

UNIVERSIDAD DE MÁLAGA
ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

EVALUACIÓN DEL MICROCONTROLADOR
CC3200 COMO PLATAFORMA IOT

GRADO EN INGENIERÍA DE
SISTEMAS ELECTRÓNICOS

JESÚS ALBERTO CEJAS LÓPEZ
MÁLAGA, 2017

Evaluación del microcontrolador CC3200 como plataforma IoT

Autor: Jesús Alberto Cejas López

Tutor: José Manuel Cano García / Eva González Parada

Departamento: Departamento de Tecnología Electrónica

Titulación: Grado en Ingeniería de Sistemas Electrónicos

Palabras clave: Sistema empotrado, sensores, internet de las cosas.

Resumen

En este proyecto se ha llevado a cabo la evaluación del microcontrolador CC3200 y de los protocolos que permiten su conexión a Internet para construir una plataforma IoT. Este microcontrolador se encuentra integrado en la placa de desarrollo CC3200-LAUNCHXL de Texas Instruments, que incluye un sensor de temperatura por infrarrojos y un sensor de aceleración de tres ejes. Además, se conectará de forma externa a través del bus SPI un sistema de iluminación con varios puntos de luz de tipo LED interconectados entre sí con capacidad para ser controlados individualmente por una sola línea.

En concreto se han desarrollado dos aplicaciones: por un lado, una aplicación que se ejecutará en el microcontrolador que proporcionará la capacidad para configurar y controlar tanto los sensores como el sistema de iluminación.

Por otro lado, se ha desarrollado una aplicación gráfica de prueba que se ejecutará en un PC. Esta aplicación dispondrá de los controles necesarios para gestionar el sistema de iluminación y la lectura de los sensores representando las medidas proporcionadas por el microcontrolador.

La comunicación entre ambas aplicaciones se llevará a cabo de forma inalámbrica ya que tanto el dispositivo en el que se ejecuta la aplicación de control como el dispositivo microcontrolador se encontrarán conectados mediante *Wi-Fi* a un *broker* MQTT, un protocolo de comunicación por mensajería.

Evaluation of CC3200 microcontroller as IoT platform

Author: Jesús Alberto Cejas López

Supervisor: José Manuel Cano García / Eva González Parada

Department: Departamento de Tecnología Electrónica

Degree: Electronic Systems Engineering Degree

Keywords: Embedded system, sensors, internet of things.

Abstract

In this project has focused on the evaluation of the CC3200 microcontroller and the protocols that allow its connection to the Internet to build an IoT platform. This microcontroller is integrated in the development board CC3200-LAUNCHXL of Texas Instruments which also includes an infrared temperature sensor and a three axis acceleration sensor. In addition, a lightning strip with several LED points is connected to the SPI bus of the microcontroller. The LEDs are connected to each other through a one-wire control interface and can be individually addressed in order to set their RGB value.

Two applications have been developed to perform this evaluation: on the one hand, an application running on the microcontroller that will provide the configuration and control of the sensors and the lighting system.

On the other hand, a graphic user interface test application that runs on a PC has been developed. This application allows to manage the lighting system, the reading of the sensors and represent the measurements provided by the microcontroller.

Both the GUI and the microcontroller application are MQTT clients that connect to a MQTT broker in order to exchange data and control messages to interact with each other.

Contenido

Capítulo 1. Introducción	1
1.1. Contexto Tecnológico.....	3
1.1.1. Redes de sensores inalámbricas	3
1.1.2. Comparativa de tecnologías.....	4
1.2. Objetivos del proyecto	5
1.3. Estructura de la memoria.....	6
Capítulo 2. Especificaciones del sistema	9
2.1. Especificaciones de conexión	10
2.2. Especificaciones del microcontrolador	11
2.3. Especificaciones de la aplicación de usuario	12
2.4. Requisitos.....	12
Capítulo 3. Desarrollo del sistema	17
3.1. Hardware.....	17
3.1.1. Sensores	19
3.1.2. Sistema de iluminación.....	20
3.2. Software.....	22
3.2.1. Capas del software.....	23
3.2.2. FreeRTOS.....	28
3.2.3. MQTT.....	28
3.2.4. JSON	29
3.2.5. Funciones y tareas del microcontrolador	29
3.3. Aplicación gráfica del PC.....	33
Capítulo 4. Pruebas y verificación	39
4.1. Resultados.....	43

Capítulo 5. Conclusiones y líneas futuras	51
Apéndice A. Manual de usuario	55
Referencias.....	61

Lista de Acrónimos

AP	Access Point
BLE	Bluetooth Low Energy
BSP	Board Support Package
CPU	Central Processing Unit
DMA	Direct Memory Access
GPIO	General Purpose Input/Output
I ² C	Inter Integrated Circuit
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LED	Light Emitting Diode
M2M	Machine to Machine
MQTT	Message Queue Telemetry Transport
OASIS	Organization for the Advancement of Structured Information Standards
PC	Personal Computer
RAM	Random Access Memory
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter

Capítulo 1. Introducción

Hace algunos años la idea de que una ciudad tuviera la capacidad de ser inteligente parecía ser sacada de una película de ciencia ficción, pero los diferentes avances tecnológicos y la aparición de dispositivos computacionales de pequeño tamaño y ultra bajo consumo han acercado esta idea hasta tal punto de hacerla realidad. Las tecnologías de la información y comunicación hacen posible la monitorización, el análisis y la planificación del comportamiento de una ciudad, así como la automatización de funciones rutinarias tanto de la propia ciudad como de sus ciudadanos.

El IoT (*Internet of Things*) es una novedosa forma de comunicación que consiste en la interconexión de dispositivos y la interacción máquina-máquina con el fin de controlar y captar información del medio que nos rodea. Esto es posible gracias a la adición de microcontroladores e internet a objetos cotidianos para dotar a estos con la suficiente inteligencia como para interactuar con otros objetos. Por ejemplo, que el frigorífico de casa pueda detectar la necesidad de reponer un producto y automáticamente lo adquiera a través de internet para que lo envíen a casa o que se controle el nivel de contaminación presente en una gran ciudad. Existen numerosos campos de aplicaciones tales como la domótica en el hogar, la automatización industrial, gestión inteligente de la energía, gestión del tráfico, ayudas en la medicina o asistencia a nuestros mayores. [1]

Este proyecto se enmarca en una nueva línea de investigación del Departamento de Tecnología Electrónica que pretende construir un entorno de desarrollo para aplicaciones de inteligencia ambiental a escala, usando microcontroladores, sensores, actuadores y dispositivos de comunicación. Dichos entornos estarían orientados a personas que no disponen de conocimientos técnicos (principalmente niños pequeños),

empleando juegos de bloques tipo Lego para construir las estructuras físicas con las que trabajar, y realizando la programación de dichos entornos mediante un lenguaje de bloques.

En nuestro caso, el proyecto se centrará en una aplicación concreta de inteligencia ambiental como son las ciudades inteligentes. Mediante este ejemplo se podrán contextualizar los objetivos que se pretenden alcanzar en este proyecto.

Podemos decir que el concepto de *Smart City* se define como el uso de tecnologías de la información y de las comunicaciones en una ciudad para hacer que sus servicios públicos sean más interactivos y eficientes para los ciudadanos, haciendo su vida más cómoda. Se puede considerar que una ciudad es inteligente cuando este uso de la tecnología fomenta la calidad de vida y la gestión de los recursos naturales. Esto permite maximizar la economía, la sociedad, el entorno y el bienestar de las ciudades y conlleva un comportamiento más sostenible.

La ciudad inteligente es un espacio con innumerables sensores y actuadores de todo tipo (incluyendo a los propios ciudadanos que pueden tener un comportamiento de sensor o actuador) que tiene la capacidad de escuchar lo que está pasando y tomar las decisiones adecuadas basándose en el principio de Recolección de datos – Procesado de datos – Toma de decisiones. [2]

En cuanto a la gestión inteligente de una ciudad podemos nombrar un proyecto del ayuntamiento del municipio madrileño de Las Rozas: “Las Rozas Smart Green City”. [3]

Este proyecto conlleva la implementación de una aplicación tanto para teléfonos inteligentes como ordenadores con la idea de que los propios ciudadanos (tanto locales como visitantes al municipio) puedan alertar de diversas deficiencias o desperfectos que pueda haber en el alumbrado, limpieza, parques y jardines, saneamiento o en la vía pública y así comunicarlo a los servicios municipales encargados de su reparación.

La aplicación permite hacer una foto de la incidencia y determinar la ubicación de la misma haciendo uso de la geolocalización que proporciona el teléfono o de forma manual. Esta incidencia se comunica tanto a los servicios técnicos como a los responsables en el ayuntamiento. Además, incluye seguimiento en tiempo real del estado de solución de la incidencia.

Esta idea de ciudad inteligente destaca por ser algo diferente a la idea original de *Smart City*. En una ciudad inteligente a priori, los encargados de proporcionar

información para actuar y mejorar la vida en la ciudad suelen ser sensores electrónicos. En este caso, los propios ciudadanos son los sensores que recogen información de las incidencias de la ciudad lo que promueve la participación ciudadana para resolver sus propias necesidades. Tal y como declaran los responsables de este proyecto: “(los ciudadanos) se convierten en un activo fundamental para la gestión municipal”.

En el apartado técnico, existen algunos elementos que son indispensables a la hora de construir una estructura *Smart City* como las redes de sensores, las cuales son imprescindibles para recoger información del medio, o las diferentes tecnologías radio que nos permiten conectar nuestros elementos de forma inalámbrica.

1.1. Contexto Tecnológico

1.1.1. Redes de sensores inalámbricas

Las redes de sensores inalámbricas son el elemento que hace posible la idea de una ciudad u hogar inteligente ya que son los sensores quienes se encargan de recoger la información que se tratará de forma inteligente para actuar en consecuencia. Conforme la tecnología avanza han ido apareciendo nuevas aplicaciones. Actualmente las redes de sensores se aplican en los siguientes ámbitos:

- Teleasistencia médica: las redes de sensores pueden medir parámetros biométricos para monitorizar de forma remota a personas con problemas de salud como pueden ser ancianos o personas discapacitadas.
- Monitorización medioambiental: el pequeño tamaño y la autonomía (en cuanto a consumo de energía se refiere) de los sensores hace posible la monitorización del estado del entorno, por ejemplo, colocando sensores en los animales.
- Monitorización preventiva: las redes de sensores pueden permitir monitorizar de forma continua parámetros de elementos mecánicos como puentes o piezas de maquinaria haciendo posible predecir una rotura o accidente.
- Agricultura: en cultivos como la vid se está aplicando el uso de redes de sensores inalámbricas para controlar la calidad del mismo. Estos dispositivos electrónicos permiten medir la calidad y escasez del agua para determinar el

momento de riego, predecir la aparición de plagas o determinar el grado de madurez de un fruto a través de la radiación solar.

- Automatización de edificios: gracias a las redes de sensores inalámbricas se pueden integrar en un solo sistema las soluciones que ya se aplican a la automatización del hogar o de oficinas como la climatización, control de luz o seguridad.
- Medidas de contadores: actualmente se están implantando contadores inteligentes que comunican las lecturas correspondientes a la empresa suministradora. También se podrán usar para ofrecer servicios al cliente como la monitorización o el análisis del consumo del suministro.
- Ciudades inteligentes: para mejorar la vida del ciudadano, las redes de sensores se pueden aplicar a la iluminación pública, la recogida de basuras o la gestión de las zonas de aparcamiento público.

1.1.2. Comparativa de tecnologías

En cuanto a las tecnologías de comunicación inalámbricas con las que se puede implementar las aplicaciones del IoT vamos a comparar el rendimiento general de dos de ellas, BLE (*Bluetooth Low Energy*) y *Wi-Fi*:

- En cuanto a consumo de energía se refiere encontramos que la tecnología BLE tiene un consumo mucho menor que Wi-Fi. Este punto es muy importante cuando hablamos de aplicaciones en dispositivos móviles ya que la alimentación de estos se lleva a cabo mediante baterías de capacidad limitada.
- Otro punto importante es el rango de funcionamiento que nos limita la distancia a la que dos dispositivos pueden comunicarse de forma inalámbrica. En este aspecto Wi-Fi presenta un mayor rango que BLE.
- El ancho de banda puede ser una característica importante dependiendo de la aplicación. Wi-Fi dispone de un mayor ancho de banda que BLE.

A la hora de escoger una tecnología concreta de comunicación debemos tener en cuenta los requerimientos de nuestra aplicación para decidir qué tecnología se adecua mejor a nuestras especificaciones.

1.2. Objetivos del proyecto

A continuación, se detallarán los objetivos contemplados en la realización de este proyecto.

Para empezar, se debería aclarar que este es un proyecto que forma parte de otro mucho más ambicioso.

Mientras que la idea general sería el diseño de una maqueta a pequeña escala de una *Smart City*, con herramientas para recolectar datos, otras para aplicar inteligencia y procesar dichos datos y otras para actuar en consecuencia, en este caso nos centraremos en estudiar y evaluar la idoneidad de una plataforma concreta y tecnología radio concretas para la futura implementación de algunos de los elementos anteriormente mencionados.

En cuanto al proyecto general podemos decir que se trata de acercar el mundo de las ciudades inteligentes a personas que no disponen de los suficientes conocimientos tecnológicos para diseñar una *Smart City* como pueden ser niños pequeños. En concreto, la idea es diseñar pequeños kits autoinstalables que contengan los distintos elementos electrónicos para la implementación de una ciudad inteligente y una aplicación gráfica clara e intuitiva para configurar su comportamiento de una forma sencilla abstrayendo al usuario de cómo se ha implementado la solución.

En nuestro caso estudiaremos la viabilidad que nos ofrecería el microcontrolador CC3200 de Texas Instruments con conectividad *Wi-Fi* como futuro kit de inteligencia haciendo uso de algunos sensores I²C (Inter Integrated Circuit) los cuales son incluidos en la placa de desarrollo CC3200-LAUNCHXL como son un sensor de temperatura por infrarrojos y un sensor de aceleración de tres ejes. Evaluaremos también el correcto comportamiento de un sistema de iluminación compuesto por un conjunto de puntos de luz de tipo LED (*Light-Emitting Diode*) con capacidad de comunicación en serie unos con otros.

Para hacer accesible el control de las funciones del microcontrolador como pueden ser el control de sensores o la gestión del sistema de iluminación se desarrollará una aplicación de usuario gráfica con una interfaz clara e intuitiva.

Analizaremos el protocolo de comunicación basado en mensajes MQTT (*Message Queue Telemetry Transport*) como sistema de transporte de la información inalámbrico desde el kit de inteligencia a la aplicación de usuario y viceversa.

Por lo tanto, podemos resumir que los objetivos de nuestro proyecto son los siguientes:

- Evaluar la idoneidad del microcontrolador CC3200 de Texas Instruments como dispositivo computacional de inteligencia para una aplicación de *Smart City*.
- Evaluar el funcionamiento de un sistema de iluminación compuesto por luces LED WS2812B y la librería para su control.
- Evaluar Wi-Fi como tecnología radio para la comunicación inalámbrica entre el microcontrolador y la aplicación de usuario.
- Evaluar MQTT como protocolo de comunicación entre el microcontrolador y la aplicación de usuario.
- Desarrollar una aplicación de usuario de prueba que haga accesible el control de los elementos (sensores y sistema de iluminación) desde un PC (Personal Computer) sin necesidad de tener conocimientos tecnológicos.

1.3. Estructura de la memoria

En este apartado procederemos a describir la estructura de este documento.

La memoria queda estructurada en los siguientes capítulos:

- Capítulo 1 – Introducción: en este capítulo se hará una breve introducción tratando el contexto tecnológico en el que nos encontramos, haciendo una comparativa entre distintas tecnologías y describiendo los objetivos generales del proyecto.
- Capítulo 2 – Especificaciones y requisitos: a continuación, se procederá a describir las especificaciones de nuestro sistema y a detallar los requisitos que este debe cumplir asociando también una serie de pruebas que verifiquen el cumplimiento de los mismos.
- Capítulo 3 – Desarrollo: este capítulo será dividido en dos grandes bloques para diferenciar el apartado del hardware y del software. En el primero serán descritos los distintos bloques que componen nuestro hardware y las principales

características de sus componentes. En el segundo se detallará la comunicación entre nuestro hardware, el firmware del microcontrolador y la aplicación gráfica para el usuario.

- Capítulo 4 – Pruebas y verificación: en este capítulo se encuentra la relación de pruebas llevada a cabo para verificar que nuestro sistema cumple los requisitos anteriormente declarados.
- Capítulo 5 – Conclusiones y líneas futuras: finalmente, se procederá a comentar las conclusiones a las que se ha llegado tras la finalización del proyecto y serán expuestas las posibles líneas futuras.
- Apéndice – Manual de usuario: como apéndice, será incluido un manual de usuario para facilitar la usabilidad de nuestro sistema.

Capítulo 2. Especificaciones del sistema

En este capítulo se detallarán cuáles van a ser las especificaciones de nuestro sistema y una relación de requisitos que deberá cumplir. Para evaluar el cumplimiento de estos requisitos se diseñará una serie de pruebas.

En la figura 2.1 se puede observar el diagrama de bloques del sistema donde distinguimos 3 bloques generales:

- Bloque microcontrolador: este bloque está constituido por el microcontrolador, los sensores y el sistema de iluminación y forma el objeto IoT introduciendo la recolección de datos (sensores), la inteligencia (microcontrolador) y actuación (sistema de iluminación).
- Bloque PC: este bloque se compone de una aplicación gráfica instalada en un PC. Esta aplicación da acceso al usuario a controlar y configurar el objeto IoT.
- Bloque de conexión: este bloque consiste en una red Wi-Fi que permitirá la conexión inalámbrica entre el bloque microcontrolador y el bloque PC.

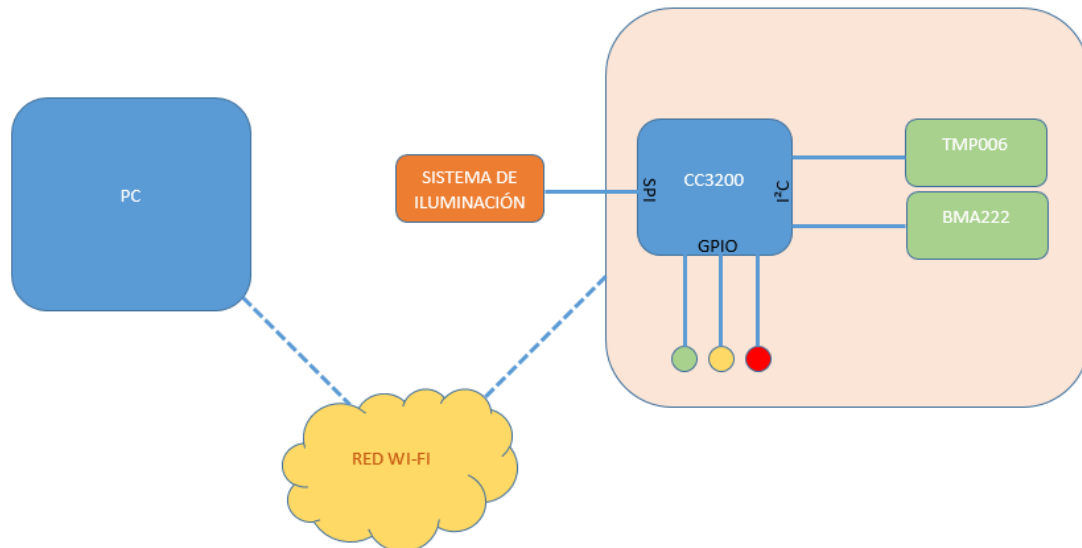


Figura 2.1. Diagrama de bloques del sistema

Para la implementación de este proyecto se ha elegido el microcontrolador CC3200 de la familia SimpleLink de Texas Instruments de arquitectura ARM Cortex-M4 el cual dispone de conectividad *Wi-Fi* y 256kB de memoria RAM (*Random Access Memory*). En concreto se ha usado la placa de desarrollo CC3200-LAUNCHXL que incluye de dos sensores integrados, TMP006 de temperatura y BMA222 de aceleración [4].

2.1. Especificaciones de conexión

La comunicación entre el microcontrolador y el equipo de usuario se hará mediante mensajes MQTT donde habrá un *broker* que gestiona la red y el microcontrolador (o microcontroladores) y el equipo de usuario sus posibles clientes.

En la figura 2.2 podemos observar un ejemplo que muestra la capacidad que tiene nuestro sistema de controlar varios dispositivos controladores distinguiendo perfectamente cada uno de sus *topics*, donde escribiremos o leeremos según nos convenga, gracias a la estructura jerarquizada.

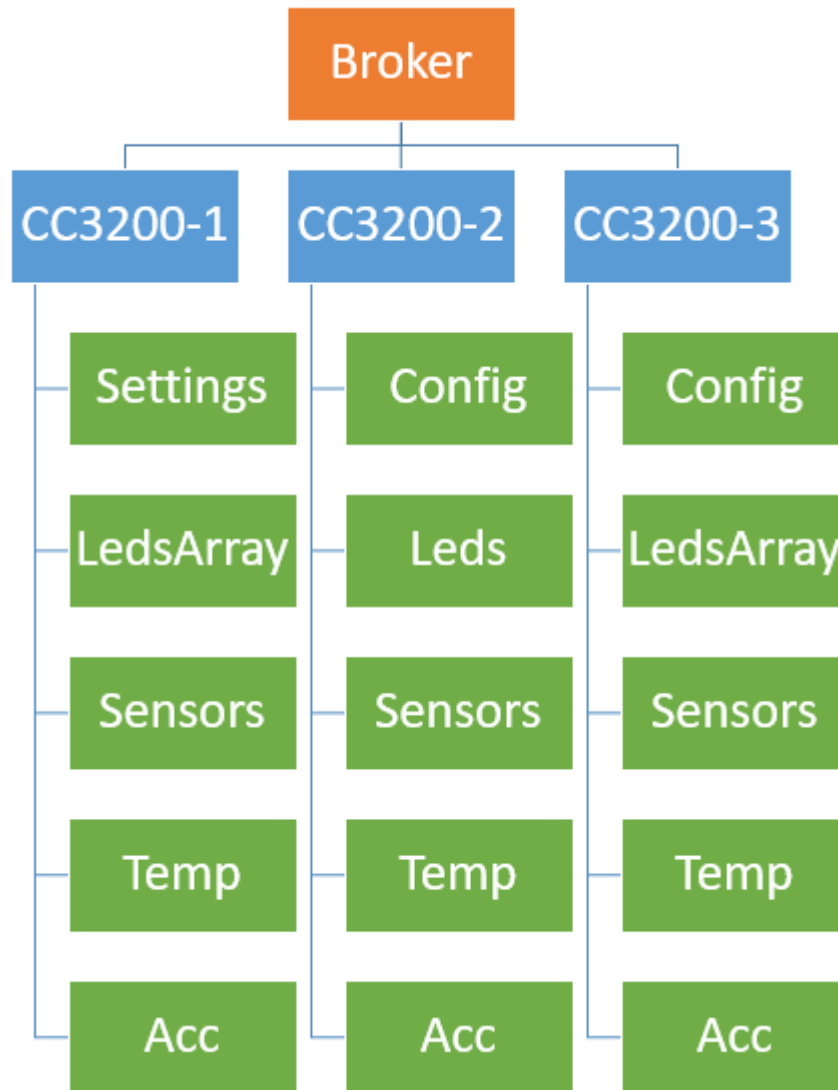


Figura 2.2. Esquema de *topics* en un sistema de varios dispositivos.

Esta forma de conexión hace posible la implementación de una red distribuida de microcontroladores con capacidad de responder ante el equipo de usuario.

2.2. Especificaciones del microcontrolador

Nuestra CPU (*Central Processing Unit*) estará gobernada por un sistema operativo de tiempo real (FreeRTOS en nuestro caso) ya que debe ser capaz de realizar varias tareas en paralelo como por ejemplo llevar a cabo la lectura de los dos sensores a la vez incluyendo la comunicación con el equipo de usuario.

El microcontrolador tendrá una estructura jerarquizada de *topics*, los cuales serán configurables en tiempo de ejecución (excepto uno de ellos que será

configurado por defecto ya que servirá para recibir la información correspondiente a la configuración deseada desde el equipo de usuario).

Se dispondrá de un *topic* para controlar el sistema de iluminación, otro para activar y configurar la lectura de los sensores (configurables individualmente) y otro por cada sensor mediante el cual se leerán las medidas desde el equipo de usuario.

2.3. Especificaciones de la aplicación de usuario

En cuanto al control de luces se establece como especificación que el usuario podrá configurar el color de cada punto de luz individualmente mediante tres valores en rangos de 0 a 255 correspondientes a los colores rojo, verde y azul (formato RGB) y apagar todos los puntos de luz si así lo desea.

Por otro lado, los sensores serán configurables individualmente de forma que el usuario pueda activarlos estableciendo un tiempo de actualización automática de la medida (tiempo de refresco en milisegundos) configurable en la interfaz de usuario. Será posible visualizar dichas medidas tanto en una gráfica que expondrá un registro de las últimas veinte medidas como en un visualizador específico para cada sensor que mostrará la última medida realizada.

2.4. Requisitos

A continuación, se detallan los requisitos del sistema, tanto del dispositivo microcontrolador como de la aplicación del equipo de usuario, en forma de tabla.

Ámbito	ID	Nombre	Descripción	Prioridad	Prueba
Firmware	1	Conexión con equipo de usuario	El sistema debe ser capaz de conectarse al servidor MQTT especificado y dar información de depuración acerca del éxito.	A	1, 2
	1.1	Creación de AP	El sistema debe crear un punto de acceso <i>Wi-Fi</i> , si así ha sido configurado, con los parámetros especificados.	B	1
	1.2	Conexión a AP	El sistema debe conectarse al punto de acceso <i>Wi-Fi</i> especificado, si así ha sido configurado.	A	2
	1.3	Información de depuración	El sistema ofrecerá información de depuración mediante el puerto serie.	C	1, 2
	1.4	Conexión servidor MQTT	El sistema debe conectarse al servidor MQTT lanzado en el equipo de usuario.	A	1, 2
	2	Configuración de sensores	El sistema debe configurar los pines correspondientes para leer de ellos las medidas de los sensores.	A	5
	3	Topic de configuración	El sistema debe suscribirse a un <i>topic</i> general por el que recibirá la información correspondiente a los <i>topics</i> de cada propósito ante los que debe responder.	B	1, 2

Ámbito	ID	Nombre	Descripción	Prioridad	Prueba
Firmware	4	Configuración de topics	El sistema debe recibir de forma adecuada la información correspondiente a los <i>topics</i> de cada propósito, suscribirse a los que proceda y almacenar en memoria los destinados a publicación.	B	3
	4.1	Recepción de información	El sistema debe recibir adecuadamente la información enviada desde el equipo de usuario y almacenarla en memoria.	B	3
	4.2	Suscripción de topics	El sistema debe suscribirse a los <i>topics</i> destinados a la recepción de órdenes desde el equipo de usuario.	B	3
	5	Control de LED	El sistema debe controlar de forma adecuada, atendiendo a las peticiones recibidas desde la aplicación de usuario, el sistema de luces LED.	A	4
	5.1	Recepción de órdenes LED	El sistema debe recibir adecuadamente las órdenes relacionadas con el sistema de iluminación enviadas desde el equipo de usuario y almacenar la información necesaria en memoria para su posterior actuación. En este caso se recibirá tanto el punto de luz a encender como el color en formato RGB.	A	4
	5.2	Actuación LED	El sistema debe actuar de forma adecuada en el sistema de iluminación atendiendo a las órdenes recibidas desde el equipo de usuario. Dichas órdenes son encender un punto de luz en particular en formato RGB o apagar todos los puntos de luz.	A	4

Ámbito	ID	Nombre	Descripción	Prioridad	Prueba
Firmware	6	Control de sensores	El sistema debe controlar de forma adecuada, atendiendo a las peticiones recibidas desde la aplicación de usuario, cada uno de los sensores de forma individual.	A	5
	6.1	Recepción de órdenes sensores	El sistema debe recibir adecuadamente las órdenes relacionadas con los sensores enviadas desde el equipo de usuario y almacenar la información necesaria en memoria para su posterior actuación. En este caso se recibirá tanto el sensor a activar como el período de refresco.	A	5
	6.2	Actuación sensores	El sistema debe configurar la lectura de los sensores atendiendo a las órdenes recibidas desde el equipo de usuario activándola de forma periódica	A	5
	7	Respuesta antes varias tareas	El sistema debe responder ante cualquier orden a pesar de que se encuentre atendiendo otra de forma periódica.	A	6
	8	Configuración de topics modificable	El sistema debe tener la capacidad de cambiar la configuración de los <i>topics</i> en cualquier momento de la ejecución.	B	7

Ámbito	ID	Nombre	Descripción	Prioridad	Prueba
Interfaz gráfica	9	Conexión con MQTT	La aplicación de usuario debe ser capaz de conectarse al servidor MQTT configurado en la entrada de texto correspondiente y suscribirse a los <i>topics</i> de recepción de medidas.	A	3
	10	Interfaz de configuración	La interfaz debe incluir todos los elementos necesarios para realizar una adecuada configuración tanto en el software del microcontrolador como en el equipo de usuario.	A	3
	10.1	Especificación de topics	La interfaz debe incluir las suficientes cajas de entrada de texto para especificar los <i>topics</i> .	B	3
	10.2	Control de configuración	La interfaz debe incluir un botón para enviar la información de configuración al microcontrolador.	B	3
	11	Control de LED	La interfaz debe incluir todos los elementos necesarios para controlar el sistema de iluminación.	A	4
	11.1	Control individual	La interfaz debe incluir un control para especificar el punto de luz concreto a controlar y una ruleta de colores para especificar el color deseado.	A	4
	11.2	Apagar sistema de iluminación	La interfaz debe incluir un control para apagar todos los puntos de luz.	C	4

Capítulo 3. Desarrollo del sistema

En este capítulo se describirá tanto el hardware usado en la implementación del proyecto como el *software* desarrollado para proporcionar la funcionalidad deseada.

3.1. Hardware

El diagrama de bloques *hardware*, el cual podemos observar en la figura 3.1 se compone por un PC en el cuál se instalará la aplicación de usuario de prueba y un bloque constituido por la placa de desarrollo CC3200-LAUNCHXL de Texas Instruments y un elemento externo conectado por el bus SPI al microcontrolador. Este elemento externo es un sistema de iluminación de varios puntos de luz de tipo LED. La conexión entre el PC y el dispositivo microcontrolador se llevará a cabo de forma inalámbrica mediante una red *Wi-Fi* a la que ambos dispositivos deberán estar conectados.

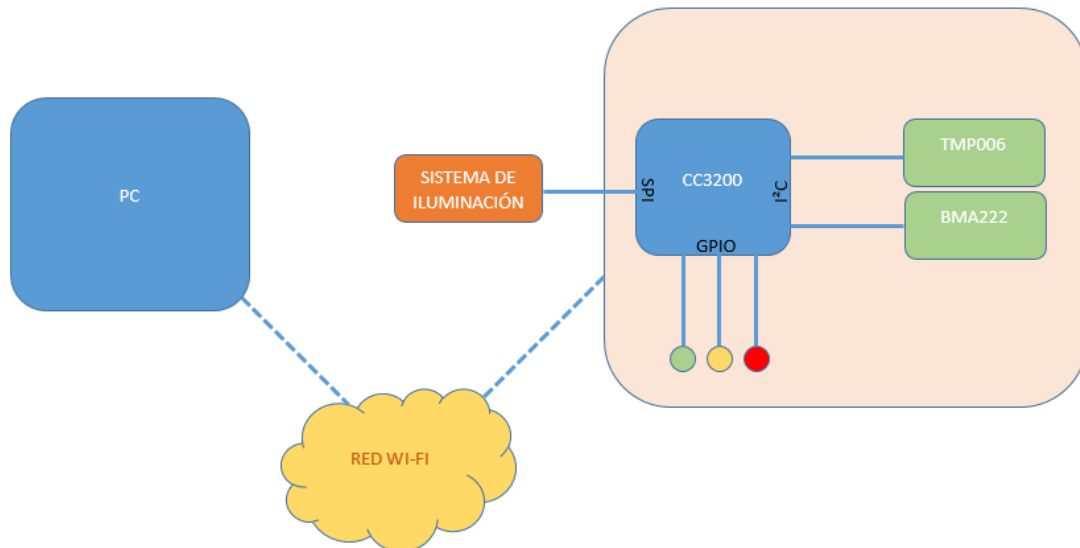


Figura 3.1. Diagrama de bloques del *hardware* del sistema.

El bloque principal es el microcontrolador CC3200 con arquitectura ARM Cortex-M4 y conectividad Wi-Fi el cual podemos encontrar integrado en la placa de desarrollo CC3200-LAUNCHXL de Texas Instruments. Esta placa dispone de dos sensores integrados (TMP006 y BMA222) los cuáles son accesibles a través del bus I²C y tres diodos LED de distintos colores. Se usará el bus SPI para controlar el sistema de iluminación el cual dispone una serie de LED con capacidad para comunicarse en serie unos con otros. Podemos observar físicamente esta placa en la figura 3.2 dónde se ubican físicamente sus principales componentes [5].

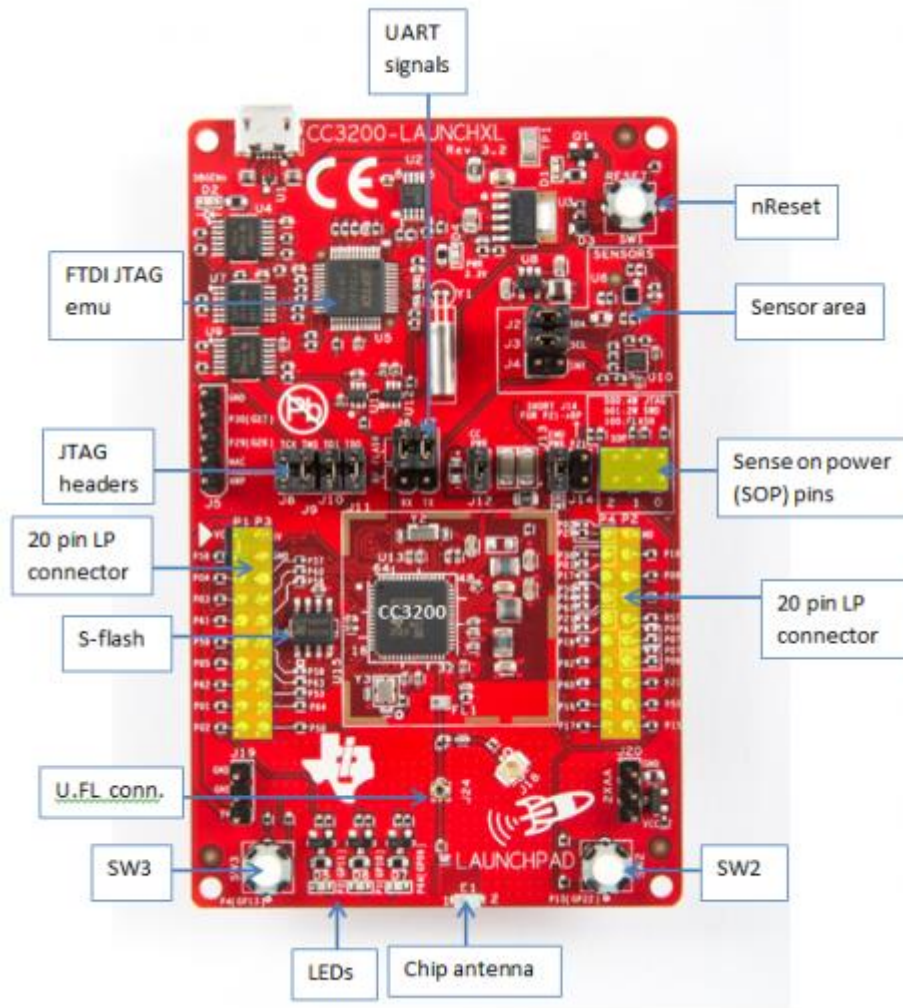


Figura 3.2. Imagen de la placa de desarrollo CC3200-LAUNCHXL. Fuente: [5]

3.1.1. Sensores

A continuación, se detallará la información de cada uno de los sensores:

- Sensor de temperatura (TMP006): se trata de un sensor infrarrojo de temperatura capaz de medir la temperatura de un objeto sin estar en contacto con el mismo. Su funcionamiento se basa en una termopila que absorbe la energía infrarroja que emite cualquier objeto para determinar su temperatura. Es un sensor limitado para trabajar en determinadas condiciones ambientales, desde -40°C hasta 125°C , aunque es posible medir temperaturas fuera de ese rango siempre que el propio sensor se encuentre en el rango de temperatura adecuado. [6]

- Sensor de aceleración (BMA222): este es un sensor de aceleración de tres ejes destinado a electrónica de bajo consumo ya que tiene un consumo máximo de corriente de $139\mu\text{A}$. Permite medir aceleraciones en tres ejes perpendiculares para detectar inclinación, movimiento, choques o vibración y es un sensor altamente configurable para dar flexibilidad al diseñador a la hora de integrarlo en un sistema. [7]

Para acceder a la configuración y lectura de dichos sensores se hará uso de las librerías TMP006drv y BMA222drv desarrolladas por Texas Instruments.

3.1.2. Sistema de iluminación

A continuación, detallaremos cómo se compone nuestro sistema de iluminación el cual podemos observar su aspecto en la figura 3.3. Este sistema se compone de varios puntos de luz WS2812B. Este tipo de luces integran un chip driver dotándolas de capacidad para la comunicación de datos en serie unas con otras. Cada punto de luz requiere de tres conexiones: dos de ellas dedicadas a la alimentación (GND y VCC) y la última dedicada a la transmisión de los datos que configurarán nuestros puntos de luz individualmente. Este sistema se usará en nuestro proyecto como simulación de lo que pudiera ser un sistema de iluminación real por ejemplo el sistema de iluminación pública de una calle en la que podríamos controlar el apagado, encendido e intensidad de cada uno de los puntos de luz de forma remota. [8]



Figura 3.3. Muestra del sistema de iluminación usado en el proyecto.

En cuanto al protocolo de comunicación que sigue este sistema podemos describirlo como una serie de pulsos que se reciben a través de la línea de datos y que se propaga de punto a punto. Podemos distinguir los diferentes pulsos que el driver reconoce en la figura 3.4.

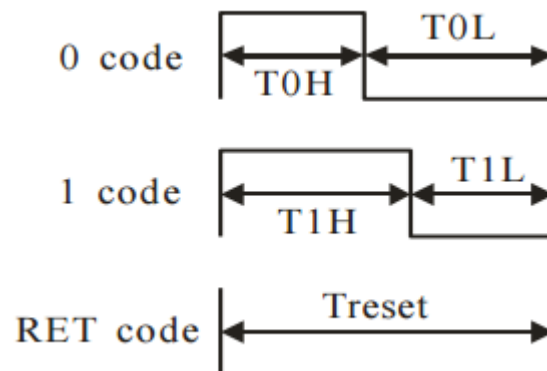


Figura 3.4. Pulsos reconocibles por el driver. Fuente: [8]

Este protocolo tiene unos requisitos temporales para reconocer los códigos 0, 1 y RESET. Podemos observar en la tabla 3.1 los valores que el fabricante nos recomienda seguir para asegurar el correcto funcionamiento del sistema de iluminación. A efectos prácticos, cabe mencionar, que estos valores admiten cierta tolerancia por lo que no se deben tomar a rajatabla.

Data transfer time(TH+TL=1.25 μ s \pm 600ns)

T0H	0 code ,high voltage time	0.4us	\pm 150ns
T1H	1 code ,high voltage time	0.8us	\pm 150ns
T0L	0 code , low voltage time	0.85us	\pm 150ns
T1L	1 code ,low voltage time	0.45us	\pm 150ns
RES	low voltage time	Above 50 μ s	

Tabla 3.1. Requisitos temporales para la comunicación de los datos del sistema de iluminación.**Fuente: [8]**

3.2. Software

En este apartado procederemos a describir el software desarrollado para la realización de este proyecto. Podemos distinguir dos partes: el *firmware* del microcontrolador y la aplicación gráfica para el PC.

Por una parte, se ha desarrollado el *firmware* del microcontrolador que implementará la parte funcional de este proyecto. Basado en un sistema operativo de tiempo real como es FreeRTOS, este software gestionará la conexión del microcontrolador a un servidor MQTT, ofreciendo la capacidad de comunicación con otras aplicaciones de forma inalámbrica y el control sobre los sensores incluidos en la placa de desarrollo y el sistema de iluminación externo.

Por otra parte, se ha desarrollado una interfaz gráfica de prueba que será de utilidad a la hora de comprobar el correcto funcionamiento del sistema ya que nos permitirá de forma más sencilla comunicar órdenes al dispositivo microcontrolador y recibir información del mismo.

La comunicación de estas dos aplicaciones es posible gracias al protocolo de comunicación MQTT que, como podemos observar en la figura 3.5 actúa de enlace entre las mismas por lo que podremos llevar el control del sistema y recibir información del mismo en la aplicación gráfica.

Esta forma de conexión, centralizada en un servidor MQTT, hace posible la comunicación entre varios equipos de usuario y varios dispositivos microcontroladores otorgando al sistema la capacidad para, por ejemplo,

gestionar varios microcontroladores con un único PC o gestionar el mismo microcontrolador desde varios PC diferentes de forma remota.

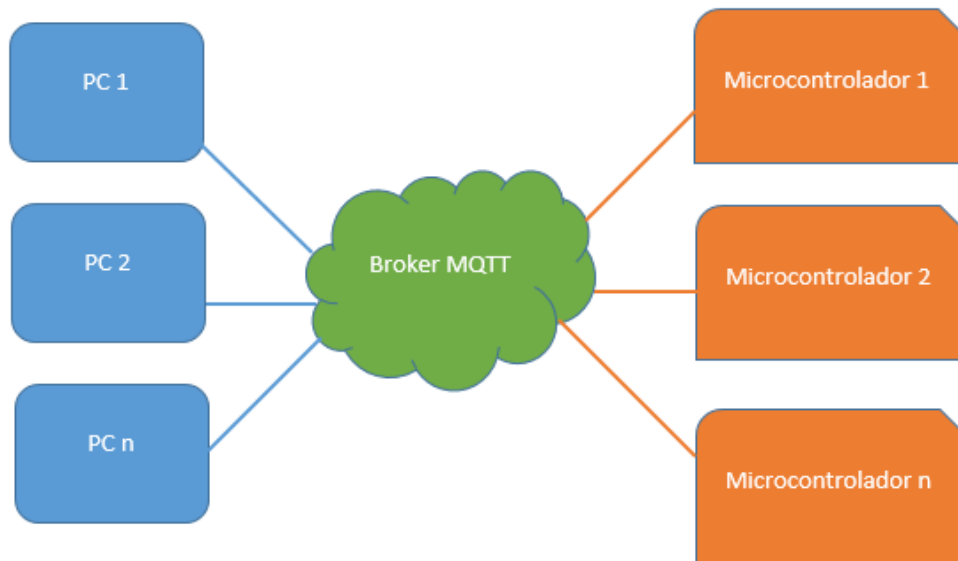


Figura 3.5. Diagrama de bloques de conexión del software.

3.2.1. Capas del software

A continuación, procederemos a describir la estructura de capas para diferenciar los distintos elementos que componen nuestro *software*.

Como podemos observar en la figura 3.6, en nuestro sistema intercalamos varias capas entre el *hardware* y la aplicación como pueden ser los drivers del hardware, el sistema operativo o las librerías usadas para la funcionalidad del proyecto. A continuación, describiremos cada una de las capas y sus relaciones.

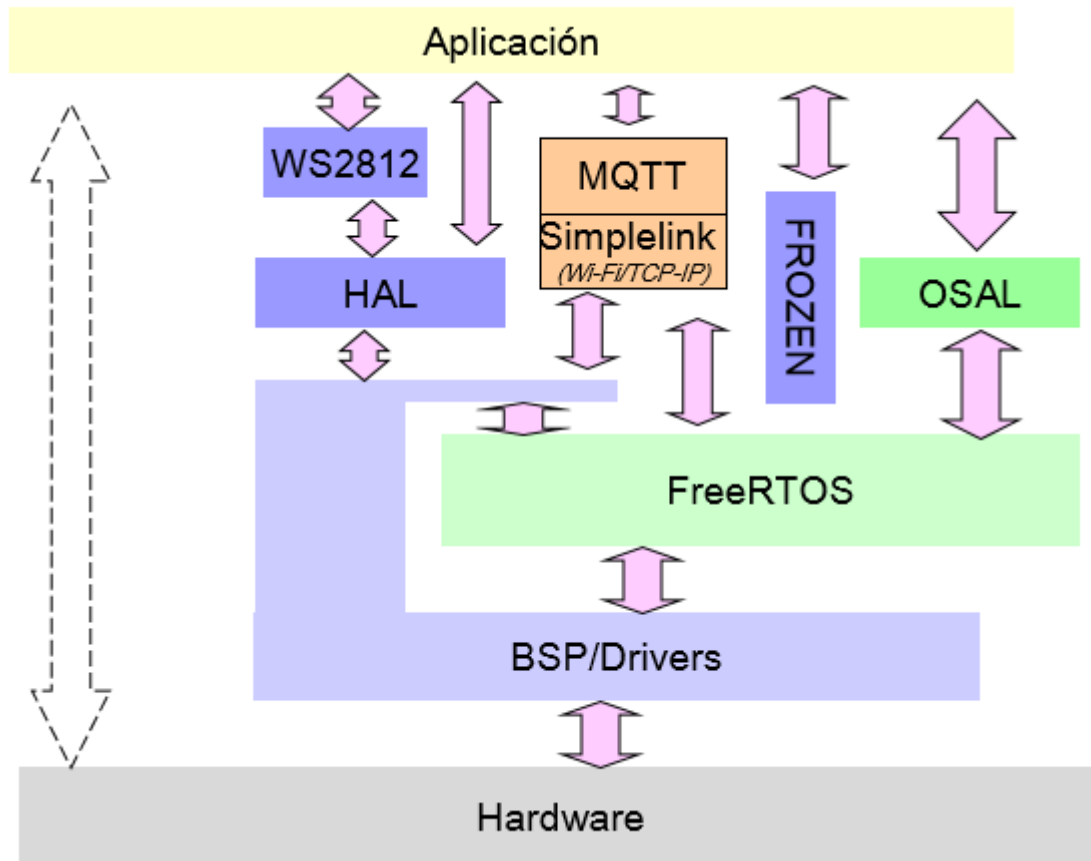


Figura 3.6. Estructura de capas del *firmware*

Hardware

El componente principal de nuestra estructura es la capa del *hardware* la cual se abstrae completamente a través de los *drivers*. Por lo tanto, no podremos acceder directamente al *hardware*, sino que lo haremos a través de las funciones que nos ofrecen los *drivers*.

BSP/Drivers

La siguiente capa es la capa de los *drivers*. Este bloque es el que nos da acceso a la configuración del *hardware* y los periféricos del microcontrolador haciendo más sencilla, por ejemplo, la configuración e implementación del puerto UART (Universal Asynchronous Receiver-Transmitter) o GPIO (General Purpose Input/Output).

FreeRTOS

El bloque FreeRTOS implementa el sistema operativo que se ejecutará en nuestro sistema y que nos dará capacidad para la creación y gestión de tareas, sus mecanismos de comunicación y la introducción a la concurrencia permitiendo ejecutar varias tareas de forma paralela.

Simplelink

La capa Simplelink nos ofrece una serie de funciones desarrolladas por Texas Instruments que nos permiten el uso de la conectividad *Wi-Fi* en el microcontrolador de forma sencilla con configuración a alto nivel.

MQTT

Haciendo uso de la librería MQTT ha sido posible implementar la conexión e interacción con un *broker* MQTT y la suscripción y publicación a diferentes *topics* configurados. Esta capa nos abstrae del protocolo MQTT permitiéndonos la implementación a alto nivel.

Frozen

Esta capa implementa la librería Frozen desarrollada por Cesanta en licencia de código abierto la cual nos permite el tratamiento de los mensajes JSON (JavaScript Object Notation) recibidos o publicados en el servidor MQTT para descomponer y extraer la información deseada.

WS2812

Esta librería, desarrollada por el Departamento de Tecnología Electrónica nos permite controlar el sistema de iluminación compuesto por LED WS2812b a través del bus SPI. Nos ofrece funciones para interactuar directamente con los puntos de luz lo que hace muy sencilla la implementación del control del sistema de iluminación.

De esta librería se hace uso fundamentalmente de dos funciones: `WSGRBtoSPI(uint8_t *pi8SPIData, uint8_t ui8Green, uint8_t ui8Red, uint8_t ui8Blue)` y `InitSPITransfer(uint8_t *pui8SPIData, uint16_t ui16DataSize)`.

La función `WSGRBtoSPI` recibe como parámetros el punto de luz que deseamos modificar y sus valores RGB. Se dispone de una zona de memoria acotada por la matriz `pui8Colors` la cual almacena las coordenadas RGB de todos los puntos de luz de nuestro sistema. Esta función modificará en dicha zona de memoria la posición recibida como parámetro y preparará la información que se enviará por el bus SPI codificando los valores correspondientes a los colores en los pulsos que el driver del LED espera recibir.

A continuación, se detalla el procedimiento de codificación con un ejemplo:

En el caso de que se desee configurar cierto punto de luz con un valor de luz verde de 179 en un rango de 0 a 255 se debería codificar una serie de pulsos atendiendo al siguiente código: 10110011 (179 en sistema binario). Dichos pulsos serán codificados mediante un *nibble* (cuatro bits) cada uno. El código 0 que se corresponde con un pulso corto será codificado con 1000 mientras que el pulso largo que se corresponde con el código 1 será codificado con 1100. Esto hace que la información codificada sea cuatro veces mayor en tamaño que los datos originales. Por lo tanto, para organizar la información correspondiente al valor 179 dispondremos los pulsos en grupos de dos, como se puede observar en la figura 3.7.

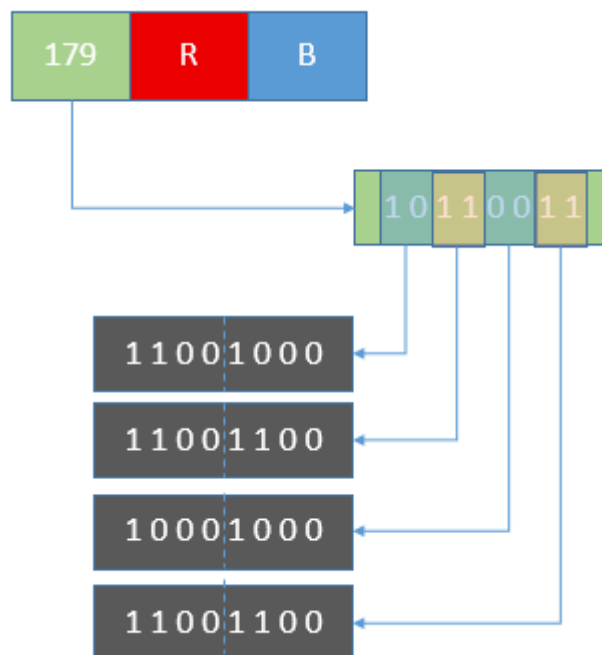


Figura 3.7. Desglose de la codificación de pulsos.

Por otro lado, la función `InitSPITransfer` recibe como parámetros la matriz de LED, cada uno con sus valores RGB, y el tamaño de esta matriz y se encarga de transferir la zona de memoria codificada por la función anterior mediante el protocolo especificado por el fabricante y anteriormente descrito. Esta función accede a la memoria a través del controlador DMA (Direct Memory Access) y transmite el bloque de memoria organizado por la función a través de SPI. La gestión de la transferencia SPI se lleva a cabo mediante interrupción asociada al DMA en lugar de interrupción de la CPU. Involucrar a la CPU para leer estos datos implicaría no cumplir estos requisitos y que la comunicación no fuera satisfactoria. Debido a la forma de codificación de los pulsos, cada transferencia SPI llevará a cabo el envío de dos bits de datos (generando dos pulsos) ya que cada pulso corresponde a cuatro bits y la transferencia SPI a ocho bits.

En la figura 3.8 podemos observar la estructura de archivos que compone nuestro proyecto dónde podemos diferenciar las diferentes capas de *software* que contiene y librerías de las que se hace uso.

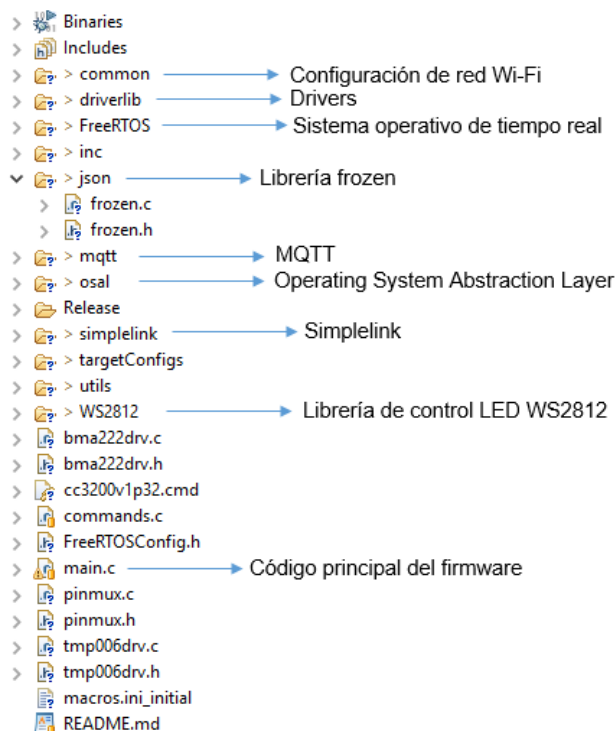


Figura 3.8. Estructura de archivos del proyecto.

3.2.2. FreeRTOS

A continuación, detallaremos las características principales de FreeRTOS, un sistema operativo de tiempo real de código abierto, el cual hemos elegido para que sea ejecutado en nuestro sistema.

Comenzaremos describiendo qué es un sistema operativo de tiempo real.

Para empezar, debemos mencionar que los sistemas empujados tienen requisitos de tiempo real por lo tanto se hace necesario el uso de un sistema operativo de tiempo real que incluya un planificador de sistema determinista. Esto es, el comportamiento del mismo es predecible. FreeRTOS estructura el *software* en forma de tareas (hilos de ejecución) las cuales serán ejecutadas de forma pseudoparalela y nos permite asignar una prioridad a cada tarea para que el planificador pueda saber qué hilo debe ejecutar en cada momento.

Cuando hablamos de que las tareas serán ejecutadas de forma pseudoparalela nos referimos a que un procesador de un único núcleo no tiene capacidad para aplicar concurrencia. Por ello el sistema operativo de tiempo real reparte los recursos del sistema entre las distintas tareas asignando cierto tiempo a cada una consiguiendo un efecto de paralelismo al cambiar rápidamente de una tarea a otra.

A continuación, nombraremos las principales características de FreeRTOS.

Este sistema operativo está diseñado concretamente para ser implementado en microcontroladores por lo que no requiere de grandes recursos lo que lo hace ideal en nuestro caso. FreeRTOS proporciona varios sistemas de sincronización de tareas como son los semáforos, *mutex* o colas de mensajes. En nuestro caso haremos uso de las colas de mensajes como sistema de comunicación entre algunas de nuestras tareas. [9].

3.2.3. MQTT

La comunicación entre nuestro dispositivo microcontrolador y la aplicación de usuario se hará a través de un *middleware* como es el protocolo de comunicación MQTT. Este es un protocolo de mensajería “publish/suscribe”, muy simple y ligero, diseñado para dispositivos con limitaciones y conexiones

de bajo ancho de banda, alta latencia o no confiables. Los objetivos de este protocolo son minimizar el ancho de banda y los recursos del dispositivo requeridos además de asegurar seguridad en la comunicación. Esto lo hace ideal para las comunicaciones M2M (*Machine to Machine*) y para aplicaciones móviles donde el ancho de banda y el consumo de energía son un factor muy importante.

Desde marzo de 2013, MQTT está en proceso de estandarización en OASIS (*Organization for the Advancement of Structured Information Standards*). La especificación del protocolo se ha publicado como licencia libre de *royalties* [10].

3.2.4. JSON

Los mensajes que recibiremos y publicaremos a través del *middleware* antes mencionado se encontrarán en formato JSON. Este es un formato de texto ligero para el intercambio de datos y fácil de analizar y generar para las máquinas. Entre sus principales características se encuentra que es completamente independiente del lenguaje del *software* y en su sintaxis usa pares no ordenados de nombre/valor.

Para analizar y generar este formato de texto de forma simple usaremos una librería de código abierto en C (*frozen.h*) desarrollada por Cesanta Software para sistemas embebidos la cual usa *scanf* y *printf* como interfaz [11].

3.2.5. Funciones y tareas del microcontrolador

En este apartado describiremos las distintas funciones que se implementan en el firmware de nuestro dispositivo y las tareas que se ejecutarán sobre el sistema operativo de tiempo real.

Podemos observar en la figura 3.9 un esquema acerca del orden de ejecución de dichas tareas. Podemos agrupar en un primer bloque las tareas que se encargan de la configuración del microcontrolador y posteriormente un bloque que lleva a cabo la conexión a Internet y otro para la conexión al servidor MQTT. Por último, tras realizarse con éxito la conexión al *broker* contemplamos un bloque que realiza la suscripción al *topic* por defecto dedicado a la

configuración y otro bloque que espera recibir algún mensaje por los *topics* suscritos. A partir de este bloque consideramos varios grupos los cuales detectarán en qué *topic* hemos recibido el mensaje y llevarán a cabo la actuación pertinente.

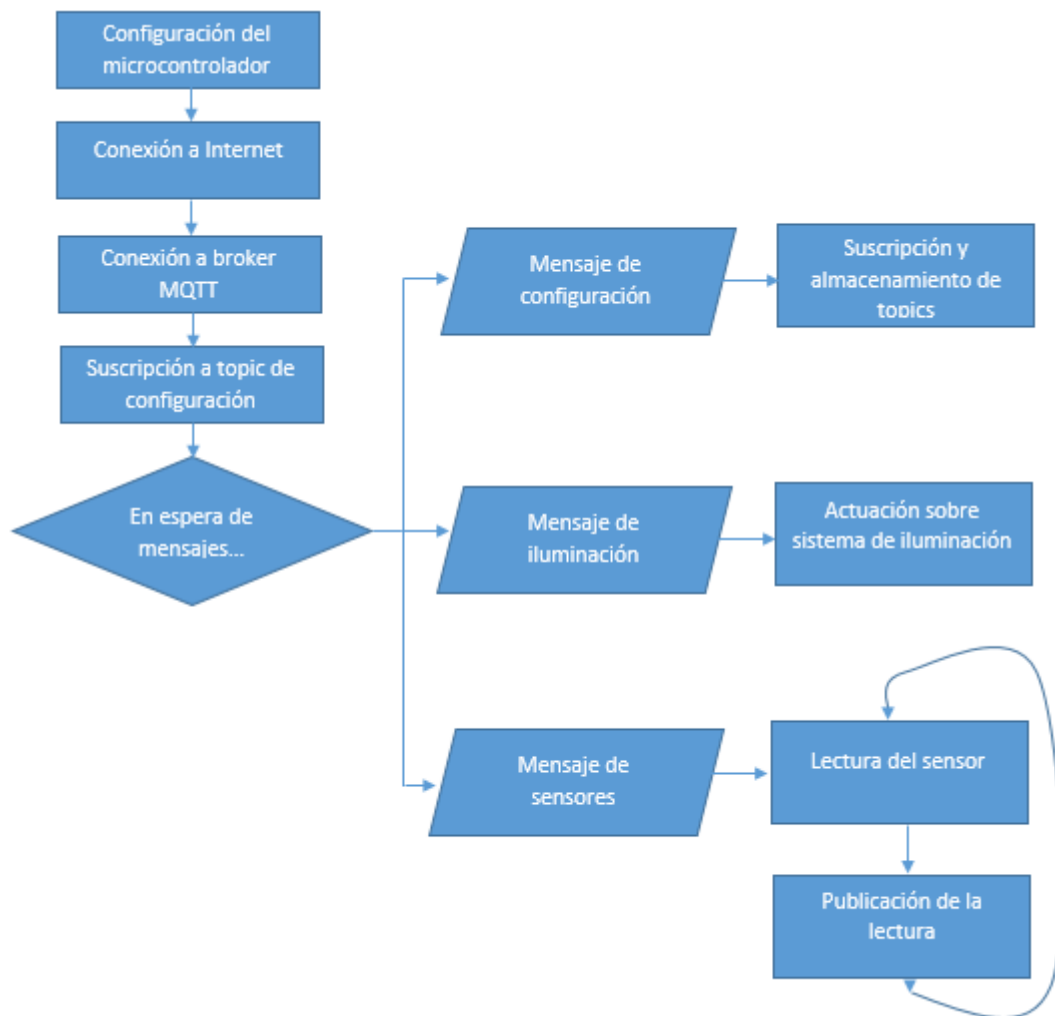


Figura 3.9. Flujograma del firmware del microcontrolador.

En primer lugar, será descrita la función **BoardInit** la cual al comienzo de la ejecución del *firmware* configurará nuestro microcontrolador, sus modos de bajo consumo y los periféricos.

A continuación, detallaremos en qué consiste la función **ConnectWifi** la cual ejecutaremos desde la función principal **main** como una tarea paralela. En esta función se configurará la conexión a la red *Wi-Fi* o se creará el punto de acceso (según se haya configurado el modo de funcionamiento en “Modo Station” o “Modo AP” con la etiqueta `#define STATION_MODE 1` o la ausencia de la misma.

Dichos modos serán detallados en el manual de usuario) parametrizados en el archivo `common.h`. En el proceso de conexión se mostrará información de depuración a través del puerto serie.

En esta función, tras el éxito de la conexión inicializaremos los sensores de temperatura y aceleración y configuraremos los pines correspondientes como I²C para habilitar las posteriores lecturas. Posteriormente, se lleva a cabo la conexión al *broker* MQTT y la suscripción al *topic* de configuración. En este momento la tarea quedará a la espera en un bucle infinito de la recepción de algún mensaje por la cola de mensajes que usaremos para la comunicación entre nuestras tareas.

En el momento en el que se lleva a cabo la conexión al *broker* MQTT se ha configurado como *callback* de recepción de mensajes la función **Mqtt_Recv**.

Cuando se recibe un mensaje por cualquiera de los *topics* a los que el microcontrolador se ha suscrito, esta función, en primer lugar, comprueba si el *topic* de procedencia del mensaje que la ha activado corresponde con alguno de los *topics* de los usados en la aplicación y dependiendo del *topic* en el que se haya escrito un mensaje se harán diferentes comprobaciones o acciones.

Si se ha recibido el mensaje de configuración de *topics* inicial por el *topic* adecuado, se procederá a recabar la información incluida en el mensaje la cual se espera que tenga el siguiente formato: "{ Leds: %Q Sensors: %Q Temp: %Q Acc: %Q }" donde %Q denota una cadena de caracteres correspondiente al *topic* a la que habrá que añadir la raíz "/cc3200/". Dicha información se almacenará en variables locales y se pasará como parámetro a una tarea, la cual será creada con el fin de llevar a cabo la suscripción de los *topics* correspondientes y dar el formato correcto a los *topics* de publicación.

Esta tarea, llamada `subTask` definirá, para cada caso (*topic* de control de iluminación y *topic* de configuración de sensores) una cadena de caracteres incluyendo la raíz de los *topics* (/cc3200/) a la cual le añadirá al final el *topic* recibido que corresponda. Posteriormente llevará a cabo la suscripción a dicho *topic* mediante la función `sl_ExtLib_MqttClientSub` disponible en la librería de MQTT. Para configurar los *topics* de publicación dedicados al envío de las lecturas de los sensores (*topic* de publicación de medidas de temperatura y

topic de publicación de medidas de aceleración) simplemente se almacenará en una variable global la información que corresponda recibida por el *topic* de configuración precedida de la raíz de los *topics* (/cc3200/).

En el caso de haber recibido un mensaje por el *topic* dedicado al control del sistema de iluminación comprobaremos si el mensaje es correcto haciendo el intento de captar la información con el siguiente formato: "{ LED:%d R:%d G:%d B:%d }". De esta forma se espera recibir en cada valor numérico, denotado por %d, el número de LED que deseamos configurar y la composición de colores rojo, verde y azul respectivamente. Posteriormente se lleva a cabo la modificación del punto de luz mediante las funciones `WSGRBtoSPI` y `InitSPITransfer` disponibles en la librería WS2812. En el caso de no recibir la información en el formato correcto no se llevará a cabo ninguna acción.

Por otro lado, al recibir un mensaje por el *topic* de alguno de los sensores, ya sea de temperatura o de aceleración, con el formato "{ TEMP: %d }" o "{ ACC: %d }" respectivamente donde %d representa el tiempo de refresco en milisegundos se procederá de la siguiente manera:

Si el tiempo de refresco es mayor que cero se crea una tarea a la cual se le pasa como parámetro dicho tiempo de actualización. Esta tarea se encargará de gestionar la lectura periódica del sensor.

En el caso de recibir cero como tiempo de refresco se está indicando el deseo de detener la lectura del sensor por lo que se procederá a eliminar la tarea que gestiona dicha lectura.

Dicha tarea (`TempTask` en el caso del sensor de temperatura y `AccTask` en el caso del sensor de aceleración) entrará en un bucle infinito en el que se enviará cada cierto tiempo (definido por el parámetro de refresco que recibe esta tarea como entrada) un mensaje por la cola de mensajes (`READ_TEMP` en el caso del sensor de temperatura y `READ_ACC` en el caso del sensor de aceleración) el cual será recibido en la función **ConnectWifi** donde se llevará a cabo la lectura correspondiente y la publicación de la medida. Para implementar la espera hasta el envío del siguiente mensaje (refresco de la medida) se introduce una suspensión de la tarea gestionada por el sistema

operativo mediante `osi_Sleep`. Esta función recibe como parámetro el tiempo en milisegundos que el sistema operativo suspenderá la tarea provocando que no se vuelva a enviar un nuevo mensaje por la cola de mensajes (lo que conlleva la lectura y publicación de la medida) hasta que no transcurra el tiempo de actualización configurado.

En el caso de recibir la orden de lectura y publicación de la medida de alguno de los sensores procedente de la tarea correspondiente (en concreto esperaremos la recepción de los mensajes `READ_TEMP` o `READ_ACC`) llevaremos a cabo la lectura del sensor correspondiente con `TMP006DrvGetTemp(&temp)` o `BMA222ReadNew(&accX, &accY, &accZ)`.

`TMP006DrvGetTemp(&temp)` es una función que encontramos disponible en la librería `TMP006drv`. Esta función lee de los registros del sensor de temperatura los valores de temperatura ambiente y temperatura objeto y hace el cálculo de temperatura devolviéndolo por referencia.

Por otro lado, `BMA222ReadNew(&accX, &accY, &accZ)`, lee del registro del sensor de aceleración los valores correspondientes a los tres ejes y devuelve por referencia aquellos que hayan sufrido cambios desde la última lectura.

Posteriormente generaremos un mensaje en formato JSON con la estructura "{
`Temperature : %f }`" en el caso de la temperatura o "{
`AccX : %d, AccY: %d, AccZ : %d }`" y llevaremos a cabo la publicación en el *topic* configurado.

3.3. Aplicación gráfica del PC

En este apartado se describirá el desarrollo de una aplicación gráfica de prueba que será de utilidad para comprobar el correcto funcionamiento del *software* embebido en el microcontrolador de una forma cómoda e intuitiva. En esta aplicación se podrá llevar a cabo la configuración inicial de los *topics*, gestionar el sistema de iluminación, configurar la medida de los sensores y representar las lecturas de forma gráfica.

Se procederá a detallar las acciones que el usuario podrá llevar a cabo, interactuando con la interfaz describiendo como se ha implementado la solución.

En las figuras 3.10, 3.11, 3.12 y 3.13 podemos observar el aspecto de la interfaz gráfica y marcados en colores los diferentes elementos con los que el usuario puede interactuar.

En la primera pestaña de la aplicación (figura 3.10), dedicada a la conexión con el servidor MQTT y la configuración inicial de los *topics*, el usuario podrá iniciar la conexión con el *broker* MQTT y enviar la información de configuración de los *topics*:

Inicio de la conexión MQTT

En este caso se ha implementado una respuesta al pulsado del botón **Inicio** (marcado en rojo) mediante la función `on_runButton_clicked()` la cual será ejecutada al pulsar sobre el botón. Esta función implementa la conexión al servidor MQTT dado en el cuadro de texto **MQTT Broker** (marcado en rojo) y la suscripción a los *topics* dados en los cuadros de texto **Topic Temperature** y **Topic Acceleration** (precedidos de la raíz de los *topics* “/cc3200/”). Se incluye un indicador que nos ofrecerá información sobre la conexión al servidor y la suscripción a los *topics* indicados (marcado en azul).

Información de configuración de topics

Para implementar el envío de la información de la configuración de los *topics* al microcontrolador se ha incluido el botón **Set Config** (marcado en verde) y una función de respuesta asignada al pulsado de dicho botón `on_pushButton_3_clicked()`. Esta función crea un objeto JSON con los nombres **LEDS**, **SENSORS**, **TEMP** y **ACC** y los valores dados en los cuadros de texto **Topic LEDs**, **Topic Sensors**, **Topic Temperature** y **Topic Acceleration** respectivamente y lo envía a través del *topic* dado en **Topic Config** (precedido de la raíz).

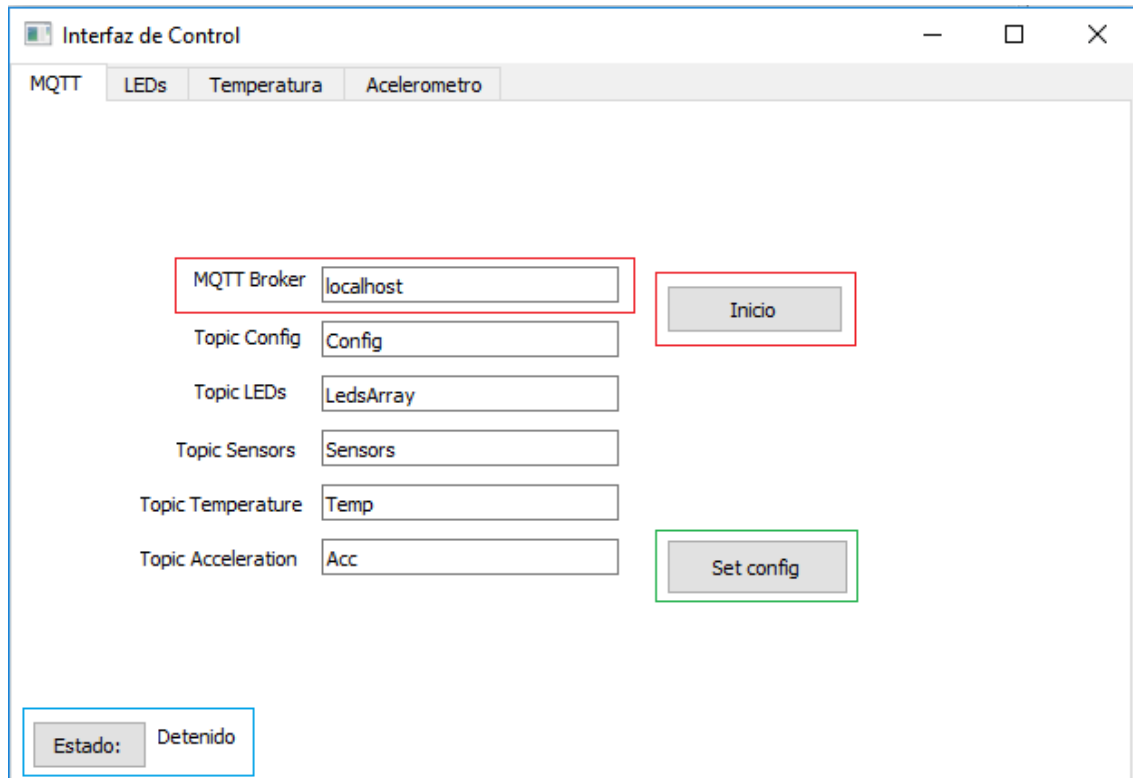


Figura 3.10. Vista de la pestaña de conexión MQTT de la interfaz.

Control del sistema de iluminación

En la segunda pestaña (figura 3.11) se encuentran los controles necesarios para el control del sistema de iluminación. Para ello disponemos de un control en forma de ruleta (marcado en rojo) que nos permite escoger un color de forma visual y codificarlo en formato RGB. Se incluye además un control (marcado en azul) con el que podremos seleccionar qué punto de luz queremos configurar. Dicho control permitirá seleccionar valores desde 0 hasta N-1 (donde N es el número de LED que contiene nuestro sistema). En nuestro caso podremos seleccionar desde 0 hasta 15 ya que nuestro sistema de iluminación dispone de 16 puntos de luz. Una vez configurado tanto el color como el punto de luz a configurar en los controles correspondientes podremos llevar a cabo la configuración mediante el botón **Actualizar LEDs** el cual tiene asociada una función de respuesta al clic (`on_pushButton_4_released()`). Esta función crea un objeto JSON con los nombres LED, R, G y B asociando al nombre LED el valor escogido en el control marcado en azul y a los nombres R, G y B los valores correspondientes al color elegido en la ruleta (marcada en rojo) en formato RGB. Tras componer un mensaje con el objeto JSON creado,

este es publicado en el *topic* configurado en el cuadro de texto **Topic LEDs** de la pestaña de configuración (precedido de la raíz).

En esta pestaña también se incluye un botón para dar la capacidad de apagar todos los puntos de luz sin la necesidad de recorrerlos uno a uno con el selector de LED (marcado en azul). Para ello se incluye el botón **Apagar LEDs** el cual tiene asociada la función `on_pushButton_5_released()` como respuesta al clic. Esta función hace un recorrido por todos los puntos de luz (limitado por el valor máximo configurado del selector de LED) creando un objeto JSON por cada punto de luz con los valores R, G y B a cero y enviando un mensaje por el *topic* configurado por cada punto de luz.

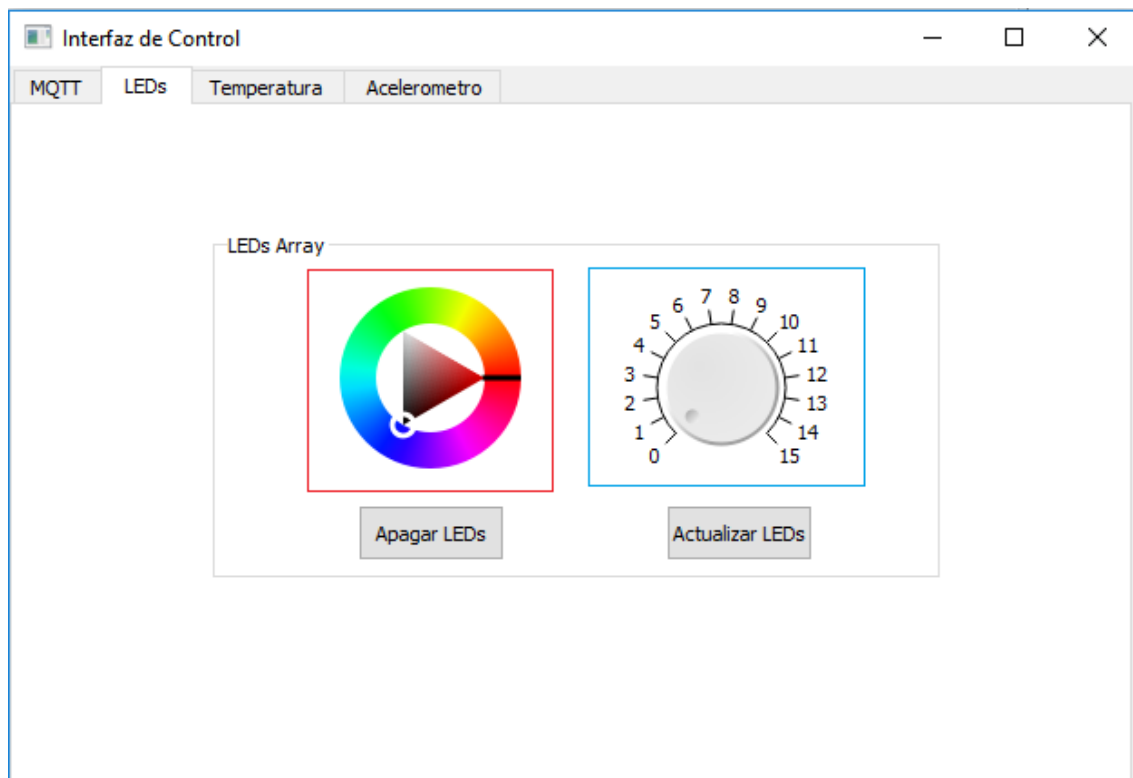


Figura 3.11. Vista de la pestaña de control de iluminación de la interfaz.

Configuración de sensores

En las pestañas 3 y 4 (figuras 3.12 y 3.13 respectivamente) se incluyen los elementos necesarios para enviar las órdenes de configuración de los sensores al microcontrolador. Para ello disponemos de un control numérico (marcado en rojo) en el cual debemos seleccionar el tiempo de actualización de la medida en milisegundos y un botón **ON** para enviar la orden. Este botón tiene asociada una función de respuesta `on_run_temp_released()` para el caso de la

temperatura y `on_run_acc_released()` en el caso de la aceleración. Ambas tienen el mismo comportamiento. Se crea un objeto JSON con el nombre TEMP o ACC según corresponda y como valor el configurado en el control numérico. Tras esto se compone el mensaje y se envía al *topic* configurado en el cuadro de texto **Topic Sensors** en la pestaña de configuración (precedido de la raíz).

Visualización de las medidas

Tras configurar la activación de los sensores, el dispositivo microcontrolador comenzará a enviar de forma periódica las medidas por el *topic* configurado en la pestaña de configuración (**Topic Temperature** en el caso de la temperatura y **Topic Acceleration** en el caso de la aceleración). En la aplicación gráfica recibiremos dichos mensajes mediante la función `onMQTT_Received(const QMQTT::Message &message)` y realizaremos el procesado adecuado para representar las lecturas de los sensores.

En primer lugar, debemos distinguir si el mensaje que hemos recibido corresponde a una lectura de temperatura o de aceleración. Para ello accederemos al campo “topic” del mensaje recibido mediante `message.topic()`. Tras esta comprobación accederemos a la entrada “Temperature” o “AccX”, “AccY” y “AccZ” según corresponda para recoger los valores medidos en el microcontrolador. Una vez recogidas las lecturas se procede a actualizar tanto los indicadores numéricos como la gráfica para mostrar las medidas.

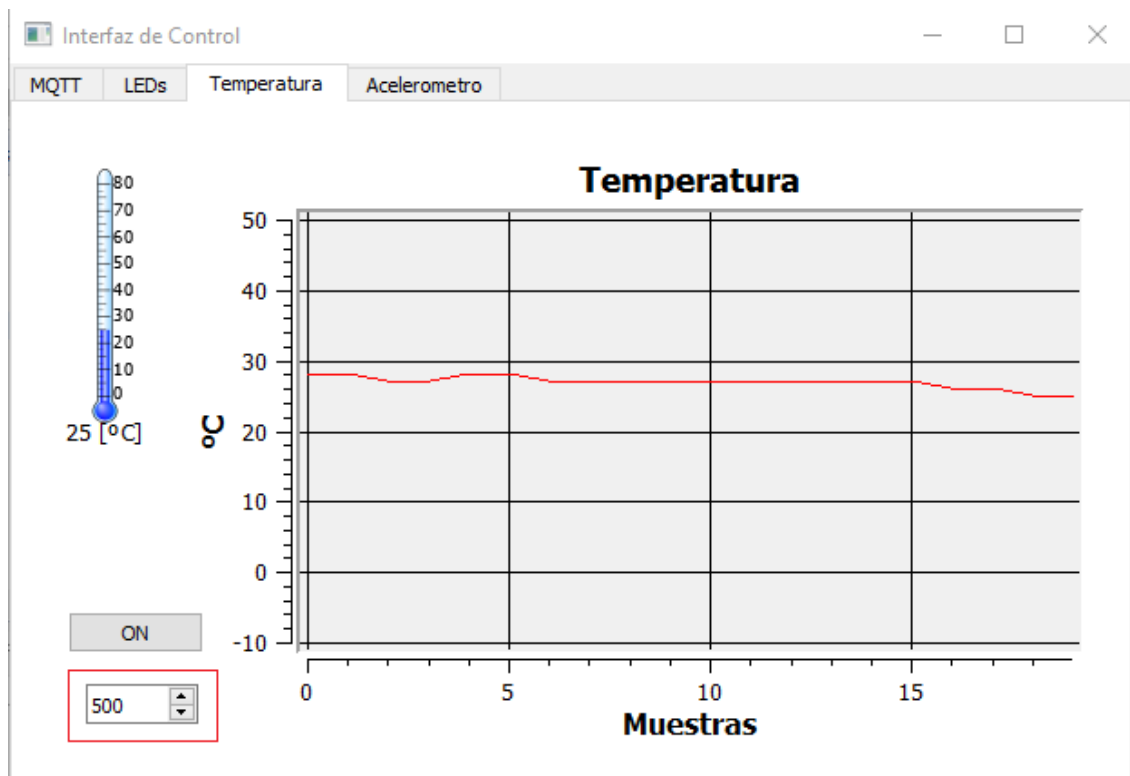


Figura 3.12. Vista de la pestaña de información de temperatura de la interfaz.

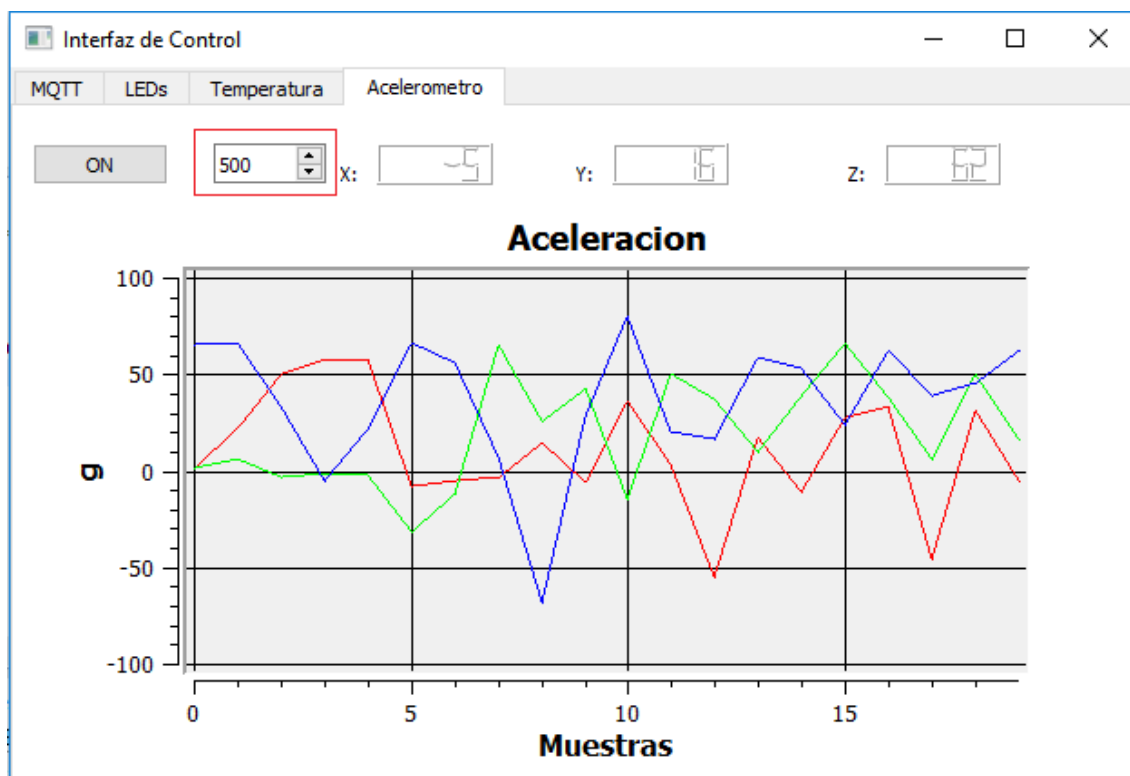


Figura 3.13. Vista de la pestaña de información de aceleración de la interfaz.

Capítulo 4. Pruebas y verificación

En este capítulo procederemos a describir las diferentes pruebas que hemos realizado para verificar el correcto funcionamiento del sistema y el cumplimiento de los requisitos anteriormente detallados. Para ello se ha usado un PC en el que hemos ejecutado la aplicación de usuario y un servidor MQTT. Destacar que todas las pruebas han finalizado con resultado favorable y que el sistema ha sido expuesto a ellas durante un tiempo considerable.

ID de la prueba	ID del requisito	Descripción	Resultado
1	1.1, 1.3, 1.4, 3	En esta prueba se ha configurado la aplicación del microcontrolador en modo AP (Access Point). Tras el arranque de la aplicación comprobamos con un PC que el microcontrolador ha creado la red <i>Wi-Fi</i> y que podemos conectarnos mediante la clave configurada. Observamos que el microcontrolador nos ofrece información de depuración en todo momento del estado de la creación de la red hasta que esta se lleva a cabo con éxito a través del puerto serie. Tras la conexión del PC a la red <i>Wi-Fi</i> comprobamos mediante la depuración por el puerto serie que el microcontrolador se ha conectado correctamente al servidor MQTT y la suscripción al <i>topic</i> de configuración se ha llevado a cabo con éxito.	Favorable
2	1.2, 1.3, 1.4, 3	En esta prueba se ha configurado la aplicación del microcontrolador en modo STATION. Tras el arranque de la aplicación observamos la información de depuración que nos ofrece el microcontrolador sobre el estado de la conexión a la red <i>Wi-Fi</i> configurada. Comprobamos mediante la depuración por el puerto serie que el microcontrolador se ha conectado correctamente a la red <i>Wi-Fi</i> y al servidor MQTT y la suscripción al <i>topic</i> de configuración se ha llevado a cabo con éxito.	Favorable

ID de la prueba	ID del requisito	Descripción	Resultado
3	4, 4.1, 4.2, 9, 10, 10.1, 10.2	<p>En esta prueba se ha ejecutado la aplicación de usuario y se ha llevado a cabo la conexión con el servidor MQTT comprobando mediante el indicador de estado de la interfaz que la conexión con el servidor y la suscripción de los <i>topics</i> correspondientes se ha llevado a cabo satisfactoriamente. Posteriormente se ha establecido la configuración de los <i>topics</i> y se ha enviado la orden al microcontrolador.</p> <p>Se ha comprobado mediante la depuración por el puerto serie que el microcontrolador ha recibido correctamente la información de configuración y se ha suscrito correctamente a los <i>topics</i> correspondientes.</p>	Favorable
4	5, 5.1, 5.2, 11, 11.1, 11.2	<p>Tras llevar a cabo con éxito la conexión al servidor MQTT y la suscripción a los <i>topics</i> correspondientes se procede a probar el control sobre el sistema de iluminación. Se comprueba en la interfaz gráfica que es posible seleccionar cualquier punto de luz del sistema de iluminación y cualquier color deseado mediante controles visuales. También es posible apagar por completo el sistema de iluminación con la pulsación de un solo botón.</p> <p>Comprobamos que el microcontrolador actúa de forma adecuada sobre el sistema de iluminación tras el envío de órdenes desde la interfaz.</p>	Favorable

ID de la prueba	ID del requisito	Descripción	Resultado
5	2, 6, 6.1, 6.2, 12, 12.1, 12.2, 12.3, 12.4, 12.5	<p>Comprobamos visualmente, en la interfaz gráfica, que se incluyen los controles necesarios para la gestión de los sensores: un control para la activación, un control numérico para indicar el período de refresco de la lectura, un control para la desactivación de las medidas, un indicador gráfico para la última lectura y una gráfica para el registro de las medidas.</p> <p>Tras realizar varias peticiones sobre la activación, modificación de período y desactivación de la medida, comprobamos que el microcontrolador actúa de forma satisfactoria y la interfaz tiene la capacidad de representar de forma visual las lecturas.</p>	Favorable
6	7	<p>Para comprobar que el sistema atiende más de una petición de forma simultanea hemos activado la lectura de los dos sensores comprobando así que el microcontrolador es capaz de gestionar varias tareas periódicas correctamente además de seguir atendiendo peticiones como el control del sistema de iluminación de forma satisfactoria.</p>	Favorable
7	8	<p>Tras configurar el sistema y comprobar su correcto funcionamiento hemos enviado una nueva petición de configuración de <i>topics</i>. Comprobamos que el sistema sigue funcionando correctamente con la nueva configuración.</p>	Favorable

4.1. Resultados

En esta sección se mostrarán capturas tanto de la información de depuración como de la interfaz gráfica tomadas durante la realización de las pruebas anteriormente descritas.

Prueba 1

Podemos comprobar en la figura 4.1 como se ha llevado la creación del punto de acceso correctamente mediante la información de depuración enviada por el puerto serie. También se puede apreciar el éxito de la conexión al servidor MQTT y la suscripción al *topic* de configuración.

```

Host Driver Version: 1.0.0.10
Build Version 2.0.7.0.31.0.0.4.1.1.5.3.3
Device is configured in default state
Started SimpleLink Device: STA Mode
Switching to AP mode on application request
[NETAPP EVENT] IP acquired by the device
Re-started SimpleLink Device: AP Mode
[NETAPP EVENT] IP acquired by the device
Manufacturer ID: 0x5449
Device ID: 0x67
Configuration register value: 0x7480
CHIP ID: 0xf8

*****
CC3200 MQTT_Client Application
*****

FreeRTOS V8.2.3

Teclee ? para ver la ayuda
> [NETAPP EVENT] IP leased to a client
mqtt on
Inicio la tarea MQTT
> Version: Client LIB 1.0.3, Common LIB 1.1.1.
C: FH-B1 0x10 to net 17, Sent (45 Bytes) [0 394]
C: Rcvd msg Fix-Hdr (Bytel) 0x20 from net 17 [0 394]
C: Cleaning session for net 17
C: Msg w/ ID 0x0000, processing status: Good

Success: conn to Broker no. 1
C: FH-B1 0x82 to net 17, Sent (21 Bytes) [0 394]
C: Rcvd msg Fix-Hdr (Bytel) 0x90 from net 17 [0 394]
C: Msg w/ ID 0x0001, processing status: Good
Client subscribed on following topics:
/cc3200/Config

```

Figura 4.1. Información de depuración en modo AP.

Prueba 2

Como se representa en la figura 4.2 la conexión a la red Wi-Fi configurada ha sido satisfactoria, así como la conexión al servidor MQTT y la suscripción al *topic* de configuración.



```

Host Driver Version: 1.0.0.10
Build Version 2.0.7.0.31.0.0.4.1.1.5.3.3
Device is configured in default state
Started SimpleLink Device: STA Mode
[WLAN EVENT] STA Connected to the AP: mi_wifi11 , BSSID: 38:60:77:5b:40:6e
[NETAPP EVENT] IP acquired by the device

Device has connected to mi_wifi11
Device IP Address is 192.168.1.19

Manufacturer ID: 0x5449
Device ID: 0x67
Configuration register value: 0x7480
CHIP ID: 0xf8

*****
CC3200 MQTT_Client Application
*****

Version: Client LIB 1.0.3, Common LIB 1.1.1.

FreeRTOS V8.2.3

Teclee ? para ver la ayuda
> C: FH-B1 0x10 to net 17, Sent (45 Bytes) [0 566]
C: Rcvd msg Fix-Hdr (Byte1) 0x20 from net 17 [0 566]

Success: conn to Broker no. 1
C: FH-B1 0x82 to net 17, Sent (21 Bytes) [0 566]
C: Rcvd msg Fix-Hdr (Byte1) 0x90 from net 17 [0 566]
Client subscribed on following topics:
/cc3200/Config
  
```

Figura 4.2. Información de depuración en modo STATION.

Prueba 3

En esta prueba se puede contemplar como la interfaz gráfica nos ofrece información del éxito de la conexión al *broker* MQTT y la suscripción a los *topics* correspondientes mediante la figura 4.3.

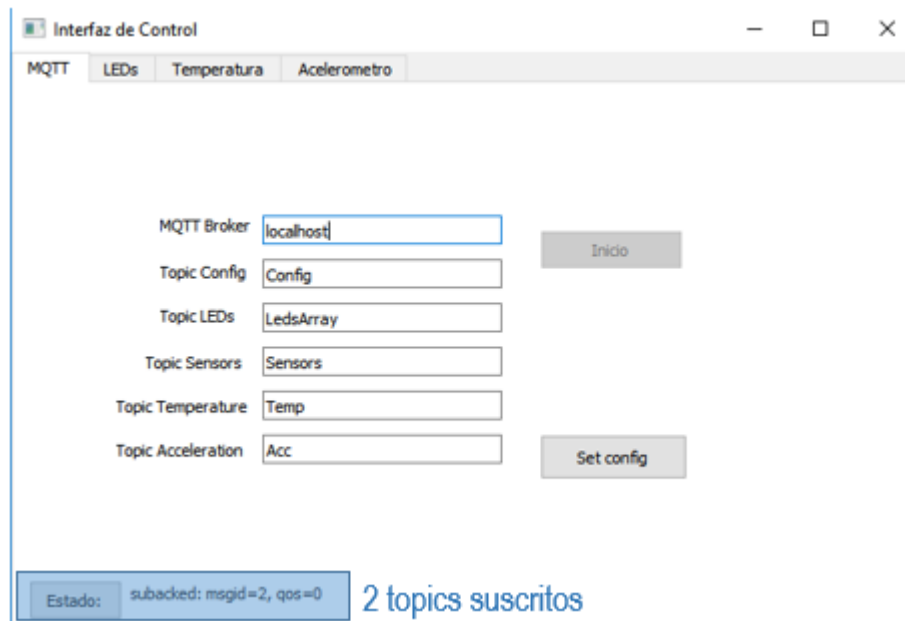


Figura 4.3. Pestaña de conexión MQTT de la interfaz gráfica.

Tras enviar la información de configuración desde la aplicación gráfica se observa en la figura 4.4 la información recibida en el microcontrolador.

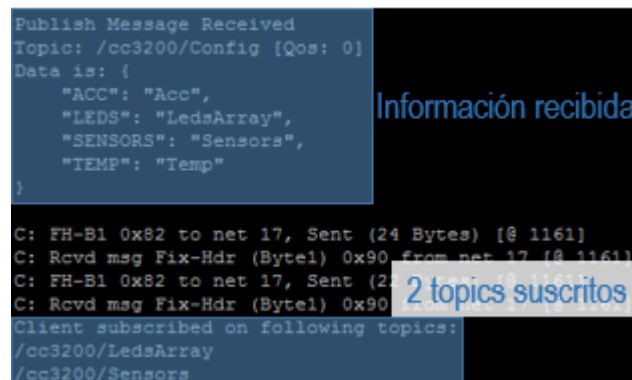


Figura 4.4. Información de depuración tras configuración de *topics*.

Prueba 4

Se puede comprobar en la figura 4.5 la información que se recibe en el microcontrolador tras modificar el color de un punto de luz en la interfaz gráfica.



Figura 4.5. Información de depuración tras modificación de un punto de luz.

En la figura 4.6 se puede observar la información que se transmite desde el microcontrolador al sistema de iluminación para encender un punto de luz en azul, rojo o verde respectivamente. Para ello se envían ocho pulsos con código uno los cuales representan un valor de 255.

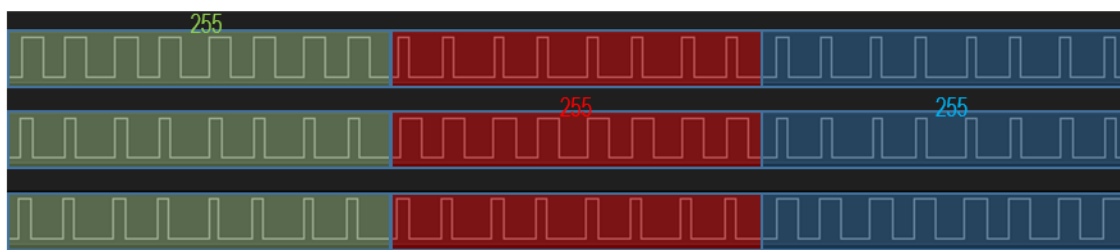


Figura 4.6. Muestra del analizador lógico en transmisiones para control de luces.

En la figura 4.7 se ilustra la información recibida tras enviar la orden de apagar el sistema de iluminación.

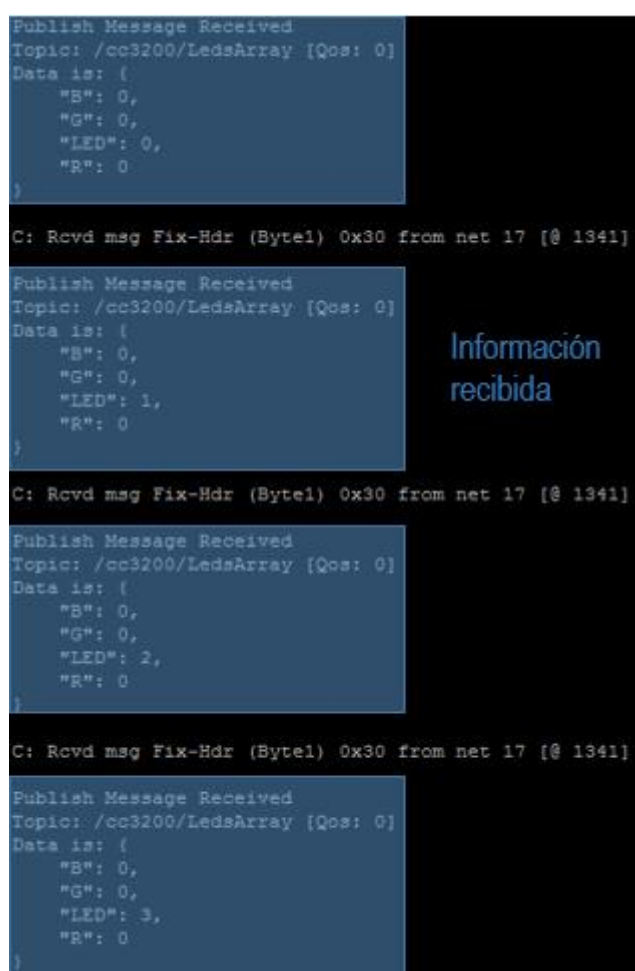


Figura 4.7. Información recibida tras petición de apagado de sistema de iluminación.

Prueba 5

Se puede observar en las figuras 4.8 y 4.9 la información recibida en el microcontrolador correspondiente a la configuración de lectura del sensor de temperatura y aceleración respectivamente y las posteriores publicaciones de la medida.

```

Publish Message Received
Topic: /cc3200/Sensors [Qos: 0]
Data is: {
  "TEMP": 500
}

C: FH-B1 0x35 to net 17, Sent (47 Bytes) [@ 1434]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1434]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1434]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1434]

CC3200 Publishes the following message
Topic: /cc3200/Temp
C: FH-B1 0x35 to net 17, Sent (47 Bytes) [@ 1434]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1434]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1434]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1434]

CC3200 Publishes the following message
Topic: /cc3200/Temp
C: FH-B1 0x35 to net 17, Sent (47 Bytes) [@ 1435]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1435]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1435]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1435]

CC3200 Publishes the following message
Topic: /cc3200/Temp
C: FH-B1 0x35 to net 17, Sent (47 Bytes) [@ 1435]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1435]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1435]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1435]

CC3200 Publishes the following message
Topic: /cc3200/Temp
C: FH-B1 0x35 to net 17, Sent (47 Bytes) [@ 1436]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1436]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1436]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1436]

```

Figura 4.8. Información recibida tras configurar el sensor de temperatura.

```

Publish Message Received
Topic: /cc3200/Sensors [Qos: 0]
Data is: {
  "ACC": 500
}

C: FH-B1 0x35 to net 17, Sent (55 Bytes) [@ 1509]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1509]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1509]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1510]

CC3200 Publishes the following message
Topic: /cc3200/Acc
C: FH-B1 0x35 to net 17, Sent (55 Bytes) [@ 1510]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1510]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1510]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1510]

CC3200 Publishes the following message
Topic: /cc3200/Acc
C: FH-B1 0x35 to net 17, Sent (55 Bytes) [@ 1511]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1511]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1511]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1511]

CC3200 Publishes the following message
Topic: /cc3200/Acc
C: FH-B1 0x35 to net 17, Sent (55 Bytes) [@ 1511]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1511]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1511]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1511]

CC3200 Publishes the following message
Topic: /cc3200/Acc
C: FH-B1 0x35 to net 17, Sent (55 Bytes) [@ 1511]
C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [@ 1512]
C: FH-B1 0x62 to net 17, Sent (4 Bytes) [@ 1512]
C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [@ 1512]

```

Figura 4.9. Información recibida tras la configuración del sensor de aceleración.

Prueba 6

Se puede comprobar en la figura 4.10 la información que se recibe al configurar de forma simultánea los dos sensores. Posteriormente en la figura 4.11 se muestra la capacidad del sistema para llevar a cabo la publicación de las medidas de ambos sensores de forma simultánea, así como la recepción de información de otras ordenes como el control del sistema de iluminación.

<pre>Publish Message Received Topic: /cc3200/Sensors [Qos: 0] Data is: { "TEMP": 1000 }</pre>	Envío de configuración
<pre>C: FH-B1 0x35 to net 17, Sent (47 Bytes) [0 1556] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1556] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1556] C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [0 1556] CC3200 Publishes the following message Topic: /cc3200/Temp C: FH-B1 0x35 to net 17, Sent (47 Bytes) [0 1557] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1557] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1557] C: Rcvd msg Fix-Hdr (Byte1) 0x30 from net 17 [0 1557]</pre>	
<pre>Publish Message Received Topic: /cc3200/Sensors [Qos: 0] Data is: { "ACC": 1000 }</pre>	Envío de configuración

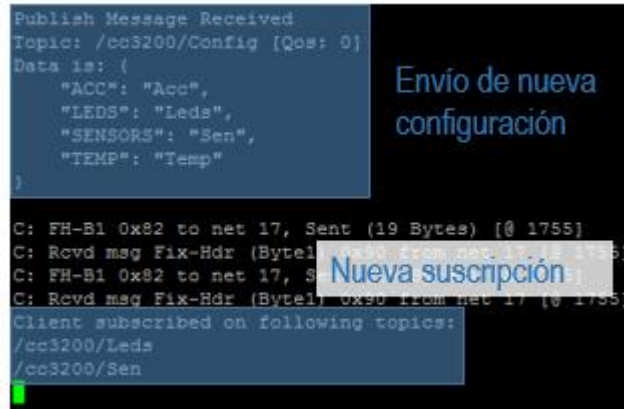
Figura 4.10. Información recibida tras configurar ambos sensores de forma simultánea.

<pre>CC3200 Publishes the following message Topic: /cc3200/Temp C: FH-B1 0x35 to net 17, Sent (55 Bytes) [0 1651] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1652] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1652] C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [0 1652]</pre>	Publicación de medidas simultánea
<pre>CC3200 Publishes the following message Topic: /cc3200/Acc C: FH-B1 0x35 to net 17, Sent (47 Bytes) [0 1652] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1652] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1652] C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [0 1652]</pre>	
<pre>CC3200 Publishes the following message Topic: /cc3200/Acc C: FH-B1 0x35 to net 17, Sent (55 Bytes) [0 1652] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1653] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1653] C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [0 1653]</pre>	
<pre>CC3200 Publishes the following message Topic: /cc3200/Acc C: FH-B1 0x35 to net 17, Sent (47 Bytes) [0 1653] C: Rcvd msg Fix-Hdr (Byte1) 0x50 from net 17 [0 1653] C: FH-B1 0x62 to net 17, Sent (4 Bytes) [0 1653] C: Rcvd msg Fix-Hdr (Byte1) 0x70 from net 17 [0 1653]</pre>	
<pre>CC3200 Publishes the following message Topic: /cc3200/Temp C: Rcvd msg Fix-Hdr (Byte1) 0x30 from net 17 [0 1653]</pre>	
<pre>Publish Message Received Topic: /cc3200/LedsArray [Qos: 0] Data is: { "B": 255, "G": 2, "LED": 10, "R": 0 }</pre>	Envío de Información de LEDs

Figura 4.11. Información recibida tras la configuración simultánea de ambos sensores y el control de un punto de luz.

Prueba 7

En la figura 4.12 se refleja la capacidad que tiene el sistema para, una vez configurados los diferentes *topics*, modificar la configuración de los *topics* de publicación y suscribirse a los nuevos *topics*.



The screenshot shows a terminal window with the following text:

```

Publish Message Received
Topic: /cc3200/Config [Qos: 0]
Data is: {
  "ACC": "Acc",
  "LEDS": "Leds",
  "SENSORS": "Sen",
  "TEMP": "Temp"
}

C: FH-B1 0x82 to net 17, Sent (19 Bytes) [0 1755]
C: Rcvd msg Fix-Hdr (Byte1: 0x82, Byte2: 0x82) [0 1755]
C: FH-B1 0x82 to net 17, Sent (19 Bytes) [0 1755]
C: Rcvd msg Fix-Hdr (Byte1: 0x82, Byte2: 0x82) [0 1755]
Client subscribed on following topics:
/cc3200/Leds
/cc3200/Sen
  
```

Annotations on the image:

- Envío de nueva configuración**: A blue text box pointing to the JSON data in the first message.
- Nueva suscripción**: A white text box with a blue border pointing to the list of topics at the bottom.

Figura 4.12. Información recibida tras una nueva configuración de los *topics*.

Capítulo 5. Conclusiones y líneas futuras

En este capítulo analizaremos el resultado del proyecto y evaluaremos si la elección tanto del *hardware* como del protocolo de comunicación ha sido acertada teniendo en cuenta los objetivos de este proyecto.

Tal y como se comentó en el capítulo 1, este proyecto presenta como objetivos evaluar la idoneidad de una plataforma concreta y tecnología radio concretas para la futura implementación de algunos de los elementos que pueden constituir una estructura de inteligencia ambiental. Uno de los elementos más importantes de dicha aplicación sería el dispositivo controlador, un elemento dotado de inteligencia computacional con capacidad para gobernar el sistema en su conjunto y para la comunicación con otros dispositivos de forma inalámbrica. En este caso, la elección del microcontrolador CC3200 de Texas Instruments parece acertada. Este microcontrolador tiene la suficiente capacidad para cumplir los requerimientos que pueda tener dicho dispositivo además de disponer de conectividad *Wi-Fi* lo que lo hace ideal para este fin. Dispone de diferentes tipos de buses destinados a la comunicación con otros elementos como el bus SPI el cual hemos evaluado en la conexión con un sistema de iluminación o el bus I²C para la comunicación con diferentes sensores. Si hubiera que destacar alguna carencia de este microcontrolador sería la baja disponibilidad de pines que ofrece para la conexión con otros elementos ya que la mayoría se encuentran multiplexados unos con otros y no permite su uso simultáneo.

Por otro lado, se ha estudiado el comportamiento de un sistema de iluminación compuesto por un conjunto de luces tipo LED las cuales incluyen un *driver* integrado en un pequeño chip. Este sistema tiene la capacidad de controlar

cada punto de luz individualmente a través de una sola línea física ya que estos LED siguen un protocolo de comunicación serie haciendo posible la comunicación de unos con otros. Este sistema resulta muy apropiado para, por ejemplo, controlar el alumbrado público de forma muy sencilla y personalizada para cada punto de luz, o la implementación de iluminación vial. La configuración de este sistema y el acceso a cada LED se hace muy sencillo gracias a la librería WS2812drv desarrollada por el Departamento de Tecnología Electrónica y que ha sido probada con la plataforma evaluada en este proyecto.

En el apartado de la aplicación de usuario, se ha desarrollado una interfaz gráfica de prueba para evaluar si la tecnología sería apropiada para una aplicación final. En este caso, se ha desarrollado una interfaz mediante el *framework* multiplataforma Qt el cual hace muy sencilla la implementación de esta gracias a ser orientado a objetos. Podemos decir que el desarrollo de una aplicación similar a esta sería suficiente para el fin general ya que se trata de una interfaz clara e intuitiva para personas que no dominen conceptos tecnológicos.

Finalmente, hemos analizado una de las partes más importantes de este proyecto: el protocolo de comunicación. Es un elemento indispensable ya que incorpora la capacidad de que nuestros dispositivos sean inalámbricos. Como conclusión, podemos decir que MQTT es un *middleware* apto para nuestro propósito ya que se trata de un protocolo ligero, que no requiere grandes recursos ni gran ancho de banda. Además, permite, gracias a su estructura de *topics*, la conexión de múltiples dispositivos y una sencilla comunicación con ellos.

Como líneas futuras para nuestro proyecto podemos contemplar la idea de desarrollar una aplicación para *smartphones*, sustituyendo u ofreciendo una alternativa a la aplicación gráfica para PC. Esto acercaría este proyecto a un público más general y haría más cómodo el uso de un sistema como el que tomamos como ejemplo (maqueta de *Smart City*).

En cuanto a líneas futuras orientadas al proyecto general podemos considerar la evaluación de otras plataformas similares o de otras tecnologías radio

elaborando una comparativa de qué plataformas o tecnologías de comunicación resultan más apropiadas para los requerimientos del proyecto.

Apéndice A. Manual de usuario

En cuanto al apartado *software* se describirá de forma detallada el funcionamiento tanto de la aplicación desarrollada para el microcontrolador como de la aplicación gráfica para el equipo de usuario.

El microcontrolador escuchará a través de un *topic* configurado en el *firmware* en el cual espera recibir información del usuario sobre a qué *topics* debe suscribirse y en qué *topics* debe publicar. Este *topic* será denominado como Topic de configuración.

Por su parte, la aplicación de usuario tendrá la capacidad de enviar a un *topic* (definido por el usuario) la anterior información. Esta puede contener el *topic* de control de luces, el *topic* de control de sensores y los *topics* de recepción de mediciones.

Una vez configurada la comunicación, el microcontrolador puede recibir peticiones sobre el control de la tira de luces o la activación y configuración de la medida de cualquiera de los sensores por parte del usuario. En el caso de activación de alguno de los sensores, el microcontrolador comenzará a enviar de forma periódica (periodo configurable) lecturas del sensor a la aplicación de usuario la cual mostrará tanto la medida actual como un registro de las últimas medidas en forma de gráfica.

A continuación, se procederá a analizar el *firmware* del microcontrolador.

En primer lugar, el microcontrolador hace un intento de conexión a la red *Wi-Fi* configurada en el fichero *common.h* o crea un punto de acceso *Wi-Fi* (modo

station o modo AP respectivamente). En el caso de que la conexión a la red *Wi-Fi* o la creación del punto de acceso se haya realizado correctamente el microcontrolador se conectará al servidor MQTT el cual debe haber sido lanzado con anterioridad en el broker. Se mostrará información de depuración a través del puerto serie sobre el éxito de la conexión.

Una vez que la conexión al servidor haya sido satisfactoria el microcontrolador quedará suscrito al *topic* de configuración por defecto y quedaremos a la espera de recibir por él la información adecuada para configurar nuestra lista de *topics* (tanto de suscripción para actuar sobre las luces y configurar las medidas de los sensores como de publicación para enviar las lecturas de los mismos).

Tras recibir dicha información a través del *topic* de configuración se llevará a cabo la suscripción de los *topics* recibidos y almacenada la información de los *topics* dedicados a publicación de medidas.

En este momento nuestro sistema queda a la espera de recibir peticiones del equipo de usuario ya sea para controlar la tira de luces o configurar la medida de los sensores.

Finalmente se describen las actuaciones que puede llevar a cabo nuestro sistema:

- Control de tira de luces: tras recibir una petición del equipo de usuario relacionada con el control de la tira de luces, el microcontrolador tendrá la capacidad de modificar la configuración de uno de los puntos de luz de la tira (seleccionado en la interfaz de usuario) mediante tres valores (la intensidad de luz roja, verde y azul) en rangos de 0-255 los cuales son seleccionables desde la interfaz pudiendo encender cada luz de casi 17 millones de colores distintos.
- Configuración de lectura de sensores: al ser recibida una petición de configuración de la lectura de uno de los sensores, el microcontrolador creará una tarea la cual de forma periódica (periodo configurable en la interfaz de usuario en milisegundos donde el valor 0 corresponde a eliminar la tarea para detener la lectura) procederá a leer del sensor

correspondiente y a publicar en el *topic* configurado al inicio dicha lectura.

A continuación, se procederá a describir la interfaz de usuario la cual podemos dividir en cuatro secciones:

- Pestaña de conexión MQTT (figura A.1): en esta sección se configurará todo lo relacionado con la conexión MQTT como es la IP (*Internet Protocol*) del servidor MQTT y los *topics* de suscripción y publicación. Se dispondrá de un botón que podemos identificar como “Inicio” el cuál, al ser pulsado, realizará la conexión al servidor MQTT y la suscripción a los *topics* reservados para la recepción de la lectura de los sensores. Por otro lado, el botón “Set config” realizará la publicación de la información correspondiente a la configuración de los *topics* necesaria en el microcontrolador en el *topic* seleccionado para ello (Topic Config).

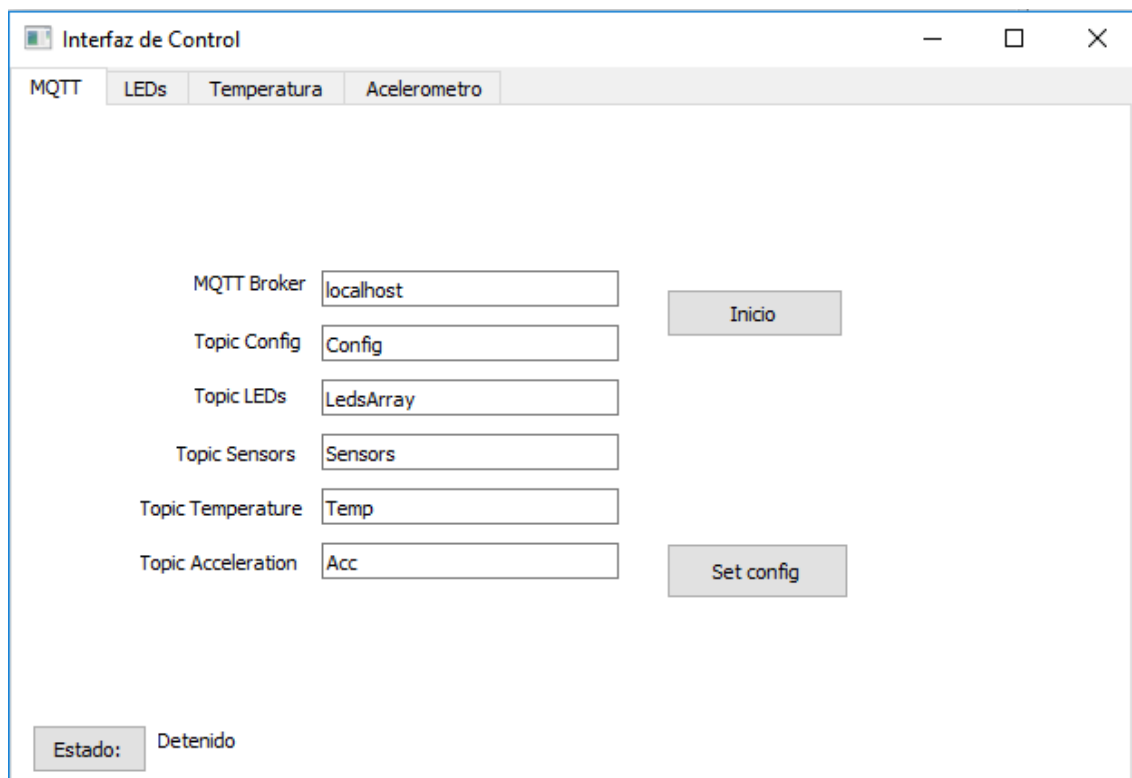


Figura A.1. Pestaña de configuración de la interfaz de control.

- Pestaña de control de luces (figura A.2): esta sección contiene los controles necesarios para actuar sobre la tira de luces. En concreto podemos seleccionar el color deseado en la ruleta de colores la cual codificará nuestra elección en formato RGB (tres valores con rangos de 0 a 255 correspondientes a los colores rojo, verde y azul) y elegir la luz que deseamos modificar en la ruleta que aparece a su derecha. El botón “Actualizar LED” llevará a cabo la publicación en el *topic* correspondiente de la configuración seleccionada en las ruletas anteriormente descritas. Por otro lado, el botón “Apagar LED” hará un recorrido por la ruleta de elección de LED publicando en el *topic* correspondiente al control de luces los valores R: 0, G: 0, B: 0 para cada LED disponible en la tira de luces lo que hará que el microcontrolador apague toda la tira de luces.

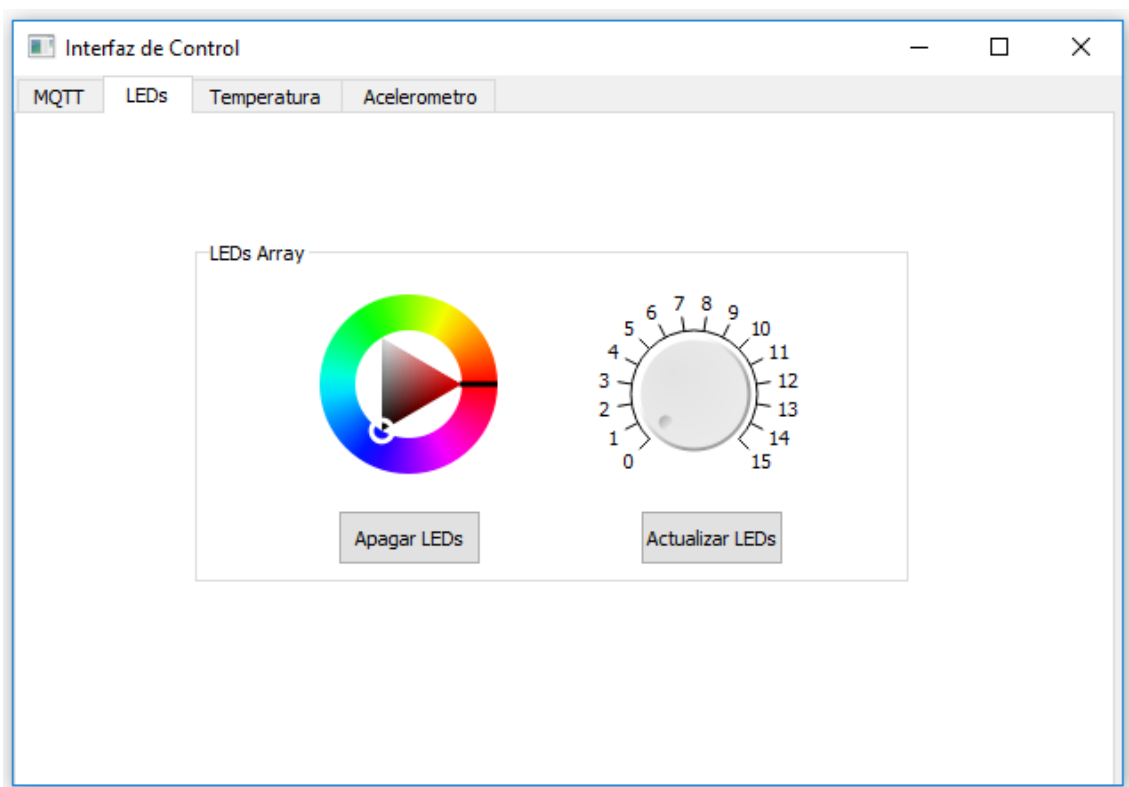


Figura A.2. Pestaña de control de sistema de iluminación.

- Pestaña de información de temperatura (figura A.3): en esta pestaña se podrá configurar la lectura del sensor de temperatura y visualizar tanto en un termómetro como en una gráfica los valores de temperatura

obtenidos por el sensor. En el control numérico se podrá configurar el periodo de actualización del parámetro en milisegundos mientras que el botón “ON” será el encargado de realizar la publicación en el *topic* correspondiente que configurará y arrancará la lectura del sensor en el microcontrolador. Una vez comience la recepción de lecturas en la interfaz se podrá observar tanto la última medida en el termómetro como un registro en forma de gráfica de las últimas 20 medidas.

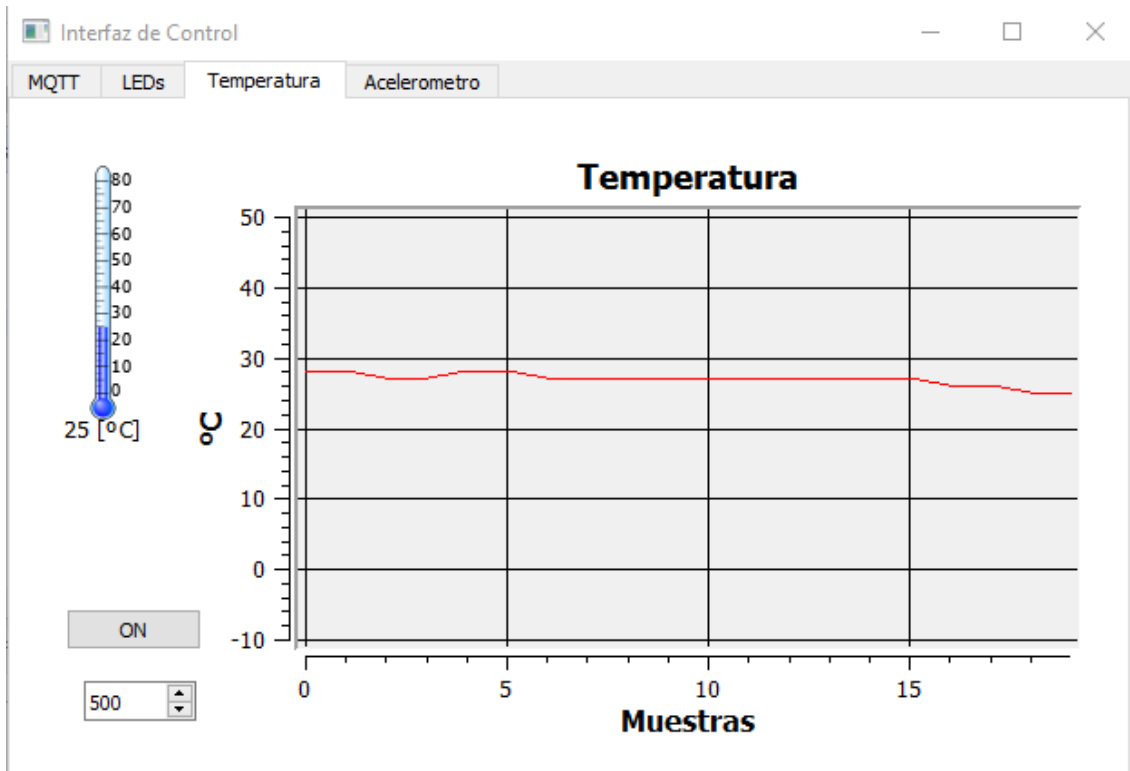


Figura A.3. Pestaña de control y visualización de sensor de temperatura.

- Pestaña de información de aceleración (figura A.4): en esta pestaña se podrá configurar la lectura del acelerómetro y visualizar tanto en indicadores numéricos como en una gráfica los valores de aceleración obtenidos por el sensor. En el control numérico se podrá configurar el periodo de actualización del parámetro en milisegundos mientras que el botón “ON” será el encargado de realizar la publicación en el *topic* correspondiente que configurará y arrancará la lectura del sensor en el microcontrolador al igual que en la pestaña anterior. Una vez comience la recepción de lecturas en la interfaz se podrá observar tanto la última

medida en los indicadores numéricos como un registro en forma de gráfica de las últimas 20 medidas.

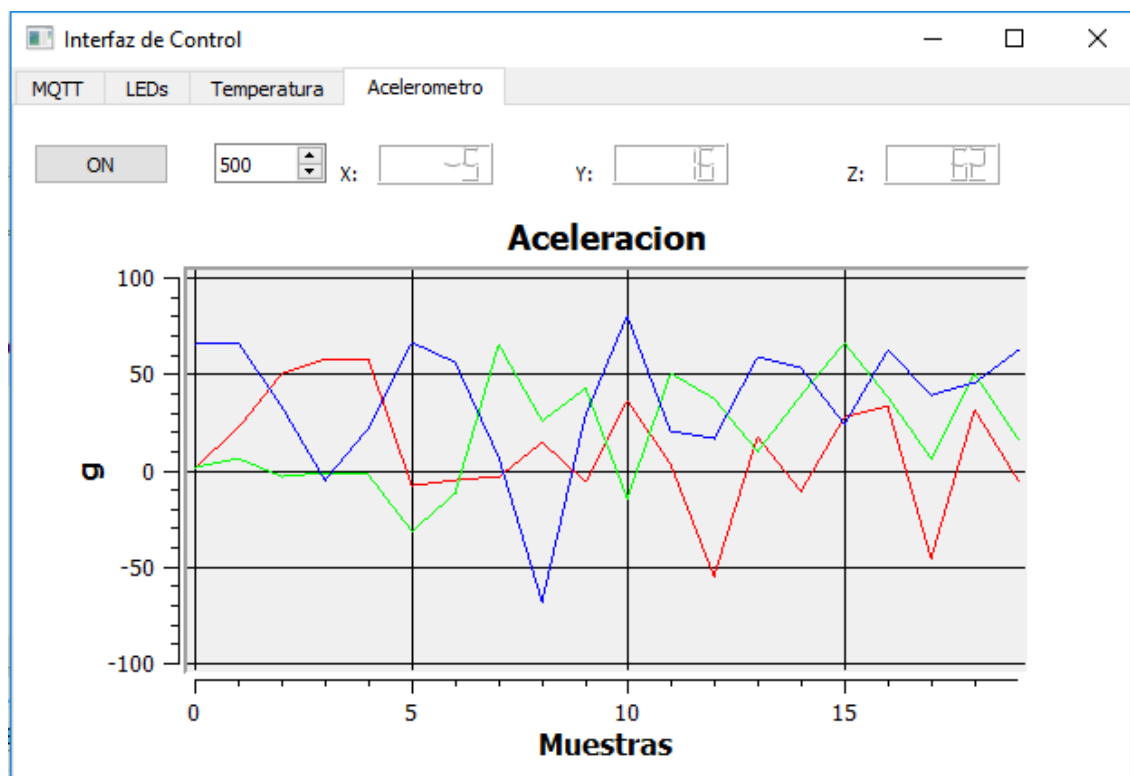


Figura A.4. Pestaña de control y visualización de sensor de aceleración.

Referencias

- [1] M. Batty, K. Axhausen, F. Giannotti, A. Pzdnoukhov, B. A., M. Wachowicz, G. Ouzounis y Y. Portugali, «Smart cities of the future» The European Physical Journal Special Topics, vol. 214, pp. 481-518, Noviembre 2012.
- [2] Fundación Telefónica, «Smart Cities: Un primer paso hacia el Internet de las cosas» Editorial Ariel, Octubre 2011.
- [3] A. Bartolomé Muñoz de Luna, M. Barquero Cabrero y J. E. González Vallés, «Las Rozas Smart Green City: construcción del municipio desde el ciudadano,» Historia y Comunicación Social, vol. 20, nº 2, pp. 549-577, 2015
- [4] “Texas Instruments, CC3220 SimpleLink™ Wi-Fi® Wireless and Internet-of-Things Solution, a Single-Chip Wireless MCU (Rev. A)”, 2017. URL: <http://www.ti.com/lit/ds/swas035a/swas035a.pdf> (última consulta el 15/Junio/2017).
- [5] “Texas Instruments, CC3200 SimpleLink Wi-Fi and IoT Solution w/ MCU LaunchPad Hardware User's Guide (Rev. B)”, 2015. URL: <http://www.ti.com/lit/ug/swru372b/swru372b.pdf> (última consulta el 15/Junio/2017).
- [6] “Texas Instruments, TMP006/B Infrared Thermopile Sensor in Chip-Scale Package”, 2015. URL: <http://www.ti.com/lit/ds/symmlink/tmp006.pdf> (última consulta el 15/Junio/2017).
- [7] “Bosch Sensortec, BMA222 – Digital, triaxial acceleration sensor”, 2012. URL: <http://www.mouser.com/ds/2/783/BST-BMA222-FL000-02-786483.pdf> (última consulta el 15/Junio/2017).

- [8] “Worldsemi, WS2812B – Intelligent control LED integrated light source”.
URL: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf> (última consulta el 15/Junio/2017).
- [9] FreeRTOS. [Online]. <http://www.freertos.org/>
- [10] MQTT. [Online]. <http://mqtt.org/>
- [11] JSON. [Online]. <http://www.json.org/>