SISTEMA DE ILUMINACIÓN INTELIGENTE EN CARRETERAS

INTELLIGENT STREET LIGHTING SYSTEM

EMANUEL GEORGIAN STANESCU

Dirigido por el Dr. Alberto A. Del Barrio y el Dr. Guillermo Botella Juan

Trabajo Fin Máster en Internet de las Cosas / Ingeniería Informática / Departamento de Arquitectura de Computadores y Automática / Sistemas Inteligentes

Facultad de Informática

Universidad Complutense de Madrid

Curso académico 2018-2019

Convocatoria de febrero

Calificación: 7 (Notable)

Autorización de difusión

EMANUEL GEORGIAN STANESCU

9 de enero de 2019

El abajo firmante, matriculado en el Máster en Internet de las Cosas de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: “Sistema de iluminación inteligente en carreteras”, realizado durante el curso académico 2018-2019 bajo la dirección de Dr. Alberto A. Del Barrio y el Dr. Guillermo Botella Juan del Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen

Con el presente Trabajo Fin de Máster se pretende abordar el diseño y la implementación de un sistema de iluminación inteligente en carreteras capaz de ofrecer la mejor iluminación en diferentes ubicaciones atendiendo a factores tales como la actividad medida o la cantidad de luz disponible. Con este tipo de sistemas se pretende reducir la contaminación lumínica de las ciudades, el coste de mantenimiento y el consumo de las luminarias instaladas en las carreteras actuales, así como abordar otros posibles problemas derivados del sistema de iluminación tradicional. Este trabajo se ubica en el contexto de las Smart Cities del Internet de las Cosas en donde sistemas que cuentan con sensores y actuadores mantienen una comunicación para cumplir con un funcionamiento determinado. Tratan, por tanto, de mejorar la calidad de vida de los habitantes y salvaguardar la de otros seres vivos.

Para abordar el problema se partirá de un diseño modular genérico para cada región de carretera que cuenta con sensores y actuadores que ayudarán a determinar las condiciones óptimas de iluminación de una región en un momento dado. Los nodos de cada región estarán conectados en red para propagar información relevante para el funcionamiento del sistema. Además, será posible recoger información de actividad de tramos, mandar información de forma periódica a un servidor y comunicar a los dispositivos el estado actual del tramo de carretera.

También se van a tratar otros aspectos relevantes tales como la comunicación con dispositivos externos por buses de datos con distintos protocolos, el uso de un protocolo de comunicación, las características de un sistema operativo de tiempo real y la programación en un entorno multihilo.

Palabras clave

iluminación inteligente sistema operativo tiempo real programación multihilo
protocolo de comunicación arquitectura en Internet de las Cosas
contaminación lumínica coste de mantenimiento
ciudad inteligente internet de las cosas

Abstract

In this thesis, an intelligent street lighting system capable of providing optimal lighting conditions to drivers will be designed and implemented. To achieve this behavior the system would measure road activity and the amount of light available. The purpose of this system is to reduce light pollution, maintenance cost and power consumption of currently installed and traditional road lights. Other related problems will also be addressed. The behavior of this system is related to Smart Cities in the context of the Internet of Things where devices with sensors and actuators can communicate important messages to the network and as such cooperate to achieve the desired behavior. Their purpose is to improve people's lives and protect those of other living beings.

To tackle this issue, we will create a generic modular piece of hardware for each region of the road that has sensors and actuators helping us determine the optimal lighting conditions always. The hardware of each region will be connected to the same network to be able to send important information that will be needed for a correct system operation. It will also be possible to get important data related to road activity, send data to a server in a periodic manner and communicate the current road state to devices.

Other important matters will be addressed like for example communication with external devices using different transmission protocols, the use of a communication protocol, the features of a real time operating system and multithread programming.

Keywords

intelligent street lighting real time operating system multithread programming
communication protocol Internet of Things architecture
light pollution maintenance cost
smart city internet of things

Índice de contenidos

Resumen iii
Palabras clave iii
Abstract iv
Keywords iv
Índice de contenidos 1
Agradecimientos 3
Capítulo 1: Introducción 4
1.1 Necesidad e importancia de la iluminación 4
1.2 Iluminación tradicional 5
1.3 Problemas derivados de la iluminación 7
1.4 Necesidad de un cambio 10
Capítulo 2: Internet de las Cosas 11
2.1 Introducción 11
2.2 Tecnologías utilizadas 13
2.3 Características 14
2.4 Arquitectura 16
2.5 Retos de desarrollo en IoT 20
Capítulo 3: Objetivo y arquitectura del sistema 23
3.1 Mejoras del sistema de iluminación tradicional 24
3.2 Trabajo existente 29
3.3 Arquitectura del sistema propuesto 32
3.4 Definición de los requisitos 36
Capítulo 4: Plan de trabajo 38
4.1 Tecnologías usadas 39
4.2 Hardware elegido para la implementación 52
4.3 Sistema operativo 62
4.4 Comunicación con los nodos y envío de mensajes 67
4.5 Lógica de funcionamiento 70
4.6 Plan de pruebas 82

Capítulo 5: Resultados obtenidos 88
Capítulo 6: Conclusiones y trabajo futuro 90
Capítulo 7: Introduction (English) 91
7.1 Lighting needs 91
7.2 Standard lighting 92
7.3 Lighting adverse effects 94
7.4 Need for a change 96
Capítulo 8: Conclusions and future work (English) 97
Bibliografía 98
Licencia de uso 100

Agradecimientos

A mis profesores sin los que no hubiese sido posible la adquisición de tanto conocimiento y por mostrarme la actitud que debo tener como profesional en el campo de la Ingeniería.

Capítulo 1: Introducción

1.1 Necesidad e importancia de la iluminación

Desde las primeras civilizaciones el ser humano ha creado distintos dispositivos para satisfacer las necesidades básicas humanas y de confort de la iluminación, ya sea usando una antorcha, una vela hecha de aceite vegetal o animal o un candelabro con abundantes decoraciones propias de la época [1].

Con el paso del tiempo se han sustituido los dispositivos de iluminación por combustión para dar lugar a aquellos que emplean electricidad (y gas inerte o semiconductores) para esta finalidad, dejando el uso de los antiguos únicamente para eventos oficiales, ceremonias religiosas o como elementos decorativos para el hogar. La innovación tecnológica y de la ingeniería junto a la mejora de las técnicas de fabricación supuso una revolución en el diseño de la iluminación, convirtiendo la iluminación LED y las lámparas de sodio en las más utilizadas actualmente para iluminar ciudades o carreteras.

1.2 Iluminación tradicional

Este trabajo se centra sobre todo en aspectos del tráfico en carreteras de alta, media y baja velocidad y podrá extenderse a los carriles bici.

Existe un gran número de vehículos que circulan por la noche y la falta de iluminación en carreteras durante la noche se asocia con un aumento de la cantidad de accidentes mortales. Esto se debe a que la propia oscuridad altera las capacidades visuales del conductor dando lugar a síntomas como la pérdida de agudeza visual, visión binocular deficiente, visión cromática limitada y menor percepción de los objetos laterales entre otros. A estas dificultades fisiológicas y psicológicas se les puede sumar la escasa información visual que ofrecen los faros y el desplazamiento del propio vehículo y de otros automovilistas de la carretera. Por tanto, destacamos la importancia del alumbrado en carreteras para disminuir los accidentes nocturnos.

Además, el alumbrado en carreteras se traduce en un importante consumo energético de un Ayuntamiento, así como la contaminación (lumínica o de emisiones de CO2) y por tanto es importante estudiar los procesos para poder ser capaz luego de ofrecer técnicas para ayudar a reducir dicho consumo y contaminación [4].

Los tipos de lámparas utilizadas en instalaciones de alumbrado público son los siguientes:

- Lámparas fluorescentes
- Lámparas de vapor de mercurio a alta presión
- Lámparas de vapor de sodio a baja presión
- Lámparas de vapor de sodio a alta presión
- Lámparas de mercurio con halogenuros metálicos
- Lámparas de descarga por inducción
- Lámparas de LED

Por lo general se pueden utilizar todo tipo de lámparas para iluminar, sin embargo, la mayoría de las instalaciones se diseñan en un 80% con lámparas de vapor de sodio de alta presión y el resto con vapor de mercurio de alta presión. En túneles y a causa de su alta eficacia luminosa se usan a menudo las lámparas de vapor de sodio de baja presión.

La regulación del nivel luminoso a ciertas horas de la noche será adecuada siempre y cuando se garantice la seguridad vial y la de los peatones y el sector objeto a estudio no sea uno con alto porcentaje de accidentalidad nocturna, zonas peatonales con riesgo significativo de criminalidad entre otros. La forma de regular dicha intensidad se puede hacer mediante el uso de balastos serie de tipo inductivo para doble nivel de potencia, reguladores estabilizadores de cabecera de línea o balastos electrónicos de potencia regulable [5].

1.3 Problemas derivados de la iluminación

En este apartado se documentarán algunos estudios o hechos relativos sobre los problemas que la iluminación puede generar en las ciudades y pueblos.

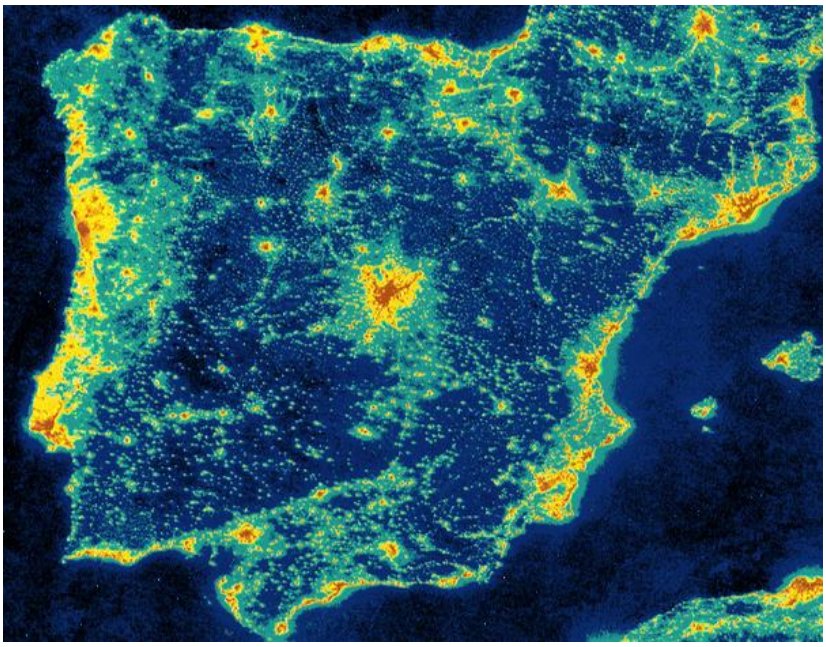

Ilustración 1: Mapa de contaminación lumínica de España (lightpollutionmap.info)

En primer lugar, la contaminación lumínica o resplandor luminoso nocturno en el cielo es uno de los problemas generados por la iluminación y ocurre por la difusión y reflexión de la luz artificial en los gases y partículas en suspensión de la atmósfera y afecta a la observación astronómica de los objetos celestes. Para paliar este inconveniente lo que se suele hacer es imponer un sistema de zonificación, en el que se procede al estudio de las zonas para analizar las exigencias fotométricas relativas a la actividad humana nocturna, la seguridad en la circulación de vehículos y peatones, calidad de vida, integridad del entorno, propiedades para determinar luego si es posible tener una solución para disminuir ciertos parámetros conducentes al resplandor luminoso nocturno. En definitiva, este sistema se impone para uniformizar la contaminación de forma

global disminuyéndola en zonas en las que es posible, puesto que estamos ante un caso en el que la contaminación causada por la iluminación no puede controlarse por completo [4].

Ilustración 2: Casa iluminada en el bosque (waterplace.net)

En segundo lugar, la luz artificial se lleva utilizando para iluminar zonas de noche durante cientos de años y continúa expandiéndose con la urbanización y el desarrollo económico. Varias investigaciones han demostrado que la luz artificial puede afectar al comportamiento de organismos invertebrados (insectos, arácnidos, moluscos, medusas, estrellas de mar o gusanos) entre otros, su éxito reproductivo, la supervivencia y en definitiva la composición de estas comunidades independientemente del momento del día. Esta variación de composición de las comunidades hace que aumente el número de depredadores y carroñeros en lugares bien iluminados. En definitiva, los resultados demuestran que la luz artificial cambia el entorno a nivel organizativo e incrementa la probabilidad de alterar su estructura y función del ecosistema [6].

Otra investigación afirma que la contaminación lumínica que afecta a los invertebrados es un problema global en crecimiento. Se han hecho pruebas con especies de murciélagos en peligro de extinción (Rhinolophus hipposideros) y se ha demostrado que afectan su desplazamiento o migración a causa de los insectos atraídos por las lámparas de vapor de mercurio [7].

Como último aspecto a contemplar acerca del tema de la iluminación artificial es el del consumo de energía, que según cómo se haya obtenido dicha energía puede resultar en un mayor o menor grado de contaminación. Es importante, por tanto, depender más de las energías renovables para reducir dicha contaminación.

1.4 Necesidad de un cambio

Resulta evidente que la tecnología ha impulsado variedad de campos y aplicaciones con el objetivo último de mejorar determinados procesos, por lo que en lo relativo al tema de la iluminación debemos mejorar la manera en la que iluminamos nuestras ciudades y pueblos siempre que sea posible.

Los sistemas convencionales de control de nivel luminoso no miden la actividad (entendido como el paso de vehículos) en la región en la que se instalan por lo que no permiten maximizar la utilidad y reducir el consumo energético durante el tiempo de menor actividad (por ejemplo, por la noche en carreteras menos transitadas).

Por tanto, se sugiere hacer uso de la tecnología para analizar la actividad en un tramo de carretera determinado y variar la intensidad de iluminación atendiendo a dicho parámetro durante la noche, maximizando la utilidad y minimizando el consumo de energía.

Capítulo 2: Internet de las Cosas

2.1 Introducción

El concepto de Internet de las Cosas (en inglés Internet of Things - IoT) fue acuñado por Kevin Ashton, un miembro de la comunidad de desarrollo de RFID (Radio Frequency Identification) en el año 1999 y hace referencia a una red distribuida de objetos físicos capaces de sentir o actuar en su entorno, comunicar con otros dispositivos o máquinas y compartir información, siendo estos interconectados por una red IP (Internet Protocol) pública o privada [8].

Ilustración 3: Internet of Things [8]

La visión que ofrece este paradigma es la presencia abundante en el entorno de una variedad de objetos o cosas que por medio de conexiones inalámbricas y cableadas y esquemas de direccionamiento únicos son capaces de interactuar el uno con el otro y colaborar con otros objetos para crear nuevas aplicaciones y servicios y alcanzar metas comunes. En este ámbito los retos de desarrollo e investigación para crear un mundo inteligente son enormes.

La diferencia es que además de conectar máquinas se pretende conectar una red de dispositivos de tamaño y tipo variables tales como vehículos, teléfonos móviles, electrodomésticos, cámaras, juguetes, instrumentos médicos, sistemas industriales, animales, personas o edificios con el objetivo de comunicar, compartir información, analizar dicha información y ser capaces de mejorar la calidad de vida de las personas o los procesos empresariales [8].

Se definen tres tipos de interacciones a través de Internet y son las siguientes:

- Comunicación persona a persona
- Comunicación persona a máquina (u objeto)
- Comunicación de máquina a máquina

La comunicación máquina a máquina hace referencia al procesamiento y al compartir los datos entre las mismas. Por otro lado, este paradigma incluye a las personas como participantes en esta red global.

La idea de conectar dispositivos físicos a Internet no es una idea nueva, sin embargo, el rápido descenso en el coste de los sensores y la tecnología RFID, el aumento de la cobertura y disponibilidad de las redes inalámbricas (Wi-Fi o Bluetooth) y redes móviles (2G, 3G o 4G) han hecho posible la aparición de nuevas oportunidades. Su uso práctico a nivel mundial comienza a proliferar en la actualidad por varias causas, entre ellas el crecimiento del número de dispositivos conectados a Internet (necesidad de IPv6 para poder direccionarlos), la comunicación de pequeños dispositivos empotrados independientes de la localización, la computación en la nube o el análisis de datos.

2.2 Tecnologías utilizadas

El concepto de Internet de las Cosas no hace referencia a una única tecnología sino a una mezcla de variadas tecnologías Hardware y Software. Estas tecnologías se usan para capturar, almacenar y procesar información que luego será transmitida a otros dispositivos o grupos de dispositivos [8].

Las tecnologías del IoT se pueden agrupar en tres categorías:

- Tecnologías que permiten a los objetos capturar datos
- Tecnologías que permiten a los objetos procesar los datos
- Tecnologías que mejoran la seguridad y privacidad

Las primeras dos categorías se pueden ver como un bloque funcional necesario para construir la inteligencia de los objetos, que es lo que diferencia el IoT del Internet habitual. La tercera categoría es más bien un requisito de facto impuesto para facilitar la penetración del IoT en el mercado. La comunicación en este caso se extiende a través de Internet y llega a todos los objetos que nos rodean, pudiendo realizar no solo la comunicación maquina a máquina sino también comunicar con redes inalámbricas de sensores, redes de sensores a través de tecnologías como 2G, 3G, 4G, GSM, GPRS, RFID, Wi-Fi, GPS, microcontroladores o microprocesadores entre otros. Estas son las tecnologías que hacen posibles las aplicaciones IoT.

Como se ha visto existe una mezcla heterogénea de tecnologías de comunicación que deben ser adaptadas para atender ciertas necesidades del IoT como pueden ser la eficiencia energética, la velocidad, la seguridad y la confianza. El objetivo es que la variedad de tecnologías disponibles pueda ser capaces de converger a un pequeño conjunto de ellas que permitan satisfacer las necesidades y estén apoyadas por alianzas tecnologías fuertes (necesidad de estándares en la comunicación).

2.3 Características

Las características principales del Internet de las Cosas se resumen de la siguiente manera [8]:

- Interconectividad: capacidad de conectar cualquier dispositivo con la infraestructura global de la información y comunicación.

- Servicios relacionados con los objetos: dentro de las capacidades de los objetos será posible ofrecer servicios relacionados tales como la protección de la privacidad y la coherencia semántica entre los objetos físicos y virtuales. Para conseguir este objetivo se debe hacer un cambio tanto en las tecnologías del mundo físico como la del mundo de la información.

- Heterogeneidad: los dispositivos son heterogéneos puesto que pueden ser de distintas plataformas hardware o redes de comunicaciones. Por tanto, destaca la capacidad de interactuar con otros dispositivos o servicios a través de redes distintas.

- Cambios dinámicos: el estado de los dispositivos cambia dinámicamente, por ejemplo, al dormirse o despertarse (variación en la actividad), conectarse o desconectarse, así como cambios en la ubicación geográfica o velocidad de conexión. Además, el número de dispositivos conectados puede variar de forma dinámica.

- Escalabilidad masiva: la cantidad de dispositivos que se comunicarán entre ellos y tendrán que ser administrados será por lo menos un orden de magnitud mayor que los dispositivos actualmente conectados a Internet. Sin embargo, un punto más crítico a considerar es el de la administración de los datos generados y su interpretación para

uso en aplicaciones. Este aspecto está relacionado con la semántica de los datos, así como el manejo eficiente de los datos.

- Seguridad: se trata de un aspecto que no debe ser dejado de lado y se deben diseñar aplicaciones pensadas en la seguridad. Se debe proteger la seguridad física de los usuarios, así como la de sus datos por lo que se pueden aplicar técnicas de seguridad en puntos de conexión, redes o los datos que viajan por las mismas asegurando un paradigma de seguridad escalable.

- Conectividad: esta característica ofrece accesibilidad (ser capaz de intercambiar información) y compatibilidad (usar un protocolo común para producir y consumir datos) a un dispositivo en la red.

2.4 Arquitectura

La arquitectura cuenta con varias capas de tecnologías relacionadas que la sustentan donde es importante destacar la escalabilidad, modularidad y configuración de los despliegues IoT en diferentes escenarios. La figura adjunta más abajo pretende mostrar una arquitectura más detallada de los posibles escenarios. A continuación, se van a describir cada una de las capas funcionales de las que está compuesto esta arquitectura [8]:

- Capa sensor o dispositivo inteligente: la capa más baja está formada por objetos inteligentes que integran sensores. Los sensores son elementos capaces de medir las propiedades físicas y convertirlas en señales eléctricas que pueden ser interpretadas por un instrumento. Los sensores permiten la interconexión del mundo físico y digital y que sea posible la recolección y procesamiento de la información en tiempo real. Estos componentes tienen la capacidad de tomar mediciones tales como la temperatura, la calidad del aire, la velocidad, la humedad, la presión, el flujo, el movimiento o la electricidad entre otros. En muchos casos también cuentan con cierto grado de memoria que les permite almacenar varios valores de mediciones. Se agrupan atendiendo a su propósito como pueden ser los sensores medioambientales, sensores del cuerpo humano, sensores para electrodomésticos o para monitorización remota de vehículos entre otros grupos. Los sensores se pueden agrupar atendiendo al tipo de conexión de la que precisan, ya sea conexión al Gateway o conexión directa con el backend de las aplicaciones o servidores. En el primer caso dicha conexión se puede realizar por LAN (Local Área Network) como Ethernet y Wi-Fi, o bien por PAN (Personal Área Network) como ZigBee, Bluetooth o UWB (Ultra Wideband). Para el segundo caso, los sensores que no precisan de conexión con concentradores pueden conectarse con el servidor o aplicación usando una red WAN (Wide Área Network) como puede ser GSM, GPRS y LTE. Los sensores que consumen poca energía y transmiten datos a baja velocidad normalmente forman una red comúnmente conocida como WSNs (Wireless Sensor

Network). Este tipo de red destaca por la capacidad de integrar más cantidad de nodos sensores, tener un consumo de batería adecuado y cubrir una amplia área de terreno.

- Gateways y redes: se necesita una infraestructura de red cableada o inalámbrica robusta y de alto rendimiento para poder transportar el volumen de datos masivo producido por los sensores. Las redes habituales cuentan con varios protocolos de comunicación y se han usado para hacer posible la comunicación M2M (Machine to Machine) en sus aplicaciones. Ahora se necesita dar soporte a un conjunto de aplicaciones y servicios más amplios (aplicaciones IoT) tales como los servicios transaccionales de alta velocidad, aplicaciones dependientes del contexto, entre otros, por lo que se precisa que las múltiples redes con varias tecnologías y protocolos de acceso trabajen conjuntamente en una configuración heterogénea. Estas redes pueden ser privadas, públicas o ser un modelo híbrido orientado a la latencia, ancho de banda y seguridad.

- Capa de administración: es la capa en la que realiza el procesamiento de los datos mediante analíticas, controles de seguridad, modelización de procesos y administración de dispositivos. Unas características importantes de esta capa son las reglas de negocio y de procesamiento. IoT ofrece conexión e interacción con objetos y sistemas proporcionando información en forma de eventos o datos contextuales como la temperatura de los bienes, ubicación actual y métricas de tráfico de datos. Algunos de estos eventos necesitan ser filtrados o enrutados a sistemas de captura de datos para almacenarlos y otros necesitan una respuesta inmediata a determinadas situaciones como actuar ante una emergencia en la salud de un paciente. Las reglas nos ayudan a imponer una lógica de decisión y desencadenar una respuesta y proceso automático en sistemas que necesitan ofrecer una acción inmediata. En términos de analítica de los datos, existen multitud de herramientas usadas para extraer información de los datos capturados por los sensores (raw data) y ser procesados rápidamente. Estos datos de carácter masivo se pueden analizar en memoria RAM (Random Access Memory) con el objetivo de reducir el tiempo de respuesta de la consulta en la base de datos y aumentar la velocidad en la toma de decisiones. También se pueden analizar los datos en streaming, un flujo continuo de datos en los que se debe contar con procesos que los analicen en tiempo real para

tomar una decisión en cuestión de segundos. La administración de los datos es necesaria para poder acceder, integrar y controlar el flujo de información. Con esta capa se priva a los procesos de las capas de aplicación de tener que procesar información innecesaria y reducir el riesgo de incumplir la privacidad. Para ello en esta capa se aplican filtros de anonimización, integración y sincronización de los datos con el objetivo de ocultar información detallada dejando únicamente la información relevante para las aplicaciones. La seguridad debe ser impuesta desde la capa más baja de los objetos inteligentes hasta la capa más alta de aplicación con el objetivo de evitar el hacking del sistema y que personal no autorizado comprometa la información, reduciendo de esta manera los riesgos potenciales.

- Capa de aplicación: se trata de la capa más alta de la arquitectura, en donde residen las aplicaciones que van a ofrecer un servicio o dar soporte en campos de aplicación inteligentes como pueden ser el transporte, la edificación, la ciudad, estilo de vida, venta de productos, la agricultura, cadenas de abastecimiento, sanidad, fabricas, emergencias, el turismo, la energía o el medio ambiente.

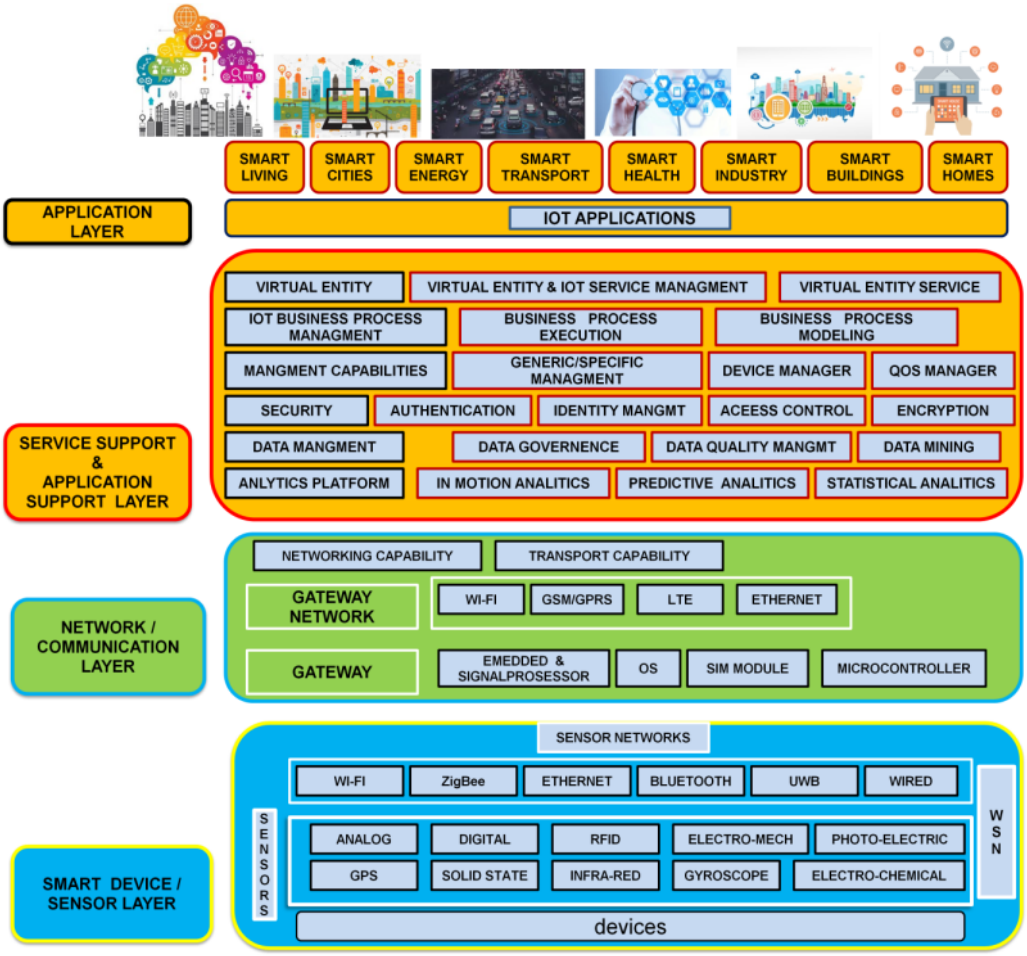

Ilustración 4: IoT Architecture [8]

2.5 Retos de desarrollo en IoT

El primero de los puntos a considerar es la **aplicación de estándares** que ayuda a impulsar las nuevas tecnologías en el mercado. Si los dispositivos de distintos fabricantes no operan bajo el mismo estándar se dificulta la interoperabilidad y requiere de gateways adicionales para hacer posible la comunicación entre ellos. La aplicación de estándares no solo se aplica a la comunicación entre dispositivos sino también al formato de los datos capturados. Si los datos están en distintos formatos los consumidores se verán atados a una familia de productos impidiendo la fácil transferencia de sus datos en caso de reemplazar el dispositivo por otro de un fabricante distinto y habiendo perdido el valor de los datos acumulados en el tiempo en el peor de los casos. La mayor parte del desarrollo en IoT ocurre en mercados verticales, donde se aplican técnicas de transmisión y topologías especificas (modelos centralizados donde los dispositivos se conectan directamente a internet o modelos descentralizados en donde los objetos están físicamente cerca y pueden hallarse e intercambiar información entre ellos). Grupos de Internet, organizaciones de telecomunicaciones y de negocios incluyendo al ETNO (European Telecommunications Operators Association) mantienen un debate en si las autoridades deben interferir en la adopción de estándares comunes o si deben ser neutrales para no ralentizar la innovación y la competencia. En Europa se reconoce que las autoridades tienen un papel relevante a la hora de establecer estándares e interoperabilidad en términos de formato de los datos, redes de transmisión y mecanismos de seguridad animando el consenso en los estándares con otros gobiernos y cuerpos de estandarización globales y emiten proyectos para adoptar ciertos estándares [9].

La segunda idea por considerar es el tema del **espectro de radio** utilizado en las comunicaciones. Con el crecimiento en el número de dispositivos inalámbricos conectados es evidente que se va a necesitar un espectro de radio más amplio. Para la elección del espectro se deben tener en cuenta el alcance de distintas tecnologías de conexión utilizadas en el IoT como pueden ser el Wi-Fi (uso de espectro sin licencia) o la red móvil inalámbrica (uso de espectro con licencia). Con el objetivo de aprovechar la producción masiva de dispositivos la elección del

espectro debe ser común en todos los países. Lo que hace aún más difícil la elección del espectro a utilizar es la larga vida útil de los dispositivos IoT (en torno a 30 años) contrario a la corta vida útil de los teléfonos móviles (entre 5 y 7 años) por lo que puede dificultar la eliminación de protocolos de comunicación antiguos. Por ejemplo, los operadores en Europa pueden tener problemas en el futuro para apagar el soporte de las comunicaciones 2G puesto que dejaría obsoletos todos los contadores inteligentes (de gas o electricidad, por ejemplo) que usan esta tecnología [9].

Los **riesgos de seguridad** son una amenaza tanto para los consumidores de servicios como para las empresas. Los cibercriminales podrían conseguir acceso no autorizado a dispositivos e interceptar comunicaciones inalámbricas con el objetivo de capturar datos sensibles, o bien, hacerse con el control de redes de área local o dispositivos para causar daño (enviando spam, apagando una fábrica o una red de abastecimiento de energía, alterando el funcionamiento normal de un dispositivo relacionado con la salud o haciendo que un coche se comporte de determinada manera con el objetivo de producir un accidente). Los servidores en el Cloud son un objetivo atractivo puesto que albergan una gran cantidad de datos agregados de dispositivos. Cifrar los datos puede ayudar a reducir los riesgos de seguridad, pero muchos de los dispositivos sensores del mercado carecen de capacidades energéticas y de computación necesarias como para implementar técnicas de cifrado sofisticadas. Actualmente no hay una solución perfecta a este problema, sin embargo, las empresas IoT pueden minimizar los riesgos teniendo en cuenta los temas de seguridad durante la fase de diseño de sus productos, recogiendo únicamente los datos necesarios, implementando varias capas de defensa contra amenazas, establecer controles de acceso en los dispositivos y actualizar el software con frecuencia para parchear las vulnerabilidades [9].

El último de los retos mencionados en este apartado tiene que ver con **la privacidad y la protección de los datos** de los clientes. Los dispositivos pueden recoger datos sobre la ubicación del usuario, sus hábitos diarios o el estado de su salud. A continuación, los datos pueden ser enviados a unos servidores centrales o ser compartidos con otros dispositivos o terceras entidades, a menudo sin la posibilidad de que el usuario estudie dichos datos a priori antes de ser enviados. Esto se debe a que muchos de los dispositivos portátiles carecen de grandes pantallas

táctiles complicando la información del usuario sobre posibles usos, así como pedir su consentimiento. El consentimiento del usuario se necesita no solo para recoger los datos sino también para analizarlos y compartirlos con terceras entidades. Las empresas pueden reducir los problemas recogiendo solo aquella información que es relevante para el propósito principal. Sin embargo, esto reduce la utilidad que se les puede dar a los datos ya que a menudo se combinan conjuntos de datos para hacer análisis en grandes cantidades y extraer patrones y modas contenidos en los mismos. Por ejemplo, los datos de patrones de conducción pueden ser usados para mejorar la seguridad vial. El usuario puede entonces decidir no interactuar con el dispositivo en absoluto, sin embargo, en muchas ocasiones esto impide que se haga un amplio uso de las funcionalidades del dispositivo. Como posible solución al tema de la privacidad y protección de los datos se podría imponer la anonimización de los datos desde los primeros momentos de la cadena de transmisión y procesamiento con el objetivo de reducir los riesgos, pero como ya se conoce no todos los dispositivos IoT tienen las capacidades de procesamiento necesarias para dicho procedimiento y los datos pueden tener determinada naturaleza que hacerlos completamente anónimos puede ser técnicamente difícil o imposible. Se conoce por varios estudios que los usuarios de Internet se sienten relativamente seguros y solo se ve comprometida su privacidad cuando los datos se usan fuera de contexto (por ejemplo, usar datos de patrones de conducción para evaluar los tipos de personalidad y luego ser usados en tomas de decisiones a la hora de contratar a un empleado). Además, el 72% de los usuarios de Internet en Europa ya se sienten preocupados por la privacidad y consideran que se les pide demasiados datos personales en línea [9].

Capítulo 3: Objetivo y arquitectura del sistema

En este trabajo se pretende ofrecer una mejora en el tema de la iluminación en carreteras desde un punto de vista tecnológico. Como ya se ha visto anteriormente, los sistemas convencionales de iluminación en carreteras no tienen en cuenta el aspecto fundamental de la actividad (paso de coches o personas) en una región determinada (basándose meramente en propiedades temporales o cantidad de luz disponible en determinadas regiones) incurriendo en un mayor gasto energético en aquellas regiones en las que la actividad no es tanta.

El objetivo principal es, por tanto, conseguir un sistema capaz de detectar y tener en cuenta la actividad en una región de carretera determinada para variar de forma inteligente las propiedades de iluminación en dicha región.

En primer lugar, se explicará detalladamente el funcionamiento deseado del sistema haciendo hincapié en los componentes involucrados y la relación entre ellos. También se van a comentar aspectos o posibles problemas derivados acerca de los módulos de comunicación o los responsables de la sensorización de la actividad.

A continuación, se dará a conocer el hardware especifico elegido y la arquitectura diseñada para implementar el funcionamiento explicado anteriormente. Se comentarán aspectos relacionados con la sensorizacion de datos del exterior, actuación con el exterior y la implementación de la lógica interna del algoritmo de funcionamiento.

3.1 Mejoras del sistema de iluminación tradicional

Como ya se ha explicado anteriormente, la tecnología y en concreto el campo de aplicación del Internet de las Cosas puede en este caso ofrecernos una serie de herramientas capaces de recoger información del exterior (de la misma manera o incluso de forma más precisa de la que lo haría un ser humano), procesarla de forma individual o juntamente con otros módulos, y generar una respuesta hacia el exterior. En nuestro caso, la información recogida del exterior es la relativa a la actividad de vehículos en una región de carretera determinada, la cantidad de luz disponible o la temperatura medida. La respuesta final de nuestro sistema debe ser el nivel de intensidad de iluminación deseado atendiendo a la actividad medida.

También se van a adoptar partes utilizadas en el sistema de iluminación convencional para saber cuándo se debe poner en funcionamiento el sistema (con ayuda de un sensor que mide la intensidad de la luz -fotorresistor), pues durante el día es muy probable que el nivel de luminosidad sea el suficiente como para no necesitar tener el sistema en funcionamiento.

Los datos recibidos de los sensores de paso de coche nos van a permitir medir la actividad de vehículos y ser capaces de controlar su entrada y abandono de una región de carretera. Para esta finalidad se pueden utilizar múltiples fuentes de información, ya sea una cámara de detección de matrícula, un sensor de inducción o un sensor de distancias entre otros. El correcto funcionamiento del sensor elegido es fundamental y todos tienen sus ventajas e inconvenientes, sin embargo, para el propósito de este trabajo se ha elegido usar una simulación de este evento mediante un sensor de detección de presencia.

Los datos relativos a los fotorresistores nos permiten saber si el sistema debe estar en funcionamiento o si es de día o de noche. Para medir este parámetro se puede usar una fuente de tiempo sincronizada con un servidor externo y una consigna establecida de antemano (similar al modo de funcionamiento del sistema de control de luminosidad crepuscular) o bien muestrear la intensidad de luz en una región determinada y compararla con una consigna medida

empíricamente de forma previa. Para esta implementación se ha elegido muestrear la cantidad de luz disponible en un área determinado mediante el uso de un fotorresistor.

La respuesta que el sistema debe generar hacia el exterior es un valor de intensidad lumínica, este puede manifestarse mediante un dispositivo actuador como puede ser un relé conectado con las luminarias de una región determinada. Como en este caso lo importante es probar el funcionamiento de una implementación determinada, se ha optado por usar una simulación de dicha intensidad lumínica mediante dos LEDs de salida. Por tanto, se ha decidido implementar tres niveles de intensidad lumínica siendo estos 0%, 50% ó 100% de intensidad.

Los criterios de elección de los sensores o actuadores a implementar en la realidad y su precisión quedan fuera del alcance de este trabajo, sin embargo, es obvio que los sensores deben ser capaces de recoger por separado los múltiples eventos de paso de coches siendo relevante para el correcto funcionamiento del sistema (en definitiva, el sistema llevará la cuenta del número de coches en cada sección para determinar los requisitos de intensidad lumínica).

A continuación, se va a explicar el funcionamiento deseado del Sistema de Iluminación Inteligente en Carreteras. Con el objetivo de abordar el problema y ofrecer una solución modular se va a dividir un tramo de carretera en varias secciones de la misma longitud. Esta división nos va a facilitar la gestión de cada sección por separado y poder tener mayor control de estas a la hora de realizar el funcionamiento deseado. Cabe destacar que el número de secciones en un tramo de carretera y la longitud de estas está relacionado con las características de las carreteras en cuestión y con el mecanismo de conexión elegido para los nodos. Por ejemplo, una carretera de longitud determinada con pocas salidas y/o entradas requiere de una sección más grande, pudiendo reducir la cantidad de dispositivos necesarios para controlarlo siempre y cuando se cuente con un mecanismo de conexión que permita más alcance. Se trata de una serie de aspectos tecnológicos y de características de las carreteras que debe ser estudiado previamente antes de instalar cualquier sistema de iluminación inteligente.

Cada una de las secciones de carretera contarán con los siguientes componentes:

- Un dispositivo hardware genérico: para la implementación de la lógica de funcionamiento y comunicación con el servidor.
- Un módulo de recogida de información del exterior: necesario para leer el nivel de luminosidad, la actividad de vehículos o información adicional como la temperatura.
- Un módulo de actuación o de control: llevará el nivel de luminosidad requerido a las lámparas en cuestión para variar su intensidad.

Ahora que conocemos todos los componentes necesarios en cada sección de carretera vamos a describir paso a paso cuál es el funcionamiento deseado del sistema ante el paso de vehículos con un ejemplo visual tal y como se puede observar en la ilustración 5:

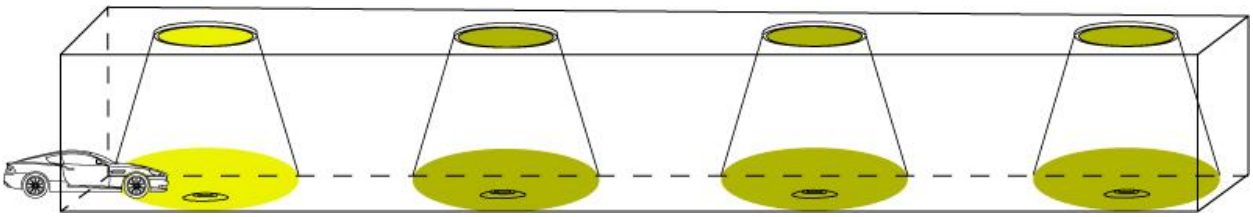

Ilustración 5: Tramo de carretera dividido en secciones (A) (elaboración propia)

El objetivo principal de este sistema es iluminar exclusivamente aquellas secciones donde hay actividad, por lo que al entrar el coche en la sección uno la iluminación de dicha sección debe estar al 100% de intensidad. Esto se puede producir ya que anteriormente se ha recibido determinada información que indica actividad próxima.

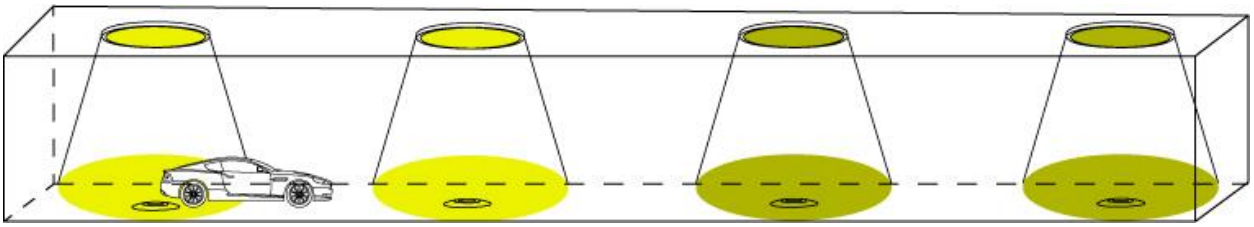

Ilustración 6: Tramo de carretera dividido en secciones (B) (elaboración propia)

A medida que avanza el coche por el tramo podemos observar en la ilustración 6 que la sección dos ha variado su intensidad de iluminación del 50% al 100% y esto se debe a que hay actividad cercana a dicha sección. Para hacer esto posible ha ocurrido antes otro evento que es la activación del sensor de actividad de la sección uno. Es evidente que se deben comunicar dichos eventos de actividad de vehículos para realizar el comportamiento deseado.

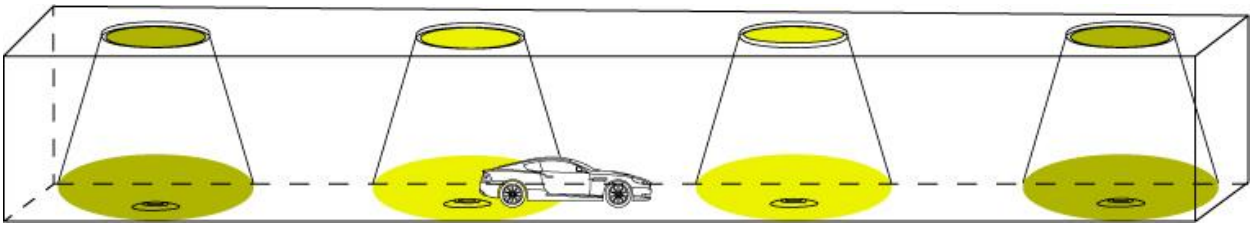

Ilustración 7: Tramo de carretera dividido en secciones (C) (elaboración propia)

El coche sigue avanzando y se activa el sensor de actividad de la sección dos lo que provoca la variación o incremento de la intensidad lumínica en la sección siguiente y el decremento de la intensidad en la sección anterior.

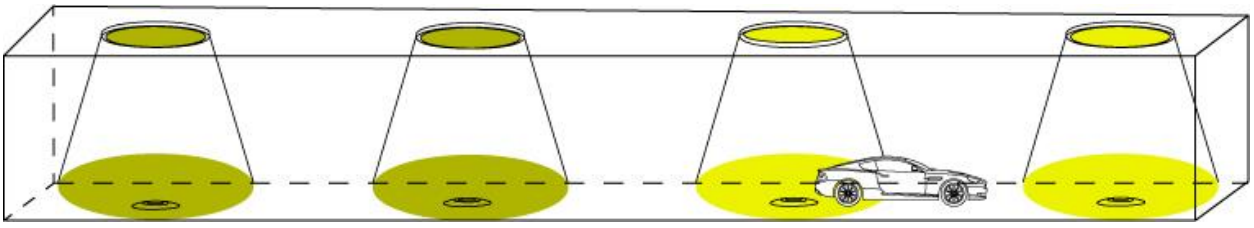

Ilustración 8: Tramo de carretera dividido en secciones (D) (elaboración propia)

El coche sigue avanzando y se activa el sensor de actividad de la sección tres lo que provoca el incremento en la intensidad lumínica en la sección siguiente y el decremento de la intensidad en la sección anterior.

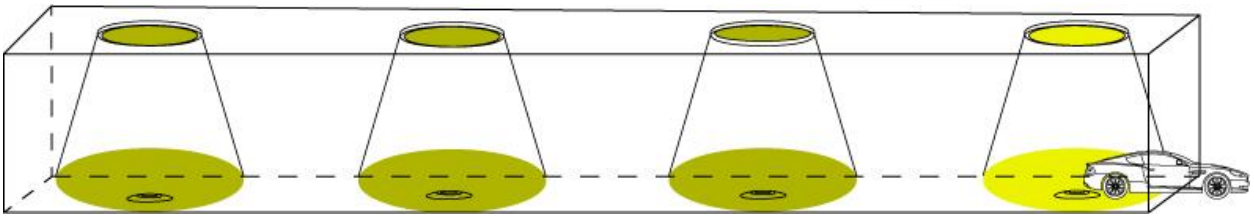

Ilustración 9: Tramo de carretera dividido en secciones (E) (elaboración propia)

Por último, el coche alcanza y activa el sensor de actividad de la sección cuatro lo que provoca el incremento en la intensidad lumínica en la sección siguiente y el decremento de la intensidad en la sección anterior.

Para la ilustración del funcionamiento de este sistema se ha considerado el ejemplo más sencillo posible, un tramo de carretera de un único sentido, carril y coche, sin embargo, no es raro encontrarse con distintos escenarios en la realidad donde se deben cubrir casos en los que los eventos concurrentes de activación deben tratarse por separado y hacer una correcta elección de los sensores. Cada una de las secciones deberá disponer de acceso a dispositivos externos como pueden ser los sensores y actuadores, módulo de procesamiento y memoria y acceso a una red de comunicaciones.

3.2 Trabajo existente

La necesidad de cambiar el sistema de iluminación tradicional (en el que la iluminación en carreteras se tiene a máxima intensidad durante la noche, incluso si no hay actividad de vehículos) por uno más inteligente resulta evidente y con ello se pretende conseguir un ahorro energético, ahorro en el coste de mantenimiento, reducción de los niveles de monóxido de carbono y contaminación lumínica.

El enfoque predominante en los sistemas de iluminación inteligentes existentes es un sistema centralizado en el que se instalan unos controladores en las luminarias y se hacen comunicar con un Gateway capaz de realizar una comunicación IP con un servidor determinado en el que se realiza la lógica de funcionamiento deseada. La comunicación de los controladores de las luminarias con el Gateway se puede realizar de múltiples maneras, bien sea de forma cableada (de forma dedicada o usando la línea de tensión existente de las luminarias adyacentes), o inalámbrica (redes WAN de largo alcance, GSM, LoRaWAN entre otros). El enfoque centralizado ofrece la ventaja de poder controlar remotamente las luminarias de una región determinada en caso de necesidad, poder conocer además las luminarias que deben ser cambiadas y poder planificar mejor el proceso de mantenimiento o poder actualizar el firmware de los dispositivos de forma inalámbrica sin necesidad de desplazarse a los nodos [10] [11].

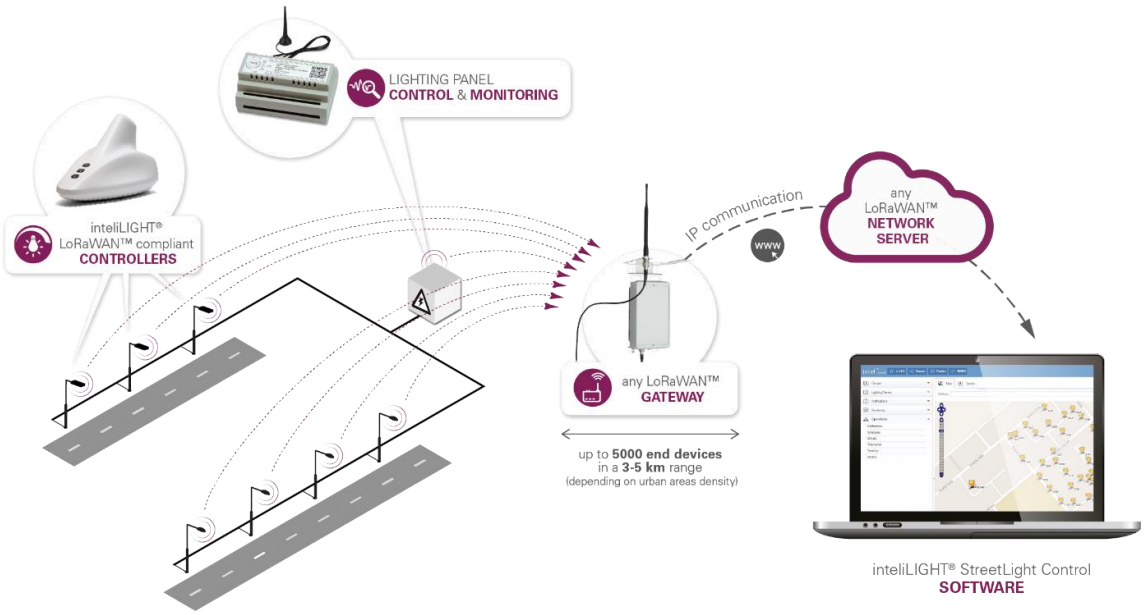

Ilustración 10: Arquitectura LoRaWAN de inteliLight [10]

El segundo de los enfoques destacables y el más novedoso sugiere un sistema distribuido en el que se crea una red mesh de sensores (un sensor sería en nuestro caso un controlador de luminaria independiente) en el que cada uno de ellos puede mandar eventos directamente a otras luminarias para informar de la futura actividad por parte de vehículos o peatones [12] [14].

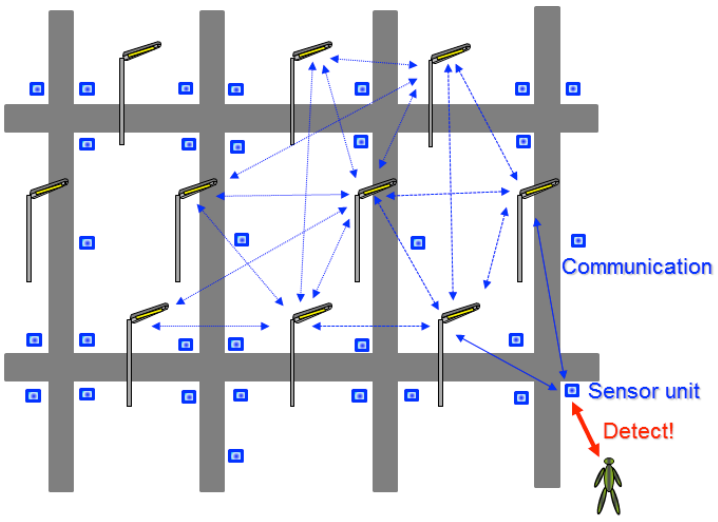

Ilustración 11: Red Mesh de Sensores [12]

En esta arquitectura distribuida se hace una comunicación directa entre los nodos y se eliminan elementos de arquitectura innecesarios para realizar el funcionamiento deseado. Sin embargo, se pueden complicar las tareas de actualización de firmware de los dispositivos o la variación del algoritmo de funcionamiento o la posibilidad de control remoto y recogida de datos de las luminarias. Esta arquitectura requiere de un algoritmo de enrutamiento para comunicar un nodo con sus nodos vecinos o adyacentes.

Estas son las arquitecturas más destacables elegidas a la hora de implementar este tipo de sistemas y la elección de uno u otro tipo de arquitectura está relacionado con las necesidades de cada aplicación. El tipo de arquitectura elegido para realizar el funcionamiento deseado en este trabajo es el centralizado con un Servidor y comunicación a través de un Gateway.

3.3 Arquitectura del sistema propuesto

Tal y como se ha ilustrado en el apartado anterior, se quiere diseñar un sistema de iluminación adaptativo para carreteras encargado de variar la intensidad lumínica atendiendo a la actividad medida en las carreteras y el nivel de luminosidad detectado en el ambiente.

En el campo de la ingeniería a menudo para ser capaces de ofrecer una solución a una necesidad determinada se opta por dividir el problema en módulos con una función independiente, que luego serán capaces de comunicar con otros módulos para conseguir algún tipo de trabajo cooperativo. Por tanto, cada una de las secciones de una carretera va a contar con un hardware genérico que nos ayude a abordar el problema y ser capaces de tener el control sobre las mismas.

El siguiente esquema corresponde a la arquitectura del hardware genérico necesaria en cada una de las secciones de carretera y en el mismo se pueden ver los bloques de los que está formado y la relación entre ellos:

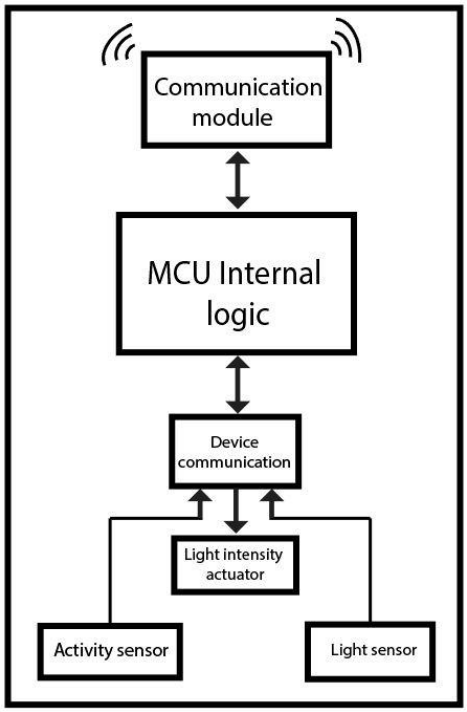

Ilustración 12: Arquitectura hardware de una sección de carretera (elaboración propia)

El elemento central de la arquitectura hardware de una sección es la unidad MCU (Microcontroller Unit) ya que permitirá la ejecución de un programa para comunicar con los dispositivos externos (recoger datos de los sensores y aplicar valores a los actuadores) y la comunicación en una red de comunicaciones. Este programa en un entorno de aplicaciones en tiempo real se traduce en un conjunto de hilos de ejecución (o programas independientes), con prioridades determinadas que se ejecutan cada cierto tiempo y comunican con otros para hacer algún tipo de trabajo cooperativo. Como en cualquier sistema de tiempo real se espera que los procesos cumplan con unas cotas de tiempo establecidas y que el algoritmo de la lógica de funcionamiento sea determinista. Con esta finalidad se suele utilizar algún tipo de sistema operativo RTOS (Real-Time Operating System) y es muy frecuente encontrarlos en sistemas o dispositivos empotrados en el Internet de las Cosas (IoT).

La información recogida de los sensores se va a comunicar a un servidor central en donde tenga lugar el análisis de los datos y la generación de datos de funcionamiento para los nodos de cada

sección (valores de predicción y funcionamiento básico). Además, los datos recogidos servirán para alimentar una interfaz de control e información en tiempo real que se encuentra en el servidor.

La unidad MCU cuenta con la posibilidad de comunicarse con sensores (nivel de luminosidad exterior, actividad de vehículos o sensor de temperatura) y actuadores (regulación del nivel luminoso) lo que le permite interactuar con el exterior (recoger y generar señales). La recogida de datos de los sensores se puede hacer mediante polling (leer valores de los sensores cada cierto tiempo ejecutando una instrucción de lectura) o instalando una interrupción software para ejecutar determinada función y poder de esta manera manejar la situación siempre que se produzca un determinado evento. La comunicación con los dispositivos externos se puede hacer mediante distintas interfaces o protocolos como pueden ser I2C, un conversor analógico - digital o un pin GPIO (General-Purpose Input / Output) del MCU.

Por otro lado, tal y como se ha visto en el apartado anterior, para implementar el funcionamiento deseado los nodos deben tener la capacidad de transmitir información (nodo entendido como el módulo de hardware genérico de una sección) y para ello la MCU debe contar con un módulo de comunicación que ofrezca dicha característica. En este caso se ha decidido crear un sistema con una comunicación centralizada, en la que los nodos comunican e intercambian información con un Servidor en red mediante un Gateway de forma inalámbrica.

En la ilustración número 13 adjunta más abajo se puede ver la arquitectura hardware completa del sistema:

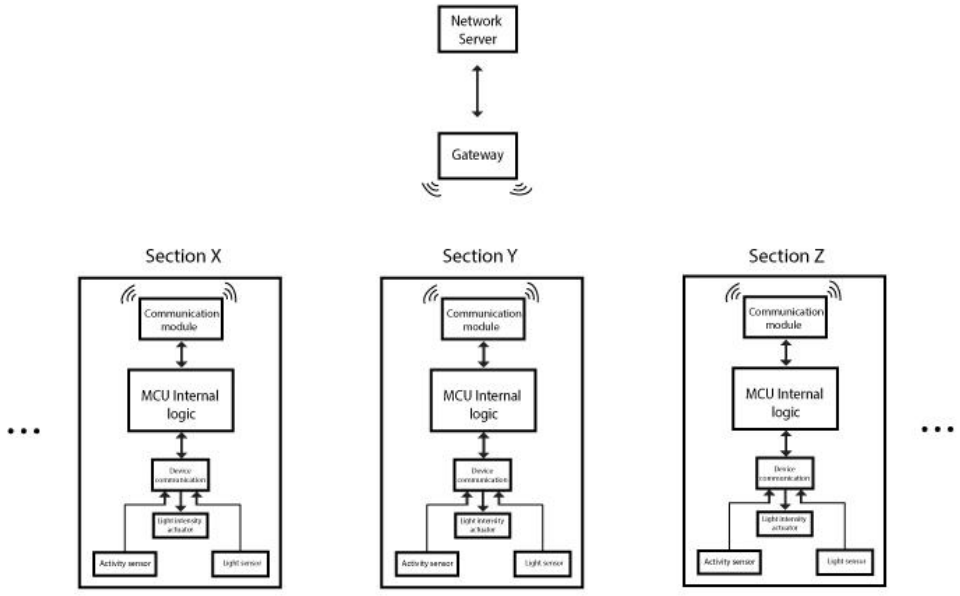

Ilustración 13: Arquitectura hardware del sistema (elaboración propia)

En la misma distinguimos en la parte inferior los módulos genéricos encargados de controlar las secciones de carretera y que cuentan con la posibilidad de recoger información del medio o actuar sobre el mismo. Los módulos se conectan de forma inalámbrica a una red común a través de un gateway.

En la misma red que los dispositivos y en la parte superior de la arquitectura encontramos el servidor con el que los dispositivos van a comunicar, recibiendo y enviando datos o eventos que nos permitirá realizar el comportamiento deseado.

3.4 Definición de los requisitos

En la arquitectura hardware del sistema presentada anteriormente se puede ver la interconexión de los distintos módulos necesarios para hacer funcionar el sistema. En este sistema centralizado, cada uno de los módulos es el encargado de unas tareas determinadas, las cuales se enuncian en las siguientes fases:

a) Generación y envío de la información: los nodos de cada sección serán capaces de obtener datos y actuar con el medio exterior. Mediante la interconexión con dispositivos externos (sensores y actuadores) se envían y obtienen señales, algunas de manera periódica y otras mediante interrupciones (se quiere obtener el evento tan pronto como se produzca). Estas señales se traducen en señales digitales mediante conversores analógico-digitales conectadas a un microprocesador. En nuestro caso los datos que se van a obtener de los sensores son el nivel de luminosidad ambiente (evento periódico), la temperatura exterior (evento periódico) y detección de actividad de vehículos (evento mediante interrupción). La información obtenida de los sensores será enviada a un servidor. La única información que se enviará a un actuador es el nivel de luminosidad que debe fijarse en la luminaria, de acuerdo con el funcionamiento del sistema. Para hacer posible la comunicación con estos dispositivos y la implementación de la lógica de funcionamiento en los nodos es necesario contar con módulos hardware capaces de procesar, almacenar y enviar datos por una red de comunicaciones.

b) Procesamiento o lectura de los datos: en el lado del servidor se van a almacenar los datos recibidos de los nodos en el tiempo, de tal manera que se puedan analizar y crear información nueva de funcionamiento. Para el sistema propuesto es posible funcionar en modo manual o modo automático. En el modo manual el nivel de luminosidad de cada sección se fijará en la interfaz de usuario del dashboard del servidor. En el modo automático el servidor va a establecer el nivel de luminosidad de acuerdo con la historia y actividad de la sección, el modo inteligente de funcionamiento propiamente dicho. Para

ello el servidor guardará estados internos de la carretera en el tiempo para averiguar el sentido de la actividad de vehículos y fijar el nivel de luminosidad de la sección antes de que se produzca la entrada de vehículos en la misma. Al mismo tiempo y nada más recibir los datos, el dashboard del sistema se va a actualizar para ofrecer al usuario información en tiempo real del funcionamiento del sistema y la posibilidad de controlar la intensidad de cada sección de forma manual, desactivando por tanto el modo automático.

c) Envío de datos de funcionamiento: el servidor, una vez establecido el modo de funcionamiento (automático o manual), debe comunicar a cada nodo los datos relativos al nivel de luminosidad de la luminaria que se debe fijar, de acuerdo con el funcionamiento establecido en cada caso. Se han establecido tres niveles de intensidad para las luminarias (0%, 50% y 100% de intensidad).

Capítulo 4: Plan de trabajo

En este apartado se va a comentar el hardware específico que se va a utilizar para implementar la arquitectura propuesta en el apartado anterior y ser capaces de tener un sistema que cumpla con el funcionamiento deseado. Para ello se debe profundizar en temas como el sistema operativo que gobierna dicho hardware, el modelo de programación, la lógica de funcionamiento necesaria y los temas relacionados con la comunicación entre elementos de la arquitectura. A posteriori se va a definir un plan de pruebas para verificar el correcto funcionamiento del sistema.

4.1 Tecnologías usadas

Con el objetivo de dotar a la arquitectura de funcionamiento se usan varias tecnologías que faciliten al desarrollador la implementación de la lógica de funcionamiento en los nodos y el servidor, y el protocolo de comunicación en la red que los conecta.

Los dispositivos van a realizar tareas de procesamiento de la información y comunicar con periféricos como pueden ser los sensores y actuadores. Para esta finalidad se usará el sistema operativo TI-RTOS con los correspondientes drivers de periféricos.

El servidor cuenta con varios procesos dentro de su arquitectura. El primer proceso es el broker MQTT que es el encargado de mantener los mensajes que se intercambian en la red de comunicaciones. Los mensajes están ordenados por topics y su contenido está relacionado con los datos de sensorización y funcionamiento de los dispositivos de cada sección de carretera.

El segundo proceso de esta arquitectura es el cliente Java Paho, una interfaz del programador cuyo papel es recibir todos los mensajes de sensorización de los dispositivos y calcular una lógica de funcionamiento para iluminar secciones de carretera. También se harán determinadas predicciones en base a la dirección del movimiento de los vehículos. Este proceso actualizará la información de funcionamiento de las secciones solo en el modo automático.

Por último, el proceso de Node - RED Dashboard es una interfaz de usuario de control y visualización de datos para la toma de decisiones y control del funcionamiento de las secciones del tramo de carretera en modo manual.

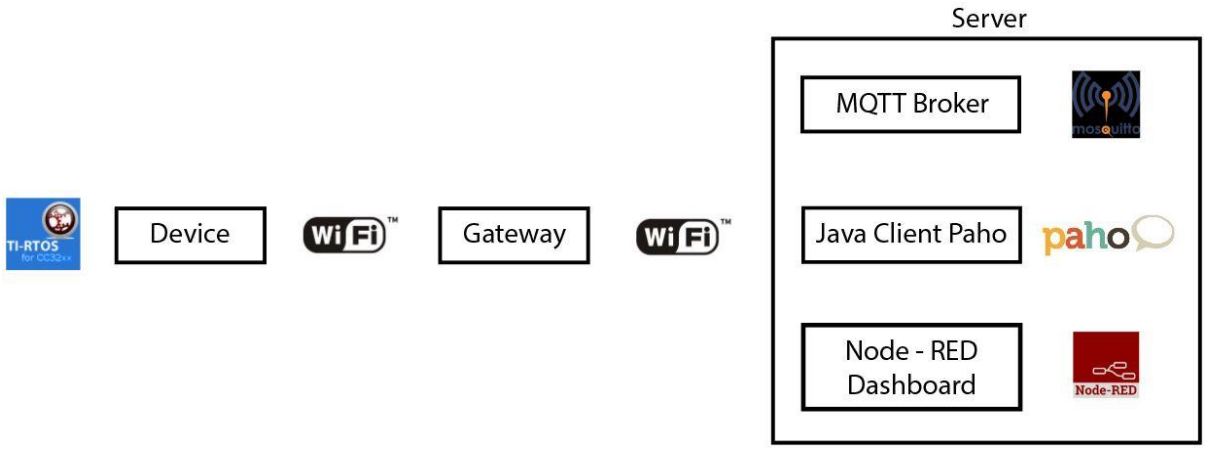

Ilustración 14: Tecnologías usadas en la arquitectura (elaboración propia)

TI-RTOS (Real-Time Operating System)

En el lado de los dispositivos inteligentes, aquellos que van a controlar una sección de carretera, se va a usar un sistema operativo de tiempo real, en este caso TI-RTOS (sistema operativo de tiempo real propietario de Texas Instruments), muy utilizados en los dispositivos inteligentes del Internet de las Cosas.

Este sistema operativo de tiempo real, y en concreto el SDK (Software Development Kit) de Simplelink ofrece un nivel de abstracción al programador y facilita la comunicación con dispositivos externos, el procesamiento y la comunicación en una red de comunicaciones, pues incluye las capas de comunicación necesarias para este tipo de tareas, tal y como se puede ver en la figura adjunta en la pila de componentes sobre la que se sustentarán las aplicaciones:

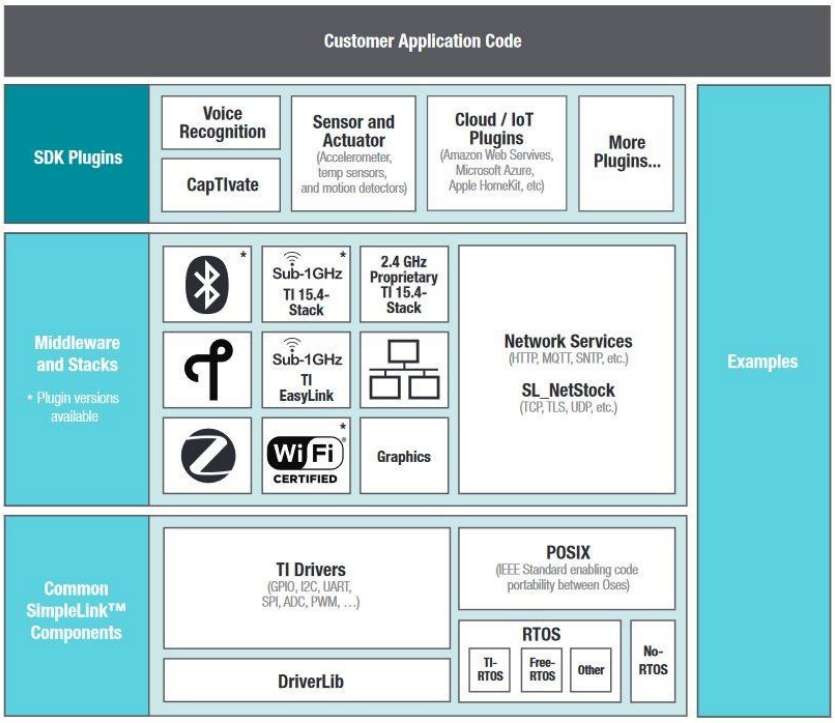

Ilustración 15: Pila de componentes RTOS (guía placa CC3220SF de Texas Instruments)

En la capa más baja de la pila de componentes nos encontramos con diversos protocolos de comunicación con dispositivos externos disponibles como pueden ser el UART (Universal Asynchronous Receiver-Transmitter), el conversor analógico digital ADC (Analog-to-digital Converter), SPI (Serial Peripheral Interface Bus), PWM (Pulse-width Modulation) o GPIO (General-purpose input/output).

El kernel de dicho sistema operativo ofrece servicios multitarea y de tiempo real como pueden ser la temporización o la planificación de tareas. Para los procesos del sistema se cuenta con el estándar IEEE POSIX (Portable Operating System Interface), un API (Application Programming Interface), creado para ofrecer una baja latencia, compatibilidad entre sistemas operativos y facilitar la migración al ser las aplicaciones independientes del sistema operativo empleado.

El nivel o capa de middleware añade funcionalidades adicionales encima de la capa de drivers, como pueden ser la comunicación a nivel de enlace físico por Wi-Fi o BLE (Bluetooth Low Energy), librerías gráficas, protocolos de comunicación (TCP, UDP, etc.) y protocolos de envíos de mensajes (HTTP, MQTT, etc.) entre otros.

La capa más alta antes de la aplicación de usuario es la capa de SDK plug-ins o complementos del kit de desarrollo software que nos sirve en caso de necesitar hacer la integración del SDK con componentes externos y sigue la filosofía del software modular. Algunos ejemplos de esta capa es dar soporte a sensores, actuadores, pantallas o conexión con la nube para controlar determinados dispositivos (Apple HomeKit).

Un sistema operativo de tiempo real o RTOS (Real-time Operating System) surge ante la necesidad de crear una aplicación que procese o sirva los datos tan pronto como estén disponibles, normalmente sin acumular retrasos. Los aspectos clave de este sistema operativo son conseguir una baja latencia en las interrupciones y durante el cambio de contexto de un hilo de ejecución a otro. Su aplicación se adecua más a una necesidad de cumplimiento de tiempos (rapidez y determinación) que a la cantidad de procesamiento que realiza en un tiempo determinado, por lo que es frecuente encontrarlos en dispositivos empotrados que tienen que hacer algo más que un par de acciones. Este sistema operativo permite escalar a medida que se añaden características de la aplicación o del sistema.

Las características de un sistema operativo de tiempo real se pueden resumir de la siguiente manera:

- Baja latencia: relativo al cambio de contexto entre hilos de ejecución y las interrupciones.

- Determinismo: el código de la aplicación debe ser siempre determinista y calcular si se cumplen las restricciones de tiempos en base al tiempo de procesamiento de cada hilo de ejecución (precisa de un estudio previo).

- Software estructurado: ofrece facilidad a la hora de añadir componentes adicionales a la aplicación.

- Escalabilidad: debe ser capaz de escalar desde una simple aplicación hasta una más elaborada que incluye pilas de protocolos, drivers, sistema de ficheros entre otros.

- Desarrollo simplificado: permite al desarrollador centrarse en su aplicación y no tener que lidiar con la planificación de procesos, la administración de energía, la administración de la tabla de interrupciones, la administración de memoria o el manejo de las excepciones entre otros.

Los componentes principales de un sistema operativo de tiempo real y que incluye el SDK de SimpleLink son los siguientes:

- Planificador de procesos: los hilos de ejecución se planifican usando el método de planificación preferente en el que se asegura que la tarea con la máxima prioridad está en ejecución. Mediante este método de planificación un proceso se sigue ejecutando hasta bien acaba, una tarea con mayor prioridad está preparada o el mismo proceso abandona el procesador esperando algún recurso (ejemplo, sleep).

- Mecanismos de comunicación: comunicación entre hilos usando semáforos, colas de mensajes, colas entre otros.

- Mecanismos de regiones críticas: existen regiones de código que actualizan o leen variables compartidas por lo que el acceso al mismo código debe controlarse y hacerse de forma ordenada por parte de los hilos. Para ello se utilizan los mecanismos de mutex (mutual exclusion), gates o locks (cerrojos).

- Servicio de temporización: temporizadores, reloj del sistema entre otros.

- Administración de energía: aplicable a dispositivos de bajo consumo que permiten conmutar entre estados de energía.

- Administración de memoria: memoria en heap configurable con un tamaño estático o variable.

- Drivers de periféricos: UART, SPI, I2C, etc.

- Pilas de protocolos: BLE, Wi-Fi entre otros.

- Sistema de ficheros: FatFS, etc.

- Administración del dispositivo: manejo de excepciones, arranque, etc.

MQTT (Message Queue Telemetry Transport)

MQTT es un protocolo de transporte de mensajes cliente - servidor (basado en el modelo publicador - suscriptor) ideado por IBM de libre uso para la conectividad M2M (Machine-to-Machine). Se trata de un protocolo muy usado en la comunicación con sensores en el Internet de las Cosas y funciona sobre TCP/IP o sobre otros protocolos de red con soporte bidireccional y sin pérdidas de datos.

Las características principales de este protocolo son las siguientes:

- Es ligero, por lo que facilita su implementación por software y es rápido en la transmisión de datos.
- Basado en un protocolo de mensajería.
- Tamaño de paquetes reducido, por tanto, reduciendo el uso de la red.
- Bajo consumo de energía en los dispositivos en los que se implementa.
- Es en tiempo real, por tanto, se asegura una cota máxima de tiempo de entrega de los mensajes.
- Cuenta con un mecanismo de notificación de desconexiones inesperadas.

- Se definen varias QoS (calidades de servicio).

La arquitectura de este protocolo sigue una topología en estrella en la que el nodo central o broker puede trabajar con un gran número de clientes:

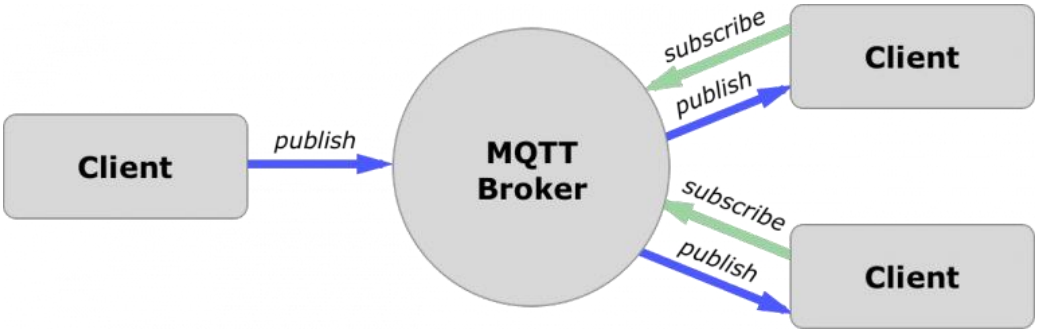

Ilustración 16: Arquitectura del protocolo MQTT (hivemq.com)

El broker MQTT es el nombre que recibe el servidor que gestiona la red y transmite los datos entre los clientes. Los datos que almacena son de tipo bytes variables y se almacenarán en el mismo junto a un topic o tema de referencia.

Los clientes pueden ser generadores o consumidores de datos (o ambos a la vez). En el caso de generar datos se van a enviar al broker mediante el proceso de publicación. De lo contrario, si un cliente es consumidor de un dato determinado y quiere recibir un mensaje del broker, tendrá que ejecutar el proceso de suscripción.

En la práctica, cada vez que un cliente publica un tema determinado, el contenido de dicho mensaje se actualiza en el broker y seguidamente los clientes suscritos a dicho tema van a recibir el mensaje actualizado con el nuevo contenido.

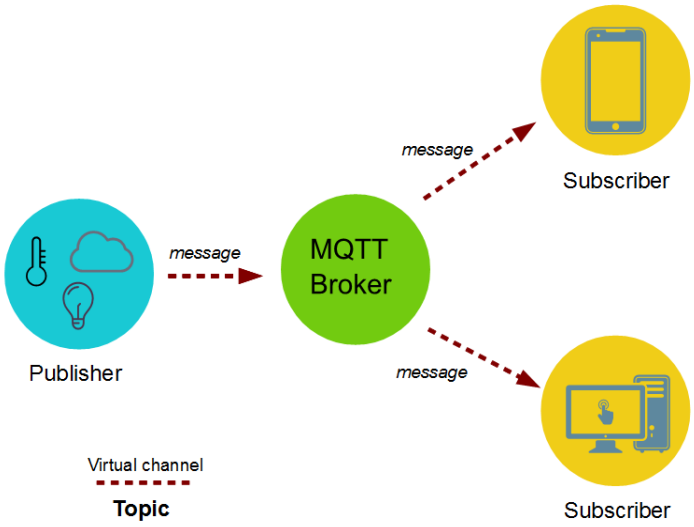

Ilustración 17: Ejemplo de aplicación MQTT (hivemq.com)

La comunicación entre los clientes puede ser de uno a uno o de uno a muchos. Este protocolo también permite establecer comunicaciones cifradas.

Los topics o temas de este protocolo siguen una estructura jerárquica que permite establecer relaciones padre - hijo por lo que al suscribirse a un topic padre, la información que se recibirá es la de todos los hijos.

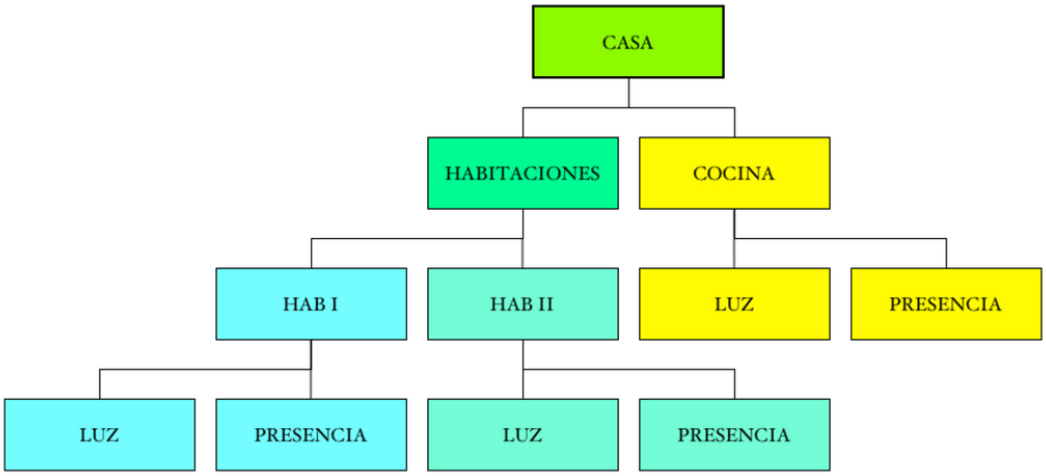

Ilustración 18: Estructura en árbol de los topics (ricveal.com)

En el ejemplo adjunto, suscribirse al topic Habitaciones (padre) devolverá toda la información de luz y presencia (hojas del árbol) que cuelgan bajo los hijos Hab I y Hab II.

Se definen varias calidades de servicio (QoS) en la entrega de mensajes en el protocolo MQTT, según las necesidades de la aplicación:

- "Como mucho una vez": funcionamiento parecido al protocolo TCP con pérdida de mensajes. El mensaje publicado se enviará a los clientes una o ninguna vez. Se usa en aplicaciones de sensorizado en las que la pérdida de un dato no es crítica.

- "Al menos una vez": el mensaje publicado se enviará una o más veces. Se asegura la entrega de los mensajes, pero deben considerarse las duplicidades que puedan llegar en el cliente.

- "Exactamente una vez": el mensaje publicado llegará a los clientes suscritos exactamente una vez. Un ejemplo de uso puede ser un sistema de pago en el que no se admite la pérdida ni la duplicidad de mensajes.

El broker que se usará en este trabajo es el broker Eclipse Mosquitto, de código abierto (licencias EPL/EDL) que implementa el protocolo MQTT versión 3.1. Este broker es ligero y se puede usar en todos los dispositivos, desde los microcontroladores o sensores de bajo consumo hasta servidores.

Ilustración 19: Logotipo de mosquitto (Eclipse mosquitto)

Java Client Paho

Java Paho Eclipse es una librería MQTT de cliente escrita en lenguaje de programación Java para el desarrollo de aplicaciones que corran en una máquina virtual de Java (JVM) u otras plataformas compatibles con java como puede ser Android.

Esta librería ofrece dos APIs, MqttAsyncClient (API asíncrono en el que los eventos se notifican mediante callbacks o llamadas a funciones registradas, independientes de la aplicación principal) y MqttClient (API síncrono donde las funciones son síncronas a la aplicación principal).

El proyecto Paho se ha creado para ofrecer implementaciones de código abierto fiables de los protocolos de mensajería estándares para nuevas aplicaciones en el Internet de las Cosas (IoT) o máquina a máquina (M2M). Entre sus objetivos está ofrecer niveles de desacoplamiento entre las aplicaciones y los dispositivos, y mantener un mercado abierto para fomentar el rápido crecimiento de las aplicaciones y middleware de forma escalable.

Las principales características del cliente Java Paho de Eclipse son:

- Especificaciones MQTT versión 3.1
- Especificaciones MQTT versión 3.1.1
- LWT (Last Will and Testament): mensajes para informar en caso de desconexión o mal funcionamiento de los nodos.
- Seguridad en la transmisión SSL/TLS.
- Persistencia de mensajes: permite la persistencia de los mensajes en caso de fallo de la aplicación.
- Reconexión automática: puede reconectarse automáticamente al broker servidor en caso de pérdida de conexión.
- Encola los mensajes si está desconectado para enviar cuando se reconecta.
- Soporte con servidores MQTT que usen WebSockets.
- Soporte TCP estándar.

- API bloqueante y no bloqueante.
- Alta disponibilidad: ofrece la posibilidad de conectarse a un broker servidor alternativo en caso de fallo de conexión al circuito principal.

Node - RED Dashboard

Node - RED es una herramienta de programación que facilita la interconexión de dispositivos hardware, APIs y servicios online.

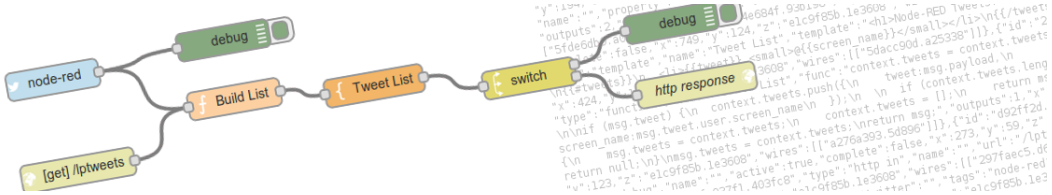

Ilustración 20: Flujo de Node – RED (flows Node - RED)

Su funcionamiento se basa en el concepto de programación basada en flujo (J. Paul Morrison, 1970) en donde el funcionamiento de la aplicación se describe mediante una red de nodos (cajas negras) interconectadas. Cada nodo tiene su objetivo bien definido, a menudo recibir un dato, procesarlo y pasarlo al siguiente nodo en la red. Este modelo de programación ofrece accesibilidad a un mayor rango de usuarios y su representación visual facilita la comprensión del funcionamiento de la aplicación.

Sus características más destacables son:

- Editor de flujo integrado en el navegador: permite la interconexión de distintos tipos de nodos, crear funciones JavaScript con el editor de código incorporado, la reutilización de funciones, plantillas o flujos y la posibilidad de desplegar la aplicación con solo un click.

- Motor construído sobre Node.js: motor en tiempo de ejecución construído sobre Node.js lo que ofrece un modelo dirigido por eventos no bloqueantes o asíncrono, haciéndolo ideal para su uso en servidores y procesamiento de borde. [13]

- Desarrollo social: los flujos de las aplicaciones creadas se almacenan en JSON lo que facilita la exportación e importación para compartir con la comunidad de desarrolladores.

En lo relacionado a este trabajo, se usará Node - RED junto al módulo de node-red-dashboard que nos ofrecerá una serie de nodos para una rápida construcción de un dashboard de control y visualización de datos para el usuario.

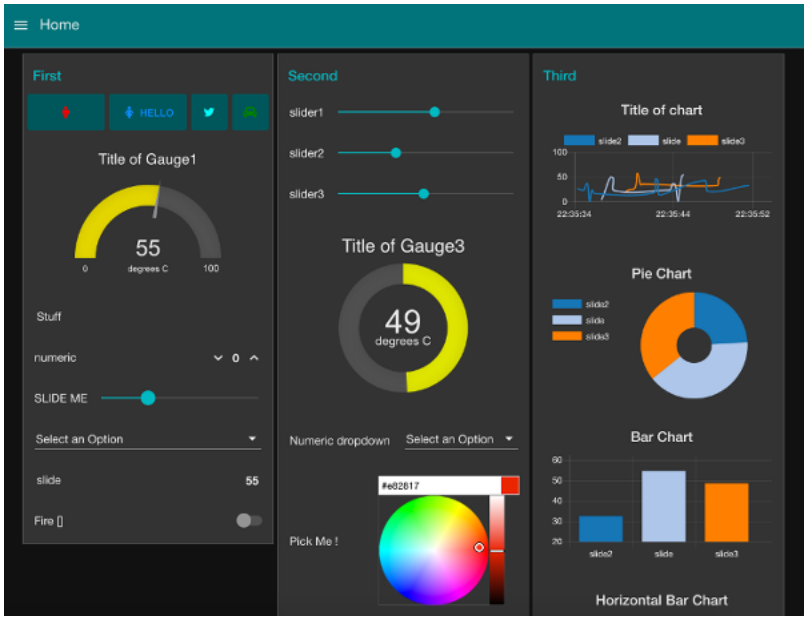

Ilustración 21: Interfaz de visualización y control (Node – RED Dashboard app)

Los nodos adicionales que proporciona este módulo nos permitirán mostrar series de datos en un gráfico en tiempo real, configurar botones, formularios o conectar con protocolos de comunicación externos para el envío y obtención de datos.

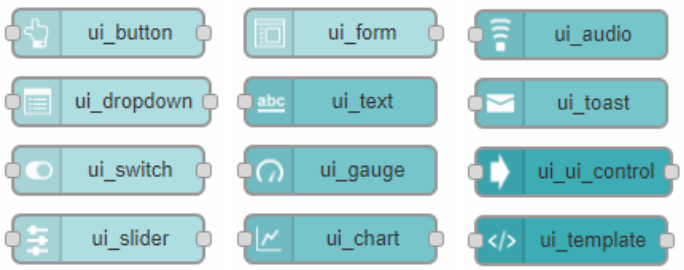

Ilustración 22: Nodos para UI (User Interface) (Node – RED Dashboard app)

4.2 Hardware elegido para la implementación

En la arquitectura anterior cada uno de los módulos hardware es el encargado de una función determinada y gracias a la interconexión y comunicación de datos de un módulo a otro es posible hacer un sistema global que responda al comportamiento que se quiere conseguir.

El módulo hardware de una sección podrá intercambiar datos con los dispositivos externos tales como sensores y actuadores, por lo que podremos recibir los eventos de tránsito de vehículos, temperatura, nivel de luminosidad exterior o contar vehículos dentro de la misma sección. Estos datos se envían de forma periódica al servidor donde se realiza el análisis y generación del nivel de luminosidad que se tiene que aplicar en la sección. Esta nueva información se envía de vuelta a la sección para su posterior uso.

El Gateway es un elemento que sirve de interconexión de los elementos principales de la arquitectura (servidor y nodos de las secciones) para permitir el uso de un protocolo de comunicaciones entre los mismos.

El servidor tiene una doble funcionalidad. La primera es recoger los datos de los dispositivos y almacenarlos temporalmente para su análisis y determinación del nivel de intensidad de luz que se debe imponer en cada sección. La segunda es permitir la visualización de la información en gráficos en tiempo real para la toma de decisiones y el control manual.

Con el objetivo de facilitar la implementación y prueba de arquitecturas en el Internet de las Cosas a menudo se opta por usar kits de desarrollo hardware que incluyen muchas de las características necesarias como pueden ser el procesamiento, la comunicación con periféricos o la comunicación con otros módulos similares. En nuestro caso y con la misma finalidad, se ha elegido usar el kit de desarrollo hardware CC3220 SimpleLink Wi-Fi LaunchPad (CC3220SF-LAUNCHXL) de Texas Instruments. Esta placa cuenta con una MCU (ARM Cortex-M4) que integra conectividad Wi-Fi en el mismo chip, así como protocolos de seguridad robustos.

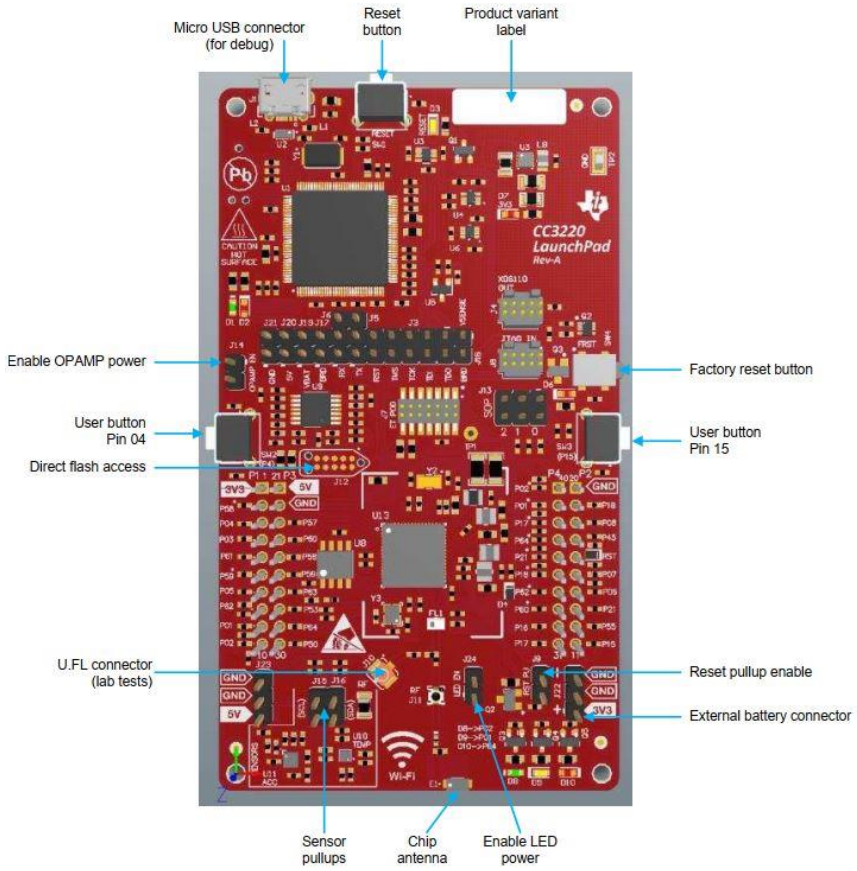

Ilustración 23: Vista previa del LaunchPad CC3220 (manual placa CC3220SF)

El diseño de esta placa integra varios dispositivos como pueden ser el sensor de temperatura o el acelerómetro, botones y LEDs programables y emulación en la misma placa para la depuración. Además, es posible ampliar las características de la misma placa conectando complementos para conseguir pantallas gráficas, códecs de audio, antenas, sensorizacion adicional del medio ambiente entre otros. A continuación, se va a analizar el diagrama de bloques para profundizar en las características del LaunchPad.

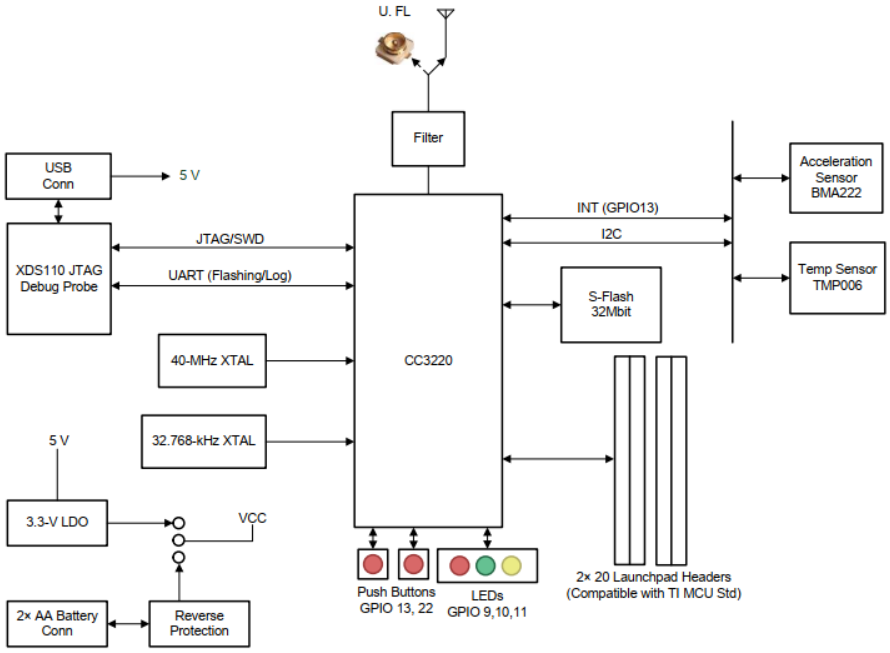

Ilustración 24: Diagrama de bloques del LaunchPad CC3220 (manual placa CC3220SF)

Las características hardware de la placa se resumen en los siguientes puntos:

- CC3220 MCU con conexión Wi-Fi integrada
- Ampliable con el BoosterPack estándar de 40 pines
- Emulación JTAG con el estándar TI XDS110
- Soporta el JTAG de 4 pines y el SWD de 2 pines
- Dos botones y tres LEDs para interacción con el usuario
- Conexión con el PC mediante UART virtual COM
- Antena integrada con opción para pruebas U.FL con resistencias de 0 ohm
- Sensores acelerómetro y de temperatura integrados y aislables del bus I2C
- Conector micro USB para alimentación y conexiones de depuración
- Cabeceras para medición de corriente y conexión externa JTAG con opción de usar el XDS110 integrado para debugging de plataformas del cliente
- Dispositivos alimentados por bus, no se necesita alimentación externa para Wi-Fi
- Transmisión de gran alcance con una antena optimizada (200 metros en espacio abierto con una antena AP de 6-dBi)
- Se puede alimentar por vía externa trabajando a un valor típico de 2.3V

En cuanto al SoC CC3220 integrado en el LaunchPad se puede diferenciar varias regiones como la memoria estática donde se sitúa el programa a ejecutar en el arranque del sistema, la memoria volátil, hardware del sistema (DMA, Timers, GPIOs, Oscillators), interfaces de comunicación con dispositivos externos o elementos relacionados con la gestión de la energía.

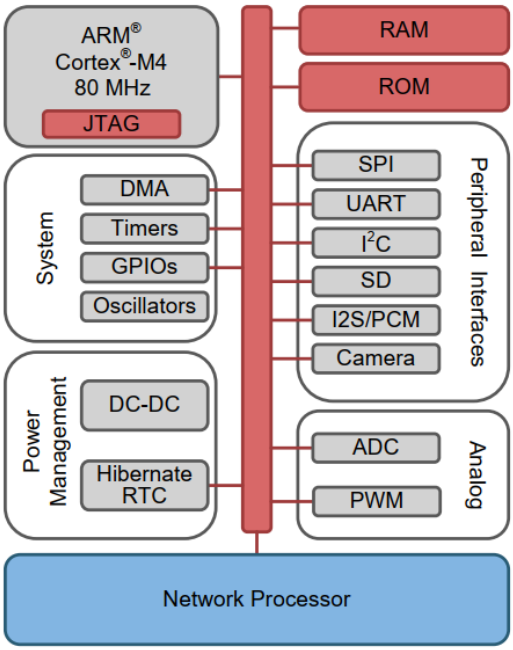

Ilustración 25: Vista previa del System-on Chip CC3220 (manual placa CC3220SF)

Las características más sobresalientes de este SoC son:

- Es un chip que cuenta con MCU para aplicaciones del usuario (ARM Cortex-M4) y un MCU para el procesador de red para comunicar por Wi-Fi y contar con todas las capas lógicas de Internet.

- El subsistema para aplicaciones del usuario cuenta con un núcleo ARM Cortex-M4 a 80MHz, memoria integrada basada en flash de 1MB de almacenamiento, 256KB de memoria RAM (Random Access Memory), posibilidad de conectar una tarjeta SD, SPI,

I2C, UART, cámara de video de 8 bits, temporizadores (Watchdog y externos por PWM), ADCs, multitud de pines GPIO e interfaces de depuración JTAG, cJTAG o SWD.

- El subsistema NWP (Wi-Fi 2.4-Ghz Network Processor) es un MCU dedicado para comunicaciones con el objetivo de quitar carga de trabajo al subsistema que ejecuta aplicaciones del usuario. Los modos de comunicación compatibles del 802.11 b/g/n son el modo Station, AP (Access Point, hasta cuatro estaciones) y Wi-Fi Direct (Client y Group Owner). Admite seguridad en la comunicación con el punto de acceso basada en WPA2 Personal y Enterprise Security (WEP, WPA/WPA2 PSK, WPA2 Enterprise 802.1.x) y sockets de comunicación seguros como SSLv3 o TLS1.2. Cuenta con las pilas de protocolos de IPv4 y IPv6 y admite un máximo de 16 sockets TCP o UDP, y 6 sockets TLS y SSL simultáneos. La versión segura de este dispositivo ofrece un motor de cifrado por hardware que incluye algoritmos como AES, DES, 3DES, SHA2, MD5 o CRC.

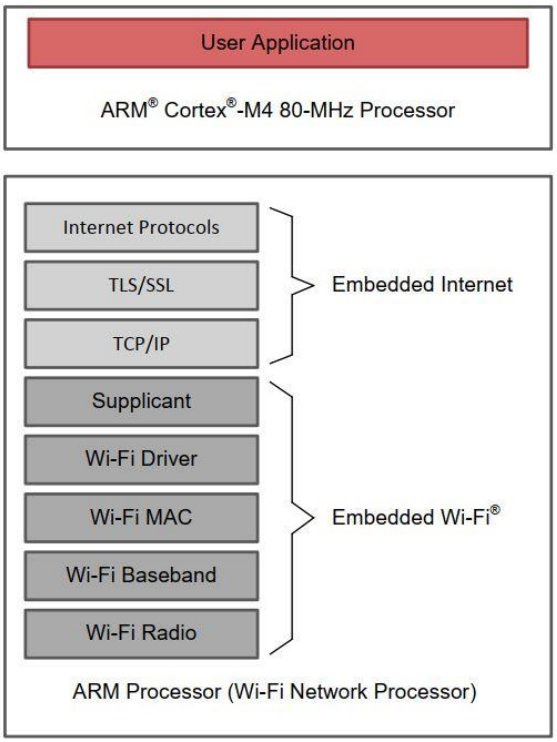

Ilustración 26: Software integrado para Wi-Fi e Internet (manual placa CC3220SF)

- El subsistema de administración de energía ofrece alimentación en un amplio rango de voltajes (por batería entre 2.1V y 3.6V) y modos avanzados de baja energía como la hibernación, el apagado o LPDS (Low-Power Deep Sleep).

- Varias fuentes de reloj con frecuencias de 40MHz y 32.7kHz y rango de temperaturas de funcionamiento entre -40 y +85 grados centígrados.

La placa de desarrollo presentada será la que en nuestra arquitectura nos va a proporcionar procesamiento y comunicación con el exterior.

Para recoger información del entorno y actuar sobre el mismo en un entorno de simulación en el laboratorio vamos a necesitar dispositivos sensores y actuadores, los circuitos empleados para dicha finalidad se explican a continuación:

- Sensor de intensidad lumínica: con el objetivo de recoger información relativa al nivel de luminosidad exterior vamos a emplear un circuito con un LDR en división de tensión para ser capaces de leer el valor analógico con el ADC.

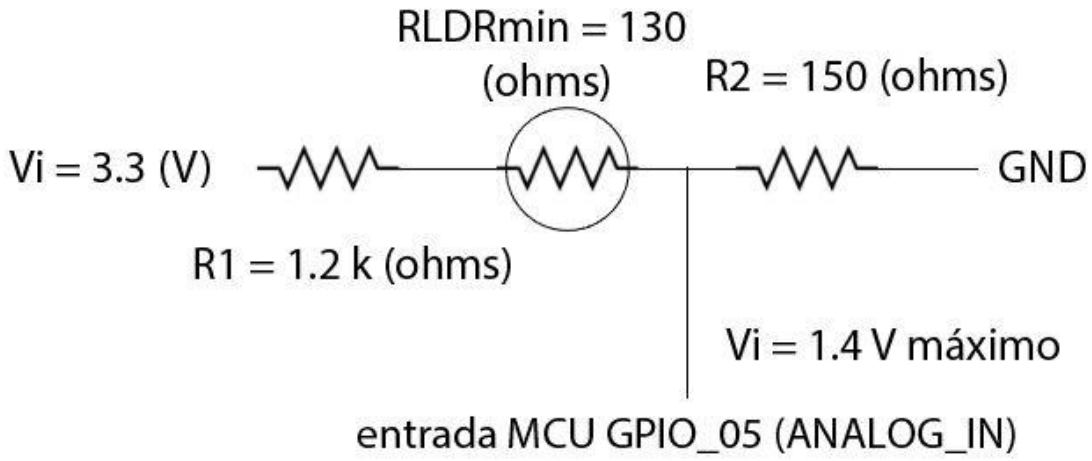

Ilustración 27: Circuito externo LDR (elaboración propia)

Los valores de voltaje variables generados en este circuito serán luego recogidos por el conversor analógico-digital de la placa de desarrollo para ser convertidos en valores digitales (12 bits de precisión) que puedan ser procesados. Su circuito interno es el siguiente:

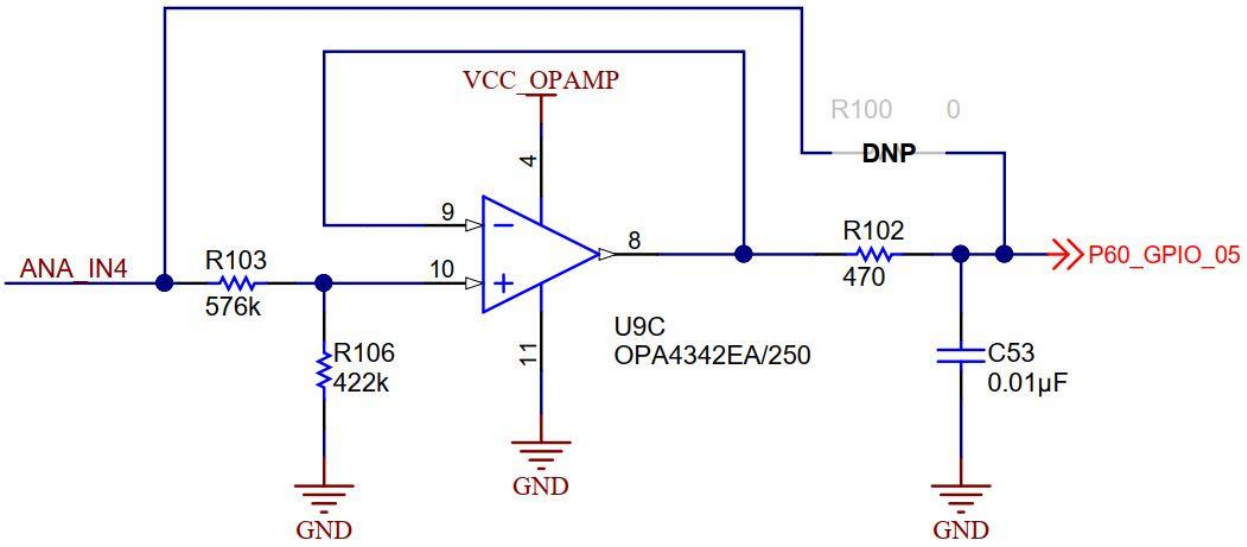

Ilustración 28: Circuito ADC del CC3220SF (manual placa CC3220SF)

- Sensor de detección de coche: con el objetivo de simular el evento de paso de coche se ha optado por usar un sensor de detección de movimiento instalado en la entrada de la sección. Este dispositivo emite un flanco de subida al detectar movimiento el cual será recogido por los pines GPIO digitales de la MCU para recoger el evento de forma asíncrona con interrupciones.

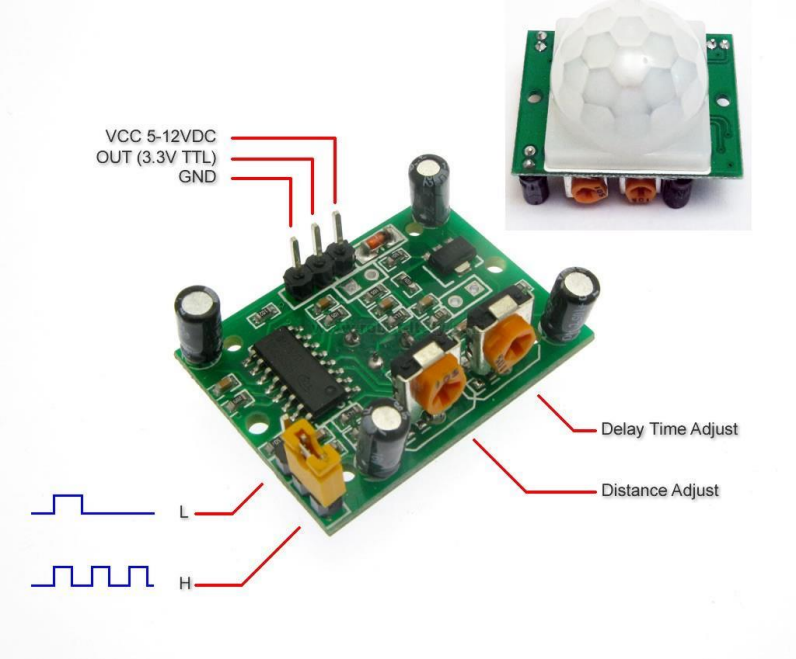

Ilustración 29: Sensor de movimiento HC-SR501 (hobbykitsindia.in)

Con el objetivo de reducir el voltaje de salida generado por este dispositivo a los rangos adecuados para la entrada GPIO del MCU se usa el siguiente circuito de división de tensión:

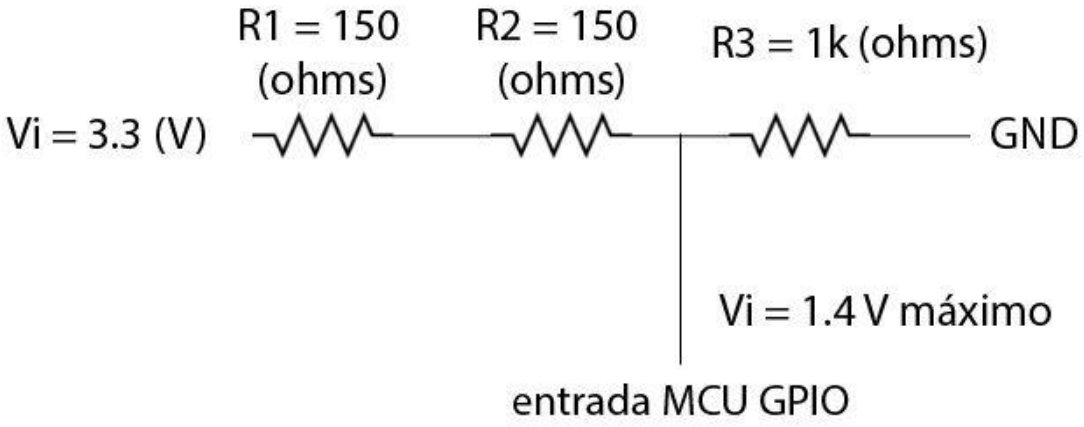

Ilustración 30: Circuito divisor de tensión para la entrada al ADC (elaboración propia)

- Sensor de temperatura: para este trabajo el sensor de temperatura se usa como medio para obtener información adicional del entorno y mandarlo al servidor de forma periódica. Para ello se ha optado por usar el sensor de temperatura TMP006 integrado de la placa de desarrollo. Dicho sensor está conectado al bus I2C de la placa por lo que se usará el controlador y funciones ya existentes en los proyectos de Texas Instruments para leer valores del mismo dispositivo.
- Actuador regulador de intensidad lumínica de la luminaria: para simular las luminarias de una sección determinada se usará dos LEDs que nos darán información sobre la intensidad de la iluminación (0%, 50% o 100% de intensidad).

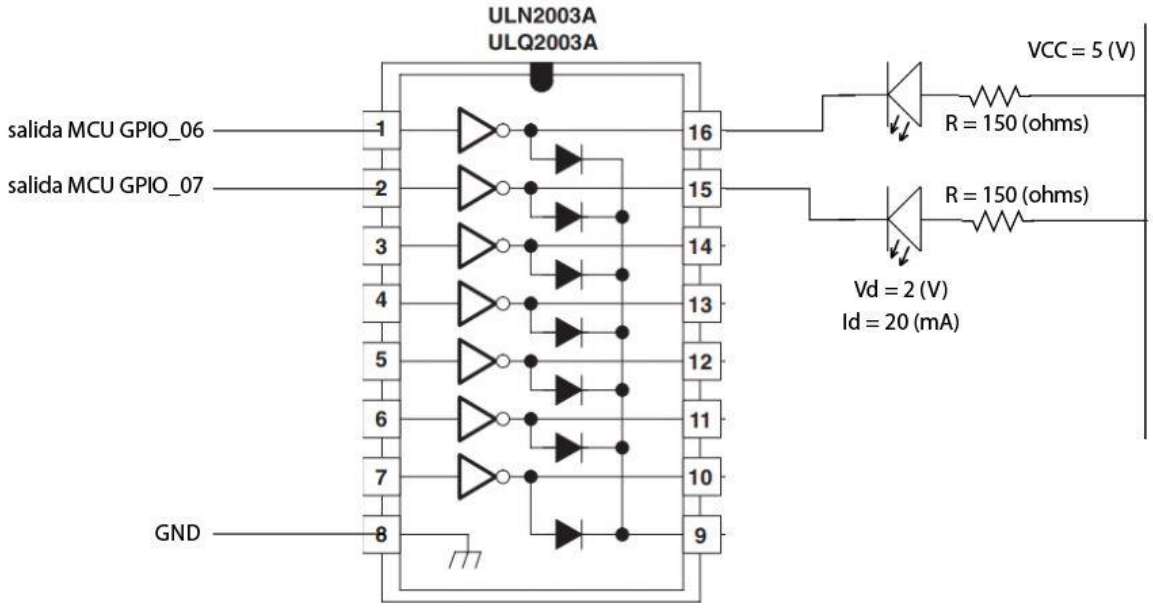

Ilustración 31: Circuito externo luminaria (elaboración propia)

Con el objetivo de interconectar los dispositivos de la arquitectura y habilitar una red de comunicaciones se usará el dispositivo Asus RT-AC1200G+ como Gateway de la arquitectura.

Ilustración 32: Gateway usado en la arquitectura Asus RT-AC1200G+ (uk.hardware.info)

Las placas de desarrollo (CC3220SF) se van a conectar de forma inalámbrica (Wifi – mediante el uso de nombre y contraseña del AP) con el Gateway a la red y en la fase de negociación de la conexión van a recibir los parámetros de conexión a la misma red mediante DHCP (Dynamic Host Configuration Protocol). A partir de ese momento los elementos de la arquitectura están conectados a la red y listos para mantener la comunicación. La conexión Wifi al punto de acceso tiene características de frecuencia de 2.4Ghz, método de autenticación WPA2-Personal y cifrado de datos con el algoritmo AES.

4.3 Sistema operativo

Para usar la placa de desarrollo SimpleLink Wi-Fi Launchpad CC3220 es necesario hacer uso de SimpleLink SDK o kit de desarrollo software que integra los componentes que se pueden ver en la figura adjunta más abajo y que facilita la creación de aplicaciones sin preocuparse por las pilas de protocolos de comunicaciones o drivers entre otros.

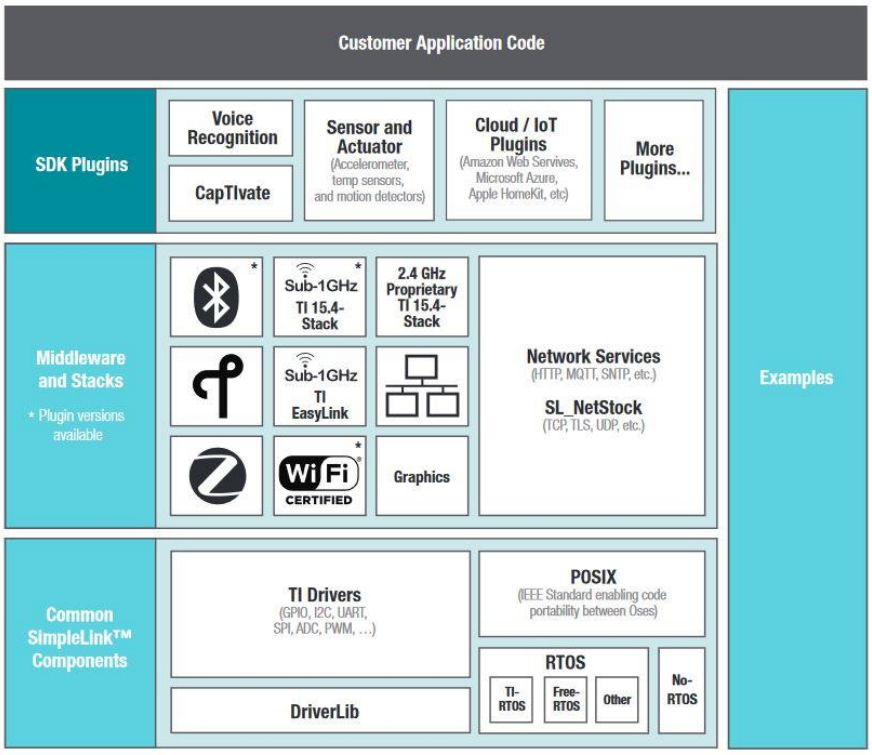

Ilustración 33: Componentes del SDK de SimpleLink (manual placa CC3220SF)

Los **drivers de TI** ofrecen al programador una interfaz de comunicación con los dispositivos periféricos de la placa construida sobre una capa de abstracción hardware que facilita la portabilidad del código de la aplicación entre dispositivos y el acceso completo a las características de los periféricos. Se trata de unos drivers de código abierto bajo la licencia BSD (Berkeley Software Distribution). Esta capa también permite a los desarrolladores hacer una programación a nivel de registros para facilitar la adaptación de la aplicación a sus necesidades

concretas. Algunos dispositivos periféricos que pueden ser utilizados mediante estos drivers son el puerto UART (Universal Asynchronous Receiver-Transmitter), el convertidor analógico digital ADC (Analog-to-digital converter), SPI (Serial Peripheral Interface Bus), PWM (Pulse-width modulation) y GPIO (General-purpose input/output) entre muchos otros.

En el mismo nivel nos encontramos con la necesidad de contar con un **OS (Operating System) o kernel** que ofrezca servicios multitarea y en tiempo real como puede ser la temporización o la planificación de tareas. Este SDK lleva preinstalado el kernel de TI-RTOS (Real Time Operating System de Texas Instruments), que cumple con el estándar POSIX (Portable Operating System Interface), fue creado para ofrecer una baja latencia. Los desarrolladores pueden optimizar sus aplicaciones para el bajo consumo, rendimiento y el tamaño de código que necesitan. Una alternativa al kernel anterior es FreeRTOS. El API (Application Programming Interface) de POSIX es un estándar IEEE en la industria para la compatibilidad entre sistemas operativos. Esto permite a las aplicaciones ser independientes del sistema operativo de tiempo real utilizado y facilitar la migración en caso de ser necesario. Como se puede ver en la figura adjunta más abajo, cuando se crea un hilo o un semáforo pthread la capa de adaptación se encarga de gestionar la creación de forma independiente al sistema operativo de tiempo real utilizado.

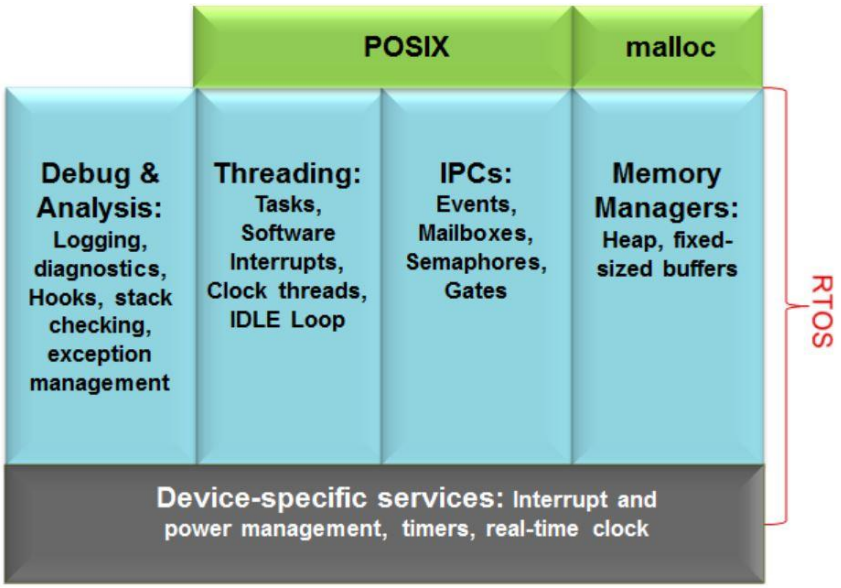

Ilustración 34: Capa de compatibilidad POSIX (foros de Texas Instruments sobre TI - RTOS)

El uso de esta capa es opcional, pero permite al usuario usar cualquier sistema operativo con el que está familiarizado o quiera usar en el futuro. Se va a profundizar en los aspectos generales de un sistema operativo de tiempo real al final de este apartado.

El nivel o capa de **middleware** añade funcionalidades adicionales encima de la capa de drivers, como pueden ser la comunicación a nivel de enlace físico por Wi-Fi o BLE (Bluetooth Low Energy), librerías gráficas, protocolos de comunicación (TCP, UDP, etc.) y protocolos de envíos de mensajes (HTTP, MQTT, etc.) entre otros.

La capa más alta antes de la aplicación de usuario es la capa de SDK plug-ins o complementos del kit de desarrollo software que nos sirve en caso de necesitar hacer la integración del SDK con componentes externos y sigue la filosofía del software modular. Algunos ejemplos de esta capa es dar soporte a sensores, actuadores, pantallas o conexión con la nube para controlar determinados dispositivos (Apple HomeKit).

Un **sistema operativo de tiempo real** o RTOS (Real-time Operating System) surge ante la necesidad de crear una aplicación que procese o sirva los datos tan pronto como estén disponibles, normalmente sin acumular retrasos. Los aspectos clave de este sistema operativo son conseguir una baja latencia en las interrupciones y durante el cambio de contexto de un hilo de ejecución a otro. Su aplicación se adecua más a una necesidad de cumplimiento de tiempos (rapidez y determinación) que a la cantidad de procesamiento que realiza en un tiempo determinado, por lo que es frecuente encontrarlos en dispositivos empotrados que tienen que hacer algo más que un par de acciones. Este sistema operativo permite escalar a medida que se añaden características de la aplicación o del sistema.

Las características de un sistema operativo de tiempo real se pueden resumir de la siguiente manera:

- Baja latencia: relativo al cambio de contexto entre hilos de ejecución y las interrupciones.

- Determinismo: el código de la aplicación debe ser siempre determinista y calcular si se cumplen las restricciones de tiempos en base al tiempo de procesamiento de cada hilo de ejecución (precisa de un estudio previo).
- Software estructurado: ofrece facilidad a la hora de añadir componentes adicionales a la aplicación.
- Escalabilidad: debe ser capaz de escalar desde una simple aplicación hasta una más elaborada que incluye pilas de protocolos, drivers, sistema de ficheros entre otros.
- Desarrollo simplificado: permite al desarrollador centrarse en su aplicación y no tener que lidiar con la planificación de procesos, la administración de energía, la administración de la tabla de interrupciones, la administración de memoria o el manejo de las excepciones entre otros.

Las componentes principales de un sistema de tiempo real y que incluye el SDK de SimpleLink tanto en los kernel TI-RTOS y FreeRTOS son los siguientes:

- Planificador de procesos: los hilos de ejecución se planifican usando el método de planificación preferente en el que se asegura que la tarea con la máxima prioridad está en ejecución. Mediante este método de planificación un proceso se sigue ejecutando hasta bien acaba, una tarea con mayor prioridad está preparada o el mismo proceso abandona el procesador esperando algún recurso (ejemplo, sleep).

- Mecanismos de comunicación: comunicación entre hilos usando semáforos, colas de mensajes, colas entre otros.

- Mecanismos de regiones críticas: existen regiones de código que actualizan o leen variables compartidas por lo que el acceso al mismo código debe controlarse y hacerse de forma ordenada por parte de los hilos. Para ello se utilizan los mecanismos de mutex (mutual exclusion), gates o locks (cerrojos).

- Servicio de temporización: temporizadores, reloj del sistema entre otros.

- Administración de energía: aplicable a dispositivos de bajo consumo que permiten conmutar entre estados de energía.

- Administración de memoria: memoria en heap configurable con un tamaño estático o variable.

- Drivers de periféricos: UART, SPI, I2C, etc.

- Pilas de protocolos: BLE, Wi-Fi entre otros.

- Sistema de ficheros: FatFS, etc.

- Administración del dispositivo: manejo de excepciones, arranque, etc.

4.4 Comunicación con los nodos y envío de mensajes

Con el objetivo de poner en práctica un sistema con el funcionamiento deseado se propone el uso de una arquitectura centralizada en la que los nodos de cada sección se van a comunicar enviando eventos e información y recogiendo datos de funcionamiento de un servidor en el que tendrá lugar la lógica de funcionamiento. El servidor, una vez calculado un nuevo estado de la carretera, va a comunicar a los nodos necesarios la nueva información de funcionamiento.

En el envío de información en el sentido nodo – servidor y servidor – nodo se precisa de una arquitectura de comunicaciones bien establecida. Para ilustrar de una manera visual los niveles de comunicación con sus protocolos se propone el siguiente esquema de pila de protocolos de comunicaciones:

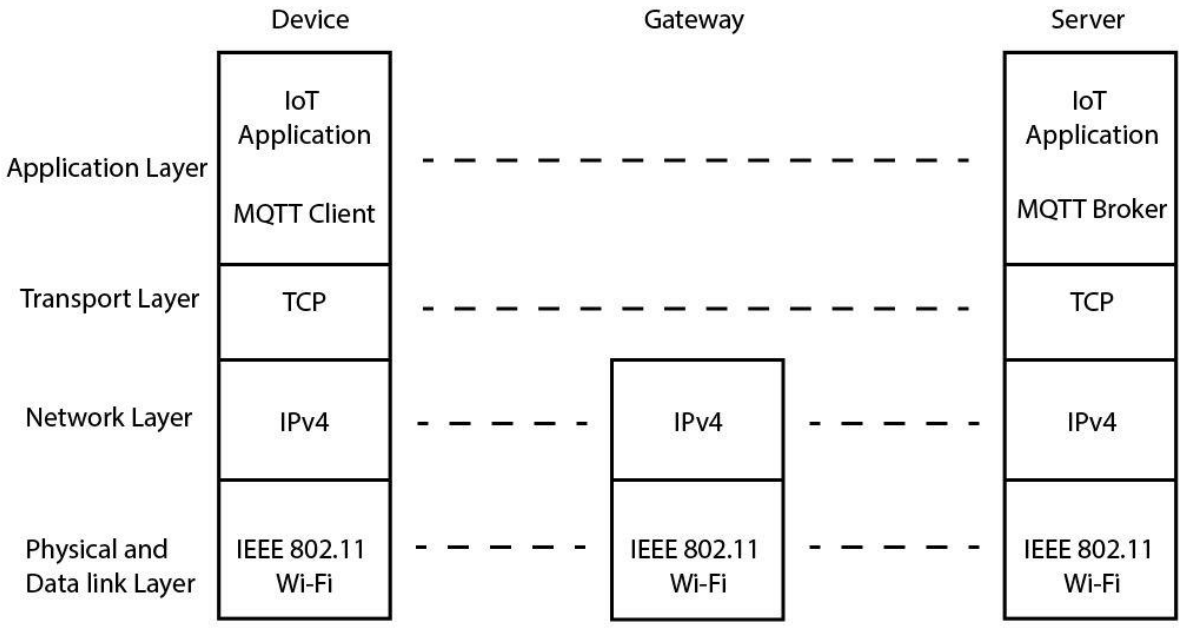

Ilustración 35: Pila de protocolos de comunicaciones (elaboración propia)

En la parte superior de la pila de protocolos se encuentra la aplicación tanto del lado de los nodos como del servidor que va a dar servicio al funcionamiento del sistema, que usarán varios recursos en red para poder comunicarse e intercambiar información con otros dispositivos.

En la capa inmediatamente inferior contamos con librerías de cliente y servidor para el intercambio de mensajes MQTT. Este protocolo nos permite enviar mensajes (de tipo bytes variables) con la estructura en árbol de los topics y mantener actualizados el estado más reciente en el broker MQTT en el lado del Servidor, donde una o varias aplicaciones pueden hacer uso de la información. Este protocolo de envío de mensajes utiliza en la capa de transporte el protocolo de comunicaciones fiable TCP y permite el uso de varias calidades de servicio (en este caso se ha establecido un QoS de un único envío de mensaje).

En la capa de red se ha establecido el encapsulado IPv4 y en la capa física el estándar IEEE 802.11 Wi-Fi inalámbrico.

Los mensajes intercambiados en el protocolo MQTT siguen una estructura jerárquica organizados por topics, lo que nos permite elaborar una estructura de topics bajo cada nodo, de tal manera que se pueda recoger un dato en concreto o todos los datos de un dispositivo según se necesite. La estructura de topics que sigue la aplicación es la siguiente:

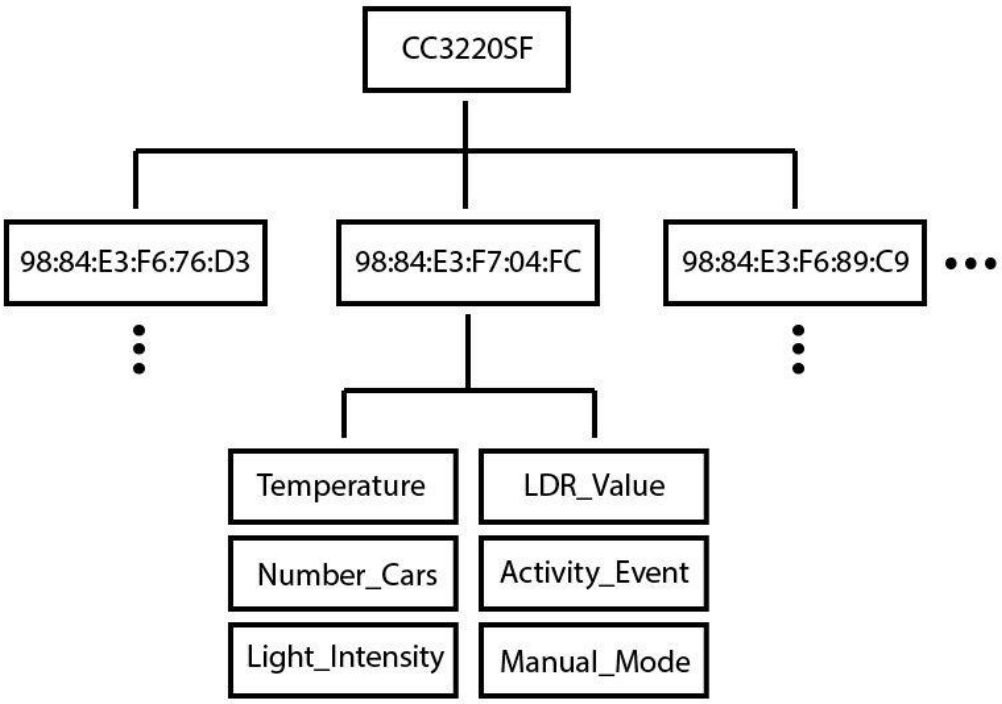

Ilustración 36: Estructura de los topics (elaboración propia)

4.5 Lógica de funcionamiento

Con el objetivo de que el sistema realice el funcionamiento deseado, cada uno de los módulos que lo componen deben realizar un papel determinado. Los módulos que van a realizar la lógica de funcionamiento son los nodos CC3220SF (módulo único) y el Servidor (módulo Paho Java y Node – RED Dashboard) que ya cuentan con una red de comunicaciones establecida.

En este apartado se van a analizar en detalle cada uno de los módulos que realizan lógica de funcionamiento en el sistema.

Los **nodos CC3220SF** cuentan con un programa único ejecutándose sobre un sistema operativo de tiempo real, con librerías de comunicación con dispositivos (GPIO, SPI o I2C) y de protocolos de mensajería como MQTT. El código fuente del programa que incluye el nodo CC3220SF con la lógica de funcionamiento se puede consultar en el enlace compartido de Google Drive [15]. El programa en cuestión es un cliente MQTT que cuenta con la posibilidad de suscribirse y publicar mensajes:

- Publicación de mensajes: los nodos van a transmitir los datos relativos a la temperatura (topic "/CC3220SF/{MAC}/Temperature"), valor medido del LDR (topic "/CC3220SF/{MAC}/LDR_Value") y evento de actividad (topic "/CC3220SF/{MAC}/Activity_Event"). Para ello el nodo tiene configurados la comunicación con los dispositivos por I2C, ADC y GPIO respectivamente. En los dos primeros casos la obtención de nuevos datos de los dispositivos y la publicación de estos al Servidor (en el topic correspondiente) se hace de forma periódica (cada tres segundos).

```
for(;; )
{
    /*waiting for signals                                                */
    mq_receive(g_PBQueue, (char*) &queueElemRecv, sizeof(struct msgQueue),
               NULL);

    switch(queueElemRecv.event)
    {
    case SEND_LDR_VALUE:
        Get_ADC_Channel(Board_ADC1, &ldrConvertedValue, &ldrRawValue);
        snprintf(buffer, sizeof(buffer), "%hu", ldrConvertedValue);

        lRetVal =
        MQTTClient_publish(gMqttClient, (char*) ldrValuePubTopic, strlen(
                    (char*)ldrValuePubTopic),
                (char*)buffer,
                strlen((char*) buffer), MQTT_QOS_2 |
                ((RETAIN_ENABLE) ? MQTT_PUBLISH_RETAIN : 0));

        break;
```

```
while(gResetApplication == false)
{
    struct msgQueue queueElement;

    queueElement.event = SEND_TEMPERATURE;
    queueElement.msgPtr = NULL;

    /* write message indicating publish message
    if(MQTT_SendMsgToQueue(&queueElement))
    {
        UART_PRINT("\n\n\rQueue is full\n\n\r");
    }

    queueElement.event = SEND_LDR_VALUE;
    if(MQTT_SendMsgToQueue(&queueElement))
    {
        UART_PRINT("\n\n\rQueue is full\n\n\r");
    }
    sleep(3);
    //;
}
```

Ilustración 37: Publicación periódica de datos (código fuente del cliente CC3220SF)

El evento de actividad es un evento aperiódico que surge por interrupción instalada en el GPIO, por lo que en cuanto ocurra se publicará el mensaje al Servidor.

```
case ACTIVITY_EVENT:
    tmpBuff = (char *) ((char *) queueElemRecv.msgPtr);
    if(strncmp(tmpBuff, "ENTRY", queueElemRecv.topLen) == 0)
    {
        snprintf(buffer, sizeof(buffer), "%s", "ENTRY");
        lRetVal =
        MQTTClient_publish(gMqttClient, (char*) activityEventPubTopic, strlen(
                    (char*)activityEventPubTopic),
                (char*)buffer,
                strlen((char*) buffer), MQTT_QOS_2 |
                ((RETAIN_ENABLE) ? MQTT_PUBLISH_RETAIN : 0));

        GPIO_clearInt(Board_GPIO_BUTTON0);      // SW2
        GPIO_enableInt(Board_GPIO_BUTTON0);     // SW2

    }
    break;
```

```
void pushButtonInterruptHandler2(uint_least8_t index)
{
    struct msgQueue queueElement;
    //TODO aqui
    /* Disable the SW2 interrupt */
    GPIO_disableInt(Board_GPIO_BUTTON0); // SW2

    queueElement.event = ACTIVITY_EVENT;
    queueElement.msgPtr = "ENTRY";

    /* write message indicating publish message
    if(MQTT_SendMsgToQueue(&queueElement))
    {
        UART_PRINT("\n\n\rQueue is full\n\n\r");
    }
}
```

Ilustración 38: Publicación de datos por interrupción GPIO (código fuente del cliente CC3220SF)

En cuanto al valor de los datos, la temperatura será un valor en punto flotante que se espera a priori que esté en un rango de [-10.0, 40.0], el valor del LDR un entero en el rango [0, 1024] y el valor del evento de actividad será una secuencia de caracteres igual a “ENTRY” indicando evento de entrada de vehículo en la sección correspondiente.

- Suscripción a mensajes: los nodos van a estar suscritos al nivel de intensidad que deben imponer en la sección (topic “/CC3220SF/{MAC}/Light_Intensity”) y su valor se espera que sea un número entero de este conjunto {0, 50 o 100} indicando el porcentaje de intensidad de luz que se debe fijar a la sección. Este tipo de mensajes llegan de forma asíncrona, por lo que una vez llegue el mensaje, se comprueba el tipo y se recoge el valor de la intensidad para traducirlos al siguiente conjunto {00, 01, 11} indicando cuántos LEDs se van a iluminar en la sección y comunicándolo debidamente a éstos por los puertos GPIO correspondientes.

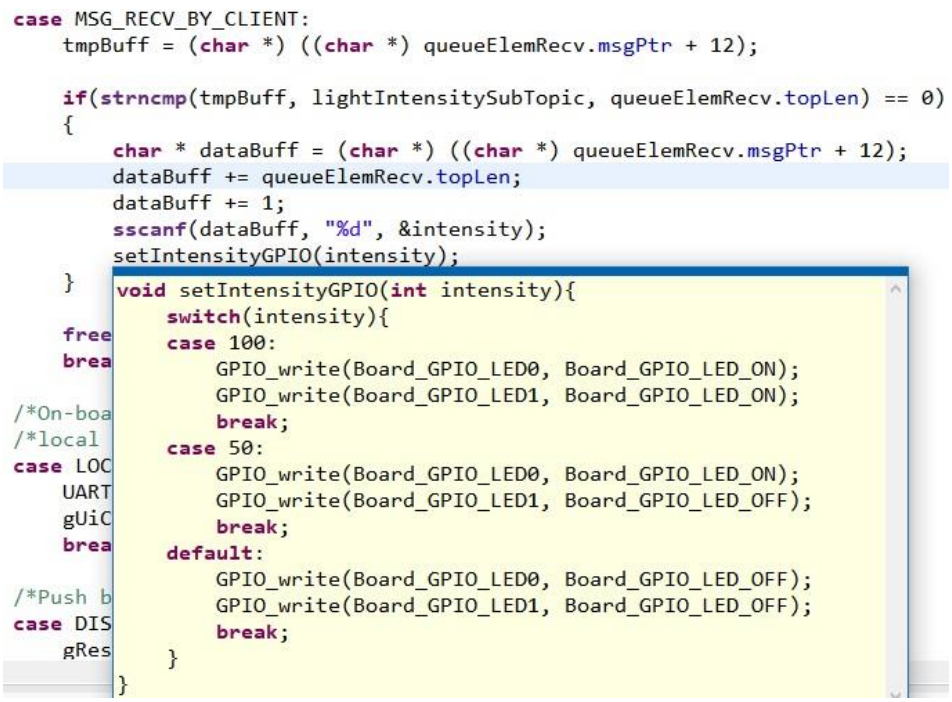

Ilustración 39: Suscripción a la intensidad de luz (código fuente del cliente CC3220SF)

El **Servidor** de la arquitectura cuenta con varios módulos cuyo papel está bien definido:

- Módulo MQTT: servicio de broker MQTT levantado en el Servidor encargado de almacenar los últimos mensajes de cada topic publicado y asegurar la entrega de mensajes a los dispositivos suscritos. Este servicio escuchará paquetes IPv4 en la interfaz predeterminada y se crea de forma separada para dotar al sistema de más estabilidad.

```
dell@homelaptop:~$ systemctl status mosquitto.service
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated)
   Active: active (running) since Sat 2018-12-29 11:49:16 CET; 3h 30min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 791 ExecStart=/etc/init.d/mosquitto start (code=exited, status=0/SUCC
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mosquitto.service
           └─909 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Dec 29 11:49:16 homelaptop systemd[1]: Starting LSB: mosquitto MQTT v3.1 message
Dec 29 11:49:16 homelaptop mosquitto[791]:  * Starting network daemon: mosquitto
Dec 29 11:49:16 homelaptop mosquitto[791]:    ...done.
Dec 29 11:49:16 homelaptop systemd[1]: Started LSB: mosquitto MQTT v3.1 message
```

Ilustración 40: Estado del broker MQTT (servicio mosquitto en Ubuntu)

- Módulo Paho Java: este programa escrito en Java será el encargado de realizar la lógica de funcionamiento en base a los mensajes publicados en el tiempo.

IntelStreetLighting.java | CC3220SF_Device.java

```
1  package project_id.artifact_id;
2
3  import java.sql.Timestamp;
15
16 public class IntelStreetLighting implements MqttCallback{
17
18 private static List<String> registeredDevices;
19 private static Hashtable<String, CC3220SF_Device> roadState;
20
21 public IntelStreetLighting() {
22     registeredDevices = Arrays.asList("98:84:E3:F6:76:D3", "98:84:E3:F7:04:FC", "98:84:E3:F6:89:C9");
23     roadState = new Hashtable<String, CC3220SF_Device>();
24
25     for(String mac: registeredDevices) {
26         roadState.put(mac, new CC3220SF_Device());
27     }
28 }
29
30 public static void subscribe_Devices_Topics(MqttAsyncClient mqttClient, List<String> registeredDevices) throws MqttExc
31     int qos = 2;
32
33     if(mqttClient.isConnected()) {
34         for(String deviceMAC: registeredDevices) {
35             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/Light_Intensity", qos);
36             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/Temperature", qos);
37             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/Number_Cars", qos);
38             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/LDR_Value", qos);
39             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/Activity_Event", qos);
40             mqttClient.subscribe("/CC3220SF/" + deviceMAC + "/Manual_Mode", qos);
41         }
```

Ilustración 41: Programa Java Paho (programa lógica funcionamiento Servidor)

Lo primero que hace es suscribirse a todos los topics de todos los dispositivos registrados (que deben figurar de antemano en la configuración del Servidor) y cuando reciban mensajes de topics de forma asíncrona actualizarlos en una lista global que refleje el estado actual de los dispositivos. Esta estructura será la que se va a analizar posteriormente para generar el nivel de iluminación que cada sección de carretera debe fijar cuando el sistema se encuentre en modo automático. En el modo manual el

programa se limitará a mantener el nivel de intensidad lumínica fijado por el módulo Node – RED.

```
public void messageArrived(String topic, MqttMessage message) throws Exception {
    String [] tokens = topic.split("[/]");
    String deviceType = tokens[1];
    String deviceMAC = tokens[2];
    String topicCategory = tokens[3];
    boolean manualMode;
    short ldrValue;
    float temperature;

    if(deviceType.equals("CC3220SF") && roadState.containsKey(deviceMAC)) {
        switch(topicCategory) {
        case "Temperature":
            temperature = Float.parseFloat(new String(message.getPayload()));
            roadState.get(deviceMAC).setTemperature(temperature);
            break;
        case "LDR_Value":
            ldrValue = Short.parseShort(new String(message.getPayload()));
            roadState.get(deviceMAC).setLdrValue(ldrValue);
            break;
        case "Activity_Event":
            roadState.get(deviceMAC).setActivityEvent(
            new String(message.getPayload()) + " " + new Timestamp(System.currentTimeMillis()));
            updateParameters(deviceMAC);
            break;
        case "Manual_Mode":
            manualMode = (new String(message.getPayload())).equals("true");
            roadState.get(deviceMAC).setManualMode(manualMode);
            break;
```

Ilustración 42: Recepción de datos de suscripciones en Java Paho (programa lógica funcionamiento Servidor)

Los topics a los que está suscrito el cliente MQTT de Java Paho, el encargado de realizar la lógica de funcionamiento, se van a recibir de forma asíncrona por lo que una vez llegados se comprueba el topic al que pertenece y se trata adecuadamente. En este caso hay datos que simplemente se almacenan en la estructura de datos de cada dispositivo CC3220SF de una sección y otros como por ejemplo *Activity_Event* requiere de una lógica adicional para realizar el comportamiento deseado del sistema.

En este caso, la función updateParameters() se ejecuta siempre y cuando se desencadena un evento de actividad en una sección, y se encarga de hacer evolucionar la cuenta de los coches de cada sección de carretera registrados y de la actualización del nivel de luminosidad atendiendo a los parámetros de modo manual, valor del LDR y el número de coches en cada sección.

```
if((currentDeviceMAC != null) && (!currentDeviceMAC.isEmpty())) {
    /* Update number of cars counters in the sections */
    for(int index = 0; index < registeredDevices.size(); index++) {
        if(registeredDevices.get(index).equals(currentDeviceMAC)) {
            /* Update current section car counter and publish to broker */
            numCars = roadState.get(currentDeviceMAC).getNumberCars();
            if(numCars < Short.MAX_VALUE) {
                roadState.get(currentDeviceMAC).setNumberCars(++numCars);
                publishToBroker("Number_Cars", currentDeviceMAC);
                updateLightIntensity(currentDeviceMAC);
            }

            /* Check if it has previous section */
            if(index > 0) {
                int previousDeviceIndex = index - 1;
                previousDeviceMAC = registeredDevices.get(previousDeviceIndex);
                /* Update previous section car counter and publish to broker */
                numCars = roadState.get(previousDeviceMAC).getNumberCars();
                if(numCars > (short) 0) {
                    roadState.get(previousDeviceMAC).setNumberCars(--numCars);
                    publishToBroker("Number_Cars", previousDeviceMAC);
                    updateLightIntensity(previousDeviceMAC);
                }
            }

            /* Check if it has next section */
            if(index < (registeredDevices.size() - 1)) {
                int nextDeviceIndex = index + 1;
                nextDeviceMAC = registeredDevices.get(nextDeviceIndex);
                roadState.get(nextDeviceMAC).setLightIntensity((short) 100);
                publishToBroker("Light_Intensity", nextDeviceMAC);
            }
        }
```

Ilustración 43: Lógica de funcionamiento del sistema en Java Paho (programa lógica funcionamiento Servidor)

- Módulo Node – RED Dashboard: este servicio ofrece una interfaz de usuario (UI) a través de la cual se puede consultar el estado de la carretera y variar su funcionamiento.

Ilustración 44: Ejecución del proceso Node – RED (servicio Node – RED en el Servidor)

La información que muestra en los gráficos y por tanto a la que se suscribe es la relativa a la temperatura, valor del LDR, la actividad de vehículos y el nivel de intensidad lumínica de cada sección.

El nodo del flujo Node – RED responsable de la suscripción se llama mqtt y en el mismo figuran los parámetros de configuración como el topic o la dirección del broker entre otros.

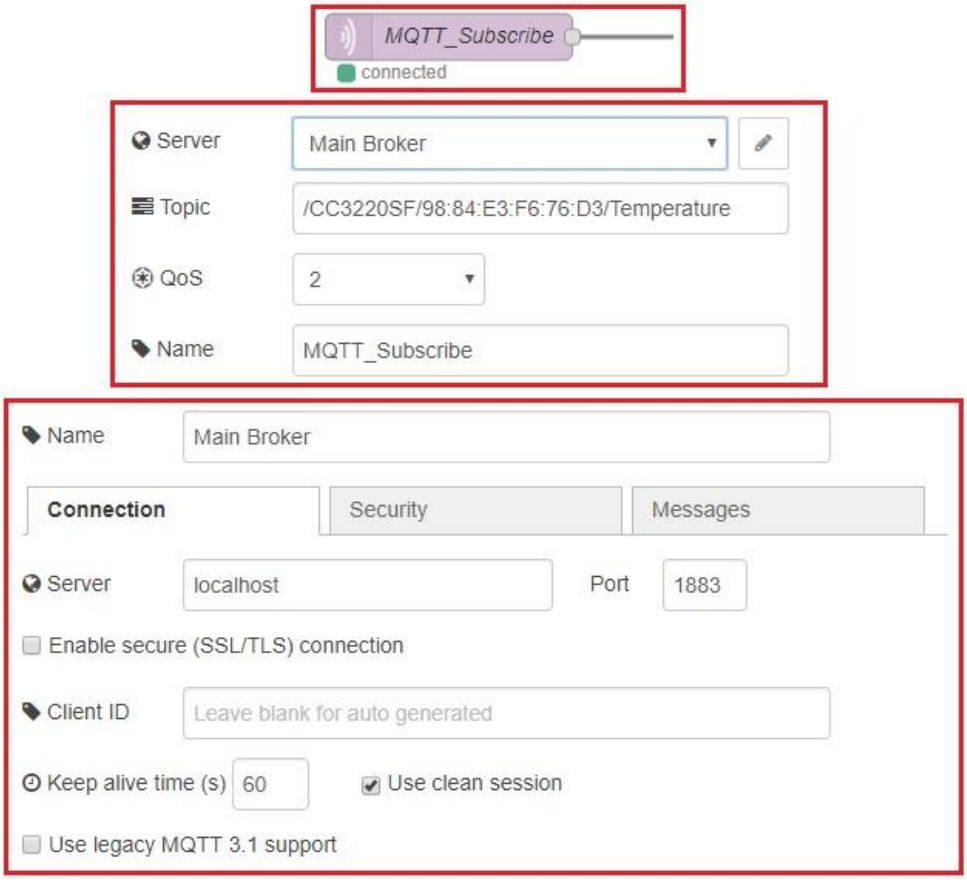

Ilustración 45: Nodo suscripción (servicio Node – RED en el Servidor)

Este tipo de nodo se repetirá para cada uno de los topics suscritos en la interfaz gráfica de usuario (“Temperature”, “LDR_Value”, “Light_Intensity” y “Number_Cars”) y para cada uno de los dispositivos MAC registrados en el Servidor (“98:84:E3:F6:76:D3”, “98:84:E3:F7:04:FC” y “98:84:E3:F6:89:C9”).

Una vez configurado, este nodo se va a quedar a la espera de recibir datos nuevos de la suscripción, y una vez obtenido un nuevo dato se lo pasa al siguiente nodo que lo va a representar en un gráfico determinado, en este caso en un gráfico que indica la variación de temperatura en el tiempo.

Ilustración 46: Nodo temperatura (servicio Node – RED en el Servidor)

Este tipo de comportamiento se hará con los demás datos a los que se suscribe, representándolos en un gráfico significativo con una determinada escala.

En la interfaz de usuario existe la posibilidad de cambiar el funcionamiento del sistema al modo manual y prefijar un valor de intensidad en cada luminaria (al cambiar estos parámetros se publica la información correspondiente al bróker MQTT que luego se hará llegar directamente a los nodos suscritos).

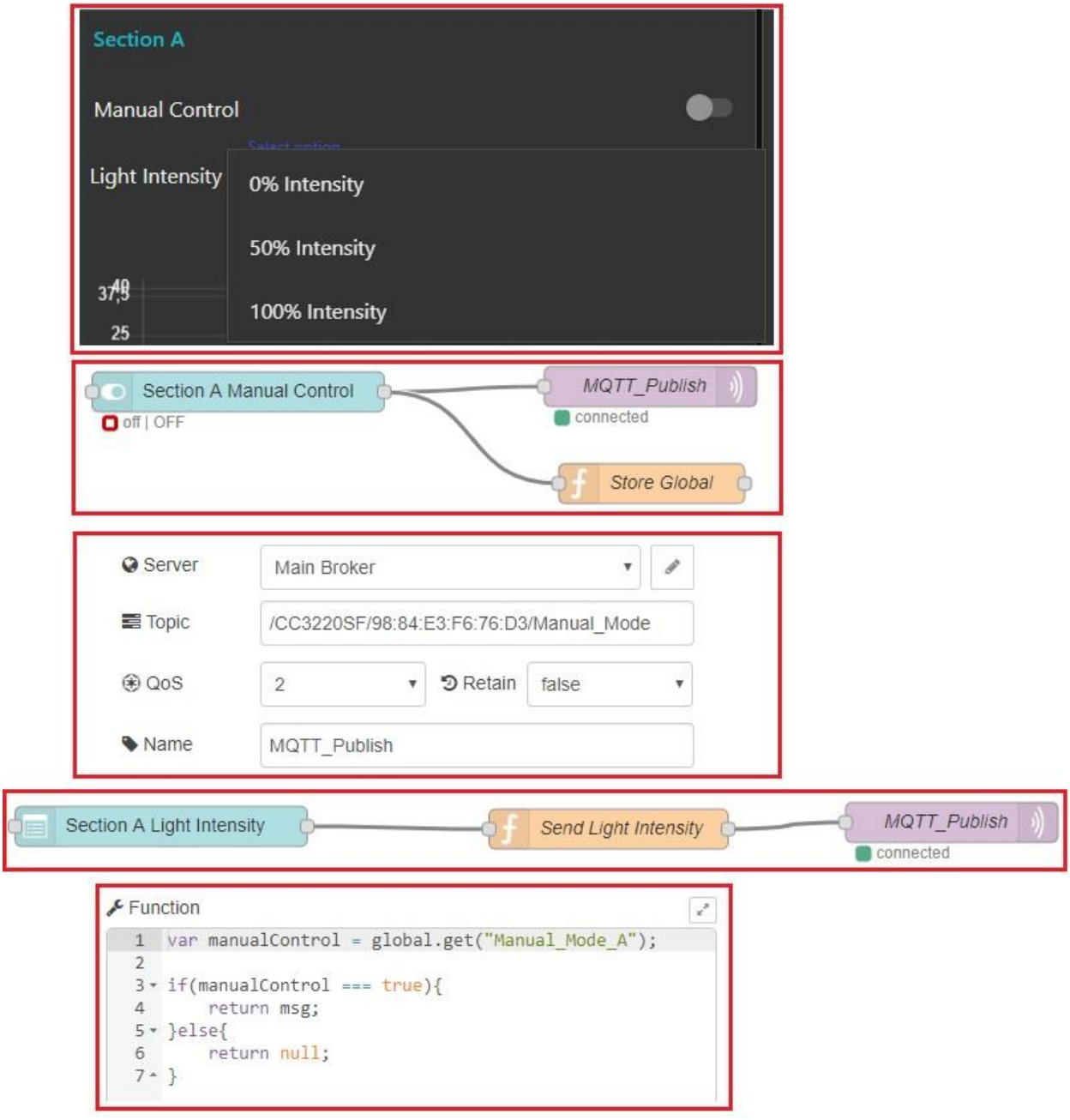

Ilustración 47: Nodos y configuración del modo manual (servicio Node – RED en el Servidor)

En la interfaz gráfica el usuario tiene la posibilidad de activar el modo manual si desea controlar las secciones de forma individual. Para ello lo que debe hacer es activar el modo manual haciendo clic en el botón de la interfaz. Esto provocará la publicación de un mensaje con valor igual a “true” y topic “Manual_Mode” de la sección de carretera o nodo en cuestión y el almacenamiento de un valor de una variable global que se usará posteriormente.

A continuación, debe desplegar la lista con valores de intensidades y hacer clic en la deseada. El nodo de la lista desplegable pasará el nivel de intensidad al nodo siguiente que se encarga simplemente de comprobar si tenemos activado el modo manual usando para ello la variable global fijada en el paso anterior. Si el modo manual está activado se pasará la información al nodo encargado de publicar el topic “Light_Intensity” con el valor seleccionado en la lista.

El funcionamiento de los nodos visto tanto para la suscripción como para la publicación se repetirá para cada una de las secciones instaladas durante la prueba.

El código fuente de los programas de la lógica de funcionamiento del cliente Java Paho y la interfaz gráfica de Node – RED se pueden consultar en el enlace compartido de Google Drive [15].

4.6 Plan de pruebas

En este apartado se pretende definir una serie de pruebas que validen el funcionamiento del sistema y de los módulos que lo componen por separado.

Para cada uno de los módulos del sistema se propone ejecutar las siguientes pruebas:

- Módulo Node – RED Dashboard: este módulo se ejecuta como un proceso en el Servidor y se entiende que la última versión ya está desplegada por completo y ejecutándose. Este módulo publica y se suscribe a datos en el broker MQTT del Servidor y la comprobación se hará por tanto publicando datos en los distintos topics con un cliente externo (y viendo que la información aparece en los gráficos de la interfaz de usuario) y publicando datos desde la interfaz de usuario y comprobando que llegan correctamente al broker MQTT.

  Por tanto, se deben definir la lista de pruebas que se va a ejecutar con sus valores determinados. El resultado se observará gráficamente en la interfaz de usuario del Dashboard. Cabe destacar que se publican valores variados incluso algunos fuera del rango de funcionamiento del sistema puesto que el objetivo es probar la llegada de los mensajes a la interfaz gráfica.

Plan de pruebas para la suscripción en Node - RED Dashboard				
		Nodos CC3220SF		
Topic	**Values**	**98:84:E3:F6:76:D3**	**98:84:E3:F7:04:FC**	**98:84:E3:F6:89:C9**
Temperature	(-15), (-5), 0, 23, 39, 60			
Number_Cars	0, 5, 10, 15			
LDR_Value	0, 500, 800, 1024, 2000			
Light_Intensity	0, 50, 57, 100, 125			

Ilustración 48: Plan de prueba suscripción Node – RED

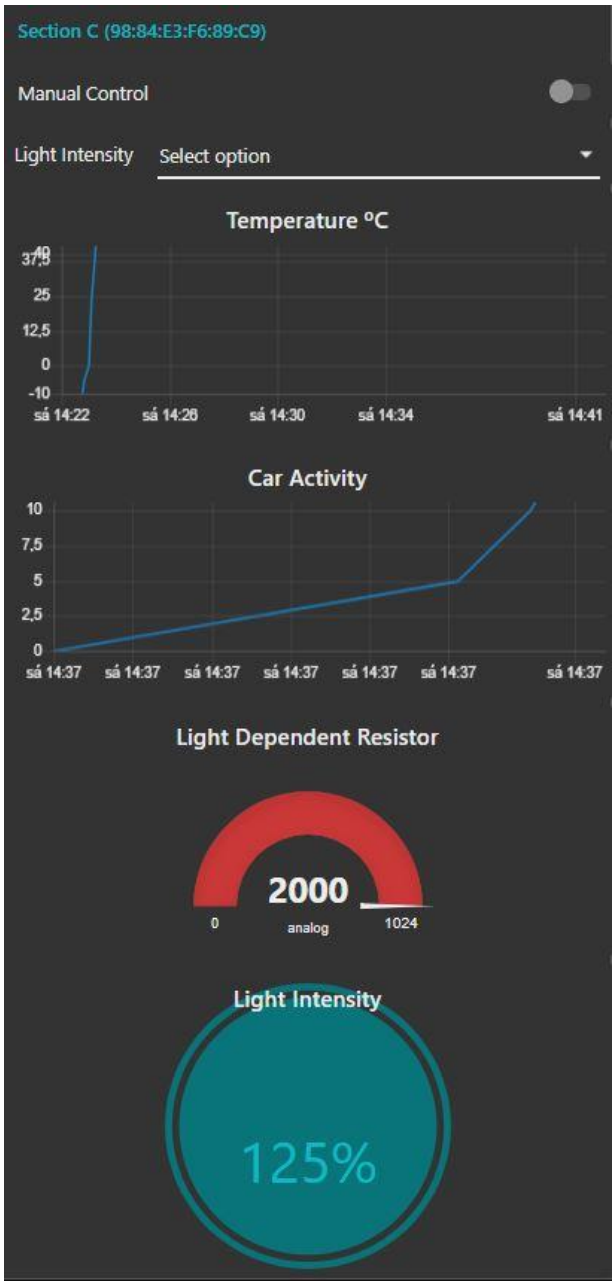

Ilustración 49: Interfaz Dashboard tras ejecución del plan de prueba

Para probar la publicación en el interfaz de usuario se procederá a ejecutar el siguiente plan de pruebas y observar los mensajes que llegan al broker MQTT.

Plan de pruebas para la publicación en Node - RED Dashboard				
		Nodos CC3220SF		
Topic	**Values**	**98:84:E3:F6:76:D3**	**98:84:E3:F7:04:FC**	**98:84:E3:F6:89:C9**
Manual_Mode	false, true			
Light_Intensity	0, 50, 100			

Ilustración 50: Plan de prueba publicación Node - RED

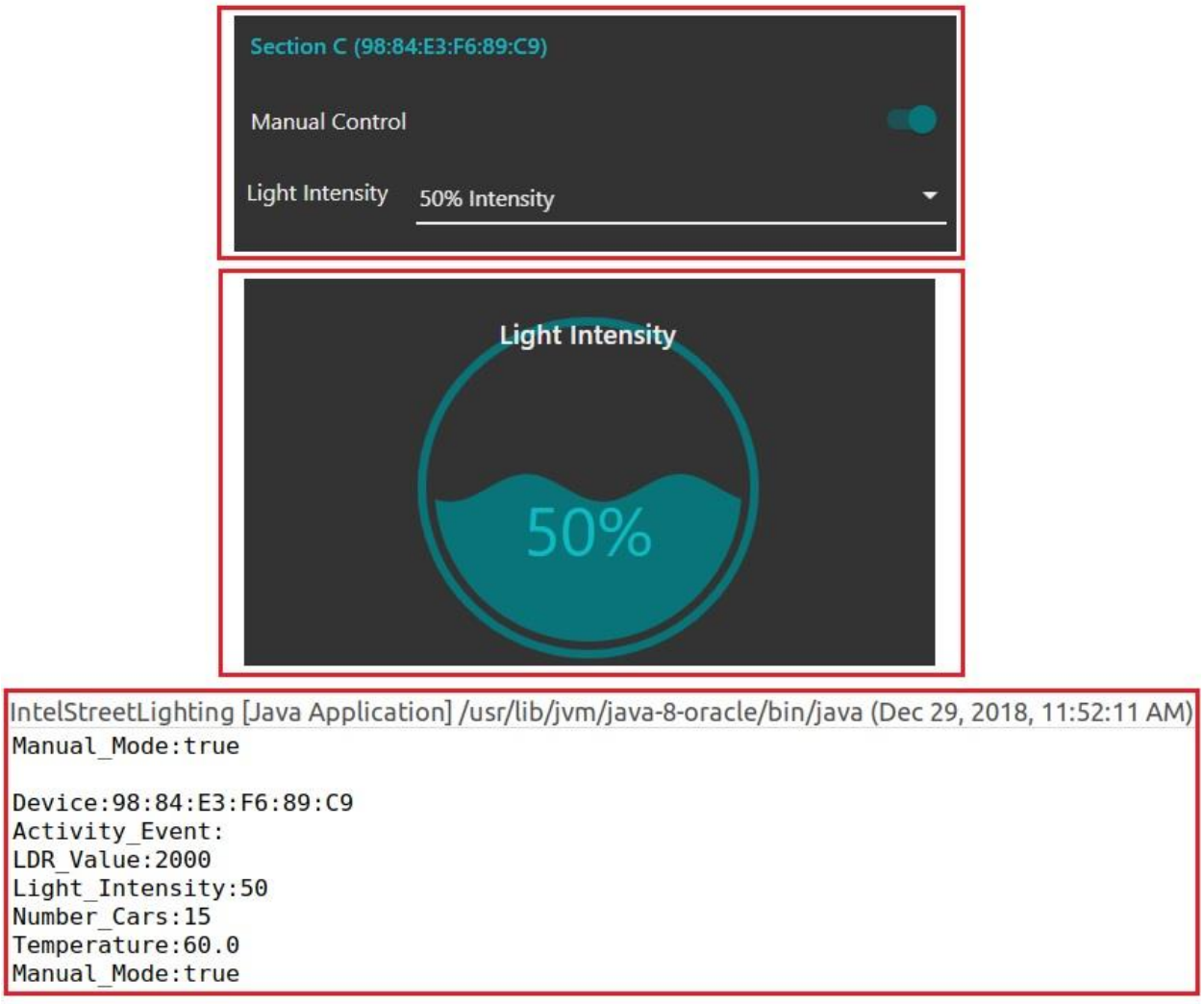

Ilustración 51: Interfaz Dashboard y broker MQTT tras ejecución del plan de prueba

- Nodo CC3220SF: para probar el dispositivo por separado vamos a ejecutar la imagen del programa en la placa e interactuar con los sensores y dispositivos externos para mandar y recibir datos del broker MQTT.

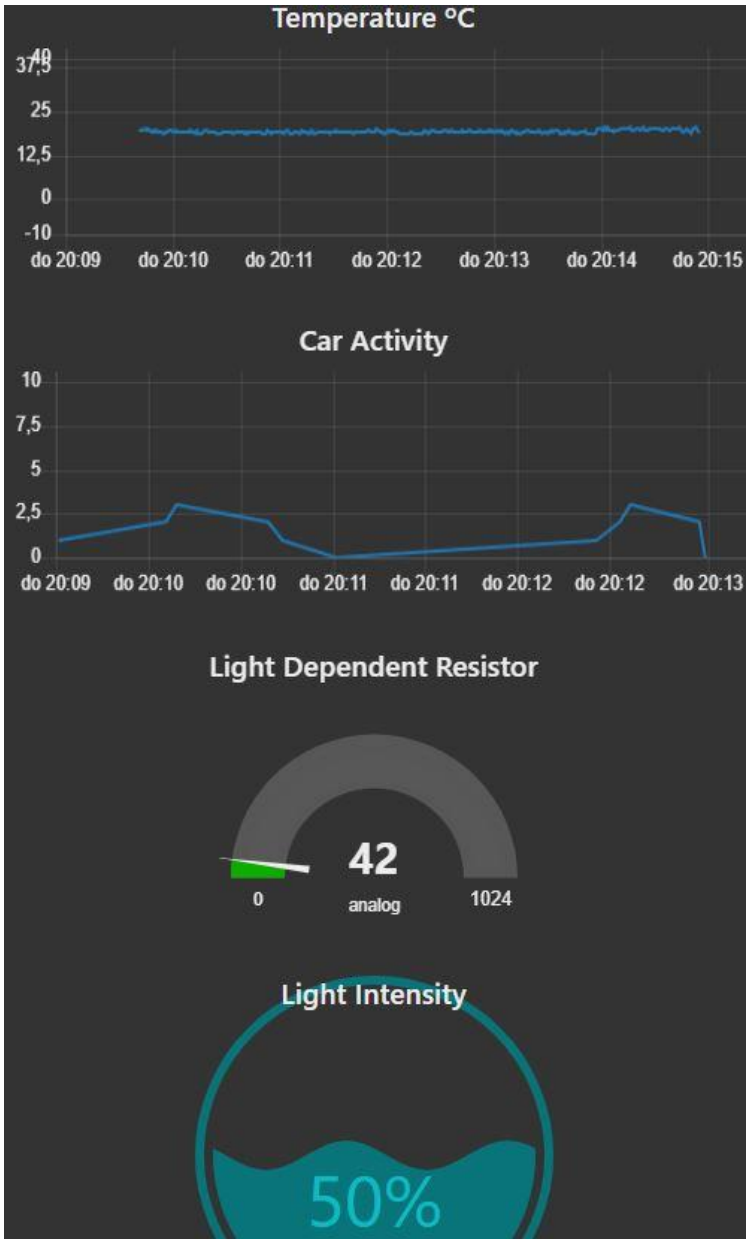

Ilustración 52: Publicación de datos desde la placa CC3220SF

Donde los mensajes recibidos en el Servidor son:

```
Device: 98:84:E3:F6:76:D3
Topic: Temperature
Intensity:50 Temp: 19.6121 Cars: 0 LDR: 42 Activity: ENTRY 2019-01-06 20:12:48.743 Manual: false
Section 0 intensity 50 cars 0
Section 1 intensity 50 cars 0
Section 2 intensity 100 cars 6
Device: 98:84:E3:F6:76:D3
Topic: LDR_Value
Intensity:50 Temp: 19.6121 Cars: 0 LDR: 43 Activity: ENTRY 2019-01-06 20:12:48.743 Manual: false
Section 0 intensity 50 cars 0
Section 1 intensity 50 cars 0
Section 2 intensity 100 cars 6
```

Ilustración 53: Mensajes recibidos desde la placa CC3220SF

Cabe la pena destacar que la última sección (número 2) cuenta como una sección adicional cuyo papel es proporcionar información para el correcto funcionamiento de la sección número 1. Se podría decir entonces que cada tramo de carretera tendrá N secciones funcionales más una sección adicional en la cola, ésta última no funcional.

- Módulo Java Paho: este componente del Servidor se va a probar para demostrar el funcionamiento básico del sistema una vez el sistema está en funcionamiento. Para ello, se arrancan correctamente los servicios del Servidor y se simulan eventos de envío de datos y de actividad de vehículos para comprobar que se lleva la cuenta correctamente de los coches que hay en cada sección y el nivel de intensidad que se calcula.

  A continuación, se probará el sistema simulando el paso de vehículos durante la noche.

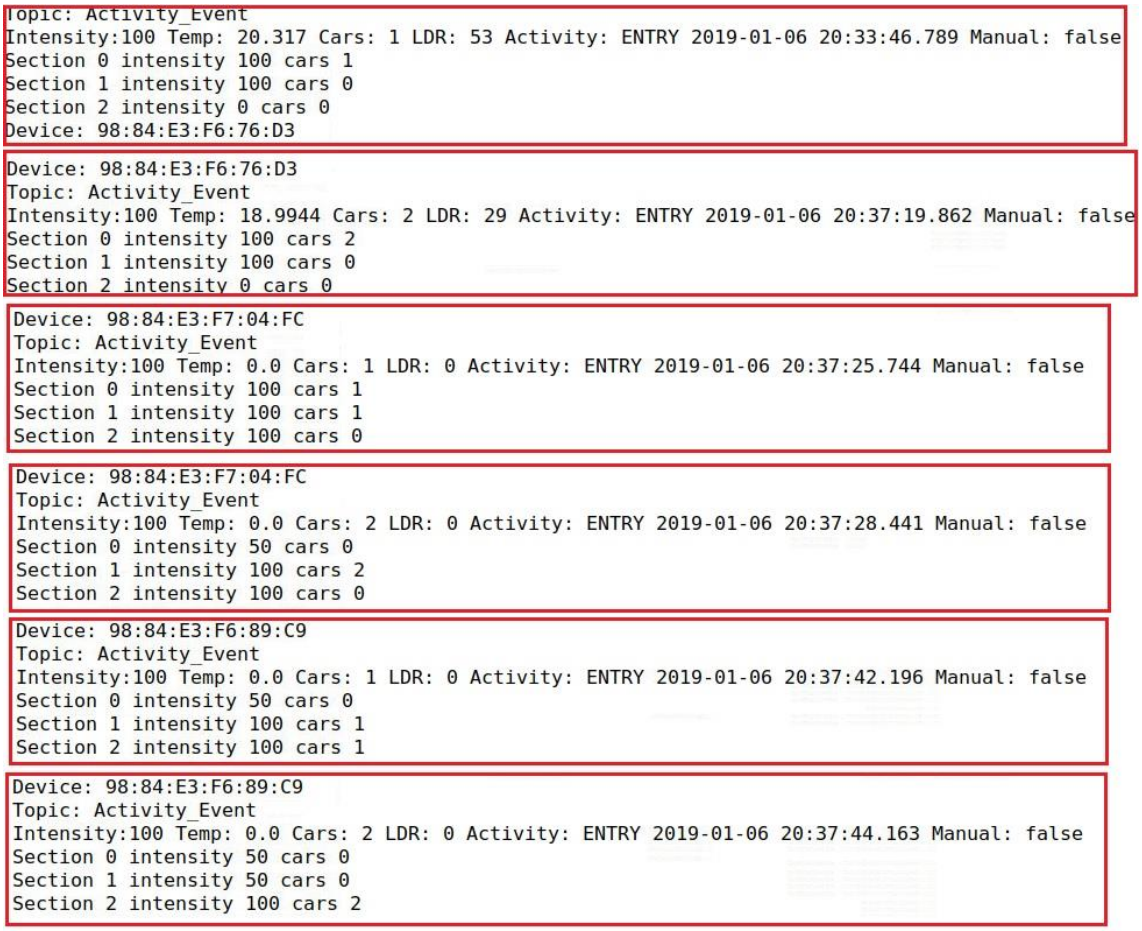

Ilustración 54: Simulación paso de coches durante la noche (LDR < 100)

Si se ejecutara la misma prueba con el valor de LDR > 100 se estaría simulando la luz natural de día por lo que el sistema seguirá llevando la cuenta de los coches, pero mantendrá la intensidad de luz en la sección correspondiente a cero.

Capítulo 5: Resultados obtenidos

En este Trabajo Fin de Máster se ha pretendido llevar a cabo el diseño e implementación de un Sistema de Iluminación Inteligente en Carreteras en el que se tengan en cuenta parámetros ambientales y de actividad en la carretera para ser capaces de regular la intensidad de luz de forma inteligente.

Tras la ejecución de las pruebas se ha visto el buen desempeño de las tecnologías empleadas, en concreto el protocolo de envío de mensajes MQTT por la facilidad de pasar mensajes con información para distintos dispositivos que lo requieran y conseguir mantenerlos actualizados en todo momento con información que generan otros dispositivos.

Por otro lado, el Dashboard de Node – RED destaca por facilitar la tarea de creación de una interfaz gráfica y de control para el usuario siempre y cuando se quiere visualizar información del comportamiento del sistema e introducir nuevos parámetros que hagan que el sistema cambie de funcionamiento.

Por último, el propósito de esta implementación se limita a un tramo de carretera con un único carril y sentido y dos secciones funcionales. La elección de los sensores y actuadores, así como los métodos de conexión inalámbrica de los módulos se adecuan a una serie de pruebas realizadas en el laboratorio.

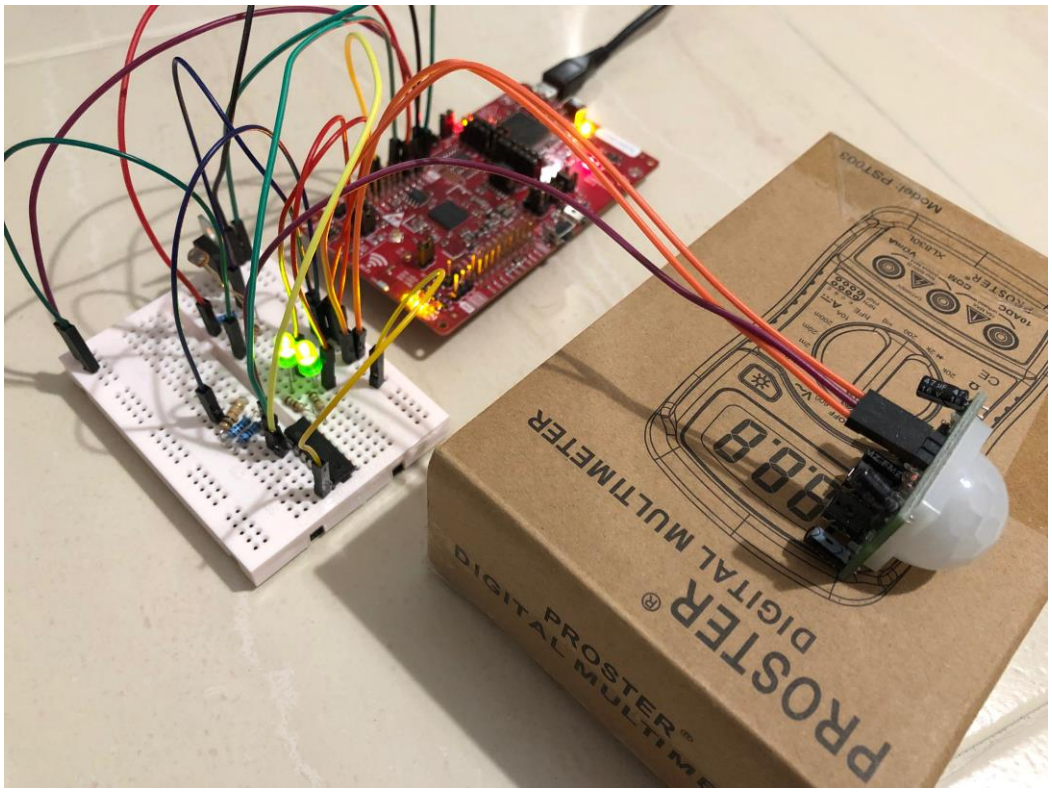

Ilustración 55: Nodo CC3220SF con sensores y actuadores

Capítulo 6: Conclusiones y trabajo futuro

En el presente Trabajo Fin de Máster se ha abordado la creación e implementación de un Sistema de Iluminación Inteligente en Carreteras. En el mismo trabajo se ha hecho un recorrido desde explicar el funcionamiento deseado del sistema detallando sus partes y el papel que desempeñan, diseñar una arquitectura del sistema, definir la red de comunicaciones que va a interconectar los elementos de la arquitectura, definir el protocolo de comunicación de nivel de aplicación entre las placas de desarrollo CC3220SF y el Servidor, diseñar un modelo de hilos de ejecución del sistema y codificarlo en el lenguaje de programación correspondiente y por último programar la imagen de la aplicación en las placas y probar el sistema en su totalidad.

Se ha podido comprobar durante la fase de ejecución del plan de pruebas que los elementos de la arquitectura y la unión de estos funcionan correctamente de acuerdo con el sencillo funcionamiento deseado presentado anteriormente en este trabajo.

En el caso de querer instalar el sistema en la carretera, se puede adaptar la conexión inalámbrica de comunicaciones actual (WiFi) para usar tecnologías como por ejemplo LoRaWAN, GSM o 5G pues también funcionan sobre TCP (requisito para el protocolo de capa de aplicación MQTT). También en un sistema real el hardware instalado en las luminarias cuenta con protección IP contra agua y polvo para proteger al dispositivo de las condiciones meteorológicas en el paso del tiempo.

Como trabajo futuro se propone el cambio de arquitectura a un sistema distribuido (véase el apartado de trabajo existente) en el que los nodos correspondientes a las luminarias se puedan comunicar directamente de forma inalámbrica usando por ejemplo una red mesh. También se propone el uso de los sockets seguros, el cifrado de la información, así como métodos de autenticación para evitar posibles ataques contra la seguridad de la información.

Capítulo 7: Introduction (English)

7.1 Lighting needs

Since the first civilizations established the human being created different types of devices to satisfy its lighting needs considered to be basic and for its comfort. Example of those devices are the torch, candles made from vegetal or animal oil, or candelabrum with plenty of decorations from each time [1].

Later, combustion lighting devices were replaced by those that use electricity (with inert gas or semiconductors) to achieve the same purpose, leaving behind the use of the old ones that were only used during official events, religious ceremonies or for home decoration purposes. The engineering and technological innovation along with the improvement of manufacturing processes led to a revolution in lighting design, transforming LED and sodium lamps lighting in the main technologies used to light cities and roads.

7.2 Standard lighting

This thesis aims the standard and high-speed traffic in roads and cycle lanes.

There's a large quantity of vehicles that travel during night time and poor lighting conditions can lead to deadly accidents. This happens because the darkness itself alters the driver's visual abilities leading to symptoms such as loss of visual acuteness, insufficient binocular vision, limited chromatic vision and a decrease in side objects sensing among others. To these psychological and physiological issues, headlights limited visual information and the movement of the vehicle itself and other vehicles adds up. Therefore, we want to emphasize the importance of road lighting to reduce nighttime car accidents.

In addition, road lighting means a high percentage of the energy use of a local government as well as pollution (light pollution or CO2 gases), therefore it is important to study the processes to be able to provide techniques to help reduce pollution and energy consumption [4].

The type of lamps that are used in public lighting facilities are the following:

- Compact fluorescent lamps (CFL)
- Mercury vapour lamps
- Low Pressure Sodium lamps (LPS)
- High Pressure Sodium lamps (HPS)
- Metal Halide lamps
- High intensity discharge lamps (HID)
- LED lamps

Any type of lamp can be used to light but about 80% of the facilities use HPS lamps and the rest use Metal Halide lamps. In tunnels, LPS lamps are more frequent because of their high lighting effectiveness.

The adjustment of the levels of intensity of light during nighttime is possible only if such a thing guarantees traffic and pedestrians security and the area of study does not have a high percentage of nighttime accidents or pedestrian walkways involving criminal activities. The intensity of light can be adjusted using series arrangements of inductive electric controllers that can double power or electronic devices that can adjust power [5].

7.3 Lighting adverse effects

In this section, several studies or facts about problems originated from lighting cities and villages will be discussed and documented.

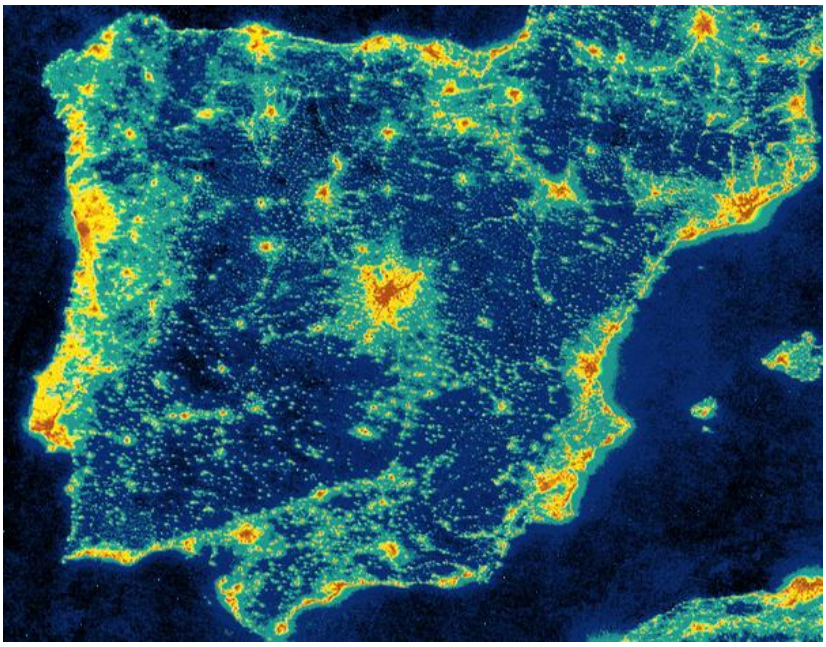

Illustration 1: Light pollution map of Spain (lightpollutionmap.info)

First, light pollution is one of the problems caused by lighting and it happens because of gasses and particles in the atmosphere undergoing diffusion and reflection of the light and it changes our astronomical observation of celestial objects. To tackle this issue what is done is study the areas that have most amount of sky glow and see if it's possible to reduce some parameters that lead to this adverse effect without altering traffic and pedestrian safety or quality of life.

All in all, the purpose of this system is to make uniform the pollution in those areas that are possible excluding those areas where light pollution cannot be reduced or controlled [4].

Illustration 56: Lighted house in the woods (waterplace.net)

Secondly, artificial light has been used to light places during night for hundreds of years and continues to expand with housing developments and economic progress. Also, several studies have shown that artificial light could alter invertebrates behavior (insects, arachnids, mollusks, jellyfish, starfish or worms) among others, their reproduction success, survival and communities' composition throughout the day. This variation causes the number of predators and scavengers to grow in places with a lot of lighting. The studies show that artificial light changes the environment organizational structure and alters the ecosystem normal function [6].

Other studies show that light pollution altering invertebrates is a global problem in growth. Several tests have been done with endangered species of bats (Rhinolophus hipposideros) and have seen that their movement and migration has been altered because of insects being attracted to Metal Halide lamps [7].

The last aspect to study related to artificial light is energy consumption and the way that energy has been generated may cause different amounts of pollution. It is therefore important to rely on renewable energy to maintain pollution at a minimum.

7.4 Need for a change

It’s obvious that technology drove various fields of study and application with the goal to make processes better and concerning the issue of lighting we must improve the way we light our cities and villages whenever is possible.

Standard light intensity controllers do not measure activity (traffic and pedestrian) in the area where they are installed and because of this they do not maximize utility or reduce energy consumption during the time of day when there is less activity (for example during nighttime in specific roads).

Therefore, we suggest using technology to analyze the activity in a road and adjust light intensity in a smart way during the night, maximizing the utility and minimizing the energy consumption.

Capítulo 8: Conclusions and future work (English)

In this thesis an Intelligent Street Lighting System has been created and implemented. The system desired behavior with its components and their purpose has been covered and the system architecture design, the communication network, the communication protocol between the CC3220SF development boards and the Server, the design of the software threads, the programming in a specific language and the flashing of the image in the devices and testing.

The test plan has been verified to execute successfully and therefore checking that the architecture components and the communication between them work as expected.

If the system was to be installed in a real application the wireless communication over WiFi can be replaced with technologies like LoRaWAN, GSM or 5G as they also work over TCP (one requirement for the application protocol MQTT). Also, in a real system the hardware installed in the lights usually have IP protection against water and dust to protect the device from adverse weather conditions and improve its lifetime.

As future work we suggest a change in the architecture to a distributed one (check the corresponding section in this thesis) where the nodes would directly communicate with each other in a wireless manner using a mesh network. Also, communication security techniques must be implemented (safe sockets, encrypted information and authentication techniques) to prevent as much as possible the attacks to the security of the information.

Bibliografía

[1] Andrés Antonio Gil Martin (2009), “Historia de la iluminación”, Tecnología en Educación Secundaria, ISSN: 1988-6047

[2] Nexia (2017), “12 Ventajas de utilizar iluminación LED”, Blog oficial, URL: http://www.nexia.es/es/blog/12-ventajas-utilizar-iluminacion-led (Último acceso: julio 2018)

[3] Revolución energética (2008), “LED: Diodos Emisores de Luz Información de la Tecnología”, Greenpeace Argentina

[4] IDEA, Comité Español de Iluminación (2001), “Guía Técnica de Eficiencia Energética en Iluminación en Alumbrado Público”

[5] Fenercom (2013), “Guía de Gestión Energética en el Alumbrado Público”, Consejería de Economía y Hacienda

[6] Thomas W. Davies, Jonathan Bennie y Kevin J. Gaston (2012), “Street lighting changes the composition of invertebrate communities”, Biology Letters Community ecology, DOI: 10.1098/rsbl.2012.0216

[7] Emma Louise Stone, Gareth Jones, Stephen Harris (2009), “Street Lighting Disturbs Commuting Bats”, University of Bristol, DOI: 10.1016/j.cub.2009.05.058

[8] Keyur K Patel, Sunil M Patel (2016), “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling of Technologies, Application & Future Challenges”, Faculty of Technology and Engineering-MSU India, DOI: 10.4010/2016.1482, ISSN: 2321 3361 (2016) IJESC

[9] Ron Davies (2015), “The Internet of Things Opportunities and challenges”, European Parliamentary Research Service, URL: http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/557012/EPRS_BRI(2015)557012_EN.pdf (Último acceso: enero 2019)

[10] InteliLight (2016), “The first LoRaWAN compatible streetlight control pilot project in Moldova implemented by Orange Moldova and InteliLight”, URL: https://intelilight.eu/the-first-lorawan-compatible-streetlight-control-pilot-project-in-moldova-implemented-by-orange-moldova-and-intelilight/ (Último acceso: enero 2019)

[11] Qulon (2018), “Fully Intelligent Lighting Control Solutions”, URL: http://www.qulon.pro/intelligent-street-lighting-control (Último acceso: enero 2019)

[12] Yusaku Fujii, Noriaki Yoshiura, Akihiro Takita, Naoya Ohta (2013), “Smart Street Lighting System with Energy Saving Function Based on The Sensor Network”

[13] Laclaustra, I.M. (2016), “Sistema domótico distribuido para controlar el riego y el aire acondicionado en el hogar”, Enseñanza y Aprendizaje de Ingeniería de Computadores, 6: 87-101, URL: http://hdl.handle.net/10481/41915 (Último acceso: enero 2019)

[14] Jesús Martín Alonso and Alberto A. Del Barrio (2016), A distributed HW-SW platform for fireworks, “Proceedings of the Summer Computer Simulation Conference” (SCSC '16), Montreal, Canada, Article 17, 7 pages

[15] Emanuel Stanescu (2019), Código Fuente de los Desarrollos, URL: https://drive.google.com/drive/folders/1gYY0FSm1nfUdWc3CQe0T3M1k1gMFQUiH?usp=sharing

Licencia de uso

El contenido de este Trabajo Fin de Máster se escribe y publica para ser usado con fines académicos.

Durante el desarrollo del software implicado en el sistema, se han usado librerías de tecnologías y código fuente para la realización de la lógica de funcionamiento deseada. El mismo se publica bajo la licencia GPLv2 cuya finalidad es respetar los derechos de autor originales de las tecnologías y código fuente. En concreto, las tecnologías empleadas con su licencia y las aportaciones propias se enumeran a continuación:

- Tecnología Eclipse Java Paho: sus licencias son *Eclipse Distribution License 1.0 (BSD) y Eclipse Public License 1.0*. Esta tecnología se ha usado para crear un programa desde cero para la lógica de funcionamiento del sistema en el Servidor.
- Tecnología Node – RED: se encuentra bajo la licencia *Apache License 2.0.* Se ha usado para crear una interfaz gráfica desde cero.
- Librerías y código fuente de Texas Instruments: para la creación de la lógica de funcionamiento en los nodos CC3220SF, se parte de un cliente MQTT cuyo código fuente pertenece a TI. El mismo se ha modificado para implementar la lógica de funcionamiento deseada.

La licencia bajo la cual se encuentra el código fuente del cliente MQTT se adjunta a continuación:

```
/*
 * Copyright (c) 2016, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
```