UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

Grado en Ingeniería Telemática

Trabajo Fin de Grado

BrainyCUM: Almacenamiento y Visualización de Datos Procedentes de Dispositivos IoT

Autor: Jesús Salguero Serrat

Mérida, Septiembre de 2017

UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

Grado en Ingeniería Telemática

Trabajo Fin de Grado

BrainyCUM: Almacenamiento y Visualización de Datos Procedentes de Dispositivos IoT

Autor: Jesús Salguero Serrat
Fdo.:

Director: Héctor Sánchez Santamaría
Fdo.:

Resumen

El Internet de las Cosas (IoT) es un concepto que está adquiriendo más popularidad cada día. Con un objetivo claro, conectar todos los objetos que se puedan a Internet, permite a las personas tener el control de lo que nos rodea. La temperatura, la humedad y la luminosidad de un lugar son ejemplos de los millones de datos que se generan por segundo. Estos datos pueden almacenarse y ser procesados, permitiendo realizar una serie de acciones automáticas en tiempo real en función de estos.

Uno de los ámbitos que ha revolucionado el IoT es la edificación inteligente. Se ha conseguido una gestión energética eficiente, un aumento en la seguridad y una mejora en el bienestar de las personas.

BrainyCUM aspira a transformar el Centro Universitario de Mérida (CUM) en un laboratorio desde el que contribuir a generar soluciones tecnológicas enfocadas a la edificación inteligente y al desarrollo sostenible, siendo este Trabajo Fin de Grado un primer acercamiento.

Se ha diseñado y desarrollado un sistema de información capaz de capturar, almacenar y tratar los datos generados por diferentes tipos de dispositivos IoT. Estos dispositivos se encuentran conectados a la red empleando diversas tecnologías y módulos de comunicación, y usando protocolos para el envío y la recepción de datos. Además, el sistema cuenta con una plataforma IoT, que se encarga de almacenar la información obtenida en una base de datos distribuida y de tiempo real, cuenta con diferentes reglas que disparan una serie de acciones en función de los datos obtenidos haciendo que el sistema sea autónomo, y transmite estos datos a una aplicación de visualización. Esta aplicación muestra a los usuarios la información obtenida de manera amena y sencilla, siendo responsive, por lo que se adapta a cualquier tipo de dispositivo.

Palabras clave: IoT, Edificación Inteligente, Sistema de Información, Datos

Abstract

The Internet of Things (IoT) is a concept that is gaining more popularity every day. With a clear goal, connecting all the objects that can be to the Internet, allows people to have control of what surrounds us. The temperature, humidity and brightness of a place are examples of millions of data generated per second. This data can be stored and processed, allowing to perform a series of automatic actions in real time depending on these.

One of the areas that has revolutionized IoT is intelligent building. Efficient energy management, increased security and improved well-being have been achieved.

BrainyCUM aims to transform the University Center of Mérida (CUM) into a laboratory from which to contribute to generate technological solutions focused on intelligent building and sustainable development, this End-of-Grade Work being a first approach.

An information system capable of capturing, storing and processing data generated by different types of IoT devices has been designed and developed. These devices are connected to the network using various technologies and communication modules, and using protocols for sending and receiving data. In addition, the system has an IoT platform, which is responsible for storing the information obtained in a distributed database and real time, has different rules that trigger a series of actions depending on the data obtained by making the system is Autonomous, and transmits this data to a display application. This application shows users the information obtained in a friendly and simple way, being responsive, so that it adapts to any type of device.

Keywords: IoT, Intelligent Building, Information System, Data

Agradecimientos

En primer lugar, quiero dar las gracias a Héctor Sánchez Santamaría, director de este Trabajo Fin de Grado, quien me propuso la realización de este proyecto y me metió de lleno en un tema desconocido para mí que ha resultado bastante interesante. También quisiera agradecer sus correcciones y ayuda proporcionada durante su realización.

Me gustaría dedicar este Trabajo Fin de Grado a mis padres, Ramón y Dolores, y a mi hermano Manuel por todo el esfuerzo que han realizado estos años que ha hecho posible el estudio de las titulaciones que estoy a punto de finalizar, y por todo el apoyo y ánimo que me han dado cada año.

También, me gustaría dedicárselo a Noelia, por las palabras de apoyo que me ha regalado cada día y por darme la fuerza y motivación necesaria para lograr superar cada reto que aparecía en el camino. Gracias por confiar en mí.

Me gustaría agradecer también a mis compañeros con los que he compartido estos años, en especial a Adrián, por acompañarme estos cinco años y por la ayuda que siempre nos hemos ofrecido.

Por último, me gustaría agradecer al Centro Universitario de Mérida, tanto el equipo docente como al personal que allí trabaja, toda la ayuda brindada durante estos años, y toda la formación que me han impartido con gran profesionalidad y cercanía.

Muchas gracias a todos.

Índice general

Índice de figuras

Índice de tablas

Algoritmos

Capítulo 1

Introducción

Contenidos

Este primer capítulo servirá para introducir este Trabajo Fin de Grado. Para ello, se abordarán los siguientes aspectos: introducción, antecedentes y objetivos que se pretende alcanzar.

En la sección 1.1 se explica la importancia del Internet de las cosas en el mundo actual, así como una breve introducción de BrainyCUM.

En la sección 1.2 se describe distintas implementaciones que se han realizado en los últimos años gracias al Internet de las cosas.

En la sección 1.3 se detallan los objetivos de este Trabajo Fin de Grado.

1.1. Introducción

Actualmente, el Internet de las Cosas, Internet of Things en inglés (IoT), es un concepto que está adquiriendo más popularidad cada día. Con un objetivo claro, conectar todos los objetos que se puedan a Internet, permite a las personas tener el control de lo que nos rodea. La temperatura, la humedad y la luminosidad de un lugar son ejemplos de los millones de datos que se generan por segundo. Estos datos procedentes de dispositivos IoT se almacenan y se procesan, permitiendo mandar distintas órdenes en tiempo real a una gran variedad de dispositivos, como por ejemplo activar la calefacción de una casa cuando se alcance una determinada temperatura.

Uno de los ámbitos que ha revolucionado el IoT es la edificación inteligente. Se ha conseguido una gestión energética eficiente, un aumento en la seguridad y el bienestar, y se ha mejorado las comunicaciones de las edificaciones, y todo gracias a la obtención y manejo de datos.

BrainyCUM aspira a transformar el Centro Universitario de Mérida (CUM) en un laboratorio desde el que contribuir a generar soluciones tecnológicas enfocadas a la edificación inteligente y al desarrollo sostenible.

La sensorización de las aulas, laboratorios y espacios comunes del CUM mediante dispositivos IoT y el almacenamiento y posterior tratamiento de los datos generados por estos dispositivos, son los principales marcos de desarrollo que se enmarcan en BrainyCUM.

1.2. Antecedentes

Desde que en 1999 Kevin Ashton propusiera el concepto de Internet de las cosas, han surgido numerosas implementaciones, algunas de ellas verdaderamente sorprendentes.

En la actualidad, nos podemos encontrar dicho concepto hasta en unas zapatillas. Estas son capaces de registrar datos como el tiempo y la distancia recorrida e incluso la ruta seguida por la persona que las lleva. Otro ejemplo destacable es un collar inteligente para las mascotas, capaz de medir la actividad del animal, la calidad del sueño y detalles sobre su comportamiento.

Uno de los terrenos que ha mejorado bastante el IoT es sin ninguna duda la agricultura. Se han instalado sistemas que optimizan el consumo de agua para reducir su desperdicio y ahorrar costes, otros realizan predicciones sobre los datos obtenidos para aumentar el rendimiento y realizar los planes de siembra. Hay algunas implementaciones sorprendentes que controlan el nivel de feromonas, lanzándolas para confundir a los insectos y sus hábitos de apareamiento, y así reducir drásticamente la cantidad de plagas.

Otro importante campo en el cual se han realizado numerosas implementaciones, es en la edificación inteligente. Actualmente, se puede encontrar este concepto en oficinas, viviendas, centros comerciales, hospitales, universidades, entre muchos otros. Se está logrando una gestión automática y eficiente de los edificios, mejorando el ahorro energético y reduciendo los gastos que se producen en un inmueble.

1.2.1. Algunos casos de interés

En el ámbito educativo también aparece el concepto de edificación inteligente y una gran cantidad de centros educativos están implantando este concepto en sus instalaciones.

Por ejemplo, el centro Tknika [1], en el País Vasco, es una de las principales referencias en domótica en nuestro país. Cuenta con una serie de sistemas que permiten el control de la iluminación, las persianas y los enchufes desde una aplicación móvil. También cuenta con sistemas de aviso al usuario, que se encargan de notificar las posibles acciones realizadas que afecten a la eficiencia energética del centro. Si una persona abre una ventana de una sala que tiene la calefacción encendida, se escuchará una voz de aviso. Además, los docentes pueden reservar un aula a través de Google Calendar y automáticamente el sistema ajustará la calefacción, las luces y activará la electricidad en el periodo de uso.

También es importante destacar un ejemplo más cercano, el proyecto SmartPoliTech de la Escuela Politécnica de Cáceres [2]. Empezó en el año 2013 y sigue en constante crecimiento. Su objetivo es implantar sistemas que creen espacios inteligentes en la facultad, logrando una gestión energética eficiente que facilite la vida social y académica. Para ello, se ha sensorizado y automatizado distintos espacios del centro. Un ejemplo importante fue la instalación de sensores para controlar el consumo de agua. Gracias a ellos, pudieron detectar en mayo de 2017 un consumo excesivo de agua debido a una cisterna estropeada. En la figura 1.1[1], se puede

[1]Imagen e información obtenida del Twitter del proyecto *SmartPoliTech* (@SmartPoliTech), accesible en https://twitter.com/SmartPoliTech?lang=es

observar la gráfica del consumo de agua de la Escuela Politécnica de Cáceres, pudiéndose ver un consumo excesivo de agua durante los días 6 y 7 de mayo.

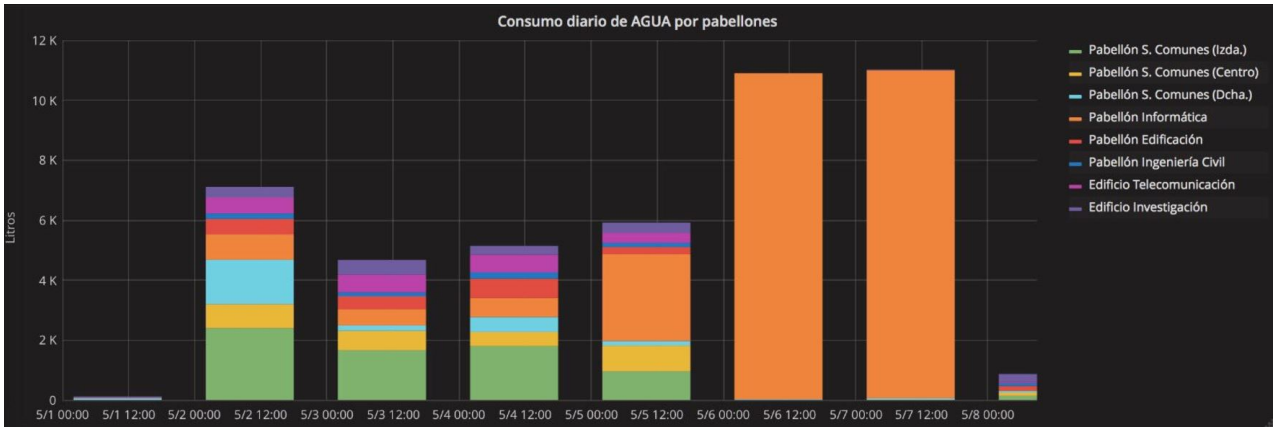

Figura 1.1: Gráfica del consumo de agua de la Escuela Politécnica de Cáceres.

1.3. Objetivos

El objetivo general de este Trabajo Fin de Grado es diseñar y desarrollar un sistema de información capaz de capturar, almacenar y tratar los datos generados por los dispositivos IoT desplegados por el CUM. Para ello, se sensorizarán distintos espacios del CUM, utilizando diferentes tipos de dispositivos tanto para la captación de datos como para la actuación en función de estos. Se pretende que el sistema sea independiente de los dispositivos IoT utilizados, por lo que se podrá utilizar cualquiera sin ningún problema. Se necesitará una plataforma IoT que se encargue de la recepción y el tratamiento de los datos obtenidos por los dispositivos. Esta plataforma será la encargada de almacenar estos datos en una base de datos especializada en IoT, y en proporcionarlos a la aplicación de visualización. Esta aplicación mostrará la información obtenida de una manera sencilla y entendible por los usuarios del sistema.

Este objetivo general se alcanzará a través de la consecución de los siguientes objetivos específicos:

- Analizar las bases de datos distribuidas, orientadas a documentos y de tiempo real para el almacenamiento de los datos obtenidos por los dispositivos IoT.
- Analizar diferentes plataformas IoT para el desarrollo de este proyecto.
- Estudiar las posibles soluciones de visualización de datos procedentes de dispositivos IoT.
- Diseñar y desarrollar un sistema de información para la captura, almacenamiento y visualización de datos procedentes de dispositivos IoT.
- Despliegue del sistema de información para su evaluación.

Capítulo 2

Análisis previo

Contenidos

En este capítulo se realizará una introducción a una serie de conceptos previos necesarios para el desarrollo de este Trabajo Fin de Grado. Además, se realizará un análisis previo del sistema a desarrollar, detallando cada una de las partes necesarias para su implementación.

En la sección 2.1 se introducirá el concepto de Internet de las Cosas y se comentará su estructura.

En la sección 2.2 se realizará una breve introducción al concepto de dispositivo IoT y se analizarán las distintas partes por la que está formado.

En la sección 2.3 se estudiará el concepto de base de datos y se introducirán algunos tipos.

En la sección 2.4 se explicará qué es una plataforma IoT y cuáles son sus funciones.

En la sección 2.5 se realizará un análisis previo del sistema a desarrollar.

2.1. Internet de las Cosas

Con el gran avance que se está produciendo continuamente en la tecnología, es cada vez más normal encontrarse con más variedad de dispositivos con la capacidad de conectarse a Internet. Existen teléfonos, electrodomésticos, automóviles, relojes, gafas e incluso zapatillas conectadas a Internet. Gracias a este avance surge el concepto de Internet de las cosas [3, 4, 5], o IoT por sus siglas en inglés, que consiste en la interconexión de objetos cotidianos a Internet.

Este concepto fue propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999, donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red y tecnología de sensores, pero no fue hasta el año 2008 cuando este concepto empezó a tomar importancia. En el año 2003, había aproximadamente 6.300 millones de personas en el

planeta y había 500 millones de dispositivos conectados a Internet, por lo que había menos de un dispositivo por persona (0,08). La aparición y crecimiento de los smartphones y las tablets elevó a 12.500 millones la cantidad de dispositivos conectados a Internet en 2010. Se observa actualmente que la cantidad de dispositivos conectados a Internet se duplica cada 5 años aproximadamente, por lo que se estima que para 2020 existan 50.000 millones de dispositivos conectados, como se puede ver en la figura 2.1.

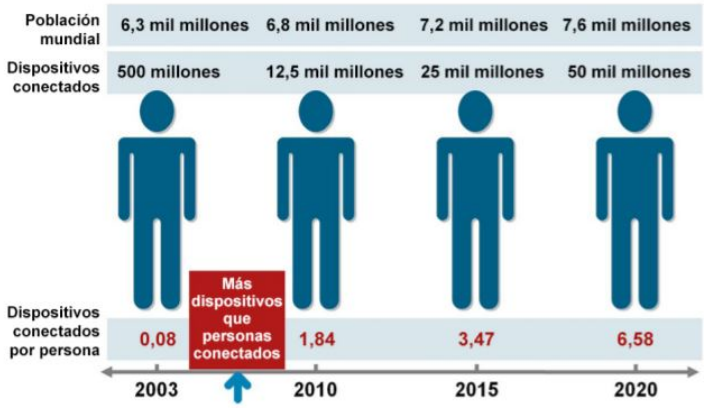

Figura 2.1: Estudio realizado por Cisco del avance del Internet de las cosas.

El IoT introduce un cambio radical en la calidad de vida de las personas, ofreciendo una gran cantidad de nuevas oportunidades de acceso a datos. Los distintos objetos que nos rodean podrían enviar datos de sus estados, por lo que una persona podría saber la temperatura que hace en su habitación y decidir si encender la calefacción, saber si tiene algún determinado producto en su frigorífico o hasta saber si hay algún aparcamiento libre en su zona.

En la figura 2.2, se puede observar que la estructura del IoT cuenta con:

- Los distintos **objetos conectados** o dispositivos IoT, que envían y/o reciben información.
- Las **tecnologías de red**, que permiten que estos objetos se conecten a la red.
- Los diferentes **protocolos de comunicación**, que permiten la comunicación de estos dispositivos con la red.
- Las **plataformas IoT**, que permiten el tratamiento de los datos.
- Las **aplicaciones de usuario**, que muestran al usuario la información deseada y les permite realizar las acciones oportunas.

2.2. Dispositivos IoT

Los dispositivos IoT son los encargados de recoger y transmitir información de su entorno y actuar en función de los datos obtenidos. Un mismo dispositivo puede comportarse de diferente manera según las diferentes condiciones que hayan sido programadas. Además, estos dispositivos pueden recibir información de la red y participar en ella, pudiéndose comunicar con más dispositivos que existan en esa red.

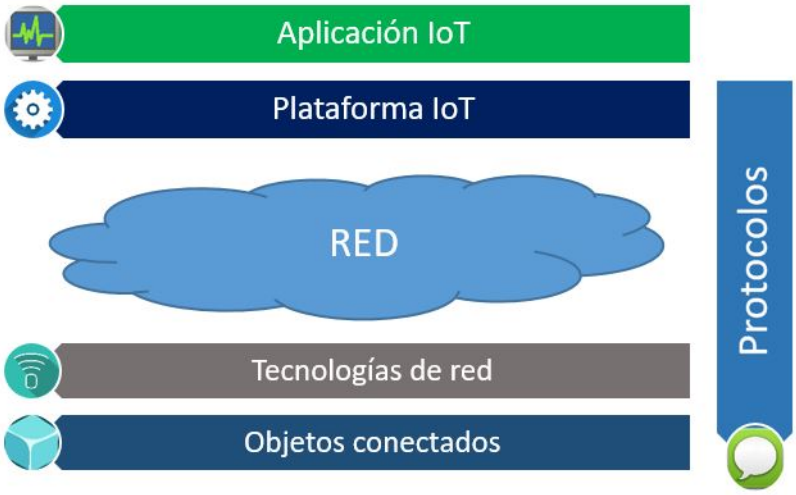

Figura 2.2: Estructura de Internet de las cosas.

2.2.1. Componentes de los dispositivos IoT

A continuación, se introducen los componentes que deben tener estos dispositivos para poder utilizarlos en el Internet de las Cosas.

2.2.1.1. Microcontroladores

Un microcontrolador [6, 7] es un circuito integrado digital que puede ser usado para diversos propósitos debido a que es programable, siendo capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto por una unidad central de proceso (CPU), memorias y periféricos de entrada/salida, bloques de funcionamiento básicos de cualquier computadora. La figura 2.3 es un esquema básico de un microcontrolador, en el cual se puede observar al microcontrolador formado por los componentes anteriormente mencionados conectados mediante buses. Fuera del microcontrolador se encuentran los distintos dispositivos que se pueden conectar a los pines de entrada/salida, además de los pines de alimentación, masa y otros circuitos necesarios para que el microcontrolador pueda trabajar.

Para usar un microcontrolador se debe especificar su funcionamiento por software a través de programas con las instrucciones que el microcontrolador debe realizar. En memoria se guardan los programas realizados y la CPU se encarga de procesar las distintas instrucciones del programa. Los lenguajes de programación típicos son de bajo nivel, como pueden ser ensamblador o C.

Una plataforma muy conocida para la programación de microcontroladores es Arduino, la cual se introducirá más adelante.

2.2.1.2. Sensores

Los sensores [6] son dispositivos capaces de captar los cambios en el tiempo de una determinada magnitud física, como puede ser la temperatura, la humedad o el movimiento, y

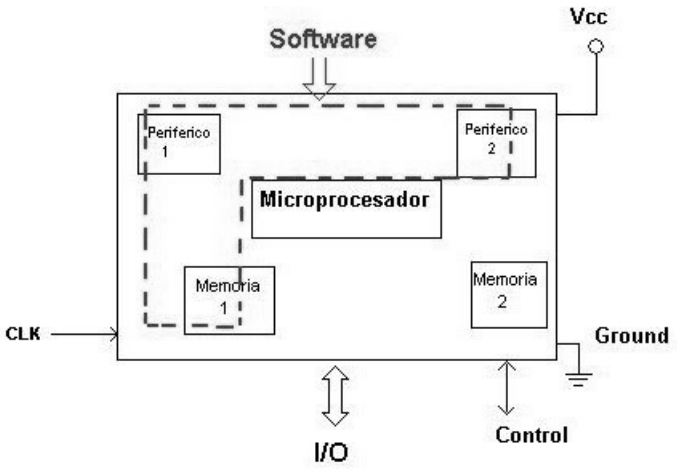

Figura 2.3: Esquema de un microcontrolador.

transformarlos en señales eléctricas. Estas señales eléctricas pueden ser interpretadas por un microcontrolador que tenga conectado a una de sus entradas la salida del sensor.

Los sensores pueden ser analógicos o digitales, dependiendo del tipo de señal que devuelvan. Los sensores analógicos son aquellos que emiten un voltaje comprendido entre dos valores, que varía en función del valor que estén midiendo. En cambio, los digitales generan una señal discreta, pudiendo tomar varios valores. Por ejemplo, la señal puede tomar el valor 1 representando un nivel alto, o un 0 representando un nivel bajo. En la figura 2.4, se puede observar una representación de ambos tipos de señal. Una señal analógica puede corresponderse, por ejemplo, con la temperatura medida en un aula. Una señal digital, por ejemplo, puede representar si un interruptor está pulsado o no.

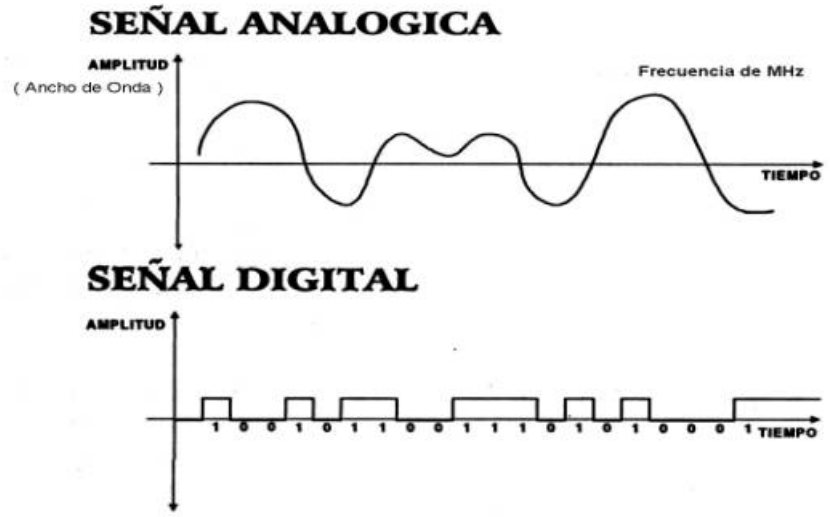

Figura 2.4: Ejemplo de señal analógica y señal digital.

2.2.1.3. Tecnologías de red

Las tecnologías de red [6, 8], a través de módulos de comunicación, permiten que estos dispositivos se conecten a la red. La conexión se puede realizar mediante cable o de manera inalámbrica. Existen numerosas tecnologías, destacando las siguientes:

- **Bluetooth:** Es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM[1] de los 2.4 GHz.
- **ZigBee:** Es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal. Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías. Destaca en el ámbito de la domótica debido a su bajo consumo, su topología de red en malla y su fácil integración.
- **Wifi:** Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos habilitados con wifi, como por ejemplo, un ordenador, un televisor inteligente o un smartphone, pueden conectarse a Internet a través de un punto de acceso de red inalámbrica.
- **Redes móviles:** Son aquellas redes pensadas para que un dispositivo, como un smartphone, pueda moverse con libertad en la zona cubierta por dicha red. Estas redes deben permitir el movimiento de los dispositivos a altas velocidades, como la velocidad de un coche o un tren, sin que exista una perdida de la conexión. Existen diferentes estándares como 3G, 4G o GPRS, que ofrecen distintas velocidades de transmisión.
- **Ethernet:** Es un estándar de transmisión de datos para redes de área local cableadas. Actualmente, es el sistema de conexión por cable de mayor difusión.

2.3. Bases de datos

Una base de datos [9] es un conjunto de datos estructurados y relacionados que se encuentran almacenados con el objetivo de facilitar su posterior utilización.

Las bases de datos pueden ser locales, siendo utilizadas en un solo equipo por un usuario, o pueden ser distribuidas, almacenando la información en equipos remotos, accediendo a ella a través de una red.

La administración de las bases de datos se realiza con un Sistema de Gestión de Bases de Datos (SGBD), también llamado DBMS por sus siglas en inglés (Database Management System). El SGBD es un conjunto de servicios que permite a los usuarios un fácil acceso a la información y proporciona las herramientas necesarias para su manipulación. Un SGBD puede dividirse en tres subsistemas:

- **Sistema de administración de archivos:** Su función es almacenar la información en un medio físico.
- **SGBD interno:** Se encarga de ubicar la información en orden.

[1] Bandas reservadas internacionalmente para uso no comercial de radiofrecuencia.

- **SGBD externo:** Representa la interfaz de usuario.

2.3.1. Tipos de base de datos

2.3.1.1. Base de datos relacionales

Las bases de datos relacionales [9] ordenan los datos en tablas compuestas de columnas y filas. Cada columna es un atributo de la entidad en cuestión, por ejemplo, nombre, apellidos, DNI o fecha de nacimiento. Se elige un atributo particular o una combinación de atributos como clave primaria, a la cual se puede hacer referencia en otras tablas, en donde se denominan clave externa. Cada fila incluye datos sobre una instancia específica de la entidad, por ejemplo, un alumno específico. Este modelo define también las relaciones existentes entre estas tablas, pudiendo ser uno a uno, uno a muchos y muchos a muchos.

El uso de bases de datos relacionales aportan una serie de ventajas [11]:

- Cuentan con gran soporte y herramientas debido a que existen desde hace bastante tiempo.
- Son ampliamente conocidas y utilizadas.
- Cuentan con transaccionalidad entre tablas, por lo que si durante una petición se produce un error, se devuelve al punto inicial sin comprometer los datos que fueron utilizados durante el proceso.
- Los datos deben cumplir con el tipo de dato definido en su estructura.

No obstante, cuenta con una serie de contras a tener en cuenta:

- No son flexibles ya que todos los objetos ingresados deben tener los mismos campos y deben estar correctamente validados.
- Mientras mas compleja se vuelvan necesitan más tiempo de procesamiento, por lo que afecta a su rendimiento.
- La escalabilidad es reducida, ya que se necesita aumentar los recursos hardware que generalmente son bastante costosos.

2.3.1.2. Bases de datos NoSQL

Este termino apareció a finales de los años 90 como respuesta a la necesidad de gestionar volúmenes masivos de información. La cantidad de información manejada por redes sociales, buscadores y otros muchos, necesitaban arquitecturas de almacenamiento con un alto rendimiento, escalables y distribuidas, siendo inadecuado el uso de bases de datos relacionales.

En general, se considera que existen cuatro tipos de bases de datos NoSQL [10]:

- **Orientadas a documentos:** Son aquellas que gestionan datos semiestructurados, como pueden ser XML, JSON o BSON. Son las bases de datos NoSQL más versátiles, pudiendo ser utilizadas en una gran cantidad de proyectos.
- **Orientadas a columnas:** Este tipo está pensado para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenan columnas de datos en vez de registros.

- **Clave - Valor:** Son las más sencillas. Una base de datos de este tipo guarda tuplas que contienen una clave y su valor. Para recuperar un dato, simplemente se debe buscar por su clave y recuperar su valor.
- **En Grafo:** Este tipo de bases de datos están basadas en la teoría de grafos y utilizan los nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como puede ser una red social.

Las bases de datos NoSQL cuentan con una serie de ventajas [11]:

- Cuenta con una alta escalabilidad gracias a su naturaleza descentralizada.
- Son más abiertas y flexibles a diferentes tipos de datos que las bases de datos relacionales.
- No necesita altos recursos para su implementación.
- Presentan una buena escalabilidad horizontal, ya que son capaces de crecer en número de máquinas fácilmente.
- No impone una estructura de datos y puede almacenarlos en diferentes formatos.
- Están preparadas para grandes volúmenes de información.

No obstante, presenta una serie de contras a tener en cuenta:

- No pueden garantizar la consistencia, la disponibilidad y la tolerancia a fallos a la vez.
- No hay una estandarización y diferentes compañías poseen su propia solución.
- No suelen tener muchas herramientas de monitorización y casi todo su mantenimiento se realiza por consola.

2.4. Plataformas IoT

Una plataforma IoT es una herramienta software encargada de la recepción y gestión de los datos obtenidos por los sensores de los dispositivos IoT. Esta herramienta suele encargarse de las funciones principales del sistema en tiempo real, almacenando los datos obtenidos en bases de datos y ofreciéndoselos a las aplicaciones encargadas de su visualización. Además, se encarga de controlar las acciones realizadas por los dispositivos según los datos obtenidos, automatizando el sistema.

Una plataforma IoT posee las siguientes propiedades [12]:

- **Conectividad y normalización:** Debe poder garantizar la interacción con el mayor número de dispositivos existentes, soportando diferentes protocolos y diferentes formatos de datos.
- **Gestión de dispositivos:** Debe poder asegurar que los dispositivos están funcionando correctamente.
- **Almacenamiento de los datos:** Debe poder almacenar los datos en diferentes bases de datos.

- **Procesamiento y gestión de la acción:** Debe poder aportar datos basados en una serie de reglas de acción, que disparen eventos que permitan la ejecución de acciones inteligentes en función de los datos obtenidos de los dispositivos.
- **Visualización:** Deben aportar herramientas para que los usuarios puedan visualizar los datos o permitir la conexión con aplicaciones que se encarguen de dicha visualización.

2.5. Sistema a desarrollar

El objetivo de este Trabajo Fin de Grado es diseñar y desarrollar un sistema de información que capture, almacene y gestione los datos generados por dispositivos IoT desplegados por el CUM. Además, se deberá habilitar una aplicación de visualización en la que se muestre estos datos a los usuarios del sistema. Por lo tanto, para realizar este proyecto se necesitará:

- Una serie de dispositivos IoT con distintos sensores, que serán los encargados de recopilar los datos del entorno. Estos dispositivos estarán colocados por las distintas instalaciones del CUM. El sistema debe ser independiente de los dispositivos utilizados, por lo que cualquiera de ellos puede ser válido. Estos dispositivos deben contar con algún módulo de comunicación para poder enviar los datos a la red.
- En el sistema se usarán distintas tecnologías de red para comunicar los dispositivos IoT con la plataforma IoT.
- Una plataforma IoT que se encargue de la gestión de la información obtenida de los distintos dispositivos. Esta plataforma obtendrá los datos a través de la red, los analizará y según una serie de reglas programadas, se disparará alguna de las funciones que activen una actuación por parte de los dispositivos, como puede ser la orden de encender la luz. Además, esta plataforma debe poder comunicarse con una base de datos en la cual almacenará la información obtenida. También es necesario que proporcione los datos a una aplicación de visualización para que el usuario pueda consultarlos.
- El sistema necesita una base de datos distribuida, orientada a documentos y que posibilite el tratamiento de datos en tiempo real. Esta almacenará la información proporcionada por la plataforma IoT.
- Una aplicación web encargada de mostrar los datos obtenidos de manera sencilla y amena al usuario. Esta aplicación web se comunicará con la plataforma IoT para obtenerlos y los mostrará en distintos formatos, como puede ser en gráficas, históricos, imágenes, entre otros. Además, esta aplicación deberá ser responsive, por lo que se adaptará al dispositivo que acceda a ella.
- El sistema contará con distintos protocolos de comunicación para que los dispositivos IoT puedan compartir los datos obtenidos, puedan recibir información por parte de la plataforma IoT, y esta pueda compartir la información con la base de datos y la aplicación de visualización.

En la figura 2.5, se puede observar un esquema del sistema a desarrollar en este Trabajo Fin de Grado, con las distintas partes anteriormente nombradas.

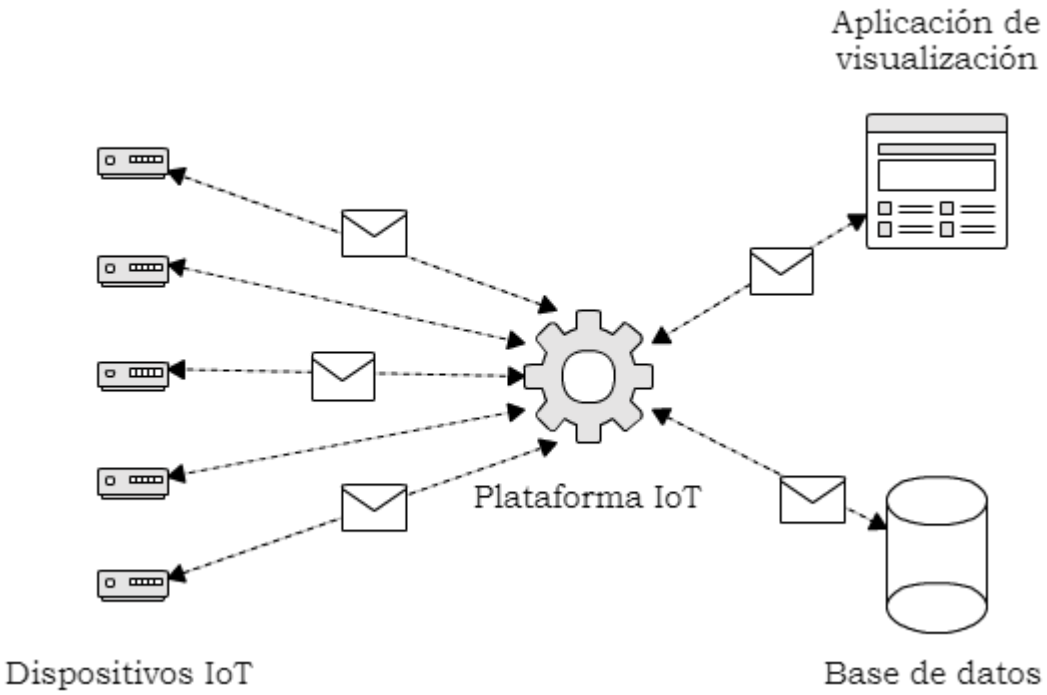

Figura 2.5: Esquema del sistema a desarrollar.

Capítulo 3

Metodología

Contenidos

En este capítulo se describe la metodología utilizada en este proyecto. Para la estructuración, planificación, y control del proyecto se aplicará una metodología iterativa e incremental, y para el desarrollo del proyecto se utilizará una metodología orientada a objetos.

En la sección 3.1 se introducirá el concepto de metodología iterativo e incremental.

En la sección 3.2 se realizará una breve explicación de la metodología orientada a objetos.

En la sección 3.3 se mostrará un esquema de las metodologías utilizadas en el proyecto.

3.1. Metodología iterativa e incremental

Durante el desarrollo de este Trabajo Fin de Grado, se empleará una metodología iterativa e incremental como marco de trabajo para estructurar, planificar y controlar el proceso de desarrollo del sistema.

3.1.1. Introducción

La metodología iterativa e incremental [13] divide el proyecto en diferentes bloques temporales llamados iteraciones. En todas las iteraciones se repite un proceso de trabajo similar para proporcionar un resultado completo, obteniendo una versión funcional de la aplicación. De esta forma, el sistema se desarrolla poco a poco y obtiene una retroalimentación continua por parte del cliente, en este caso el director del Trabajo Fin de Grado.

En esta metodología, cada requisito se debe completar en una única iteración, debiendo realizar el equipo de trabajo todas las tareas necesarias para completarlo, incluyendo pruebas

y documentación. En la figura 3.1, se puede observa el modelo de ciclo de vida iterativo e incremental.

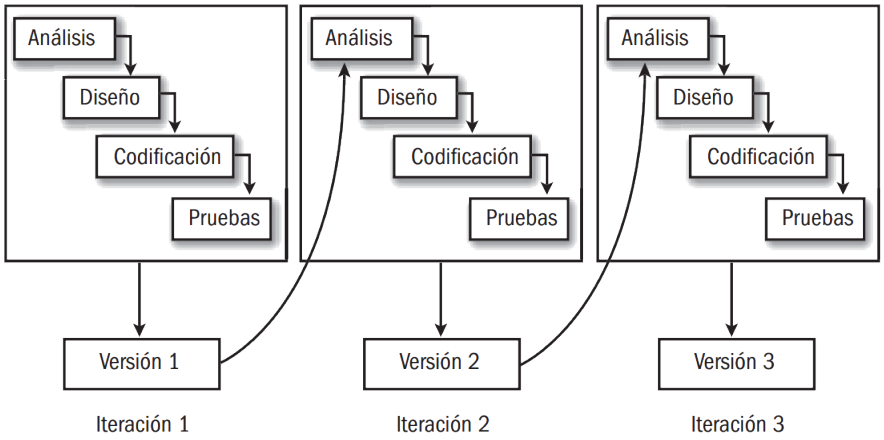

Figura 3.1: Modelo de ciclo de vida iterativo e incremental.

En cada iteración el equipo de trabajo va evolucionando el producto a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos o mejorando los que ya fueron completados.

Un aspecto fundamental de esta metodología es priorizar los objetivos o requisitos en función del valor que aportan al cliente.

3.1.2. Etapas del modelo iterativo e incremental

La metodología iterativa e incremental, aunque debe adaptarse a las características del proyecto, existen una serie de fases que se deben tener en cuenta a la hora de implementarla [14]:

1. **Requisitos:** En esta fase se identifica los objetivos centrales y específicos que persigue el proyecto.

2. **Definición de las tareas y las iteraciones:** Teniendo en cuenta el objetivo, el siguiente paso es hacer una lista de tareas y agruparlas en iteraciones que tendrá el proyecto.

3. **Diseño de los incrementos:** Una vez establecidas las iteraciones, es necesario definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe generar un producto más evolucionado respecto a la anterior.

4. **Desarrollo del incremento:** En esta fase se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior. Para su desarrollo se utilizará una metodología orientada a objetos. Cada incremento consta de las siguientes fases[15]:

   a) **Análisis:** En esta fase se debe analizar y comprender los requisitos del sistema y de la iteración.

   b) **Diseño:** En esta fase se han de estudiar las posibles alternativas de implementación.

   c) **Implementación:** Una vez realizado el análisis y el diseño de la iteración, se debe comprender correctamente el problema y se realiza la implementación.

d) **Pruebas:** Esta fase tiene como objetivo detectar los posibles errores cometidos en las fases anteriores y corregirlos.

5. **Validación de los incrementos:** Al finalizar cada iteración, se debe validar el incremento realizado. Si los resultados no son los esperados es necesario volver al principio del incremento y buscar las causas de esto.

6. **Integración de incrementos:** Una vez que son validados, los incrementos dan forma a la denominada línea incremental o evolución del proyecto en conjunto. Cada incremento ha contribuido al resultado final.

7. **Entrega del producto:** Una vez que el producto en su conjunto ha sido validado, se procede a su entrega final al cliente.

3.1.3. Ventajas y desventajas del modelo iterativo e incremental

Utilizar el modelo iterativo e incremental en el proyecto tiene una serie de beneficios[13]:

- Se puede gestionar las expectativas del cliente de manera regular, pudiendo tomar decisiones en cada iteración.
- No es necesario conocer al principio del proyecto todos los requisitos, únicamente será necesario conocer con detalle los requisitos de las primeras iteraciones y los requisitos de alto nivel que se irán refinando en las diferentes iteraciones.
- El cliente puede obtener resultados importantes y usables desde las primeras iteraciones.
- Se pueden gestionar de manera natural los cambios que van apareciendo durante el proyecto.
- Se pueden identificar riesgos desde el inicio del proyecto, debiendo gestionar los problemas desde la primera iteración.
- En este modelo se minimiza el número de errores que se producen durante el desarrollo y aumenta la calidad del proyecto.

Aunque este modelo presenta también una serie de restricciones[13]:

- El cliente debe estar disponible durante el proyecto participando de manera continua.
- Se necesita una relación más cercada al cliente, basándose en los principios de colaboración y ganar/ganar.
- Cada resultado de una iteración debe ser funcional.
- Es necesario tener conocimientos, herramientas y experiencia en este modelo para poder realizar cambios fácilmente.

3.2. Metodología orientada a objetos

El desarrollo de este proyecto se realizará con una metodología orientada a objetos[16]. Esta metodología pretende ayudar al desarrollador del sistema a explotar el poder de los lenguajes de programación orientados a objetos, utilizando las clases y objetos como bloques de construcción básicos.

El objetivo del uso de esta metodología consiste en analizar los diferentes usuarios que existirán en el sistema, como actuarán con él y que funciones se espera que pueda cumplir.

A continuación, se describe la etapa de análisis, uno de los aspectos fundamentales a destacar de la metodología orientada a objetos.

3.2.1. Análisis

El objetivo de este proceso es analizar el problema, llegando hasta una especificación completa y minuciosa del comportamiento del mismo, y consistencia, en la que se especifican las características funcionales y operacionales del mismo [17].

Este proceso debe dar información de las necesidades del sistema, cuáles van a ser las funciones del sistema sin introducir información sobre su implementación, y mostrar una visión completa del sistema y de lo que este debe hacer.

El fin del análisis orientado a objetos es desarrollar una serie de modelos que describan el sistema para satisfacer un conjunto de requerimientos definidos por el usuario.

3.2.1.1. Diagrama de casos de uso

La función del diagrama de casos de uso [17, 18] es documentar el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto, los casos de uso determinan los requisitos funcionales del sistema, representando las funciones que el sistema puede realizar.

Su principal ventaja es la facilidad que tienen para ser interpretados, logrando que la comunicación con el cliente sea más fácil.

En el próximo capítulo, en este apartado, se mostrará los casos de uso del sistema de forma reducida, después de forma extendida, y tras ello se describirán individualmente mediante tablas.

Los elementos básicos de los casos de uso de forma reducida son:

- **Actores:** Representan un tipo de usuario del sistema. Un usuario es cualquier persona, animal o cosa externa que interactúa con el sistema. Se representa mediante la figura de un muñeco.

- **Casos de uso:** Son las acciones o tareas que se realizan tras una orden de uno o varios actores en el sistema con un objetivo determinado.

- **Asociaciones:** Son las relaciones entre los actores y los casos de uso cuando el usuario interactúa con el sistema.

La tabla 3.1 representa el formato que siguen las tablas que describen los casos de uso específicos.

Código del caso de uso	Código al que corresponde el caso de uso de la lista
Nombre del caso de uso	Nombre completo del caso de uso
Descripción	Descripción del caso de uso
Actores	Actores que realizan la acción
Precondiciones	Condiciones que deben cumplirse para poder realizar la acción.
Flujo normal	Etapas que se realizan para el desarrollo de la función.
Flujo alternativo	Camino alternativo para realizar la función
Postcondiciones	Resultado que debe cumplirse tras la ejecución de la función.

Tabla 3.1: Formato de las tablas de los casos de uso específicos.

3.3. Esquema de las metodologías utilizadas

En la figura 3.2, se muestra un esquema de las metodologías utilizadas en el proyecto para el desarrollo del sistema.

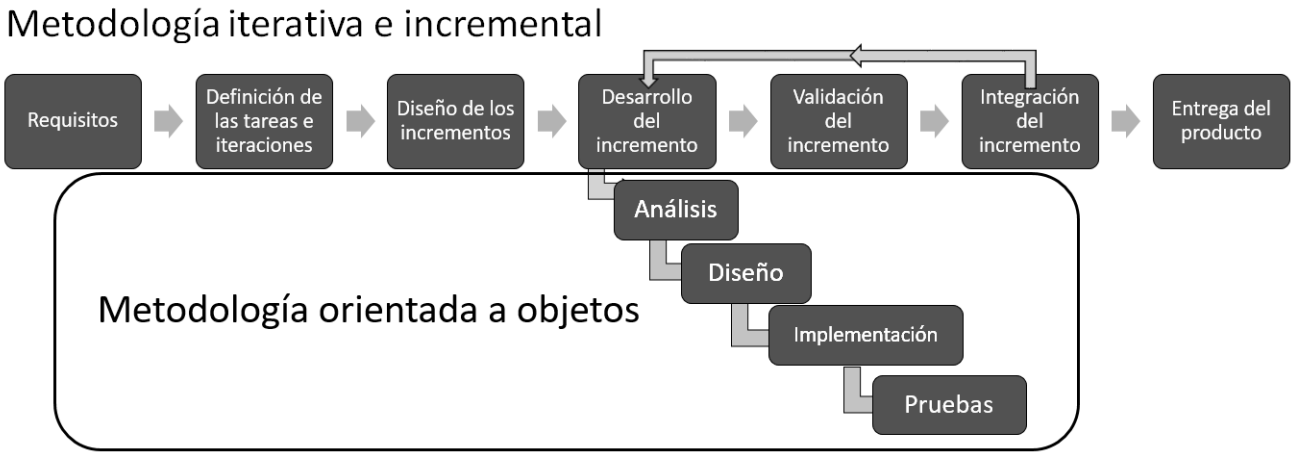

Figura 3.2: Esquema de las metodologías utilizadas en el proyecto.

Capítulo 4

Análisis del sistema

Contenidos

En este capítulo se detalla el análisis del sistema propuesto, exponiendo de forma pormenorizada su comportamiento, consistencia y viabilidad, así como las características funcionales y operacionales.

En la sección 4.1, se realizará un estudio de viabilidad del proyecto, analizando la viabilidad técnica, operacional y económica.

En la sección 4.2, se mostrará la planificación del proyecto, realizando un cronograma formado por los distintos paquetes de trabajo en los que se ha dividido el proyecto, con las fechas y duraciones estimadas de cada uno de ellos.

En la sección 4.3, se presenta el análisis funcional del sistema, analizando en profundidad cada una de las operaciones que ofrece.

En la sección 4.4, se describen los diagramas de casos de uso.

4.1. Estudio de viabilidad

En este apartado se realizará un estudio de viabilidad, analizando la viabilidad técnica, la viabilidad operacional y la viabilidad económica del proyecto.

4.1.1. Viabilidad técnica

Para la realización del proyecto es necesario tener conocimientos sobre los distintos dispositivos IoT para la captura de datos mediante sensores y realizar acciones mediante actuadores.

Además, es necesario conocer el funcionamiento de las aplicaciones utilizadas como plataforma IoT, base de datos y software de visualización.

Todos estos apartados poseen numerosa documentación, facilitando su realización y la solución de posibles problemas y cuestiones, por lo que es viable su realización desde el punto de vista técnico.

4.1.2. Viabilidad operacional

El usuario final únicamente interaccionará con la aplicación web elegida como software de visualización de los datos recogidos por los distintos dispositivos IoT. Este software debe ser responsive, por lo que será adaptable a cualquier dispositivo, necesitando solamente un navegador web.

Es obligatorio, para el proyecto, que los dispositivos IoT como las distintas aplicaciones software están siempre en funcionamiento, para que la aplicación web esté disponible y se visualicen correctamente los datos.

4.1.3. Viabilidad económica

Para analizar la viabilidad económica del proyecto se realizará un estudio del coste de los distintos dispositivos IoT que se utilizarán, que serán los vistos en el capítulo anterior, y el coste del desarrollo del proyecto.

En la tabla 4.1, se muestra el coste de los distintos dispositivos que se utilizarán.

Dispositivo	**Precio**
Arduino UNO Elegoo kit de iniciación Arduino UNO	55,99€
Raspberry Pi 2 model B	45,85€
NodeMCU v3	8,99€
Raspberry Pi Module Camera v2	25€
TOTAL	**138,07€**

Tabla 4.1: Coste de dispositivos IoT.

El kit de iniciación de Arduino cuenta con una gran cantidad de sensores y actuadores, que serán los utilizados en el proyecto. Las aplicaciones software que se utilizarán serán libres y gratuitas, no suponiendo coste ninguno.

En la tabla 4.2, se muestra el coste del desarrollo del proyecto.

	Horas	**Salario/Hora**	**Precio**
Análisis del proyecto	70	20€	1.400€
Diseño del proyecto	100	18€	1.800€
Despliegue del sistema	70	15€	1.050€
Pruebas	10	18€	180€
Informes	50	18€	900€
TOTAL	**300**		**5.330€**

Tabla 4.2: Coste del desarrollo del proyecto.

Tras estos estudios, se puede observar que el coste total del proyecto junto con los dispositivos sería aproximadamente 5.468€.

El Centro Universitario de Mérida gasta 85.340€ aproximadamente de luz al año[1]. Según un estudio realizado por *Consumer Technology Association*[2], la automatización de edificios a través del Internet de las Cosas puede suponer un ahorro de un 10 % del consumo de energía. Por lo tanto, si se aplica este estudio y se instala el sistema en todas las instalaciones, el Centro Universitario de Mérida podría ahorrar hasta 8.534€ anuales.

4.1.4. Viabilidad de la solución

A partir de los análisis anteriores, se puede concluir que la realización del proyecto es viable técnicamente, operacionalmente y económicamente. Los requerimientos de los usuarios se cumplen, económicamente es asumible y técnicamente hace uso de una tecnología asentada, con bastante popularidad y documentación.

4.2. Planificación estimada

Se ha realizado un cronograma (figura 4.1) en el cual se divide el proyecto en diferentes paquetes de trabajo. Se ha estimado el tiempo de realización de cada una de ellos, así como el periodo de tiempo en el que se espera completarlas.

Los paquetes de trabajo son los siguientes:

- Analizar bases de datos distribuidas, orientadas a documentos y de tiempo real (Tiempo estimado: 20 horas):
  - Buscar diferentes bases de datos que cumplan estas condiciones.
  - Analizar cada una de ellas.
  - Seleccionar la mejor opción para este proyecto.
- Analizar diferentes plataformas IoT (Tiempo estimado: 25 horas):
  - Buscar diferentes plataformas IoT.
  - Estudiar cada una de ellas y analizar la compatibilidad con el proyecto.
  - Seleccionar la mejor opción para el sistema.
- Estudiar posibles soluciones de visualización de datos procedentes de dispositivos IoT (Tiempo estimado: 25 horas):
  - Buscar diferentes soluciones de visualización.
  - Analizar cada una de ellas y su compatibilidad con el sistema a desarrollar.
  - Seleccionar la mejor opción para el proyecto.

[1] Dato proporcionado por la administración del centro

[2] Estudio disponible en http://smart-lighting.es/la-domotica-hogar-puede-ahorrar-costes-10-energia-segun-estudio-americano/

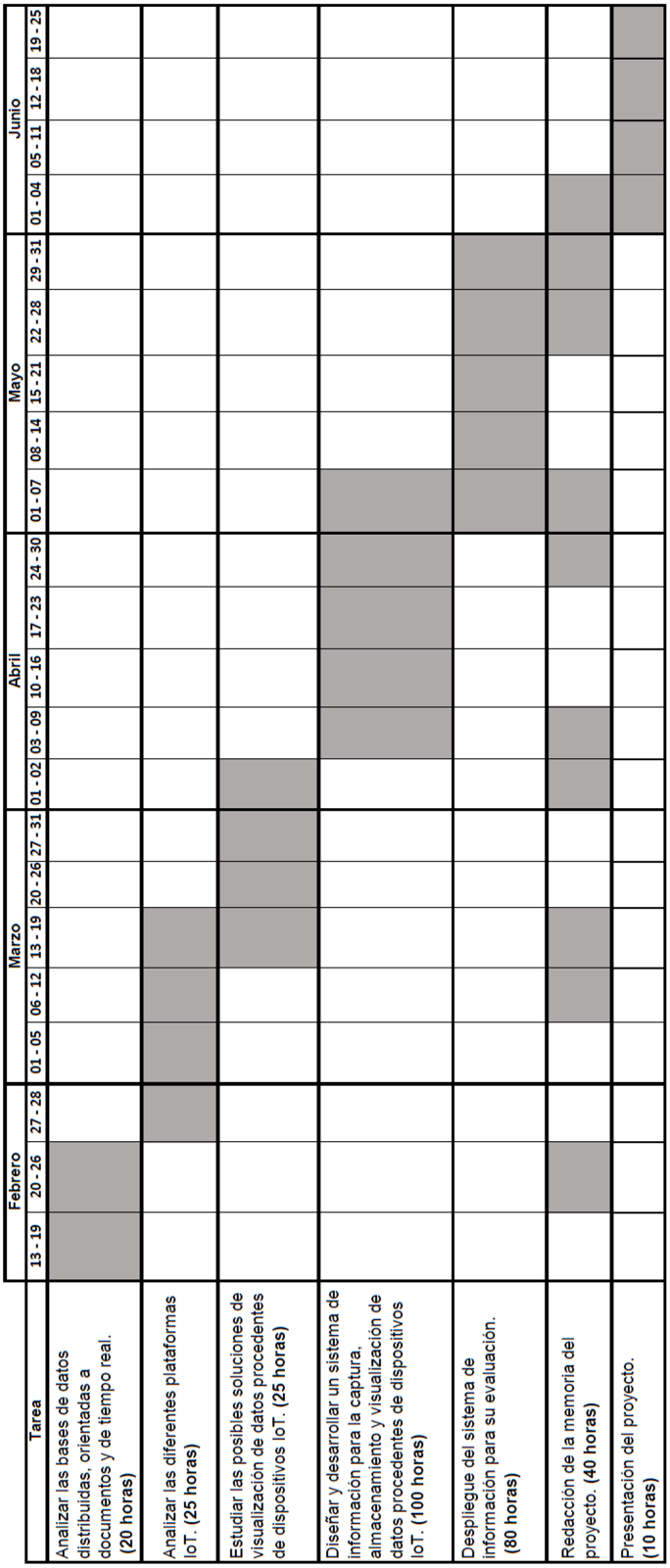

Figura 4.1: Cronograma estimado.

- Diseñar y desarrollar un sistema de información para la captura, almacenamiento y visualización de datos procedentes de dispositivos IoT (Tiempo estimado: 100 horas):
  - Analizar los distintos dispositivos IoT utilizados para el proyecto.
  - Analizar distintos protocolos de comunicación para que los dispositivos puedan comunicarse con la plataforma IoT.
  - Diseñar la conexión de todos los elementos del sistema: dispositivos, plataforma, base de datos y aplicación de visualización.
  - Diseñar la interfaz de visualización de datos.
  - Establecer las reglas de automatización.
  - Desarrollar el sistema, conectando todos los elementos y probando su funcionamiento.
  - Realizar distintas pruebas para ver que todo funciona correctamente.
- Despliegue del sistema de información para su evaluación (Tiempo estimado: 80 horas):
  - Analizar y diseñar la colocación de los dispositivos IoT en las instalaciones permitidas por el centro.
  - Colocar los dispositivos IoT.
  - Instalar el sistema diseñado en el centro.
  - Comprobar el correcto funcionamiento del sistema desplegado en el centro.
- Redacción de la memoria del proyecto (Tiempo estimado: 40 horas):
  - A medida que se realizan los anteriores paquetes de trabajo se irá completando la memoria del proyecto.
- Presentación del proyecto (Tiempo estimado: 10 horas):
  - Al finalizar el proyecto, se realizará una presentación para exponer este Trabajo Fin de Grado.

4.3. Análisis funcional

En esta sección se realiza un estudio de las funcionalidades del sistema, utilizando las distintas metodologías expuestas en los puntos del capítulo anterior. Se analizarán los distintos tipos de usuarios que podrán interaccionar con el sistema y cuáles son las distintas acciones que pueden realizar.

Se observa la necesidad de dos actores principales:

- **Administrador:** Será el encargado de la gestión y el mantenimiento del sistema.
- **Visitante:** Serán aquellos miembros de la comunidad educativa y otras personas que deseen consultar la información proporcionada por los distintos dispositivos IoT desplegados a través de la aplicación de visualización.

4.3.1. Casos de uso globales

En la figura 4.2, se muestran los casos de uso del actor Visitante.

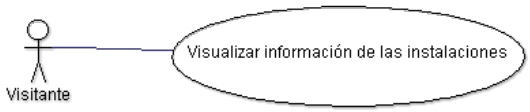

Figura 4.2: Casos de uso del actor Visitante.

En la figura 4.3, se muestran los casos de uso principales a destacar del actor Administrador.

Figura 4.3: Casos de uso principales a destacar del actor Administrador.

4.4. Diagrama de casos de uso detallados

Una vez definido los diagramas de uso del sistema, en este apartado se realizará un análisis de los requisitos de cada uno de ellos.

Código del caso de uso	VII
Nombre del caso de uso	Visualizar información de las instalaciones
Descripción	Permite a los usuarios del sistema consultar la información proporcionada por los distintos dispositivos IoT desplegados por las instalaciones del centro.
Actores	Administrador y Visitante.
Precondiciones	La aplicación web debe estar disponible para el acceso de los usuarios.
Flujo normal	1. Acceder a la aplicación web de visualización. 2. Seleccionar la instalación de la que se desea consultar la información.
Flujo alternativo	
Postcondiciones	Se mostrará la información de la instalación del centro acompañada con distintas gráficas e imágenes.

Tabla 4.3: Visualizar información de las instalaciones

Código del caso de uso	ND
Nombre del caso de uso	Dar de alta un nuevo dispositivo
Descripción	Permite al administrador dar de alta a un nuevo dispositivo IoT en el sistema.
Actores	Administrador.
Precondiciones	Se necesita tener instalado en el equipo un software que permita cargar el programa en el dispositivo.
Flujo normal	1. Acceder al software de programación del dispositivo. 2. Escribir el código para las acciones que tenga que realizar el dispositivo. 3. Escribir la configuración de la red y del dispositivo. 4. Establecer el modo de comunicación del dispositivo con la plataforma IoT. 5. Cargar el programa realizado en el dispositivo. 6. Guardar el programa realizado. 7. Instalar el dispositivo en la instalación elegida del centro.
Flujo alternativo	1. No se ha programado correctamente el código, por lo que se le avisará al usuario del error. 2. No se ha cargado correctamente el programa en el dispositivo, por lo que se le avisará al usuario del error.
Postcondiciones	El dispositivo comenzará a realizar las acciones para las que ha sido programado y se comunicará con la plataforma IoT.

Tabla 4.4: Dar de alta a un nuevo dispositivo

Código del caso de uso	MD
Nombre del caso de uso	Modificar un dispositivo
Descripción	Permite al administrador modificar un dispositivo IoT del sistema.
Actores	Administrador.
Precondiciones	Se necesita tener instalado en el equipo un software que permita cargar el programa en el dispositivo.
Flujo normal	1. Acceder al software de programación del dispositivo. 2. Seleccionar el programa que se realizó para configurar el dispositivo. 3. Modificar las partes deseadas del programa. 4. Coger el dispositivo IoT a modificar y conectarlo al equipo. 5. Cargar el programa realizado en el dispositivo. 6. Guardar el programa realizado. 7. Instalar el dispositivo en la instalación elegida del centro.
Flujo alternativo	1. No se ha programado correctamente el código, por lo que se le avisará al usuario del error. 2. No se ha cargado correctamente el programa en el dispositivo, por lo que se le avisará al usuario del error.
Postcondiciones	El dispositivo comenzará a realizar las acciones para las que ha sido programado con las nuevas modificaciones y se comunicará con la plataforma IoT.

Tabla 4.5: Modificar un dispositivo

Código del caso de uso	ARU
Nombre del caso de uso	Añadir una nueva regla de automatización.
Descripción	Permite al administrador añadir una nueva regla de automatización al sistema.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la plataforma IoT.
Flujo normal	1. Acceder al software de configuración de la plataforma IoT. 2. Seleccionar la opción de añadir una nueva regla o función. 3. Programar la nueva regla. 4. Pulsar en guardar.
Flujo alternativo	1. No se ha programado correctamente la regla, por lo que se le avisará al usuario del error.
Postcondiciones	La nueva regla estará activada y se disparará al cumplir una serie de condiciones.

Tabla 4.6: Añadir una nueva regla de automatización

Código del caso de uso	BRU
Nombre del caso de uso	Borrar una regla de automatización.
Descripción	Permite al administrador eliminar una regla de automatización del sistema.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la plataforma IoT.
Flujo normal	1. Acceder al software de configuración de la plataforma IoT. 2. Seleccionar la regla a eliminar. 3. Pulsar en eliminar regla o función. 4. Confirmar el borrado de la regla del sistema.
Flujo alternativo	1. No se ha confirmado el borrado, por lo que la regla seguirá en el sistema.
Postcondiciones	La regla ha sido eliminada y ya no disparará una acción al cumplir una serie de condiciones.

Tabla 4.7: Borrar una regla de automatización

Código del caso de uso	MRU
Nombre del caso de uso	Modificar una regla de automatización.
Descripción	Permite al administrador modificar una regla de automatización del sistema.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la plataforma IoT.
Flujo normal	1. Acceder al software de configuración de la plataforma IoT. 2. Seleccionar la regla o función a modificar. 3. Cambiar el código a modificar por el nuevo código. 4. Pulsar en guardar.
Flujo alternativo	1. No se ha programado correctamente la regla, por lo que se le avisará al usuario del error.
Postcondiciones	La regla modificada estará activada y se disparará al cumplir una serie de condiciones.

Tabla 4.8: Modificar una regla de automatización

Código del caso de uso	CT
Nombre del caso de uso	Crear una tabla.
Descripción	Permite al administrador crear una nueva tabla en la base de datos.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la base de datos del sistema.
Flujo normal	1. Acceder al software de configuración de la base de datos. 2. Seleccionar la opción de crear una nueva tabla. 3. Insertar el nombre de la tabla a crear. 4. Pulsar en guardar.
Flujo alternativo	1. No se ha escrito un nombre válido, por lo que se le avisará al usuario del error.
Postcondiciones	La nueva tabla estará creada y podrá almacenar información.

Tabla 4.9: Crear una tabla en la base de datos

Código del caso de uso	BT
Nombre del caso de uso	Borrar una tabla.
Descripción	Permite al administrador borrar una tabla de la base de datos.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la base de datos del sistema.
Flujo normal	1. Acceder al software de configuración de la base de datos. 2. Seleccionar la tabla a eliminar. 3. Seleccionar la opción eliminar. 4. Confirmar el borrado de la tabla.
Flujo alternativo	1. No se ha confirmado el borrado, por lo que la tabla seguirá en la base de datos.
Postcondiciones	La tabla y los datos almacenados en ella estarán borrados y no se podrán volver a consultar.

Tabla 4.10: Borrar una tabla de la base de datos

Código del caso de uso	AIM
Nombre del caso de uso	Añadir una nueva información a mostrar.
Descripción	Permite al administrador añadir una nueva información a mostrar en la aplicación de visualización.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la aplicación de visualización.
Flujo normal	1. Acceder al software de configuración de la aplicación de visualización. 2. Seleccionar la opción de agregar un nuevo dato. 3. Seleccionar el dato a mostrar y la forma de mostrarlo (en gráficas, numérico...). 4. Pulsar en guardar.
Flujo alternativo	1. No se ha seleccionado todos las opciones necesarias para mostrar la información, por lo que se avisará al usuario del error.
Postcondiciones	La nueva información se mostrará a los usuarios en la aplicación de visualización.

Tabla 4.11: Añadir una nueva información a mostrar

Código del caso de uso	MIM
Nombre del caso de uso	Modificar una información mostrada.
Descripción	Permite al administrador modificar una información mostrada en la aplicación de visualización.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la aplicación de visualización.
Flujo normal	1. Acceder al software de configuración de la aplicación de visualización. 2. Seleccionar la información a modificar. 3. Pulsar en la opción modificar. 4. Modificar el dato a mostrar o la forma de mostrarlo. 5. Pulsar en guardar.
Flujo alternativo	1. No se ha seleccionado todos las opciones necesarias para mostrar la información, por lo que se avisará al usuario del error.
Postcondiciones	La información modificada se mostrará a los usuarios en la aplicación de visualización.

Tabla 4.12: Modificar una información mostrada

Código del caso de uso	BIM
Nombre del caso de uso	Borrar una información mostrada.
Descripción	Permite al administrador borrar una información mostrada en la aplicación de visualización.
Actores	Administrador.
Precondiciones	Se necesita tener acceso a la configuración de la aplicación de visualización.
Flujo normal	1. Acceder al software de configuración de la aplicación de visualización. 2. Seleccionar la información a borrar. 3. Pulsar en la opción borrar. 4. Confirmar la eliminación de la información mostrada.
Flujo alternativo	1. No se ha confirmado el borrado, por lo que la información seguirá visible a los usuarios.
Postcondiciones	La información borrada ya no estará disponible para los usuarios de la aplicación.

Tabla 4.13: Borrar una información mostrada

Capítulo 5

Diseño

Contenidos

Tras establecer las distintas funcionalidades del sistema, en este capítulo se realizará su diseño. Para ello, se analizarán las distintas partes que forman el proyecto, viendo las alternativas más destacables entre las existentes.

En la sección 5.1 se estudiarán los distintos dispositivos IoT disponibles para la realización del proyecto.

En la sección 5.2 se realizará un estudio de diferentes herramientas IoT para la gestión de la aplicación a realizar y se seleccionará la plataforma IoT más conveniente para este proyecto.

En la sección 5.3 se analizarán diferentes bases de datos distribuidas, orientadas a documentos y de tiempo real, seleccionando una de ellas para la realización del proyecto.

En la sección 5.4 se estudiará las posibles soluciones de visualización de datos procedentes de dispositivos IoT, seleccionando la aplicación más adecuada para el sistema.

En la sección 5.5 se introducirán distintos protocolos de comunicación, seleccionando los protocolos necesarios para el proyecto.

En la sección 5.6 se mostrará el sistema completo diseñado para este proyecto.

5.1. Dispositivos IoT

Para la realización de este proyecto se necesitará distintos dispositivos IoT, que serán los encargados de recopilar la información, transmitirla a la red, y actuar en función de la información obtenida. Este proyecto será independiente de los dispositivos usados, por lo que se podrá utilizar cualquier dispositivo sin ningún problema.

5.1.1. Arduino

Para definir correctamente Arduino [6, 19], es necesario analizar las tres partes que lo componen:

- Una **placa hardware** libre que incorpora un microcontrolador reprogramable y una serie de pines unidos internamente a las patillas de entrada/salida del microcontrolador, que permiten conectar los diferentes sensores y actuadores. Existen numerosos modelos oficiales de placas Arduino, cada una con diferentes características, como el tamaño, el número de pines o el modelo de microcontrolador incorporado. Todos los microcontroladores son de tipo AVR, una arquitectura de microcontroladores desarrollada y fabricada por la marca Atmel. En la figura 6.4, se puede observar el Arduino UNO, la placa estándar de Arduino.

- Un **software libre y multiplataforma.** Más concretamente, un entorno de desarrollo que funciona en diferentes sistemas operativos, como Linux, MacOS o Windows. Este software permite escribir, verificar y guardar en la memoria del microcontrolador de la placa el conjunto de instrucciones que se desea ejecutar. La manera estándar de conectar un computador con la placa para poder enviarle y grabarle dichas instrucciones es mediante un cable USB, ya que la mayoría de las placas Arduino incorporan un conector de este tipo.

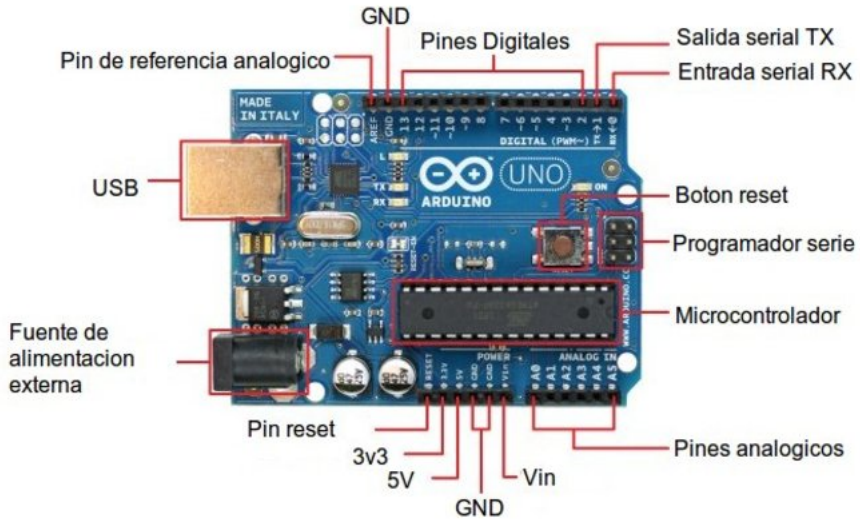

Figura 5.1: Arduino UNO.

- **Un lenguaje de programación libre.** Dentro del lenguaje Arduino se puede encontrar elementos parecidos a otros lenguajes de programación existentes, como bloques condicionales, bloques repetitivos o variables, así como diferentes comandos que permiten especificar de forma coherente y sin errores las instrucciones exactas que se desea programar en el microcontrolador de la placa.

El diseño de la placa Arduino está inspirado originalmente en el de otra placa hardware libre existente, la placa Wiring. Además, tanto el lenguaje de programación como el entorno de desarrollo están basados en otro entorno y lenguaje existente llamado Processing. Tanto Wiring como Processing fueron diseñados en el mismo centro que Arduino.

Con Arduino se pueden realizar multitud de proyectos, desde robótica hasta domótica, pasando por la monitorización de sensores ambientales, sistemas de navegación, proyectos telemáticos, entre otros.

Arduino presenta las siguientes características:

- **Arduino es libre y extensible:** esto significa que cualquiera puede ampliar y mejorar tanto el diseño hardware de las placas como el entorno de desarrollo software y el propio lenguaje de programación. Hay multitud de extensiones no oficiales, tanto de las placas como de las librerías software, que permiten la adaptación de Arduino a diferentes necesidades.

- **Arduino tiene una gran comunidad:** un gran número de personas lo utilizan, mejoran la documentación existente y comparten continuamente sus ideas.

- **Su entorno de programación es multiplataforma:** se puede instalar y ejecutar en sistemas Windows, Mac OS X y Linux.

- **Su entorno y el lenguaje de programación son simples y claros:** son fáciles de aprender y de utilizar, además de ser flexibles y completo para que los usuarios avanzados puedan aprovechar al máximo todas las posibilidades del hardware. También

están bien documentados, con bastantes ejemplos detallados y gran cantidad de proyectos publicados.

- **Las placas Arduino son baratas:** la placa Arduino UNO, cuesta alrededor de 20 euros.
- **Las placas Arduino son reutilizables y versátiles:** la placa se puede aprovechar para distintos proyectos gracias a la facilidad que presenta reprogramarla. Además, son versátiles porque proveen varios tipos diferentes de entradas y salidas de datos, las cuales permiten capturar información de sensores y enviar señales a actuadores de distintas formas.

En la tabla 5.1, se comparan las especificaciones técnicas de los tipos de Arduino más conocidos:

Nombre	Microcontrolador	Voltaje	Frecuencia de Reloj	E/S digitales	E/S analógicas
Arduino Due	AT91SAM3X8E	3,3V	84 MHz	54	12
Arduino Leonardo	ATmega32U4	5V	16 MHz	20	12
Arduino UNO	ATmega328	5V	16 MHz	14	6
Arduino Mega	ATmega2560	5V	16 MHz	54	16
Arduino Mini	ATmega328	5V	16 MHz	14	6
Arduino Pro Micro	ATmega32U4	5V	16 MHz	12	4

Tabla 5.1: Especificaciones técnicas de diferentes tipos de Arduino.

5.1.2. Raspberry Pi

Raspberry Pi [20, 21] es un ordenador de placa reducida o SBC de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de fomentar la enseñanza de la computación y la programación en las escuelas.

Raspberry Pi, en definitiva, es una placa de pequeño tamaño con un microprocesador ARM integrado con un chip Broadcom. En su primera versión, Raspberry Pi modelo A, cuenta con un microprocesador de 700MHz, con 256MB de RAM, una GPU VideoCore IV y todo lo necesario para ejecutar programas básicos, navegar por Internet e incluso programar. Únicamente no cuenta con disco duro, por lo que es necesario adquirir una tarjeta SD para el almacenamiento.

En estos años han sacado nuevos modelos y han realizado revisiones de algunos de sus modelos antiguos, como por ejemplo una nueva versión de Raspberry Pi modelo A, llamada Raspberry Pi modelo A+. En la figura 5.2, se puede observar la Raspberry Pi 3 modelo B junto con los distintos componentes que la forman.

El hardware de Raspberry Pi es de propiedad privada pero de uso libre, por lo que mantienen el control de la plataforma pero permiten el uso libre tanto a nivel educativo como particular. En la tabla 5.2, se puede observar una comparativa entre algunos de los distintos modelos de Raspberry Pi.

Los diferentes modelos de Raspberry Pi cuentan con puertos GPIO. Estos puertos permiten la comunicación de la placa con otros dispositivos mediante mecanismos de lectura y escritura de información en sus correspondientes pines, que se realiza por medio de algún lenguaje de programación como Python o C.

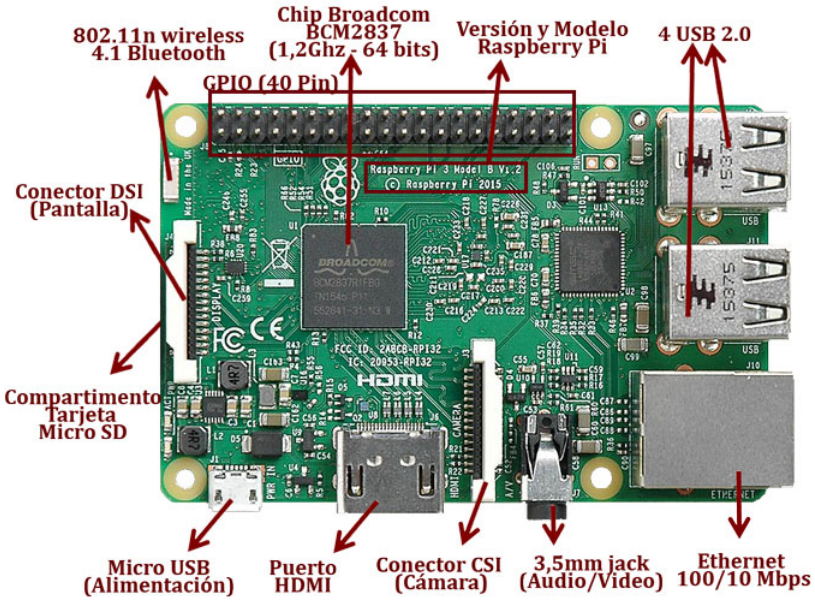

Figura 5.2: Raspberry Pi 3 modelo B.

Raspberry Pi	Modelo A	Modelo B	2 Modelo B	Zero	3 Modelo B	Zero W
SoC Broadcom	BCM2835	BCM2835	BCM2836	BCM2835	BCM2837	BCM2835
CPU	ARM1176JZF-S	ARM1176JZF-S	ARM Cortex-A7	ARM1176JZF-S	ARM Cortex-A53	ARM1176JZF-S
	700MHz	700MHz	900MHz Quad-core	1GHz	1,2GHz Quad-core	1GHz
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256MB	512MB	1GB	512MB	1GB	512MB
USB	1	2	4	1 Micro	4	1 Micro
Vídeo	RCA,HDMI	RCA,HDMI	Jack,HDMI	mini HDMI	Jack,HDMI	mini HDMI
Audio	Jack,HDMI	Jack,HDMI	Jack,HDMI	mini HDMI	Jack,HDMI	mini HDMI
Boot	SD	SD	microSD	microSD	microSD	microSD
ARed		Ethernet 10/100	Ethernet 10/100		Ethernet,Wifi,BT	Wifi,BT
Consumo	300mA/1,5W/5V	700mA/3,5W/5V	800mA/4W/5V	160mA/0,8W/5V	2,5A/12,5W/5V	160mA/0,8W/5V
Alimentación	MicroUSB/GPIO	MicroUSB/GPIO	MicroUSB/GPIO	MicroUSB/GPIO	MicroUSB/GPIO	MicroUSB/GPIO
Tamaño	85,6x53,98mm	85,6x53,98mm	85x56mm	65x30mm	85x56mm	65x30mm
Precio	~25€	~30€	~45€	~10€	~45€	~11€

Tabla 5.2: Especificaciones técnicas de diferentes modelos de Raspberry Pi.

Además, Raspberry Pi cuenta con numerosos accesorios que se han ido realizando estos años. En 2013 se puso a la venta un módulo de cámara de 5 megapíxeles y un módulo de cámara de infrarrojos. También, empresas ajenas a la Fundación Raspberry Pi han sacado distintos periféricos y carcasas compatibles con los distintos modelos. Un ejemplo destacable sería la Gertboard, que sirve para hacer uso de los puertos GPIO de la Raspberry Pi para poder interaccionar la placa con leds, interruptores, sensores y otros dispositivos. Además, incluye un controlador opcional para Arduino para poder interaccionar con la Raspberry Pi.

El software de Raspberry Pi es open source, siendo su sistema operativo oficial una versión

adaptada de Debian, denominada Raspbian, aunque permite otros sistemas operativos. Mayoritariamente Raspberry Pi utiliza sistemas operativos GNU/Linux, aunque también admite una versión de Windows 10.

Raspberry Pi cuenta además con una gran comunidad en varios idiomas, entre ellos el español, donde se comparte una gran cantidad de ejemplos, información y documentación, y se resuelven las dudas y problemas que pueden tener los distintos usuarios.

5.1.3. NodeMCU

NodeMCU [22, 23, 24] es una placa de desarrollo basada en el chip ESP8266 que revolucionó los sistemas embebidos. Con este módulo se puede realizar el prototipo de cualquier sistema para el IoT. El concepto es muy similar al de Arduino, un microcontrolador conectado a través de un puente USB-Serial que interactúa con un software en el ordenador. NodeMCU contiene el protocolo TCP/IP para proporcionar conexión Wifi y puede albergar su propia aplicación o servir como conexión entre cualquier microcontrolador e Internet.

El chip ESP8266 es un SoC[1] que integra en una sola pieza de silicio un procesador de aplicaciones con la electrónica necesaria para la comunicación por Wifi. Este chip tiene potentes capacidades de procesamiento y almacenamiento que le permiten integrarse con sensores y dispositivos específicos a través de sus puertos GPIOs. NodeMCU permite aprovechar el microprocesador que hay dentro del ESP8266 y realizar el software que se ejecutará en él. Además, todos los pines disponibles del ESP8266 están en el exterior, pudiendo colocar el NodeMCU en un protoboard. También incluye un conector miniUSB para programar el chip interno y comunicarse con el ordenador. En la figura 6.5, se puede observar la placa NodeMCU.

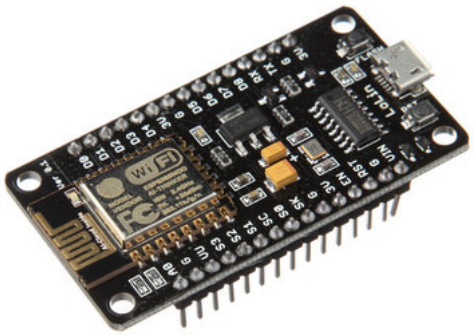

Figura 5.3: NodeMCU.

Existen tres versiones de NodeMCU:

- La primera versión de la placa NodeMCU destaca por tener un tamaño de 47x31mm, y la placa suele ser de color amarillo. Es más ancha que sus sucesoras y presenta problemas a la hora de colocarla en una protoboard debido a su tamaño, ya que no dejaba pines libres para su utilización, haciendo que su uso fuese muy engorroso.

[1]System on Chip o SoC: Chip con todos o gran parte de los módulos que componen un computador.

- La segunda versión de la placa solucionó el problema del tamaño, pudiendo ser utilizada en protoboards sin incomodidades.
- La tercera versión presenta una serie de innovaciones menores. El puerto USB pasa a ser más robusto y uno de los pines reservados pasa a ser utilizado para la alimentación USB de salida y otro de los pines reservados para una conexión GND adicional.

Utiliza el lenguaje de programación LUA para crear un ambiente de desarrollo propicio para aplicaciones que requieran conectividad Wifi de manera rápida. LUA es un lenguaje potente, eficiente y ligero, compatible con la programación procedimental y la programación orientada a objetos. Además, NodeMCU es compatible con el lenguaje Arduino, por lo que se podrá programar en Arduino sin necesidad de emplear un nuevo lenguaje.

Las características de NodeMCU con ESP8266 son las siguientes:

- **Procesador principal:** ESP8266.
- 160KB de RAM y 4MB de memoria flash externa.
- Consumo mínimo y opciones de suspensión (sleep mode y deep sleep mode).
- **Protocolo inalámbrico:** 802.11 b/g/n.
- **TCP/IP** integrado.
- **Sensor de temperatura** integrado.
- **Código abierto.**
- **Programable.**
- **Precio** bastante **bajo**, alrededor de 9 euros.
- Compatible con Arduino.

5.1.4. Sensores y módulos

Los dispositivos IoT contarán con distintos sensores encargados de obtener datos del entorno. A continuación, se introducirán distintos sensores que serán utilizados en el proyecto.

5.1.4.1. Sensor de temperatura y humedad del aire: DHT-11 y DHT-22

Los sensores de la familia DHT proporcionan de forma digital la temperatura y la humedad del aire con diferente precisión según el modelo utilizado. Esta familia se componen de dos variantes, el sensor DHT-11 y el DHT-22.

El sensor DHT-11 [25] es el sensor más básico de la familia para medir la temperatura y la humedad. Es muy barato, rondando su precio los 2€. Sus características son las siguientes:

- Funciona con una alimentación de 3.3V y 5 V.
- Puede medir temperaturas desde 0ºC a 50ºC sin ninguna cifra decimal.
- Puede medir la humedad desde el 20 % hasta el 80 % con un 5 % de precisión.

- Es capaz de obtener una muestra cada segundo.

El sensor DHT-22 [26] es un sensor digital encargado de obtener la temperatura y la humedad del aire. Se considera un sensor válido para proyectos caseros y semiprofesionales cuando no se requiera una medición constante ni milimétrica. Es un sensor bastante económico, con un precio en torno a 6€.

Este sensor posee las siguientes características:

- Necesita una alimentación entre 3.3V y 5V, tomando como valor recomendado 5V.
- Los valores tanto de humedad como de temperatura proporcionados cuentan con una cifra decimal.
- Tarda 2 segundos en obtener nuevos datos, por lo que solo podrá ofrecer información cada 2 segundos.
- Puede medir temperaturas entre -40ºC hasta 80ºC con una precisión de ±0.5ºC a ±1ºC y un tiempo de respuesta menor a 10 segundos. Esto quiere decir que refleja un cambio de temperatura real en el entorno al menos cada 10 segundos.
- Puede obtener la humedad desde un 0 % hasta un 99.9 % con una precisión de ±2 %RH, a una temperatura de 25ºC, y con un tiempo de respuesta inferior a 5 segundos.

En la figura 5.4, se puede observar el sensor DHT-22 con la información sobre sus pines.

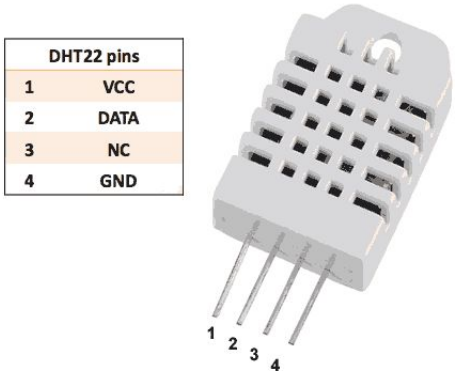

Figura 5.4: Sensor de temperatura y humedad del aire DHT-22.

5.1.4.2. Sensor de luz: Fotorresistencia LDR

Una fotorresistencia LDR [27] es un sensor de luz de tipo resistivo, por lo que es un tipo de resistencia especial que cambia su valor según la cantidad de luz que incide sobre ella. Según este valor de resistencia se puede calcular el porcentaje de luz existente.

El valor de una fotorresistencia LDR no varía de forma instantánea cuando se pasa de luz a oscuridad o al contrario, y el tiempo que dura este proceso no es siempre igual, por lo que no ofrece una una exactitud perfecta de cuando se produce un cambio de estado. Su tiempo de respuesta típico es aproximadamente 0.1 segundos.

En la figura 5.5, se puede observar una fotorresitencia LDR.

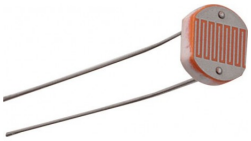

Figura 5.5: Fotorresistencia LDR.

5.1.4.3. Sensor de movimiento: PIR HC-SR501

Los seres humanos desprenden calor en forma de radiación infrarroja, y este calor puede ser detectado por los dispositivos adecuados, como son los sensores de movimiento PIR. Estos sensores [28] de movimiento son capaces de detectar cambios en la radiación infraroja que reciben y disparar un aviso cuando se produce. Para detectar la presentacia de un humano compara el calor emitido por el ser humano y el espacio en el que se encuentra.

Su componente principal es el sensor piroeléctrico. Se trata de un componente electrónico diseñado para detectar cambios en la radiación infrarroja recibida. Normalmente, incorporan un transistor de efecto campo que amplifica la señal eléctrica que general cuando se podruce una variación de radiación.

Un ejemplo de sensor de movimiento PIR es el HC-SR501, que cuenta con 3 pines de conexión (Alimentación de 5V, Salida de 3.3V y GND), y dos resistencias variables de calibración:

- **Ch1:** Esta resistencia sirve para establecer el tiempo que se va a mantener activada la salida del sensor. El tiempo mínimo que se puede establecer se encuentra alrededor de 3 segundos. Existen nuevas versiones con resistencias de 100K que bajan el tiempo mínimo a 0.5 segundos.
- **RL2:** Esta resistencia sirve para establecer la distancia de detección, que puede variar desde 3 metros hasta 7 metros.

En la figura 5.6, se puede observar el funcionamiento de un sensor PIR (izquierda), un sensor PIR (centro), y un sensor HC-SR501 que cuenta con el sensor PIR, un circuito de estabilización y control, y una lente de plástico para mejorar el ángulo de detección (derecha).

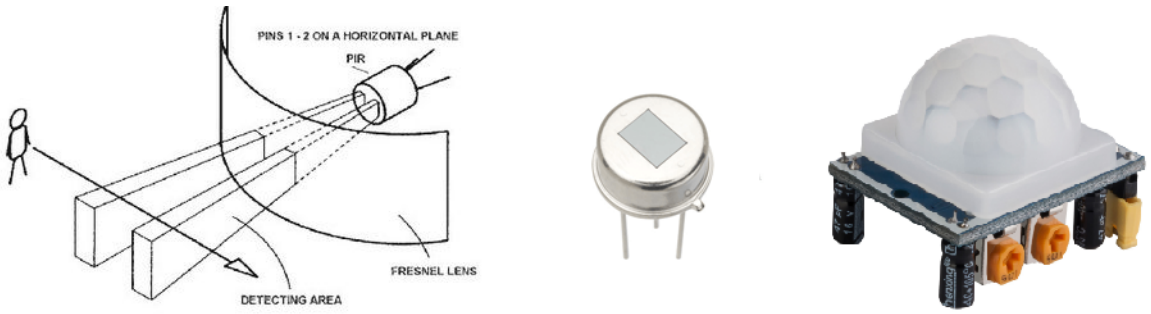

Figura 5.6: Sensor PIR y su funcionamiento.

5.1.4.4. Raspberry Pi Módulo de Cámara v2

El módulo de cámara de Raspberry Pi [29] apareció por primera vez en el año 2013. Esta primera versión permitía utilizar una Raspberry Pi para captar y grabar imágenes con una resolución de 5 megapíxeles. En 2016, apareció una segunda versión mejorando la anterior, con una resolución de 8 megapíxeles. Además, aparecieron dos tipos de módulos, uno que permitía fotografiar y grabar en luz visible, y el otro en infrarrojo. Su precio ronda alrededor de los 25€.

Esta segunda versión cuenta con las siguientes características:

- Permite grabar en 1080p30 (Full HD), 720p60 (HD), entre otros.
- La resolución de la cámara es de 8 megapíxeles.
- Su peso es de 3 gramos con un tamaño aproximado de 25 x 24 x 9 mm.
- Permite guardar en los formatos de imagen JPEG, RAW, GIF, PNG, entre otros.
- Solo permite el formato H.264 de vídeo.

En la figura 5.7, podemos ver un módulo de cámara v2 de Raspberry Pi.

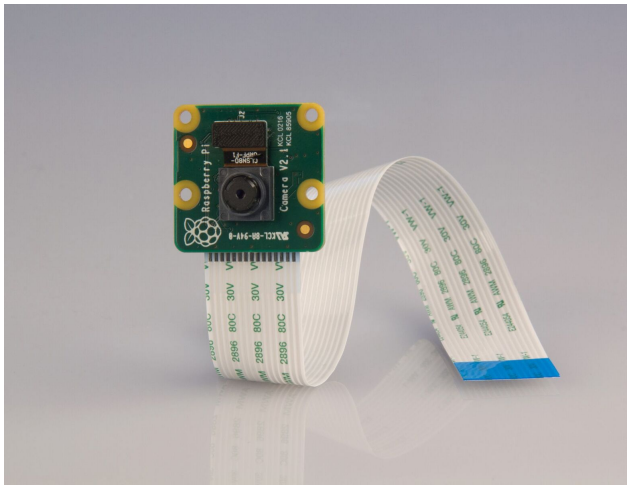

Figura 5.7: Raspberry Pi Camera Module v2.

5.1.5. Conclusiones

Para la realización de este proyecto se necesitará distintos dispositivos IoT, que serán los encargados de recopilar los datos mediante los sensores incorporados, enviarlos mediante algún módulo de comunicación a la plataforma IoT elegida y actuar en función de la información recopilada.

Estos dispositivos IoT serán colocados por distintos lugares del CUM, sensorizando las aulas, laboratorios y espacios comunes.

Existe gran variedad de dispositivos IoT, cualquiera de ellos válido. Este proyecto es independiente de los dispositivos IoT elegidos, por lo que se seleccionarán los dispositivos IoT disponibles en el CUM para su utilización, siendo prácticamente los vistos anteriormente. Específicamente, se utilizarán los siguientes dispositivos IoT:

- NodeMCU v3.
- Arduino UNO.
- Raspberry Pi 2 Model B.

Respecto a las tecnologías de red, se realizó una breve descripción de las más destacables en el capítulo 2 de este Trabajo Fin de Grado. Las diferentes tecnologías de red permiten que los dispositivos IoT se conecten a la red y compartan así los datos obtenidos. De las diferentes tecnologías de red vistas, han sido seleccionadas para su utilización las tecnologías Wifi y Ethernet.

5.2. Plataformas IoT

Para la realización del proyecto se necesitará una herramienta que sea la encargada de la gestión de la aplicación. A continuación, se presentan distintas plataformas, seleccionando la más conveniente para este proyecto.

5.2.1. OpenHAB

OpenHAB [30, 31, 32] es un software de código abierto y fácilmente extensible que permite la integración de diferentes sistemas y tecnologías IoT en una única solución. Permite la creación de reglas automáticas y ofrece interfaces unificadas buscando ser neutral respecto a los diferentes dispositivos y proporcionando un protocolo de comunicación común para estos dispositivos y sus correspondientes tecnologías .

Esta plataforma está escrita en Java y cubre una gran cantidad de sistemas, como por ejemplo Windows, OSX, Linux, en definitiva, todos aquellos capaces de ejecutar Java.

Proporciona diferentes interfaces de usuario basadas en HTML capaces de funcionar en cualquier dispositivo, así como aplicaciones para iOS y Android. Cuenta, por lo tanto, con un diseño responsive adaptando la disposición de los elementos de los paneles de control al tamaño de la pantalla. Además, puede interactuar con diferentes sistemas mediante el servicio REST API, permitiendo el acceso en lectura a los elementos así como actualizaciones de estado o el envío de comandos hacia los dispositivos.

Cuenta con una amplia comunidad para solucionar las diferentes dudas y problemas que tienen los usuarios. El principal problema es que el proyecto es alemán, por lo que gran parte de la comunidad utiliza dicho idioma, dificultando su acceso. Además, el proyecto no cuenta con una gran documentación, dificultando el uso de la aplicación.

OpenHAB tiene dos canales diferentes de comunicación interna: un bus de eventos asíncronos y un repositorio de estados que puede ser consultado.

El bus de eventos es el servicio base de OpenHAB y todos los paquetes que no requieren un comportamiento basado en estados deben usarlo para informar a otros sobre los eventos y para ser actualizados por eventos externos. Hay dos tipos de eventos, uno informa sobre el cambio de estado de algún dispositivo y el otro activa una opción o un cambio de estado en algún dispositivo. Todos los bindings de protocolo, que son los que proporcionan el vínculo a los dispositivos reales, deben estar comunicándose a través del bus de eventos. Estos paquetes permiten la conexión de dispositivos que usen tecnologías diferentes. En la figura 5.8, se muestra como son estos canales de comunicación.

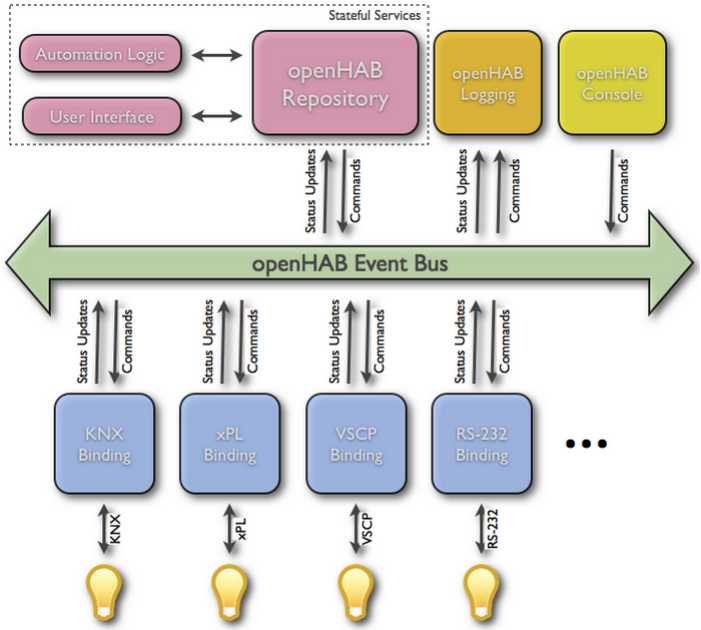

Figura 5.8: Canales de comunicación de OpenHAB.

En definitiva, OpenHAB es una aplicación que sirve para integrar los distintos dispositivos existentes, actuando como mediador entre los diferentes protocolos para lograr una comunicación entre estos dispositivos.

5.2.2. Node-RED

Node-RED es una herramienta de código abierto, disponible en github, que permite interconectar todos los elementos IoT. Estos elementos pueden ser desde dispositivos hardware a APIs o servicios online. Esta programada en Node.js, por lo que a continuación se realizará una pequeña introducción sobre este concepto.

5.2.2.1. Node.js

Node.js [33] es un entorno JavaScript del lado del servidor, basado en eventos. Utiliza el motor V8, desarrollado por Google para uso de su navegador Chrome. Aprovechando el motor V8 permite a Node.js proporcionar un entorno de ejecución del lado del servidor que compila y ejecuta Javascript a altas velocidades. Esto se debe a que V8 compila JavaScript en código máquina nativo, en lugar de interpretarlo.

Fue creado por Ryan Dahl en 2009 con la finalidad de ser útil en la creación de programas de red altamente escalables, como por ejemplo servidores web. Está programado en C++ y JavaScript y puede ser usado en los sistemas operativos Windows, Mac OS X, Linux, Solaris, FreeBSD, OpenBSD y webOS. Cuenta con una licencia de software libre permisiva llamada MIT.

Node.js funciona con un único hilo de ejecución, usando entradas y salidas asíncronas las cuales pueden ejecutarse concurrentemente.

JavaScript tiene la ventaja de poseer un modelo de eventos excelente, ideal para la programación asíncrona, además de ser muy conocido. Esto reduce la curva de aprendizaje de Node.js, ya que no es necesario aprender un nuevo lenguaje de programación.

Node.js puede ser combinado con una base de datos documental, como MongoDB o RethinkDB, y JSON.

En definitiva, Node.js es una plataforma JavaScript del lado del servidor que usa un controlador de eventos y un modelo de entradas y salidas para crear aplicaciones intensivas en datos y en tiempo real que puedan funcionar sin problemas a lo largo de muchos dispositivos en arquitecturas distribuidas.

5.2.2.2. Node-RED

Node-RED [34, 35, 36, 37] es una herramienta visual desarrollada en Node.js y de código abierto creada por IBM. Permite añadir dispositivos a una red y gestionarlos sin necesidad de tener grandes conocimientos de programación. Además, se trata de una herramienta muy ligera, pudiéndose ejecutar en dispositivos tan limitados como una Raspberry.

Node-RED proporciona una interfaz basada en HTML que nos permite crear flujos de eventos e interconectarlos todos ellos a través de un ligero entorno. En dicha interfaz se muestra visualmente las relaciones y las funciones, pudiendo añadir o eliminar nodos y conectarlos entre sí con el fin de hacer que se comuniquen entre ellos.

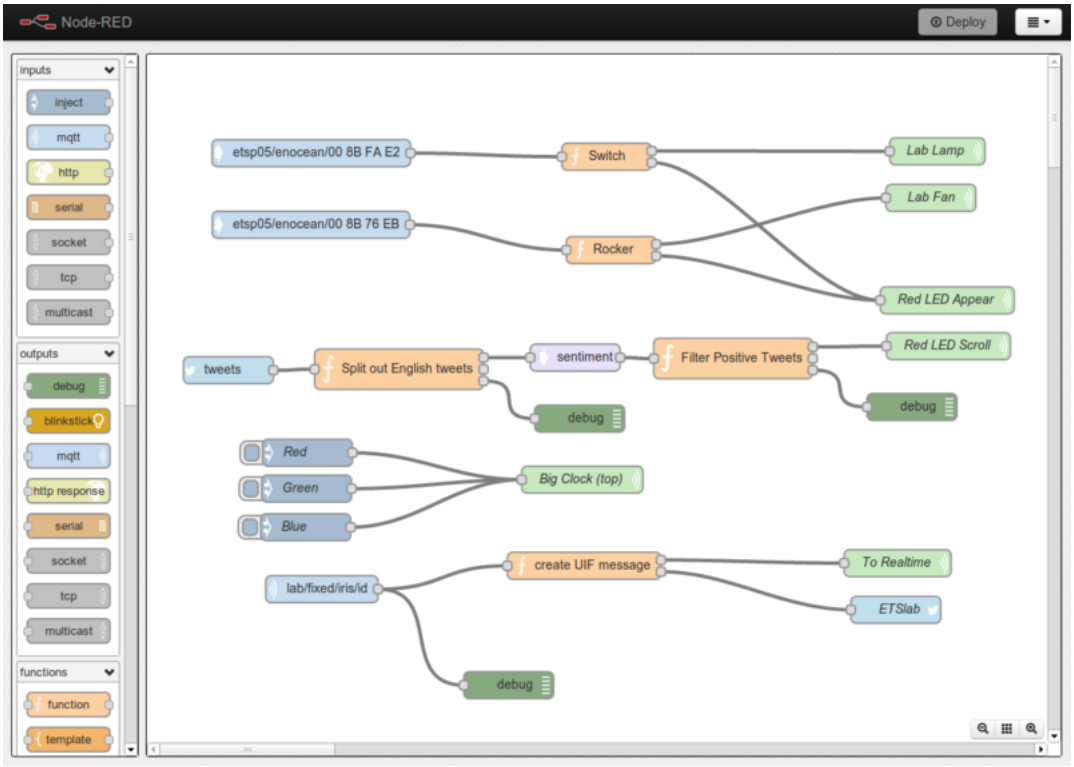

Figura 5.9: Interfaz de Node-RED.

Como se puede observar en la figura 5.9, la interfaz proporcionada es un entorno gráfico sencillo, accesible desde cualquier navegador. En editor se estructura de la siguiente manera:

- **Paleta de nodos:** Muestra todos los nodos disponibles que se pueden utilizar. Además, existe un repositorio de nodos desarrollados por otros usuarios que se pueden instalar fácilmente desde la propia interfaz. Existen dos tipos de nodos, los nodos de inyección y los nodos de función. Los primeros producen un mensaje sin necesidad de alimentarlos con una entrada, lanzando el mensaje al siguiente nodo conectado. Los nodos de función, por el contrario, tienen una entrada y realizan un trabajo en base a ella. Existe una gran cantidad de nodos para elegir como, por ejemplo, nodos para interactuar con la base de datos RethinkDB, o incluso nodos para publicar mensajes en Twitter.

- **Ventana de edición:** Es el espacio habilitado para arrastrar los nodos desde la paleta, conectarlos y configurarlos. De este modo se irá creando el flujo de operación. En Node-RED, los flujos pueden contar con una entrada, un procesamiento y una salida.

Gran parte del éxito que tiene Node-RED se fundamenta en que los nodos y flujos desarrollados por cualquier usuario pueden ser aprovechados por otros. En la página web oficial de Node-RED se encuentra una sección de contribuciones de terceros, con más de mil nodos y flujos subidos por la comunidad y listos para utilizar. Además, cuenta con bastante documentación y una gran comunidad para resolver cualquier problema que exista con la herramienta.

Node-RED cuenta con bastante seguridad, pudiendo implementar el acceso con usuario y contraseña, así como con certificado SSL para acceder al editor por protocolo seguro HTTPS. También dispone de una API REST de administración y operación, de manera que puede interactuar y ser controlado por un sistema externo. Estas características son las que hacen que Node-RED sea adecuado para ejecutarse casi en cualquier plataforma.

5.2.3. Conclusiones

La plataforma IoT será la encargada de la gestión de la información. Su función será recibir los datos de los distintos dispositivos IoT, almacenarlos en la base de datos elegida, compartirlos con la aplicación de visualización para mostrar esta información al usuario y mandar las instrucciones correspondientes a los distintos actuadores según los datos recibidos.

De las distintas plataformas vistas, se ha seleccionado la herramienta Node-RED. El entorno gráfico que proporciona Node-RED es bastante amigable, facilitando la comprensión de los distintos flujos de eventos que se desarrollarán en el proyecto. Además, permite su creación de manera sencilla, únicamente arrastrando los distintos nodos e interconectándolos. La configuración de cualquier nodo no es muy complicada y cuenta con bastante documentación.

Cuenta con API REST, por lo que no existirá ningún problema con los dispositivos IoT elegidos, que podrán conectarse sin problema a la plataforma.

Además, es una plataforma muy ligera y accesible desde cualquier navegador, por lo que la elección del equipo donde vaya a instalarse la plataforma no adquiere importancia.

5.3. Bases de datos

Para la realización de este proyecto se necesita una base de datos distribuida, orientada a documentos y que proporcione librerías JavasScript. Además, debe posibilitar el tratamiento de datos en tiempo real. Cumpliendo estas características se han seleccionado las siguientes bases de datos.

5.3.1. MongoDB

MongoDB [38, 39, 40, 41] es un sistema de base de datos NoSQL orientado a documentos y multiplataforma, lanzado en 2009. De código abierto, cuenta con una licencia GNU AGPL v3.0. MongoDB usa el formato BSON para guardar la información, haciendo que la integración de los datos sea más fácil y rápida. BSON es una versión modificada de JSON, siendo más pesado pero también mucho mas rápido y menos costoso a la hora de acceder a la información.

MongoDB está programado en C++, y está disponible para los sistemas operativos Windows, Linux, OS X y Solaris.

Las consultas en MongoDB se realizan utilizando JavaScript, contando con las diferentes funcionalidades de este lenguaje. Además, existen numerosos drivers oficiales para trabajar en otros lenguajes, como pueden ser C, C++, Java, PHP o Python.

5.3.1.1. Características principales

A continuación se explicarán brevemente las características principales de MongoDB:

- Puede almacenar **todo tipo de datos**, sean estructurados, semiestructurados o no estructurados.
- **Estructura no relacional:** Es una de las características clave de MongoDB. No utiliza tablas como las bases de datos relacionales, sino que utiliza un sistema de esquemas dinámicos. Cada base de datos está formada por colecciones y son el equivalente a las tablas. Los documentos serían equivalentes a las filas y conforman la información que es almacenada en la base de datos.
- **Está orientada a documentos**: Un documento puede almacenar toda la información necesaria sin seguir un esquema predefinido. Como se puede observar en la figura 5.10, una colección contiene distintos documentos, y cada documento puede estar formados por distintos campos sin que haya ningún problema.

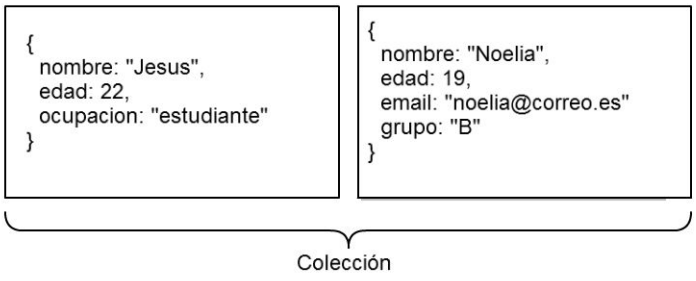

Figura 5.10: Ejemplo de colección en MongoDB.

- **Consultas:** MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares.
- **Replicación:** MongoDB garantiza alta disponibilidad de los datos gracias al conjunto de replicas. MongoDB funciona con un solo servidor principal, considerándose el resto como secundarios. Las aplicaciones clientes se comunican generalmente con el servidor

principal para realizar operaciones de lectura y escritura, y los cambios se distribuirán a los servidores secundarios para mantener la integridad.

- **Balanceo de carga y escalabilidad:** MongoDB puede escalar de forma horizontal usando el concepto de *shard*. Como se puede ver en la figura 5.11,cada *shard* es una instancia independiente de MongoDB que almacena una partición de la base de datos. Los *shards* en conjunto, conforman una única base de datos lógica. Este es el método que utiliza MongoDB para brindar soporte a data sets muy grandes o aplicaciones de gran rendimiento. MongoDB tiene la capacidad de ejecutarse en múltiples servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso de fallo.
- Permite utilizar **Map-Reduce** para el procesado de la información.

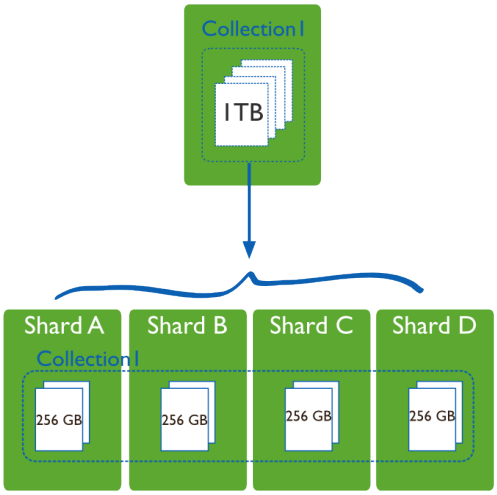

Figura 5.11: Shards en MongoDB.

En resumen, MongoDB ha sido diseñada para que sea rápida, flexible y escalable. Además cuenta con una amplia aceptación, ya que grandes compañías como Telefónica, CISCO, Coca Cola, Leroy Merlin, entre otras, utilizan esta base de datos. Con esta amplia aceptación y la gran cantidad de documentación disponible hacen que sea una de las bases de datos más populares.

5.3.1.2. Principales problemas

A continuación, se enumerarán los principales problemas que presenta la base de datos MongoDB:

- **No implementa las propiedades ACID:** ACID es el acrónimo de Atomicity, Consistency, Isolation y Durability, en español Atomicidad, Consistencia, Aislamiento y Durabilidad. Estas cuatro propiedades garantizan que las transacciones en las bases de datos se realicen de forma confiable.
    - Respecto a la consistencia, es posible que la lectura se realice sobre una versión obsoleta del documento que aún no ha sido actualizado, devolviendo datos incorrectos.

- MongoDB bloquea la base de datos a nivel de documento ante cada operación de escritura, por lo que solo se podrán hacer operaciones de escritura concurrentes entre distintos documentos.
- Las escrituras no son durables ni verificables debido a que MongoDB retorna la respuesta a la escritura cuando aún no ha escrito la información completamente, pudiendo ocasionar pérdidas.
- MongoDB tiene problemas de rendimiento cuando el volumen de datos supera los 100GB.

5.3.2. RethinkDB

RethinkDB [42, 43, 44, 45] es una base de datos Open Source, NoSQL, distribuida y orientada a documentos JSON. Como se puede ver en la figura 5.12, se define como la base de datos de la web en tiempo real, siendo una de sus principales características. Funciona en los sistemas operativos Windows, Linux y OS X y es en parte competencia de MongoDB. Escrita en C++ cuenta con una comunidad de miles de desarrolladores.

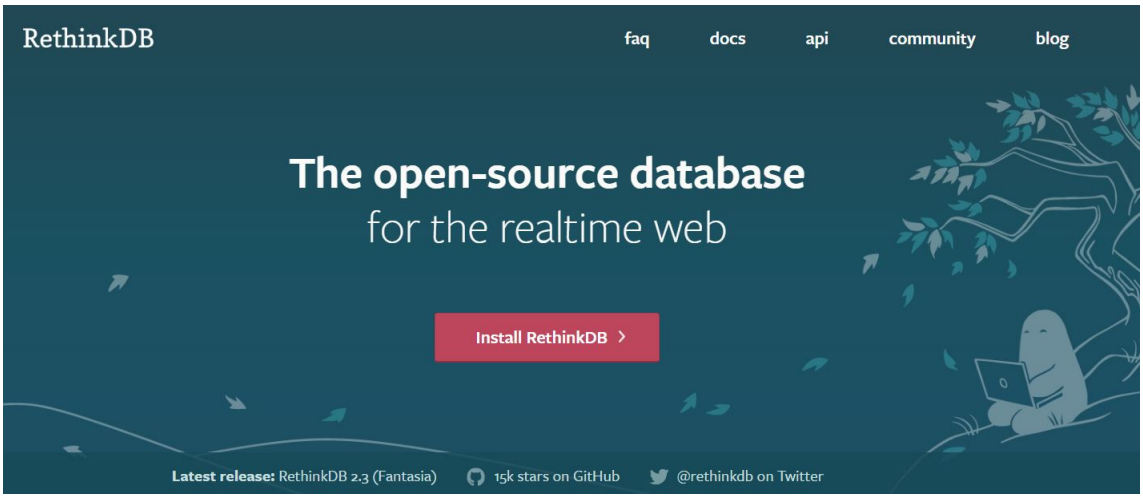

Figura 5.12: Página web de RethinkDB.

Es una base de datos muy joven, liberando la primera versión de su código en 2012 y sacando en 2015 su primera versión preparada para producción.

Sus desarrolladores la recomiendan para proyectos de mercados en tiempo real, juegos multijugador, dispositivos IoT, entre otros.

El soporte para ACID no es completo, pero en realidad sigue otro teorema: CAP. También llamado Teorema de Brewer, CAP enuncia que es imposible garantizar simultáneamente:

- **Consistencia (Consistency):** Que todos los nodos vean la misma información al mismo tiempo.
- **Disponibilidad (Availability):** Garantía de que cada petición a un nodo reciba una confirmación de si ha sido o no resuelta satisfactoriamente.
- **Tolerancia al particionado (Partition Tolerance):** El sistema sigue funcionando incluso si algunos nodos fallan.

Según el teorema, un sistema solo puede tener dos de las tres características simultáneamente. RethinkDB sacrifica la disponibilidad, centrándose en la consistencia y en la tolerancia al particionado.

Tiene drivers oficiales para Ruby, Python, Java y JavaScript/Node.js y más de una docena soportados por la comunidad como PHP.

5.3.2.1. Características principales

RethinkDB cuenta con dos características destacables:

- RethinkDB cuenta con un lenguaje de consulta propio: **ReQL**. Es un lenguaje muy intuitivo comparado con SQL de las bases de datos relacionales y Javascript empleado por MongoDB. Este lenguaje ofrece una forma muy potente para manipular documentos JSON y se puede incrustar en cualquier lenguaje de programación importando los paquetes correspondientes, como se puede observar en el algoritmo 5.1. Por lo tanto, se puede utilizar el entorno de programación habitual.

Algoritmo 5.1 Conexión y creación de una tabla en RethinkDB.

```
import rethinkdb as r  # import the RethinkDB package
conn = r.connect()       # connect to the server on localhost and default port

r.table_create('users').run(conn)   # create a table `users`
r.table('users').run(conn)          # get an iterable cursor to the `users` table
```

- Otra de las características más destacables es ***Changefeeds***, que permite que los clientes reciban los cambios que se producen en una tabla, en un documento o incluso en los resultados de una consulta especifica en tiempo real, sin necesidad de realizar peticiones por parte del cliente. Para ello únicamente hay que indicar a través de un comando que se desea suscribir al seguimiento. En su página web se encuentra un ejemplo de esta funcionalidad, mostrando los resultados de un juego multijugador en tiempo real, como se puede observar en la figura 5.13.

Además, cuenta con otras características:

- Ofrece soporte a Join en las tablas.
- Cuenta con un dashboard web integrado para la administración de las bases de datos y el servidor.
- Ofrece de manera fácil escalamiento horizontal.
- Tiene soporte para Docker[2]

5.3.2.2. RethinkDB vs MongoDB

Los principales puntos de comparación entre RethinkDB y MongoDB se organizan en tres categorías:

[2]Docker es una herramienta para empaquetar aplicaciones y sus dependencias en un contenedor virtual, pudiéndose ejecutar en cualquier servidor Linux.

Figura 5.13: Seguimiento de un juego multijugador utilizando RethinkDB.

	RethinkDB	**MongoDB**
Plataformas	Linux. OS X, Windows	Linux. OS X, Windows, Solaris
Modelo de datos	Documentos JSON	Documentos BSON
Lenguajes de acceso	Protocolo JSON, 3 bibliotecas oficiales y muchas ofrecidas por la comunidad	Protocolo BSON, 13 bibliotecas oficiales y muchas ofrecidas por la comunidad
Tipos de índice	Clave primaria, compuesta, secundaria, geoespacial y arbitraria.	Clave única, compuesta, secundaria, geoespacial y esparcida.

Tabla 5.4: Comparación de desarrollo entre MongoDB y RethinkDB.

- **Desarrollo:** En la tabla 5.4 se puede encontrar la comparación entre MongoDB y RethinkDB.
- **Administración:** La principal diferencia en cuanto a la administración entre las bases de datos RethinkDB y MongoDB es el lenguaje de las consultas. RethinkDB cuenta con su propio lenguaje ReQL, mucho más amigable e intuitivo. Además, puede ser utilizado con cualquier lenguaje y entorno de programación, importando sus librerías. En cambio, MongoDB utiliza JavaScript. Otra diferencia importante es que RethinkDB cuenta con una interfaz para administrar las bases de datos y el servidor. MongoDB ofrece una sencilla interfaz que muestra sólo la lectura de la información del servidor.
- **Arquitectura:** Respecto al almacenamiento, MongoDB utiliza un mapeo de memoria, en el cual el sistema operativo está al cargo de la localización de las entradas y salidas. RethinkDB organiza sus datos en árboles B y los almacena usando un motor de

almacenamiento estructurado.

5.3.3. Conclusiones

Para la realización del proyecto se necesitará una base de datos distribuida, orientada a documentos, que proporcione librerías JavaScript y posibilite el tratamiento de datos en tiempo real. En los apartados anteriores, se destacaron distintas bases de datos que cumplían estos requisitos, siendo la elegida para la realización de este proyecto RethinkDB.

Aunque MongoDB también hubiese sido una buena elección, ya que cumple con todas las características antes mencionadas y tiene bastante más fama y documentación, se ha seleccionado RethinkDB por ser una base de datos más orientada a tiempo real y a aplicaciones IoT, como las utilizadas en este proyecto. Además, hay que destacar que el lenguaje utilizado por esta base de datos, ReQL, es más intuitivo que el utilizado por MongoDB, y que posee la característica *Changedeeds*, que informa la base de datos directamente a los clientes de los cambios que se producen en sus tablas.

RethinkDB funciona en los principales sistemas operativos, por lo que no existirá ningún problema a la hora de instalarlo en cualquier equipo. Además, Node-RED tiene un conjunto de nodos para interactuar con RethinkDB, por lo que desde la aplicación IoT se podrá almacenar los datos recogidos por los diferentes dispositivos IoT en RethinkDB sin ningún problema.

5.4. Visualización

Para completar el proyecto, se integrará en el sistema un módulo de visualización de información en tiempo real, que posibilite a la comunidad universitaria del CUM conocer el estado de las instalaciones del centro. Para esta visualización se desea que la aplicación sea responsive, siendo adaptable a cualquier dispositivo. Además, se deberá visualizar la información obtenida en un tablero de mandos. En este punto se realizará un estudio de las posibles soluciones de visualización de datos procedentes de dispositivos IoT.

5.4.1. Graphene

Graphene [46, 47] es un dashboard desarrollado por jondot de código abierto disponible en github. Esta basado en D3.js, Backbone.js y Graphite. Las dos primeras herramientas son librerías JavaScript para la visualización de datos utilizando estándares web, como HTML, y la tercera es una herramienta de monitorización. En la figura 5.14, se puede observar el dashboard de Graphene.

Graphene proporciona distintas librerías de gráficos básicos, y los usuarios pueden realizar sus propias librerías en función del gráfico que deseen. Para ello Graphene proporciona un kit de herramientas en las que pueden modificar el CSS a los gráficos existentes. Para cualquier modificación mayor de la aplicación es necesario hacerla en el lenguaje de programación Ruby, siendo este uno de sus mayores problemas. Al estar codificada la aplicación en Ruby, es necesario tener un gran conocimiento sobre este lenguaje para poder realizar cambios significativos.

Otro de los grandes problemas es la poca documentación existente proporcionada por el desarrollador, además de no tener una comunidad detrás que respalde la aplicación y resuelvan las dudas y problemas de los usuarios.

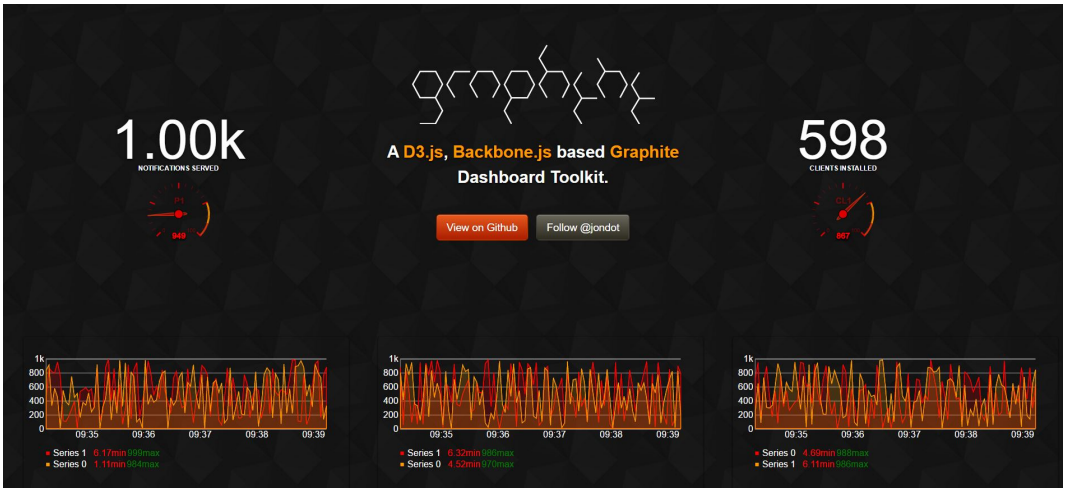

Figura 5.14: Dashboard de Graphene.

5.4.2. Emoncms

Emoncms [48, 49, 50] es un CMS[3] libre desarrollado por la empresa OpenEnergyMonitor. Es una aplicación diseñada para el procesamiento, registro y visualización de datos de carácter ambiental, como pueden ser la temperatura y la energía. Esta aplicación puede ser instalada en un servidor local o conectarse a un servidor externo en la red.

Emoncms hace uso de un identificador conocido como API Key, existiendo dos tipos distintos:

- **Read API Key:** Permite la lectura de datos pero no su modificación.
- **Write API Key:** Se permite tanto la lectura como la escritura de los datos.

Con este identificador se puede hacer uso de la API de emoncms, la cual es necesaria para el envío de los datos capturados por los sensores a la aplicación, encargada de su tratamiento y procesamiento.

Emoncms funciona con un sistema de entradas. Cada unidad que manda información hacia emoncms crea una entrada con una clave y un número de nodo. Con estas entradas se pueden crear diferentes fuentes. Una fuente consiste en exponer la información de una manera distinta como puede ser una gráfica de puntos o de barras. También cuenta con un apartado de dashboard, como se puede ver en la figura 5.15, donde se pueden poner las diferentes fuentes en distintos formatos en forma de widgets.

Emoncms está dividido en módulos con una estructura definida, de forma que agregar nuevas funcionalidades resulte más sencillo. Un módulo es un directorio con todos los archivos que correspondan a una funcionalidad interna. Los módulos principales de emoncms son *user*, *input*, *feed*, *vis* y *dashboard*.

[3]Sistema de gestión de contenidos, p.e. Wordpress.

Figura 5.15: Dashboard de emoncms.

5.4.3. dweet.io y freeboard.io

5.4.3.1. dweet.io

dweet.io [51, 52, 53, 54] es una plataforma de mensajería M2M (Machine-2-Machine) para IoT. Utiliza una API REST para publicar o leer mensajes, llamados dweets, desde dispositivos, máquinas, gadgets o cualquier cosa conectada a Internet.

Para poder usar dweet.io no es necesario registrarse, simplemente hay que utilizar el mismo identificador, que puede ser cualquier cadena de texto, para todos los dweets del mismo dispositivo. Todos los dweets son públicos y pueden ser leídos por cualquier usuario o dispositivo que conozca el identificador. Además, existe el riesgo de que otro dispositivo utilice el mismo identificador, por lo que se mezclarán los datos recopilados por ambos. Para evitar estos problemas es necesario elegir bien el identificador, comprobando que no existe previamente y que será imposible que cualquier otro dispositivo lo use. También existe la opción de reservar un identificador comprándolo, por lo que pasaría a ser privado y no podría ser ni utilizado por otro dispositivo ni consultada su información por usuarios externos.

Para enviar un dweet hay que utilizar el método POST del protocolo HTTPS con la url https://dweet.io:443/dweet/for/{thing}, donde {thing} es el identificador elegido para el dispositivo, y los datos se mandan en el cuerpo en formato JSON.

Para leer el último dweet de un dispositivo se utiliza el método GET del protocolo HTTPS con la url https://dweet.io:443/get/latest/dweet/for/{thing}.

Esta plataforma permite consultar los datos enviados a través de la url https://dweet.io/follow/{thing},en la cual se podrá ver una evolución gráfica de los resultados (figura 5.16), así como los últimos datos recibidos (figura 5.17).

Existen nodos en Node-RED creados por su comunidad, para que los datos recibidos de los distintos dispositivos se puedan mandar directamente a dweet.io, por lo que es una buena opción para el proyecto.

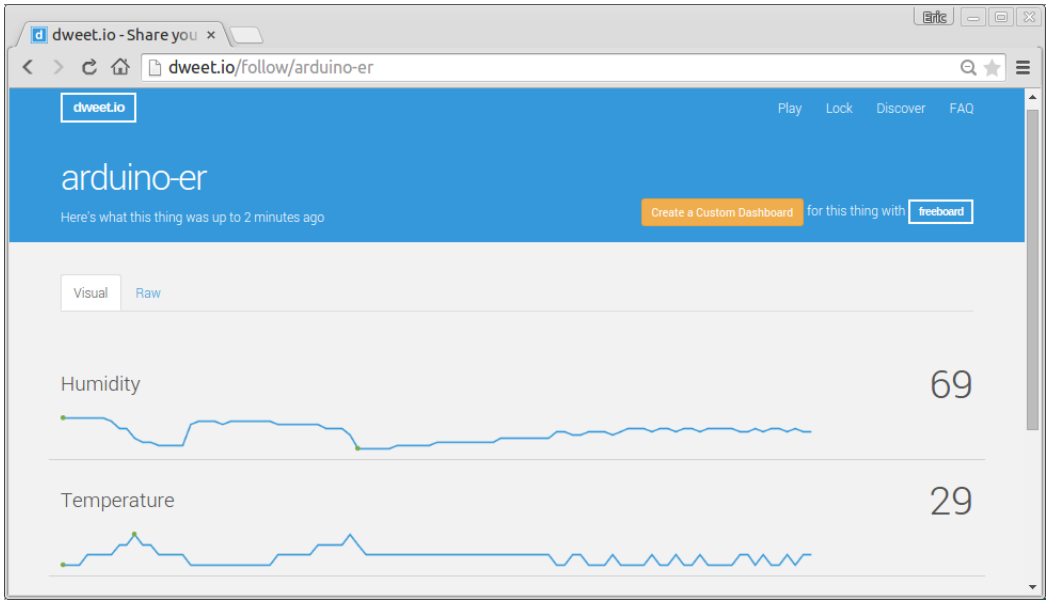

Figura 5.16: Gráficas proporcionadas por dweet.io

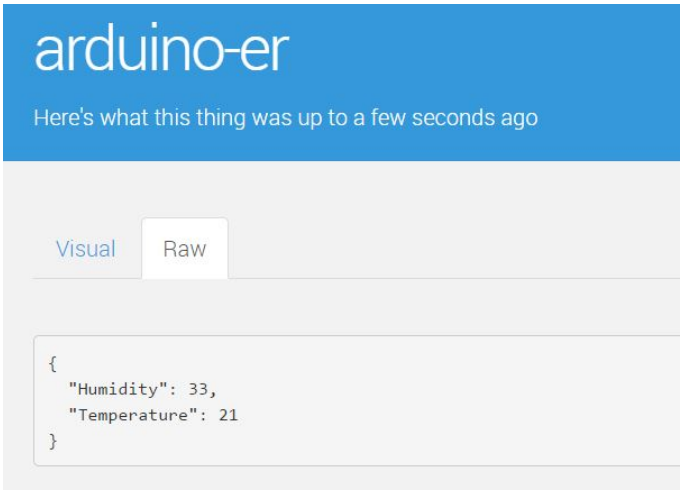

Figura 5.17: Datos proporcionados por dweet.io

5.4.3.2. freeboard.io

Freeboard [55, 56] es un dashboard open-source en tiempo real, desarrollada para IoT, que proporciona integración con dweet.io. Esta aplicación consulta los datos disponibles en dweet.io y los muestra en una interfaz gráfica configurable por el usuario.

Es necesario registrarse en su página web para disfrutar del servicio. Es sencillo de configurar, únicamente hay que señalar la url de dweet.io en la cual se localizan los datos y elegir el tipo de gráfico que los represente. Freeboard no es la plataforma más sofisticada, pero destaca por su facilidad de usar.

Al igual que la aplicación anterior, es gratuita, pero pública, por lo que cualquiera puede acceder al dashboard creado y visualizar los datos. Existen distintos planes de pago, cuya

única diferencia con la versión gratuita es que permite ocultar un número de dashboards, dependiendo del plan elegido.

Las características principales son las siguientes:

- **Fuentes de datos flexibles:** Se integra perfectamente con dweet.io y puede tener acceso a cualquier otra aplicación web.
- **Acceso público o privado:** Según el plan elegido se puede tener dashboards privados a los que ningún usuario ajeno pueda acceder.
- **Desarrollo de widgets:** Se puede seleccionar los widgets incluidos en la aplicación, o añadir uno propio si ninguno se adapta a las necesidades exigidas.
- **Clonación:** Permite clonar dashboards. Si se quiere un dashboard de otro usuario se enviará una petición, y en el caso de que sea aceptada se podrá clonar su dashboard.
- **Simplicidad:** Se puede cambiar rápidamente la apariencia del dashboard, pudiendo arrastrar y soltar los distintos widgets y colocarlos según el usuario.
- **Compartir fácilmente:** Se puede compartir con otras personas fácilmente, pudiendo enviar la url del dashboard a través de correo electrónico, SMS y redes sociales.

Además, Freeboard es responsive, por lo que se puede consultar un dashboard en cualquier dispositivo sin ningún problema. En la figura 5.18, se puede observar el entorno gráfico que proporciona Freeboard.

5.4.4. Conclusiones

Se necesitará una aplicación para la visualización de la información proporcionada por los dispositivos IoT, en tiempo real, que posibilitará a la comunidad universitaria del CUM conocer el estado de las instalaciones del centro. Esta aplicación deberá ser responsive para que pueda ser utilizada en cualquier tipo de dispositivo, incluido un tablero de mandos.

Anteriormente, se realizó un estudio de diferentes aplicaciones que permitían la visualización de datos en tiempo real, cumpliendo las características anteriormente mencionadas. Se ha elegido para el proyecto utilizar la combinación de las aplicaciones dweet.io y freeboard.io, debido a que puede integrarse perfectamente con el resto del sistema diseñado, ya que Node-RED posee un conjunto de nodos para proporcionar los datos obtenidos por los dispositivos IoT a la aplicación dweet.io. Además, se utilizará freeboard.io, que proporciona un dashboard bastante atractivo y que puede integrarse con dweet.io, obteniendo de esta aplicación los datos a mostrar.

5.5. Protocolos de comunicación

Para que los distintos dispositivos IoT puedan enviar y recibir datos es necesario un protocolo de comunicación. Existen una gran cantidad de protocolos especialmente diseñados para estos dispositivos de baja potencia, de los que destacan los siguientes protocolos[59].

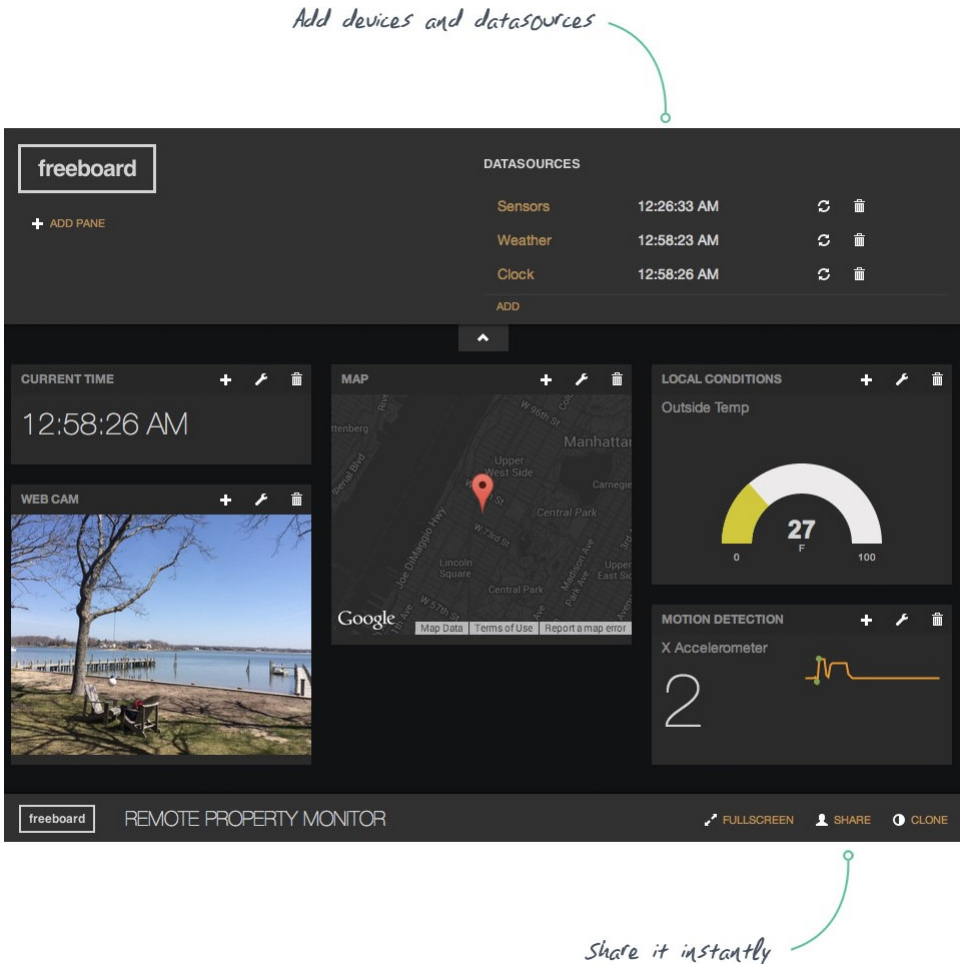

Figura 5.18: Entorno gráfico de Freeboard.

5.5.1. REST

REST [65] es un tipo de arquitectura de interfaces de comunicación basado en el protocolo cliente/servidor HTTP. Es muy útil para la mayoría de servicios, aplicaciones web y dispositivos, siendo ampliamente utilizado y sencillo de implementar, facilitando el desarrollo de servicios en el Internet de las cosas.

Su funcionamiento se basa en mandar peticiones HTTP a determinadas URLs, definiéndose cuatro operaciones[66]:

- **CREATE**: En esta operación, el cliente manda al servidor una petición para crear un nuevo recurso. El servidor le responde con el identificador otorgado al nuevo recurso.
- **DELETE:** En esta operación, el cliente elimina un recurso del servidor, mandando el identificador de dicho recurso.
- **READ:** Con esta operación, el cliente puede leer el estado del recurso, mandando el identificador del recurso y pudiendo seleccionar los tipos de representación soportadas.
- **UPDATE:** Con esta operación, el cliente puede sobrescribir un recurso subiendo una

copia local.

En la figura 5.19, se puede observar el funcionamiento de REST realizando las funciones básicas CRUD (Create, Read, Update y Delete).

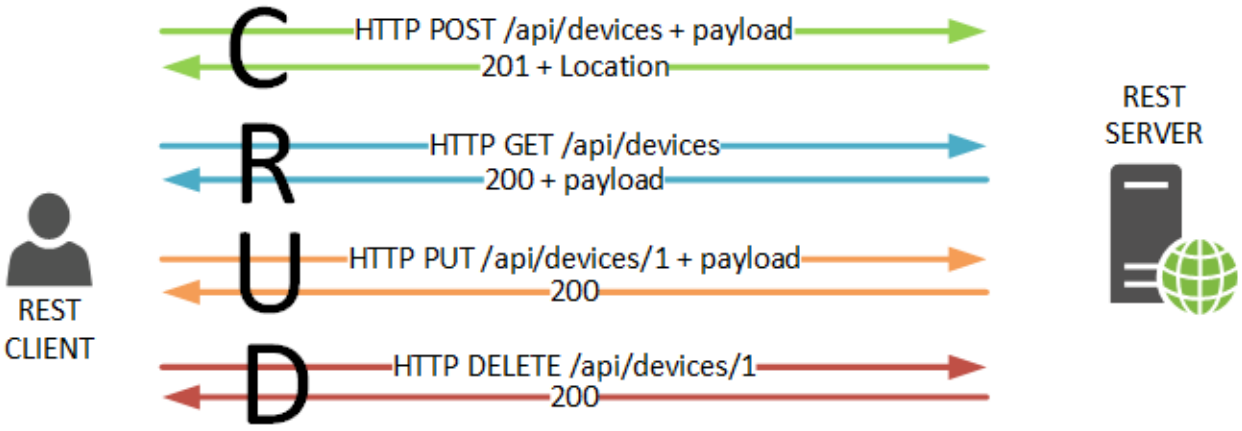

Figura 5.19: Funcionamiento de REST.

REST se puede utilizar para el intercambio de documentos en diferentes formatos, como XML o JSON.

5.5.2. CoAP

CoAP [62, 63] es un protocolo software a nivel de aplicación pensado para ser utilizado en dispositivos electrónicos simples, permitiendo la comunicación entre ellos.

Está basado en el protocolo HTTP, incluyendo otras funciones, como multicast, baja sobrecarga, mensajes asíncronos y mayor simplicidad, siendo muy importantes para el Internet de las cosas.

CoAP implementa REST, con sus operaciones GET, POST, PUT y DELETE, pero utilizando cabeceras reducidas y limita el intercambio de mensajes, usando UDP. Además, permite añadir a la operación GET la función '*observe*', que permite al cliente seguir recibiendo los cambios que se produzcan de un recurso.

CoAP define cuatro tipos de mensajes, con una estructura idéntica entre ellos, formados por una cabecera de tamaño fijo, un número variable de opciones y un payload. Los tipos son:

- **Confirmable (CON):** Los mensajes de este tipo requieren una confirmación de recepción por parte del receptor.
- **Non-Confirmable (NON):** Este tipo de mensaje no requiere confirmación. Suelen encontrarse cuando se puede asumir la pérdida del mensaje, como, por ejemplo, lecturas repetidas de un sensor.

- **Acknowledgement (ACK):** Tipo de mensaje para confirmar la recepción de un mensaje tipo CON o para responder una petición de tipo GET.
- **Reset (RST):** Este mensaje sirve para responder a un mensaje CON o NON recibido en el que el receptor es incapaz de procesar su contenido.

La estructura del paquete CoAP se puede observar en la figura 5.20, estando compuesta por:

- **Ver:** Versión del protocolo CoAP.
- **T:** Tipo de mensaje.
- **TKL:** Longitud del token.
- **Message ID:** Identificador del mensaje.
- **Token:** Es opcional y se utiliza cuando hay una secuencia de paquetes.
- **Options**: En este campo se señalan los tipos de opciones activadas.
- **Payload:** Carga util.

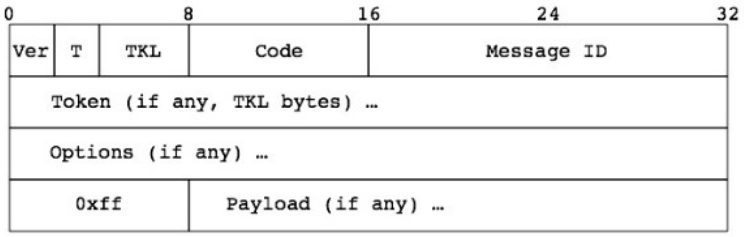

Figura 5.20: Estructura del paquete CoAP.

5.5.3. XMPP

XMPP [64], anteriormente conocido como Jabber, es un protocolo del nivel de aplicación abierto basado en XML, originalmente pensado para mensajería instantánea.

Las características de XMPP son:

- La arquitectura de redes XMPP es descentralizada, por lo que no existe ningún servidor central, sino que cuenta con numerosos servidores conectados entre sí.
- Es una tecnología con bastante presencia, ya que lleva usándose desde 1998. Existen numerosas implementaciones de los estándares XMPP para clientes, servidores y dispositivos, apoyados por compañías como Google o Sun Microsystems.
- Posee robustos sistemas de seguridad y cifrado.
- Sobrecarga demasiado la red con datos de presencia, siendo cerca de un 70 % del tráfico total.

XMPP utiliza HTTP de dos formas distintas:

- **Polling**: Aunque ya no está en uso, utilizaba las funciones GET y POST para enviar mensajes a un servidor a intervalos de tiempo regulares.
- **Binding**: Con este método el cliente utiliza conexiones HTTP de vida más larga para recibir los mensajes de respuesta tan pronto como son enviados las peticiones.

5.5.4. MQTT

El protocolo del nivel de aplicación MQTT [57] es usado para la comunicación machine to machine (M2M) en el Internet de las cosas.

Este protocolo consume muy poco ancho de manda y puede ser utilizado por la mayoría de los dispositivos IoT actuales, ya que requiere pocos recursos.

La arquitectura MQTT se basa en la arquitectura cliente/servidor y posee una topología de estrella, con un nodo central que hace de servidor o broker con una capacidad máxima de hasta 10.000 clientes. Este broker es el encargado de gestionar la red y transmitir los mensajes entre los nodos.

El cliente manda la información al broker, que recibe la información y se la envía a los suscriptores. Este envío de mensajes se basa en el protocolo TCP.

La estructura de un paquete MQTT [58] se puede observar en la figura 5.21, y está compuesta por:

- Una **cabecera fija**, donde se especifica el tipo de paquete MQTT y flags o banderas de control. Existe 15 tipos de paquetes, a destacar el de conectar (valor 1), publicar (valor 3) y suscribir (valor 8).
- Una **cabecera opcional** que utilizan algunos paquetes MQTT, formado por el identificador de paquete que incluye algunos tipos de paquete. Este identificador consta de 2 Bytes (MSB: Byte más significativo; LSB: Byte menos significativo).
- Una carga útil o **payload** que contiene algunos paquetes y varía en función del tipo de paquete.

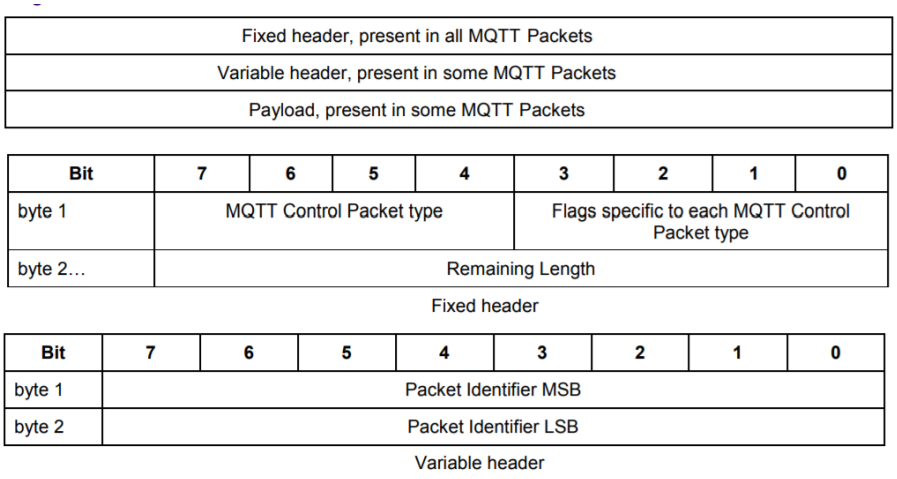

Figura 5.21: Estructura del paquete MQTT.

La comunicación entre los distintos nodos se basa en unos topics, que el cliente crea y los nodos que deseen recibirlo deben suscribirse a él, por lo que la comunicación puede ser de uno a uno, o de uno a muchos. Un topic se representa mediante una cadena y tiene una estructura jerárquica, donde cada jerarquía se separa con '/'. Un ejemplo de topic podría ser "Aulario/Planta1/Aula4B/Arduino2/temperatura". De esta forma un nodo puede subscribirse a un topic concreto o a varios, indicando la parte variable del topic con '#', por ejemplo "Aulario/Planta1/Aula4B/#". En la figura 5.22, se puede observar un ejemplo de jerarquía creada en MQTT.

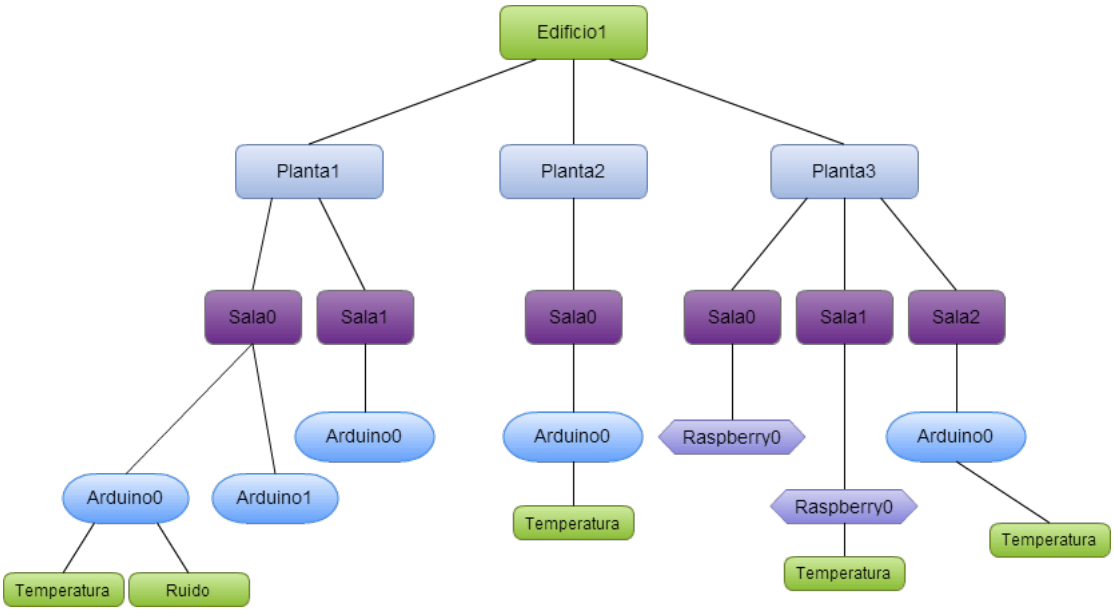

Figura 5.22: Ejemplo de jerarquía en MQTT.

MQTT cuenta con tres modos de funcionamiento o de calidad de servicio (QoS)[60]:

- **QoS0:** Es el modo menos fiable pero también el más rápido. La publicación se envía pero no se recibe confirmación.
- **QoS1:** Se asegura que el mensaje es entregado al menos una vez, pero pueden recibirse duplicados.
- **QoS2:** Es el modo más fiable y que más ancho de banda consume. Se controlan los duplicados para garantizar que el mensaje es entregado una única vez.

Las características de MQTT son las siguientes [61]:

- Está adaptado para utilizar muy poco ancho de banda.
- Es ideal para utilizar redes inalámbricas.
- Consume muy poca energía y requiere pocos recursos.

5.5.5. Conclusiones

Para que los distintos dispositivos IoT puedan compartir los datos obtenidos por los sensores y puedan recibir información es necesario un protocolo de comunicación. El protocolo elegido para este proyecto es MQTT, siendo un protocolo orientado a IoT con bastante aceptación y fama actualmente. Para su implementación se utilizará uno de los brokers más conocidos que existen: Mosquitto.

Mosquitto es un broker open source ampliamente utilizado debido a su ligereza y a la escasa necesidad de recursos que necesita.

Además, para la comunicación entre la plataforma IoT Node-RED y la aplicación dweet.io, es necesario utilizar la API REST, ya que para mandar los datos recibidos a través de los dispositivos IoT a esta aplicación es necesario enviarlos en formato JSON a través de REST.

5.6. Conclusiones de la fase de diseño

En los anteriores puntos se ha comentado las distintas tecnologías y aplicaciones que se utilizarán para el sistema que se realizará en este proyecto. En la figura 5.23, se puede observar como quedará finalmente la estructura del sistema.

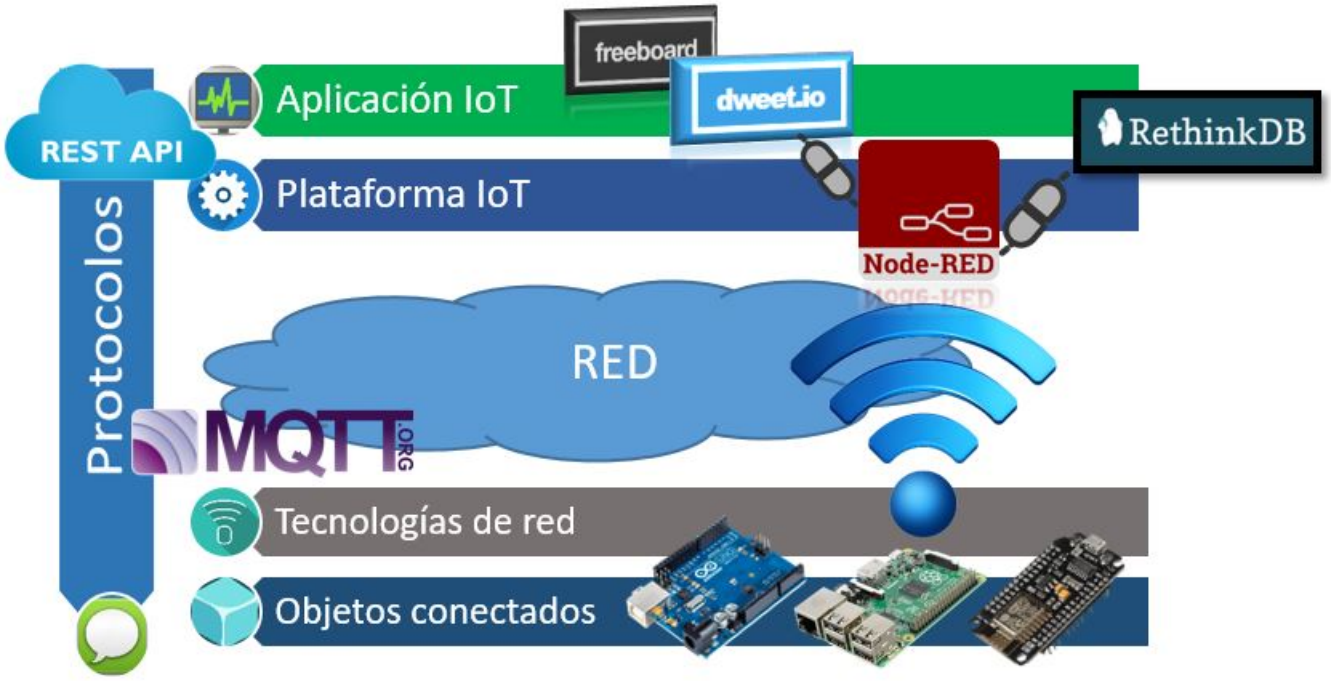

Figura 5.23: Sistema completo a realizar en el proyecto.

La plataforma Node-RED será la parte central del sistema, encargándose de la recepción de los datos proporcionados por los distintos dispositivos IoT para su transmisión a la base de datos RethinkDB y a la aplicación de visualización dweet.io. De esta última aplicación se encargará de recoger los datos el dashboard creado con freeboard.io, que será la aplicación

que interaccione con los usuarios finales del sistema, proporcionando los datos obtenidos de manera gráfica.

Capítulo 6

Implementación

Contenidos

Tras haber realizado el análisis y el diseño del sistema a desarrollar, en este apartado se describirá en detalle su implementación.

En la sección 6.1 se realizará una breve introducción a la implementación del proyecto.

En la sección 6.2 se comentará la estructura y configuración de la red utilizada en el proyecto.

En la sección 6.3 se desarrollará la implementación de los dispositivos IoT del sistema, viendo tanto el montaje como la programación necesaria para su funcionamiento.

En la sección 6.4 se explicará la implementación realizada de la plataforma IoT Node-RED, comentando los distintos nodos creados.

En la sección 6.5 se mostrará la implementación de la base de datos RethinkDB realizada en este proyecto.

En la sección 6.6 se comentará la implementación realizada de las aplicaciones de visualización para que los usuarios puedan consultar la información generada.

6.1. Introducción

La implementación de este sistema se iba a realizar en las instalaciones del Centro Universitario de Mérida. En concreto, se asignó para su implementación el aula 3B, también conocida como el laboratorio *FabLab*, que se encuentra en el edificio Antonio Castillo Martínez, en la planta baja. Este aula aún no estaba adaptada a las necesidades del proyecto, por lo que fue necesario contactar con BATUSI para que habilitaran la red y con un electricista para poder

implantar los dispositivos IoT por el aula. Tras un tiempo, se ha visto imposible la realización en una instalación real, por lo que se ha diseñado el aula en una maqueta realizando un prototipo del sistema.

La maqueta realizada se puede observar en la figura 6.1. Se compone de dos partes, el aula (figura 6.2) y la parte donde se encuentran los dispositivos (figura 6.3).

Figura 6.1: Maqueta realizada para este Trabajo Fin de Grado.

Figura 6.2: Parte de la maqueta: Aula.

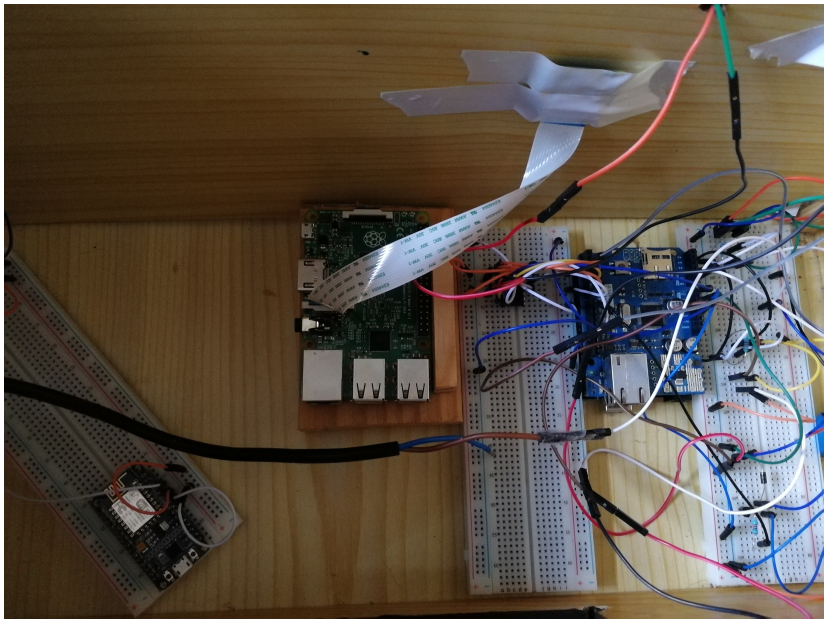

Figura 6.3: Parte de la maqueta: Dispositivos.

6.2. Estructura y configuración de la red

El sistema que se ha desarrollado en este Trabajo Fin de Grado debe contar con una red que permita la conexión de los distintos dispositivos y de la plataforma IoT para el envío y recepción de datos. Además, esta red debe tener una salida a Internet para la plataforma IoT pueda enviar los datos a las aplicaciones de visualización. También es necesario tener en cuenta que algunos de los dispositivos se conectan de manera inalámbrica mediante WiFi y otros están directamente conectados mediante Ethernet.

Debido a los requisitos comentados ha sido necesario el uso de un router tanto para la creación de la red LAN que conectan todos los dispositivos con la plataforma y la base de datos, como para la conexión a Internet. En este proyecto, el número de dispositivos es aún bajo, por lo que al aumentar el número de dispositivos se hará necesario el uso de switches. Prácticamente cualquier router actual que se puede encontrar en las redes domésticas es válido, ya que proporciona salida a Internet, cuentan con puertos Ethernet y permiten la conexión WiFi.

Cada dispositivo IoT conectado a la red debe tener una dirección IP única, que actúa como identificador del dispositivo dentro de la red. Estas direcciones constan de 32 bits, de los cuales una parte se destina a identificar la red y la otra parte se destina a identificar el dispositivo, siendo la máscara de subred la encargada de indicar esta partición.

Se ha decidido utilizar una subred privada de tipo C con 24 bits de red y 8 bits para los dispositivos. Por lo tanto, la dirección de red utilizada es la 192.168.1.0 con máscara de subred 255.255.255.0 (/24). Con este tipo de red se podrá tener un total de 256 direcciones disponibles (2^8), teniendo en cuenta que la dirección de red es la 192.168.1.0 y que la dirección de broadcast es la 192.168.1.255, quedan disponibles 254 direcciones para dispositivos.

Estas direcciones se irán asignando a los dispositivos de manera automática gracias al protocolo de red DHCP. Este protocolo, presente en todos los routers domésticos actuales,

proporciona automáticamente a los dispositivos una dirección IP y el resto de la configuración necesaria para su conexión.

6.3. Dispositivos IoT

En esta sección se describe el proceso de montaje de los dispositivos IoT utilizados en el proyecto, así como la programación que ha sido necesaria para su funcionamiento.

6.3.1. Montaje de los dispositivos

Para la realización de este proyecto se han usado distintos dispositivos IoT, en concreto:

- Un arduino UNO, que cuenta con dos sensores de luz LDR, un sensor de movimiento PIR HC-SR501, un motor para actuar en la persiana del aula, y sistema formado por un relé y un transistor BD137 para el control de la iluminación. Es importante destacar que se le ha dado un plazo de entre 30 segundos y 1 minuto al sensor de movimiento para que cuando no detecte una persona no apague inmediatamente la luz por si vuelve a entrar o se trata de un dato erróneo. Además, a este dispositivo IoT se le ha añadido un Shield Ethernet que permite la conexión del dispositivo en la red y, por tanto, la comunicación con el resto del sistema. El esquema de montaje se puede observar en la figura 6.4.

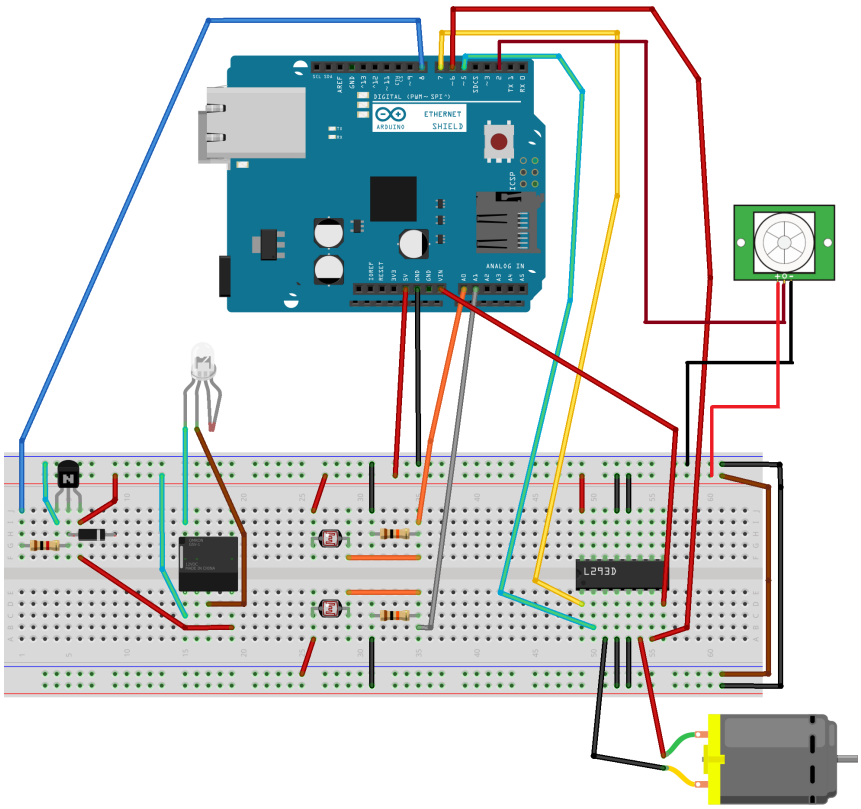

Figura 6.4: Esquema del montaje del dispositivo Arduino UNO en el proyecto.

- Un NodeMCU con un sensor de temperatura y humedad DHT-11. En la figura 6.5, se puede observar el esquema de montaje del dispositivo en este proyecto.

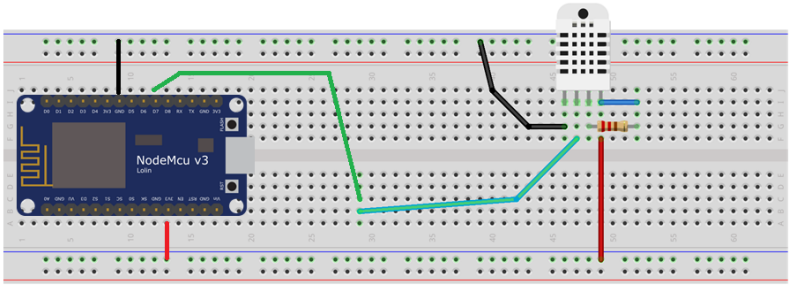

Figura 6.5: Esquema del montaje del dispositivo NodeMCU en el proyecto.

- Una Raspberry Pi 2 modelo B con un módulo de cámara. Este dispositivo proporciona las imágenes del aula y se encarga de albergar parte del sistema, en concreto el servidor MQTT y la plataforma IoT Node-RED.

6.3.2. Programación de los dispositivos

Los dispositivos con sensores y actuadores son el Arduino UNO y el NodeMCU, ambos programables en el lenguaje de programación Arduino, basado en C++. No existe una gran diferencia entre el código de ambos dispositivos. Las diferencias principales serán a la hora de establecer la conexión con la red, ya que Arduino será mediante Ethernet y NodeMCU a través de WiFi.

6.3.2.1. Establecer la conexión

Algoritmo 6.1 Establecer conexión Ethernet.

```
#include <Ethernet.h>

EthernetClient ethClient;
PubSubClient client(ethClient);

// Direccion MAC
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };


void setup()
{
  Serial.begin(9600);
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
  }
  ...
}
```

Para realizar la conexión a través de Ethernet es necesario incluir el código del algoritmo 6.1. En primer lugar, se importa la librería que permite la conexión Ethernet. Tras esto,

definimos una dirección MAC para el dispositivo y creamos una instancia de cliente Ethernet. Por último, en la función *setup()*, se intentará conseguir una dirección mediante DHCP, devolviendo un 1 si lo ha conseguido, y un 0 en caso contrario.

Algoritmo 6.2 Establecer conexión WiFi.

```
const char* ssid = "Orange-AE8D";
const char* password = "E24E6F32";

void setup_wifi() {
   delay(100);
    Serial.print("Conectando a ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
      delay(500);
      Serial.print(".");
    }
  randomSeed(micros());
  Serial.println("");
  Serial.println("Conectado");
  Serial.println("IP: ");
  Serial.println(WiFi.localIP());
}
```

En cambio, para realizar la conexión mediante WiFi, es necesario incluir el código del algoritmo 6.2. Se debe definir el nombre de la red y su contraseña. Tras esto, se debe crear un método encargado de establecer la conexión que será llamado en la función *setup()*.

6.3.2.2. Envío de datos

Algoritmo 6.3 Conexión y publicación MQTT en Arduino.

```
// IP del servidor MQTT
IPAddress mqtt_server(192, 168, 1, 100);

void loop()
{
 ...
 long now = millis();
  if (now - lastMsg > 1000) {
    lastMsg = now;
    String json = buildJson();
    char jsonStr[200];
    json.toCharArray(jsonStr,200);
    boolean pubresult = client.publish("aulaFABLAB/arduino",jsonStr);
 }
 client.loop();
}
```

Para el envío de datos se debe definir la IP del servidor MQTT. Una vez definida, se podrá publicar mensajes a través de MQTT mediante el uso de topics. En el algoritmo 6.3, se puede

observar el ejemplo utilizado por el Arduino en este proyecto, donde se puede ver que el topic utilizado es *'aulaFABLAB/arduino'*.

Algoritmo 6.4 Preparación de los datos a enviar.

```
String buildJson() {
  String data = "{";
  data+="\n";
 // data+= "\"d\": {";
  //data+="\n";
  data+="\"myName\" : \"Arduino\",";
  data+="\n";
  data+="\"presen\" : ";
  data+=(int)digitalRead(PIRPin);
  data+= ",";
  data+="\n";
  data+="\"lumDentro\" : ";
  data+=(int)analogRead(A0)/10;
  data+= ",";
  data+="\n";
  data+="\"lumFuera\" : ";
  data+=(int)analogRead(A1)/10;
  data+= ",";
  data+="\n";
  data+="\"luz\" : ";
  data+=(int)luz;
  data+= ",";
  data+="\n";
  data+="\"persiana\" : ";
  data+=(int)persiana;
  data+="\n";
  data+="}";
  return data;
}
```

En el algoritmo 6.4, se puede observar como se preparan los datos para su publicación en formato JSON. Los datos, presencia, luminosidad dentro del aula y luminosidad fuera del aula, en este caso, se leen a través de las entradas del Arduino.

6.3.2.3. Recepción de datos

Los dispositivos IoT con actuadores se han suscrito a topics en MQTT, y a través de ellos reciben la información enviada desde la plataforma IoT referente a los actuadores. En el algoritmo 6.5, se puede ver el código desarrollado en el proyecto. Se ha indicado el topic al cual debe suscribirse para la recepción de los datos enviados desde la plataforma IoT. Según el dato enviado puede ocurrir:

- Si la plataforma IoT indica un '1', significa que la persiana debe subirse.
- Si se recibe un '2', significa que la persiana debe bajarse y se debe encender la luz del aula.
- Si se recibe un '3', únicamente se encenderá la luz del aula.

- Si se recibe otro dato, se deberá apagar la luz del aula.

Algoritmo 6.5 Suscripción a un topic MQTT y recepción de datos.

```
const char* topicName = "actuarAulaFABLAB";

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  int i=0;
  for (i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
   if((char)payload[0]=='1'){
    persiana=1;
    digitalWrite(E1, HIGH);      // Activamos Motor1
    digitalWrite(I1, HIGH);      // Arrancamos
    digitalWrite(I2, LOW);
    delay(200);
    digitalWrite(E1, LOW);       // Paramos Motor 1
  }else if((char)payload[0]=='2'){
    persiana=0;
    luz=1;
    digitalWrite(ledPIN , HIGH);
    digitalWrite(E1, HIGH);      // Activamos Motor1
    digitalWrite(I1, LOW);       // Arrancamos con cambio de direccion
    digitalWrite(I2, HIGH);
    delay(200);
    digitalWrite(E1, LOW);       // Paramos Motor 1
  }else if((char)payload[0]=='3'){
    digitalWrite(ledPIN , HIGH);
    luz=1;
  }else{
    digitalWrite(ledPIN , LOW);
    luz=0;
  }
}
```

6.4. Plataforma IoT

En este proyecto, se ha utilizado la plataforma Node-RED para la gestión y tratamiento de los datos obtenidos por los dispositivos IoT. Esta plataforma es el cerebro y la parte más importante del sistema desarrollado, ya que se encarga de la recepción y tratamiento de los datos, cuenta con una serie de reglas que hacen al sistema autónomo, almacena la información en la base de datos y se la proporciona a la aplicación de visualización.

Tras su instalación siguiendo los pasos que se encuentran en el *Anexo I* de este documento, se procedió a la creación de los distintos nodos por los que estaría formada la plataforma. La estructura creada para el proyecto se puede observar en la figura 6.6, y está compuesta por los siguientes nodos:

1. **Suscripción a un topic de MQTT:** Para poder suscribirse a un topic, es necesario

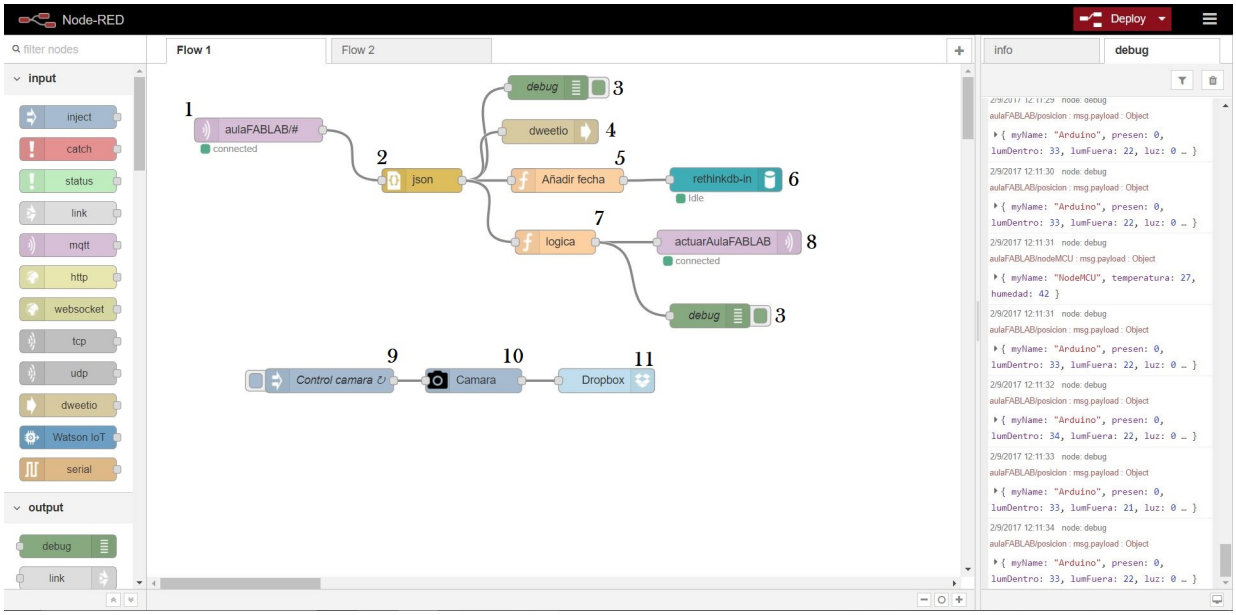

Figura 6.6: Estructura creada en Node-RED para este proyecto.

configurar el nodo de suscripción MQTT existente en Node-RED. Es necesario indicar la IP del servidor MQTT, el puerto que utiliza (por defecto 1883), el topic al que se desea suscribir y el tipo de calidad de servicio que se desea. El topic al que está suscrito Node-RED es *'aulaFABLAB/#'*, por lo que todos los topics que empiecen por la palabra *aulaFABLAB* serán recibidos por la plataforma. El símbolo '#' indica que la palabra puede tomar cualquier valor, en nuestro caso, el nombre del dispositivo IoT que manda los datos. La calidad de servicio seleccionada para este proyecto es la número 2, que controla los duplicados para garantizar que el mensaje es entregado una sola vez.

2. **Conversor a JSON:** Los datos se encuentran en el payload o carga útil de los mensajes MQTT. Esta función convierte esa carga útil en un documento JSON, recuperando el formato con el que es enviada la información desde los dispositivos IoT.

3. **Debug:** Estos nodos permiten mostrar los datos recibidos en la ventana que se encuentra en la parte derecha de la plataforma, permitiendo al administrador hacer un seguimiento más específico de los datos.

4. **Nodo dweetio:** Este nodo es el encargado de transmitir la información recibida a la aplicación de visualización. La única configuración que necesita es indicar el identificador con el que se desea publicar los datos, siendo *'cumUnexIoTjesalguerTFG'* el elegido para este proyecto.

5. **Añadir fecha:** Esta función se encarga de añadir la fecha a los datos recibidos y que están a punto de ser almacenados en la base de datos.

6. **Nodo rethinkdb-in:** Este nodo permite ejecutar sentencias en la base de datos RethinkDB. En este proyecto ejecuta la sentencia que se encuentra en el algoritmo 6.6,

almacenando los datos en formado JSON en la tabla *'aulaFABLAB'* de la base de datos *'CUM'*. Para poder ejecutar esa sentencia es necesario establecer la conexión con la base de datos, configurando el nodo. La información necesaria es la IP del servidor donde se encuentre RethinkDB, el puerto que utiliza (por defecto el 28015), y el nombre de la base de datos.

7. **Nodo lógica:** En este nodo se encuentra el código necesario para que el sistema sea autónomo. Este código analiza los datos obtenidos para ver si es necesario realizar alguna acción en el aula. El código se puede observar en el algoritmo 6.7, y realiza las siguientes comparaciones:

   a) Si el sensor de movimiento detecta la presencia de una persona en el aula, existe suficiente luminosidad fuera del aula y la persiana está bajada, mandará un '1' a través de MQTT para que el dispositivo encargado suba la persiana.

   b) Si el sensor de movimiento detecta la presencia de una persona en el aula, no existe suficiente luminosidad fuera y está la luz apagada, mandará un '2' si la persiana está subida para que esta se baje y se encienda la luz, o mandará un '3' si la persiana ya está bajada para que se encienda directamente la luz del aula.

   c) Si el sensor de movimiento ya no detecta una persona en el aula y la luz está encendida, mandará un '4' para apagar la luz del aula.

8. **Publicar mensajes MQTT:** Este nodo permite enviar mensajes a través del topic *'actuarAulaFABLAB'*, que recibirán los dispositivos IoT y en función del mensaje enviado realizarán una determinada acción. Para el funcionamiento correcto de este nodo, se ha tenido que indicar el topic con el que se publicarán los mensajes, la IP del servidor MQTT y el puerto a utilizar (por defecto el 1883).

9. **Control cámara:** Este nodo se encarga de realizar una petición cada 20 segundos a la Raspberry Pi, para que realice una fotografía del aula.

10. **Cámara:** Para poder sacar la fotografía es necesario configurar este nodo correctamente. En él hay que indicar el formato de la fotografía, el nombre que desea recibir, la resolución, así como diferentes aspectos de la fotografía, como es el brillo y el contraste. Para este proyecto se ha pedido que la fotografía se llame *'capturaNode'* en formato PNG con una resolución de 1024x768.

11. **Dropbox:** Esta fotografía se subirá a Dropbox para mostrarla en la aplicación de visualización. Para ello es necesario configurar la cuenta de Dropbox en la que se subirán las fotografías, así como la ruta donde se quiere subir.

Algoritmo 6.6 Almacenar los datos obtenidos en la base de datos RethinkDB.

```
r.table('aulaFABLAB').insert(msg.payload);
```

6.5. Base de datos: RethinkDB

En este proyecto, se ha utilizado la base de datos RethinkDB para almacenar los datos obtenidos por los dispositivos IoT. Esta base de datos ha sido instalada en un dispositivo

Algoritmo 6.7 Código para la automatización del sistema.

```
if(msg.payload.presen==1){
    if(msg.payload.persiana===0 && msg.payload.lumFuera>40){
        var newMsg={ payload: 1 };
    }
    else if(msg.payload.luz===0 && msg.payload.lumFuera<40){
        if(msg.payload.persiana==1){
            var newMsg={ payload: 2 };
        }
        if(msg.payload.persiana===0 && msg.payload.luz===0){
         var newMsg={ payload: 3 };
        }
    }
}
else if(msg.payload.presen===0 && msg.payload.luz==1){
    var newMsg={ payload: 4 };
}
return newMsg;
```

diferente a la Raspberry Pi, ya que no soportaba su instalación, siguiendo los pasos que se encuentran en el *Anexo I* de este documento.

El primer paso para el correcto funcionamiento del sistema fue establecer el equipo donde se localizaba la base de datos como un nodo de esta, permitiendo el acceso desde otros dispositivos a través de su IP. La manera de permitirlo fue mediante el comando *'rethinkdb --join IPdelEquipo:29015 --bind all'*. Una vez establecido, la plataforma IoT se podía conectar a la base de datos y almacenar en ella la información obtenida.

Se creó una base de datos con el nombre *'CUM'*, formado por una serie de tablas donde cada una de ellas almacena los datos obtenidos en cada una de las instalaciones del Centro Universitario de Mérida. La tabla que ha sido utilizada para este proyecto se denomina *'aulaFABLAB'*.

La interfaz que proporciona RethinkDB para los usuarios muestra el número de documentos que posee actualmente la tabla y una gráfica de las inserciones que se realizan en tiempo real. También permite la consulta de los datos almacenados, a través del comando *'r.table("nombreTabla")'*, que proporciona un listado de todos los documentos almacenados como se puede observar en la figura 6.7. Estos documentos se encuentran en formato JSON y a cada uno de ellos se les ha añadido un campo *ID* para identificar el documento.

6.6. Aplicación de visualización: dweet.io y freeboard.io

En este proyecto, se ha decidido usar para la visualización de la información las aplicaciones dweet.io y freeboard.io.

La plataforma IoT Node-RED publica los datos obtenidos por los dispositivos en la aplicación web dweet.io junto con un identificador que permite su consulta. El identificador usado en este proyecto es *'cumUnexIoTjesalguerTFG'*, por lo que si se accede al enlace *https://dweet.io/follow/cumUnexIoTjesalguerTFG* se podrá observar los datos enviados por Node-RED como se puede observar en la figura 6.8.

Esta aplicación ya permite la visualización de los datos, pero es bastante sencilla y no se puede configurar otros tipos de gráficos ni usar otro tipo de información diferente a la

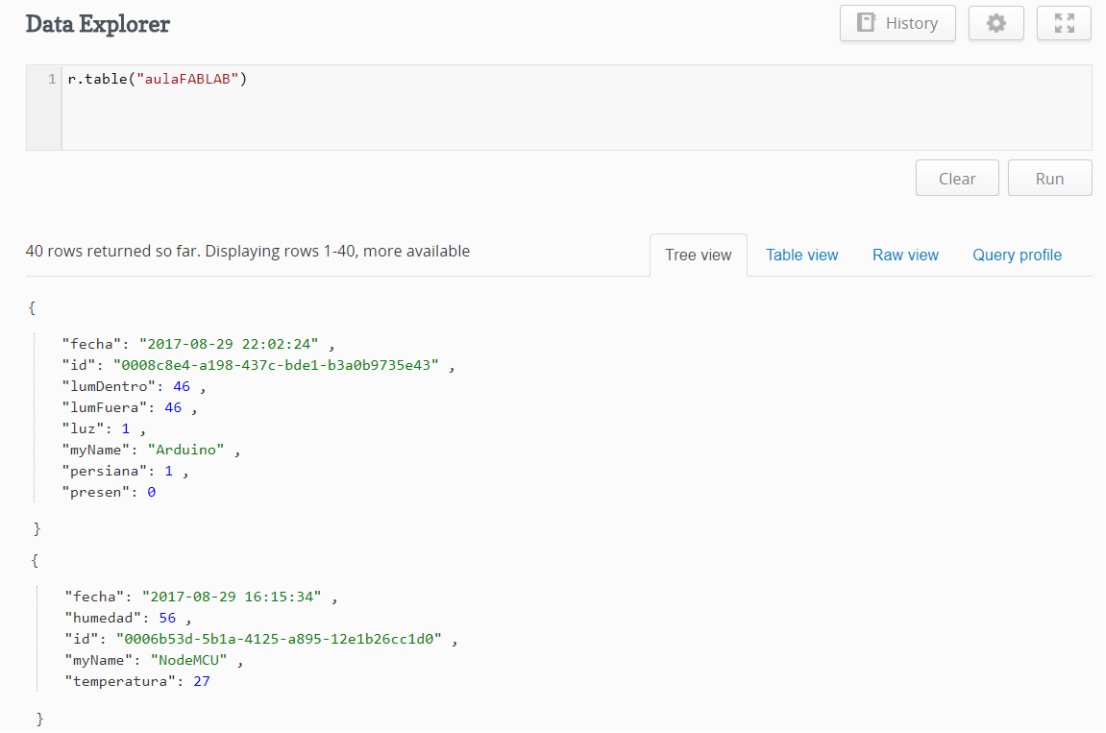

Figura 6.7: Consulta de los documentos almacenados en RethinkDB.

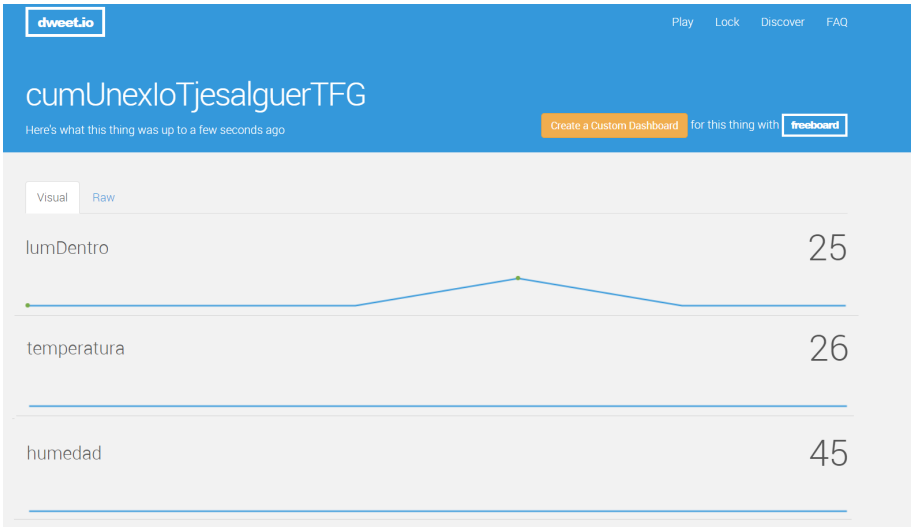

Figura 6.8: Datos publicados en la aplicación dweet.io.

proporcionada por Node-RED. Por eso, se decidió usar esta aplicación como intermediario, facilitando los datos a la aplicación de visualización freeboard.io, que permite la creación y gestión de dashboards.

El primer paso tras registrarse en la aplicación fue la creación de un dashboard llamado *'Aula FABLAB'*. Tras su creación fue necesario proporcionarle una fuente de datos vinculando la aplicación dweet.io. Para ello, se pulso en *'ADD'* en el apartado *'Datasource'*. Tras pulsar,

apareció una nueva ventana pidiendo la configuración de la fuente de datos, solicitando la siguiente información:

- **Tipo:** Indica cual es la fuente de datos, en nuestro caso, la aplicación dweet.io.
- **Nombre:** El nombre que se le desea dar a la fuente de datos. En este proyecto, el nombre recibido es *'Aula FABLAB'*.
- **Identificador:** El identificador usado en la aplicación dweet.io.
- **Contraseña:** La contraseña de la fuente de datos si tuviese.
- **Información adicional:** Se puede permitir obtener más información de los datos obtenidos en la aplicación dweet.io, como puede ser la fecha de publicación del dato. En este proyecto se ha permitido la obtención de información adicional.

Una vez vinculado, ya se puede obtener los datos existentes en dweet.io, y por lo tanto gestionar el dashboard. Un dashboard está compuesto por una serie de paneles, que a su vez están formados por distintos widgets. En la figura 6.9, se puede observar el dashboard creado para este proyecto, que está compuesto por:

Figura 6.9: Dashboard creado para este proyecto.

- Un panel superior que indica el nombre del centro y del aula utilizada para el proyecto.
- Un panel izquierdo, formado por los siguientes widgets:
  - Un gráfico de líneas con la temperatura del aula.
  - Un indicador del nivel de temperatura en el aula.
  - Un gráfico de líneas con la humedad del aula.
  - Un indicador del porcentaje de humedad del aula.

- Un panel central, formado por:
  - Un widget que permite visualizar la imagen del aula.
  - Un widget que muestra en un mapa la localización del centro.
- Un panel derecho, formado por los siguientes widgets:
  - Un gráfico de líneas con el porcentaje de luminosidad dentro del aula.
  - Un indicador del porcentaje de luminosidad dentro del aula.
  - Un gráfico de líneas con el porcentaje de luminosidad fuera del aula.
  - Un indicador del porcentaje de luminosidad fuera del aula.

Freeboard.io es una aplicación web responsive, por lo que se adapta a cualquier tipo de dispositivo. En la figura 6.10, se puede observar el dashboard del proyecto visto desde un smartphone.

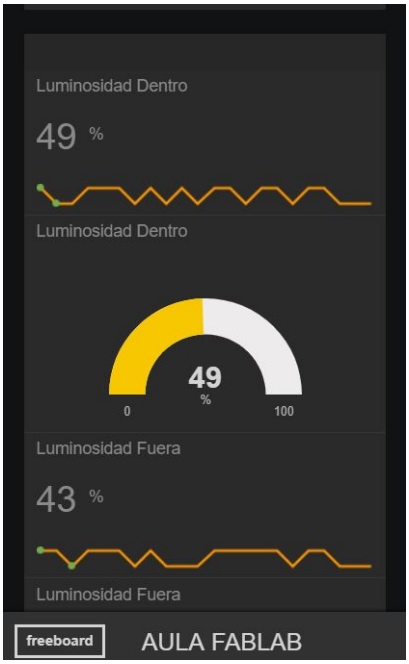

Figura 6.10: Dashboard del proyecto visto desde un smartphone.

Capítulo 7

Conclusiones y Trabajos Futuros

Contenidos

En este capítulo se resumirán las principales conclusiones obtenidas tras la realización de este Trabajo Fin de Grado, así como los trabajos futuros que pueden plantearse y la planificación real del proyecto.

En la sección 7.1 se encuentra las principales conclusiones del proyecto.

En la sección 7.2 se comenta algunos trabajos futuros que pueden plantearse teniendo como punto de partida este proyecto.

En la sección 7.3 se muestra la planificación real del proyecto argumentando las variaciones sufridas respecto a la planificación estimada.

7.1. Conclusiones

En primer lugar, se repasará los objetivos expuestos al comienzo de este proyecto, para verificar que se han cumplido correctamente. Tras esto, se comentarán las distintas conclusiones personales que se han obtenido tras su finalización.

7.1.1. Conclusiones del proyecto

El objetivo general de este Trabajo Fin de Grado consistía en la realización de sistema de de información capaz de capturar, almacenar y tratar los datos generados por dispositivos IoT desplegados por el CUM. Aunque el sistema realizado no cuenta al final con dispositivos IoT desplegados por el CUM, se ha simulado mediante un prototipo de aula donde se localizan los distintos dispositivos IoT con los diferentes sensores y actuadores. El sistema es capaz de capturar los datos utilizando el protocolo MQTT mediante conexiones realizadas por Wifi y Ethernet, es capaz de almacenar los datos en una base de datos RethinkDB y es capaz de

tratar los datos, a través de una serie de reglas que hacen al sistema autónomo. Además, proporciona una aplicación de visualización gracias a dweet.io y freeboard.io que permite a los usuarios consultar estos datos.

Este objetivo general se ha alcanzado a través de la realización de estos objetivos específicos:

- Se realizó un estudio de distintos dispositivos IoT para la obtención de los datos que se utilizan en el sistema. Se realizó una pequeña introducción a estos dispositivos, se estudió su funcionamiento y los distintos módulos de comunicación posibles para enviar y recibir datos en la red.
- Se analizaron diferentes plataformas IoT para la gestión del sistema, siendo la elegida Node-RED.
- Se analizaron diferentes bases de datos distribuidas, orientadas a documentos y de tiempo real para el almacenamiento de los datos, siendo la elegida RethinkDB.
- Se estudiaron distintas opciones de visualización de los datos, para que los usuarios del sistema pudieran consultar y conocer en tiempo real la información proporcionada por los dispositivos IoT. De entre las opciones disponibles, se eligió el uso de las aplicaciones web dweet.io y freeboard.io.
- Se realizó un estudio de distintos protocolos para la comunicación entre las diferentes partes del sistema, siendo el protocolo del nivel de aplicación MQTT el utilizado para la comunicación entre los distintos dispositivos IoT y la plataforma IoT, y la API REST para la comunicación entre la plataforma IoT y la aplicación de visualización.
- Se diseñó y desarrolló un sistema de información formado por las distintas partes analizadas y estudiadas en los puntos anteriores.
- Por último, se realizó el despliegue del sistema en un prototipo de instalación del Centro Universitario de Mérida.

Tras analizar los objetivos tanto generales como específicos, consideramos que se han cumplido correctamente. Aunque no ha sido posible su despliegue en un entorno real, este Trabajo Fin de Grado supone un primer acercamiento a que algún día el Centro Universitario de Mérida cuente con instalaciones modernas, autónomas y eficientes que permitan al centro ser un referente en dómotica.

7.1.2. Conclusiones personales

Este Trabajo Fin de Grado es el último paso para finalizar una etapa que comenzó hace cinco años, cuando pisé por primera vez el Centro Universitario de Mérida. La finalización de este proyecto no significa únicamente la finalización de una etapa, sino el comienzo de una nueva más orientada al mundo laboral.

No solo he aprendido estos años contenido relacionado con el mundo de las telecomunicaciones, más concretamente con el mundo de la telemática, sino que he adquirido una serie de competencias transversales. Algunas de estas, se han visto reforzadas en la realización de este proyecto, queriendo destacar las siguientes:

- **Búsqueda de información:** Ha sido fundamental para poder empezar y avanzar a lo largo del proyecto.
- **Comunicación escrita:** Sin duda la más destacable. A lo largo de estos años he realizado documentos, pero ninguno con este volumen, tiempo dedicado y una estructura tan correcta.
- **Planificación y organización del trabajo:** Para el correcto desarrollo de este proyecto se realizó una planificación previa y se organizó el trabajo en función de ella.
- **Comunicación en lengua inglesa:** La mayor parte de la documentación existente se encuentra en ingles.
- **Toma de decisiones:** A lo largo del desarrollo del proyecto se han tenido que ir tomando una serie de decisiones, algunas acertadas y otras no tan acertadas.
- **Comunicación oral:** Con la presentación que realizaré de este trabajo.

7.2. Trabajos futuros

Desde la amplitud del proyecto,existen múltiples enfoques de trabajo futuro sobre él:

Despliegue en un entorno real Aunque estaba propuesto inicialmente en este Trabajo Fin de Grado, debido a que las instalaciones no estaban correctamente adaptadas a las necesidades del proyecto no se pudo llevar a cabo. En un futuro, se podrían adaptar distintas instalaciones del centro, como aulas, baños o laboratorios para desplegar el sistema en un entorno real.

Añadir seguridad al sistema Se puede añadir seguridad a la comunicación entre los dispositivos IoT y la plataforma IoT, ya que MQTT [67] permite identificar y autorizar a sus clientes. Para ello, MQTT proporciona diferentes formas de definir la identidad del cliente:

- Mediante un identificador de cliente definido gracias al constructor de la clase MqttClient.
- Definiendo el ID de usuario del cliente como un atributo de la clase MqttConnectOptions.
- Mediante un certificado digital del cliente.

Se tendría que realizar un estudio para su implementación, ya que de momento las librerías utilizadas en Arduino no soportan su uso.

Además, utilizar la aplicación de visualización dweet.io de intermediario entre la plataforma IoT y la aplicación web freeboard.io supone un cierto riesgo a la seguridad, ya que si se conoce el identificador creado para la compartición de datos se puede llegar a generar datos falsos por parte de terceros. Por lo tanto, una posible mejora del sistema sería hallar la manera de conectar la plataforma IoT y la aplicación de visualización freeboard.io sin la necesidad de usar dweet.io

Mejorar el sistema de detección de presencia en la instalación En el proyecto, para averiguar si había alguna persona en la instalación se ha utilizado el sensor de movimiento PIR HC-SR501. Este sensor detecta cambios de radiación infrarroja, por lo que si la persona detectada por el sensor permanece parada sin moverse, el sensor no notará ningún cambio y parecerá que la persona ha abandonado la instalación. Por este motivo, sería interesante mejorar el sistema de detección de presencia. Existen diversas alternativas, queriendo destacar el software OpenCV [68], que se puede instalar en una Raspberry Pi. Este software detecta la presencia de personas analizando las imágenes que captura a través de una cámara basándose en la sustracción de fondo. Para que el programa funcione correctamente debe ser debidamente entrenado con una gran cantidad de imágenes como cualquier programa relacionado con la inteligencia artificial.

7.3. Planificación estimada vs Planificación real

.En la figura 7.1, se puede observar un cronograma con la planificación real de este proyecto. En comparación con la planificación estimada que se realizó en el capítulo 4 *Análisis del sistema* de este documento, se puede apreciar que es ligeramente superior. Esto se debe a varios motivos:

- Otras actividades académicas coincidieron en el mismo periodo de tiempo que este proyecto, no pudiéndole dedicar el tiempo esperado.
- Durante el desarrollo de este Trabajo Fin de Grado estuve disfrutando de una beca en el Campus Virtual de la Universidad de Extremadura, limitando parte del tiempo disponible que tenía.
- El proyecto estuvo un tiempo bloqueado a la espera de la adaptación del aula concedida para su desarrollo. Tras varios meses, se vio necesario la construcción de una maqueta debido a la imposibilidad de adaptar el aula antes de la última convocatoria para la presentación de Trabajos Fin de Estudios de este curso.

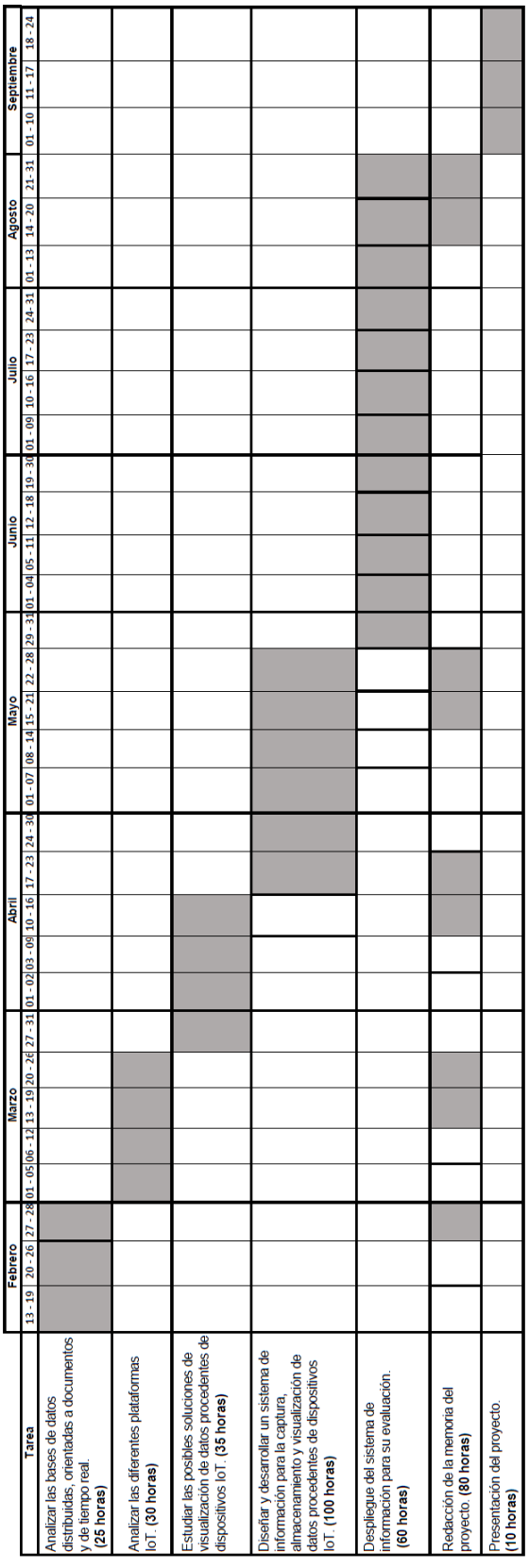

Figura 7.1: Cronograma: Planificación real.

Bibliografía

[1] *Tknika, referente en transferir la innovación a la FP y a las empresas.* Accedido el 3 de abril de 2017. Accesible en http://www.naiz.eus/eu/hemeroteca/gara/editions/2016-02-28/hemeroteca_articles/tknika-referente-en-transferir-la-innovacion-a-la-fp-y-a-las-empresas

[2] *¿Qué es el proyecto SmartPoliTech?.* Accedido el 3 de abril de 2017. Accesible en https://gim.unex.es/blogs/pablogr/2013/04/02/que-es-el-proyecto-smartpolitech/

[3] Evans, Dave. *Internet de las cosas: Cómo la próxima evolución de Internet lo cambia todo.* Accedido el 3 de abril de 2017. Accesible en http://s3.amazonaws.com/academia.edu.documents/34766160/internet-of-things-iot-ibsg.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1493137583&Signature=%2BMbdMEkHJxO7jyMSO1efv%2FG6H3s%3D&response-content-disposition=inline%3B%20filename%3DInternet-of-things-iot-ibsg.pdf

[4] Londoño Ortiz, Roby Nelson. *Internet de las cosas.* Accedido el 3 de abril de 2017. Accesible en http://ridum.umanizales.edu.co:8080/xmlui/bitstream/handle/6789/2916/informe%20final%20monografia.pdf

[5] Wikipedia: *Internet de las cosas.* Accedido el 3 de abril de 2017. Accesible en https://es.wikipedia.org/wiki/Internet_de_las_cosas

[6] Martín Cantalejo, Samuel. *Plataforma de control domótico escalable y gestionable mediante dispositivos móviles.* Accedido el 19 de mayo de 2017. Accesible en http://dehesa.unex.es/xmlui/bitstream/handle/10662/3723/TFGUEX_2015_Martin_Cantalejo.pdf?sequence=1

[7] Wikipedia: *Microcontrolador.* Accedido el 19 de mayo de 2017. Accesible en https://es.wikipedia.org/wiki/Microcontrolador

[8] *Las 3 tecnologías clave para el Internet de las cosas.* Accedido el 19 de mayo de 2017. Accesible en https://www.xataka.com/internet-of-things/las-3-tecnologias-clave-para-el-internet-de-las-cosas

[9] *Bases de datos.* Accedido el 19 de mayo de 2017. Accesible en http://volaya.github.io/libro-sig/chapters/Bases_datos.html

[10] *Bases de datos NoSQL.* Accedido el 19 de mayo de 2017. Accesible en https://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades

[11] *NoSQL vs SQL: ¿cuál elegir?*. Accedido el 19 de mayo de 2017. Accesible en http://slashmobility.com/blog/2016/12/nosql-vs-sql-cual-elegir/

[12] *¿Qué es una plataforma IoT?* Accedido el 19 de mayo de 2017. Accesible en https://secmotic.com/blog/plataforma-iot/

[13] *Desarrollo iterativo e incremental.* Accedido el 13 de junio de 2017. Accesible en https://proyectosagiles.org/desarrollo-iterativo-incremental/

[14] Project Management: *Características y fases del modelo incremental.* Accedido el 13 de junio de 2017. Accesible en http://www.obs-edu.com/es/blog-project-management/metodologias-agiles/caracteristicas-y-fases-del-modelo-incremental

[15] Fernando Berzal. *El ciclo de vida de un sistema de información.* Accedido el 13 de junio de 2017. Accesible en http://elvex.ugr.es/idbis/db/docs/lifecycle.pdf

[16] *Conceptos de la metodología orientada a objetos.* Accedido el 25 de junio de 2017. Accesible en http://profesores.fi-b.unam.mx/carlos/aydoo/conceptos_oo.html

[17] Pressman, Roger S. *Ingeniería del Software: Un enfoque práctico.* McGraw-Hill (2002)

[18] Alarcón, Raúl. *Diseño orientado a objetos con UML.* Grupo EIDOS (2000)

[19] Torrente Artero, Óscar. *ARDUINO. Curso práctico de formación.* Editorial RC Libros. Accedido el 19 de mayo de 2017. Accesible en https://books.google.es/books?hl=es&lr=lang_es&id=6cZhDmf7suQC&oi=fnd&pg=PR15&dq=arduino&ots=AZhCi_OAxN&sig=vc7Nk3mWvj171fvL9rm9m3MO91A#v=onepage&q=arduino&f=true

[20] Wikipedia: *Raspberry Pi.* Accedido el 20 de mayo de 2017. Accesible en https://es.wikipedia.org/wiki/Raspberry_Pi

[21] *Comparativa y análisis: Raspberry Pi vs competencia.* Accedido el 20 de mayo de 2017. Accesible en http://comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/

[22] *Programando NodeMCU con Arduino IDE.* Accedido el 21 de mayo de 2017. Accesible en http://www.prometec.net/nodemcu-arduino-ide/

[23] *NodeMCU con ESP8266 tarjeta Wifi.* Accedido el 21 de mayo de 2017. Accesible en http://www.geekfactory.mx/tienda/radiofrecuencia/nodemcu-esp8266-tarjeta-wifi/

[24] Méndez Blanco, Pablo. *Desarollo de un sistema de transmisión de vídeo con módulos Wifi.* Accedido el 21 de mayo de 2017. Accesible en http://oa.upm.es/43935/1/TFG_PABLO_MENENDEZ_BLANCO.pdf

[25] *Sensores de temperatura y humedad DHT11.* Accedido el 21 de mayo de 2017. Accesible en http://www.prometec.net/sensores-dht11/

[26] *Sensor de temperatura y humedad DHT22 o AM2302.* Accedido el 21 de mayo de 2017. Accesible en http://rduinostar.com/documentacion/datasheets/dht22-caracteristicas-am2302/

[27] *LDR - Fotorresitencia - Fotorresistor.* Accedido el 21 de mayo de 2017. Accesible en http://unicrom.com/ldr-fotorresistencia-fotorresistor/

[28] *Los sensores PIR. Detección de movimiento mediante un sensor piroeléctrico PIR.* Accedido el 22 de mayo de 2017. Accesible en http://www.prometec.net/sensor-pir/

[29] Página oficial de Raspberry Pi: *Camera Module.* Accedido el 22 de mayo de 2017. Accesible en https://www.raspberrypi.org/documentation/hardware/camera/

[30] Matínez Vargas, Daniel Eduardo. *Sistema de domótica para control y supervisión de una habitación de manera remota.* Accedido el 6 de abril de 2017. Accesible en https://repository.javeriana.edu.co/bitstream/handle/10554/16522/MartinezVargasDaniel Eduardo2015.pdf

[31] Sol Toledano, Víctor. *Sistema para la monitorización y asistencia en el hogar usando OpenHAB y robótica móvil.* Accedido el 6 de abril de 2017. Accesible en https://riuma.uma.es/xmlui/bitstream/handle/10630/12985/Victor%20del%20Sol%20 ToledanoMemoria.pdf

[32] Vega Alonso, Ricardo. *Instalación domótica basada en OpenHAB y Raspberry Pi.* Accedido el 6 de abril de 2017. Accesible en https://uvadoc.uva.es/bitstream/10324/19084/1/TFM-P-473.pdf

[33] Wikipedia: *Node.js.* Accedido el 7 de abril de 2017. Accesible en https://es.wikipedia.org/wiki/Node.js

[34] *Node-RED: Construye el Internet de las cosas.* Accedido el 7 de abril de 2017. Accesible en https://ricveal.com/blog/node-red-construye-el-internet-de-las-cosas/

[35] *Programación visual con Node-RED: Conectando el Internet de las Cosas con facilidad.* Accedido el 7 de abril de 2017. Accesible en https://www.toptal.com/nodejs/programación-visual-con-node-red-conectando-el-internet-de-las-cosas-con-facilidad/es

[36] *¿Qué es Node-RED?* Accedido el 7 de abril de 2017. Accesible en https://about.sofia2.com/2016/11/16/que-es-nodered/

[37] *Introducción Node-RED.* Accedido el 7 de abril de 2017. Accesible en http://raspberrynodered.blogspot.com.es/2016/07/introduccion-node-red.html

[38] Web oficial de MongoDB. Accesible en https://www.mongodb.com/

[39] García Jiménez, Antonio. *Diseño e implementación de un sistema intermediario para Internet de las cosas.* Accedido el 4 de abril de 2017. Accesible en https://riuma.uma.es/xmlui/bitstream/handle/10630/12714/AntonioGarciaJimenezmemoria.pdf

[40] Wikipedia: *MongoDB.* Accedido el 4 de abril de 2017. Accesible en https://es.wikipedia.org/wiki/MongoDB

[41] *MongoDB: qué es, cómo funciona y cuándo podemos usarlo.* Accedido el 4 de abril de 2017. Accesible en https://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

[42] Web oficial de RethinkDB. Accesible en https://www.rethinkdb.com/

[43] *RethinkDB: La base de datos para la web en tiempo real.* Accedido el 5 de abril de 2017. Accesible en http://www.openexpo.es/rethinkdb-bbdd-para-web-tiempo-real/

[44] *RethinkDB, la revolución de NoSQL, qué es y cómo instalarlo.* Accedido el 5 de abril de 2017. Accesible en https://platzi.com/blog/como-instalar-rethinkdb-docker/

[45] *RethinkDB. Una base de datos distribuida de código abierto.* Accedido el 5 de abril de 2017. Accesible en http://html5facil.com/informacion/rethinkdb-una-base-de-datos-distribuida-de-codigo-abierto/

[46] Ferreras Astorqui, Ignacio María. *Sensor IoT para la monitorización de consumo de energía en continua.* Accedido el 23 de mayo de 2017. Accesible en http://eprints.ucm.es/38565/1/tfg.pdf

[47] *Make Your Dashboard Graphs Using Graphene Toolkit.* Accedido el 23 de mayo de 2017. Accesible en http://yemista.com/make-your-dashboard-graphs-using-graphene-toolkit/

[48] Bonilla Agualongo, Paúl Agustín. *Diseño e implementación de un sistema de control en red, utilizando sensores y actuadores inteligentes para los mecanismos de control, monitoreados mediante una aplicación por la nube.* Accedido el 23 de abril de 2017. Accesible en http://www.dspace.ups.edu.ec/bitstream/123456789/13078/1/UPS%20-%20ST002256.pdf

[49] Gómez Fuentes, Rubén y Lago Aguado, Daniel. *Inteligencia ambiental en el Internet de las Cosas.* Accedido el 23 de abril de 2017. Accesible en http://eprints.ucm.es/38509/1/Memoria %20TFG.pdf

[50] *Sensor IoT para monitorizar la cadena de frío en el transporte y almacenamiento de alimentos.* Accedido el 23 de abril de 2017. Accesible en http://eprints.ucm.es/39875/1/MEMORIA %20FINAL %20Freeze %20Sense.pdf

[51] Web oficial de dweet.io. Accesible en https://dweet.io/

[52] *Una interfaz sencilla para el Internet de las cosas.* Accedido el 24 de abril de 2017. Accesible en https://www.technologyreview.es/s/4168/una-interfaz-sencilla-para-el-internet-de-las-cosas

[53] *Visualización de datos en la nube. Internet de las cosas.* Accedido el 24 de abril de 2017. Accesible en http://www.practicasconarduino.com/manualrapido/visualizacin_de_datos_en_la_nube_internet_de_las_cosas.html

[54] *Enviar datos a Dweet.io.* Accedido el 24 de abril de 2017. Accesible en http://portodosmisbytes.blogspot.com.es/2014/11/iot-enviar-datos-a-dweet-desde-un-powershell.html

[55] Web oficial de Freeboard. Accesible en http://freeboard.io/

[56] *Simple IoT Dashboards with Dweet.io & Freeboard.io.* Accedido el 25 de abril de 2017. Accesible en https://www.robertputt.co.uk/2017/01/01/simple-iot-dashboards-with-dweet-io-freeboard-io/

[57] Yébenes Gálvez, José Antonio. *¿Qué es MQTT?* .Accedido el 12 de junio de 2017. Accesible en https://geekytheory.com/que-es-mqtt

[58] Oasis Standard: *MQTT Version 3.1.1.* Accedido el 12 de junio de 2017. Accesible en http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[59] Juárez Manzano, José Luis. *Protocolos IoT: la siguiente revolución tecnológica.* Accedido el 15 de junio de 2017. Accesible en http://vintegris.info/protocolos-iot-001/

[60] *Protocolos del Internet de las Cosas.* Accedido el 15 de junio de 2017. Accesible en http://equipo.altran.es/protocolos-iot-internet-de-las-cosas/

[61] *MQTT: un protocolo específico para el internet de las cosas.* Accedido el 15 de junio de 2017. Accesible en http://www.digitaldimension.solutions/es/blog-es/opinion-de-expertos/2015/02/mqtt-un-protocolo-especifico-para-el-internet-de-las-cosas/

[62] Miguel Gracia, Luis. *¿Qué es CoAP?* Accedido el 15 de junio de 2017. Accesible en https://unpocodejava.wordpress.com/2013/04/28/que-es-coap/

[63] Gimeno Gimenez, Xavi. *Desarrollo y Estudio del Protocolo Observe para CoAP.* Accedido el 15 de junio de 2017. Accesible en http://webcache.googleusercontent.com/search?q=cache:6TXiz-Jx3ZEJ:upcommons.upc.edu/bitstream/handle/2099.1/18314/Xavi_Gimeno_Gimenez.pdf%3Fsequence%3D1+&cd=4&hl=es&ct=clnk&gl=es

[64] Wikipedia: *Extensible Messaging and Presence Protocol.* Accedido el 15 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol

[65] Wikipedia: *Transferencia de Estado Representacional.* Accedido el 15 de junio de 2017. Accesible en https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[66] *Servicios web: ¿Qué es REST?* Accedido el 15 de junio de 2017. Accesible en https://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-es-rest/

[67] IBM: *Seguridad de MQTT.* Accedido el 30 de agosto de 2017. Accesible en https://www.ibm.com/support/knowledgecenter/es/SS9D84_1.0.0/com.ibm.mm.tc.doc/tc00150_.htm

[68] *Entrenar OpenCV en Detección de Objetos.* Accedido el 30 de agosto de 2017. Accesible en http://acodigo.blogspot.com.es/2015/12/entrenar-opencv-en-deteccion-de-objetos.html

Capítulo 8

Anexo I. Instalaciones

En este capítulo se recogen los procesos de instalación de los distintos programas que se han necesitado para el desarrollo de este proyecto.

En la sección 8.1 se muestran los pasos a seguir para la instalación tanto del servidor como del cliente MQTT.

En la sección 8.2 se muestra el proceso de instalación de la plataforma IoT Node-RED.

En la sección 8.3 se muestran los pasos para instalar la base de datos RethinkDB.

8.1. Instalación MQTT

Para poder usar el protocolo MQTT es necesario la instalación de un broker. Un broker es la aplicación encargada de la gestión y uso del protocolo, tanto como servidor como cliente. El broker utilizado en este proyecto se llama Mosquitto y se ha instalado en una Raspberry Pi 2 con el sistema operativo Raspbian. Los pasos a seguir para su instalación son los siguientes:

- El primer paso será actualizar nuestros repositorios, para ello ejecutamos el siguiente comando en el terminal:

Algoritmo 8.1 Actualizar los repositorios.

```
$ sudo apt-get update
```

- Tras esto, instalaremos el servidor MQTT ejecutando el siguiente comando:

Algoritmo 8.2 Instalar servidor MQTT.

```
$ sudo apt-get install mosquitto
```

- Por último, instalamos el cliente MQTT:

Algoritmo 8.3 Instalar cliente MQTT.

```
$ sudo apt-get install mosquitto-clients
```

Una vez seguido los pasos anteriores ya se tendrá instalado tanto el cliente como el servidor MQTT.

8.1.1. Funcionamiento MQTT

Para subscribirse a un topic es necesario ejecutar el siguiente comando:

Algoritmo 8.4 Subscribirse a un topic.

```
$ mosquitto_sub -d -h <IpServidorMosquitto> -t "<topic>"
```

Donde:

- **-d** indica que se muestren los mensajes que se reciban de manera detallada.
- **-h** indica la IP donde se ha instalado el broker Mosquitto.
- **-t** indica el topic al cual subscribirse.

Para enviar un mensaje con un topic basta con ejecutar el siguiente comando:

Algoritmo 8.5 Enviar un mensaje con un topic.

```
$ mosquitto_pub -h <IpServidorMosquitto> -t "<topic>" -m "<mensaje>"
```

Donde *-m* indica el mensaje que se desea publicar.

8.2. Instalación Node-RED

La plataforma IoT se ha instalado en una Raspberry Pi 2 con el sistema operativo Raspbian para su funcionamiento en el proyecto. Los pasos a seguir para su instalación son los siguientes:

- El primer paso será actualizar nuestros repositorios, para ello ejecutamos el siguiente comando en el terminal:

Algoritmo 8.6 Actualizar los repositorios.

```
$ sudo apt-get update
```

- Tras esto, instalaremos Node-RED ejecutando el siguiente comando:

Algoritmo 8.7 Instalar Node-RED.

```
$ sudo apt-get install nodered
```

- Seguidamente, instalamos el manejador de paquetes por defecto de Node.js:

Algoritmo 8.8 Instalar Node-RED. Manejador de paquetes.

```
$ sudo apt-get install npm
```

- Una vez instalado, descargamos todos los paquetes necesarios de Node-RED:

Algoritmo 8.9 Instalar paquetes necesarios para Node-RED.

```
$ sudo npm install -g --unsafe-perm node-red
```

Una vez realizado los pasos anteriores, se podrá ejecutar la plataforma IoT con el siguiente comando:

Algoritmo 8.10 Ejecución Node-RED.

```
$ node-red
```

Por último, para usar la interfaz de usuario de la plataforma es necesario acceder a la direccion *http://localhost:1880* desde un navegador web.

8.3. Instalación RethinkDB

La base de datos RethinkDB se instaló en un equipo con Windows 10 para la realización de este proyecto. Los pasos a seguir para su instalación son los siguientes:

- En primer lugar, se deberá acceder a la página oficial de RethinkDB a través del siguiente enlace: *https://www.rethinkdb.com/*
- Una vez dentro de la página, se deberá pulsar en la opción *'Install RethinkDB'*, disponible la parte superior central de la misma.
- Tras pulsar, se mostrará un listado de sistemas operativos compatibles con su instalación. Se deberá pulsar en este caso la opción *'Windows'*.
- Aparecerá una nueva ventana donde se informa que solo se podrá utilizar sistemas operativos de 64 bits a partir de Windows 7. Para comenzar la descarga basta con pulsar en el botón *'Download'*.
- Se descargará un archivo ZIP que contiene una carpeta. Es necesario extraer esa carpeta en el lugar donde se quiera instalar la base de datos.
- Una vez descomprimido el archivo, se accederá a la carpeta y se pulsará sobre el archivo *'rethinkdb.exe'*. Tras esto, se creará una carpeta en el mismo directorio donde se encuentra el archivo ejecutable llamada *'rethinkdb_data'*.

Ya estará instalada la base de datos RethinkDB en el equipo. Para acceder a su interfaz de usuario basta con acceder a la dirección *http://localhost:8080* desde un navegador web.