algún tipo de relación controlados por uno o varios nodos de control.
Espacio de interacción	Espacio cubierto por un elemento. Por ejemplo, el espacio que cubre la detección de un sensor de ultrasonidos o el espacio que ilumina una farola.
Ethernet	Estándar empleado para la comunicación entre los Arduinos [12]
Luminaria	Sistema de iluminación compuesto por bombillas incandescentes, halógenas y/o leds
Microcontrolador	Es un circuito integrado programable capaz de ejecutar órdenes grabadas en su memoria.
Nodo de comunicaciones	Nodos que, además de poder tener sensores y actuadores, interconectan otros nodos entre sí, u otros nodos con los servidores.
Nodo de control	Conjunto de sensores y actuadores y el Arduino que los controla. Para ser considerado un nodo, el Arduino debe estar conectado al sistema por medio de cualquier tecnología de comunicaciones.
PIR	Passive Infrared Sensor. Sensor pasivo de infrarrojos capaz de detectar movimiento.
Sección	Conjunto de nodos y un servidor que están conectados.
Sensor	Componente capaz de percibir estímulos del entorno y que es capaz de transmitirlo
Servidor	Dispositivo al que están conectados un conjunto de nodos y que recibe peticiones o envía órdenes a los nodos.
Sistema Arduino	Conjunto de sensores y actuadores unidos directamente a un micro controlador Arduino. Quedan excluidos aquellos que se unan por una tecnología diferente de la proporcionada por el Arduino.
Sistema de interconexión	Conjunto de componentes encargados de la comunicación entre elementos, nodos y servidor.
Ultrasonido	Sensor que es capaz de detectar la distancia entre este y un obstáculo mediante el uso de ultrasonidos, basándose en el tiempo en que tarda en volver rebotada una señal.

Tabla 5: Tabla de definiciones, acrónimos y abreviaturas

3.1.5. Referencias

Al ser el uno de los primeros Trabajos Finales de Grado del proyecto común ArduEntorno, las siguientes referencias se refieren a trabajos finales de grado o proyectos finales de carrera que se están realizando en paralelo al presente.

Referencia	Titulo	Fecha	Autor
[en preparación]	Diseño y desarrollo de una fuente de jardín inteligente programable	N/A	Carlos Quer
[en preparación]	Sistema de seguridad perimetral programable inteligente	N/A	Jesús Brun Conejos
[en preparación]	Control vía móvil de un jardín inteligente	N/A	Alberto Ramírez
[en preparación]	Sistema de creación inteligente de ambientes en entornos exteriores	N/A	Carlos Gil

Tabla 6: Tabla de referencias.

3.1.6. Resumen

Este proyecto surge de la necesidad de reducir el impacto ambiental que supone tener una iluminación siempre encendida.

Se relata la elaboración de un sistema de gestión inteligente de iluminación, donde cada luminaria dispondrá de un Arduino con distintos sensores y actuadores además de un sistema de interconexión, para el uso eficiente de la electricidad.

3.2. Descripción general

3.2.1. Perspectiva del producto

El sistema ArduEntorno – Energía exterior forma parte de un sistema mayor llamado ArduEntorno aunque puede funcionar perfectamente sin cualquiera de las otras funcionalidades que pueda albergar ArduEntorno. Específicamente el sistema ArduEntorno – Energía está centrado en la gestión eficiente de un sistema de iluminación ubicado en espacio abierto, generalmente un jardín.

3.2.2. Funcionalidad del producto

El sistema Arduino está compuesto por múltiples sensores de movimiento para detectar la presencia de una entidad y así encenderse las luminarias su paso. Cuando no detecta nada el sistema Arduino, este atenuara la luz para así reducir el consumo energético. Además dispondrá de sensores para activarse únicamente cuando sea necesaria la iluminación. El sistema Arduino estará conectado a una toma eléctrica y a una toma Ethernet.

3.2.3. Características de los usuarios

Tipo de usuario	Usuario Final
Formación	Básica en el control de dispositivos móviles y páginas Web
Habilidades	Conocimiento del medio sobre el que va a usar el sistema ArduEntorno
Actividades	Manejo del sistema

Tabla 7: Características del usuario final

Tipo de usuario	Administrador
Formación	Control de dispositivos móviles y páginas Web. Conceptos de programación de dispositivos (conceptos básicos de programación)
Habilidades	Gestión y manejo del sistema.
Actividades	Gestión de la aplicación

Tabla 8: Características del usuario gestor

Tipo de usuario	Programador
Formación	Amplios conocimientos de informática
Habilidades	Programador
Actividades	Mantenimiento de la aplicación

Tabla 9: Características del usuario programador

El usuario final será quien de uso al sistema de forma cotidiana sin necesidad de programar o comprender el funcionamiento del sistema.

Un administrador será el encargado de la configuración y mantenimiento del sistema. Entre sus responsabilidades se halla la de sustituir elementos defectuosos y la de ampliar y configurar el sistema.

Los programadores son los encargados de poner a punto la jerarquía y el software que llevara el sistema, de forma que un administrador no tenga que programar nada relacionado con los nodos.

3.2.4. Restricciones

Las conexiones de los nodos se realizan mediante estándar Ethernet. Todo nodo deberá de disponer de un trasformador de 220 voltios para poder estar alimentado. Tanto Arduinos como sensores y actuadores deberán disponer de carcasas para aislarse de los infortunios de fenómenos medioambientales.

3.2.5. Suposiciones y dependencias

Podría darse la necesidad de una actualización del software de gestión de iluminación si el lenguaje de programación utilizado, alterase algunas de las funciones actuales, haciendo inviable el uso de la aplicación.

Otro elemento que afectaría es que el actual protocolo Ethernet empleado en el proyecto, acabe siendo obsoleto, haciendo necesario la migración a un nuevo protocolo de comunicación más actual.

3.2.6. Evolución previsible del Sistema

Como posibles mejoras para el sistema se pueden plantear un sistema de espectáculo luminoso y sonoro. Otra mejora factible es el abandono del estándar Ethernet y adoptar uno inalámbrico (bluetooth, WIFI, zigbee, etc.) o uno cableado de mayor tasa de transferencia. Otra evolución seria el cambio de los arduinos UNO por otra micro-controladora más potente.

3.3. Requisitos específicos

3.3.1. Requisitos comunes de los interfaces

3.3.1.1. Interfaces de usuario

La interfaz de usuario estará implementada en forma de aplicación JAVA en un ordenador. Por lo que será necesario un ordenador, una pantalla, un teclado y un ratón. La interfaz es sencilla y de fácil uso y la curva de aprendizaje es lo suficiente baja como para que cualquier tipo de usuario pueda manejar el software correctamente.

3.3.1.2. Interfaces de hardware

Cada elemento tendrá una interfaz específica, dependiendo de la infraestructura del elemento. Cada elemento tendrá una configuración específica que puede modificarse de forma lógica o física.

3.3.1.3. Interfaces de software

Sera necesario que el ordenador que monitoriza los nodos, disponga de JAVA instalado para el correcto funcionamiento de la aplicación de gestión. Cumpliendo con todos estos requisitos, el manejo de la aplicación de gestión será posible.

3.3.1.4. Interfaces de comunicación

Para la infraestructura necesaria para la interconexión de todos los nodos será necesario el uso de múltiples switchs y/o router además de una toma de red adicional en el ordenador para poder conectarse a la red de sensores y actuadores.

Hará falta que los switchs, los routers o el propio ordenador ejecuten el papel de servidor DHCP, ya que los nodos requieren de una dirección IP. Para la infraestructura de comunicación se empleara el protocolo Ethernet.

3.3.2. Requisitos funcionales

3.3.2.1. Detección de presencia

El sistema permite detectar la entrada de alguien en el área de interacción de una serie de sensores. Además de encender la iluminación, lo notificara al servidor.

Entrada	Sensor PIR - Sensor ultrasonidos
Proceso	Los elementos reciben la detección, la comunican al nodo al cual estén conectados y este procesa la comunicación con el servidor y dependiendo de la configuración previa, enciende la luz o no.
Salida	Iluminación del camino, señal Ethernet.

Tabla 10: Requisito funcional detección de presencia

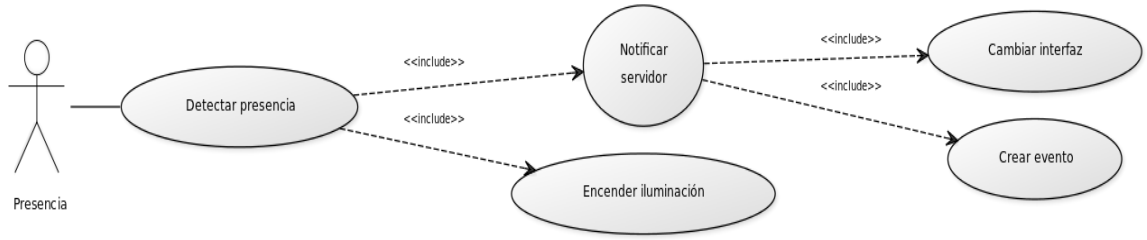

Ilustración 6: Caso de uso de detección de presencia

3.3.2.2. Detección parcial

Tras detectar presencia en un espacio de interacción pero no todos los que detectan en el mismo espacio. Notificara al servidor y este dará la orden de encender o no la iluminación.

Entrada	Sensor PIR – Sensor Ultrasonido
Proceso	Tras procesar el nodo una detección parcial, se manda una consulta al servidor, donde este, basándose en el estado que tienen los nodos vecinos al solicitante, le responde avisando de si debe o no encender la luz.
Salida	Iluminación del camino, señal Ethernet

Tabla 11: Requisito funcional Detección parcial

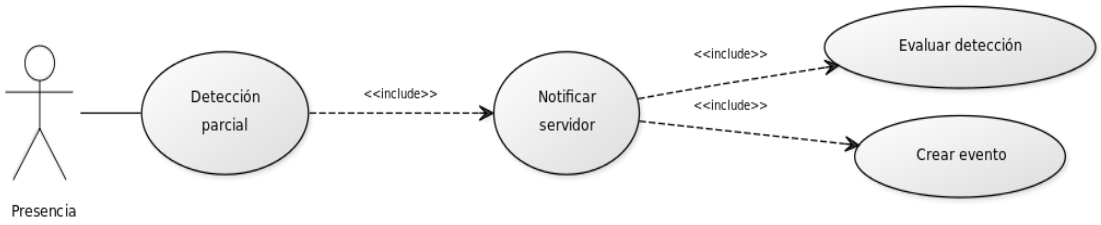

Ilustración 7: Caso de uso de detección parcial

3.3.2.3. Fin de detección de presencia

Tras dejar de detectar presencia y un tiempo de gracia, la iluminación vuelve a un estado de reposo o apagado. Se notificara al servidor y este genera un evento.

Entrada	Sensor PIR - Sensor ultrasonidos
Proceso	Los elementos que intervienen en un mismo espacio de interacción dejan de detectar presencia y tras pasar varios segundos sin detectar

	una presencia, da la orden de apagado de la iluminación y lo notifica al servidor.
Salida	Iluminación del camino, señal Ethernet.

Tabla 12: Requisito funcional fin de detección de presencia

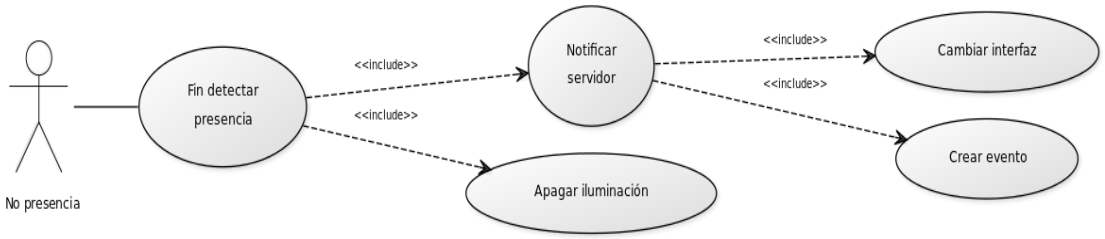

Ilustración 8: Caso de uso de fin de detección de presencia

3.3.2.4. Encendido programado

Al alcanzar cierta oscuridad o franja horaria, la iluminación se enciende. El encendido por franja horaria dependerá de la configuración previa que se le haya puesto al área. Adicionalmente se genera el correspondiente evento.

Entrada	Sensor de luz, señal Ethernet
Proceso	El servidor detectara la hora actual y dependiendo de la configuración de las áreas, les mandara una orden de encendido a los nodos.
Salida	Iluminación encendida

Tabla 13: Requisito funcional encendido programado

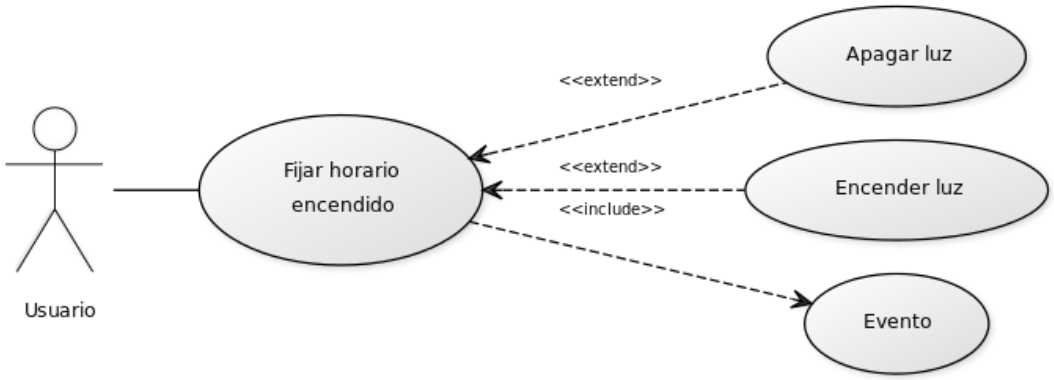

Ilustración 9: Caso de uso de encendido programado

3.3.2.5. Reducción de luminosidad programada

Según la configuración de las áreas, al alcanzar cierta hora, la iluminación pasara a una potencia reducida hasta que detecte presencia y pase a un estado encendido con la potencia al 100%. Adicionalmente se genera el correspondiente evento.

Entrada	Señal Ethernet
Proceso	El servidor detectara la hora actual y dependiendo de la configuración de las áreas, les mandara al nodo la orden de consumo reducido.
Salida	Iluminación encendida a baja potencia

Tabla 14: Requisito funcional reducción de luminosidad programada

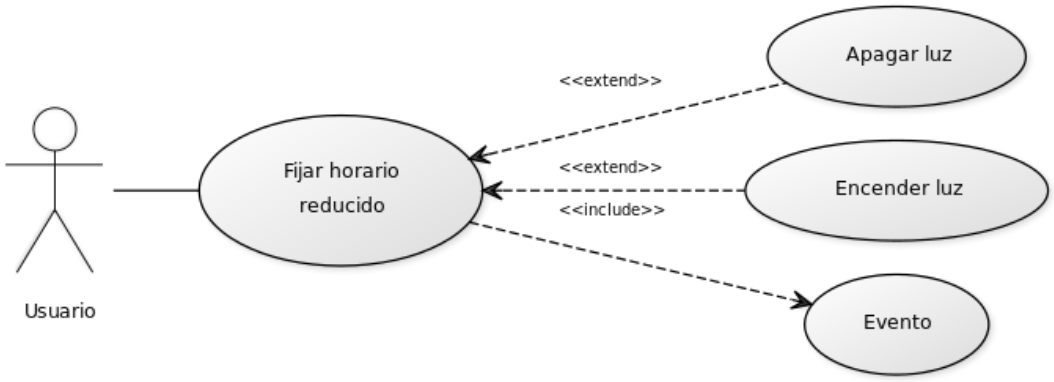

Ilustración 10: Caso de uso de luminosidad reducida programada

3.3.2.6. Apagado programado

Al llegar a una hora determinada, la luz se apaga. Esta hora dependerá de la configuración realizada en el servidor al área. Adicionalmente se genera el correspondiente evento.

Entrada	señal Ethernet
Proceso	Se configura previamente en el servidor la hora de apagado. Una vez alcanzada esa hora, mandara a los correspondientes nodos la orden de apagado.
Salida	Iluminación apagada

Tabla 15: Requisito funcional apagado programado

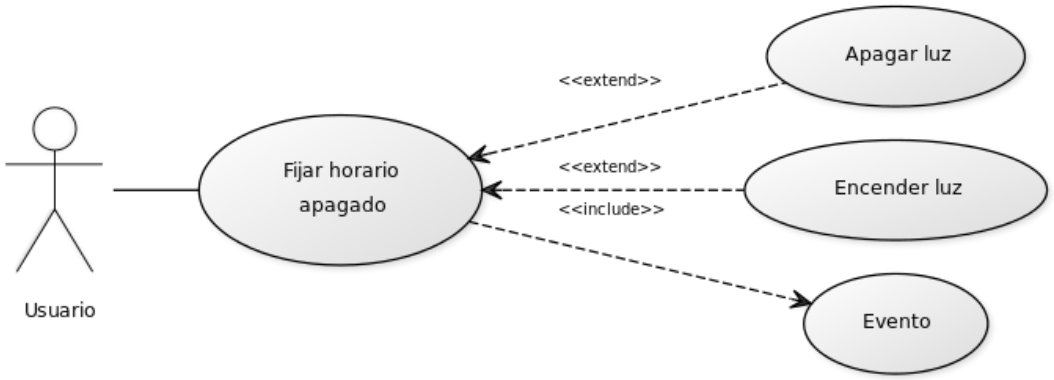

Ilustración 11: Caso de uso de apagado programado

3.3.3. Requisitos no funcionales

3.3.3.1. Requisitos de rendimiento

El número máximo de usuarios simultáneos será de 1 mientras que el número de transacciones simultáneas que podrá realizar será igual al número de arduinos instalados. Todas las transacciones deberán ser de computadas en un tiempo menor a un segundo.

3.3.3.2. Seguridad

Se plantean los siguientes ámbitos de seguridad

- Nivel de elemento: Cada elemento se encontrara instalado en una caja herméticamente cerrada para así prevenir de cualquier daño físico que pueda recibir, pero sin reducir la capacidad de sensorización del propio sensor o funcionalidad del actuador.
- Nivel de nodo: La seguridad del controlador se basara en una caja herméticamente cerrada permitiendo solo la entrada del cableado necesario para las comunicaciones entre otros nodos y elementos. Esta caja prevendrá daños físicos en el controlador e intentos de modificación no autorizada de los controladores.
- Nivel de comunicaciones: Las comunicaciones estarán protegidas mediante una codificación específica del proyecto donde nadie fuera del entorno ArduEntorno conoce esta codificación.
- Nivel de interfaz de usuario: dado que la aplicación de interfaz de monitorización y control no tiene validación de usuarios, la seguridad será la que proporcione el sistema en el que se ejecuta la aplicación.

3.3.3.3. Fiabilidad

Se plantean los siguientes ámbitos de fiabilidad

- Nivel de elemento: los sensores y actuadores que se empleen deberán tener la fiabilidad descrita en sus especificaciones. No se contempla el uso de elementos redundantes (duplicidad o sistemas de votación de sensores y actuadores).
- Nivel de nodo: se plantea la fiabilidad diferenciada entre controlador del nodo y código
    - Controlador: la fiabilidad será la correspondiente a las especificaciones del controlador empleado. No se contempla el uso de controladores redundantes (duplicidad o sistemas de votación de control).
    - Código: la fiabilidad del código se garantiza en cuanto que estará preparado para detectar valores no factibles (fuera de rango) y la conexión con los elementos.
- Nivel de comunicaciones: la fiabilidad dependerá de las especificaciones de los medios empleados y de las condiciones en las que se encuentren éstos.
- Nivel de interfaz de usuario: la fiabilidad está garantizada en el sentido que comprueba las conexiones con los nodos y envía y recibe la información de errores de los mismos.

Se proporcionará al cliente, en la documentación correspondiente a su sistema, la información de fiabilidad del hardware empleado.

3.3.3.4. Disponibilidad

Se plantean los siguientes ámbitos de la disponibilidad

- Nivel de elemento: La disponibilidad correspondiente a los sensores y actuadores será la especificada en sus especificaciones. No se contempla el uso de elementos redundantes (duplicidad o sistemas de votación de sensores y actuadores).
- Nivel de nodo: se plantea la disponibilidad diferenciada entre controlador del nodo y código
    - Controlador: la disponibilidad será la correspondiente a las especificaciones del controlador empleado. No se contempla el uso de controladores redundantes (duplicidad o sistemas de votación de control).

    - Código: la disponibilidad del código estará garantizada siempre que el controlador siga estando disponible.
- Nivel de comunicaciones: la disponibilidad dependerá de las especificaciones de los medios empleados y de las condiciones en las que se encuentren éstos.
- Nivel de interfaz de usuario: la disponibilidad está garantizada en el sentido que la aplicación siempre estará disponible, siempre que el usuario lo requiera.

3.3.3.5. Mantenibilidad

Se plantean los siguientes ámbitos de la mantenibilidad

- Nivel de elemento: La mantenibilidad correspondiente a los sensores y actuadores será la especificada en sus especificaciones. No se contempla el uso de elementos redundantes (duplicidad o sistemas de votación de sensores y actuadores).
- Nivel de nodo: se plantea la mantenibilidad diferenciada entre controlador del nodo y código
    - Controlador: la mantenibilidad será la correspondiente a las especificaciones del controlador empleado. No se contempla el uso de controladores redundantes (duplicidad o sistemas de votación de control).
    - Código: la mantenibilidad del código estará garantizada para que sea innecesaria y en todo caso solo se modifique para aplicar tareas de actualización software.
- Nivel de comunicaciones: la mantenibilidad dependerá de las especificaciones de los medios empleados y de las condiciones en las que se encuentren éstos.
- Nivel de interfaz de usuario: la mantenibilidad está garantizada en el sentido de que un usuario final nunca deberá de realizar labores de mantenimiento y si se produjesen, las realizaría un administrador.

3.3.3.6. Portabilidad

Se plantean los siguientes ámbitos en la portabilidad

- Nivel de elemento: La portabilidad siempre será posible entre distintos elementos, siempre que los requisitos que tengan y las señales que emitan, sean similares al proyecto original.
- Nivel de nodo: se plantea la portabilidad diferenciada entre controlador del nodo y código
    - Controlador: La portabilidad en los controladores siempre será posible, pudiendo usar en un caso distintos controladores.
    - Código: Como podemos tener controladores de distintos tipos, la portabilidad del código es posible ya que cada controlador dispone su propio lenguaje de programación y por lo tanto, el código usado en un controlador, debe ser posible usar en otro controlador.
- Nivel de comunicaciones: la mantenibilidad dependerá de las especificaciones de los medios empleados y de las condiciones en las que se encuentren éstos.
- Nivel de interfaz de usuario: la mantenibilidad está garantizada en el sentido de que un usuario final nunca deberá de realizar labores de mantenimiento y si se produjesen, las realizaría un administrador.

3.4. Conclusión

Gracias a este apartado hemos podido definir todas las definiciones que usaremos durante el desarrollo del sistema. Adicionalmente hemos desarrollado todos los requisitos funcionales y no funcionales que tendremos en cuenta en el siguiente capítulo que es el diseño del sistema.

4. Diseño del sistema

4.1. Introducción

En este capítulo se detalla el proceso de diseño del sistema, a través de los requisitos funcionales del capítulo anterior. Se detallara la topología que se usara en el sistema, el porqué de uso y su descripción formal. También se especificara el hardware que se usara en el sistema y los UMLs [13] que se usaran para el desarrollo del software de gestión del sistema.

4.2. Topología del sistema

4.2.1. Descripción conceptual

En este punto se explica los distintos aspectos de la topología del sistema empleado en el proyecto. Esta topología es necesaria para poder tener todos los elementos bien ubicados y con una fácil escalabilidad.

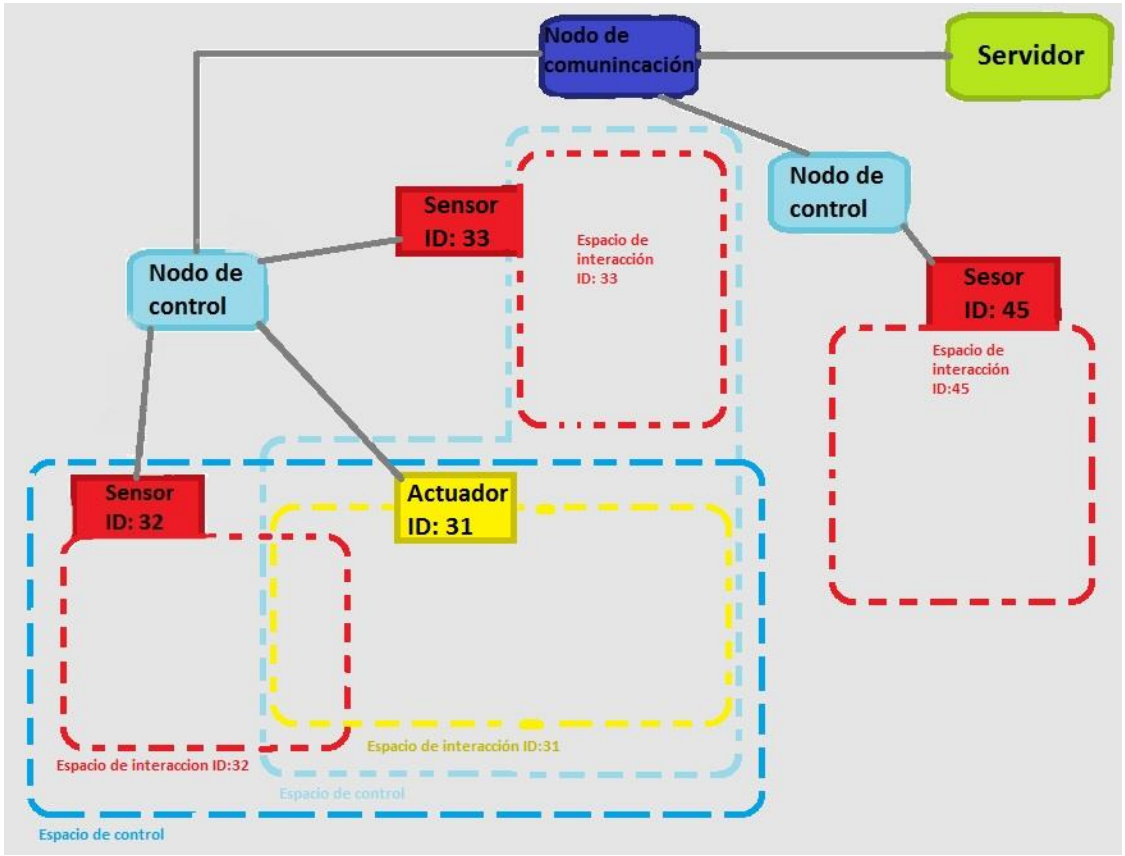

Ilustración 12: Descripción conceptual del sistema

En la ilustración se ha podido ver un ejemplo de la topología y de cómo estarán interconectados los elementos. Los elementos, que son los sensores y actuadores, estarán conectado a un nodo de control y adicionalmente estos disponen de un espacio de interacción. Un nodo de control, que es un microcontrolador, tendrá conectados múltiples elementos y de la unión de sus espacios de interacción surge un espacio de control. El nodo de control de la izquierda dispone de dos espacios de control. Los nodos de control se comunicaran con los nodos de comunicación que desempeñan un papel de interconexión con el servidor.

4.2.1.1. Elemento

Un elemento puede ser un sensor, un actuador o incluso un microcontrolador.

Un sensor es lo que le permitirá al sistema percibir el entorno mientras que un actuador le permitirá a nuestro sistema intervenir en el entorno. Un microcontrolador es el que recibirá los valores de los sensores y fijara los de los actuadores.

4.2.1.2. Elemento virtual

Cualquier elemento puede ser un elemento virtual. La característica principal de un elemento virtual es la posibilidad de que pueda ser usado por un nodo de control al cual no está conectado directamente, pasando a través del nodo de control por el cual si está conectado directamente.

4.2.1.3. Espacio de control

Es el espacio producto de la unión de múltiples espacios de interacción con algún tipo de relación controlado por uno o varios nodos de control. Como ejemplo es la unión de los espacios de interacción de dos sensores PIR más el actuador de una farola.

4.2.1.4. Espacio de interacción

Es el espacio físico en el cual un usuario interactúa con un elemento. Como ejemplo, el rango de actuación de un sensor de ultrasonidos.

4.2.1.5. Nodo de comunicación

Nodo especial usado para interconectar múltiples nodos de control con el servidor. De esta forma aporta el valor de escalabilidad y seguridad a nuestro proyecto al usar esta jerarquía.

4.2.1.6. Nodo de control

Un elemento está compuesto por un microcontrolador Arduino junto con su Shield Ethernet, el cual posee control sobre varios elementos. Es este componente es el que se encarga de comunicarse con el servidor por la red Ethernet y quien gestiona los elementos conectados a él.

4.2.1.7. Sección

Conjunto de elementos, nodos y un servidor que están interconectado entre sí. Una sección englobara a todos los elementos que estarán presentes en el sistema.

4.2.1.8. Servidor

En el servidor es donde dispondremos del software para el control de nuestro sistema. Comúnmente será un ordenador con conectividad a internet. En este además del software de gestión, también dispondremos de la base de datos, aunque puede estar ubicado en otra localización. Adicionalmente ofrece una capa de seguridad extra ya que para acceder a los nodos, antes hay que acceder al servidor.

4.2.1.9. Área

Las áreas están delimitadas por decisión del cliente. Un área abarca una parte de la totalidad del terreno a cubrir con el sistema y estará compuesto por múltiples nodos y elementos. Un nodo puede contener elementos que se ubican en distintas áreas.

4.2.2. Ejemplo

A continuación se muestra una ilustración con un ejemplo de jardín donde nos encontramos con un entorno separado en tres áreas. Donde la área 1 cubre el camino de entrada a la casa y a la fuente, la área 2 representa a la fuente más la zona de jardín,

representada con un verde más oscuro, y la área 3 es la fuente y el camino de regreso a la casa.

Como se puede apreciar en la Ilustración 13, un nodo de comunicación controla uno o más nodos de control que están repartidos por todo el jardín. Puede mantener comunicación con nodos que se encuentran en distintas áreas como los nodos de comunicación 1 y 3.

Además debemos considerar que un nodo de control, también puede tener sus elementos en áreas distintas. Como ejemplo el nodo de control 1 que dispone de elementos en la área 1 y en la área 2.

Ilustración 13: Ejemplo casuística

En el caso concreto del control de iluminación, la siguiente ilustración muestra una sección de camino y como es controlado. Como se puede ver, disponemos de un nodo de control, el cual estará conectado en este caso por Ethernet un nodo de comunicación, al cual se le conectan 4 elementos sensores y 3 elementos actuadores. Conforme los sensores vayan detectando presencia, estos lo comunicaran al nodo quien dará la orden a los actuadores de activarse o no.

Por ejemplo, el sensor ID=1 y 2, cuando detecten presencia, el nodo mandara la orden de activación al actuador ID=5. Los sensores 2 y 3, activaran el actuador 6 y los sensores 3 y 4 activan el actuador 7.

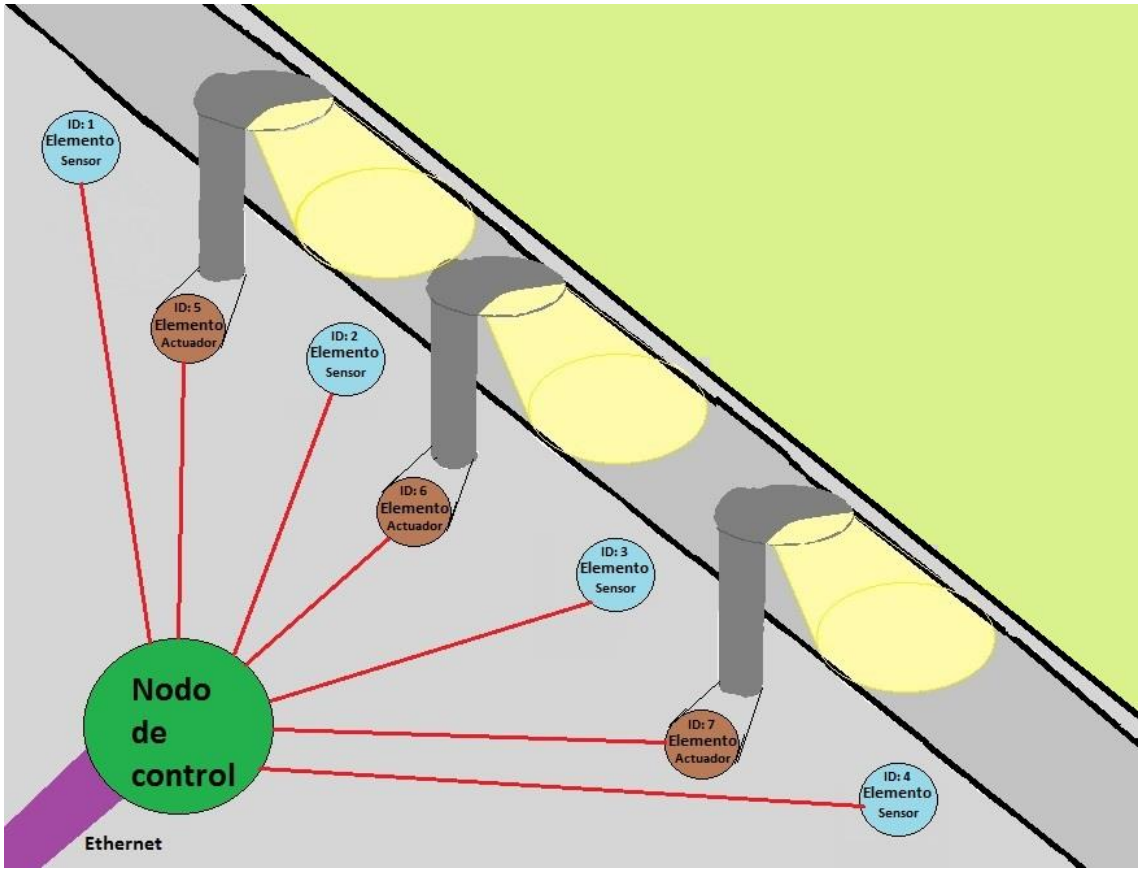

Ilustración 14: Ejemplo camino

Todo proceso de activación debe pasar antes por el nodo de control que, según su configuración, dará o no la orden de activación al actuador y además notificara al servidor de la detección de presencia en el espacio de interacción del elemento correspondiente. En este ejemplo tendremos 3 zonas de control pertenecientes al nodo, donde la primera zona estará formada por los espacios de interacción de los sensores 1 y 2 y del actuador 5. La segunda zona será con los sensores 2 y 3 y el actuador 6 y la tercera zona será con los sensores 3 y 4 y el actuador 7. Como es posible apreciar, un sensor y un actuador pueden pertenecer a múltiples zonas de control.

4.2.3. Descripción formal

A continuación se hace una explicación detallada de la descripción formal de proyecto presente. Pero antes se hace una sencilla explicación del significado de todos los símbolos que se puede encontrar en la ilustración.

1...*	De este objeto se tiene en la relación de 1 a infinito objetos
0...*	De este objeto se tiene en la relación de 0 a infinito objetos
1	De este objeto solo hay uno en la relación
	El objeto que tiene este símbolo en la relación, estará compuesto por objetos que están en el otro extremo de la relación
	Este símbolo significa que el objeto del que sale, heredara los atributos del objeto al cual está apuntando.

Tabla 16: Objetos UML

Esta es la descripción formal del proyecto. Ahora a continuación se ira detallando cada parte de la estructura mediante una ontología [14] que representa, en UML, el sistema de control [15] y [16].

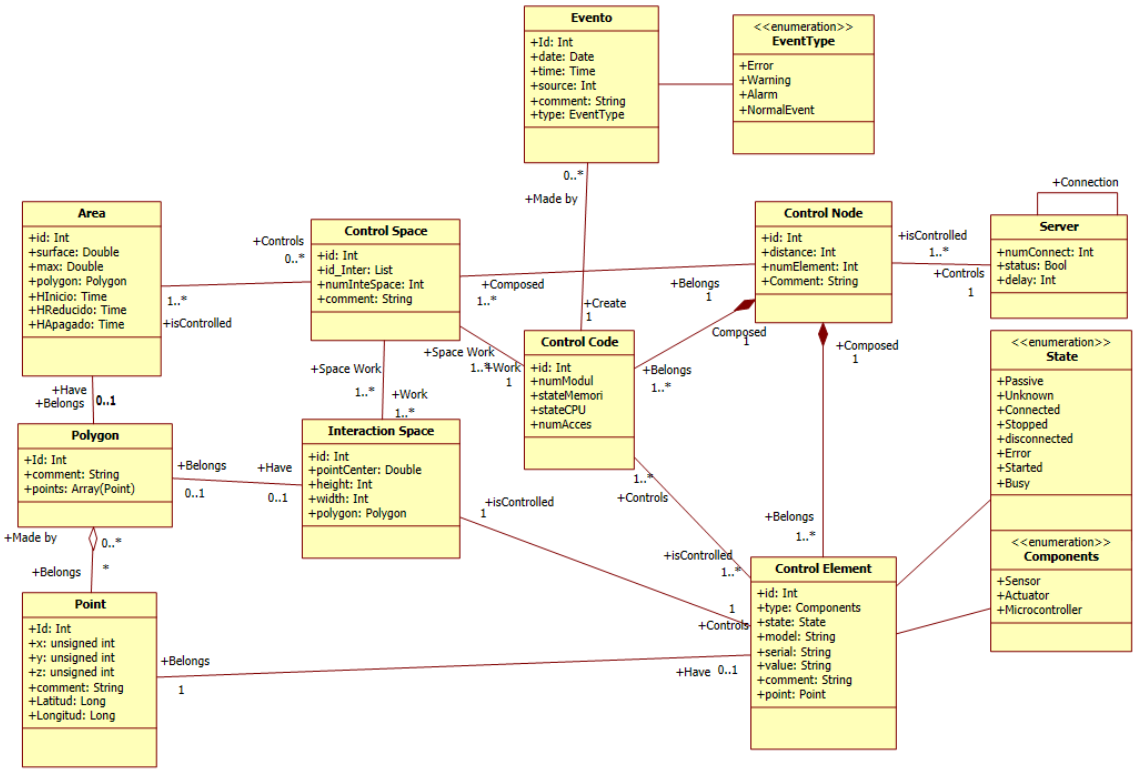

Ilustración 15: Diagrama del sistema

En esta sección podemos ver la relación ente “área” y “espacio de control” donde un area pueden tener ninguno o muchos espacios de control y un espacio de control puede tener una o muchas áreas relacionadas. Con esto detallamos que obligatoriamente un espacio de control estará como mínimo en un área mientras que un área puede no tener ningún espacio de control, ya que puede, por especificación del cliente, no querer sensorizar esa área. A su vez un espacio de control puede ubicarse en múltiples áreas. Por ejemplo el espacio de control de un nodo, puede encontrarse ubicado en la entrada de la casa estando simultáneamente tanto en el área del camino principal como en el área de acceso al jardín.

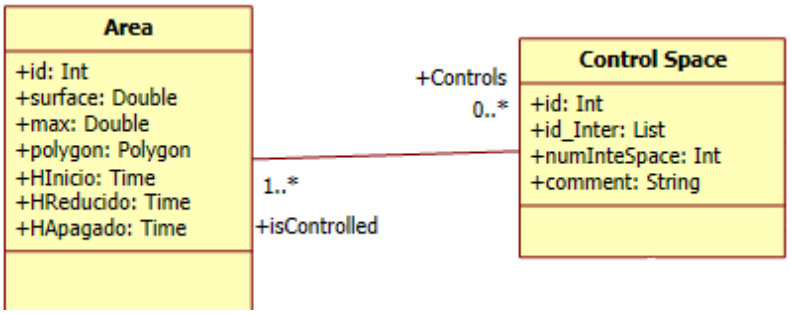

Ilustración 16: Descripción formal, área - espacio de control

En la Ilustración 17 podemos visualizar la relación entre los objetos “Elemento de control” y “espacio de interacción”, donde un espacio de interacción está formado por solo un elemento de control y este solo tiene un espacio de interacción. Esto se explicaría con el sencillo razonamiento de que un sensor no puede estar físicamente controlando dos espacios de iteración distintos y simultáneamente.

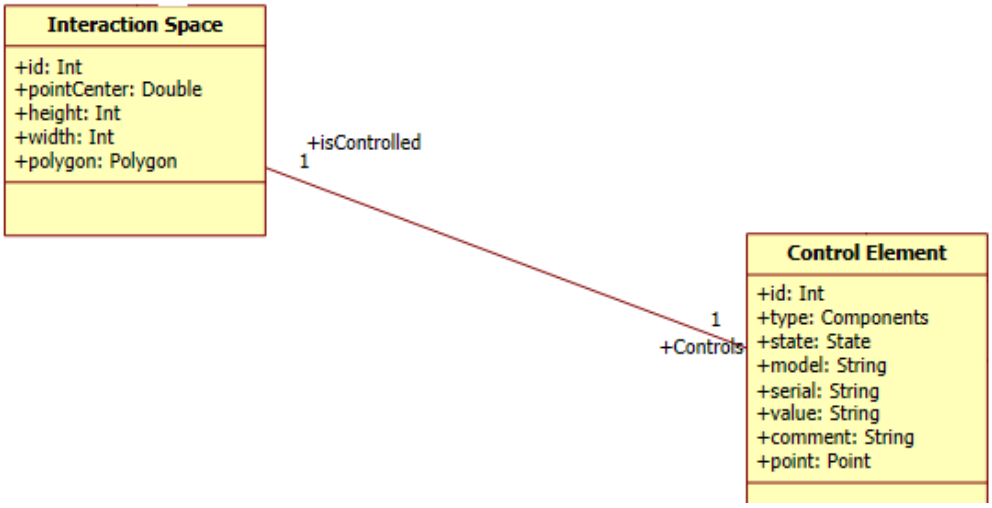

Ilustración 17: Descripción formal, espacio de interacción – elemento

En la siguiente ilustración se puede apreciar la relación entre un sensor y su espacio de interacción.

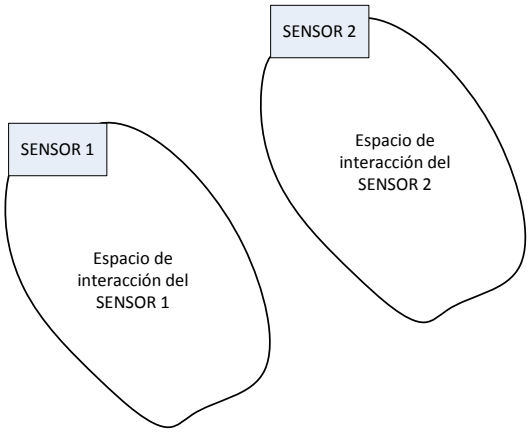

Ilustración 18: Relación sensor - Espacio de interacción

A continuación, en la Ilustración 19, podemos ver una relación entre los objetos "espacio de control" y "espacio de interacción". Un objeto espacio de control puede tener 1 o muchos espacios de interacción mientras y un espacio de interacción también está en uno o muchos espacios de control. Esta relación se explica con que como un elemento tiene solo un espacio de interacción, como se vio en la ilustración anterior, cada elemento puede formar parte de distintos espacios de control y cada espacio de control tendrá múltiples elementos que forman el espacio de control. Un ejemplo de este caso sería el espacio de control formado por los elementos que controlan la entrada al área del camino principal. El espacio de control estaría formado por la agregación de todos los espacios de interacción que forman ese espacio de control, aunque alguno de los elementos puede pertenecer simultáneamente a otro espacio de control.

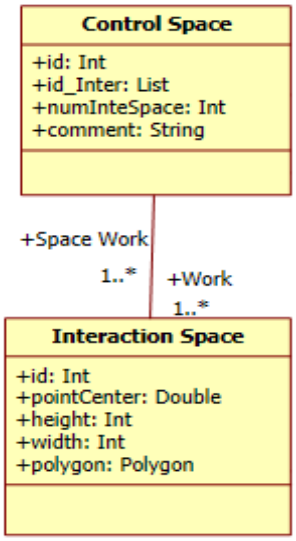

Ilustración 19: Descripción formal, espacio de control - espacio de interacción

En la Ilustración 20 podemos apreciar la relación entre un espacio de control y un nodo de control. Un nodo de control estará compuesto por 1 o infinitos espacios de control mientras que cada espacio de control pertenecerá únicamente a un solo nodo de control. Un ejemplo de este caso es que un espacio de control que es una composición de espacios de interacción que cada uno es de un elemento. Estos elementos solo pueden tener solo una conexión directa a un nodo de control, haciendo imposible que esté conectado a múltiples nodos.

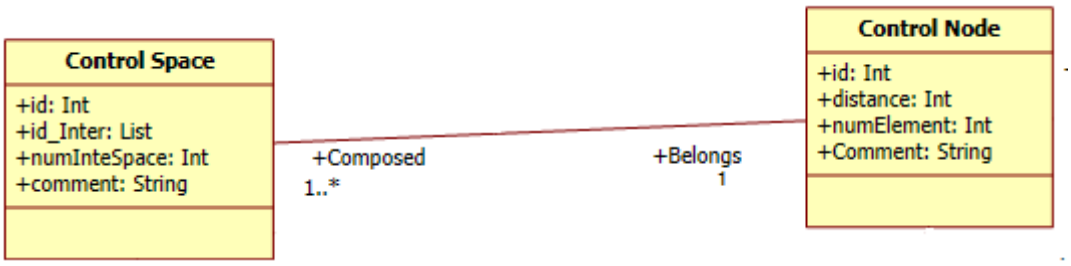

Ilustración 20: Descripción formal, espacio de control - Nodo de control

La siguiente sección de la descripción formal muestra la relación entre el nodo de control y servidor donde un servidor puede tener uno o muchos nodos de control pero estos solo tienen un servidor. Como ejemplo de este caso, es la conexión que mantienen los nodos con el servidor donde cada nodo solo se conecta a un servidor pero este gestiona el funcionamiento de múltiples nodos.

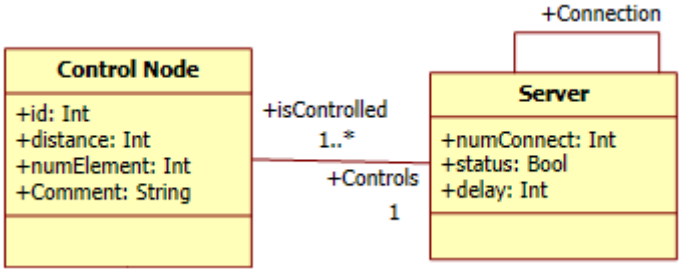

Ilustración 21: Descripción formal, nodo de control – servidor

En esta ilustración siguiente podemos la relación que tiene un área y un espacio de interacción con un polígono. Donde en ambos casos un polígono pertenece a un área o un espacio de interacción mientras que un polígono puede pertenecer a uno o a ningún área o espacio de interacción. Los polígonos tienen su relación con áreas y espacios de interacción modelado a una OR exclusiva donde un polígono solo puede estar relacionado con uno. Adicionalmente, un polígono está compuesto por una serie de puntos que estos están ubicados en coordenadas tridimensionales. Un polígono estará compuesto por una serie de puntos mientras que un punto puede pertenecer a un polígono o ninguno, ya que como se verá en la próxima ilustración, un punto puede pertenecer a un elemento. Estas relaciones nos sirven para ubicar en el espacio las áreas y los espacios de interacción.

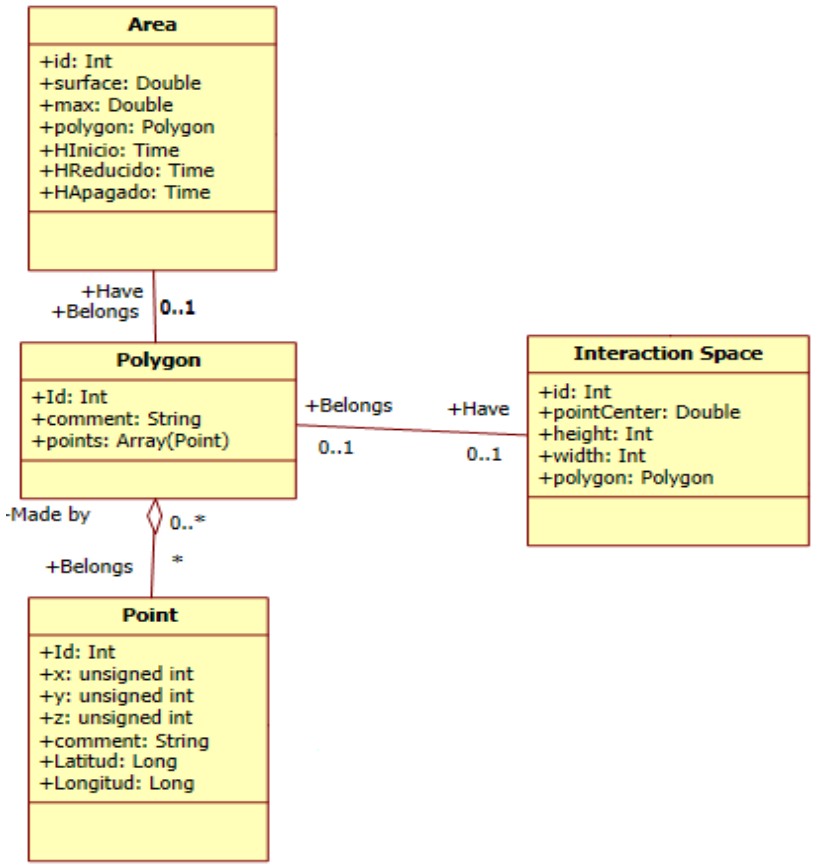

Ilustración 22: Relación Área/Espacio de interacción con polígono y punto

Como se comentó en la ilustración anterior, un punto también puede no formar parte de un polígono y pertenecer a un elemento, indicando la ubicación de este en el sistema. Además un elemento puede no tener un punto, ya que por motivos del entorno, sea imposible obtener la ubicación de este.

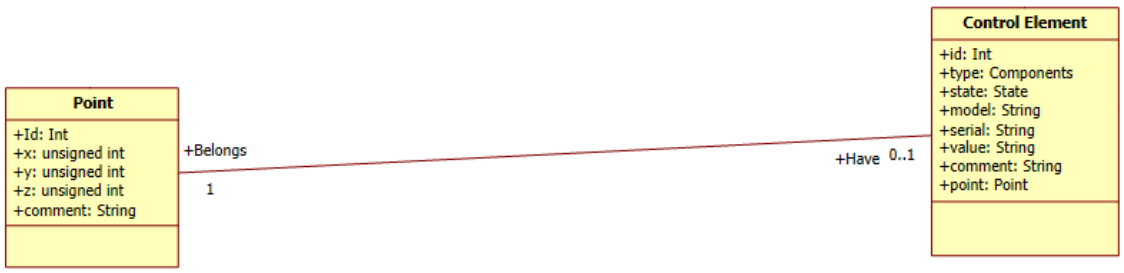

Ilustración 23: Relación Punto con elemento de control

A continuación vemos la relación entre el nodo de control y el elemento de control. Un elemento de control formara parte exclusivamente de solo un nodo de control mientras que un nodo de control controlara múltiples elementos de control. Esta relación es debida a que cada elemento solo podrá conectase físicamente a un nodo de control.

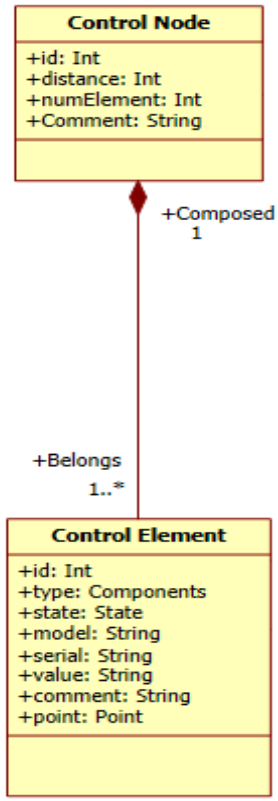

Ilustración 24: Relación nodo de control con elemento de control

El objeto código de control, representa el código implementado en cada Arduino. Por ese motivo, un nodo de control solo tendrá un código de control aunque un código de control este en múltiples nodos. Este código de control será capaz de controlar a múltiples elementos de control y cada elemento puede ser controlado por múltiples código de control ya que un mismo sensor puede afectar a distintas funcionalidades aplicadas en el nodo de control. Adicionalmente, el código de control funcionara sobre los múltiples espacios de control a los que este asociado el nodo de control.

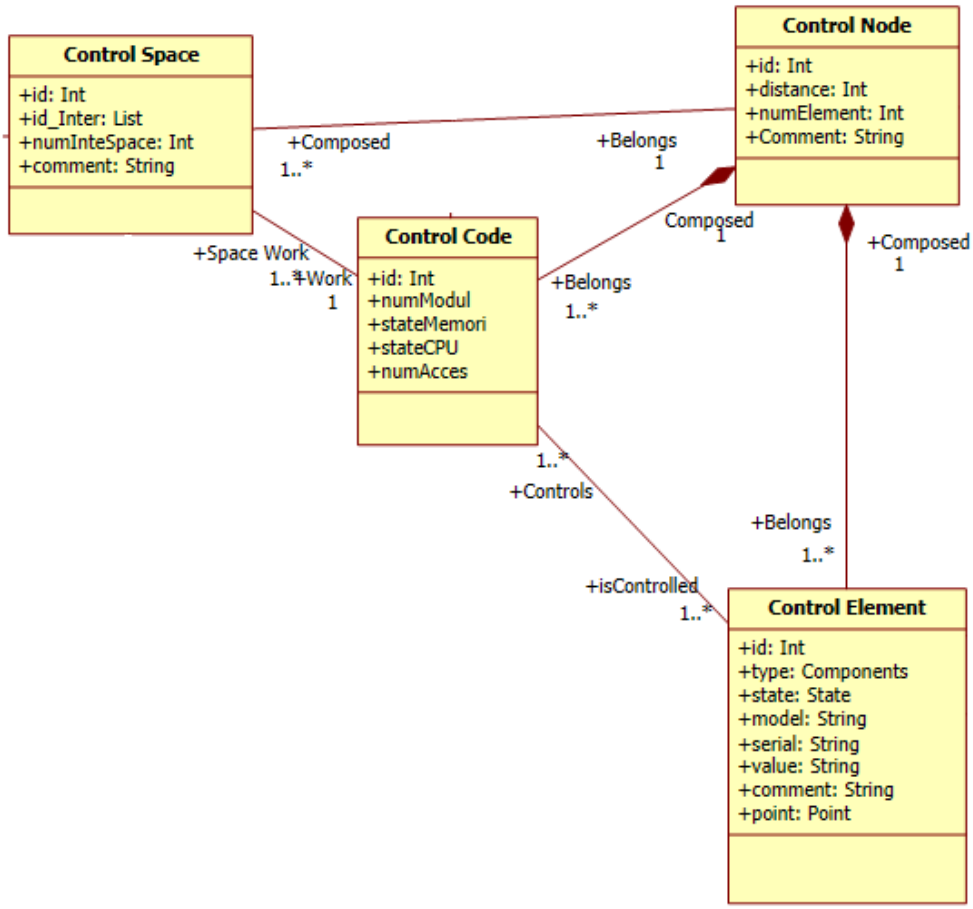

Ilustración 25: Relación código de control con espacio de control, nodo de control y elemento de control

En la Ilustración 26 tenemos la relación existente entre el código de control y evento. Esta relación es debido a que los códigos de control serán capaces de emitir diferentes tipos de eventos. Un código de control puede tener 0 o infinitos eventos mientras que un evento pertenece exclusivamente a un código de control.

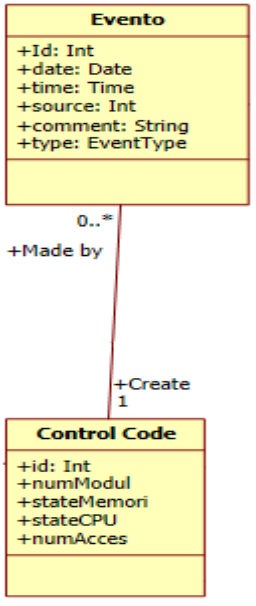

Ilustración 26: Relación código de control con evento

Con este fragmento enumeramos todos los tipos que puede ser un evento. Cada evento puede ser un error, una alerta, una alarma o un evento normal.

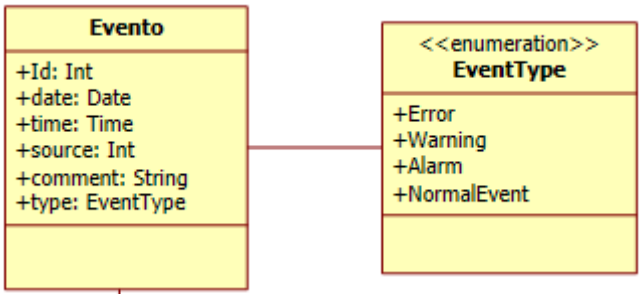

Ilustración 27: Enumeración Tipo del objeto evento

En la última ilustración del apartado, vemos las enumeraciones de estado y componentes que puede tener un elemento de control.

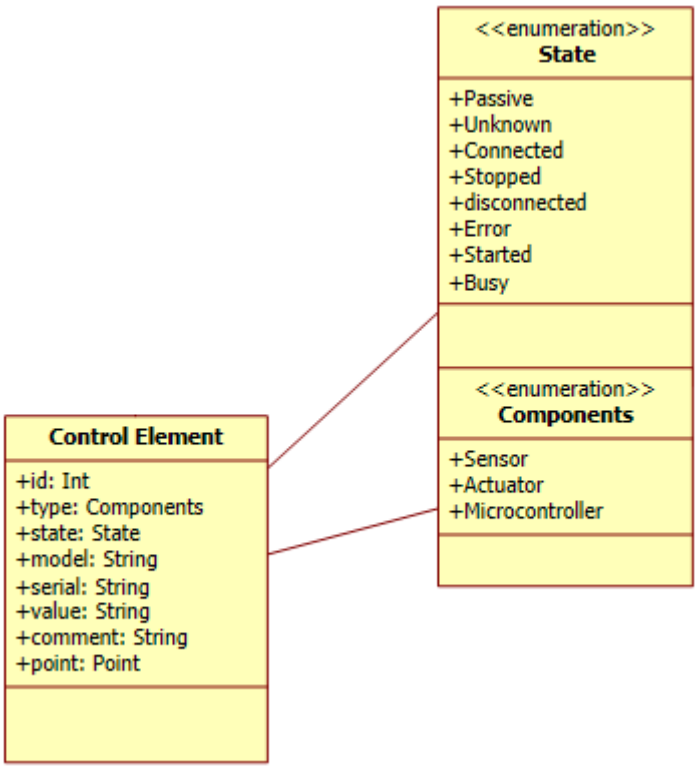

Ilustración 28: Enumeración estado y componentes del elemento de control

Como estado, puede tener los valores de pasivo, desconocido, conectado, parado, desconectado, error, iniciado y ocupado. En el campo componentes, puede tener los valores de sensor, actuador y microcontrolador.

4.2.4. Funcionalidad

A continuación, se va a realizar una breve demostración del funcionamiento del sistema con un sencillo ejemplo.

Empezando desde un sistema correctamente configurado e instalado sin casos excepcionales de funcionamiento, empezamos con una persona recorriendo un camino controlado por un nodo de control y sus sensores y actuadores. Al entrar en el rango de interacción del sensor de presencia, este lo notifica al nodo de control y este ensambla el mensaje adecuado y lo manda al servidor.

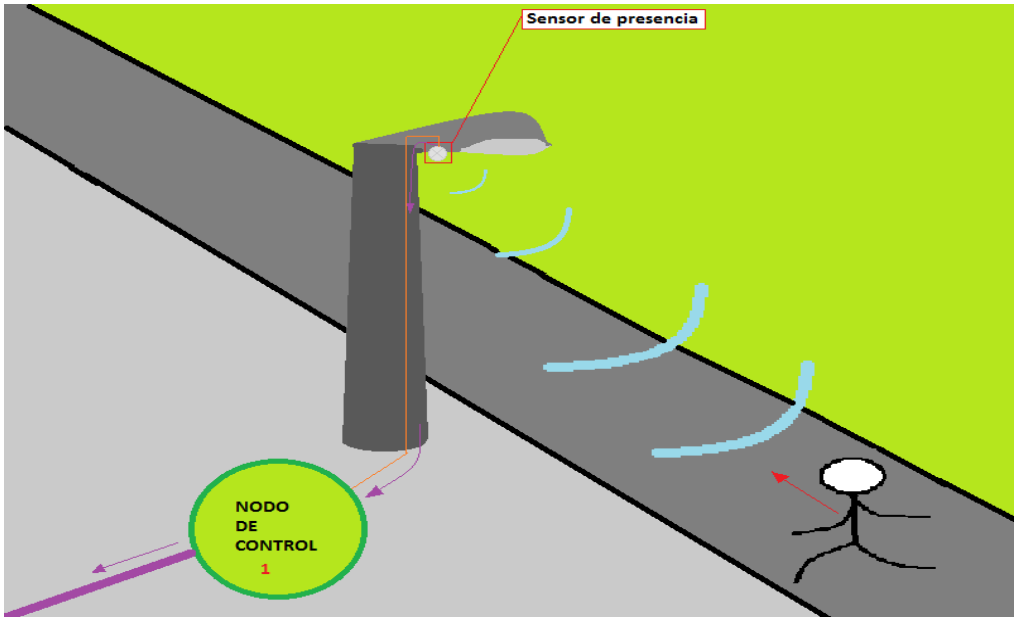

Ilustración 29: Ejemplo funcionalidad 1

Cuando el servidor recibe un paquete, este es desensamblado, obteniendo un código y un cuerpo. El código recibido en este caso, será el código de presencia detectada. El cuerpo del mensaje indicara el nodo que manda el mensaje, el sensor que detecta presencia y el valor de ese sensor. El servidor cambiara el valor del sensor en la base de datos. Adicionalmente el servidor generara un nuevo evento con origen el nodo y sensor recibidos en el cuerpo del mensaje..

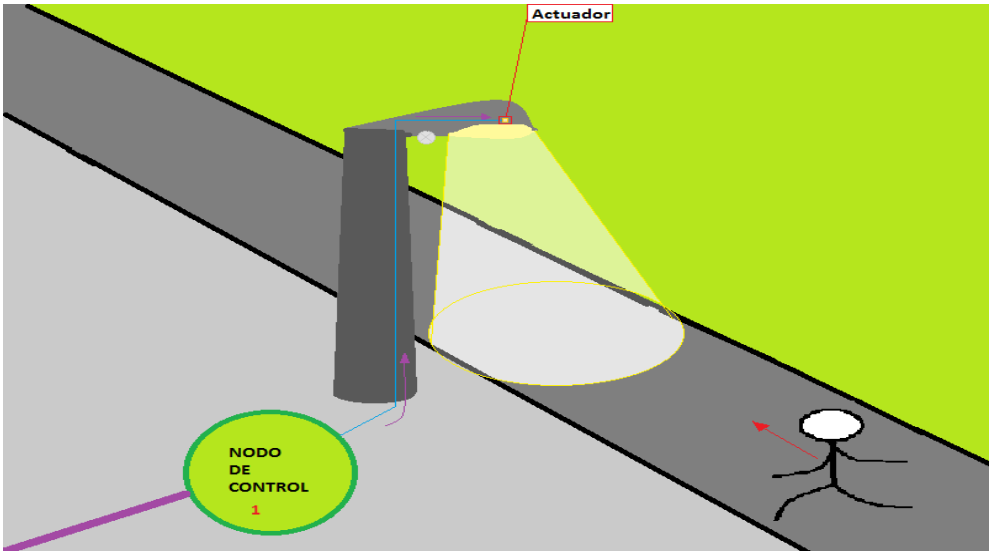

Ilustración 30: Ejemplo funcionalidad 2

Mientras sucede todo lo referente al servidor, el nodo de control ya ha dado la orden al actuador para que se encienda la luminaria. Este paso puede no producirse si se configuran los nodos para que requieran la orden del servidor para encender la iluminación, en cuyo caso no se encenderá la farola hasta recibir la orden.

A continuación se presentara un ejemplo del funcionamiento de las áreas. Como se puede ver, la sección controlada por el sistema está compuesta por múltiples áreas con tres horarios cada una. Tendrán la hora de encendido, que es a partir de qué hora la iluminación se encenderá, la a hora de luminosidad reducida, que es cuando la iluminación pasa a un 30% de potencia si no detecta presencia, y por último la hora de apagado que sin importar si detecta presencia o no, la iluminación estará apagada.

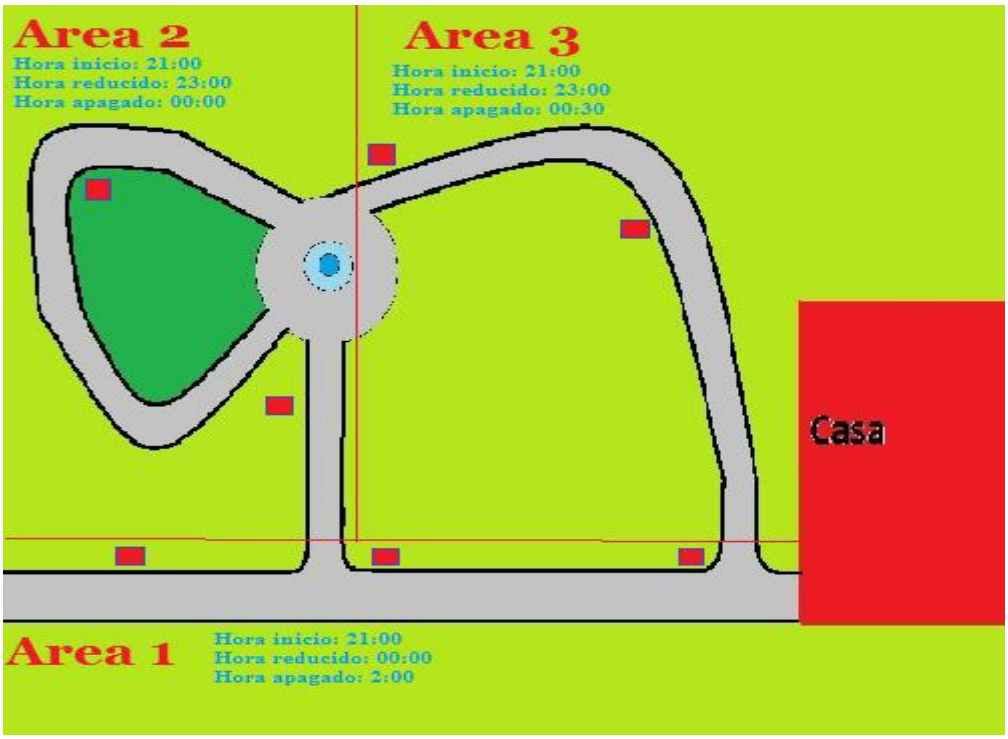

Ilustración 31: Ejemplo áreas 1

En el caso práctico de que a las 23:30 alguien saliese de la casa a pasear por el jardín, se encontraría con que todas las áreas menos la primera están en estado de consumo reducido y vería que las farolas por las que pasase se irían iluminando para ofrecerle la correcta iluminación. Las farolas del área 1 seguirían al 100% aun si no detecta presencia.

Si esta misma persona decidiese pasear a las 00:45, se encontraría con que las farolas del área 1 se encuentran en bajo consumo y se encenderían con su paso, mientras que las farolas de las áreas 2 y 3 seguirán estando apagadas aunque detectasen su presencia.

Una vez pasadas las 2:00, todas las farolas estarán apagadas y no se encenderán al paso de nadie.

Cabe destacar que sin importar el horario en que se encuentre una farola, si detecta presencia, seguirá notificándolo al servidor que este ira registrando en eventos todos los sucesos.

4.3. Especificación Hardware

A continuación, se hace un análisis detallado de todos los elementos hardware que se emplean en este proyecto.

4.3.1. Arduino Uno rev.3

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. El hardware está basado en el chip ATmega328. Dispone de 14 entradas/salidas digitales (De las cuales 6 pueden usar modulación por pulso de ancho), 6 entradas analógicas, velocidad de reloj de 16MHz, conexión USB, entrada de alimentación Jack y un botón de reset.

Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

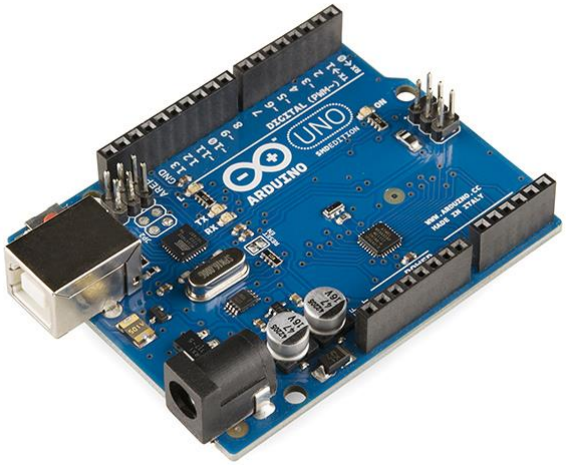

Ilustración 32: Arduino Uno Rev3

Especificaciones técnicas	
Microcontrolador	ATmega 328
Voltaje de funcionamiento	5 V
Voltaje de entrada (Recomendado)	7-12 V
Voltaje de salida (Limites)	6-20 V
Pines digitales E/S	14
Entradas analógicas	6
Corriente continua de Pin E/S	40 mA
Corriente continua de pin de 3.3V	50 mA
Memoria flash	32 KB *
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 Hz

Tabla 17: Características técnicas Arduino Uno Rev3

* Se usan 0.5 KB de memoria para el gestor de arranque.

4.3.2. Arduino Shield Ethernet

La Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Está basada en el chip Ethernet Wiznet W5100. Este chip provee de una pila de red IP capaz de TCP y UDP. Además es capaz de soportar hasta cuatro conexiones de sockets simultáneas.

Ilustración 33: Shield Ethernet

Dispone de unos conectores que permiten la conexión de otras placas encima y así apilarlas. La Ethernet Shield hará uso de los pines digitales 10, 11,12 y 13 para comunicarse con el chip W5100.

La Shield provee de un conector Ethernet estándar Rj45, un botón reset que resetea tanto al chip W5100 como a la placa Arduino, una serie de Leds para información de uso de conexión y un slot para una tarjeta SD.

4.3.3. Sensores

4.3.3.1. Sensor PIR

Un sensor PIR (Passive Infrared Sensor) es un sensor electrónico que mide la luz infrarroja (IR) radiada de los objetos situados en su campo de visión.

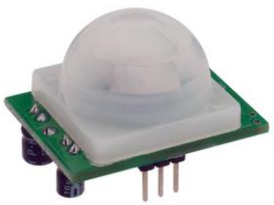

Ilustración 34: Sensor PIR

Todos los objetos con una temperatura por encima del cero absoluto emiten calor en forma de radiación infrarroja. Por lo general, esta radiación es invisible para el ojo humano, ya que irradia en longitudes de onda infrarrojas, pero puede ser detectado por dispositivos electrónicos diseñados para tal propósito. El término pasivo, en este caso, se refiere al hecho de que los dispositivos PIR no generan o irradian cualquier energía para fines de detección y no hay que confundir con que un sensor PIR sea un detector de calor.

4.3.3.2. Sensor de ultrasonidos

Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias de hasta 8m. El sensor emite un sonido y mide el tiempo que la señal tarda en regresar. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración.

Ilustración 35: Sensor de ultrasonidos

Estos sensores trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, colores, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo han de ser deflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

4.3.4. Actuadores

4.3.4.1. Relé

El relé es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

Ilustración 36: Relé

El relé permite que una vez recibida una señal digital del Arduino, este se accione y deje pasar la corriente para el encendido de una luminaria.

4.3.4.2. Luminaria

Una luminaria es lo que comúnmente se conoce como una farola o lámpara. En esta es donde tendremos puesto la bombilla que ilumine el camino. En este proyecto haremos uso de luminarias LED.

Ilustración 37: Luminaria

Al usarse tecnología LED, ya podemos asegurar de inicio un ahorro energético de un 20% en comparación a iluminaciones tradicionales. Además usando un programa de uso eficiente de energía como el que usaremos en este proyecto podemos garantizar un ahorro de aproximadamente en un 80% de la factura de la luz.

4.4. Especificación Software (UML)

En la próxima ilustración, se analiza el funcionamiento del sistema de detección, que engloba los requisitos funcionales de detección de presencia y fin de detección de presencia.

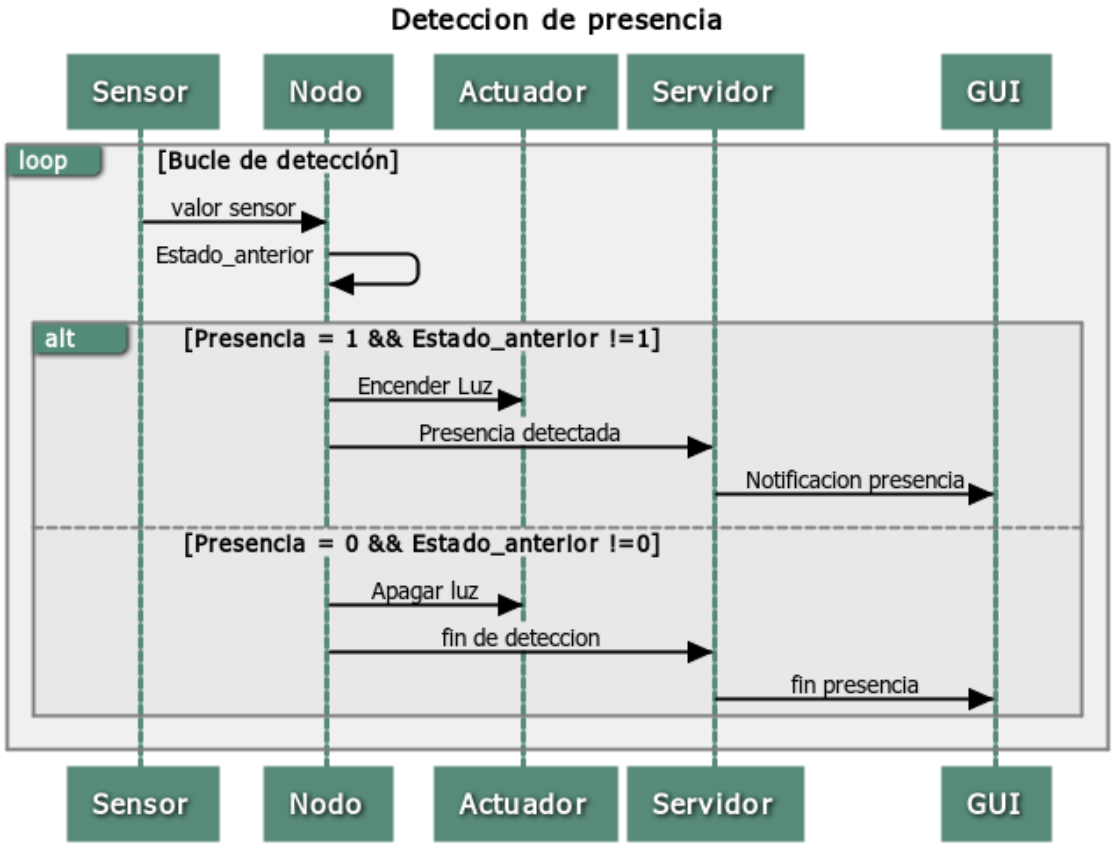

Ilustración 38: Sistema de detección de presencia

Podemos ver que los sensores están continuamente mandando valores al nodo. Este procesa esos valores recibidos y según estos, la configuración que tenga el nodo y el estado de detección anterior que tuviese el nodo, se considerara si ha detectado presencia o no. En el caso de que se considere que si se ha detectado presencia y en el estado anterior no había detectado presencia, este dará la orden al actuador de encender la iluminación y a la vez lo notificara al servidor, donde este reflejara el cambio en la GUI de la aplicación de gestión.

Este proceso se da de forma idéntica para cuando no se detecta presencia y en el estado anterior si se había detectado. La iluminación siempre permanecerá estable en el estado en que este, siempre que no se produzca un cambio de estado en el nodo.

A continuación tenemos otra ilustración donde podemos ver el funcionamiento del sistema ante los casos funcionales relacionados con las zonas, donde según la hora, la configuración de los nodos cambia, afectando así al funcionamiento de las luminarias.

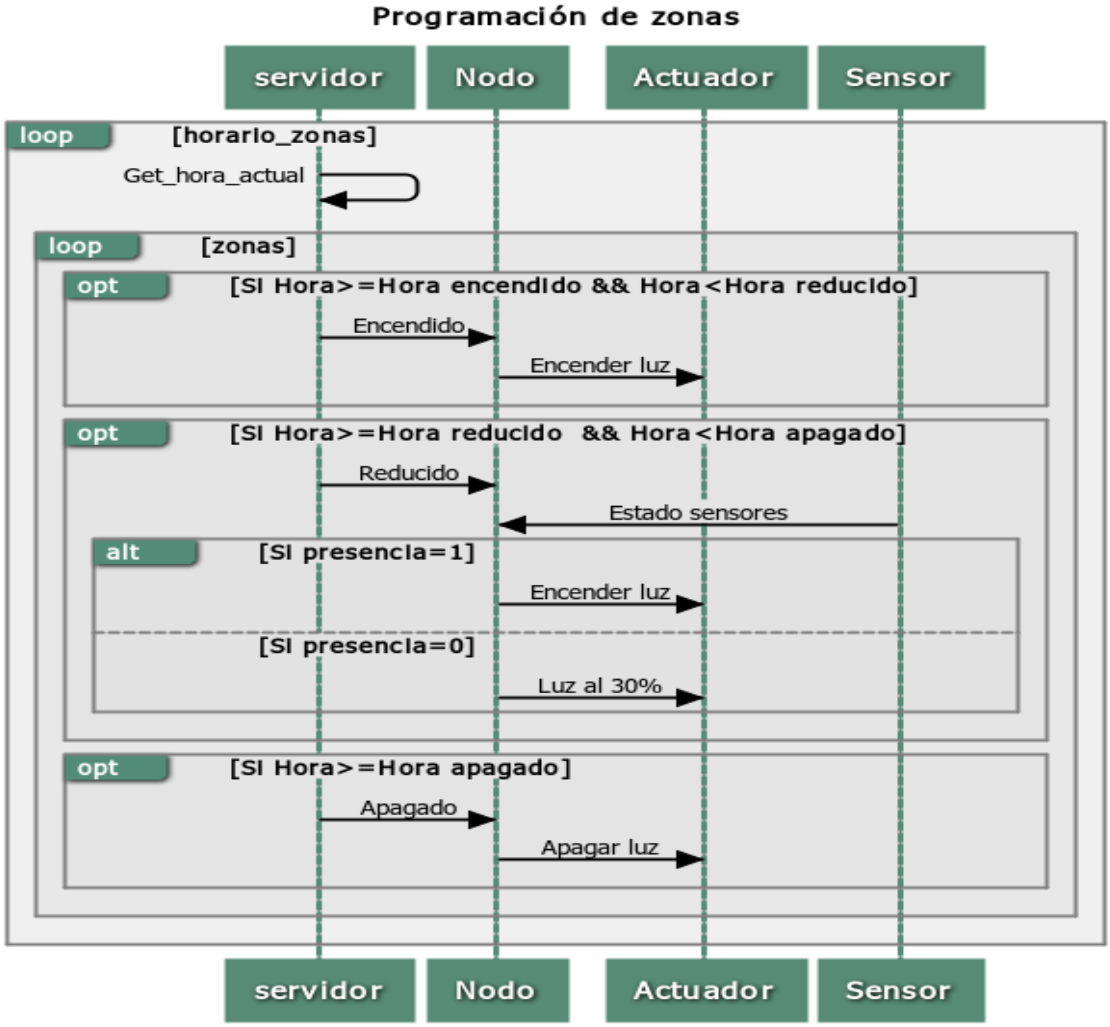

Ilustración 39: Sistema de programación de zonas

El servidor es el encargado de controlar la hora actual. Una vez obtenida la hora, revisa los horarios de todas las zonas. Comprueba en que rango se encuentra de los configurados para cada zona. Si la hora actual se encuentra en el primer rango, le dará al nodo la orden de encendido, cambiando así su configuración y haciendo que el actuador de la luminaria mantenga encendida está al 100% de potencia sin importar la presencia detectada. Igualmente el nodo seguirá notificando al servidor de la presencia.

Si se encuentra en el segundo rango horario, la configuración del nodo será la de la ilustración anteriormente explicada, donde la luminosidad será de un 100% en caso de detectar presencia y de un 30% si no hay presencia. En la última opción, sirve para dar la orden de que todos los nodos de la zona apaguen las luminarias. Estos nodos seguirán informando del estado de presencia al servidor.

La próxima ilustración detallara el funcionamiento del caso funcional de detección parcial.

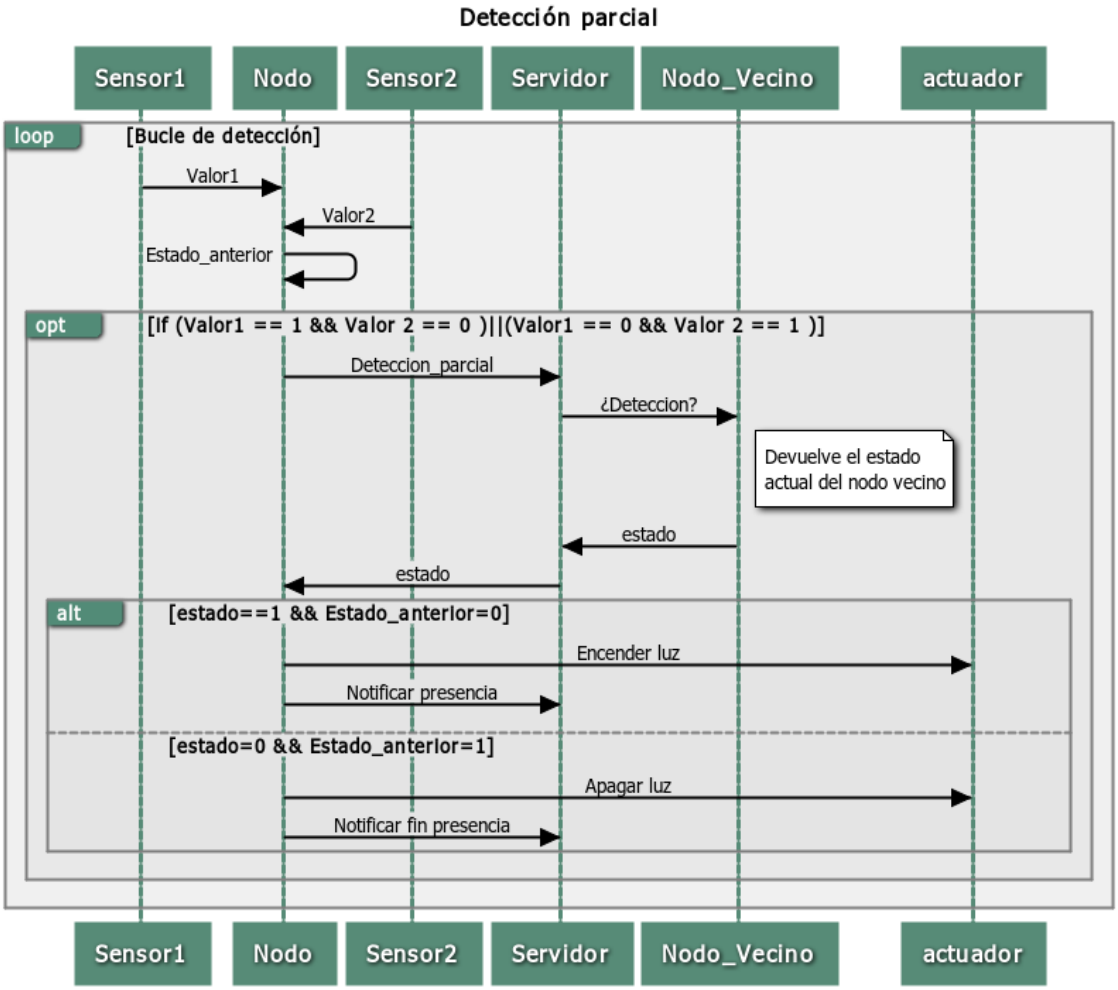

Ilustración 40: Sistema de detección parcial

En el caso de un nodo con múltiples sensores, que formen parte del mismo espacio de control, los cuales no todos detectan presencia. El nodo recibe los valores de esos sensores y al darse el caso especial de que no todos detectan presencia, el nodo de control realiza una consulta al servidor donde este le preguntara a los nodos vecinos si detecta presencia en su superficie de control. Una vez obtenidas las respuestas, en el caso en que algún vecino detecte presencia, el servidor responderá de forma afirmativa, con lo que el nodo, dependiendo de su estado anterior, no hará nada, encenderá o apagara las luces.

4.5. Conclusión

A lo largo de todo el capítulo se ha hecho una descripción formal del sistema y la especificación de software a través de UMLs. Específicamente la descripción formal del sistema ha sido de tal complejidad y ha requerido tal cantidad de trabajo que incluso ha

sido posible la presentación de un artículo en las jornadas de automática de valencia que se realizan del 3 al 5 de septiembre del 2014.

Ambos elementos, serán necesarios para la implementación e implantación del sistema que se detallara en el próximo capítulo.

5. Implementación e implantación del sistema

5.1. Introducción

En este capítulo implementaremos todo lo diseñado en el capítulo anterior. Se mostrara un diseño de prototipo, la interfaz del software y fragmentos de código correspondientes a los UMLs diseñados en el capítulo anterior.

5.2. Implementación Hardware

5.2.1. Sensores de presencia y actuadores implicados

A continuación se detallan los sensores de presencia y actuadores implicados en la implementación del sistema.

Primero empezamos con el sensor de ultrasonidos, modelo HC-SR04. El sensor se alimenta con una corriente de 5 voltios, permitiendo estar conectado directamente al microcontrolador. Su rango de alcance es de 5 metros con una resolución de 1 cm.

Ilustración 41: Sensor de ultrasonidos

Para medir la distancia, el sensor emite un pulso de un ancho o tiempo de 10 µs como mínimo por el pin trigger. Al mismo tiempo debemos monitorizar por el pin hecho hasta recibir la señal y así calcular la distancia. La siguiente ilustración detalla el funcionamiento del sensor ultrasonidos

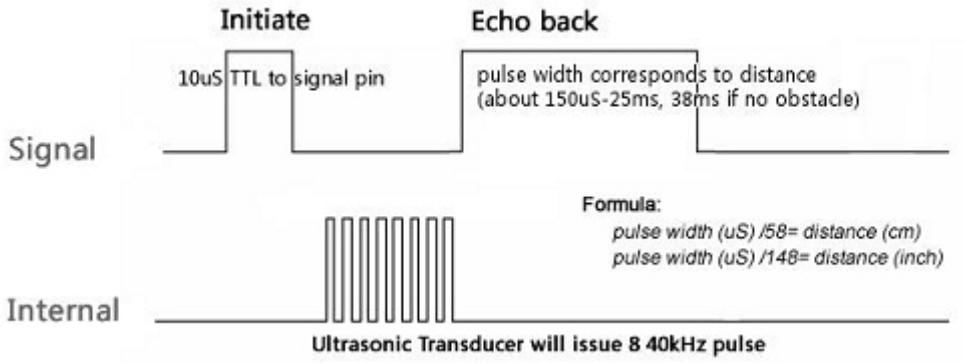

Ilustración 42: Funcionamiento ultrasonidos

El siguiente sensor es el sensor de presencia PIR, modelo HC-SR501. Este sensor esta alimentado con 5 voltios, permitiendo estar conectado directamente al microcontrolador. Su alcance de detección es de 7,7 metros y su rango es de 140º

Ilustración 43: Sensor pasivo de infrarrojos

Su funcionamiento está basado en el calor que emite todo aquello con temperatura mayor que el 0 absoluto. Se le considera pasivo ya que no emite ningún tipo de radiación y solo recibe la radiación infrarroja que emitimos con el calor.

Por ultimo tenemos el relé de 5 voltios, que nos permitirá activar la iluminación a orden del microcontrolador.

Ilustración 44: Relé

Un relé está compuesto comúnmente por 4 pins mínimo, dos a cada lado. Dos de esos pines es donde se conectara el aparato que queremos controlar a través del relé, estando este por en medio. De los otros dos pines, uno se conectara a tierra del microcontrolador y el ultimo se conecta a una señal output del microcontrolador. Llegado a este punto, solo necesitamos dar una señal de 5 voltios a este último pin y activaremos el relé, cortocircuitando el otro par de pir.

5.2.2. Prototipo

En la siguiente ilustración se muestra el prototipo de un nodo de control en el cual disponemos de un sensor de presencia PIR, un sensor de presencia de ultrasonidos, un relé, un microcontrolador arduino y un Shield Ethernet.

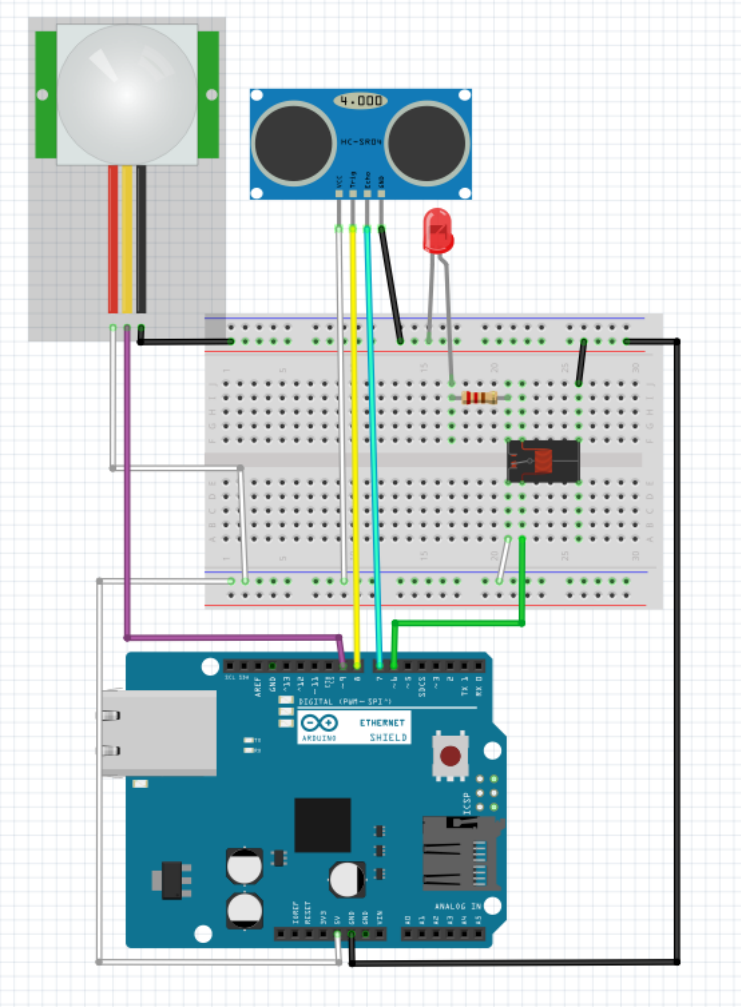

Ilustración 45: Esquema prototipo

El Shield Ethernet es necesario para que el nodo pueda mantener comunicación con el servidor y el resto de nodos. Además para simular el funcionamiento de una luminaria, usaremos un led en su lugar.

5.3. Protocolo de comunicación

A continuación se detalla el protocolo que se ha usado para la comunicación entre los nodos y el servidor. La estructura que siguen los mensajes que van desde un nodo a l servidor es la siguiente

Código	#	ID Nodo de control	#	ID espacio de control	#	Mensaje

Tabla 18: Empaquetado de mensajes Nodo->Servidor

Usamos las "#" como separador entre los distintos campos que tiene un mensaje. La longitud del mensaje es siempre la misma y si se da el caso de que un mensaje proviene de un nodo de control y afecta a todos los espacios de control que tiene el nodo, en el campo "Espacio de control" recibiremos un "-1". En el caso de que sea el servidor el que mande un mensaje a un nodo, la estructura del mensaje omitirá los campos intermedios mandando mensajes con la estructura de la siguiente tabla.

Código	#	Mensaje

Tabla 19; Empaquetado de mensajes Servidor -> Nodo

En algunos casos, ni siquiera dispondrá de mensaje, pudiendo este estar en blanco.

A continuación en la siguiente tabla se detallan todos los códigos de operación que distinguirán las distintas peticiones. Hay que tener en cuenta que la comunicación es bidireccional, por lo que cada código vale para un sentido de la comunicación pero para el otro no funcionan.

Del servidor al nodo		**Del nodo al servidor**	
Código	**Explicación**	**Código**	**Explicación**
1	Inicialización del nodo. Con el mensaje se le pasa al nodo su identificador.	100	Mensaje de OK
2	Pregunta estado de iluminación	101	Mensaje de ERROR
3	Fija la iluminación al horario programado ya que el nodo pertenece a un área. Mensaje =1 horario encendido, =2 horario reducido, =3 horario apagado y =0 libre de horarios	103	Mensaje de presencia. Si el mensaje =1, presencia detectada. Si =0, fin de detección.

Tabla 20: Códigos de comunicación

Actualmente son estos los códigos implementados aunque perfectamente se pueden ampliar mucho más los códigos que se deseen usar.

5.4. Implementación Software

5.4.1. Interfaz de configuración

A continuación en se explican los motivos por los cuales una interfaz gráfica es importante para el proyecto.

La interfaz del programa está compuesta principalmente por tres pestañas. Cada una se encarga de 1 o varios casos funcionales. La primera pestaña es la de “Visualizar”. Esta interfaz le otorga al usuario la posibilidad de visualizar los cambios que se producen en su sistema, haciendo más amigable y más fácil de comprender el software para el usuario.

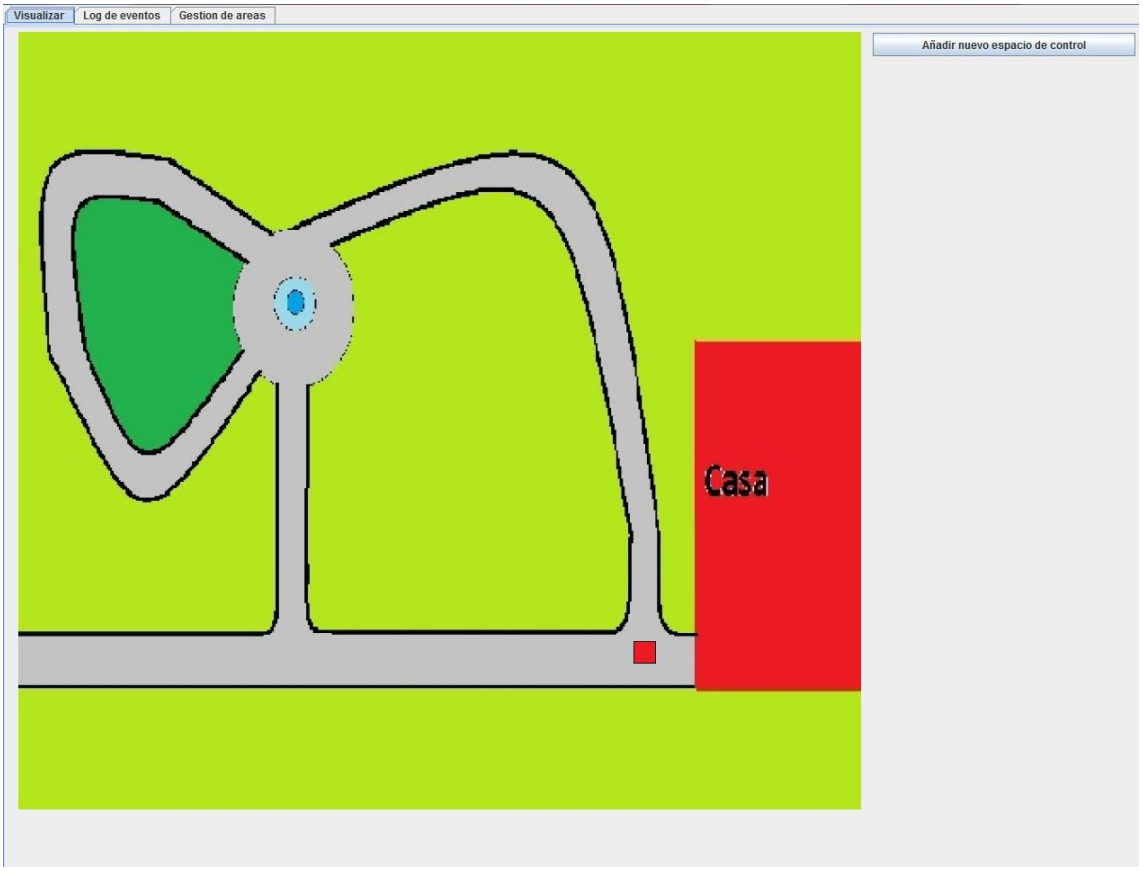

Ilustración 46: Interfaz de configuración – pestaña visualización

La segunda pestaña de la interfaz se encarga del registro de eventos del sistema. Su presencia se vuelve muy útil ya que queda todos registrados en el sistema y se puede revisar en cualquier momento.

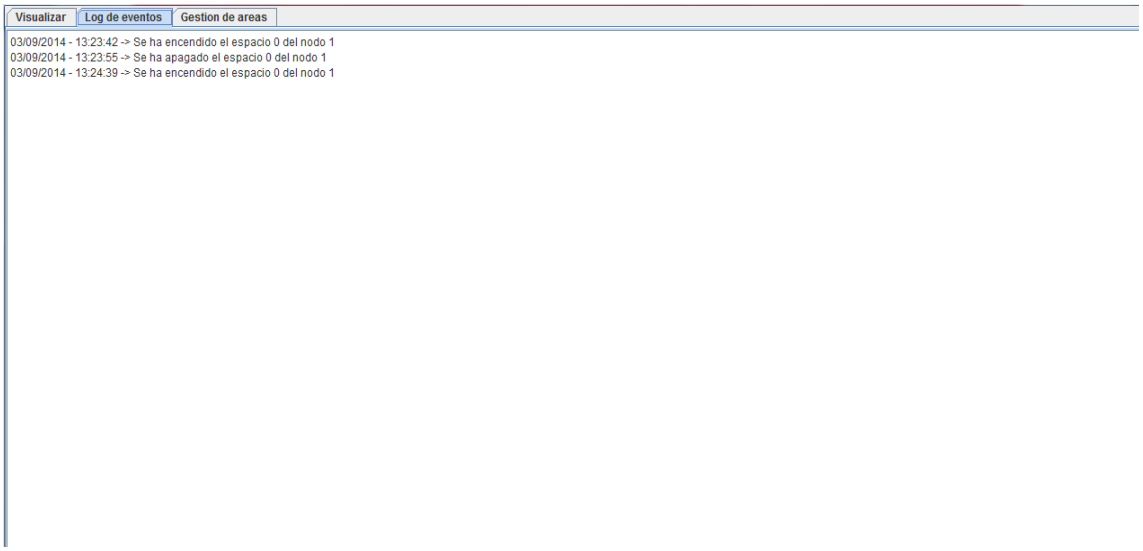

Ilustración 47: Interfaz de configuración - pestaña log de eventos

Por último, tenemos la pestaña de configuración de las áreas donde podemos configurar todo lo referente a las áreas. Podemos añadir o eliminar espacios de control de un área, podemos fijar sus horarios.

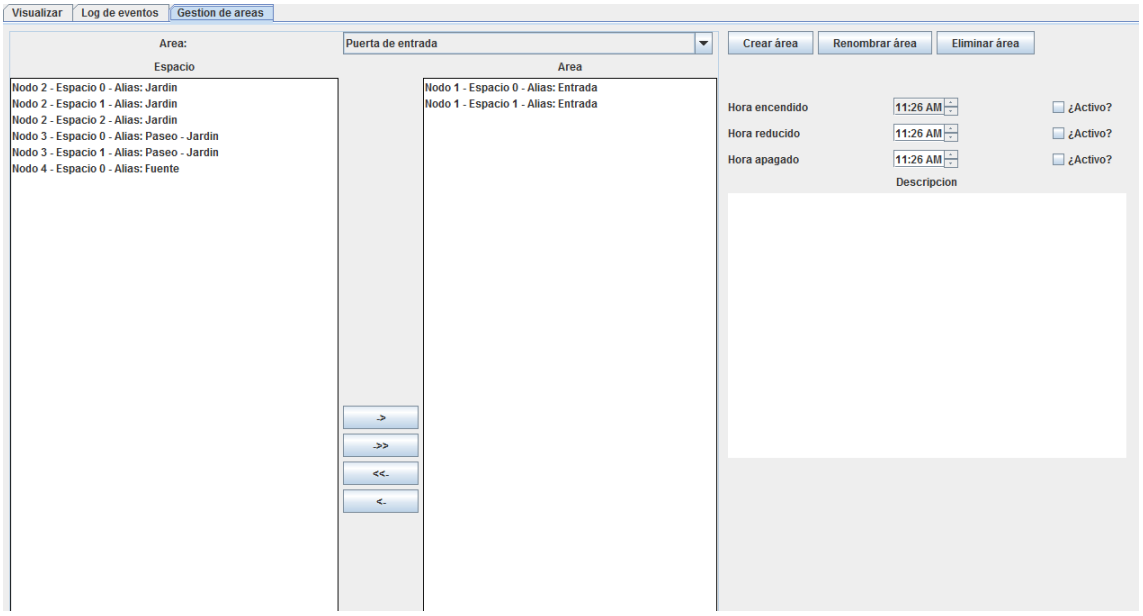

Ilustración 48: Interfaz de configuración - Pestaña gestión de áreas

Todas estas pestañas unidas a una gran interfaz son muy importantes para el funcionamiento de la aplicación, ya que la realización de estas tareas sin estas interfaces sería un trabajo muy tedioso o incluso imposible para un usuario común.

5.4.2. Códigos de implementación del control

En este apartado se podrán observar múltiples fragmentos de código que se pueden encontrar en la programación de los microcontroladores Arduino

5.4.2.1. Decodificación de mensajes

En la primera ilustración, podemos observar el código de decodificación de mensajes. Todos los paquetes enviados por Ethernet seguirán el patrón “código#mensaje”, donde cada código significa una operación distinta.

```
void decoder (String msg)
{
  String cod="";
  String mensaje="";
  int c;
  for (int i=0; i<=msg.length();i++)
  {
    if(msg[i]>=32 && msg[i]<=127)//Con este filtro, evitamos caracteres raros
    {
      if(msg[i]!=35) //usamos # para separar codigo de mensaje
        cod=cod+msg[i];
      else
      {
        c=i+1;
        break;
      }
    }
  }
  for(int i=c; i<=msg.length();i++)
  {
    mensaje=mensaje+msg[i];
  }
  switch(cod.toInt())
  {
```

Código 1. Decodificación de mensajes

Una vez extraído el código, este se cotejara con un switch donde dependiendo del código, llamara a una función u otra dándole el cuerpo del mensaje.

5.4.2.2. Detección de presencia

El siguiente fragmento del código es el encargado de la detección de presencia. En este ejemplo, se considera que se detecta presencia cuando el sensor PIR detecte presencia y el sensor ultrasonidos detecte a 15 centímetros o menos. Todo esto puede ser modificado para aumentar la distancia de presencia y/o la detección de solo un sensor.

```
if(ID1!=-1)
{
 if(estado==1)
   digitalWrite(led,HIGH);
 else if (estado==3)
   digitalWrite(led,LOW);
  mensaje="103#";
  mensaje.concat(ID1);
  dist_uS=get_dist(1);
  int pir =digitalRead(pirPin1);
  if (dist_uS <=15 && dist_uS > 0 &&  pir == HIGH)
  {
    contador1=0;
    if(!deteccion1)
      {
        deteccion1=true;
        if(estado==0 || estado ==2)
          digitalWrite(led, HIGH);
        mensaje=mensaje+"#1;";
        client.println(mensaje);
      }
  }
  else
    contador1++;
  if (contador1 >= 10 && deteccion1 ==true)
  {
    deteccion1=false;
    mensaje=mensaje+"#0;";
    client.println(mensaje);
    if(estado!=1)
      digitalWrite(led, LOW);
  }
}
```

Código 2. Detección de presencia

Lo primero que hace este fragmento de código es obtener los valores de los ultrasonidos y del PIR. Si la presencia está confirmada se inicia a 0 una variable integer que nos servirá de contador, se procederé a encender la iluminación que está conectada al pin LED y se mandara un mensaje al servidor con el aviso de detección, siguiendo el mismo patrón de mensaje que el explicado en el anterior fragmento de código. En el momento en que nuestros sensores dejen de detectar presencia, comenzara el contado a incrementar su valor hasta alcanzar 10, donde se da por terminada la detección de presencia dando como resultado el apagado de la iluminación y el aviso al servidor.

5.4.2.3. Horarios programados

En el fragmento de código anterior ya teníamos incluido el código de los horarios programados. Existe una variable llamada estado que es de tipo integer. Esta variable puede tener 4 valores posibles.

- 0: Significa que no hay ningún horario establecido y tendrá un funcionamiento normal.
- 1: Horario de encendido. La iluminación siempre estará encendida y nunca se apagara aunque deje de detectar presencia.

- 2: Horario reducido. La iluminación se encenderá si detecta presencia y se apagara si deja de detectarla.
- 3: Horario apagado. La iluminación siempre estará apagada sin importar si detecta presencia.

```
if(ID1!=-1)
{
 if(estado==1)
   digitalWrite(led,HIGH);
 else if (estado==3)
   digitalWrite(led,LOW);
  mensaje="103#";
  mensaje.concat(ID1);
  dist_uS=get_dist(1);
  int pir =digitalRead(pirPin1);
  if (dist_uS <=15 && dist_uS > 0 &&  pir == HIGH)
  {
    contador1=0;
    if(!deteccion1)
      {
        deteccion1=true;
        if(estado==0 || estado ==2)
          digitalWrite(led, HIGH);
        mensaje=mensaje+"#1;";
        client.println(mensaje);
      }
  }
  else
    contador1++;
  if (contador1 >= 10 && deteccion1 ==true)
  {
    deteccion1=false;
    mensaje=mensaje+"#0;";
    client.println(mensaje);
    if(estado!=1)
      digitalWrite(led, LOW);
  }
}
```

Código 3. Horarios programados

El código remarcado en amarillo es el que controla como debe de reaccionar el actuador que controla la iluminación dependiendo del estado en el que se encuentra.

5.4.3. Códigos de implementación de la interfaz

En este apartado se explicaran pequeños fragmentos de código del programa usado en el servidor para gestionar el resto del sistema. El programa de interfaz está compuesto por tres hilos que funcionan simultáneamente.

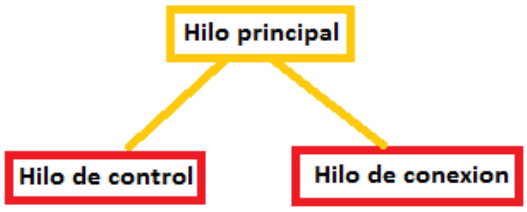

Ilustración 49: Hilos de interfaz

El primer hilo es el hilo principal encargado de la gestión de eventos que se produzcan en la interfaz. El segundo hilo tendrá el papel de recibir todos los mensajes que manden los nodos, así como mandar los mensajes que deban de llegar a un nodo. Y por último, el último hilo es el de control, que se encargara de gestionar todos los mensajes que recibió el hilo de control para así realizar las acciones pertinentes. También se encarga de ensamblar los mensajes que el hilo de conexión deberá de mandar a sus destinatarios.

Para este intercambio de información se hará uso de un objeto llamado "Listados" el cual ambos hilos tienen acceso a él.

Adicionalmente, el comportamiento del hilo de conexión será distinto del hilo de control. El hilo de conexión funcionara a través de un thread, mientras que el hilo de control funciona con un SwingWorker, ya que afectara a objetos presentes en la interfaz.

5.4.3.1. Recepción de mensajes

Fragmento de código perteneciente al hilo de conexión, este fragmento de código se centra específicamente en la recepción de mensajes.

```
public void run()
{

    System.out.println("Arrancando servidor | Connect");
    while ( cierre )
    {
        espacios=lista.get_espC();//volvemos a recoger el listado por si se ha añadido algun nodo nuevo
        for (Nodo i:lista.getListado_nodo())
        {
            cs=i.getSc();
            try {
                    b = new BufferedReader( new InputStreamReader ( cs.getInputStream() ) );
                    if(b.ready())
                    {
                        mensaje=b.readLine();
                        if(mensaje!="")
                        {
                            peticiones.add(mensaje);
                        }
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
}
```

Código 4. Recepción mensajes

El hilo se queda encerrado en un bucle escuchando continuamente todos los mensajes que reciba. Como el objeto nodo, tiene un atributo Socket, va recorriendo todos los sockets de todos los nodos de control implementados en el sistema.

Si existe mensaje en el socket, lo añade a un arraylist que existen en llamado peticiones. Este arraylist es accesible por el hilo de control. Cuando este listado es obtenido por el hilo de control, acto seguido es limpiado para que el hilo de control no procese un mismo mensaje varias veces.

5.4.3.2. Procesamiento de peticiones

El siguiente fragmento de código pertenece al hilo de control en el cual se procesan todos los mensajes recibidos desde el hilo de conexión.

```
public String doInBackground(){
    long start_time = System.currentTimeMillis();
    while(salida){
        long end_time = System.currentTimeMillis();

        if(end_time-start_time>=1000){
            start_time=end_time;
            peticiones=conexion.get_petitions();

            for(int i=0; i<peticiones.size();i++)
                publish(peticiones.get(i));
        }
        temporizacion();
    }
    return null;
}

protected void process(List<String> chunks){
    for (int i=0; i<chunks.size()-1;i++){
        String mensaje=chunks.get(i);
        String codigo,espacio, cuerpo, id;
        String[] spliteado=mensaje.split("#");
        if(spliteado.length==4) {
            codigo=spliteado[0].toString();
            id=spliteado[1].toString();
            espacio=spliteado[2].toString();
            cuerpo=spliteado[3].toString();
            cuerpo=cuerpo.substring(0,cuerpo.length()-1);
            //Falta añadir mas casos
            switch(Integer.parseInt(codigo))
            {
```

Código 5. Procesamiento de peticiones

El hilo se queda encerrado en bucle en el método “doInBackground”. Cada segundo, obtendrá el listado de peticiones del hilo de conexión. Una vez obtenido el listado, publicara todas las peticiones. Estas peticiones publicadas, será procesadas por el método “Process”. En este método, desglosara cada petición en código, id, espacio y cuerpo. Después pasara el código de operación por el switch donde dependiendo de este, se realizara una operación u otra.

Es este método el único que puede modificar objetos del hilo principal.

5.4.3.3. Tiempo programado

Este fragmento es el encargado de gestionar en qué estado se encuentran los espacios de control según a los tiempos programados a la zona a la que pertenece.

Después de procesar todas las peticiones existentes en el sistema, se procede a comprobar los tiempos establecidos a las áreas. Esta llamada se produce en el método “doInBackground”.

```
public String doInBackground(){
    long start_time = System.currentTimeMillis();
    while(salida){
        long end_time = System.currentTimeMillis();

        if(end_time-start_time>=1000){
            start_time=end_time;
            peticiones=conexion.get_petitions();

            for(int i=0; i<peticiones.size();i++)
                publish(peticiones.get(i));
            }
            temporizacion();
        }
        return null;
}
```

Código 6. Procesamiento de peticiones, llamada a temporización

El método "Temporización" es el encargado de revisar todas las áreas. Recorre las áreas y comprueba si se ha alcanzado alguno de los tiempos que se pueden fijar para cada área. Para comprobar si se ha alcanzado una hora fijada, se pasa la hora actual al método "toSeconds" que sacara el tiempo actual en segundos para así comprarlo con el resto de tiempos.

```
private void temporizacion()
{
    DateFormat df= new SimpleDateFormat("HH:mm:ss");
    Calendar cal = Calendar.getInstance();
    long actual = toSeconds(df.format(cal.getTime().toString()));
    long ini,red,fin;
    for(Zona i:lista.getListado_zonas())
    {
        ini=i.getHini();
        red=i.getHred();
        fin=i.getHapa();
        if(ini!=0 && actual>ini && ((actual-ini)<(actual-red)||(actual-ini)<(actual-fin)))
        {
            temporizacion(i,1);
        }
        if(actual>red && ((actual-red)<(actual-ini)||(actual-red)<(actual-fin)))
        {
            temporizacion(i,2);
        }
        if(actual>fin && ((actual-fin)<(actual-ini)||(actual-fin)<(actual-red)))
        {
            temporizacion(i,3);
        }
    }
}
private void temporizacion(Zona i, int stat)
{
    for(EspacioC e: i.getEspacio())
    {
        String mensaje="3#"+e.getNodo().getId()+"#"+e.getId()+"#"+stat;
        conexion.send(e.getNodo().getSc(), mensaje);
    }

}
```

Código 7. Temporización

Una vez se cumpla que se ha alcanzado la hora de inicio, reducido o fin, se llamara al método sobrescrito "temporización" recibiendo como atributos la zona y un número Integer que significa en qué estado se pondrán los nodos. Recorreremos todos los nodos de la área mandando el mismo mensaje con la estructura "3#id nodo# id espacio de acción#estado".

5.5. Uso y mantenimiento

En este apartado se explica el funcionamiento de la aplicación y como usarla. Partiendo desde la Ilustración 46, vamos a ir explicando todo lo que se puede realizar.

5.5.1. Creación de un nuevo espacio de control

Este es el proceso para la creación de espacios de control que aún no estén configurados en la aplicación. Para ello debemos de estar en la pestaña "Visualizar" y ahí podemos ver el mapa de nuestro sistema y un botón

Ilustración 50: Botón de añadir nuevo espacio de control

Al pulsar el botón nos pedirá que hagamos clic en el mapa indicando donde estará ubicado el espacio de control que queremos añadir. Acto seguido nos preguntara si el espacio de control pertenece a un nodo de control existente o es un nuevo nodo de control. En caso de ser un nuevo nodo de control, pedirá la dirección IP del nodo, un alias y el id que le pertenece a ese nodo. Si has seleccionado "nodo existente", te pedirá que le digas cual es el nodo al cual pertenece seleccionando su alias en el combobox.

Una vez configurado el nuevo espacio de control, ya aparecerá un recuadro como cualquiera de los que sale en la siguiente ilustración.

Ilustración 51: Colores de los espacios de control

Dependiendo del color, significara que el espacio de control está en un estado u otro, siendo verde presencia detectada, rojo presencia no detectada y amarillo espacio de control configurándose.

5.5.2. Log de eventos

Dentro de la propia aplicación, podremos visualizar un log de eventos del sistema. En la propia pestaña "log de eventos" podemos ver de primeras los eventos.

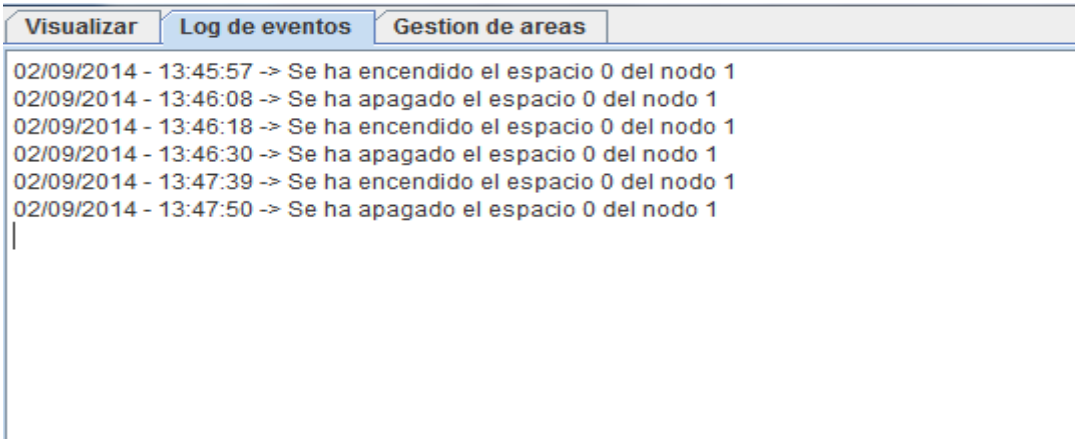

Ilustración 52: Log de eventos

En este log nos mostrara la fecha del evento y que es lo que se ha producido. Estos eventos también serán almacenados en la base de datos.

5.5.3. Gestión de áreas

Aquí es donde podremos gestionar las áreas. Dentro de la pestaña “Gestión de áreas”, podemos acceder para gestionar todo lo referente a áreas.

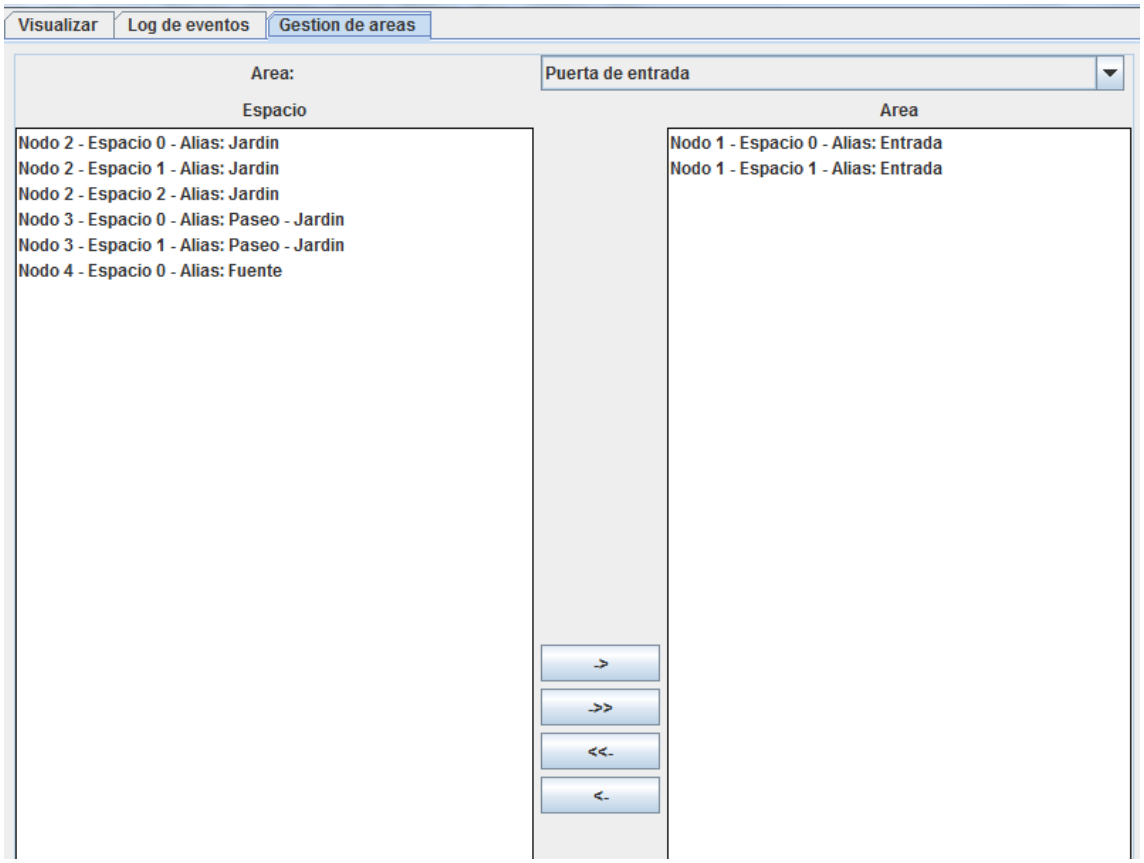

Ilustración 53: Gestión de áreas 1

Podemos seleccionar el área en el combobox de arriba y abajo en dos listados nos saldrán todos los espacios de control. Al listado izquierdo, están todos los espacios que no pertenecen al área y al lado derecho los que sí. En medio tenemos 4 botones con las funcionalidades de:

- Añadir espacio seleccionado de la lista izquierda a los espacios pertenecientes al área.
- Añadir todos los espacios al área.
- Eliminar todos los espacios del área.
- Eliminar el área de control seleccionada de la lista derecha del área.

A la derecha de este fragmento de interfaz, podemos crear una nueva área, eliminar el área actual o editarla.

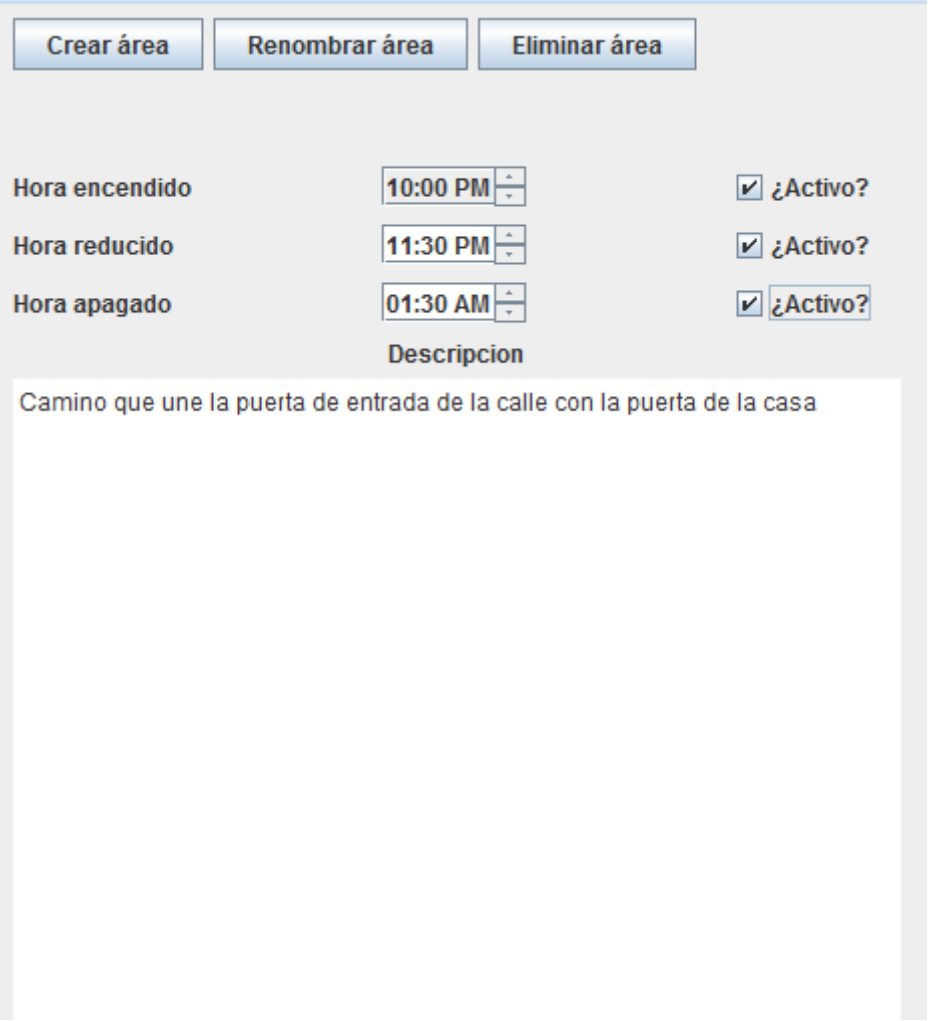

Ilustración 54: Gestión de áreas 2

Podemos configurar los horarios de encendido, reducido o apagado y si queremos que estén activas. Además podemos añadir una pequeña descripción a nuestra área.

5.6. Conclusiones

A lo largo del capítulo se ha explicado distintos fragmentos de código tanto para el microcontrolador como en el servidor. También se ha detallado en el funcionamiento de la interfaz que esta estará en el servidor.

Cabe destacar el desarrollo de un protocolo de comunicación entre el servidor y el Arduino.

6. Conclusiones

6.1. Introducción

Es en este capítulo donde se recopila todo lo elaborado en el proyecto y se extraen las aportaciones y las ampliaciones que podría recibir el proyecto.

6.2. Aportaciones

A continuación se analizaran todas las aportaciones realizadas durante el desarrollo del proyecto, planteándose por tanto todos los aspectos de mayor innovación.

Como primera aportación del proyecto, está el estudio de todos los sistemas con una funcionalidad similar a la expuesta en el proyecto. De esta forma, el estudio puede servir para ubicar proyectos similares que guardan aspectos en común con estos productos.

Por otro lado, se ha llevado a cabo una Especificación de Requisitos de un sistema que emplea conjuntamente conexión Ethernet, java, bases de datos y hardware de control. Las alternativas aquí presentadas no son habituales en los diferentes sistemas que podemos encontrarnos y esta especificación puede ser utilidad para el desarrollo de proyectos similares al presente, pudiendo aprovecharse elementos aquí presentados.

Adicionalmente se ha diseñado un sistema integral que emplea comunicación en red (Ethernet), servicios de persistencia (Bases de datos), un entorno grafico (Java) y hardware de control (microcontrolador Arduino).

A consecuencia del proyecto, se ha diseñado una ontología de control la cual ha supuesto tal dificultad y complejidad para el proyecto, que ha permitido la publicación de un artículo en las XXXV jornadas de automática de valencia que se dieron del 3 al 5 de septiembre del 2014.

Finalmente, y como elemento más resaltado, se ha aportado un sistema de iluminación de jardín inteligente programable completamente funcional. La iluminación está controlada gracias a la placa Arduino junto a un Shield Ethernet, sobre la que se conectarán sensores y actuadores. Mientras disponemos de una aplicación de monitorización y gestión del sistema junto con una base de datos que nos aporta la persistencia para la topología del sistema.

1.1. Ampliaciones

El sistema muestra múltiples aspectos que pueden ser ampliados según las necesidades de los usuarios. A continuación se enumeran algunas aplicaciones posibles.

- Un sistema de control de acceso a zonas. Pudiendo especificar que ciertos nodos de control actúen de control de acceso, para así después poder obtener estadísticas de uso de las distintas zonas que se hayan en el sistema.
- La interfaz del software de gestión ubicada en un periférico táctil, pudiendo ser este portable y así gestionar el sistema desde cualquier zona del sistema, mientras llegue la conexión inalámbrica o tenga una toma de cable.
- Habilitar una interfaz de gestión a través de página web y de aplicación móvil, pudiendo así gestionar el sistema desde cualquier dispositivo.
- Programar patrones de encendido en la iluminación para así simular el paso de una persona, dando la apariencia de que la casa y el jardín está habitado, creando así un efecto disuasorio para ladrones. Adicionalmente poder realizar esta tarea a través de una página web, pudiendo así cambiar los patrones, haciendo que se mas difícil que posibles ladrones se percaten de los patrones.

- La adición de nuevos tipos de sensores y/o actuadores posibilitando que el sistema sea más estable y robusto a errores y falsas detecciones. También se incluyen versiones mejoradas del microcontrolador arduino o microcontroladores similares como la Raspberry pi. Con esta ampliación se pueden añadir nuevos tipos de elementos, que no tengan nada que ver con el sistema actual, permitiendo añadir nuevas funcionalidades al sistema.
- Cambio de la iluminación led por iluminación led RGB, permitiendo cambiar la luz de las áreas según valores que desee el usuario como la temperatura, la previsión del tiempo o la hora actual.

Estas son algunas de las ampliaciones que se le podrían aplicar al proyecto expuesto en este trabajo.

7. Referencias

[1] Arduino. Página principal del proyecto Arduino. www.arduino.cc (visita junio 2014)

[2] Intel Galileo. Página principal del proyecto Intel Galileo gen 2. http://www.intel.es/content/www/es/es/do-it-yourself/galileo-maker-quark-board.html (visita agosto 2014)

[3] Raspberry pi. Página del proyecto Raspberry. http://www.raspberrypi.org/(visita agosto 2014)

[4] Bernaras, A; Laresgoiti, I; Bartolome, N.; Corera, J., "An ontology for fault diagnosis in electrical networks," Intelligent Systems Applications to Power Systems, 1996. Proceedings, ISAP '96, International Conference on , vol., no., pp.199,203, 28 Jan-2 Feb 1996. doi: 10.1109/ISAP.1996.501068

[5] Sánchez P, Sánchez M, Jiménez F, Rosique B, Álvarez A, Iborra. A framework for developing home automation systems: From requirements to code. The Journal of systems and software 2011; 84(6):1008-1021.

[6] Dorf, R.C., Bishop, R.H. Modern Control Systems, 11th Edition, Prentice Hall. 2008

[7] TVlight http://www.tvilight.com/ (Visitada Agosto 2014)

[8] Delft university, holland http://www.tudelft.nl/ (visita junio 2014)

[9] E-Street http://www.e-streetlight.com/ (Visitada Agosto 2014)

[10] LUIX http://www.iluminacionluix.com (Visitada Agosto 2014)

[11] Banzi, Massimo. Getting Started with arduino. " O'Reilly Media, Inc.", 2009.

[12] Metcalfe, R. M., Boggs, D. R. (1976). Ethernet: distributed packet switching for local computer networks. Communications of the ACM, 19(7), 395-404.

[13] Booch, G., Jacobson I., Rumbaugh, J. The UML Specification Document. Rational Software Corp., 1997

[14] Sowa, J. F. (1995). "Top-level ontological categories". International Journal of Human-Computer Studies 43 (5-6 (November/December)): 669–85

[15] Marcos, M., Estévez, E., Gangoiti, U., Sarachaga, I., & Barandiarán, J. (2004). UML modelling of industrial distributed control systems. In Proceedings Sixth Portuguese Conference on Automatic Control (pp. 7-9).

[16] Poza, J. L.; Posadas, J. L. & Simó, J. E. (2009), Adding an Ontology to a Standardized QoS-Based MAS Middleware., in Sigeru Omatu; Miguel Rocha; José Bravo; Florentino Fernández Riverola; Emilio Corchado; Andrés Bustillo & Juan M. Corchado, ed., 'IWANN (2)' , Springer, , pp. 83-90 .

ANEXO: Artículo presentado a las XXXV Jornadas de Automática 2014

Explicación:

Debido a la gran dificultad y complejidad de la ontología, se ha podido contemplar que es perfectamente utilizable por cualquier sistema similar. Por tal motivo, se ha realizado y presentado un artículo para las jornadas de automática 2014.

Los coautores José-Luis Jiménez García, José-Luis Poza Lujan y Eduardo Munera y Raul Simarro surgen de la necesidad de llevar la ontología al artículo. Dada mi inexperiencia en la elaboración de artículos y a la falta de tiempo por las actividades lectivas y por la elaboración del resto de aspectos del proyecto, ellos se encargaron de sintetizar la ontología que he desarrollado en este proyecto de forma que pudiese crear el artículo. Adicionalmente hubo varias reuniones donde nos reunimos todos y terminamos de desarrollar aspecto que aún estaban sin completar.

A continuación, esta anexado el articulo presentado.

PROPUESTA DE ONTOLOGÍA PARA EL CONTROL DE ENTORNOS EXTERIORES

Miguel Juan-Monter, Jose-Luis Jimenez-Garcia, Jose-Luis Poza-Lujan, Eduardo Munera, Raúl Simarro
Escuela Técnica Superior de Ingeniería Informática (ETSINF), Instituto de Automática e Informática Industrial (ai2). Universitat Politècnica de València (UP), Camino de Vera s/n, 46022 Valencia, Spain
{mijuamon, jojigar1}@inf.upv.es, {jopolu, emunera, rausifer}@ai2.upv.es

Resumen

El control distribuido inteligente de sistemas en entornos exteriores tiene ciertas dificultades dada la extensión y la agresión que el sistema tiene dadas las condiciones naturales, como lluvia o temperaturas variables, a los que estás expuestos los componentes. Para controlar los entornos distribuidos en exteriores se requiere que las características de los sistemas distribuidos (modularidad, escalabilidad, etc.) sean muy robustas. Para proporcionar robustez al sistema, las ontologías, y en general los lenguajes de modelado, permiten organizar los componentes proporcionando robustez al sistema, debido a que, independientemente del tipo de componente que se trate, la estructura es la misma, por lo que la modularidad está implícita y la escalabilidad sólo depende de los aspectos tecnológicos. En la actualidad han aparecido una gran cantidad de sistemas de microcontroladores de bajo coste y cierta potencia, por lo que su empleo en sistemas distribuidos está aumentando considerablemente. En este artículo es expone una ontología como método de organización de los componentes de un sistema distribuido de control inteligente de exteriores. La ontología parte del entorno a controlar en lugar de centrarse en los elementos de control y permite desarrollar una gran variedad de software aislado de los detalles de los elementos de control.

Palabras Clave: Control, Sistema Distribuido, Ontología, Arduino.

1 INTRODUCCIÓN

El control inteligente de sistemas distribuidos es uno de los campos más complejos del control especialmente por las dificultades de la sincronización entre los diferentes nodos [4]. La organización de los nodos permite establecer unas bases de funcionamiento eficiente de las comunicaciones. Con la aparición de los lenguajes de modelado, especialmente UML [3], los sistemas de control adoptan estos lenguajes para definir la arquitectura de los mismos [5].

En este artículo se presenta una ontología del control [10] de las características (iluminación, presencia y similares) para entornos abiertos. En estos entornos, el espacio es acotado al aire libre y todos los elementos del sistema distribuido no se encuentran en un entorno protegido, exceptuando el servidor o el sistema de comunicaciones troncal del sistema. La variedad de estos entornos es muy grande y va desde los jardines y espacios lúdicos (donde hay una alta cohabitación entre personas y naturaleza) hasta las amplias extensiones de naturaleza.

El artículo se organiza de la siguiente forma: el apartado 2 revisa los aspectos más destacables de los entornos abiertos y del sistema Arduentorno, dedicado al control de entornos abiertos con sistemas de bajo coste. El apartado 3 expone la propuesta de componentes del sistema y la ontología que permite organizarlos de forma eficiente. El apartado 4 muestra un ejemplo de uso del sistema en uno de los módulos del proyecto Arduentorno. Finalmente el apartado 5 expone las conclusiones y las líneas de trabajo a desarrollar.

2 ENTORNOS EXTERIORES

2.1 CARACTERÍSTICAS

Cuando un sistema distribuido interactúa con un espacio donde la extensión del terreno es notable, sugiere un número cuantioso de sensores, actuadores y nodos que los controlen. Además de una cierta complejidad en las comunicaciones. El sistema, distribuido al encontrarse en un entorno abierto, implica que el dominio en el que actúa es más agresivo que un entorno cerrado. El entorno tiene una gran cantidad de variables físicas y químicas, que pueden influir de una manera negativa. Además existe la posibilidad de que los elementos del sistema, reciban daño físico de animales e incluso personas.

El conjunto de elementos de control que conforman un nodo, están situados en un área concreta del entorno abierto, dispone de una serie de variables difíciles de controlar. Un elemento de control, por ejemplo un sensor de temperatura, en un entorno cerrado donde la temperatura tiene una deriva temporal de largo plazo, es fácil de controlar, sin embargo, en un entorno abierto ese largo plazo puede convertirse en corto plazo, dando pasó a una variabilidad condicionada por los factores externos de esa área (rachas de viento o zonas que cubre la sombra rápidamente). Esta variabilidad implica ciertas dificultades en el control.

Algunas consideraciones a tener en cuenta, que caracterizan los entornos abiertos, y son relevantes en el diseño de un sistema de control exterior son las siguientes:

- Los elementos de control (sensores y actuadores), así como los nodos que los controlan, deben disponer de una red de comunicaciones inalámbrica para que todos los elementos estén interconectados.
- Cuando un sistema distribuido actúa en un entorno donde los datos que manejan los sensores, o actuadores, puedan estar comprometidos (especialmente por fallos provocados por las condiciones exteriores), es necesario replicar o emplear los diferentes elementos de control para poder validar la información que transmiten o realizar una acción de control de mejor calidad.

La organización de los elementos está implícita en el uso de las comunicaciones ya que debe haber conexiones que ya definen una relación. La necesidad de disponer del mayor número de elementos para determinar la acción de control hace que sea de gran interés disponer de los datos de todos los elementos posibles y, por tanto, conocer cómo obtener esos datos, por ejemplo que un controlador pueda pedir el valor de un sensor que no está conectado directamente. Las dos características (organización y ubicación) de elementos del sistema se pueden obtener por medio de una ontología que permite a todos los elementos un conocimiento del sistema, tanto de las características de los otros elementos que lo forman como de las funcionalidades que estos elementos ofrecen.

El objetivo principal de esta ontología es fomentar la escalabilidad y la mantenibilidad del sistema distribuido. En este sistema distribuido, cada nodo transmite su información a otro nodo, que la puede emplear, o a un servidor que la centraliza. De esta manera se permite desplegar una amplia red de nodos, que cubre grandes extensiones de terreno y cumple los requisitos del sistema.

2.2 PROYECTO ARDUENTORNO

La ontología presentada en este artículo, se aplica dentro de un proyecto denominado ArduEntorno (dado que el control básico se realiza por medio de Arduinos), compuesto por varios módulos, cada uno de ellos dedicado a una funcionalidad específica. Estas funcionalidades se enmarcan dentro de las áreas clásicas de la domótica [9], pero aplicadas a entornos exteriores (figura 1). Actualmente estos módulos se encuentran en fase de desarrollo y experimental.

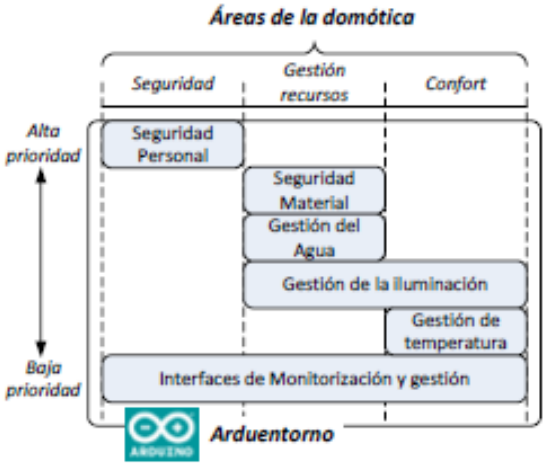

Figura 1: Módulos en desarrollo del sistema Arduentorno y su ubicación en las áreas de la domótica clásica.

Además de los módulos presentados en la figura 1, también se están desarrollando los sistemas de monitorización y gestión remota, tanto vía Web como por medio de dispositivos móviles. Este aspecto implica que la información que el sistema suministre (por lo general la que procede de los sensores) y la que sea necesaria suministrarle al sistema (habitualmente la de los actuadores o configuración) deberá transmitirse a los controladores de cada módulo.

Como todo sistema modular debe estar preparado para crecer por medio de la adición de nuevos módulos, por lo que los aspectos relacionados con la escalabilidad se deberán tener en cuenta. Esto último implica que la organización de los elementos del sistema debe ser común a todos los módulos.

Finalmente, se debe tener en cuenta el aspecto de la optimización de recursos empleados, en cierto modo, relacionado con la gestión energética. Esta optimización se realiza por medio del uso de un mismo sensor en módulos diferentes. Por ejemplo, un sensor de presencia (PIR), que cubre una zona

concreta del entorno, puede usarse por parte del módulo de seguridad para detectar intrusiones, y por parte del módulo de iluminación para encender o apagar las luces de un camino en función de si hay presencia.

Por todo lo anterior, se hace necesaria una organización distribuida y muy bien organizada que permita cubrir el control modular y además sea escalable y homogénea a todos los elementos del sistema. Para ello se ha desarrollado un ontología que organice y describa unívocamente los elementos y las relaciones entre ellos. Las ontologías son una herramienta importante y ampliamente utilizadas en todas las áreas de la tecnología [2] y también son aplicables al control [8]. En el siguiente apartado se presenta la ontología desarrollada que cubre las necesidades organizativas del proyecto Arduentorno.

3 PROPUESTA DE ONTOLOGÍA PARA EL CONTROL DE ENTORNOS ABIERTOS

3.1 ORGANIZACIÓN DE LOS COMPONENTES

Para desarrollar la ontología que describa y organice los elementos e interacciones entre ellos, es necesario reconocer los elementos fundamentales con los que se va a trabajar (figura 2).

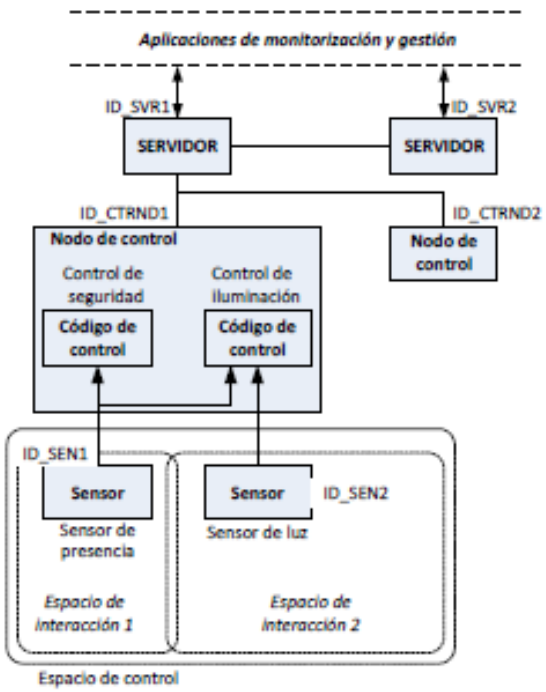

Figura 2: Principales elementos de control del sistema Arduentorno.

En este caso, se partió de los sensores y actuadores, a los que, en el sistema Ardu-Entorno y consiguientemente en la ontología, se les denomina: *elementos de control*. Cada elemento de control tiene un espacio físico donde interviene. A ese espacio se le denomina *espacio de interacción*. Por ejemplo, en el caso de un sensor PIR, el espacio de interacción es la zona, o volumen, donde puede detectar presencia; en el caso de un actuador, por ejemplo una luminaria, el espacio de interacción es el volumen que dicha luminaria es capaz de iluminar.

Para manejar los elementos de control se emplean los llamados *nodos de control*. El objetivo principal de los nodos de control, es controlar la zona de interacción de los elementos que están conectados al nodo. Dependiendo del tipo de sensor que conforme el nodo de control, se obtendrá un tipo de información diferente (iluminación, detección de presencia y similares). De manera análoga en cuanto a la interconexión con el nodo de control, existen una serie de actuadores encargados de actuar en un área concreta. Dependiendo del tipo de actuador interconectado, se pueden controlar una serie de dispositivos como puede ser la iluminaria de una zona o incluso la acústica instalada en una terraza.

Cada elemento de control puede ser visto por nodos de control a los que no esté conectado. En este caso, se trata de un elemento virtual, ya que en algunos casos, un nodo de control puede solicitar, la información de un sensor interconectado a otro nodo de control. La unión de los diferentes espacios de interacción de los diferentes elementos de control, incluidos los virtuales, se conoce como *espacio de control*. Los nodos de control pueden ejercer la labora de comunicación, encargándose de interconectarse con otros nodos y así garantizar la conexión de todos los elementos de control,

Finalmente, los servidores son los componentes del sistema responsables de gestionar la información transmitida por los diferentes nodos de comunicaciones.

3.2 IMPLEMENTACION DE LA ONTOLOGIA

La figura 3 muestra la ontología desarrollada para el control de espacios abiertos descrito en el punto anterior. El espacio físico a controlar se divide en distintas áreas de control. Estas áreas se organizan por medio de la entidad *Area*, que describe un espacio limitado por el usuario, como por ejemplo, la zona de entrada, o la zona de seguridad. El sistema puede tener tantas entidades *Area* como el usuario decida.

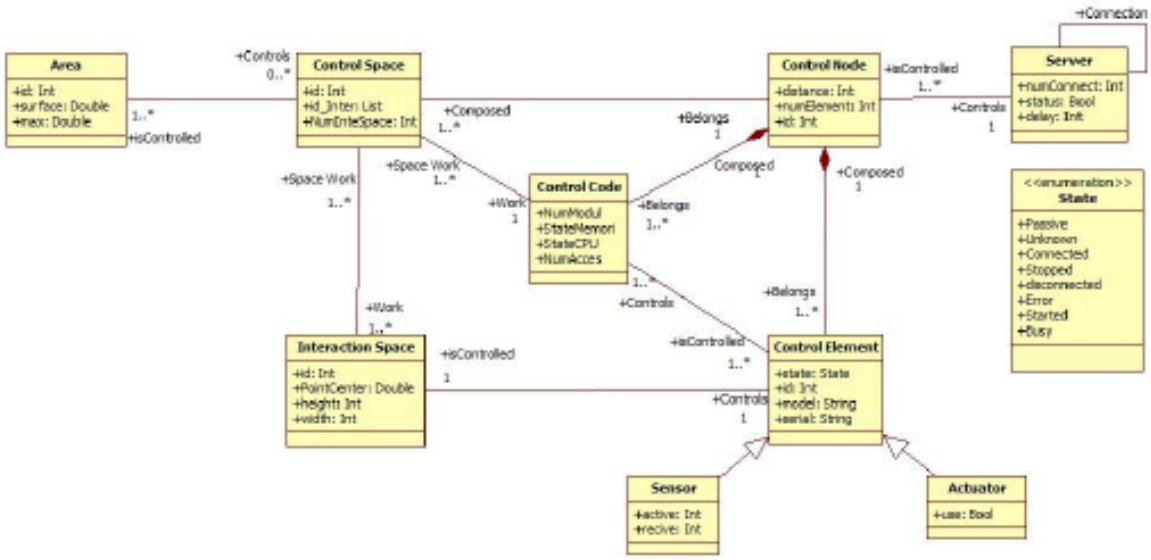

Figura 3: Ontología de control de espacios abiertos.

Un *Area* está puede tener, o no, elementos de control (*ControlElement* en la ontología), por ejemplo, sensores que midan temperatura o humidificadores que varíen la humedad relativa. Sin embargo, y por las necesidades presentadas en los apartados anteriores, estos elementos de control no se asocian directamente al *Area*. Los elementos de control disponen de un *InteractionSpace* que es la zona donde interactúan, por ejemplo en el caso de un PIR el *InteractionSpace* es la zona de detección de presencia y en el caso de una luminaria, el *InteractionSpace* es la zona que ilumina (a partir de un determinado umbral). Un *ControlElement* es empleado por uno o más códigos de control (*ControlCode*) y esto es la clave de la ontología. El *ControlCode* se procesa en un *ControlNode* que, a su vez, dispone de muchos *ControlElement*. Esta estructura no es la clásica estructura jerárquica, sino que forma un grafo, donde el *ControlNode* puede disponer de sensores o actuadores (*ControlElement*) que estén directamente conectados o que estén conectados en otro *ControlNode*. De esta forma, por ejemplo, un sensor PIR puede ser compartido por el módulo de seguridad (por ejemplo para hacer sonar una sirena) y por el módulo de iluminación (para encender o apagar la luz). Cada *ControlElement* tiene un *InteractionSpace*, como un *ControlNode* tiene varios *ControlElement* que intervienen en el control, la unión de los *InteractionSpace* forma un *ControlSpace*, que es la zona que se asocia a las *Area* definidas por el usuario.

En pocas palabras, en el sistema no se controlan áreas definidas por el usuario, aunque éste tenga la impresión de que los sensores y actuadores de un área están empleándose en dicha área. De esta forma, se aísla la estructura que el usuario ha decidido organizar el espacio exterior con los nodos de control.

4 EJEMPLO DE USO

Para la validación del uso de la ontología, se ha desarrollado un prototipo de sistema básico que controla la iluminación de un entorno. Para ello se ha empleado un microcontrolador empotrado Arduino UNO Rev.3 [1] junto a diversos sensores de luz, de presencia y de distancia (por ultrasonidos). El objetivo del control es determinar el nivel de luz que se debe regular en diferentes farolas en un camino (figura 4).

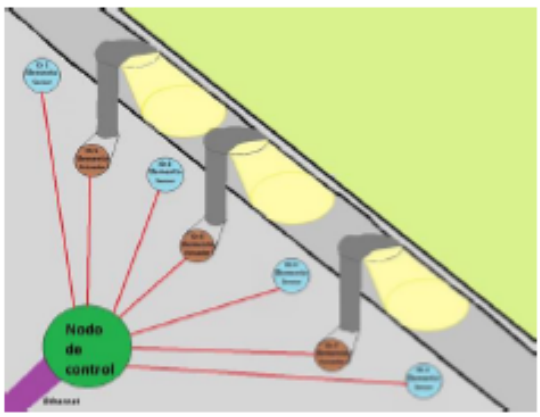

Figura 4: Ejemplo del sistema controlado por el prototipo que emplea la ontología presentada.

Sin embargo, el sistema de la figura 4, se puede enriquecer con la información proporcionada por los sensores del módulo de seguridad

A partir de estos criterios y en función del escenario planteado en la figura 4, se pasa a montar el circuito que implementa las funciones anteriores (figura 5). Los criterios que el código de control va a emplear son diferentes según las necesidades del usuario y se muestran en la tabla 1.

Tabla 1. Acciones de control y prioridades del sistema de iluminación.

Prioridad	Modo y descripción
Alta	Manual. El usuario puede accionar las luces independientemente de los valores de los sensores.
Media	Predictivo. La luz se enciende en el caso de que la luminosidad exterior no supere un umbral definido y se detecte presencia.
Baja	Ambiental. La luz se enciende en el caso de que la luminosidad exterior no supere un umbral definido sin tener en cuenta la presencia.

El código que el Arduino ejecuta es el encargado de recoger el modo de funcionamiento del camino y actuar en consecuencia. Para configurar los modos y los parámetros de cada modo (umbral de luz o umbral de detección) se emplea una aplicación desarrollada en JAVA, instalada en el servidor y se comunica con el nodo de control por medio de una conexión TCP sobre Ethernet (figura 6).

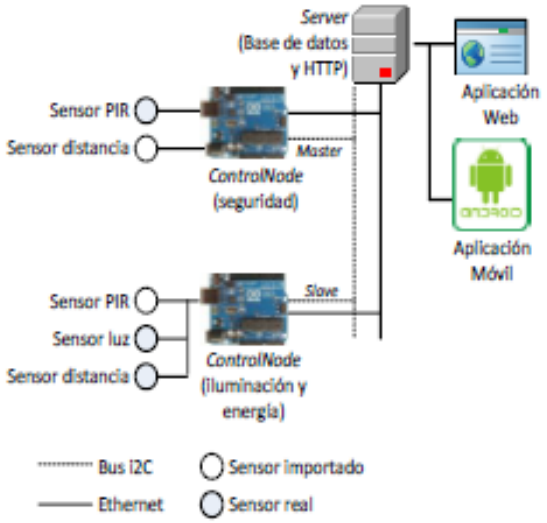

Figura 5: Esquema del sistema implementado. Los actuadores se han omitido para simplificar la visualización.

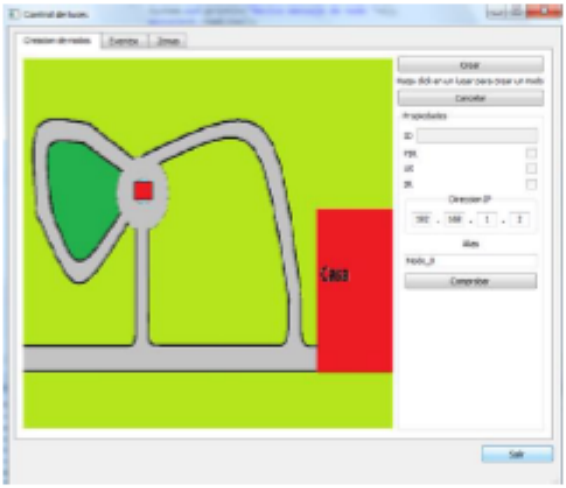

Figura 6: Captura de pantalla de la aplicación JAVA de control del prototipo que emplea la ontología presentada.

Además también se ha desarrollado una aplicación móvil con la misma funcionalidad. No es objetivo de este artículo mostrar los detalles de las comunicaciones ni del sistema gestor de base de datos que se emplea. Sin embargo sí que conviene puntualizar que las comunicaciones entre los nodos de control (Arduinos) se realizan tanto por conexión serie i2C [7] con el nodo de control que está conectado a Ethernet como nodo maestro, mientras que la conexión entre los Arduinos y los servidores se realiza por e como por conexión Ethernet [6]. Sin embargo, cabe destacar que el hecho de emplear la ontología uniformiza tanto las comunicaciones como la gestión de la base de datos. Lo que permite visualizar el sistema sin entrar en los detalles de los sensores reales o importados. Es decir, a efectos del usuario, en el sistema de la figura 6, se visualiza el control de seguridad con sensores presencia y sensores de distancia y el control de la iluminación con sensores de presencia, sensores de distancia y sensores de luz.

5 CONCLUSIONES

En este artículo se ha presentado una ontología empleada para poder controlar espacios abiertos exteriores, aunque no es excluyente a espacios interiores. La ontología permite incrementar los algoritmos de un nodo de control reutilizando los elementos que el sistema dispone. Un servidor es el responsable de guardar las configuraciones del sistema y proporcionar las conexiones entre los elementos de control que permiten reutilizarlos en otros nodos de control.

Actualmente está implementado el sistema de control de iluminación compartiendo sensores con el sistema

de seguridad, de forma que lo sensores necesarios para el sistema de seguridad están directamente conectados con los nodos de control de seguridad y se comparten vía serie con el nodo de control de iluminación, dado que la seguridad es más prioritaria que la iluminación.

Como línea futura se plantea el estudio y la experimentación de diversos escenarios mediante la medición parámetros que permitan demostrar la utilidad del uso de una ontología. En concreto la inclusión del sistema de control energético y la medición de un sistema con duplicidad en sensores y sin sensores duplicados.

Finalmente, es de gran interés para los autores, estudiar y experimentar la programación de patrones de control, de tal forma que sea posible configurar el código del Arduino empleando una aplicación en la que se definan los umbrales de los sensores y las acciones a tomar en cuenta a partir de esos umbrales (a semejanza de las funcionalidades mostradas en la tabla 1).

Agradecimientos

Este trabajo se está realizando bajo las ayudas del proyecto coordinado COBAMI "Control Basado en Misiones" del Ministerio de Educacion y Ciencia del Gobierno Español. CICYT: MICINN: DPI2011-28507-C02-01/02 y el proyecto "Gestión Eficiente De Las Comunicaciones En Sistemas De Control Distribuido En Tiempo Real" financiado por el Vicerrectorado de Investigación de la UPV bajo el programa PAID-06-12 ref. SP20120834.

Referencias

[1] Banzi, Massimo. Getting Started with arduino. " O'Reilly Media, Inc.", 2009.

[2] Bernaras, A; Laresgoiti, I; Bartolome, N.; Corera, J., "An ontology for fault diagnosis in electrical networks," Intelligent Systems Applications to Power Systems, 1996. Proceedings, ISAP '96., International Conference on , vol., no., pp.199,203, 28 Jan-2 Feb 1996. doi: 10.1109/ISAP.1996.501068

[3] Booch, G., Jacobson I., Rumbaugh, J. The UML Specification Document. Rational Software Corp., 1997

[4] Dorf, R.C., Bishop, R.H. Modern Control Systems, 11th Edition, Prentice Hall. 2008

[5] Marcos, M., Estévez, E., Gangoiti, U., Sarachaga, I., & Barandiarán, J. (2004). UML modelling of industrial distributed control systems. In Proceedings Sixth Portuguese Conference on Automatic Control (pp. 7-9).

[6] Metcalfe, R. M., Boggs, D. R. (1976). Ethernet: distributed packet switching for local computer networks. Communications of the ACM, 19(7), 395-404.

[7] Philips. The I2C-bus specification. Philips Semiconductors, 2000, vol. 9397, no 750, p. 00954.

[8] Poza, J. L.; Posadas, J. L. & Simó, J. E. (2009), Adding an Ontology to a Standardized QoS-Based MAS Middleware., in Sigeru Omatu; Miguel Rocha; José Bravo; Florentino Fernández Riverola; Emilio Corchado; Andrés Bustillo & Juan M. Corchado, ed., 'IWANN (2)' , Springer, , pp. 83-90 .

[9] Sánchez P, Sánchez M, Jiménez F, Rosique B, Álvarez A, Iborra. A framework for developing home automation systems: From requirements to code. The Journal of systems and software 2011;84(6):1008-1021.

[10] Sowa, J. F. (1995). "Top-level ontological categories". International Journal of Human-Computer Studies 43 (5-6 (November/December)): 669–85