industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

DISEÑO Y DESARROLLO DE UN SISTEMA INTELIGENTE DE CONTROL DE LA ILUMINACIÓN BASADO EN DISPOSITIVOS MÓVILES INTELIGENTES, BLUETOOTH LOW ENERGY Y FOG COMPUTING.

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

Autor: **ANDRÉS IZQUIERDO MARTÍNEZ**
Director: JUAN ANTONIO LÓPEZ RIQUELME
Codirector: NIEVES PAVÓN PULIDO

Cartagena, 10 de octubre de 2017

Universidad Politécnica de Cartagena

Agradecimientos

QUIERO DAR MI MÁS SINCERO AGRADECIMIENTO A JUAN ANTONIO LÓPEZ RIQUELME Y A NIEVES PAVÓN PULIDO POR HABERME DADO LA OPORTUNIDAD DE HACER ESTE PROYECTO, Y CON CUYA GUÍA HE CONSEGUIDO COMPLETARLO.

TAMBIÉN AGRADECER A TODA LA GENTE QUE HE CONOCIDO EN LA UNIVERSIDAD, TANTO COMPAÑEROS COMO PROFESORES, CON LOS QUE HE COMPARTIDO TANTO Y CON LOS QUE HE APRENDIDO MUCHÍSIMO EN TODO ESTE TIEMPO.

POR ÚLTIMO Y NO MENOS IMPORTANTE, A MI FAMILIA, POR APOYARME Y AYUDARME DURANTE TODO ESTE TIEMPO EN TODO LO QUE HE NECESITADO Y GRACIAS A ELLOS TODO ES SIEMPRE MUCHO MÁS FÁCIL.

GRACIAS A TODOS.

Índice

Capítulo 1

Introducción

1.1 Introducción

Durante la pasada década, la inclusión de distintos avances tecnológicos, tales como los Dispositivos Móviles Inteligentes, ha proporcionado que la comunicación entre seres humanos haya dado un gran paso. Hoy en día, estos avances han ido a más, ya que el usuario de estos dispositivos ya no sólo se comunica con otros usuarios, sino que cada vez es más común que se comunique con el medio que le rodea.

La posibilidad de un entorno inteligente, que proporcione respuesta al usuario mediante la conexión a los distintos dispositivos que le rodean de manera automática es algo que cada vez se hace más necesario. Permitir que la persona sea un elemento más de un sistema donde los elementos se envían información de una manera rápida, segura y eficiente, es quizás el mayor avance que pueda estar teniendo la tecnología en estos momentos.

El gran número de personas que disponen de al menos un dispositivo móvil inteligente, tipo *Smartphone*, abre un campo de posibilidades infinito de como este dispositivo puede recibir información del medio, procesarla, mostrársela al usuario e incluso interactuar de forma automática con elementos del entorno para hacer la vida del usuario, no sólo más cómoda y segura para él, sino incluso pensar en poder cuidar el propio ambiente que le rodea.

Por otro lado, gracias a la tecnología *iBeacon,* es posible ya poder situar a un usuario dentro de recintos más pequeños en interiores y permitir que su *Smartphone,* reaccione según la zona por donde el sujeto se esté moviendo.

Debido a esta creciente tendencia, existen actualmente diversas plataformas que hacen posible el desarrollo de aplicaciones propias para conectar a los usuarios mediante esta tecnología, y dar una nueva visión del marketing.

1.2 Objetivos

El objetivo principal de este proyecto es el diseño de una arquitectura hardware y software para controlar la iluminación de salas de manera automática mediante Dispositivos Móviles Inteligentes, *Bluetooth Low Energy* y *Fog Computing*. Estas tecnologías permitirán localizar al usuario en espacios reducidos. Para ello, se proponen los siguientes sub-objetivos:

- Estudio y diseño de las arquitecturas hardware y software necesarias para resolver el problema planteado.

- Estudio, tanto a nivel hardware como software, de los posibles componentes que formarán parte del sistema, considerando los paradigmas de *Cloud Computing* y *Fog Computing*.

- Diseño de un actuador inalámbrico de bajo coste para controlar la iluminación.

- Diseño de un sistema de localización de usuarios de bajo coste para interiores basado en el uso de dispositivos móviles inteligentes y balizas Bluetooth Low Energy

- Diseño de una aplicación (o conjunto de aplicaciones), para dispositivos móviles inteligentes cuyo propósito será la localización del usuario y la iteración con el sistema de forma transparente.

1.3 Desarrollo de la Memoria

Capítulo 1: Introducción

En el Capítulo 1 se presenta de que va a constar el sistema a desarrollar, los objetivos del mismo y la estructura de la memoria.

Capítulo 2: Estado del Arte

En este capítulo se tratará la tecnología *Beacons* y algunas aplicaciones en la industria sobre dispositivos móviles inteligentes (tanto a nivel Hardware como Software) y cuáles serían las prestaciones mínimas necesarias. También se hablará sobre distintos protocolos de comunicación (HTTP, MQTT, etc..) y se terminará hablando de dispositivos empotrados, es decir, se analizarán alternativas existentes para desarrollar la arquitectura hardware y software necesaria para este caso de estudio.

Capítulo 3: Descripción del Sistema

En este capítulo se seleccionan los dispositivos empotrados, el microordenador y dispositivo móvil elegido. Posteriormente, se describen todos los elementos y protocolos utilizados con un mayor nivel de detalle.

Capítulo 4. Caso de estudio. Descripción del software desarrollado.

Se desarrolla un caso de estudio concreto en el que se emplean los elementos seleccionados en el capítulo anterior. Se detallan las características principales del software desarrollado en este proyecto.

Capítulo 5. Conclusiones y trabajos futuros.

En este capítulo se enumeran las principales conclusiones tras la realización de este trabajo, así como los futuros trabajos a desarrollar.

Capítulo 2

Estado del Arte

2.1 Introducción

El proyecto consiste en diseñar un sistema automático de control de la iluminación de salas, de tal manera que cuando la sala esté vacía, todas las luces estén apagadas. Si la sala está ocupada por uno o varios usuarios, el sistema debe ser capaz de detectarlos, ubicarlos y proceder a encender las luces más apropiadas.

Para conseguir el objetivo será necesario diseñar y desarrollar una arquitectura hardware y software que estará compuesta por diferentes elementos.

En este capítulo se analiza el estado de la técnica con objeto de presentar algunas alternativas que se pueden utilizar para diseñar y desarrollar la arquitectura mencionada. En concreto, se analizarán tecnologías de localización en interiores, dispositivos móviles inteligentes, y, en concreto, los sistemas operativos de los mismos, algunos protocolos de comunicación, así como dispositivos empotrados y de tipo *Single Board Computer* (SBC) para integrar en la arquitectura hardware.

2.2 Beacons

Un *beacon* es un pequeño dispositivo que utiliza la tecnología Bluetooth para transmitir mensajes o avisos directamente a cualquier dispositivo compatible que entre en su radio de acción.

Para hacerse una idea del funcionamiento de los beacons, podemos decir que su comportamiento es parecido al de un faro, ya que transmite repetidamente una señal de radio que se compone de una combinación de letras y números transmitidos en un intervalo regular de aproximadamente de una décima de segundo. Cualquier dispositivo equipado con BLE, como un teléfono inteligente, podría escuchar esta señal una vez que está en rango, y así poder estimar la distancia midiendo la intensidad de la señal recibida (RSSI), ya que cuanto más fuerte sea esta señal, más cerca esta.

Ilustración 2-1: Vista real de un Beacon de Estimote

Como se ha comentado, un *beacon* funciona gracias a la tecnología Bluetooth (concretamente Bluetooth Low Energy o BLE) y la única configuración que requiere es establecer qué tipo de mensaje o aviso dará al usuario u otro dispositivo concreto. En el caso de los usuarios, estos no deben hacer nada. A lo sumo tener instalada en su dispositivo móvil una aplicación que muestre aquellos mensajes que llegarán a través de los beacons.

Una de las grandes ventajas del uso del BLE respecto al Bluetooth tradicional [8], a parte del consumo de batería, es que el BLE no requiere emparejamiento con el dispositivo, lo que supone que un teléfono puede escuchar a varios "Beacons" al mismo tiempo. Esto permite más oportunidades, como por ejemplo la "localización indoor".

2.2.1 Protocolos de comunicación

Como se puede ver en la Ilustración *2-2***¡Error! No se encuentra el origen de la referencia.**, dentro de la tecnología beacon como tal, existen diferentes protocolos encargados de permitir la comunicación entre dispositivos. Actualmente, los más usados son:

- iBeacon: protocolo de comunicación desarrollado por Apple basado en BLE.
- Eddystone: protocolo de código abierto de Google.
- AltBeacon: protocolo creado por Radius Networks.

Ilustración 2-2: Diferentes protocolos de comunicación de los Beacons

2.3 Dispositivos Móviles Inteligentes

Hoy en día es habitual contar con uno o varios dispositivos denominamos inteligentes (Smartphone). Son dispositivos que se conectan a Internet, y que son capaces de ejecutar tareas que hasta no hace mucho estaban reservadas para los ordenadores personales.

Su implantación ha supuesto un salto enorme en la forma de comunicarnos con el mundo, y han abierto un mar de nuevas posibilidades, que han permitido que mientras que, por ejemplo, antes hiciera falta varios aparatos distintos para llevar a cabo cada tarea, ahora sólo es necesario un único dispositivo.

Un sistema operativo móvil o SO móvil es un sistema operativo que controla un dispositivo móvil al igual que los PCs. Los dispositivos móviles tienen sus sistemas operativos tales como Android e iOS, entre otros. Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. Mencionar también que los sistemas operativos utilizados en los dispositivos móviles están basados en el modelo de capas.

2.3.1 Sistemas Operativos para el móvil

En este apartado vamos a hablar de los 3 Sistemas Operativos más importantes que hay actualmente: Android, iOS y Windows Phone.

Ilustración 2-3: Principales Sistemas Operativos para móviles

2.3.1.1 Android

El sistema operativo Android [1] es sin duda el líder del mercado móvil en sistemas operativos en el mercado Europeo. Inicialmente fue diseñado para cámaras fotográficas profesionales, luego fue vendido a Google y modificado para ser utilizado en dispositivos móviles como los teléfonos inteligentes.

Android está basado en Linux, disponiendo de un Kernel en este sistema y utilizando una máquina virtual sobre este Kernel, que es la responsable de convertir el código escrito en Java de las aplicaciones a código capaz de comprender el mismo.

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. También proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Esta sencillez, junto a la existencia de herramientas de programación gratuitas, hacen que una de las cosas más importantes de este sistema operativo sea la cantidad de aplicaciones disponibles, que extienden casi sin límites la experiencia del usuario.

Una de las mejores características de este sistema operativo es que es completamente libre. Es decir, ni para programar en este sistema ni para incluirlo en un teléfono hay que pagar nada. Y esto lo hace muy popular entre fabricantes y desarrolladores, ya que los costes para lanzar un teléfono o una aplicación son muy bajos.

Cualquiera puede bajarse el código fuente, inspeccionarlo, compilarlo e incluso cambiarlo. Esto da una seguridad a los usuarios, ya que algo que es abierto permite detectar

fallos más rápidamente. Y también a los fabricantes, pues pueden adaptar mejor el sistema operativo a los terminales.

Ilustración 2-4: Android Nougat

2.3.1.2 iOS

iOS [16] es un sistema operativo propiedad de Apple orientado a sus dispositivos móviles táctiles como el iPhone, el iPod touch o el iPad. Cuenta con actualizaciones periódicas que están disponibles para su descarga y actualización a través de iTunes, que es el software gratuito e indispensable para manipular y sincronizar toda clase da archivos en estos dispositivos.

Una de las novedades que ha incluido Apple en sus últimos dispositivos, es la actualización del sistema vía OTA (*On The Air*), lo que se hace directamente desde el propio terminal y sin tener que conectarlo a iTunes ni necesidad de poseer un ordenador personal, ya que sólo se requiere una conexión WiFi.

Este sistema operativo está orientado específicamente para su uso mediante dispositivos móviles con pantalla Táctil y es una variante del Mac OS X**,** que es el sistema operativo para computadoras de la marca Apple y, del mismo modo, está basado en Unix.

Una de las peculiaridades más valoradas por los usuarios de este sistema operativo móvil, es su funcionalidad y capacidad para trabajar con múltiples programas a la vez y en segundo plano, lo que es conocido como la multi-tarea (a partir del iOS 4). Además, al ser un sistema operativo orientado exclusivamente para dispositivos móviles con pantalla táctil, incorpora la tecnología *multi-touch*, la cual es capaz de reconocer múltiples gestos y toques en la pantalla, así podremos, por ejemplo, pellizcando en la pantalla ampliar o reducir una imagen.

Las aplicaciones para iOS se escriben y desarrollan en Swift usando el IDE Xcode.

2.3.1.3 Windows Phone

Windows Phone [24]es un sistema operativo móvil desarrollado por la empresa Microsoft para teléfonos inteligentes y otros dispositivos móviles. Fue lanzado al mercado el 21 de octubre de 2010 en Europa y el 8 de noviembre en Estados Unidos, con la finalidad de suplantar el conocido Windows Mobile.

Microsoft decidió realizar un cambio completo en este nuevo sistema operativo con respecto al otro, no sólo se cambió el nombre, sino que se desarrolló desde cero, presentando una interfaz completamente nueva, mejor comportamiento y un mayor control sobre las plataformas de hardware que lo ejecutan, todo con el propósito de volver a ser competitivo en el mundo de los móviles.

La primera generación de Windows Phone es Windows Phone 7 Series conocido también como Windows Phone 7. Dicho número fue tomado debido a que su antecesor en el mercado era Windows Mobile 6.5. Cabe señalar que el Windows Phone presenta incompatibilidad con los Windows Mobile anteriores, los usuarios no serán capaces de actualizar el Windows en su teléfono y por ende deberán comprar uno nuevo con el reciente sistema operativo.

2.3.2 Componentes de un Smartphone

La evolución del hardware de los *Smartphone* viene dada por la miniaturización de los componentes electrónicos que lo forman y una mejora en el proceso de producción/fabricación con menor consumo y mayores velocidades.

Las velocidades de micro-procesamiento guardan una relación directa con el número de transistores incluidos sobre el chip, y cuanto más pequeño sea el transistor, mayor cantidad de ellos podrán ser empleados dentro de un mismo chip.

2.3.2.1 Procesadores ARM

ARM es la responsable de la rápida evolución en los últimos años de los dispositivos móviles en una gran parte de los casos. Los procesadores ARM se basan en el modelo RISC y están licenciados por la compañía británica ARM Holdings, que realizó su introducción en el mercado en su primer modelo en el año 1985. Desde entonces la tecnología ARM se ha actualizado de forma constante hasta alcanzar la madurez suficiente para convertirse en la arquitectura de 32 bits más exitosa del mundo. De hecho, cerca del 75 % de los procesadores de 32 bits montados en cualquier tipo de dispositivo poseen este

chip en su núcleo (*tablets*, videoconsolas, *routers*, etc.) y está presente en más del 95 % de los *Smartphone* actuales.

El diseño de los procesadores de los Smartphone está muy relacionado con el desarrollo del concepto multi-núcleo y disminución del proceso de fabricación en nm. Por ejemplo, 45nm de tamaño en el proceso de fabricación es más pequeño que 65nm, a menor tamaño menos calor y menor consumo eléctrico, al ser más pequeños permiten un mayor número de ellos en el chip y se gana un mejor rendimiento.

El microprocesador es la parte más importante de cualquier equipo electrónico, y desde hace unos años la tendencia es duplicar, triplicar e incluso cuadriplicar el núcleo de dicho microprocesador.

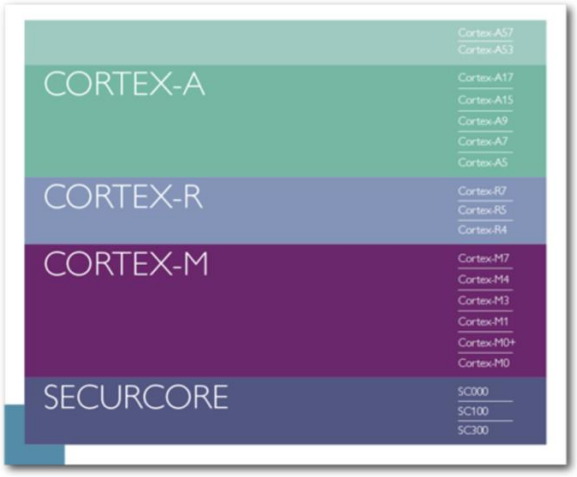

Ilustración 2-5: Familia de Procesadores ARM Cortex

2.3.2.2 GPU

La unidad de procesamiento gráfico o GPU (*Graphics Processing Unit*) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante. Existe básicamente para aligerar la carga de trabajo en videojuegos o en aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la unidad central de procesamiento (CPU) puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos).

La GPU implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el *antialiasing* o suavizado de bordes (evita el *aliasing* que es un efecto visual tipo "sierra" o "escalón"). Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos.

Las GPU modernas son descendientes de los chips gráficos monolíticos (circuitos integrados que están fabricados en un solo monocristal, habitualmente de silicio) de finales de la década de 1970 y 1980.

La GPU en los *Smartphone* está especializada en mostrar los gráficos de la interfaz de usuario, efectos 3D y 2D, reproducción de vídeo en HD Ready (720p) o full HD (1080p), reproducción de gráficos avanzados 3D y 2D en videojuegos.

2.3.2.3 Memoria RAM

La memoria RAM es uno de los componentes críticos del móvil, junto con los núcleos de procesamiento de la CPU y GPU. Sin RAM, cualquier tipo de sistema de computación sería incapaz de realizar tareas básicas y acceder a los archivos de su memoria secundaria sería inaceptablemente lento.

Los archivos críticos que necesita el procesador se almacenan en la memoria RAM, que siempre ha de estar lista en espera de ser leída o escrita. Estos archivos críticos para el dispositivo pueden ser: los componentes del sistema operativo, datos de aplicaciones y gráficos de un juego, o en general cualquier cosa a la que se deba acceder a velocidades mayores que las de acceso a memorias de almacenamiento secundario.

El tipo de memoria RAM que se utiliza en móviles *Smartphone* es, técnicamente, DRAM (RAM Dinámica). La estructura de la DRAM es tal que cada condensador de la placa de RAM almacena un bit, y estos condensadores requieren de un constante "refresco" o actualización de los datos que están almacenados. El contenido del módulo de memoria DRAM se puede cambiar rápida y fácilmente para almacenar diferentes datos.

2.3.2.4 Pantalla Táctil

La tecnología actual para la función táctil de las pantallas se conoce como capacitiva. Se basa en conectar el toque del usuario a través de una lámina protectora que se encuentra al frente de la pantalla, de tal manera que cada unidad pueda instalarse por detrás de materiales de protección o vidrio anti-roturas. El sistema integrado resistente a golpes, rayones y vandalismo, además de no verse afectado por agentes tales como humedad, calor, lluvia, nieve o granizo y líquidos corrosivos. El *touchscreen* -sólido y estable- y su controladora, aumentan el nivel de calidad y vida útil del equipo ofreciendo una respuesta rápida y libre de error, además de requerir bajos niveles de mantenimiento y recalibración.

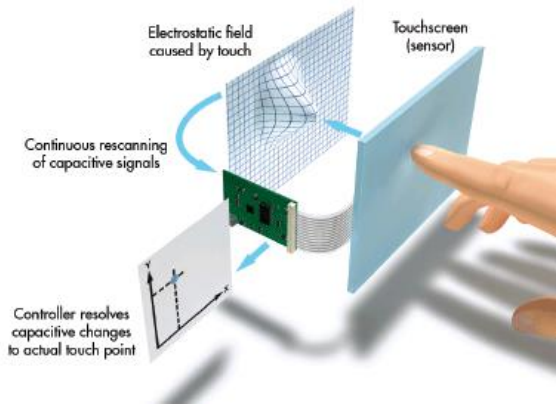

Ilustración 2-6: Funcionamiento de una pantalla táctil capacitiva

El componente principal que proporciona el campo electrostático es transparente, por lo que en la mayoría de las pantallas táctiles no es posible ver la cuadrícula de electrodo capacitivo en la capa de sistema integrado.

Las principales tecnologías de pantallas táctiles son:

- LCD: Una pantalla de cristal líquido o LCD (siglas del inglés *Liquid Crystal Display*) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.

- OLED: Acrónimo de *Organic Light-Emitting Diode*. En este caso lo que se usa son emisores LED, pero en cuyos componentes se incluye el carbono y de ahí la palabra orgánico en su nombre. Estas pantallas prometen menor consumo, mejores tiempos de respuesta y colores más realistas.

2.3.2.5 Cámara

Una de las características de los *Smartphone* más apreciadas por buena parte de los usuarios es su capacidad a la hora de tomar fotografías. De hecho, las principales firmas del sector han realizado un esfuerzo importante para mejorar las prestaciones de sus teléfonos inteligentes en este sentido.

El hardware que hace posible la adquisición con un *Smartphone* está formado por los mismos elementos que podemos encontrar en una cámara fotográfica, es decir, un bloque óptico y un sensor. El bloque óptico se responsabiliza de confinar la luz visible y transportarla sin provocar distorsiones ni aberraciones cromáticas hasta el sensor.

El sensor de imagen es una matriz o cuadrícula de pequeñísimos dispositivos electrónicos sensibles a la luz, conocidos como fotorreceptores, fotosensores, o, sencillamente, celdas. En el ámbito que nos ocupa debemos destacar dos tipos de sensores: los CCD (*Charge-Coupled Device*) y los CMOS (*Complementary Metal Oxide Semiconductor*).

2.3.2.6 Sensores

Los dispositivos móviles inteligentes están equipados con algunos sensores, siendo los más comunes los siguientes:

- Acelerómetro: Se denomina acelerómetro a cualquier dispositivo capaz de detectar las fuerzas de aceleración a las que se ve sometida una masa. Se puede utilizar para detectar la inclinación, la vibración, el movimiento, el giro y el choque. Actualmente es posible construir acelerómetros de tres ejes (X, Y, Z) en un sólo chip de silicio, incluyendo en el mismo la parte electrónica que se encarga de procesar las señales.

- Giroscopio: El giróscopo o giroscopio es un dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría. Detecta la rotación a la que se somete el dispositivo.

- Brújula Digital: La brújula es un instrumento que sirve de orientación y nos permite conocer la dirección del *Smartphone* en relación a los polos magnéticos de la Tierra o al norte geográfico.

- Sensor de Luz: Este pequeño componente se encargará de recoger información sobre la luz ambiental y pasársela a nuestro dispositivo para que éste ajuste el brillo de nuestra pantalla con el fin de hacer que la visibilidad y comodidad de uso sea lo más satisfactoria posible.

- Sensor de Proximidad: El sensor de proximidad es un transductor que detecta objetos o señales que se encuentran cerca del elemento sensor. Estos sensores de proximidad se basan en un LED infrarrojo y también en un receptor IR.

2.4 Protocolos de Comunicación

2.4.1 HTTP

El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP [18]. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

Etapas de una transacción HTTP

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo *Location* del cliente Web.

- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.

- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
  Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, etc…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP

empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor.

- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.

- Se cierra la conexión TCP.

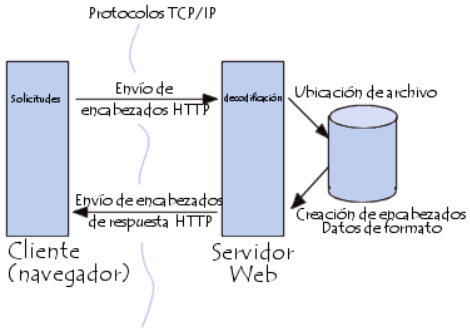

Ilustración 2-7: Etapas de una comunicación HTTP

2.4.2 MQTT

Protocolo usado para la comunicación *Machine-to-Machine* (M2M) en el "*Internet of Things*". Este protocolo está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en la mayoría de los dispositivos empotrados con pocos recursos (CPU, RAM, etc.).

La arquitectura de MQTT [19] sigue una topología en estrella, con un nodo central que hace de servidor o "bróker" con una capacidad de hasta 10000 clientes. El bróker es el encargado de gestionar la red y de transmitir los mensajes para mantener activo el canal. Los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del bróker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones.

La comunicación se basa en unos "*topics*" (temas), que el cliente que publica el mensaje crea y a los que los nodos que deseen recibirlo deben subscribirse. La comunicación puede ser de uno a uno, o de uno a muchos. Un "*topic*" se representa mediante una cadena y tiene una estructura jerárquica. Cada jerarquía se separa con '/'. Por ejemplo, "edificio1 /planta5 /sala1 /rapsberry2 /temperatura" o "edificio3 / planta0 /sala3 /arduino4 /ruido". De esta forma se pueden crear jerarquías de clientes que publican y reciben datos, y un nodo puede subscribirse a un "topic" concreto ("edificio1/planta2/sala0/arduino0/temperatura") o a varios ("edificio1/planta2/#").

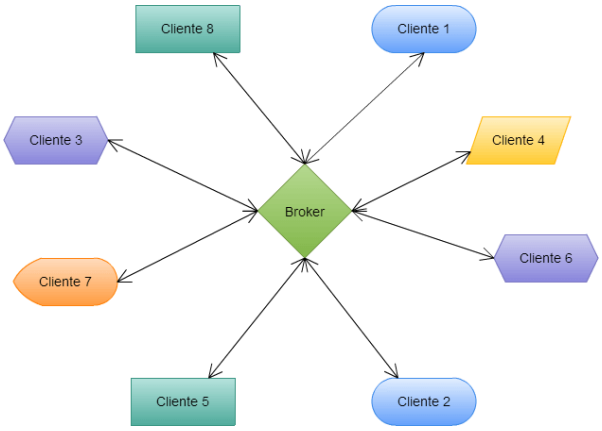

Ilustración 2-8: Esquema Protocolo MQTT

2.4.3 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE), a veces conocido como "Bluetooth Smart", se introdujo como parte de la especificación de Bluetooth 4.0 [8]. Aunque existe cierto solapamiento con el Bluetooth clásico, BLE proviene de un proyecto inicialmente desarrollado por Nokia y conocido como "*Wibree*", antes de que fuera adoptado por Bluetooth SIG (*Special Interest Group*).

Existen muchos protocolos *wireless* para uso en IoT, pero lo que hace que BLE sea tan interesante es que es el más sencillo para implementar la comunicación entre pequeños dispositivos y una aplicación en cualquier plataforma móvil actual (iOS, Android, Windows Phones, etc.), y particularmente en el caso de los dispositivos Apple, es el único método que permite la interacción de periféricos con aplicaciones, sin necesidad de certificaciones MFI y otros requisitos legales que exige iOS.

Bluetooth 4.0 y BLE son soportados en la mayoría de plataformas a partir de las versiones listadas a continuación:

- iOS5+ (iOS7+ preferred)
- Android 4.3+ (numerous bug fixes in 4.4+)
- Apple OS X 10.6+
- Windows 8
- GNU/Linux Vanilla BlueZ 4.93+

2.4.3.1 GAP

Es el acrónimo para el *Generic Access Profile*, y se encarga de controlar las conexiones y los anuncios en BLE. GAP epermite que un dispositivo sea público hacia el exterior y determina como dos dispositivos pueden (o no) interactuar entre ellos.

El GAP define varios roles para los dispositivos, pero lo único que debemos tener claro es que vamos a tener dispositivos centrales y los periféricos.

- Periféricos: Son dispositivos pequeños, de baja potencia, de bajos recursos, que pueden conectarse a dispositivos centrales mucho más potentes. Un ejemplo de periférico puede ser un glucómetro, un medidor de pulsaciones, un beacon, etc.

- Dispositivo Central: Se corresponde normalmente con un teléfono móvil o una Tablet, que tienen una capacidad de proceso mucho mayor.

2.4.3.2 GATT

GATT es el acrónimo de *Generic Attribute Profile*, y define la manera en que dos dispositivos BLE pueden comunicarse usando los Servicios y Características. La comunicación se realiza mediante un protocolo conocido como ATT, que se usa para almacenar los servicios, características y datos relacionados en una tabla usando identificadores de 16-bit para cada entrada en la tabla.

GATT entra en juego una vez se ha establecido una conexión dedicada entre dos dispositivos, lo que significa que ya hemos pasado previamente por el GAP.

Lo más importante a tener en cuenta con el GATT y las conexiones, es que las conexiones son exclusivas. Un periférico BLE sólo puede ser conectado a un dispositivo central (un teléfono móvil, etc.) a la vez. Tan pronto como un periférico se conecta a un dispositivo central, dejará de anunciarse y otros dispositivos ya no podrán verlo o conectarse a él, hasta que se finalice la conexión existente.

La única forma de permitir la comunicación bidireccional es establecer una conexión, en la que el dispositivo central puede enviar datos al periférico y viceversa.

2.5 Dispositivos Empotrados

2.5.1 Arduino

Arduino (ver Ilustración *2-9*) es una plataforma de hardware y software de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación *Processing*. Es decir, una plataforma de código abierto para prototipos electrónicos [3].

Al ser *Open Source*, tanto su diseño como su distribución, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin necesidad de licencia.

El proyecto fue concebido en Italia en el año 2005 por el zaragozano David Cuartielles, ingeniero electrónico y docente de la Universidad de Mälmo (Suecia) y Massimo Banzi, italiano, diseñador y desarrollador Web.

Ilustración 2-9: Vista de la placa Arduino Uno

Arduino está constituido en el hardware por un microcontrolador principal AVR de 8 bits (que es programable con un lenguaje de alto nivel) de la compañía Atmel, presente en la mayoría de los modelos de Arduino, encargado de realizar los procesos lógicos y matemáticos dentro de la placa, además de controlar y gestionar los recursos de cada uno de los componentes externos conectados a la misma.

Consta además de una amplia variedad de sensores eléctricos como cámaras VGA, sensores de sonido, seguidores de línea, botones de control de sensores, e incluso, otras placas de micro controladores más conocidos como *Shields* (ver Ilustración *2-10*), que pueden adaptarse fácilmente gracias a que Arduino cuenta con entradas de pines analógicos y digitales para integrar estos componentes sin necesidad de alterar el diseño original de esta placa.

Ilustración 2-10: Placa Arduino Uno con 2 Shields

Estos a su vez son controlados junto con el procesador primario por otros componentes de menor jerarquía, pero de igual importancia y prioridad, como el Atmega168, Atmega328, Atmega1280 y el Atmega8, que son lo más utilizados debido a sus bajos precios y gran flexibilidad para construir diversidad de diseños.

Además, Arduino cuenta con la ventaja de tener entre sus elementos principales puertos serie de entrada /salida (input/output), lo que le permite conectarse por medio de un cable USB a una computadora para poder trabajar con ella desde nivel software, ya que es dónde se le darán las "ordenes" que ejecutarán cada uno de los componentes conectados a la placa, e incluso, para operar como un dispositivo más.

2.5.1.1 Software

El lenguaje que opera dentro de Arduino se llama *Wiring*, basado en la plataforma Processing y primordialmente en el lenguaje de programación C/C++, que se ha vuelto popular a tal grado de ser el más preferido para enseñar programación a alumnos de nivel superior que estudian computación y robótica, gracias que es muy fácil de apender y brinda soporte para cualquier necesidad de computación.

Para poder trabajar desde el nivel programación del procesador, debe descargarse el software que incluye las librerías necesarias para poder utilizar el lenguaje de manera completa. Otra ventaja es que este software puede descargarse desde el sitio web oficial de Arduino, ya que opera bajo licencia libre y está disponible a todo público. Su versión más reciente para todos los sistemas operativos es la versión Arduino 1.8.5.

Ilustración 2-11: Imagen del IDE de Arduino

2.5.2 Texas Instruments LaunchPad

La tarjeta de evaluación LaunchPad (ver Ilustración *2-12*) es una herramienta de desarrollo y de evaluación para los dispositivos MSP-430 de Texas Instruments [23].

Concretamente, la tarjeta MSP430-G2 dispone de un socket de 20 pines que puede albergar uno de los dos microcontroladores de 16 bits de la familia MSP430 que vienen con el kit, dispone además de una conexión USB que permite descargar y depurar programas directamente en el hardware. Fuera de eso, solamente disponemos de dos botones (uno de ellos es de reset), un par de leds y unos headers (hembra/macho) para poder acceder a los pines del microcontrolador, por lo que el hardware específico para la aplicación habrá que implementarlo externamente.

Ilustración 2-12: Vista de MSP-430 LaunchPad

Esta tarjeta tiene un cierto parecido con la plataforma Arduino basada en un micro AVR. La gran diferencia que existe entre ambos es que el microcontrolador ATMEGA328 dispone de una memoria RAM de 2KB, mientras que los dispositivos MSP430 que vienen incluidos apenas alcanzan los 128 bytes de RAM, lo cual los deja algo limitados para ciertas aplicaciones, aun así, existen microcontroladores MSP430 bastante poderosos y esta tarjeta constituye un punto de entrada excelente para el desarrollo con microcontroladores de TI.

2.5.3 ESP8266

El ESP8266 (ver Ilustración *2-13*)es un chip *Wi-Fi* de bajo coste con pila TCP/IP completa y capacidad de MCU (*Micro Controller Unit*) producida por el fabricante chino Espressif Systems, con sede en Shanghai [11].

El chip primero llegó a la atención de los fabricantes occidentales en agosto de 2014 con el módulo ESP-01. Este pequeño módulo permite a los microcontroladores conectarse a una red Wi-Fi y realizar conexiones TCP/IP sencillas utilizando comandos de tipo Hayes. Sin embargo, en ese momento casi no había documentación en inglés sobre el chip y los comandos que aceptaba. El precio muy bajo y el hecho de que había muy pocos componentes externos en el módulo que sugiere que podría ser muy barato en el volumen, atrajo a muchos hackers para explorar el módulo, el chip y el software en él, así como para traducir La documentación china.

El ESP8285 es un ESP8266 con 1 MB de flash incorporado, lo que permite dispositivos de un solo chip capaces de conectarse a Wi-Fi. Muchos encapsulados del ESP8266 viene con 1 MB de flash.

A finales de octubre de 2014, Espressif lanzó un kit de desarrollo de software (SDK) que permite programar el chip, eliminando la necesidad de un microcontrolador por separado. Desde entonces, ha habido muchos lanzamientos oficiales de SDK; Espressif mantiene dos versiones del SDK – una basada en RTOS y la otra basada en callbacks.

Ilustración 2-13: Vista del módulo ESP8266

2.5.4 ESP32

El módulo ESP32 (ver Ilustración *2-14*)es una evolución del ESP8266 ya que, entre otras ventajas, incorpora Bluetooth.

Las características técnicas principales son realmente impresionantes en cuanto a velocidad de procesamiento con 600 DMIPS, conectividad WiFi (con protocolos de seguridad mejorados) y Bluetooth (clásico y BLE), 32 puertos GPIO PWM (entrada y

salida) con buses 3 x UART, 3 x SPI, 2 x I^2S, 12 x ADC, 22 x DAC, 2 x I^2C e interface para tarjetas SD. También dispone de amplificador de bajo ruido, sensor Hall y 10 x sensores capacitivos.

Algunas marcas ya empiezan a comercializarlo como Bangood en dos variantes, como placa de desarrollo y como módulo de producción. Otros conocidos fabricantes como Adrafruit también lo incorporan a su catálogo de productos.

Ilustración 2-14: Vista del módulo ESP32

2.6 Dispositivos SBC´s

Para ejercer el control en nuestro sistema se requiere de un microcontrolador que actúe como servidor, y también como controlador. Una buena solución sería el uso de un ordenador de placa reducida, o SBC. Una placa computadora u ordenador de placa reducida es una computadora completa en un único circuito. El diseño se centra en un solo microprocesador con la RAM, E/S y todas las demás características de un computador funcional en una sola tarjeta que suele ser de tamaño reducido, y que tiene todo lo que necesita en la placa base. Las más utilizadas son las tarjetas de tipo Raspberry Pi.

2.6.1 Raspberry PI

Raspberry Pi (ver Ilustración *2-15*) es un SBC de bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi. El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (aunque existe la posibilidad de realizar overclock de hasta 1 GHz), un procesador gráfico (GPU) VideoCore IV y 512 MB de memoria RAM. El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente. Tampoco incluye fuente de alimentación ni carcasa. En febrero de 2012 la fundación empezó a aceptar órdenes de compra del modelo A, en febrero de 2013 del modelo B y en julio de 2014 el modelo B+. La evolución de un modelo a otro ha sido la adición de puertos USB en cada versión (1 en el modelo A, 2 en el modelo B y 4 en el modelo B+), el incremento de la RAM (de 256 MB en el modelo A hasta 512 MB en el B) y la aparición del puerto

Ethernet en la versión B. Además, en la última versión se ha reducido el consumo del sistema.

Dispone de conexiones de audio (Jack 3,5 mm), Ethernet, USB, microUSB para alimentación (5 V), ranura para tarjeta micro-SD, salida HDMI y hasta 40 pines GPIO (General Purpose Input/Output) en el último modelo.

En cuanto al software, la fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora), y promueve principalmente el aprendizaje del lenguaje de programación Python y otros lenguajes como Tiny BASIC, C y Perl. Sin embargo, es posible instalar muchos otros sistemas operativos e incluso se ha llegado a implementar Android.

La conexión a Internet se puede realizar mediante una conexión Ethernet, a través del puerto incluido en la placa (a partir del modelo B), o mediante WiFi, conectando un dongle USB WiFi. No obstante, los últimos modelos de Raspberry Pi ya incluyen WiFi.

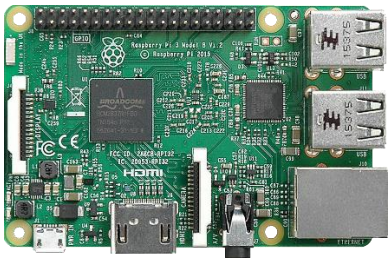

Ilustración 2-15: Vista de la Raspberry Pi

2.6.2 Intel Compute Stick

Intel Compute Stick (ver Ilustración *2-16*¡**Error! No se encuentra el origen de la referencia.**) es un diminuto dispositivo del tamaño de un paquete de chicles que se conecta a tu televisor o monitor por el puerto HDMI. Cuenta con un procesador Intel Atom Z3735F de cuatro núcleos y viene con 2 GB de RAM y 32 GB de espacio de almacenamiento. La ranura para tarjetas microSD en un costado apoya tarjetas de hasta 128 GB, así que tendrás bastante capacidad para maniobrar. También tiene un puerto USB, para que puedas conectar dispositivos externos o una llave USB. Tiene incorporada conectividad Wi-Fi b/g/n, al igual que Bluetooth 4.0, de manera que puedes conectar tus equipos periféricos.

Se trata de un miniPC completo que es compatible con sistemas operativos como Windows 8.1 o Ubuntu.

El Compute Stick no se puede alimentar por el puerto HDMI, lo que significa que necesitas mantenerlo conectado a la electricidad por su puerto Micro-USB.

Ilustración 2-16: Vista del Intel Computer Stick

Capítulo 3

Descripción del Sistema

3.1 Introducción

En el capítulo anterior se han estudiado algunos dispositivos empotrados, microcontroladores y sistemas operativos para dispositivos móviles más importantes.

Dentro de las alternativas propuestas, se ha seleccionado el Intel Computer Stick como SBC, los módulos *ESP32* como hardware de control para la iluminación e *Android* como sistema operativo en el que se desarrollará la aplicación que conecte con los *Beacons*. También entre las alternativas de *Beacons* existentes, se ha elegido el modelo ofrecido por *Estimote*.

Se ha decidido utilizar *Estimote* debido a la existencia de su software libre y por la sencillez de su configuración. En cuanto al sistema operativo para el *Smartphone* se ha elegido Android por su mayor simplicidad a la hora de diseñar y desarrollar la App. Finalmente, la elección del módulo ESP32 se debe a su simplicidad, bajo coste y ser suficiente para controlar una luz debido a las diversas líneas de E/S digitales que incorpora.

En cuanto a los dispositivos empotrados, se ha seleccionado Intel Computer Stick, ya que se trata de una opción de gran interés y es por ello que ha sido elegida para formar parte de este proyecto. A su vez, este dispositivo puede venir con el Sistema Operativo Windows o Ubuntu. Para el proyecto, se ha seleccionado Ubuntu.

3.2 Arquitectura del Sistema

La arquitectura del proyecto va desde el momento en que el Beacon detecta al usuario, hasta el encendido de la luz, pasando por varias fases.

Los puntos que se van a ir desarrollando son los diferentes elementos que se resumen a continuación:

- *Beacon*: Dispositivo que servirá para detectar la posición de las personas que entran en la sala.

- Aplicación Android: Permite la localización de los usuarios y el aviso de la zona en la que se encuentran.

- Servidor o *Bróker*: Dispositivo que sirve para la comunicación entre el móvil y los módulos WiFi. Está implementado en el Intel Computer Stick.

- Los dispositivos hardware para el control de la iluminación a través de los módulos ESP32.

3.3 Beacons

Un *beacon* es un dispositivo de bajo consumo que emite una señal *broadcast*, y son suficientemente pequeños para fijarse en una pared o mostradores. Utiliza conexión *Bluetooth* de bajo consumo (BLE) para transmitir mensajes o avisos directamente a un dispositivo móvil sin necesidad de una sincronización de los aparatos. Concretamente, la señal es captada por estos dispositivos y se transmite a menudo a un servidor en la nube a través de Internet. El servidor de la nube procesa la información y lleva a cabo análisis más detallado para guiar los comportamientos basados en la localización específica del dispositivo móvil.

A diferencia de la tecnología GPS, los *beacons* pueden ser utilizados para la localización exacta dentro de un entorno cerrado. Existen numerosas aplicaciones que han surgido, como el marketing, siendo uno de los usos más mayoritarios. Otras como el servicio a clientes basadas en la localización, asistencia personalizada, etc.

Su implementación está presente en los sistemas operativos móviles más recientes. Android y iOS ya incorporan esta funcionalidad, gracias al soporte de BLE.

Ilustración 3-1: Dispositivo Beacon

3.3.1 Hardware

El *hardware* consiste en un microcontrolador con un chip de radio Bluetooth LE y una batería, generalmente del denominado tipo botón. Los nuevos chips están optimizados para trabajar con BLE.

El chip de radio BLE es generalmente fabricado por dos grandes empresas: Texas Instruments y Nordic Semiconductor.

Generalmente las empresas proveedoras de beacons utilizan el hardware fabricado por los anteriores fabricantes mencionados, pero con sus propios firmwares.

Las baterías de botón son las opciones más populares para la mayoría de estos dispositivos. Concretamente, se suelen emplear las de tecnología de iones de litio y con capacidades comprendidas entre 240 y 1000 mAh. También se pueden encontrar soluciones de beacons que están alimentados mediante baterías alcalinas tipo AA.

También se pueden encontrar soluciones comerciales que se pueden conectar en una toma de corriente o un puerto USB. Estos dispositivos no necesitan un reemplazo de baterías, con el inconveniente de la disponibilidad de una toma de corriente cercana.

3.3.2 Firmware

Cada *beacon* tiene un *firmware* específico, según el proveedor, que permite al *hardware* del beacon funcionar adecuadamente. El *firmware* puede controlar varias características que afectan a la batería.

Potencia de Transmisión (Tx Power)

Los beacons transmiten una señal con una potencia fija, conocida como Tx Power. A medida que la señal viaja en el aire la intensidad de la señal va disminuyendo con la distancia. Con un Tx Power superior se consiguen mayores alcances, lo que significa mayor

consumo. Del mismo modo, si se configura un Tx Power menor se traduce a menor rango de alcance, pero menos consumo de batería.

Advertising Interval

La frecuencia con la que un beacon emite una señal se conoce como *advertising interval.* Un intervalo de 100 ms significa que la señal se emite cada 100 milisegundos, es decir, 10 veces por segundo. Un mayor intervalo, por ejemplo 500 ms, significa que la señal emite sólo dos veces por segundo, lo que significa menos consumo de energía. Cuando se aumenta el *advertising interval,* aumenta la duración de la batería, pero la capacidad de respuesta del dispositivo receptor disminuye. No hay una elección óptima del *advertising interval*, y las aplicaciones que necesiten baja latencia deben elegir intervalos más bajos, en cambio los que necesiten mayor duración de la batería necesitan un intervalo mayor.

En las especificaciones de Apple para iBeacon especifica que el advertising interval recomendado es de 100 ms.

Cada *beacon* ofrece su propia forma de configurar el *hardware* y los parámetros asociados. Algunos proveedores de *beacons* proporcionan su propia aplicación para el *Smartphone* para configurar los *beacons*. Otros *beacons* proporcionan una interfaz abierta a través de cualquier cliente GATT. La principal ventaja de utilizar GATT es que cientos de *beacons* pueden ser configurados a la vez.

3.3.3 Protocolos de beacons

Bluetooth Low Energy (BLE) tiene la capacidad de intercambiar datos en dos estados: modo conectado y modo *advertising*. En el modo Conectado se utiliza el atributo genérico (GATT) para transferir datos en una conexión. En el modo *advertising* se utiliza el perfil de acceso genérico (GAP) para transmitir datos a cualquiera que esté escuchando. Concretamente, el modo *Advertising* es una transferencia de uno a muchos y no tiene garantías sobre la coherencia de los datos. El BLE de los beacons aprovecha el modo *Advertising* GAP para transmitir datos en paquetes *Advertising* periódicos, especialmente formateados. Cada tipo de *beacon* utiliza una especificación personalizada para particionar los datos *Advertising*, dándoles un significado.

3.3.3.1 iBeacon

Es un protocolo creado por Apple, que se introdujo por primera vez en la *Worldwide Developers Conference* 2013, y que opera sobre la tecnología BLE.

iBeacon utiliza BLE para transmitir un identificador único universal (UUID), que es recogido por una aplicación o sistema operativo compatible con el protocolo. El identificador más otros bytes enviados se pueden usar para identificar la posición física del dispositivo, o lanzar acciones basadas en la localización como notificación *push*, etc.

iBeacon ofrece dos métodos de API para detectar dispositivos *ibeacons*:

- ***Ranging***: que sólo funciona cuando la aplicación está activada y proporciona estimaciones de proximidad

- ***Monitoring***: que funciona incluso si la aplicación está en *background*, y proporciona información relativa a entrar y salir de la región definida por un *beacon* determinado.

Las tramas que envían los dispositivos constan de tres campos:

- **UUID** que identifica el beacon.

- **Major** es el número que identifica un subgrupo de beacons dentro de un grupo más grande.

- **Minor** número de identificación a un beacon específico.

La App asociada a los *beacons*, recibe el UUID, Major y Minor. Además, se utiliza la potencia de la señal recibida para estimar la distancia entre el *beacon* y el *Smartphone*. La App utiliza los tres campos para buscar en una base de datos local o remota la información que se debe mostrar al cliente.

3.4 Android

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró.

Android fue presentado en 2007 junto la fundación del *Open Handset Alliance* (un consorcio de compañías de *hardware*, *software* y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el *HTC Dream* y se vendió en octubre de 2008.

El éxito del sistema operativo se ha convertido en objeto de litigios sobre patentes en el marco de las llamadas guerras de patentes entre las empresas de teléfonos inteligentes,

ya que Android es el sistema operativo más usado en España con una cuota de mercado del 92,2% de todos los teléfonos inteligentes.

La versión básica de Android es conocida como *Android Open Source Project (AOSP).*

El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el hardware como el software y ampliar mercados.

El 17 de mayo de 2017, se presentó Android Go. Una versión más ligera del sistema operativo para ayudar a que la mitad del mundo sin smartphone consiga uno en menos de cinco años. Incluye versiones especiales de sus aplicaciones donde el consumo de datos se reduce al máximo.

3.4.1 Características

Algunas de las características y especificaciones actuales del Sistema Operativo Android son las siguientes:

- **Diseño del Dispositivo:** La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.

- **Conectividad:** Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+, NFC y WiMAX, GPRS, UMTS y HSDPA+.

- **Mensajería:** SMS y MMS son formas de mensajería, incluyendo mensajería de texto, además del servicio de *Firebase Cloud Messaging (FCM)* siendo la nueva versión de *Google Cloud Messaging (GCM)* bajo la marca Firebase con los nuevos SDK para realizar el desarrollo de mensajería en la nube mucho más sencillo.

- **Navegador Web:** El navegador web incluido en Android está basado en el motor de renderizado de código abierto *WebKit*, emparejado con el motor JavaScript V8 de *Google Chrome*.

- **Soporte de Java:** Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma, sino que se utiliza una Máquina Virtual Dalvik, que está diseñada específicamente para Android.

- **Soporte Multimedia:** Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, WAV, JPEG, PNG, GIF y BMP.

- **Soporte para Hardware adicional:** Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.

- **Entorno de Desarrollo:** Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android (ADT). Ahora se considera como entorno oficial Android Studio, descargable desde la página oficial de desarrolladores de Android.

- **Multitactil:** Android tiene soporte nativo para pantallas capacitivas con soporte multitáctil que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de *kernel* (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google publicó una actualización para el Nexus One y el Motorola Droid que activa el soporte multitáctil de forma nativa.

- **Multitarea:** Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj.

- **Tethering:** Android soporta tethering, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico (todos los teléfonos desde la versión 2.2. Para permitir a un PC usar la conexión de datos del móvil Android se podría requerir la instalación de software adicional.

3.4.2 Arquitectura

Los componentes principales del Sistema Operativo Android son:

- **Aplicaciones:** Android incluye algunas apps base como un cliente de correo electrónico, un programa de SMS, calendario, mapas, navegador y contactos. Todas las aplicaciones están escritas en lenguaje de programación Java.

- **Marco de trabajo de Aplicaciones *(FRAMEWORKS)*:** Los desarrolladores tienen acceso completo a las mismas *API* del entorno de trabajo usados por las

aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes, ya que cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- **Bibliotecas:** Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Algunas son: *System C library* (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite.

- **Runtime de Android**: incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik, que ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida dx. Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la App.

- **Núcleo Linux**: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

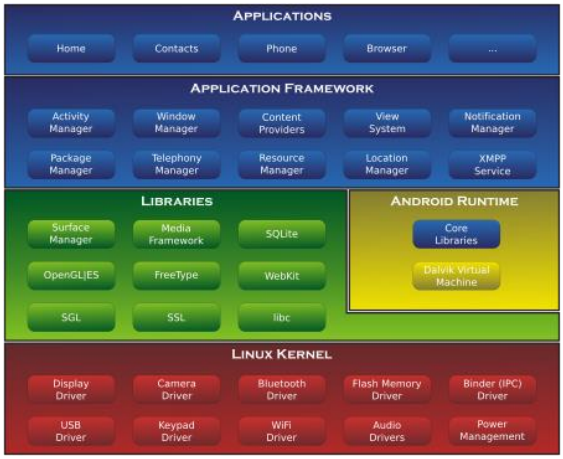

Ilustración 3-2: Arquitectura del Sistema Android

3.4.3 Versiones

Las versiones de Android reciben, en inglés, el nombre de diferentes postres o dulces. En cada versión el postre o dulce elegido empieza por una letra distinta, conforme a un orden alfabético:

Letra	Nombre	Versión	Traducción
A	Apple Pie	1.0	Tarta de manzana
B	Banana Bread	1.1	Pan de plátano
C	Cupcake	1.5	Cupcake
D	Donut	1.6	Rosquilla o dónut
E	Éclair	2.0 / 2.1	Pepito orelámpago
F	Froyo	2.2	Yogur helado
G	Gingerbread	2.3	Pan de jengibre
H	Honeycomb	3.0 / 3.1 / 3.2	Panal
I	Ice Cream Sandwich	4.0	Sándwich de helado
J	Jelly Bean	4.1 / 4.2 / 4.3	Gominolaopastilla de goma
K	KitKat	4.4	Kit Kat
L	Lollipop	5.0 / 5.1	Piruleta[58]
M	Marshmallow	6.0 / 6.0.1	Malvavisco o nube[59]
N	Nougat	7.0 / 7.1 / 7.1.2	Turrón
O	Oreo	8.0	Oreo

Ilustración 3-3: Tabla de las distintas versiones de Android

3.4.4 Aplicaciones

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK), pero están disponibles otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones, Google App Inventor, Android Studio y varios marcos de aplicaciones basadas en la web multiteléfono. También es posible usar las bibliotecas "Qt" gracias al proyecto "Necesitas SDK".

El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de *software* o SDK provisto por Google, el cual se puede descargar gratuitamente.

Todas las aplicaciones están comprimidas en formato APK, que se pueden instalar sin dificultad desde cualquier explorador de archivos en la mayoría de dispositivos.

3.4.5 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run para aplicar cambios mientras tu App se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

3.4.5.1 Estructura del Proyecto

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android.
- Módulos de bibliotecas.

- Módulos de Google App Engine.

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de **Secuencias de comando de *Gradle*** y cada módulo de la aplicación contiene las siguientes carpetas:

- **Manifests**: contiene el archivo `AndroidManifest.xml`.
- **Java**: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- **Res**: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

También se puede personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu App. Por ejemplo, al seleccionar la vista *Problems* de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante para un elemento XML en un archivo de diseño.

3.5 Intel Computer Stick

Intel Compute Stick es un diminuto dispositivo del tamaño de un paquete de chicles que se conecta a tu televisor o monitor por el puerto HDMI. Cuenta con un procesador Intel Atom Z3735F de cuatro núcleos y viene con 2 GB de RAM y 32 GB de espacio de almacenamiento. La ranura para tarjetas microSD en un costado apoya tarjetas de hasta 128 GB. También tiene un puerto USB, para poder conectar dispositivos externos. Tiene incorporada conectividad Wi-Fi b/g/n, al igual que Bluetooth 4.0, de manera que puedes conectar tus equipos periféricos.

Ilustración 3-4: Vista del Intel Computer Stick

3.5.1 Especificaciones

- Procesador Quad-core Intel Atom
- 32 GB de memoria
- 2 GB de RAM
- WiFi 802.11b/g/n
- Bluetooth 4.0
- Gráficos Intel HD
- Audio Intel HD

Además de las especificaciones ya descritas, dispone de los siguientes puertos:

- Puerto USB 3.0
- Entrada de Alimentación
- Puerto HDMI
- Ranura tarjeta MicroSD

3.5.2 Instalación del Sistema Operativo

Pasos recomendados para la instalación de Windows * en una unidad de cómputo Intel®:

1. Revisar la tabla de modelo específico para toda la información especial que se aplica a su modelo Intel Compute Stick (ver Ilustración *3-5*).
2. Se recomienda la actualización del BIOS a la versión más reciente, ya que a menudo se incluyen las correcciones importantes y las actualizaciones de seguridad.
3. Asegurarse de que la unidad de cómputo Intel cuenta con puertos USB disponibles por medio de instalación, el mouse y teclado.
4. Utilizando otro equipo conectado a Internet hay que descargar el controlador de red inalámbrica más reciente para el Intel Compute Stick. Guárdela en un

dispositivo USB portátil. Tendrá que instalar este controlador después de instalar el Sistema Operativo.

5. Prepare su imagen del Sistema Operativo Ubuntu, en un dispositivo portátil USB o en medio de CD/DVD.

6. Conecte el dispositivo que contiene la imagen de instalación (puerto USB).

7. En el símbolo del sistema durante el arranque, pulse la tecla **F10** para abrir el menú Inicio y seleccione la unidad de instalación.

8. Comienza la instalación de Windows. Siga todas las instrucciones de instalación.

9. Tras completar la instalación de Windows, instale al controlador de red inalámbrica que ha descargado en el paso 4.

10. Conectarse a Internet y descargar e instalar el resto de los controladores de Intel Compute Stick desde el centro de descargas.

Modelo de Intel® Compute Stick	Sistema operativo instalado de fábrica	Sistemas operativos compatibles (instalados por el usuario)	Notas
STK2mv64CC STK2m364CC	N/A	Windows® 10 (64 bits) Windows 8.1 * (64 bits) Windows 10 IoT Enterprise Windows Embedded 8.1 industria *	
STK2m3W64CC	Página de inicio de Windows 10 (64 bits) *Puede actualizar a Windows 10 Pro†*	N/A	
STK1A32SC	N/A	Windows 10 (32 bits y 64 bits) Windows 8.1 (32 bits y 64 bits) Windows 10 IoT Enterprise Windows Embedded 8.1 Industry, Pro	Asistencia para Windows 8.1 de 32 bits es limitada: • Gráficos: Controlador de 32 bits de Windows 8.1 no está disponible • Sistema-on-a-Chip (SOC): Paquete de controladores SOC de 32 bits de Windows 8.1 no está disponible • TXE: Controlador de 32 bits de Windows 8.1 no está disponible
STK1AW32SC	Windows 10 Home de 32 bits *Puede actualizar a Windows 10 Pro†*	N/A	
STCK1A32WFCL	Windows 10 Home de 32 bits *Puede actualizar a Windows 10 Pro†*	N/A	
STCK1A32WFCR STCK1A32WFC	Windows 8.1 con Bing * de 32 bits *Puede actualizar a Windows 10 de 32 bits† o* *Windows Pro/Enterprise 32-bit†*	N/A	
STCK1A8LFCR STCK1A8LFC	Ubuntu * 14.04 LTS * de 64 bits	N/A	

Ilustración 3-5: Tabla de Sistema Operativos compatibles con Intel Computer Stick

3.6 ESP32

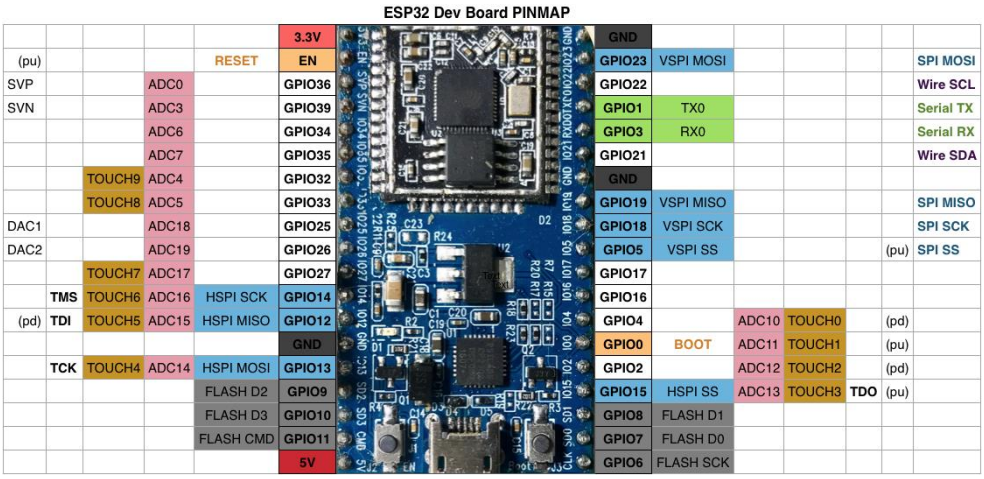

Ilustración 3-6: PinMap del ESP32 Dev Board

El ESP32 es un dispositivo mucho más potente que ESP8266, no sólo porque sea más rápido, sino porque está diseñado para que sea un microcontrolador para IoT.

Utiliza un procesador Xtensa Dual-Core LX6 de 32 bits a 160 ó 240 MHz. El usar dos núcleos permite dedicar uno de ellos a la comunicación IP y WiFi, y el otro al resto de procesos. También dispone de una memoria RAM de 520 kB, accesible por ambos procesadores, y puede utilizar memoria RAM externa adicional de hasta 8 MB.

Al igual que en el ESP8266, la memoria flash ha de ser externa y soporta hasta 16 MB. La diferencia respecto al ESP8266 es que dispone de una memoria ROM interna de 448 kB, utilizada para el bootloader y las funciones internas.

El ESP32 permite la comunicación utilizando los protocolos I^2C, serie, I^2S, SPI e IR. Adicionalmente tiene interfaces *Ethernet*, SDIO y CAN bus 2.0. Integra un sensor de efecto Hall y un termómetro. Además, los conversores ADC pueden activar un preamplificador de bajo ruido para soportar señales de muy bajo nivel.

Una característica muy importante que se ha añadido al ESP32, es una interfaz *Bluetooth* 4.2. Esto amplía el abanico de posibilidades de comunicación.

El ESP32 permite la encriptación del código binario en la memoria flash para que no sea posible leer el código binario desde la memoria.

Por último, además de los dos procesadores principales tiene un coprocesador de muy bajo consumo asociado al RTC. Éste permite realizar ciertas tareas como el control de los pines de entrada salida mientras el chip está en modo de sueño profundo.

3.7 Servicios Web REST

Los servicios web REST definen un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

Estos emergieron en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

Al principio, no tuvieron mucha atención cuando Roy Fielding lo presentó por primera vez en el año 2000 en la Universidad de California, durante la charla académica "Estilos de Arquitectura y el Diseño de Arquitecturas de Software basadas en Redes", la cual analizaba un conjunto de principios arquitectónicos de software para usar a la Web como una plataforma de Procesamiento Distribuido. Ahora, años después de su presentación, comienzan a aparecer varios frameworks REST y se convertirá en una parte integral de Java 6 a través de JSR-311.

3.7.1 Los 4 principios de los Servicios Web REST

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

- Utiliza los métodos HTTP de manera explícita
- No mantiene estado
- Expone URIs con forma de directorios
- Transfiere XML, JavaScript Object Notation (JSON), o ambos

A continuación, vamos a ver en detalle estos cuatro principios, y se explicará la importancia a la hora de diseñar un servicio web REST.

3.7.1.1 Utiliza los métodos de HTTP de manera explícita

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. De acuerdo a esta asociación:

- Se usa POST para crear un recurso en el servidor.
- Se usa GET para obtener un recurso.
- Se usa PUT para cambiar el estado de un recurso o actualizarlo.
- Se usa DELETE para eliminar un recurso.

3.7.1.2 Los Servicios Web REST no mantienen el Estado

Los servicios web REST necesitan escalar para poder satisfacer una demanda en constante crecimiento. Se usan *clusters* de servidores con balanceadores de carga y alta disponibilidad, *proxies*, y *gateways* de manera de conformar una topología serviciable, que permita transferir peticiones de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio web. El uso de servidores intermedios para mejorar la escalabilidad hace necesario que los clientes de servicios web REST envíen peticiones completas e independientes; es decir, se deben enviar peticiones que incluyan todos los datos necesarios para cumplir el pedido, de manera que los componentes en los servidores intermedios puedan redireccionar y gestionar la carga sin mantener el estado localmente entre las peticiones.

Una petición completa e independiente hace que el servidor no tenga que recuperar ninguna información de contexto o estado al procesar la petición. Una aplicación o cliente de servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta. De esta manera, el no mantener estado mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor elimina la necesidad de sincronizar los datos de la sesión con una aplicación externa.

3.7.1.3 Los Servicios Web REST exponen URLs con forma de directorios

Desde el punto de vista del cliente de la aplicación que accede a un recurso, la URL determina qué tan intuitivo va a ser el *web servicie* REST, y si el servicio va a ser utilizado tal como fue pensado al momento de diseñarlo. La tercera característica de los servicios web REST es justamente sobre las URIs.

Las URIs de los servicios web REST deben ser intuitivas, hasta el punto de que sea fácil adivinarlas. Pensemos en las URL como una interfaz auto-documentada que necesita de

muy poca o ninguna explicación o referencia para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.

Una forma de lograr este nivel de usabilidad es definir URLs con una estructura al estilo de los directorios. Este tipo de URLs es jerárquica, con una única ruta raíz, y va abriendo ramas a través de las subrutas para exponer las áreas principales del servicio. De acuerdo a esta definición, una URI no es solamente una cadena de caracteres delimitada por barras, sino más bien un árbol con subordinados y padres organizados como nodos. Por ejemplo, en un servicio de hilos de discusiones que tiene temas varios, se podría definir una estructura de URIs como esta:

```
http://www.miservicio.org/discusion/temas/{tema}
```

La raíz, */discusión*, tiene un nodo */temas* como hijo. Bajo este nodo hay un conjunto de nombres de temas (como pueden ser tecnología, actualidad, y más), cada uno de los cuales apunta a un hilo de discusión.

3.7.1.4 REST transfiere XML, JSON o ambos

La última restricción al momento de diseñar un servicio web REST tiene que ver con el formato de los datos que la aplicación y el servicio intercambian en las peticiones/respuestas.

Los objetos del modelo de datos generalmente se relacionan de alguna manera, y las relaciones entre los objetos del modelo de datos (los recursos) deben reflejarse en la forma en la que se representan al momento de transferir los datos al cliente.

Por último, es bueno construir los servicios de manera que usen el atributo HTTP Accept del encabezado, en donde el valor de este campo es del tipo MIME. De esta manera, los clientes pueden pedir por un contenido en particular que mejor pueden analizar. Algunos de los tipos MIME más usados para los servicios web REST son:

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Esto permite que el servicio sea utilizado por distintos clientes escritos en diferentes lenguajes, corriendo en diversas plataformas y dispositivos. El uso de los tipos MIME y del encabezado HTTP Accept es un mecanismo conocido como "negociación de contenido", el cual les permite a los clientes elegir qué formato de datos puedan leer, y minimiza el acoplamiento de datos entre el servicio y las aplicaciones que lo consumen.

3.8 Bases de Datos

Una base de datos (cuya abreviatura es BD) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y diferentes usuarios deben poder utilizar estos datos. Por lo tanto, el concepto de base de datos generalmente está relacionado con el de red, ya que se debe poder compartir esta información. Generalmente se habla de un "Sistema de información" para designar a la estructura global que incluye todos los mecanismos para compartir datos.

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Se convierte más útil a medida que la cantidad de datos almacenados crece.

Una base de datos puede ser local, es decir que puede utilizarla sólo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red.

3.8.1 Administración de Base de Datos

Rápidamente surgió la necesidad de contar con un sistema de administración para controlar tanto los datos como los usuarios. La administración de bases de datos se realiza con un Sistema de Gestión de Bases de Datos (SGBD) también llamado DBMS (*Database Management System*). El DBMS es un conjunto de servicios (aplicaciones de *software*) que permite a los distintos usuarios un fácil acceso a la información y proporciona las herramientas para la manipulación de los datos encontrados en la base (insertar, eliminar, editar).

El DBMS puede dividirse en tres subsistemas: el sistema de administración de archivos, cuya función es almacenar la información en un medio físico; el DBMS interno, que sirve para ubicar la información en orden; y el DBMS externo, que representa a la interfaz de usuario.

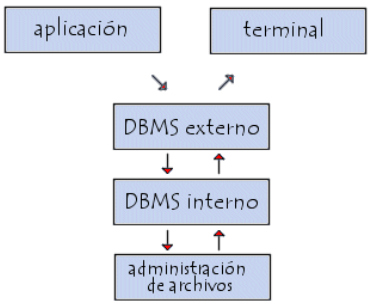

Ilustración 3-7: Sistema de Administración de BD

3.8.1 phpMyAdmin

phpMyAdmin es una herramienta de *software* libre escrita en PHP, destinada a manejar la administración de MySQL a través de la Web. *phpMyAdmin* soporta una amplia gama de operaciones en MySQL y MariaDB. Las operaciones de uso frecuente (gestión de bases de datos, tablas, columnas, relaciones, índices, usuarios, permisos, etc.) se pueden realizar a través de la interfaz de usuario, mientras que todavía tiene la capacidad de ejecutar directamente cualquier sentencia SQL.

Las especificaciones proporcionadas por el programa incluyen:

- Interface Web para la gestión gráfica.
- Manejador de base de datos MySQL, MariaDB y Drizzle.
- Importación de datos desde CSV y SQL.
- Exporta datos a varios formatos: CSV, SQL, XML, PDF (vía la biblioteca TCPDF), ISO/IEC 26300 - OpenDocument Text y Spreadsheet, Word, Excel, LaTeX y otros.
- Administración de múltiples servidores.
- Crea gráficos PDF del diseño de la base de datos.
- Crea consultas complejas usando *Query-by-Example* (QBE).
- Búsqueda global en una base de datos o un subconjunto de la misma.
- Transforma datos almacenados a cualquier formato usando un conjunto de funciones predefinidas, tal como BLOB.
- Live charts para monitorizar las actividades del servidor MySQL, tales como conexiones, procesos, uso de CPU/Memoria, etc.

Capítulo 4

Caso de estudio. Descripción del software desarrollado.

4.1 Introducción

Tras la descripción realizada en los capítulos anteriores, ya es posible llevar a cabo el desarrollo de un caso de estudio concreto que muestre el desempeño de la aplicación y el sistema de iluminación inteligente. En primer lugar, se detallará el caso de estudio de forma general, comentando su funcionamiento y objetivos. Posteriormente, se describirá el hardware y software de forma más específica.

El estudio, en líneas generales, consiste en la implementación de un sistema de iluminación inteligente que haga más eficiente la tarea de uno convencional.

La iluminación inteligente surge como aplicación de las nuevas tecnologías de entornos inteligentes. Con el sistema propuesto será posible que la iluminación de cualquier sala responda automáticamente a la presencia humana, a la vez que conseguir una optimización de la energía eléctrica consumida.

Esta aplicación está pensada tanto para espacios reducidos como amplios, tanto de trabajo como de ocio, habiéndose desarrollado como ejemplo un caso único, pero pudiendo fácilmente escalar el número de dispositivos.

4.2 Descripción de la Arquitectura

El caso de estudio consta de una sale en la que se encuentran 3 luces. Cada luz tiene asociada un *beacon*. Los *beacons*, para identificarlos, tendrían el mismo *UUID*, el mismo *Major* (ya que están en la misma sala), y diferente *Minor* (ya que cada uno está asociado a una luz distinta).

Cuando la App desarrollada detecta el dispositivo beacon principal, creará una lista con los diferentes *beacons* que hay en la zona y se la mandará mediante peticiones HTTP al *bróker* (Intel Computer Stick), que será el encargado de calcular que luz tiene más cerca el usuario, y cuál sería la cantidad de luz necesaria para el usuario, ya que este previamente ha mandado sus preferencias mediante la App.

El *bróker* manda esta información a los ESP32 de la luz, o conjunto de luces mediante peticiones MQTT. Los ESP32 serían los encargados de encender o apagar las luces necesarias.

De esta forma, los dispositivos que componen el sistema son en definitiva los enumerados a continuación:

- Dispositivo 1: El *beacon*, que emite periódicamente mensajes con el *UUID*, el *Major*, y el *Minor*.

- Dispositivo 2: El *Smartphone*, que ejecuta la App desarrollada y recibe los mensajes enviados por el beacon. De esta manera, se puede saber si se ha entrado o salido en la zona que identifica el dispositivo beacon. También es el encargado de crear la lista de *beacons* cercanos, y de mandársela al bróker.

- Dispositivo 3: El bróker, basado en el Intel Computer Stick y sirve de puente entre el controlador de la lámpara (ESP8266) y el dispositivo móvil inteligente. Es el encargado de calcular que luces o conjunto de luces hay que encender o apagar.

- Dispositivos 4: El módulo WiFi ESP32, que habiéndose programado en Arduino, recibirá las órdenes del bróker para encender o apagar las luces.

- Dispositivo 5: La luz, que, en este caso, utilizamos un LED para simular la misma.

A continuación, se describirán en profundidad cada uno de los dispositivos enumerados.

4.3 Dispositivo 1: Beacon

El Dispositivo 1, como se ha comentado anteriormente es el encargado de enviar periódicamente mensajes que incluyen los campos UUID, Major y Minor, tal y como está especificado en la descripción de la tecnología iBeacon.

Ilustración 4-1: Vista de un Beacon

4.3.1 Software

El software principalmente deriva del Smartphone que recibe información de los *beacons*, pero si hay una pequeña configuración que se puede realizar a través de la Aplicación de Estimote.

Al ejecutar la aplicación de la compañía Estimote sale la siguiente pantalla:

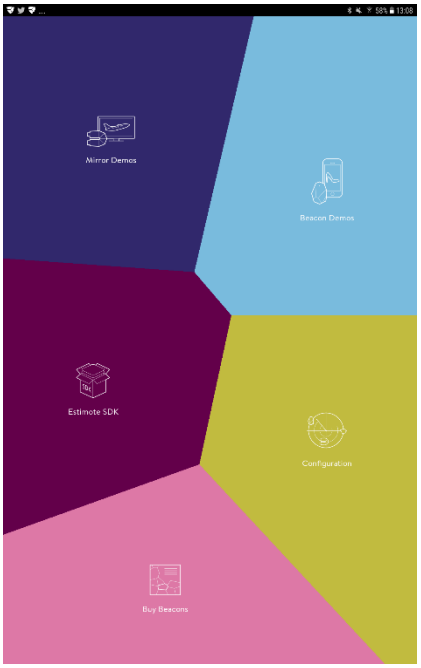

Ilustración 4-2: Pantalla principal Estimote App

Si se pulsa el botón de Configuración, la aplicación se pone a buscar los beacons cercanos:

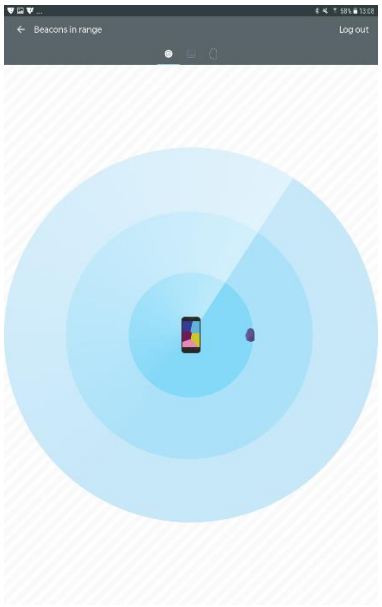

Ilustración 4-3: Buscando los Beacons cercanos

Una vez que aparece el *beacon* que se quiera configurar, se pulsa sobre él, y se abre una pantalla de configuración (ver Ilustración *4-4*), desde la cual se puede configurar el UUID, Major, Minor, el *Advertising Interval* y el *Transmit Power* del *beacon.*

Además, en dicha pantalla se encuentra el número de identificación del beacon, su nombre, el color y la Temperatura del *beacon.*

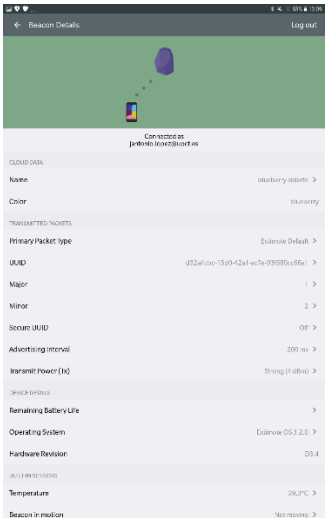

Ilustración 4-4: Pantalla de Configuración

4.4 Dispositivo 2: Smartphone

El *Smartphone* ejecuta la App desarrollada y recibe los mensajes enviados por el beacon. De esta manera, se puede saber si se ha entrado o salido en la zona que identifica el dispositivo beacon. También es el encargado de crear la lista de *beacons* cercanos, y de mandársela al bróker.

4.4.1 Software

La App se ha realizado con el entorno de desarrollo Android Studio, y está escrita en lenguaje Java.

Como cualquier App para Android, el primer paso habitual es definir la interfaz de usuario. En este caso consta de dos *LabelText*, desde los cuales el usuario introduce su DNI, y porcentaje de intensidad lumínica que prefiere. También consta de un botón para que mande dicha información.

Después se declara las librerías de Estimote y de HttpClient. Para ello hay que ir a "build.gradle (Module: app)" y copiar lo siguiente:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.estimote:sdk:1.0.12'
    compile group: 'org.apache.httpcomponents' , name: 'httpclient-
android' , version: '4.3.5.1'
    compile 'org.apache.httpcomponents:httpclient:4.5'
    compile
'org.jbundle.util.osgi.wrapped:org.jbundle.util.osgi.wrapped.org.apach
e.http.client:4.1.2'
}
```

Una vez que tenemos esto, nos vamos a Android.Manifest y se le da permiso a la aplicación para poder conectarse a Internet introduciendo la siguiente línea:

```
<android:name="android.permission.INTERNET" />
```

La aplicación consta de 2 *Activities*: MainActivity y MyApplication. A continuación, se va a explicar lo más importante de cada una.

En la Activity MyApplication es donde se ha programado toda la parte relacionada con los *beacons*. Para empezar, se ha definido un objeto de la clase *BeaconManager* y otro de la clase *BeaconRegion*.

```
public class MyApplication extends
DefaultRequirementsCheckerCallback.Activity {
    private BeaconManager beaconManager;
    private BeaconRegion region;
    boolean apagar=false;
        .
        .
}
```

Lo siguiente que habrá que hacer será configurar la region del beacon con la que va a trabajar en modo Monitoring:

```
beaconManager.connect(new BeaconManager.ServiceReadyCallback() {
      @Override
      public void onServiceReady() {
      beaconManager.startMonitoring(new BeaconRegion("monitored
region",UUID.fromString("D32A4CBC-15D0-42A1-AC7E-93F580CC66A1"),
1,1));
      }
});
```

Cuando el usuario entra en la region definida por el beacon, se inicia el modo Ranging. En cambio, cuando se sale de la región, se llama a la Tarea Asíncrona Encender, a la que le mandamos un JSON con la información del porcentaje de luz preferido por el usuario, su DNI y el valor de la variable apagar.

```
beaconManager = new BeaconManager(getApplicationContext());
beaconManager.setMonitoringListener(new
BeaconManager.BeaconMonitoringListener() {
      @Override
      public void onEnteredRegion(BeaconRegion beaconRegion,
List<Beacon> beacons) {
      showNotification("Entrando en la sala"," ");
      beaconManager.setForegroundScanPeriod(10000, 10000);
      beaconManager.startRanging(region);
      }
      @Override
      public void onExitedRegion(BeaconRegion beaconRegion) {
      showNotification("Saliendo de la sala", "");
      beaconManager.stopRanging(region);
      apagar=true;
      JSONObject json = new JSONObject();
      try {
   json.put("ID", getIntent().getExtras().getString("id"));
   json.put("LUZ", getIntent().getExtras().getString("intensidad"));
   json.put("APAGAR", String.valueOf(apagar));
   TareaWSEncender tarea = new TareaWSEncender ();
   tarea.execute(json);
     } catch (JSONException e) {
       e.printStackTrace();
     }
    }
});
```

Si se ha iniciado el modo Ranging, lo primero que se hace es buscar los beacons cercanos durante 10 s y se va creando una lista con su UUID, Major y Minor.

```
beaconManager.setRangingListener(new
BeaconManager.BeaconRangingListener() {

@Override
            public void onBeaconsDiscovered(BeaconRegion beaconRegion,
final List<Beacon> Beacons) {
                Handler handler = new Handler();
                handler.postDelayed(new Runnable() {
                    public void run() {
                        beaconManager.stopRanging(region);
                        apagar=false;
                    }
                }, 10000);
            }
        });
        region = new BeaconRegion("ranged region",
UUID.fromString("D32A4CBC-15D0-42A1-AC7E-93F580CC66A1"), null, null);
        }
```

Una vez transcurridos los 10 s, se crea un JSON con la lista de los beacons localizados, el porcentaje de luz que prefiere el usuario, su DNI y el valor de la variable apagar. A continuación, se llama a la Tarea Asíncrona Encender.

```
if (Beacons.size() != 0) {
    try {
           showNotification(String.valueOf(Beacons.size()), "");
            JSONArray Beacon = new JSONArray();
            JSONObject Beacon1 = new JSONObject();
            for (int i = 0; i < Beacons.size(); i++) {
               Beacon beacons = Beacons.get(i);
                  JSONObject beacon = new JSONObject();
                  beacon.put("UUID", beacons.getProximityUUID());
                  beacon.put("Major", beacons.getMajor());
                  beacon.put("Minor", beacons.getMinor());
                  beacon.put("RSSI", beacons.getRssi());
                  Beacon.put(beacon);
            }
           Beacon1.put("Beacons", Beacon);
           Beacon1.put("LUZ",
getIntent().getExtras().getString("intensidad"));
           Beacon1.put("ID", getIntent().getExtras().getString("id"));
           Beacon1.put("Tamano", Beacons.size());
           Beacon1.put("APAGAR", String.valueOf(apagar));
           TareaWSEncender tarea = new TareaWSEncender();
           tarea.execute(Beacon1);
   } catch (Exception ex) {
   Log.e("ServicioRest", "Error!", ex);
   }
}
```

La Tarea Asíncrona Encender lo que hace es mandar el JSON que recibe al Servicio Web encender, que será el encargado de decidir si hay que encender o apagar la luz.

```
    private class TareaWSEncender extends AsyncTask<JSONObject,
Integer, Boolean> {
        protected Boolean doInBackground(JSONObject... params) {
            HttpClient httpClient = new DefaultHttpClient();
            HttpPost post = new
HttpPost("http://192.168.1.41/TFG/v1/encender");
            post.setHeader("content-type", "application/json");
            try {
                StringEntity entity = new
StringEntity(params[0].toString());
                post.setEntity(entity);
                HttpResponse resp = httpClient.execute(post);
                String respStr =
EntityUtils.toString(resp.getEntity());
                Log.e("Respuesta: ", respStr);
            } catch (Exception ex) {
                Log.e("ServicioRest", "Error!", ex);
            }
        }
    }
```

Para finalizar la aplicación, hablaremos de la Activity *MainActivity*, que es donde se ha definido todo lo relacionado con la interfaz de usuario. Tenemos dos *LabelText* (TxtId y TxtLuz), desde los cuales el usuario introduce su DNI, y porcentaje de intensidad lumínica que prefiere. Al pulsar el botón, se manda ambos valores a la *Activity MyApplication*.

```
public class MyApplication extends
DefaultRequirementsCheckerCallback.Activity {
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final TextView texto =(TextView)findViewById(R.id.lbl);
    final Button btnLuz =(Button)findViewById(R.id.BtnLuz);
    final TextView txtLuz = (TextView)findViewById(R.id.TxtLuz);
    final TextView txtId = (TextView)findViewById(R.id.TxtId);

        btnLuz.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            Intent intent = new Intent(MainActivity.this,
MyApplication.class);
            intent.putExtra("id", txtId.getText().toString());
            intent.putExtra("intensidad",
txtLuz.getText().toString());
            startActivity(intent);
            texto.setText(txtLuz.getText().toString());
        }
    });
}

}
```

4.5 Dispositivo 3: Bróker

El dispositivo 3 sirve de puente entre el controlador de la lámpara (ESP32) y el dispositivo móvil inteligente. Es el encargado de calcular que luces o conjunto de luces hay que encender o apagar.

4.5.1 Instalación y configuración del servidor MQTT "Mosquitto".

Para instalar Mosquitto, se abre una terminal de Linux y se ejecuta las siguientes sentencias:

"sudo apt-get install php-pear"

"sudo apt-get install php5-dev"

"pecl install Mosquitto-alpha"

Una vez instalado Mosquitto en el bróker, podemos empezar a comunicarnos con los siguientes comandos (abrir 2 consolas, una para suscribirnos al topic y otra para publicar):

"mosquitto_pub -t prueba -m HelloWorld"

"mosquitto_sub -t prueba"

4.5.2 Instalación de XAMPP y configuración en el arranque.

Lo primero es descargar XAMPP para Linux desde:

https://www.apachefriends.org/es/index.html

Esto nos descarga un archivo ".run" que vamos a instalar desde el terminal. Para ello, lo primero que tenemos que hacer es darle permisos al archivo con la siguiente sentencia:

"sudo chmod 755 xampp-linux-x64-7.1.6-0-intaller.run"

Y ejecutamos la instalación:

"sudo . /xampp-linux-x64-7.1.6-0-intaller.run"

Una vez que se haya instalado XAMPP, se abrirá una ventana como la que muestra la siguiente figura:

Ilustración 4-5: Pantalla de inicio de instalación del XAMPP

Se pulsa sobre *Manage Servers,* y se comprueba que esta todo activado, como se muestra en la Ilustración *4-6*. Si no, habría que darle a Start All:

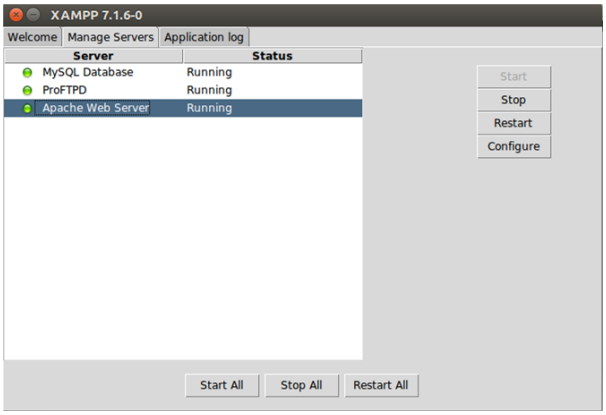

Ilustración 4-6: Activar los programas de XAMPP

Para configurarlo en el arranque, abrimos una terminal de Linux y ejecutamos la siguiente sentencia:

"sudo gedit /etc/rc.local"

Esto lo que hace es crear un ejecutable en el arranque de Linux. Lo siguiente que hay que hacer es buscar el ejecutable, abrirlo y añadimos la siguiente sentencia "/opt/lampp/lampp start" como se muestra en la Ilustración *4-7*.

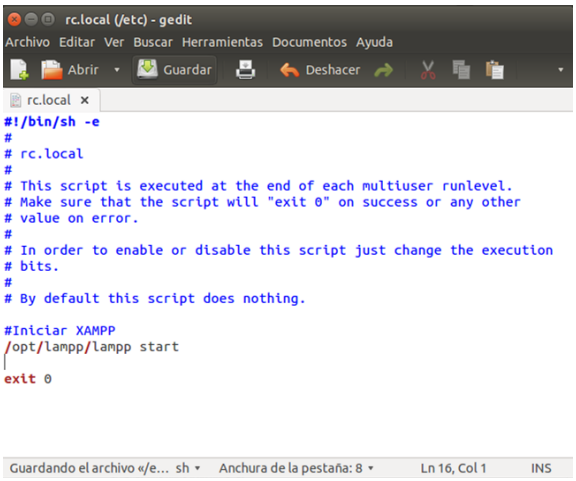

Ilustración 4-7: Inicio de XAMPP en el arranque de Linux

4.5.3 Servicios Web

Se han diseñado 3 Servicios Web REST que estarán en el servidor del bróker, y serán los encargados de mandar la orden de encender o apagar la luz, insertar una nueva luz o beacon en la Base de Datos, y de actualizar los datos del formulario que se va a utilizar para insertar las luces y beacons.

4.5.3.1 Servicio Web Encender/Apagar la Luz

Método Usado: POST

Nombre: encender

Parámetros: JSON con los campos encender, estado y lista de beacons cercanos.

Descripción Código: Lo primero que se hace es declarar el Servicio Web y obtener los parámetros del JSON que nos indica el DNI del usuario, el porcentaje de luz que prefiere y si tenemos que encender o apagar la luz:

```
$app->post('/encender', function(Request $request, Response $response)
{
 $bodyPost = $request->getParsedBody();
 $apagar = $bodyPost['APAGAR'];
 $Estado = $bodyPost['LUZ'];
 $ID = $bodyPost['ID'];
.
.
.
});
```

Se comprueba si se quiere encender o apagar la luz, y se calcula cuál sería el *beacon* más cercano:

```
if ($apagar=="false"){
      $beacon = $bodyPost['Beacons'];
      $tamano = $bodyPost['Tamano'];
      $RSSI = $beacon[0]['RSSI'];
      $i = 0;
      for ($j=1; $j<$tamano; $j++){
            $RSSI1 = $beacon[$j]['RSSI'];
            if($RSSI<$RSSI1){
                  $RSSI=$RSSI1;
                  $i=$j;
            }
      }
      $UUID=$beacon[$i]['UUID'];
      $major=$beacon[$i]['Major'];
      $minor=$beacon[$i]['Minor'];
 }
```

Y se intenta conectarse a la Base de datos. Si hay algún error en la conexión a la Base de Datos, saltará un error que dirá que ha fallado dicha conexión.

```
// Parámetros de la base de datos
 $servername = "localhost";
 $username = "root";
 $password = "";
 try {
      $dbname = "ControlLuces";
      $conn = mysqli_connect($servername, $username, $password,
$dbname);
      // Check connection
      if (!$conn) {     // Se devuelve lo siguiente si falla la
conexión
                  $data["error"] = true;
                  $data["message"] = "Connection failed: " .
mysqli_connect_error();
                  $newResponse = $response->withJson($data);
                  $newResponse = $newResponse->withHeader ('Content-
type',    'application/json');
                  return $newResponse;
      .
      .
      .
      } catch(PDOException $e) {
      $data["error"] = true;
        $data["message"] = "Connection failed: " . $e->getMessage();
        $newResponse = $response->withJson($data);
        $newResponse = $newResponse->withHeader('Content-type',
'application/json');
        return $newResponse;
    }
    mysqli_close($conn);
```

Una vez que nos hemos conectado, lo primero que hay que ver es si se quiere encender una luz o apagarla. Si lo que se tiene que hacer es apagar la luz, primero se busca la luz asociada al estudiante en la tabla "Estudiantes", se borra al estudiante de dicha tabla, y se comprueba si hay más estudiantes asociados a dicha tabla. Si no hay más gente en esa luz, la

luz habría que apagarla. En cambio, si hay alguien más, habría que calcular el nuevo estado de la luz.

```
if ($apagar=="true") {
      //Vemos cual es la luz asociada al estudiante
      $idluz=LuzAsocEst($ID,$conn);

      //Quitamos al estudiante de la DB y vemos si hay alguien más en
esa luz
      $result=ComprobarLuz($idluz, $ID, $conn);

      if($result) {
            //Calculamos el nuevo valor que tiene que tener la luz
            $Estado=Calcular($idluz,$conn);
      } else {
            $Estado=0;
      }

      }
```

Y si en cambio, lo que hay que hacer es encender la luz, se busca la luz asociada al *beacon* más cercano al estudiante, se registra al estudiante en la tabla "Estudiantes", y se obtiene el estado de la luz asociada.

```
} else {
      //Buscamos la Luz asociada al Beacon
      $idluz = BeaconAsociado($UUID, $major, $minor, $conn);

      //Registramos el estudiante en la Base de Datos
      RegistroEstudiante ($idluz, $Estado, $ID, $conn);

      //Obtenemos el valor de la luz
      $Estado = EstadoLuz ($idluz, $conn, $Estado);
}
```

Por último, se busca el tópic de la luz que hay que encender o apagar, se guarda el nuevo estado de la luz, se registra que ha habido un nuevo cambio en el estado de la luz, y se manda los datos al ESP32 por MQTT.

```
//Buscamos el topic asociado a la Luz
$topic = LuzAsociada($idluz, $conn);

//Actualizamos el valor de la Luz
NuevoEstadoLuz($Estado, $idluz, $conn);

//Registramos el nuevo valor de la luz en la Base de Datos
RegistroAcceso ($idluz, $Estado, $conn);

//Mandamos el valor de la luz
MandarDato($topic, $Estado);
```

4.5.3.2 Servicio Web Insertar

Método Usado: POST

Nombre: insertar

Parámetros: JSON con los parámetros de la luz o del *beacon* que quieras insertar.

Descripción del Código: Como se ha visto en el apartado anterior (ver 4.5.3.1), lo primero que hacemos es declarar el Servicio Web y obtener el parámetro del JSON. En este caso, los parámetros del JSON indican la tabla en la que hay que insertar los datos.

```
$app->post('/insertar', function(Request $request, Response $response)
{
 // Obtenemos los parámetros del body
 // Que vienen codificados en JSON
 $bodyPost = $request->getParsedBody();
 $tabla = $bodyPost['Tabla'];
 if($tabla=="Luz") {
      $topic = $bodyPost['Topic'];
      $potencia = $bodyPost['Potencia'];
      $tipo = $bodyPost['Tipo'];
 } else if ($tabla=="Beacon") {
      $idbeacon = $bodyPost['idbeacon'];
      $UUID = $bodyPost['UUID'];
      $major = $bodyPost['major'];
      $minor = $bodyPost['minor'];
 } else if ($tabla=="Vecinos") {
      $luz1 = $bodyPost['Luz1'];
      $luz2 = $bodyPost['Luz2'];
 }
.
.
.
});
```

Lo siguiente que hay que hacer es conectarse con la Base de Datos. Para ello, utilizaremos el mismo código que se ha explicado en el Servicio Web anterior. Una vez que estamos conectados a la Base de Datos, se comprueba si ya existe dicha luz en la Base de Datos. Si ya existe, nos salta un error (ver Ilustración 4-8). En cambio, si no existe, procedemos a insertar los datos.

```
if ($tabla=="Luz") {
      //Comprobar si existe la luz ya en la tabla
      $rsp=Comprobar($tabla, $topic, $conn); //Función para comprobar
si existe la luz en la tabla
      if ($rsp){
            $msg["error"] = true;
            $msg["message"] = "La luz ya existe";
            $newResponse = $response->withJson($msg);
```

```
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
       } else {
      $result=InsertarLuz($tipo, $topic, $potencia, $conn);
      if($result == TRUE){
            $msg["error"] = false;
            $msg["message"] = "Luz insertada correctamente";
            $newResponse = $response->withJson($msg);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
      } else {
            $msg["error"] = true;
            $msg["message"] = "Error insertando la luz";
            $newResponse = $response->withJson($msg);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
      }
}
}
```

Para el resto de tablas (tabla beacon y tabla vecinos), se seguiría la misma secuencia.

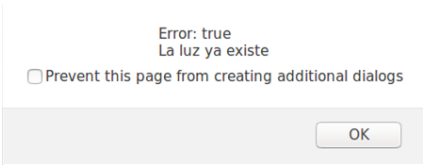

Ilustración 4-8: Imagen de la notificación de error

4.5.3.3 Servicio Web Consultar

Método Usado: GET

Nombre: consultar

Parámetros: Nombre de la tabla que se quiere consultar.

Descripción: Lo primero que se hace es declarar el Servicio Web, obtener la tabla en la que hay que buscar de la URl e intentar conectarse a la Base de Datos.

```
$app->get('/consultar/{tabla}', function(Request $request, Response
$response) {
//Obtenemos la categoría
 $tabla = $request->getAttribute('tabla');
 // Nos conectamos a la base de datos
 $servername = "localhost";
 $username = "root";
 $password = "";
 try {
      $dbname = "ControlLuces";
```

```
      $conn = mysqli_connect($servername, $username, $password,
$dbname);

      // Check connection
      if (!$conn) {
                  $data["error"] = true;
                  $data["message"] = "Connection failed: " .
mysqli_connect_error();
                  $newResponse = $response->withJson($data);
                  $newResponse = $newResponse->withHeader('Content-
type', 'application/json');              return $newResponse;
}
.
.
.
    catch(PDOException $e)
    {
      //echo "Connection failed: " . $e->getMessage();
            $data["error"] = true;
        $data["message"] = "Connection failed: " . $e->getMessage();
        $newResponse = $response->withJson($data);
        $newResponse = $newResponse->withHeader('Content-type',
'application/json');
        return $newResponse;
    }
    mysqli_close($conn);
});
```

Una vez que se ha conseguido conectarse a la Base de Datos, se busca el parámetro que nos interesa en la tabla. Para ello, lo primero que se hace es seleccionar todas las luces que haya en la tabla de la Base de Datos. Si no hay ninguna, se devuelve un error. En cambio, si hay alguna luz, se crea un array en el que vamos guardando el parámetro "idluz" de cada luz, y se devuelve en formato JSON.

```
if( $tabla=="Tipo") {
      $sql = "SELECT idtipoluz, descripcion FROM Tipo_Luz";
}
$result = mysqli_query($conn, $sql);
$data = array();

if (mysqli_num_rows($result) > 0) {
      // output data of each row
      if($tabla=="Tipo"){
            while($row = mysqli_fetch_assoc($result)) {
            $objeto = array('id' => $row["idtipoluz"], 'descripcion' =>
$row["descripcion"]);
            array_push($data, $objeto);
      }
}
$msg["error"] = false;
$msg["message"] = $data;
$newResponse = $response->withJson($msg);
$newResponse = $newResponse->withHeader('Content-type',
'application/json');
```

```
return $newResponse;
} else {
 //echo "0 results";
$msg["error"] = true;
$msg["message"] = "0 results" . $categoria;
$newResponse = $response->withJson($msg);
 $newResponse = $newResponse->withHeader('Content-type',
'application/json');
 return $newResponse;
}
}
```

4.5.4 Formulario

Para la creación del formulario se ha usado *Bootstrap* y *CCS*. Lo primero que se hace es diseñar los *headers*, y después el *body*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>LUCES</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.mi
n.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"
></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.
js"></script>
</head>
<body>
.
.
.
</body>
</html>
```

Dentro del *body*, lo primero que se hace es definir el tipo de formulario y el encabezado del mismo.

```
<div class="container-fluid">
<!-- Encabezado -->
<div class="jumbotron">
<h1>Interfaz LUCES</h1>
</div>
```

El formulario se va a dividir en 3 paneles para insertar en la Base de Datos: uno para insertar luces, uno para insertar *beacons* y otro para insertar que luces son contiguas.

Como ejemplo, se va a explicar el panel de insertar luces, ya que los otros dos se diseñarían de forma parecida.

Dentro del panel de las luces, se tiene un "*labeltext*", donde se introduciría el topic asociado a la luz.

```
<!-- Panel para insertar luces -->
<div class="panel panel-default">
<div class="panel-heading">Inserta una luz</div>
<div class="panel-body">

<!-- Formulario Luz-->
<form>
<!-- TOPIC -->
<div class="form-group">
    <label for="topic">TOPIC:</label>
    <input type="text" class="form-control" id="topic" maxlength="50"
required></input>
 </div>
.
.
.
</div>
</form>
</div>
</div>
```

Después se va a dividir la pantalla en dos columnas de igual tamaño: una para la potencia y otra para el tipo de luz. Dentro de la parte de la Potencia, se configura un *labeltext* para introducir el valor de la potencia de la luz. Y dentro de la parte de Tipo, se va a configurar una lista con los distintos tipos de luces guardados en la Base de Datos.

```
<!-- POTENCIA -->
<div class="row">
      <div class="col-sm-6">
      <div class="form-group">
      <label for="potencia">Potencia (W):</label>
      <input type="text" class="form-control" id="potencia"
maxlength="50" required></input>
      </div>
</div>

<!-- TIPO -->
      <div class="col-sm-6">
      <div class="form-group">
      <label for="tipo">Selecciona el tipo de Luz:</label>
      <select class="form-control" id="tipo" onclick="tipoluz()"
required>
                <option> Tipo</option>
      </select>
      </div>
</div>
```

Por último, tenemos el botón que está configurado para que cuando se pulse, se llama a la función insertarluz(), que es la encargada de consumir el Servicio Web de Insertar explicado en el apartado 4.5.3.2.

```
<!-- Botón para almacenar -->
<button type="button" class="btn btn-default" id ="botonalmacenarluz"
onclick = "insertarluz ()" > Almacenar Luz</button>
```

Para poder meter funciones en el formulario, habría que hacerlo entre los *headers* y el *body*. Se ha usado el lenguaje Javascript para ello, y como ejemplo tenemos la función insertarluz ():

```
<script type="application/javascript">

function insertarluz() {
      var Potencia = document.getElementById("potencia");
      var Topic = document.getElementById("topic");
      var Tipo = document.getElementById("tipo");
      var data = JSON.stringify({
            Potencia: Potencia.value,
            Topic: Topic.value,
            Tipo: Tipo.value,
            Tabla: "Luz"
            });
      insertar(data);
}
</script>
```

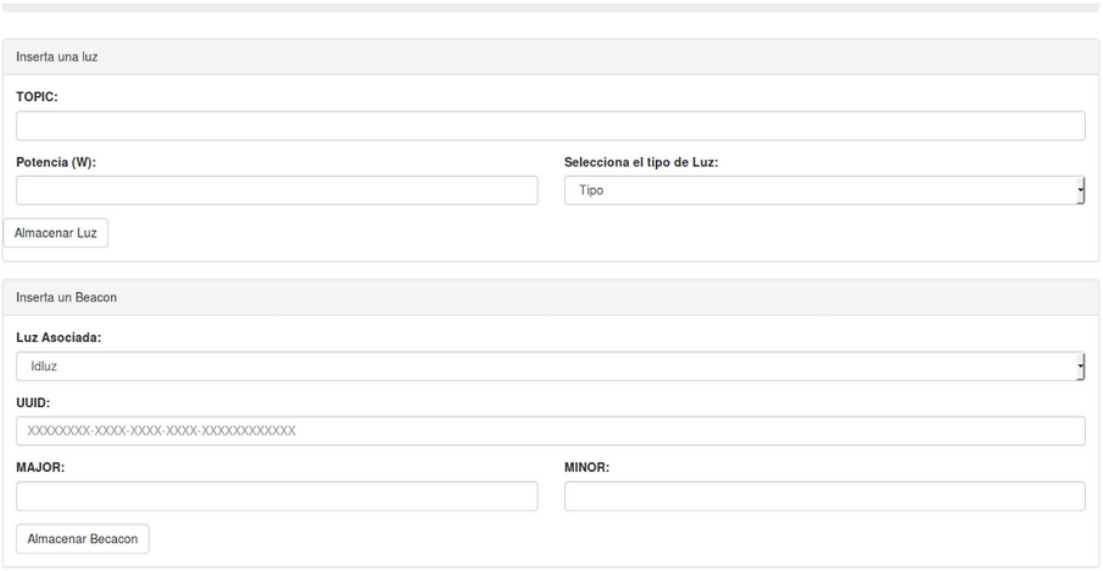

Ilustración 4-9: Imagen del formulario para insertar datos

4.5.5 Base de Datos

El almacén de datos local diseñado se muestra a través del modelo Entidad—Relación, o diagrama E-R (DER). Se ha usado este modelo, ya que se trata de una herramienta

muy conocida y de amplia implantación en el ámbito del modelado de datos que permite representar las entidades relevantes de un sistema de información, junto con sus propiedades o atributos y sus interrelaciones.

Las entidades relevantes se describen a continuación:

- Luz: Representa a cada una de las luces de las diferentes salas. En este caso, la información que es necesaria almacenar para cada luz del sistema está formada por el conjunto de atributos definidos como: idluz, tipo, estado, topic y potencia. El atributo idluz actúa como clave, ya que permite identificar, de un modo unívoco, a cada luz.

- Beacon: Representa a cada uno de los beacons asociados a cada luz. Además de idbeacon, se almacenan otros atributos tales como UUID, Major y Minor.

- Estudiantes: Representa a cada uno de los estudiantes que entran en la sala. Además de la clave primaria idluz, se almacenan otros atributos tales como idluz, DNI y estado.

- Tipo Luz: Representa a cada uno de los posibles tipos de luces. En este caso, la información viene dada por los atributos idluz y tipo.

- Tiene_Vecinos: Representa que luces están contiguas. En este caso tenemos dos claves primarias: idluz e idluz_vecino.

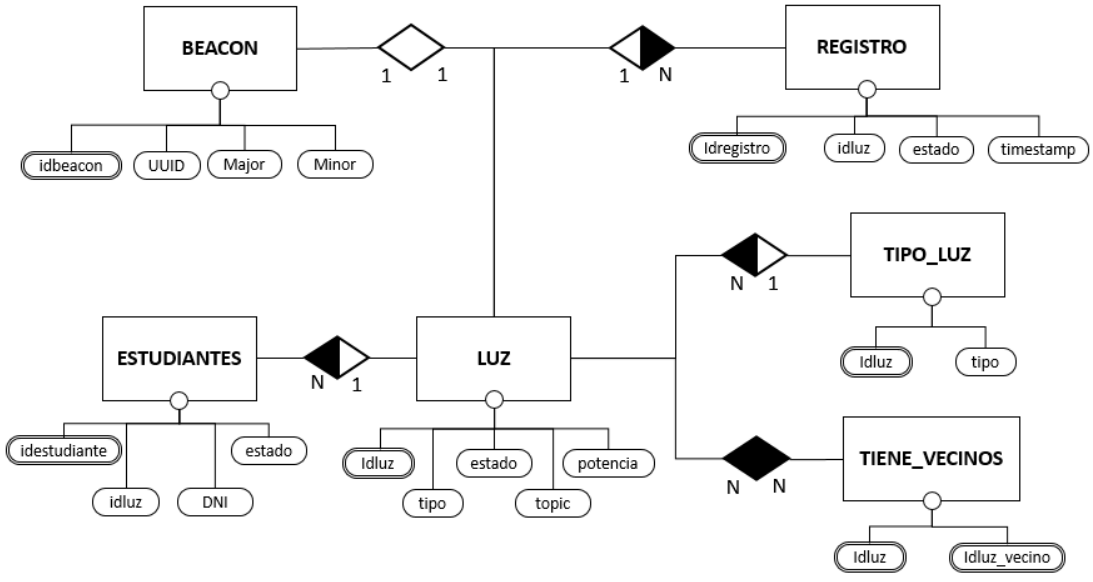

Ilustración 4-10: Modelo Entidad Relación

4.5.6 Fog Computing: Implementación con la nube

En este proyecto, los datos y la computación están distribuidos tanto en recursos locales, como en la nube. Para la parte de la nube, se ha desarrollado un Backend con la aplicación Google App Engine (GAE). De todos los lenguajes disponibles para ello, se ha optado por JAVA.

4.5.6.1 Modelo de Datos

Como se puede ver en la Ilustración 4-11, el modelo de Datos está definido por 2 entidades: Entidad Sala y Entidad Beacon.

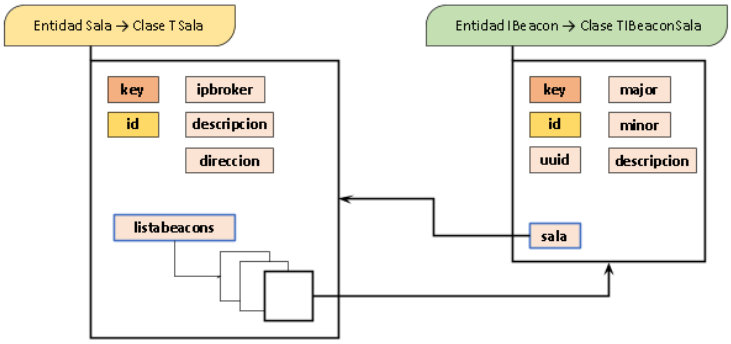

Ilustración 4-11: Entidades del Backend

En la entidad sala se almacena una "key" y un "id", que diferencian a las distintas salas. También se almacena la ip del bróker, una descripción de la sala y la lista de beacon asociada a la sala.

Para la obtención de la ip del bróker, se ha diseñado un programa en C gracias al cual cada vez que se reinicia el bróker, se obtiene la ip, se crea un JSON y se ejecuta una petición HTTP a la nube para actualizarla.

En el caso de la entidad IBeacon se almacena la "key" y un "id" asociado a cada beacon. También hay otros parámetros como el UUID, el major y el minor del beacon, y a que sala está asociado dicho beacon.

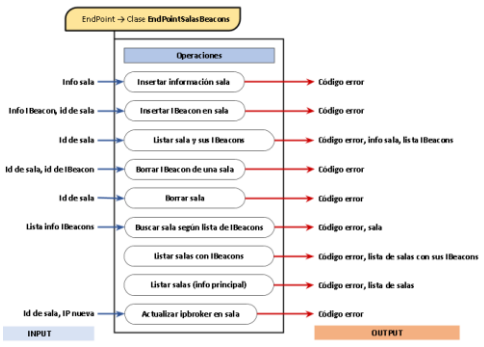

Ilustración 4-12: Ilustración del Endpoint asociado al Backend

La Ilustración 4-12 muestra las diferentes operaciones que se han implementado en el backend. A modo de ejemplo, se va poner el código asociado a la operación insertar información principal de la sala:

```
@ApiMethod(name = "insertarInfoPrincipalSala", httpMethod = "POST",
path = "insertar_info_principal_sala")
public ResultadoOperacion insertarInfoPrincipalSala(TSala sala){
PersistenceManager mgr_t;
ResultadoOperacion res = new ResultadoOperacion();
res.setCodigo(Integer.valueOf(0));
res.setDescripcion("OK");
mgr_t = getPersistenceManager();
Transaction transaccion = mgr_t.currentTransaction();
try{
transaccion.begin();
sala.createKey();
sala.initListabeacons();
if (!existesala(sala.getKey(), mgr_t)){
mgr_t.makePersistent(sala);
}
else{
res.setCodigo(Integer.valueOf(-1));
res.setDescripcion("La entidad ya existe en el Datastore");
}
}catch(Exception e){
res.setCodigo(Integer.valueOf(-2));
res.setDescripcion("Excepcion: " + e.getMessage());
}finally{
transaccion.commit();
if (transaccion.isActive()){
transaccion.rollback();
res.setCodigo(Integer.valueOf(-3));
res.setDescripcion("Transaccion en rollback");
}
mgr_t.close();
}
return res;
}
```

4.6 Dispositivo 4: ESP32

Como método de comunicación entre la luminaria y el bróker se ha usado un módulo WiFi embebido, concretamente el módulo ESP32, del fabricante ESPRESSIF.

4.6.1 Hardware

Como se puede ver en la Ilustración *4-13*, al módulo wifi ESP32 se le ha conectado al pin GPIO05 un LED (para simular una luz de la sala), y una resistencia de 330 Ω. En cuanto a la alimentación, usamos 4 pilas de 1,5V.

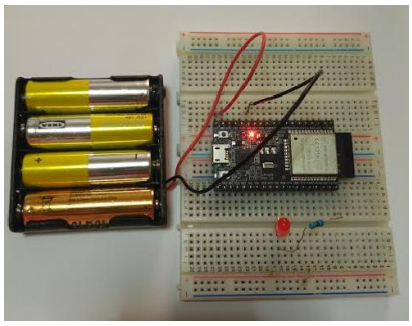

Ilustración 4-13: Hardware asociado al ESP32

4.6.2 Software

Una vez instalada la tarjeta en el entorno de Arduino, se podrá seleccionar la misma accediendo al menú "Herramientas" y a la opción "Placa". Dentro de este menú se podrá seleccionar la tarjeta "ESP32 DevModule".

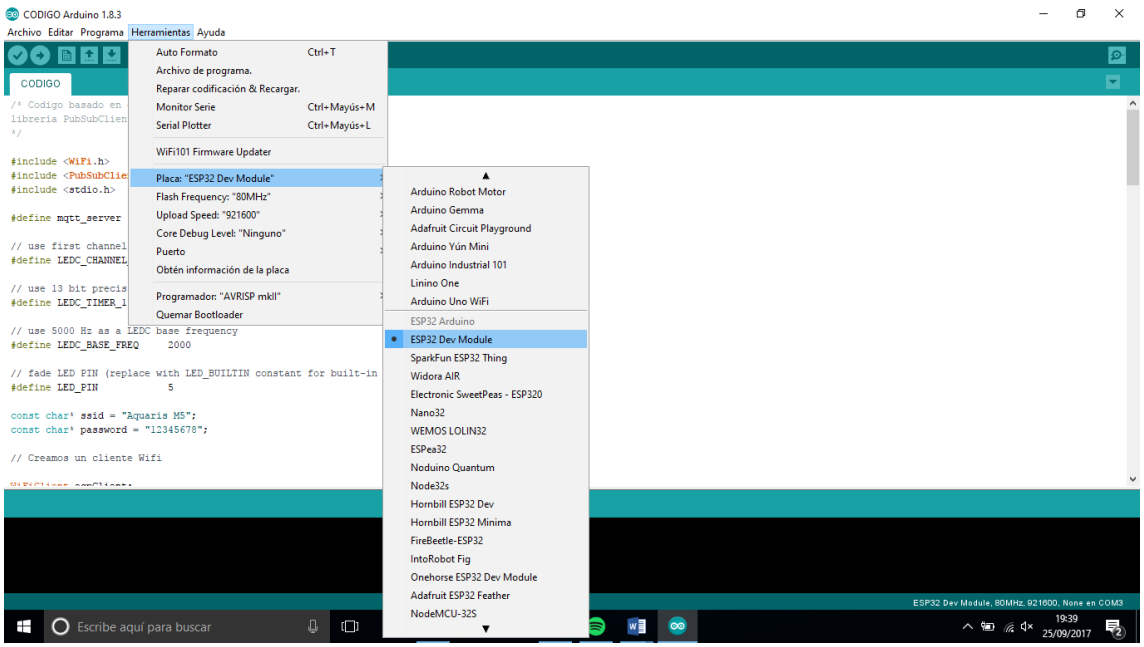

Ilustración 4-14: Selección de la tarjeta en el IDE de Arduino

A continuación, se va a describir el código que se ha desarrollado para el módulo ESP32, que como se ha comentado anteriormente es el encargado de activar y desactivar la luz.

La siguiente figura muestra la definición de la IP del bróker, así como los datos necesarios para establecer la conexión inalámbrica con la red WiFi. También se ha definido los parámetros de la función "ledcPin", que será la encargada de encender o apagar la luz.

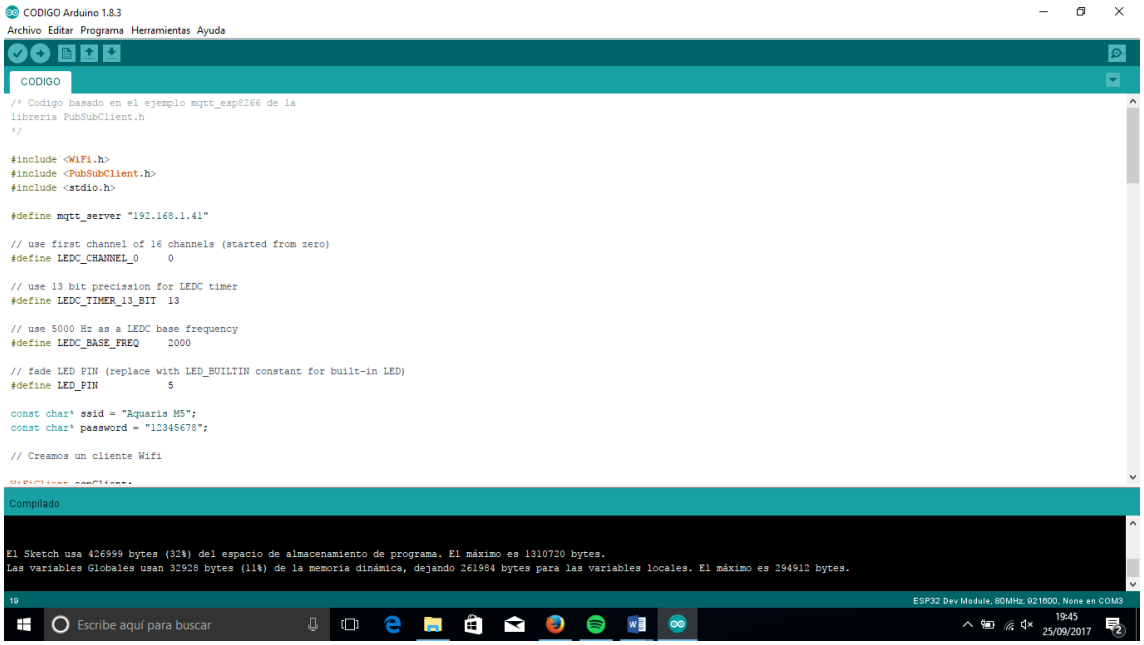

Ilustración 4-15: Cabecera y definición de los parámetros de ledcPin

En la siguiente ilustración, se muestra el código asociado a la declaración de un objeto de la clase "WifiClient". A continuación, se intenta conectar a la red wifi configurada.

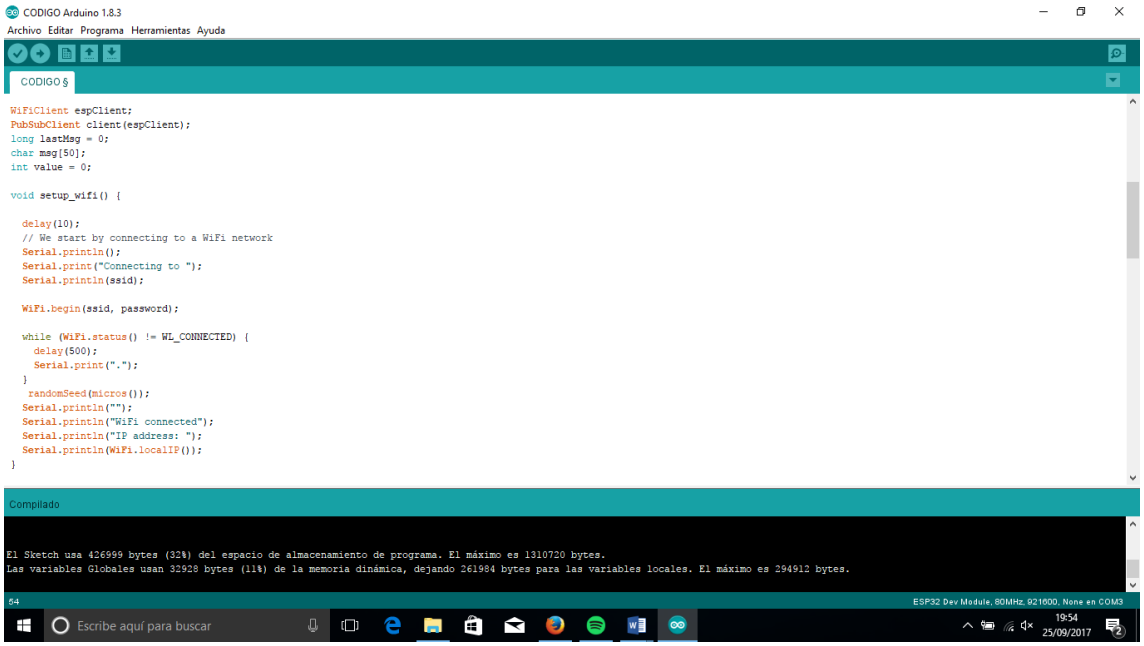

Ilustración 4-16: Configuración de un cliente wifi, y conexión a la red

Para el correcto funcionamiento del proceso es importante que la conexión a la red WiFi y al servidor MQTT no se pierda y se realice una reconexión automática en caso de que dicha conexión se haya perdido.

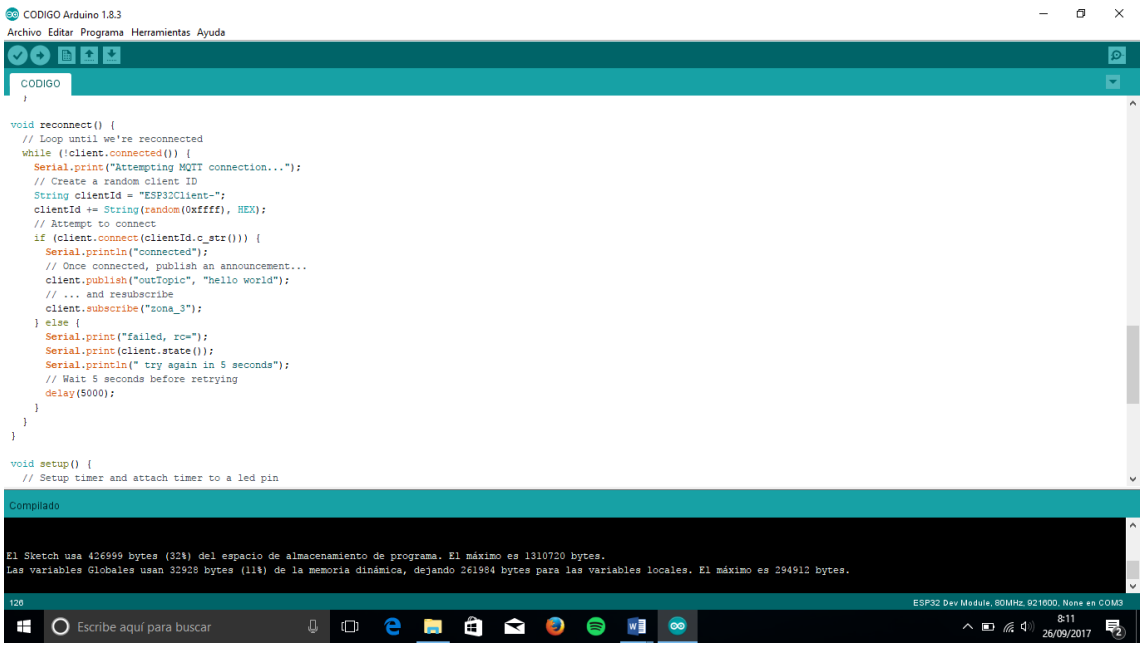

Ilustración 4-17: Función reconectar

A continuación, definimos la función "ledcAnalogWrite", que es la encargada de encender o apagar el LED. También se define la función "callback", en la que se obtiene el dato enviado por MQTT. Como lo que se envía es un *String* en código ASCII, lo que hay que hacer es pasarlo a *Int*, y por último se llama a la función "ledcAnalogWrite" para encender o apagar el LED utilizado.

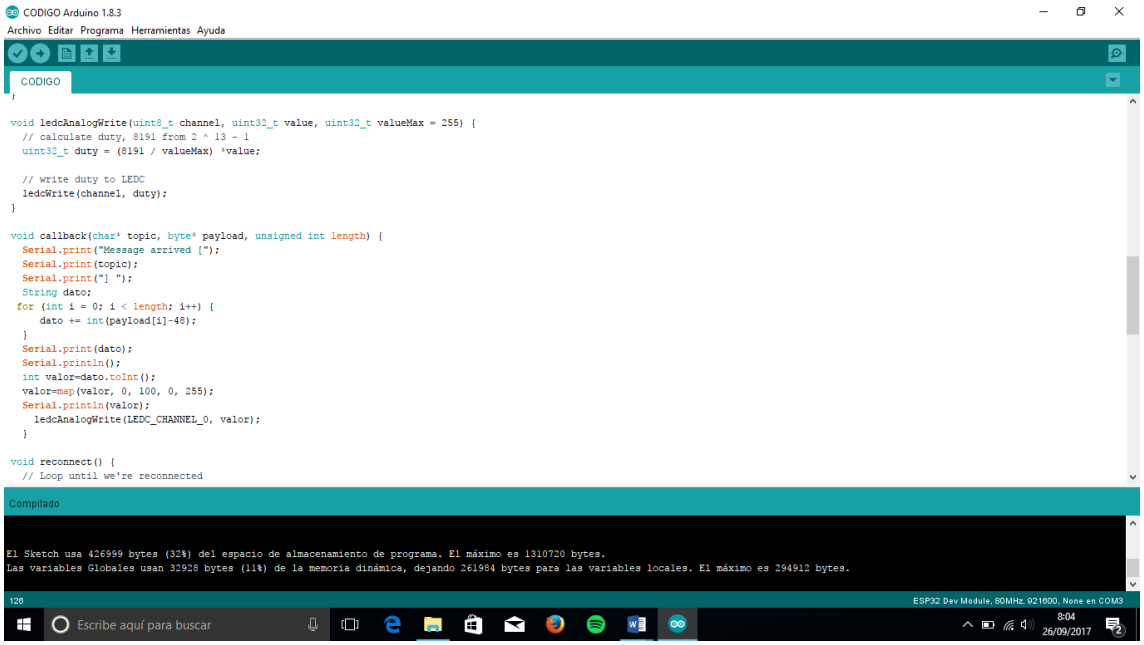

Ilustración 4-18:Declaración de las funciones ledcAnalogWrite y Callback

Por último, tenemos las funciones "*setup*" y "*loop*". La función "*setup*" es la encargada de configurar el pin del LED y de iniciar la comunicación MQTT. Dentro del método loop se comienza un bucle de conexión permanente (client. loop).

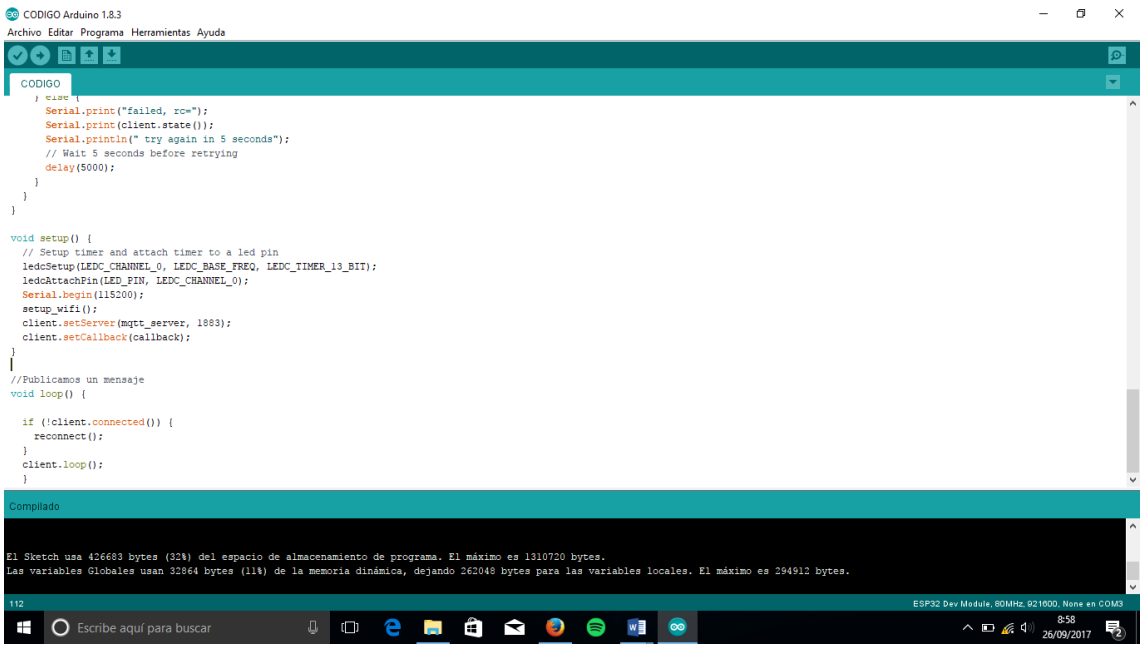

Ilustración 4-19: Funciones setup() y loop ()

Capítulo 5

Conclusiones y Trabajos Futuros

5.1 Conclusiones

En este proyecto se ha abordado el estudio de sistemas automatizados. Concretamente, se ha diseñado un sistema de iluminación inteligente, basado en dispositivos empotrados de bajo coste, eficiente, y que podría resultar de gran utilidad, tanto para el usuario como para el gerente de la empresa. Además de un ahorro de energía beneficioso para todos. Dicho sistema de iluminación ha sido el caso de estudio.

Se ha realizado un estudio para escoger los dispositivos empotrados y la programación de estos más adecuada. Para los dispositivos empotrados, se ha tenido en cuenta el coste, su facilidad de integración, sus prestaciones y su grado de extensión en el panorama actual.

Para desarrollar el bróker MQTT, se ha seleccionado Intel Computer Stick. En el caso de la comunicación entre dispositivos, se ha elegido los módulos ESP32 por su simplicidad y efectividad. Para la detección del usuario se ha escogido la tecnología *Beacon* por su interés en la aplicación en otros campos.

Se ha llevado a cabo un caso de estudio en el que se ha diseñado y ejecutado un modelo de iluminación inteligente. Para ello, se ha implementado el hardware necesario mediante la integración de sensores a los dispositivos empotrados y se ha obtenido el software mediante el desarrollo de los archivos pertinentes. En el caso del módulo ESP32, se ha creado el Sketch necesario para implementar todos los componentes software

necesarios. Finalmente, se ha creado una aplicación en Android que aporta la app para Smartphone. Con ella se pueden llevar a cabo la detección del usuario y la comunicación a través de protocolo MQTT con el sistema. Se ha utilizado este protocolo de comunicación por su ancho de banda mínimo, por su poco consumo de energía, por su rapidez, que posibilita un tiempo de respuesta superior al resto de protocolos web actuales y su gran fiabilidad y poco requisito de recursos y memorias

Finalmente, y a modo de resumen, con este estudio se ha pretendido comprobar la utilidad de uno de los muchos servicios posibles que pueden ser creados mediante la tecnología de sistemas inteligentes. La creciente evolución de las ciudades hacia la inteligencia y la automatización, da muestra del verdadero potencial de ésta tecnología.

5.2 Trabajos Futuros

Como posible trabajo futuro, se puede crear una aplicación de mayor dimensión mediante la introducción de más salas en las que hayan *beacons*. También se podría llevar a cabo un estudio de eficiencia energética calculado a partir de la potencia que consume cada luz.

Otro aspecto que podría ser explorado es la mejora del algoritmo desarrollado para el control de las luces, ya que se pueden introducir otros parámetros como la luminosidad que hay en ese momento en la sala, o que no se supere un consumo máximo.

Finalmente, también podría ser explorado en el futuro el diseño y desarrollo de la app para el sistema operativo iOS, ya que, actualmente sólo se ha realizado para el sistema operativo Android.

Anexo I

Funciones Auxiliares de los Servicios Web REST

1.1 Funciones del Servicio Web "Encender"

Este anexo recoge el código fuente de los diferentes servicios que proporciona el servidor local de cada sala.

1.1.1 Luz Asociada

```
function LuzAsociada($idluz, $conn) {
      $sql= "SELECT topic FROM Luz WHERE idluz = \"";
      $sql=$sql.$idluz;
      $sql=$sql."\"";
      $result = mysqli_query($conn, $sql);
      if (mysqli_num_rows($result) > 0) {
            if($row = mysqli_fetch_assoc($result)) {
                  $topic = $row["topic"];
            }
            } else {
                  $data["error"] = true;
                  $data["message"] = "Fallo Obtener Topic";
                  $newResponse = $response->withJson($data);
                  $newResponse = $newResponse->withHeader('Content-
type', 'application/json');
                  return $newResponse;
            }
      return $topic;}
```

1.1.2 Comprobar Luz

```
function ComprobarLuz($idluz, $ID, $conn) {
	$sql= "DELETE FROM `Estudiantes` WHERE `DNI` = \"";
	$sql= $sql.$ID;
	$sql=$sql."\"";
	$result = mysqli_query($conn, $sql);
	if($result){
		$sql= "SELECT idluz FROM Estudiantes WHERE idluz = \"";
		$sql=$sql.$idluz;
		$sql=$sql."\"";
		$result = mysqli_query($conn, $sql);
		if (mysqli_num_rows($result) > 0) {
			$rsp=true;
		} else {
			$rsp=false;
		}
	return $rsp;
	}
}
```

1.1.3 Calcular

```
function Calcular($idluz, $conn){
	$sql = "SELECT estado FROM Estudiantes WHERE idluz = \"";
	$sql=$sql.$idluz;
	$sql=$sql."\"";
	$Estado=$i=0;
	$result = mysqli_query($conn, $sql);
	while($row = mysqli_fetch_assoc($result)) {
		$Estado1 = $row["estado"];
		$Estado=$Estado+$Estado1;
		$i++;
	}
	$Estado=$Estado/$i;
	return $Estado;
}
```

1.1.4 Beacon Asociado

```
function BeaconAsociado($UUID, $major, $minor, $conn) {
	$sql= "SELECT idbeacon FROM Beacon WHERE UUID = \"";
	$sql=$sql.$UUID;
	$sql = $sql. "\" AND major = \"" ;
	$sql=$sql . $major;
	$sql = $sql . "\" AND minor = \"";
	$sql=$sql.$minor;
	$sql = $sql . "\"";
	$result = mysqli_query($conn, $sql);
	if (mysqli_num_rows($result) > 0) {
		if($row = mysqli_fetch_assoc($result)) {
			$idluz = $row["idbeacon"];
		}
```

```
      } else {
            $data["error"] = true;
            $data["message"] = "Fallo Obtener Luz";
            $newResponse = $response->withJson($data);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
      }
return $idluz;
}
```

1.1.5 Registro Estudiante

```
function RegistroEstudiante ($idluz, $Estado, $ID, $conn){
      $sql= "INSERT INTO `Estudiantes`(`idestudiantes`, `idluz`,
`DNI`, `estado`) VALUES (NULL,'";
      $sql = $sql . $idluz;
      $sql = $sql . "', '";
      $sql = $sql . $ID;
      $sql = $sql . "', '";
      $sql = $sql . $Estado;
      $sql = $sql . "')";
      $result = mysqli_query($conn, $sql);
      if($result == FALSE){
            $msg["error"] = true;
            $msg["message"] = "Error registrando el estudiante en la
Base de Datos";
            $newResponse = $response->withJson($msg);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
                  return $newResponse;
            }
      }
```

1.1.6 Estado Luz

```
function EstadoLuz ($idluz, $conn, $Estado) {
      $sql= "SELECT estado FROM Luz WHERE idluz = \"";
      $sql=$sql.$idluz;
      $sql=$sql."\"";
      $result = mysqli_query($conn, $sql);
      if (mysqli_num_rows($result) > 0) {
            if($row = mysqli_fetch_assoc($result)) {
                  $Estado1 = $row["estado"];
                  $Estado=($Estado+$Estado1)/2;
            }
      } else {
            $data["error"] = true;
            $data["message"] = "Fallo Obtener Estado";
            $newResponse = $response->withJson($data);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
            }
      return $Estado;
}
```

1.1.7 Nuevo Estado Luz

```
function NuevoEstadoLuz($Estado, $idluz, $conn) {
      $sql = "UPDATE `Luz` SET `estado`='";
      $sql = $sql. $Estado;
      $sql = $sql. "' WHERE idluz ='";
      $sql = $sql. $idluz;
      $sql = $sql. "'";
      $result = mysqli_query($conn, $sql);
      if($result == FALSE){
            $msg["error"] = true;
            $msg["message"] = "Error registrando la luz en la Base de
Datos";
            $newResponse = $response->withJson($msg);
            $newResponse = $newResponse->withHeader('Content-type',
'application/json');
            return $newResponse;
      }
}
```

1.1.8 Registro Acceso

```
function RegistroAcceso ($idluz, $Estado, $conn) {
      $sql= "INSERT INTO `Registro`(`idregistro`, `idluz`, `Estado`,
`timestamp`) VALUES (NULL,'";
      $sql = $sql. $idluz;
      $sql = $sql. "', '";
      $sql = $sql. $Estado;
      $sql = $sql. "', NULL)";
      $result = mysqli_query ($conn, $sql);
      if ($result == FALSE) {
            $msg["error"] = true;
            $msg["message"] = "Error registrando la luz en la Base de
Datos";
            $newResponse = $response->withJson($msg);
            $newResponse = $newResponse->withHeader ('Content-type',
'application/json');
            return $newResponse;
      }
}
```

1.1.9 Mandar Dato

```
function MandarDato($topic, $Estado) {
      // Parámetros de MQTT
      $host = "localhost";
      $port = 1883;
      $username = "username";
      $password = "password";
      //MQTT client id to use for the device. "" will generate a
client id automatically
      $mqtt = new phpMQTT($host, $port, "ClientID".rand());
      if ($mqtt->connect(true,NULL,$username,$password)) {
```

```
            $mqtt->publish($topic, $Estado, 0);
            $mqtt->close();
            $data = "Dato mandado correctamente";
      }else{
            $data = "Error en la comunicacion MQTT";
      }
      return $data;
}
```

1.2 Funciones del Servicio Web "Insertar"

1.2.1 Comprobar

```
function Comprobar ($tabla, $param, $conn) {
      if($tabla=="Luz") {
            $sql = "SELECT topic FROM `Luz` WHERE topic = \"";
            $sql = $sql. $param;
            $sql = $sql. "\"";
      } else {
            $sql = "SELECT idbeacon FROM `Beacon` WHERE idbeacon = \"";
            $sql = $sql. $param;
            $sql = $sql. "\"";
      }
      $result = mysqli_query ($conn, $sql);
      if (mysqli_num_rows($result) > 0) {
            $rsp=true;
      } else {
            $rsp=false;
      }
      return $rsp;
 }
```

1.2.2 Insertar Luz

```
function InsertarLuz($tipo, $topic, $potencia, $conn) {
      $sql= "INSERT INTO `Luz`(`idluz`, `tipo`, `estado`, `topic`,
`potencia`) VALUES (NULL,'";
      $sql = $sql. $tipo;
      $sql = $sql. "', '0', '";
      $sql = $sql. $topic;
      $sql = $sql. "', '";
      $sql = $sql. $potencia;
      $sql = $sql. "')";
      $result = mysqli_query ($conn, $sql);
      return $result;
 }
```

1.2.3 Insertar Beacon

```
function InsertarBeacon($idbeacon, $UUID, $major, $minor, $conn) {
      $sql= "INSERT INTO `Beacon`(`idbeacon`, `UUID`, `major`,
`minor`) VALUES ('";
      $sql = $sql. $idbeacon;
      $sql = $sql. "', '";
      $sql = $sql. $UUID;
      $sql = $sql. "', '";
      $sql = $sql. $major;
      $sql = $sql. "', '";
      $sql = $sql. $minor;
      $sql = $sql. "')";
      $result = mysqli_query ($conn, $sql);
      return $result;
 }
```

1.2.4 Insertar Vecino

```
function InsertarVecino($luz1, $luz2, $conn) {
      $sql= "INSERT INTO `Tiene_Vecinos`(`idluz`, `idluz_vecino`)
VALUES ('";
      $sql = $sql. $luz1;
      $sql = $sql. "', '";
      $sql = $sql. $luz2;
      $sql = $sql. "')";
      $result = mysqli_query ($conn, $sql);
      return $result;
}
```

Bibliografía

ANDROID:

[1] ¿Qué es Android? Disponible online en: https://www.xatakandroid.com/sistema-operativo/que-es-android

[2] Conoce Android Studio. Disponible online en: https://developer.android.com/studio/intro/index.html?hl=es-419

ARDUINO

[3] ¿Qué es Arduino? Tecnología para todos. Disponible online en: http://arduinodhtics.weebly.com/iquestqueacute-es.html

BASE DE DATOS

[4] Introducción a las Bases de Datos. Disponible online en: http://es.ccm.net/contents/66-introduccion-a-las-bases-de-datos

BEACONS:

[5] Intro to a Estimote API´s. Disponible online en: http://developer.estimote.com/

[6] What is a Beacon? Disponible online en: https://kontakt.io/beacon-basics/what-is-a-beacon/

[7] 9 usos reales para comprender que son los Beacons. Disponible online en: http://www.nobbot.com/redes/9-usos-reales-comprender-los-beacons/

BLUETOOTH LOW ENERGY (BLE):

[8] Bluetooth Low Energy: el conocido desconocido. Disponible online en: https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido?lang=es

ESP32

[9] Aprendiendo Arduino: Wifi en Arduino. Disponible online en: https://aprendiendoarduino.wordpress.com/tag/esp32/

[10] Arduino a muerte: Presentación del módulo ESP32. Disponible online en: http://arduinoamuete.blogspot.com.es/2016/12/un-adelanto-del-modulo-esp32.html

[11] Comparativa y análisis completo de los módulos ESP8266 y ESP32. Disponible online en: http://blog.bricogeek.com/noticias/electronica/comparativa-y-analisis-completo-de-los-modulos-wifi-esp8266-y-esp32/

INTEL COMPUTE STICK

[12] Descripción del Intel Computer Stick. Disponible online en: https://www.cnet.com/es/analisis/intel-compute-stick-2015/primer-vistazo/

[13] Instalación del Sistema Operativo para Intel Compute Stick. Disponible online en: https://www.intel.la/content/www/xl/es/support/boards-and-kits/intel-compute-stick/000023437.html

[14] Sistemas Operativos compatibles con Intel Compute Stick. Disponible online en: https://www.intel.es/content/www/es/es/support/boards-and-kits/intel-compute-stick/000005899.html

[15]Introducing the Intel Computer Stick. Disponible online en: https://www.intel.es/content/www/es/es/compute-stick/intel-computestick-introduction-video.html

iOS

[16] iOS: El sistema Operativo de Apple. Disponible online en: http://culturacion.com/ios-el-sistema-operativo-movil-de-apple/

phpMyAdmin

[17] Página oficial de phpMyAdmin. Disponible online en: https://www.phpmyadmin.net/

PROTOCOLO HTTP

[18] El protocolo HTTP. Disponible online en: HTTP: http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html

PROTOCOLO MQTT

[19] ¿Qué es MQTT? Disponible online en: https://geekytheory.com/que-es-mqtt

RASPBERRY PI

[20] RPi educative fórum. Disponible online en: https://www.raspberrypi.org/forums/

REST WEB SERVICE

[21] Introducción a los Servicios Web RESTful. Disponible online en: https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful

SMARTPHONE

[22] Características y hardware de los dispositivos móviles. Disponible online en: https://mastermoviles.gitbooks.io/tecnologias2/content/caracteristicas_y_hardware_de_los_dispositivos_moviles.html

TEXAS INSTRUMENTS LAUNCHPAD

[23] Reseña Texas Instruments MSP430 Launchpad. Disponible online en: https://www.geekfactory.mx/resenas/texas-instruments-msp430-launchpad/

WINDOWS PHONE

[24] ¿Qué es Windows Phone? Disponible online en: http://conceptodefinicion.de/windows-phone/