uc3m | Universidad Carlos III de Madrid

Grado Universitario de Ingeniería Informática

Curso 2019-2020

Trabajo Fin de Grado

"Alternativas a un sistema domótico basado en IoT"

Alfonso López-Contreras Martín

Tutora

María Calderón Pastor

Colmenarejo, Septiembre 2020

Agradecimientos

Aprovecho la oportunidad que me da este apartado para agradecer a mi familia, profesores y amigos, toda la ayuda prestada teniendo en cuenta las dificultades de comunicación debidas a la pandemia que hemos sufrido durante la elaboración de este trabajo.

En especial a mi tutora que, entre otras cosas, me entregó parte de los dispositivos en un parking en cuanto tuve oportunidad de ir a Madrid después del confinamiento.

Contenido

Índice de Figuras

Índice de Tablas

Lista de abreviaturas

IoT	Internet Of Things (Internet de las Cosas)
IA	Inteligencia Artficial
BLE	Bluetooth Low Energy (Bluetooth baja energía)
GW	Gateway
LOPD	Ley Orgánica de Protección de Datos
SBC	Single Board Computer (Ordenador en una sola placa)
Raspberry	Raspberry Pi
I2C	Inter-Integrated Circuit (Circuito Inter-Integrado)
BD	Base de Datos
COAP	Constrained Application Protocol (Protocolo de aplicación restringida)
DTLS	Datagram Transport Layer Security
SO	Sistema Operativo
JSON	JavaScript Object Notation
BSON	Binary JSON

Resumen

En este trabajo se analizan dos alternativas a un sistema de domótica basado en IoT.

La primera, una arquitectura basada en un despliegue local, donde hay un sensor que detecta distintas medidas como temperatura, luz o presión y las envía al sistema que se encuentra en una Raspberry.

La implementación cuenta con un programa para procesar, guardar y mostrar las distintas medidas detectadas. De acuerdo con los valores recibidos, el sistema permitirá actuar de forma automática sobre distintos dispositivos como una bombilla o un altavoz.

La segunda opción se respalda en la nube. En cuanto a funcionalidad es parecida a la solución local, pero en este caso los datos son guardados y mostrados en una plataforma IoT en la nube, accesible desde cualquier dispositivo con acceso a internet, sin que sea posible manejar el sistema a través de este medio.

1 Introducción

1.1 Motivación del trabajo

El IoT (Internet Of Things) es una forma de conectar diferentes tipos de dispositivos a internet para recibir datos o emitir ordenes, la recogida de datos mediante sensores que recaban información del entorno para recibirla en otro sitio a través de internet y la emisión de ordenes sirve para controlar actuadores a través de internet. Por ejemplo, un sensor de temperatura que envía los datos al usuario y este desde su móvil envía la orden de activar el termostato para subir o bajar la temperatura.

La tecnología IoT cada vez tiene más presencia en nuestras vidas ya que los sensores son más baratos que hace unos años y se pueden conectar a prácticamente cualquier aparato

Además, la conectividad ha mejorado mucho por lo que es posible mandar los datos recogidos en los sensores a donde van a ser leídos y analizados de la manera que necesitemos (por ejemplo, ahora es más viable la conexión de varios sensores a la nube, cosa que hace unos años sería más complicada y/o más costosa).

Gracias a todo esto se estima que en 2020 habrá unos 30.000 millones de dispositivos IoT, este crecimiento conlleva que se deban tener las siguientes consideraciones para el futuro [1] [2]:

- La IA (Inteligencia Artificial) será más necesaria, al recibir gran cantidad de datos de diferentes sensores es muy útil para analizarlos de diferentes maneras.
- Cuando se despliegue de manera efectiva el 5G, los sensores en lugares que no tengan internet por cable crecerán. El envío/recepción de datos será más rápido y podremos mandar más cantidad.
- Avanzará el Edge Computing, es decir, se descentralizará la carga de procesamiento de una red digital para minimizar la latencia/ancho de banda de una red, gracias a que los datos se procesarán mediante dispositivos como la Raspberry que estén cercanos a los sensores.
- Será necesario un nuevo modelo de seguridad porque la incorporación de dispositivos tanto a entornos domésticos como industriales con una estructura centralizada puede ser una fuente de vulnerabilidad. Existen

muchas propuestas algunas muy novedosas proponen un modelo de seguridad basado en el Blockchain.

Dado que el IoT es tan barato y sencillo de implementar, puede encontrarse en multitud de campos [3] como, por ejemplo:

- **Domótica**: cada vez es más habitual que el IoT esté integrado en diferentes electrodomésticos y estos estén conectados a un aparato que hace de centro de control para todos ellos, los más populares son los de Google y Amazon que funcionan por voz o a través de su aplicación.
- **Salud**: en este campo empieza a haber diferentes aparatos que permiten supervisar a pacientes a través de sensores sin que tengan que ir físicamente al hospital para controlarlos.
- **Industria**: los dispositivos IoT se usan para recoger todo tipo de datos, desde comprobar que las distintas máquinas funcionan correctamente hasta aparatos que comprueban la actividad de los empleados y su productividad.
- **Seguridad**: en este ámbito se están empezando a emplear para recibir los datos de seguridad de los sensores (cámaras, sensores de movimiento…) en tiempo real a través de internet, aunque al estar en desarrollo muchas de ellas son vulnerables y sólo se usan a nivel doméstico o en pequeños negocios.
- **Otros usos**: entre ellos se encuentra la utilización en ciudades para hacerlas más eficientes, por ejemplo, sensores para activar los semáforos y que el tráfico sea más fluido o sensores que detecten fallos en el alumbrado o la distribución de agua. Otro ejemplo, sería su uso en los coches, donde sirve para controlar música, temperatura, etc. Dentro de este mismo campo se utiliza para los primeros intentos de piloto automático de diferentes compañías.

1.2 Objetivos del proyecto

El objetivo de este trabajo es comparar dos opciones a la hora de guardar los datos procedentes de un sensor, tratarlos para ser usados en dispositivos IoT y visualizarlos para, por último, producir acciones en diferentes actuadores. Las opciones son las siguientes:

Local: sin necesidad de conexión a internet, un sensor CC2650 se conecta a una Raspberry por BLE (Bluetooth Low Energy) y mediante node-red se tratan los datos recibidos para almacenarlos en local y según sus valores tiene diferentes efectos como encender una luz en caso de que el sensor reciba poca luz. Los datos se muestran de manera local en el dashboard que tenemos en node-red. Además, cuenta con un sistema de alertas basado en Google Nest, que avisa si existen temperaturas muy altas o si el sensor se ha movido mediante el propio altavoz del Google Nest.

En la nube: lo mismo que antes, pero conectando la Raspberry a la nube y que los datos se guarden en la plataforma de Watson IBM IoT para poder ser monitorizados a través de internet. En este último caso sería posible conectar la parte de control del sistema a través de internet para controlar la iluminación, pero no ha sido implementado.

Además, en ambos se pueden tener alertas de temperatura por mail y usar el Google Nest para cambiar el estado de la bombilla y obtener los datos del sensor, pero necesitan internet para funcionar ya que el reconocimiento de voz lo hace enviándolo a Google y el mail necesita el servidor de Gmail.

Lo que se pretende con este trabajo es crear un sistema de bajo coste, robusto, seguro y escalable que permita monitorizar en tiempo real la temperatura, luz, presión y humedad de las distintas partes de una casa y controlar la iluminación en estas (aunque podría incluirse el control de un termostato, por ejemplo).

1.3 Marco regulador

Como el IoT recoge datos domésticos como la temperatura del hogar, cuando se conectan diferentes electrodomésticos o las entradas/salidas de los usuarios o datos industriales con sensores conectados para controlar distintas máquinas, se deberá tener muy en cuenta la protección de datos, estos deben ser tratados con seguridad y tanto su uso por terceras personas como el almacenamiento de estos debe ser consensuado y con arreglo a la LOPD.

1.4 Entorno socioeconómico

Como se puede leer en el apartado anterior, el IoT se está convirtiendo en algo crucial en cualquiera de los colectivos de la sociedad, desde los procesos industriales a la domótica en el hogar. En este apartado se tratará cómo influye esta nueva tecnología desde un punto de vista social y económico.

Según un estudio hecho por la multinacional francesa Capgemini [4] se puede ver el impacto del IoT según las diferentes industrias (figura 1).

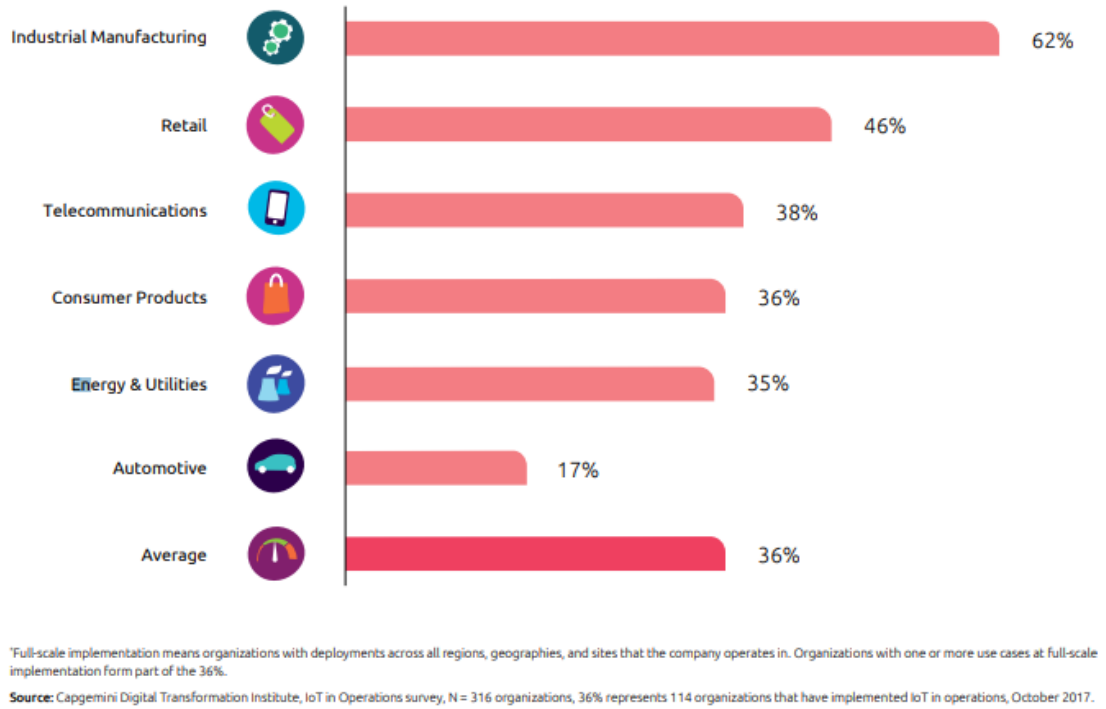

Fig. 1 Principales industrias de IoT. Fuente: [4].

Es bastante significativo que en industrias donde los procesos ya están automatizados, se adopte menos el IoT, esto se debe a que realmente no encuentran ninguna justificación para usarlo que mejore notablemente los procesos ya existentes. Sin embargo, en industrias donde no existen estos procesos, se implanten mejor ya que supone un gran avance (por ejemplo, instalar sensores en oleoductos que informen constantemente de la estabilidad del sistema, comprobando el estado del material para prevenir cualquier rotura o fuga).

Haciendo una comparativa por países [4] (figura 2), se puede ver la importancia que tiene en las sociedades avanzadas la existencia de multitud de *start-ups* que encuentren nuevos casos de uso en comparación a sociedades que pueden tener la tecnología pero no encuentran la necesidad de su uso habitual.

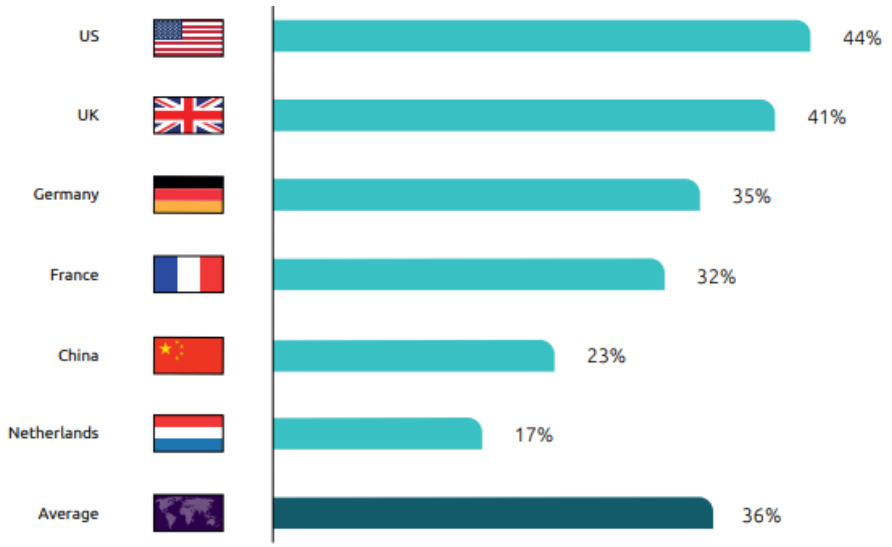

Fig. 2 Principales países con uso de IoT. Fuente: [4].

En España, según un estudio realizado por Telefónica [5], el IoT de uso personal ha tenido el siguiente crecimiento respecto a hace 2 años. Se puede ver en la figura 3:

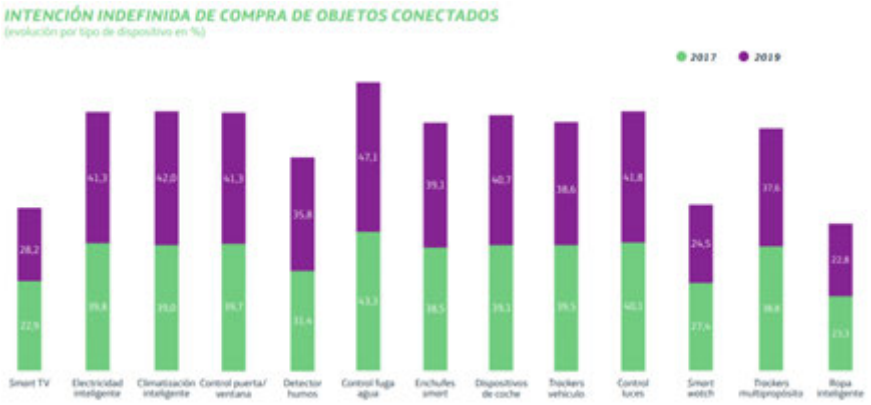

Fig. 3 Principales compras de dispositivos IoT en España. Fuente: [5]

Es posible observar que la mayoría de los dispositivos están dentro del ámbito doméstico, por tanto, este proyecto se enmarca en los avances de uno de los sectores más crecientes del IoT.

Una de las mejoras necesarias para que esta tecnología sea más usada es la seguridad. Es necesario entender que la mayoría de los dispositivos IoT se basan en la simplicidad de su uso y creación, lo que hace que sean más vulnerables que un sistema más complejo. Por tanto, puede frenar el despliegue de esto por las preocupaciones (por ejemplo, hace dos semanas, unos investigadores consiguieron violar la seguridad del dispositivo Alexa de Amazon, abriendo la posibilidad de grabar a su entorno si escuchaba ciertas palabras [6]) de los usuarios tanto a nivel doméstico como industrial por las posibles violaciones a su intimidad/propiedad intelectual.

De aquí que sea muy importante el desarrollo legislativo para evitar que las grandes corporaciones puedan usar los datos recogidos por los sensores IoT para su propio beneficio y no para el uso acordado en el producto.

Como conclusión, es un sector creciente de gran impacto sobre la sociedad y los negocios. Cada vez crecerá más con la implementación de tecnologías futuras como el 5G, que extiendan la accesibilidad de internet y la cantidad de datos que es posible enviar desde cualquier punto del planeta.

1.5 Estructura de la memoria

En esta sección se pretende hacer un recorrido del trabajo explicando cada una de las secciones que lo estructuran.

En primer lugar, en el estado del arte, se presentan brevemente las tecnologías ya existentes que son parecidas a las que serán implementadas y se comparan entre ellas.

Tecnologías usadas es un capítulo dónde, como su propio nombre indica, se explica qué medios se usarán en el trabajo, tanto software como hardware.

El siguiente capítulo es estructura a alto nivel del sistema, aquí se explica la arquitectura de las diferentes opciones implementadas y los requisitos funcionales o no funcionales que tienen ambas.

A continuación, se entra en la configuración e instalación de la solución, esta parte es donde se explica cómo se ha implementado el sistema a nivel hardware y software.

Una vez configurado el sistema, la siguiente sección está dedicada a las pruebas y comparación de ambas arquitecturas, en la parte de pruebas se explican cómo se han hecho y qué resultados se espera de ellas. La comparación de las arquitecturas sirve para comentar las ventajas y desventajas que tienen las distintas soluciones al problema.

El próximo apartado habla de los aspectos más importantes para que un proyecto sea viable, los presupuestos necesarios para crear el sistema, haciendo un análisis de los costes de personal y material software/hardware.

Por último, se analiza todo lo escrito anteriormente para llegar a una conclusión sobre el proyecto.

2 Estado del arte

Existen diferentes tipos de sistemas domóticos comerciales que recogen datos del entorno doméstico y controlan sistemas como luces o termostatos. Los más relevantes en la actualidad son los siguientes:

2.1 Samsung SmartThings

Es la opción de Samsung [7] para acercarse a la domótica, tiene una gran variedad de productos a su disposición (figura 4), como la mayoría de los sistemas, incluido el presentado en este trabajo, tiene control de luces/temperatura. La diferencia fundamental respecto a lo desarrollado en este trabajo es que utiliza un sensor para cada medida y este trabajo usa un solo sensor para todas las medidas, con ello se ahorra costes y espacio.

La empresa provee el software y algunas partes del hardware (por ejemplo, el sensor de luz puede ser de Philips Hue (figura 5) y las bombillas del propio Samsung).

Fig. 4 Un ejemplo de variedad de productos. Fuente: [7].

Fig. 5 Sensor de luz y movimiento de Philips Hue. Fuente: [8]

2.2 Google Nest

Google tiene su propio sistema de domótica, está integrado con una empresa llamada Nest. Esta, a diferencia de Samsung, tiene un número más limitado de sensores y aparatos, centrados en el control de temperatura, la seguridad doméstica y los productos de Google (básicamente altavoces inteligentes y Chromecast) como se puede ver en la figura 6, aunque cada vez se pueden añadir más dispositivos de otras marcas.

Su integración es parecida a la anterior, con un sensor diferente para cada medida y necesitando un hub (un sistema central) al que conectar todo.

Fig. 6 Gama de productos nest. Fuente: [9].

2.3 Bosch Smart Home

Las soluciones comercializadas por Bosch son similares a las vistas anteriormente de Google y Samsung, tiene dos gamas, una como Google, centrada en luz/temperatura/seguridad llamada Bosch Smart Home (figura 8) y otra para integrar electrodomésticos a esta red llamada Bosch Home Connect. En la figura 7 podemos ver un sensor que hace algo similar al sistema, detecta si falta luz en la casa y la enciende para hacer parecer que hay gente en casa, a mayores de estar programada para encenderse en unos horarios determinados.

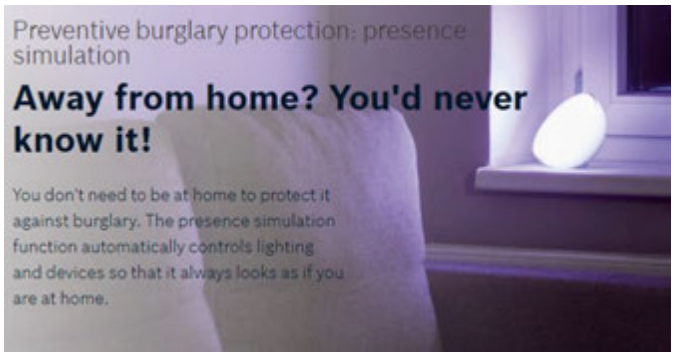

Fig. 7 El sensor más parecido al nuestro. Fuente: [10].

Fig. 8 Gama de productos Bosch Smart System. Fuente: [11]

Hay alternativas parecidas a este trabajo que son open-source (código abierto) permitiendo usar sensores/actuadores independientemente del fabricante, incluso usando sistemas cerrados como el Apple Homekit. Algunos ejemplos son: openHab, Home Assistant, OpenMatics, etc. Muchas de estas se integran dentro de una Raspberry debido a su facilidad de uso y su tamaño.

Como se ha comentado antes, aunque existen muchas alternativas parecidas a nuestro sistema, muchas requieren sensores individuales para cada medida, lo que incrementa el coste bastante, mientras que el nuestro con un solo sensor se puede obtener diferentes medidas y controlar temperatura/presión/luz/etc. Además de que es más personalizable pues con la herramienta de programación visual node-red es posible integrar dispositivos de diferentes marcas sin mucho problema.

3 Tecnologías usadas

En este capítulo se comentará el estado de las diferentes tecnologías usadas en el desarrollo del sistema.

3.1 Hardware

En esta sección enseñaremos los diferentes componentes hardware del sistema

3.1.1 Raspberry Pi

Es un SBC (Single Board Computer), un ordenador que funciona en un solo circuito, con todos sus componentes soldados a esta placa a los que conectamos diferentes dispositivos externos para dar otras funcionalidades como sensores, dispositivos de e/s, almacenamiento, etc.

Aunque existen muchas opciones de SBC en el mercado (por ejemplo, Arduino o Gumstix), Raspberry Pi es la más completa para nuestro propósito ya que cuenta con BLE integrado y permite usar Raspberry Pi OS, que será explicado más tarde, en concreto se usa la Raspberry Pi 4 de 4GB de RAM para obtener mejor rendimiento, cuando se empezó a realizar el trabajo no existía la de 8. Los componentes de la Raspberry Pi están detallados en la figura 9.

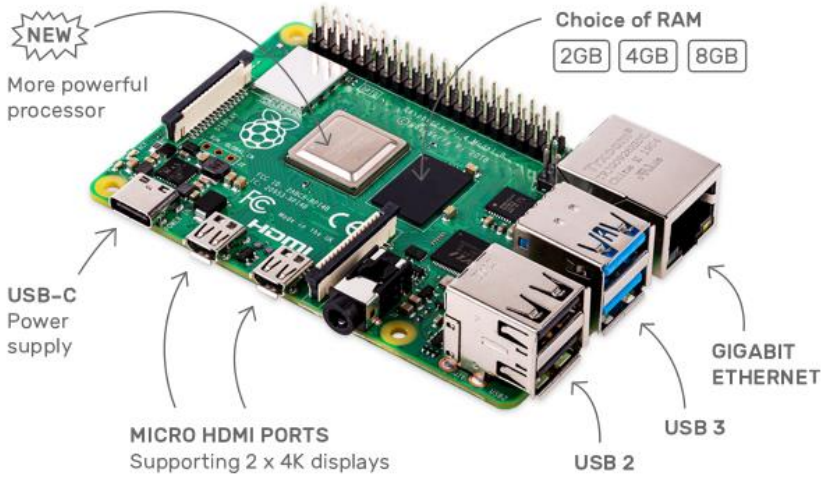

Fig. 9 Esquema de una Raspberry Pi. Fuente: [12]

3.1.2 Sensortag Texas Instruments CC2650

Es un microcontrolador (un chip programable) con diferentes sensores integrados [13] que se puede conectar mediante distintos sistemas inalámbricos como BLE o Zigbee. Tiene los siguientes sensores:

1. Micrófono
2. Altímetro
3. Sensor de luz
4. Sensor de movimiento (giroscopio + acelerómetro + brújula)
5. Sensor de humedad y temperatura
6. Sensor de temperatura mediante infrarrojos

La estructura interna del sensorTag se presenta en la figura 10, donde se puede ver que la mayoría de los sensores se comunican por I2C, un sistema que se usa principalmente para comunicaciones entre diferentes partes de un circuito.

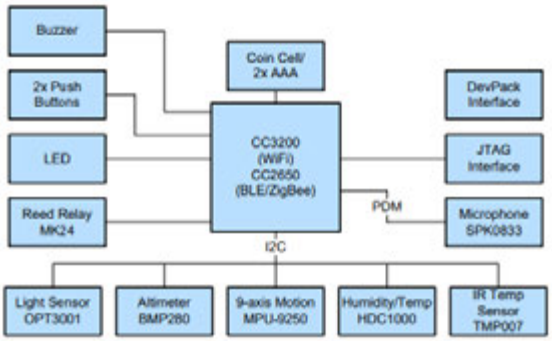

Fig. 10 Diagrama de bloques del SensorTag CC2650. Fuente: [13]

Un esquema a más alto nivel es el mostrado en la Figura 11, dónde se ve cómo se ubican los diferentes sensores en el microcontrolador físico.

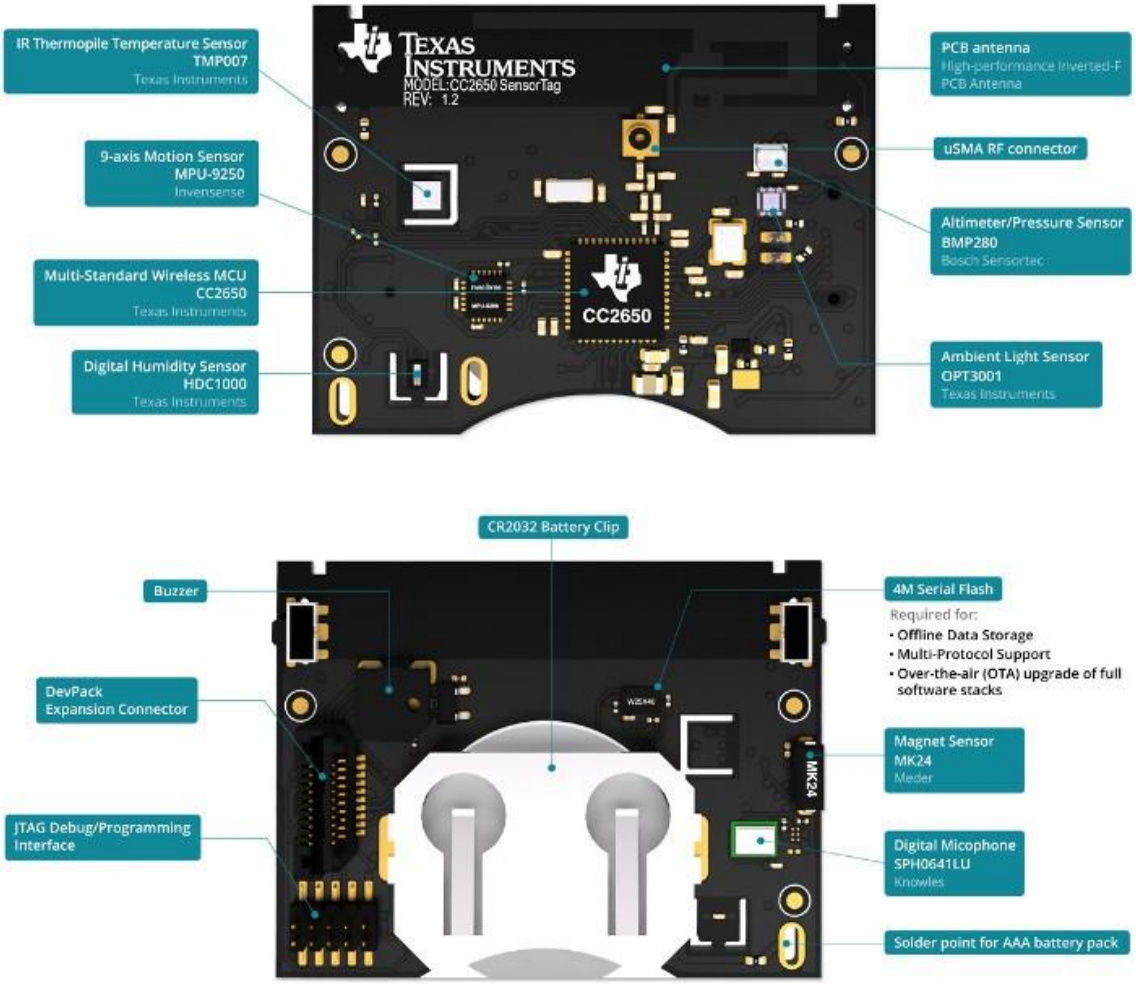

Fig. 11 Imágenes del SensorTag CC2650. Fuente: [6]

3.1.3 Trädfri

Es un conjunto de aparatos desarrollado por IKEA para tener un sistema de iluminación en la casa. El sistema permite controlar las luces de forma remota usando un mando, una aplicación en el móvil, o automática mediante un sensor de movimiento. En la solución de IKEA las bombillas se conectan a una Gateway Trädfri que hace de sistema intermedio (ver Figura 12). En este trabajo las luces se controlan de forma automática o manual conectando directamente la Raspberry al Gateway (una puerta de enlace, permite la comunicación entre dos dispositivos), sin necesidad de mando o móvil. Consta de las siguientes partes que se pueden ver en la Figura 12:

a. Bombilla – Se conecta al Gateway mediante Zigbee y recibe las órdenes de este para encenderse/apagarse, cambiar la intensidad o el color.

b. Gateway – Está conectado mediante ethernet a la red y los dispositivos que estén en la misma red pueden mandarle peticiones para controlar la bombilla.

c. Mando – Se conecta al Gateway por medio de zigbee y sirve para controlar las bombillas de manera manual.

Fig. 12 Aplicación del móvil, el mando, dos bombillas y el Gateway. Fuente: [14].

3.1.4 Google Nest Mini

Es un altavoz con un asistente integrado de la gama Nest de Google, centrada en domótica, se conecta al router mediante wifi y se usa para emitir alertas del sistema. Además, permite controlar el sistema Tradfri siempre y cuando esté en la misma red que el Gateway de Ikea. Además, tiene un sistema de reconocimiento de voz si está conectado a internet, lo que permite usarlo como control por voz.

3.2 Software

En esta sección se explicarán las herramientas y software usados en el desarrollo del sistema.

3.2.1 Node-red

Es un programa open source creado por IBM que permite programar con diagramas de flujo, funciona con el lenguaje JavaScript y guarda los datos en JSON. Su editor está basado en la web [15] y al iniciarlo proporciona una dirección IP a la que conectarse para crear los diagramas. Permite la inclusión de nodos creados por los usuarios, por lo que tiene una gran versatilidad. Está especialmente pensado para conectar distintos dispositivos hardware, ya que al tener un diseño de "cajas negras" (no es necesario saber ¿? cómo funciona por dentro cada nodo, sólo configurar sus valores según haya dispuesto el creador y conectarlos al diagrama). Esta aproximación hace que la resolución de problemas sea tan fácil como ver qué está conectado y cómo está conectado (figura 13).

Se pueden tener varios diagramas de flujo abiertos a la vez para tener mejor organizados los procesos y se guardan en formato JSON, por lo que importar o exportar un flujo es tan fácil como copiar o pegar una cadena de texto. Por ejemplo, si se quiere traspasar a otra persona sólo hay que mandar la cadena de texto y el propio node-red lo interpretará, aunque no tenga los nodos instalados, que avisará de que no existen y sugerirá instalarlos.

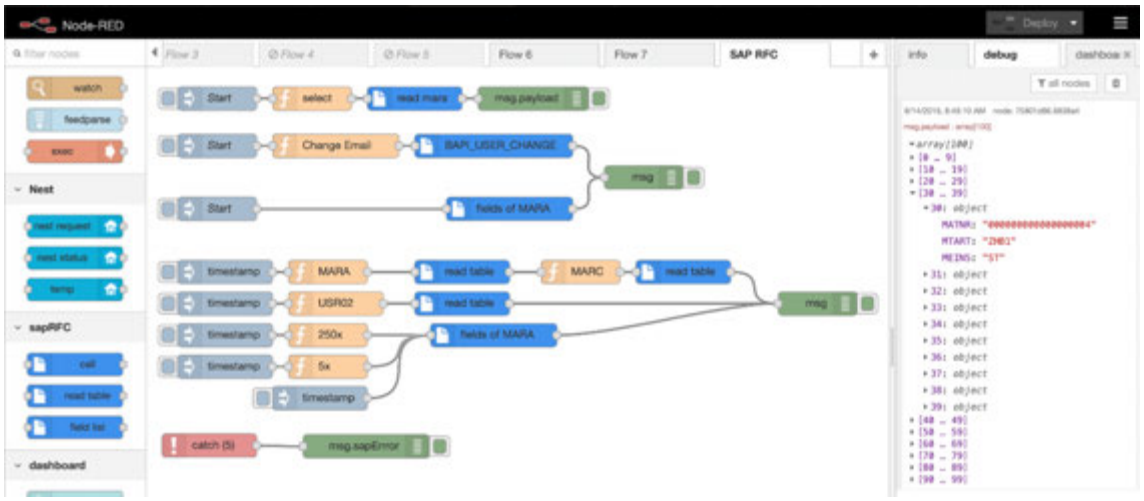

Fig. 13 Un ejemplo de cómo se ven los esquemas. Fuente: [16].

Por último, al ser muy ligero viene preinstalado en la Raspberry, así que no hace falta instalarlo y cuenta con un nodo para el desarrollo de dashboards (Fig. 14)

Fig. 14 Ejemplo del dashboard. Fuente [17].

3.2.2 Mongodb

Es un programa [18] que sirve para crear bases de datos NoSQL. Un tipo de base de datos que no tiene por qué ser relacional, no está basada en el lenguaje SQL ni usa tablas para almacenar la información. MongoDb guarda la información en colecciones dentro de la BD (Base de Datos) en un formato BSON (Binary JSON), muy parecido a JSON (JavaScript Object Notation). Las búsquedas también se hacen en JSON, por lo que es muy fácil encontrarla con distintos campos.

También cuenta con un servicio (MongoDB Atlas) de monitorización en la nube muy parecido al de Watson IBM, que recoge los datos locales, los sube a la red y se pueden configurar alertas y distintas mediciones. Sin embargo, se decidió usar Watson IBM porque Atlas está en beta y no existe tanta información sobre cómo integrarlo en el sistema como con la propuesta de IBM.

La decisión de usar esta tecnología en lugar de cualquier otra BD tanto SQL como NoSQL es que inicialmente se pensaba utilizar Graphana como dashboard (alternativa que finalmente se descartó) y por ello se eligió una BD compatible con Graphana) y de entre las probadas MongoDB era la única que no daba problemas al ejecutarse en Raspberry Pi OS.

Se intentó usar Graphana e InfluxDB como opción para el dashboard y la base de datos, pero InfluxDB no conseguía crear una BD en la Raspberry y Graphana no es compatible con MongoDB, así que se decidió usar el otro sistema.

3.2.3 Python

Es el lenguaje de programación que se usará para recoger todos los datos del sensor. Python es un lenguaje de alto nivel y bastante versátil [19], con un tipado dinámico (no se definen directamente el tipo de las variables) y usa diferentes paradigmas, como programación orientad a objetos. Se usará un script (un conjunto de código) previamente obtenido hecho en la versión 2.7 que se encarga de conectarse al sensor desde la Raspberry y empezar a devolver todos los mensajes que recibe del sensor. El script se iniciará desde node-red.

3.2.4 Raspberry Pi OS

Es el Sistema Operativo que se usará con la Raspberry Pi, está basado en Debian (un SO GNU/Linux basado en software libre) porque no existía una versión que funcionase en la CPU de la Raspberry Pi, así que la propia fundación Raspberry Pi creó una edición para que funcionara. Será utilizado por varias razones, es el recomendado por la propia fundación y es compatible con todo el software que se necesita.

3.2.5 IFTTT

Es un sistema web [20] que se encarga de encadenar diversas acciones (IFTTT son las siglas de If This, Then That (si esto, entonces aquello)). En este caso, encadena los comandos de voz con su acción correspondiente.

3.2.6 Webhookrelay

Un webhook es una retrollamada HTTP, al hacer una solicitud a una URL, el webhook hace otra solicitud a una URL dada por el usuario. Webhooks es un servicio que gestiona estos webhooks, en el problema actual, recibe la llamada hecha por IFTTT después de un comando de voz y manda a node-red el resultado.

3.3 Tecnologías de conexión usadas

En esta sección se enseñarán las distintas tecnologías que usaremos para conectar las partes del sistema

3.3.1 Ethernet

Es un estándar para redes locales que se conectan a través de cableado físico que hace pasar los diferentes tramas (parte de los datos de un mensaje) por la red. Se usa por su velocidad y su facilidad de conexión.

Estas tramas que viajan por la red tienen la estructura indicada en la figura 15.

Estructura de la trama de 802.3 Ethernet

Preámbulo	Delimitador de inicio de trama	MAC de destino	MAC de origen	802.1Q Etiqueta(opcional)	Ethertype (Ethernet II) o longitud (IEEE 802.3)	Payload	Secuencia de comprobación (32-bit CRC)	Gap entre frames
7 Bytes	1 Byte	6 Byte	6 Bytes	(4 Bytes)	2 Bytes	De 46 (o 42) hasta 1500 Bytes	4 Bytes	12 Bytes
		64–1522 Bytes						
72–1530 Bytes								
84–1542 Bytes								

Fig. 15 Trama de ethernet. Fuente: [21]

3.3.2 BLE (Bluetooth Low Energy)

Bluetooth es el nombre de las redes inalámbricas de área personal que se emiten por radiofrecuencia en la banda de los 2.4 GHz [22], se crea para intentar eliminar los cables con equipos móviles y poder crear redes inalámbricas.

BLE (Bluetooth Low Energy), es una versión de bluetooth creada para proveer de conexión a distintos aparatos sin gastar mucha energía. Esto se hace minimizando la potencia de la señal y el radio de acción del bluetooth normal. El BLE se idea para que los entornos de IoT que puedan tener sensores encendidos durante meses sin tener que cambiar la batería. Como se ha comentado previamente, la Raspberry usa este tipo de conexiones de manera predeterminada a partir de la versión 3.

3.3.3 MQTT

Es un protocolo de red desarrollado basándose en TCP/IP para las comunicaciones de dispositivos IoT [23]. Se crea en 1999 para la extracción de gas y de petróleo donde había miles de sensores que necesitaban mandar datos a lugares lejanos a través de un satélite usando el menor ancho de banda posible ya que esta transmisión era bastante cara.

Está basado en una topología publish/subscribe (figura 16) donde existen dos tipos de sistemas: clientes y brokers. El bróker es el que recibe las comunicaciones de diferentes clientes y luego las envía a otros clientes. Los clientes son los que se subscriben a la información o la publican.

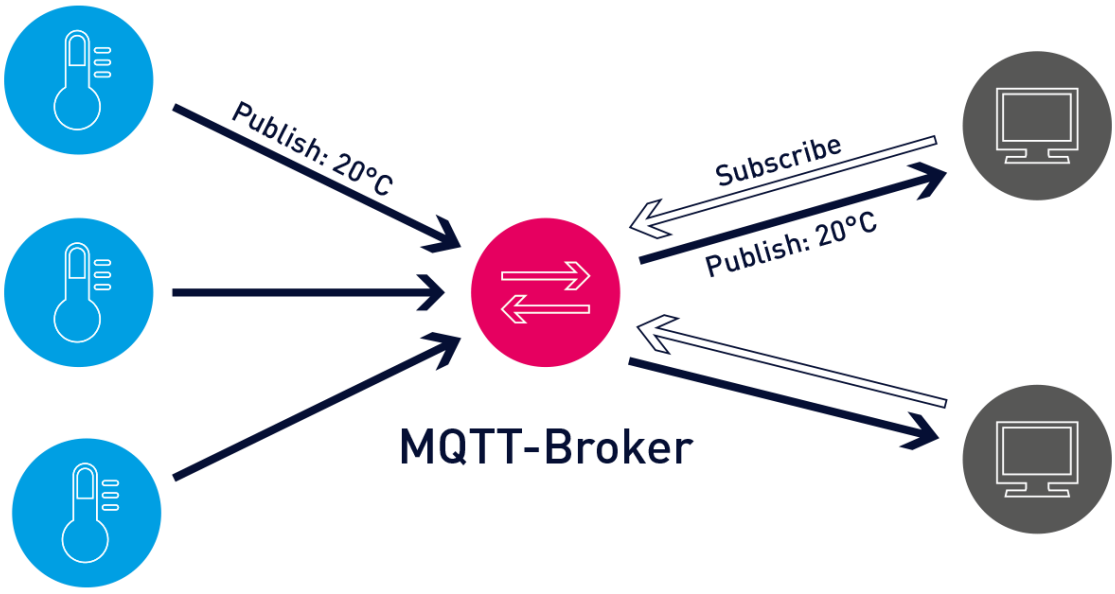

Fig. 16 Topología de un sistema MQTT. Fuente: [23].

3.3.4 Zigbee

Son otras redes inalámbricas muy parecidas a bluetooth, pero con mucho menor consumo y con menor velocidad (2MB en bluetooth y 250 Kbps en Zigbee) [24] [25], siendo estas más útiles para controlar los dispositivos IoT que para transmitir datos. Además, puede tener muchos más nodos que Bluetooth en la misma subred. El sistema Trädfri de IKEA basa sus conexiones en este protocolo.

Hay diferentes tipos de dispositivos Zigbee:

Según su papel en la red, existen 3 tipos de dispositivos:

a. Coordinador Zigbee: solo debe existir uno y se encarga de controlar la red y los caminos que siguen los dispositivos.
b. Router zigbee: conecta dispositivos entre sí y puede interactuar con el usuario.
c. Dispositivo final Zigbee: solo manda información al coordinador/router, pero no puede enviar información a otros dispositivos, para gastar la menor batería posible.

Según su funcionalidad pueden ser:

a. Funcionalidad completa (nodos activos): son los que pueden ser routers o coordinadores ya que tienen memoria y capacidad de computar mensajes recibidos.
b. Funcionalidad reducida (nodos pasivos): son los sensores /nodos finales ya que tienen una capacidad reducida por su simplicidad.

4 Descripción alto nivel del sistema

En esta parte se explicará cómo funciona el sistema tanto a nivel de software como de hardware y otros aspectos como la seguridad o los requisitos del sistema.

La diferencia entre las arquitecturas de los dos sistemas es la siguiente:

- La primera, está basada enteramente en local sin necesidad de conexión a internet, se guardan los datos con mongodb en la propia Raspberry y muestra los datos con el propio node-red.
- La segunda, guarda los datos en la base de datos de Watson IBM y son accesibles sólo accediendo a la cuenta de IBM por internet. Sin embargo, el control de la bombilla es local, así que sólo sirve para guardar y ver los datos, no para controlar el sistema de manera remota (aunque, como se ha dicho antes, es posible programar un sistema para hacerlo). Además, tiene un sistema de alertas en Google Nest y por mail en caso de que la temperatura sea muy alta/baja. Por último, cuenta con un sistema basado en el reconocimiento de voz de Google Nest que permite cambiar el estado de la bombilla y obtener las medidas del sensor.

4.1 Estructura del sistema basado en conexión local

Como se ve en la figura 17, el sistema basado en conexión local tiene unos componentes hardware indicados en negro que se conectan entre ellos a través de diferentes tipos de conexión, la bombilla al Gateway de Ikea mediante Zigbee, este (que manda los mensajes con el protocolo COAP) y el Google Nest al router para poder conectarse a la Raspberry, que está conectada al SensorTag por BLE. Los protocolos de alto nivel se indicarán en verde.

La decisión de usar el router es por facilidad a la hora de trabajar con la Raspberry desde un ordenador externo, porque el Gateway de IKEA necesita ethernet, por lo que mínimo sería necesario un switch y por evitar la configuración de redes manualmente.

Podría haberse construido el sistema sin necesidad del router, incluso sin usar un switch, usando un módulo que dé conexión Zigbee a la Raspberry y por tanto prescindir del Gateway de IKEA, que lo que hace es traducir los mensajes por COAP a Zigbee. Pero

se decidió usar el Gateway por facilidad de configuración con la bombilla y disponibilidad.

Respecto al software, están indicados en rojo los dos programas que usaremos para almacenar y visualizar la información, así como controlar la bombilla.

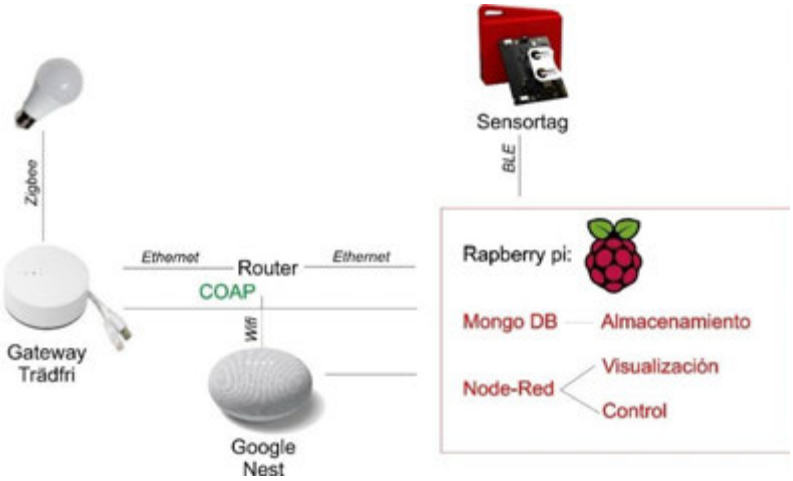

Fig. 17 Estructura del sistema local.

4.2 Estructura del sistema basado en conexión a la nube

El sistema en la nube (figura 18), es semejante que el local sólo que los datos obtenidos, se mandan a la plataforma en internet de Watson IBM mediante MQTT. Los datos enviados por MQTT se mandarían desde el router, ya que es el que tiene conexión a internet, pero para la claridad del dibujo, los datos de la Raspberry van a Watson IBM donde podemos visualizarlos y almacenarlos.

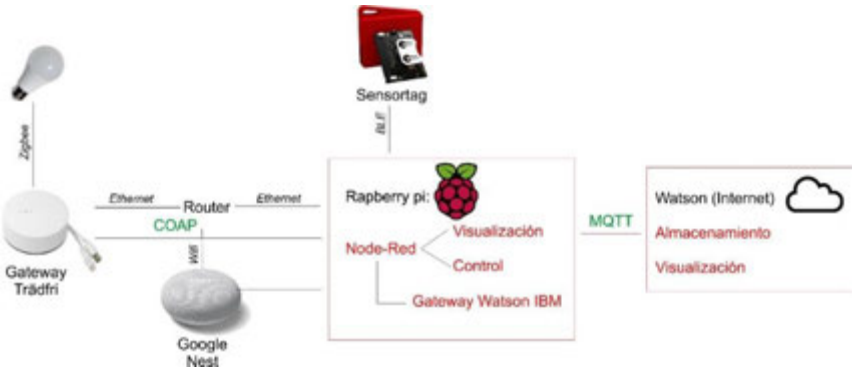

Fig. 18 Estructura del sistema basado en la nube

4.3 Requisitos

- Funcionales
    - Requisito Funcional 1

Nombre	Uso del sensor CC2650
Prioridad	Alta
Importancia	Alta
Descripción	El sensor CC2650 es necesario para detectar la temperatura, presión o luz

Tabla 1. *Requisito Funcional 1.*

    - Requisito Funcional 2

Nombre	Uso de Raspberry Pi
Prioridad	Alta
Importancia	Alta
Descripción	La Raspberry pi se encargará de albergar todo el software, de conectarse a las otras partes de hardware y de hacer de Gateway con la base de datos en la nube

Tabla 2. Requisito Funcional 2.

    - Requisito Funcional 3

Nombre	Uso del node-red
Prioridad	Alta
Importancia	Alta
Descripción	Node-red se encargará de integrar las diferentes partes de software para que funcionen juntas, desde detectar el sensor a sacar los datos en el dashboard y guardarlo en una bdd

Tabla 3. Requisito Funcional 3.

    - Requisito Funcional 4

Nombre	Uso de mongodb
Prioridad	Alta
Importancia	Alta
Descripción	Para guardar los datos de manera local se usará la base de datos de mongod

Tabla 4. Requisito Funcional 4.

    - Requisito Funcional 5

Nombre	Uso de Watson IBM
Prioridad	Alta
Importancia	Alta
Descripción	Para guardar los datos en la nube se usa la base de datos de Watson, una plataforma de IBM

Tabla 5. Requisito Funcional 5

- Requisito Funcional 6

Nombre	Uso del GW de Ikea
Prioridad	Alta
Importancia	Media
Descripción	El Gateway de trädfri servirá para encender y apagar la bombilla según haya mucha o poca luz.

Tabla 6. Requisito Funcional 6.

- Requisito Funcional 7

Nombre	Uso de Google Nest
Prioridad	Alta
Importancia	Opcional
Descripción	Google Nest se encargará de notificar de las diferentes alertas mediante su altavoz.

Tabla 7. Requisito Funcional 7.

- Requisito Funcional 8

Nombre	Uso de Gmail
Prioridad	Alta
Importancia	Opcional
Descripción	Gmail se encargará de notificar de las diferentes alertas mediante correo electrónico.

Tabla 8. Requisito Funcional 8.

- Requisito Funcional 9

Nombre	Uso de node-dashboard
Prioridad	Alta
Importancia	Alta
Descripción	El dashboard se encargará de mostrar los datos y de tener un control manual para la bombilla.

Tabla 9. Requisito Funcional 9.

- No Funcionales
  - Requisito No Funcional 1

Nombre	Tiempo de conexión del sensor
Prioridad	Alta
Importancia	Alta
Descripción	El sensor deberá conectarse a la Raspberry pi en menos de un minuto.

Tabla 10. Requisito No Funcional 1.

  - Requisito No Funcional 2

Nombre	Tiempo de respuesta de node-red
Prioridad	Alta
Importancia	Alta
Descripción	Una vez conectado, node-red debe empezar a mostrar los datos en menos de 30 s

Tabla 11. Requisito No Funcional 2.

- Requisito No Funcional 3

Nombre	Tiempo de respuesta del Gateway de Ikea
Prioridad	Alta
Importancia	Alta
Descripción	Si la temperatura se muestra superior al límite, debe encender la bombilla en menos de 10 s.

Tabla 12. Requisito No Funcional 3.

- Requisito No Funcional 4

Nombre	Conexión por BLE
Prioridad	Alta
Importancia	Alta
Descripción	La conexión entre el sensor y la Raspberry deberá hacerse por BLE

Tabla 13. Requisito No Funcional 4.

- Requisito No Funcional 5

Nombre	Conexión por Ethernet
Prioridad	Alta
Importancia	Alta
Descripción	La conexión entre la rasberry pi y el Gateway de Ikea deberá ser por Ethernet.

Tabla 14. Requisito No Funcional 5.

5 Instalación y configuración de la solución

5.1 Preparación de la Raspberry Pi

Para empezar a usar el sistema [26], se debe configurar la Raspberry con todos los programas y ajustes necesarios. Primero, es necesario instalar el Raspberry OS, se descarga de su página web [27] desde un ordenador que cuente con lector de tarjetas SD y lo instalamos en una tarjeta con el programa balenaEtcher [28]. Una vez está el SO montado, se inserta la tarjeta en su lugar de la Raspberry y se inicia.

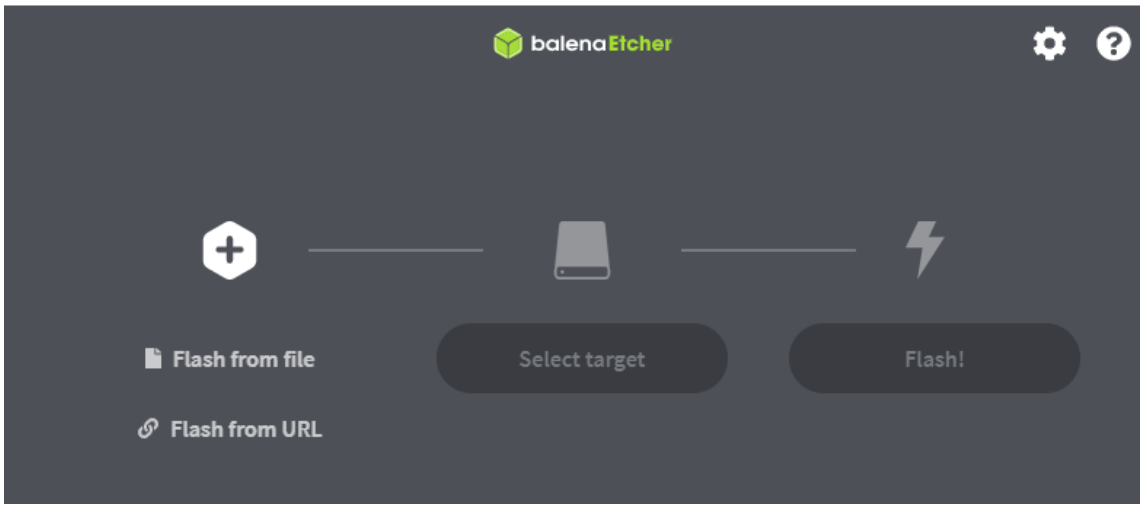

Fig. 19 Funcionamiento de balenaEtcher

Dentro de Raspberry OS es necesario activar VNC para poder controlarla de manera remota (antes tiene que estar conectada a la misma red que el otro ordenador), esto se hace abriendo una terminal, corriendo

sudo raspi-config

a continuación, se accede a interfacing options y se activa VNC. Hecho esto, se puede controlar la Raspberry desde cualquier ordenador que tenga VNC instalado insertando el usuario y contraseña correspondiente.

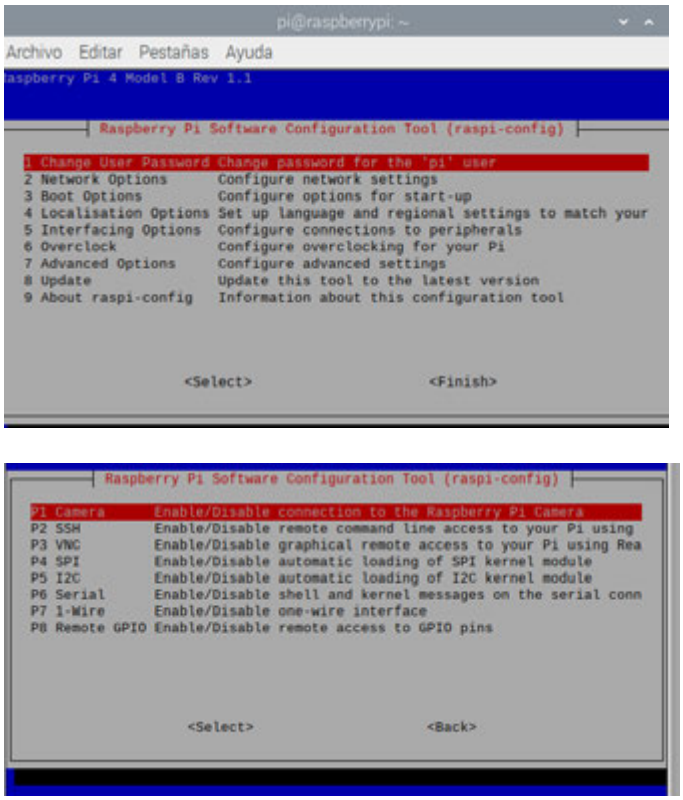

Fig. 20 Activación de VNC en la Raspberry

Fig. 21 Vista de VNC en Windows

Por último, hay que instalar unos programas que se encargan de buscar al sensor mediante BLE. Esto se hace con

sudo apt-get -y install bluez build-essential libglib2.0-dev libdbus-1-dev python-dev

sudo pip install bluepy == 1.2.0

Al último comando se debe añadir la versión porque las siguientes no funcionan.

5.2 Instalación de mongodb

Para guardar los datos, se usa mongodb, se instala usando

sudo apt-get install mongodb

una vez instalado, se inicia con el comando *sudo mongod*, el sistema inicia la base de datos y proporciona una dirección ip para hacer solicitudes y guardar los datos. Para probar que funciona usamos *sudo mongo.* Como se puede ver en las figuras, el primer comando se queda esperando a una conexión a un puerto predeterminado y el segundo intenta hacer una conexión a este puerto (si no se le indica otro), si lo consigue abre una terminal para poder dar órdenes a la base de datos.

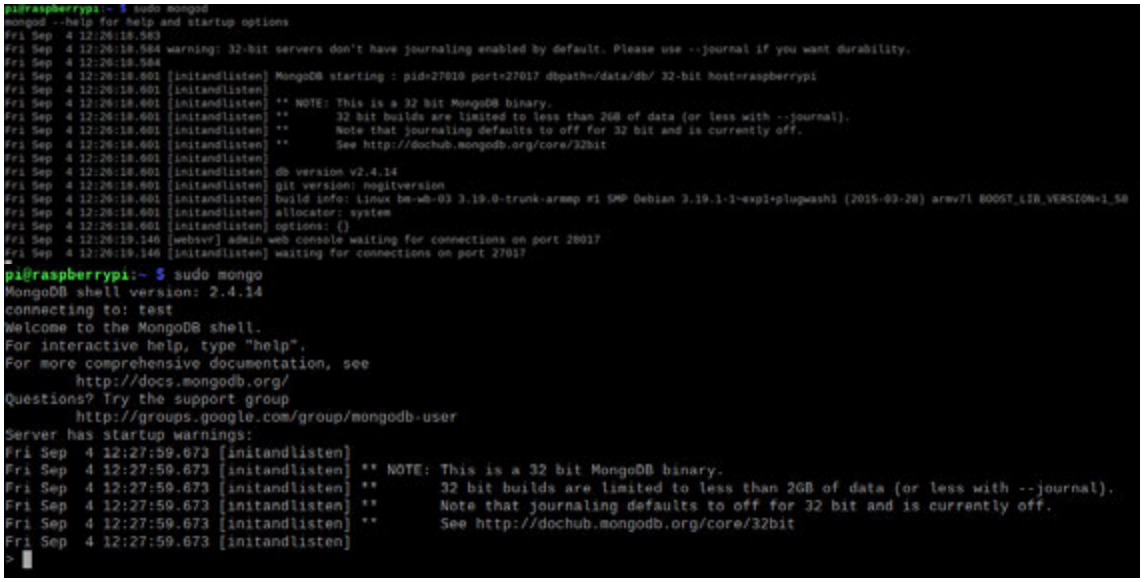

Fig. 22 Inicio de mongodb

5.3 Conexión de diferentes partes hardware

La conexión de las diferentes partes se puede ver en el capítulo anterior, pero las conexiones en concreto son las siguientes: el Gateway de Ikea viene conectado mediante zigbee de manera predeterminada a la bombilla, pero hay que conectarlo mediante un cable ethernet al router para que la Raspberry pueda usarlo con node-red.

La Raspberry se conecta al router también mediante ethernet para tener acceso a internet y acceso a toda la red. Esta se conecta al sensor mediante BLE con el script de Python.

Por último, el Google Nest se conecta mediante wifi al router, para ello es necesaria la aplicación de Google Home en un móvil, una vez esta detecte que hay un Google Nest cerca, solicitará usar el wifi del propio móvil para conectar el Nest a la red que debe usar y poder utilizarse mediante node-red.

5.4 Configuración de sistema de reconocimiento de voz

Para que el sistema de reconocimiento de voz funcione correctamente son necesarios dos programas a mayores del propio software de Google Nest y Node red: IFTTT [20] y Webhookrelay [29].

Primero hay que configurar Webhookrelay [30], para ello debemos crearnos una cuenta en su web. Dentro de esta, creamos un nuevo *bucket* (figura 23) (un sistema que, si recibe un dato por una URL, lo devuelve por otra), este nos da una URL que será usado en IFTTT y un token y una key para usar en node-red, se insertan en su nodo correspondiente pasando al siguiente paso.

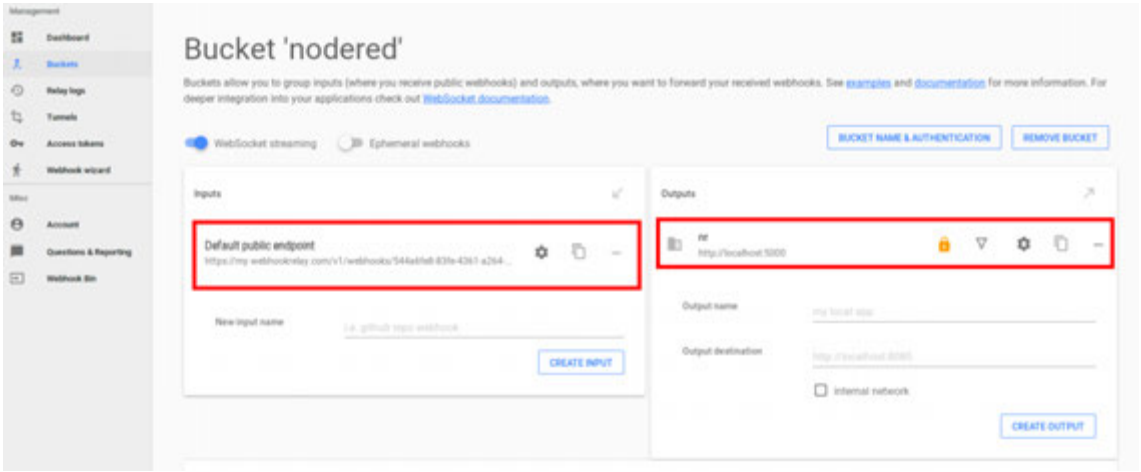

Fig. 23 Valores de un bucket en Webhook relay

Después, hay que hacer un registro en IFTTT y crear una nueva orden (figura 24), que recibe la señal del Google Nest a través de Google Assistant (el software encargado de atender a peticiones) y manda una solicitud a la URL que se ha obtenido de Webhookrelay, que se encargará entonces de hacer POST (publicar) en el flujo de node-red.

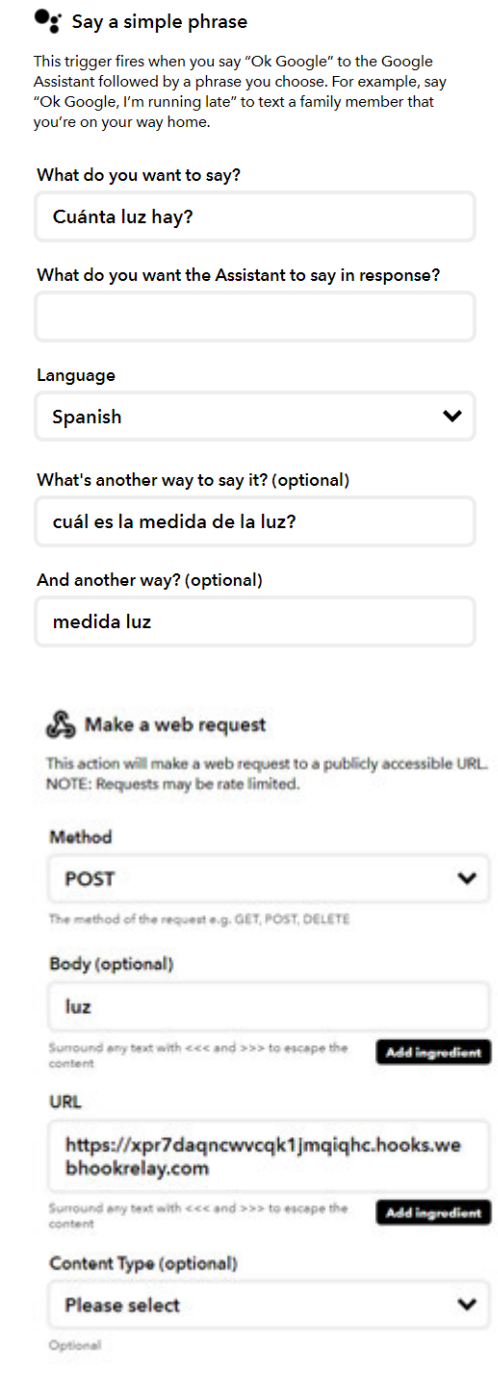

Fig. 24 Configuración de órdenes en IFTTT

5.5 Configuración de Node-Red

Node-red viene preinstalado en Raspberry OS, así que no es necesario instalarlo, pero si es necesario comprobar que está actualizado usando:

sudo apt-get upgrade

sudo apt-get -y install -y npm –upgrade

npm i -g npm@2.x

Una vez hecho esto, se inicia el servidor de node-red (es necesario hacerlo desde la Raspberry, pero una vez hecho, se puede acceder desde cualquier ordenador de la red usando la ip de la Raspberry junto al puerto 1880) y es necesario instalar los nodos para tratar la información que llega de los sensores y mostrarla. Se hace desde el menú de node-red Manage Palette, ahí se puede quitar o añadir nodos. Los nodos usados han sido los siguientes:

node-red-contrib-ibm-watson-iot - se encarga de poder conectar el sistema en la nube con Watson IBM.

node-red-node-daemon – permite correr el script de Python [26] que conectará al sensor con el sistema de node-red.

node-red-contrib-mongodb2 – Se encarga de los nodos de entrada/salida de la base de datos de mongodb.

node-red-contrib-tradfri – Se encarga de controlar los dispositivos tradfri de IKEA, en nuestro caso las bombillas

node-red-node-email – Sirve para mandar mensajes por correo.

node-red-dashboard – Añade el dashboard de node-red para sacar datos y tener botones de control.

node-red-contrib-cast – Permite mandar notificaciones al Google nest mini.

node-red-contrib-ibm-watson-iot – Sirve para mandar datos a la plataforma Watson IBM.

Una vez han sido instalados todos los nodos, se conectan mediante la interfaz gráfica de node-red de la siguiente manera:

Primero (figura 25) se conecta el nodo Daemon que contiene el script de Python [26] para conectar el SensorTag a la Raspberry mediante BLE. Después, el nodo Split se encarga de separar los mensajes que llegan para que cada línea sea un mensaje diferente, en vez de mandar varias líneas seguidas. El último nodo, convierte el string que le llega en un objeto para poder usar sus diferentes variables y no sólo un string largo.

Fig. 25 Primera parte del sistema en node-red

El segundo paso consiste en sacar las diferentes medidas en 4 diferentes ramas para tratarlas de manera diferente.

La primera es la presión (figura 26), es la más simple de las 4, sólo muestra el valor de presión actual y un gráfico de los últimos valores, asimismo guarda los valores en la base de datos de mongodb.

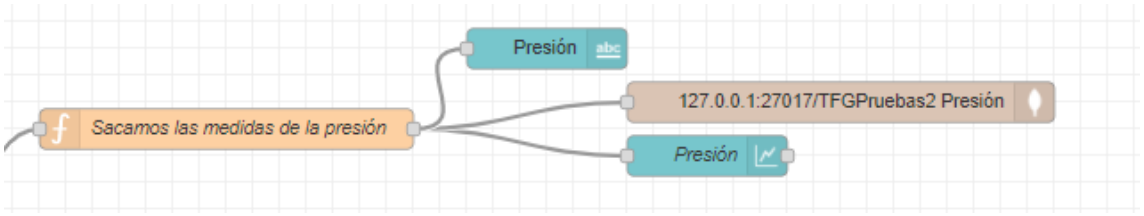

Fig. 26 Extracción de medidas de presión

La segunda (figura 27) es la temperatura, a mayores de hacer lo que hace el sistema de presión, tiene un sistema de alertas que, si la temperatura es demasiado alta, hace saltar una notificación por mail (requiere una cuenta de Google a la que se ha desactivado la comprobación de seguridad para poder mandar mails) y Google Nest cada diez minutos da la alarma por el altavoz. Además, cada hora informa de la temperatura actual en el Google Nest.

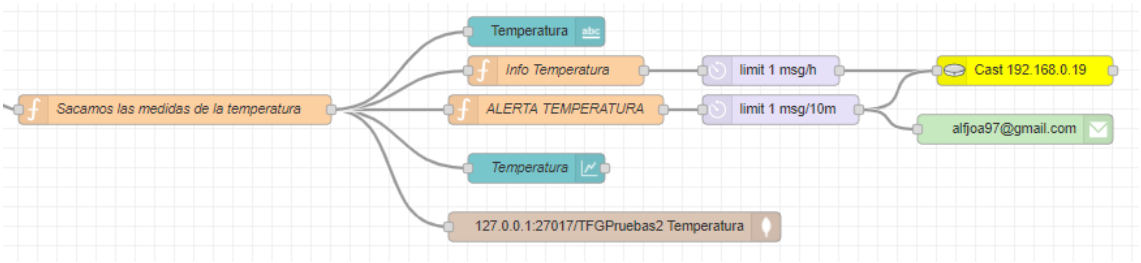

Fig. 27 Extracción y alertas de medidas de temperatura.

La tercera (figura 28) es la luz, hace lo mismo que la presión, pero tiene un sistema que enciende y apaga la bombilla según la luz que haya (se usa el nodo de trädfri, que tiene la MAC, la ip y el código de seguridad del Gateway), si hay mucha luz se apaga y al revés si hay poca. Tiene un limitador para que no entre en bucle, cada 10 minutos puede cambiar el estado de la bombilla, para que no se encienda/apague continuamente si hubiera algún fallo. Cada vez que lo enciende o apaga, manda un mensaje con el Google Nest (el nodo tiene la ip del Google nest) informando de que se ha apagado o encendido.

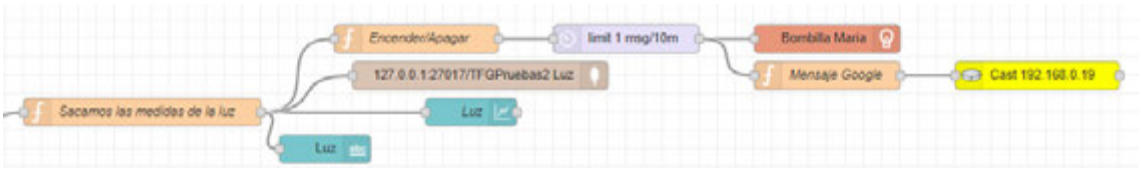

Fig. 28 Extracción de medidas de luz y funcionamiento de la bombilla.

La última, mide la aceleración y la guarda. Además, manda un mensaje al Google Nest cada 10 minutos si se está moviendo el sensor.

Fig. 29 Extracción de medidas de aceleración y alerta de movimiento.

Esta parte (figura 30) crea el sistema de control manual, proporciona un botón para encender la bombilla y una slider para poder modificar el brillo de la bombilla. Si se enciende o apaga la bombilla, el Google Nest manda un aviso.

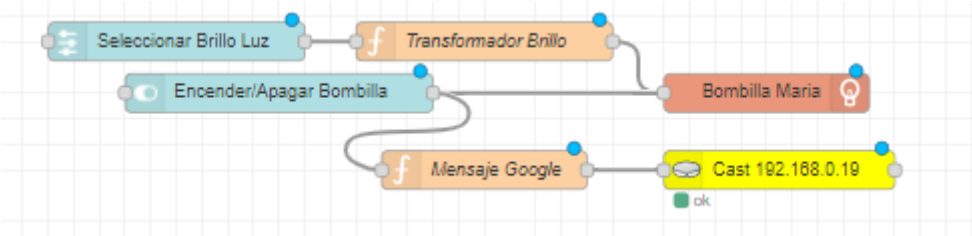

Fig. 30 Control manual de la bombilla.

La figura 31 muestra cómo es el dashboard de node-red. A la izquierda están los controles manuales de la bombilla, pudiendo modificar el brillo de la luz y apagarla o encenderla. En el centro se encuentra el histórico de las diferentes medidas y a su derecha el valor actual de estas medidas.

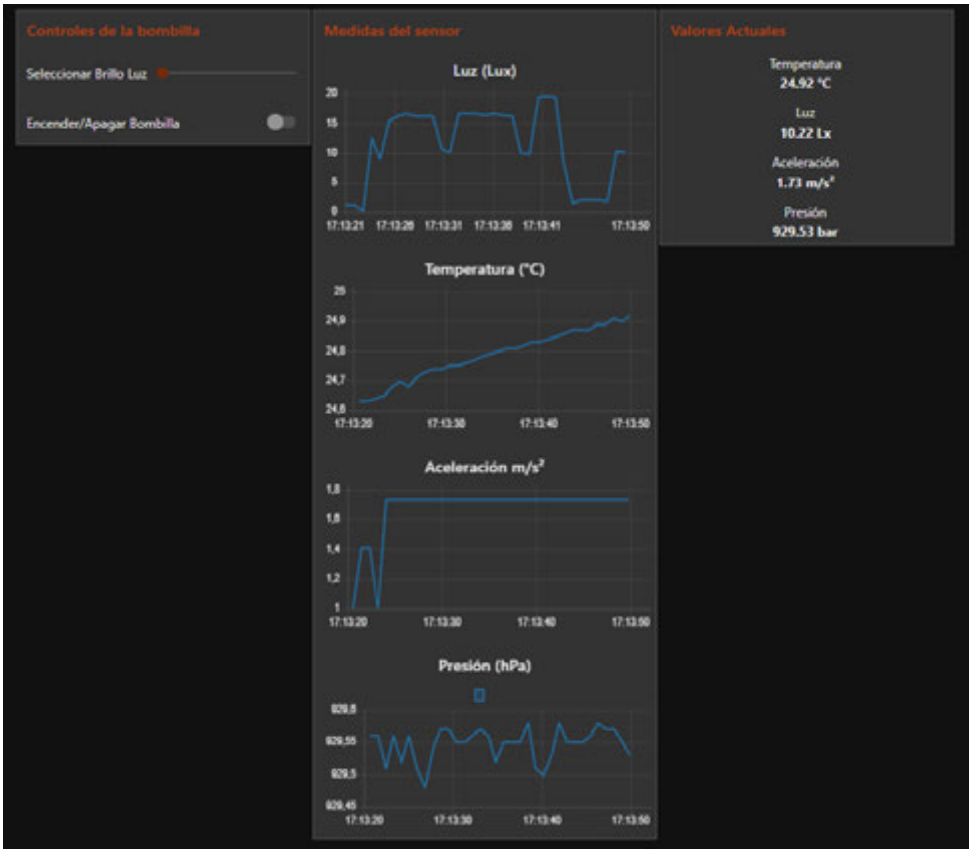

Fig. 31 Dashboard de node-red.

Para el reconocimiento de voz, el sistema funciona como se ha explicado antes, una vez llega la petición a Webhookrelay, este hace un POST de lo que anteriormente se haya configurado.

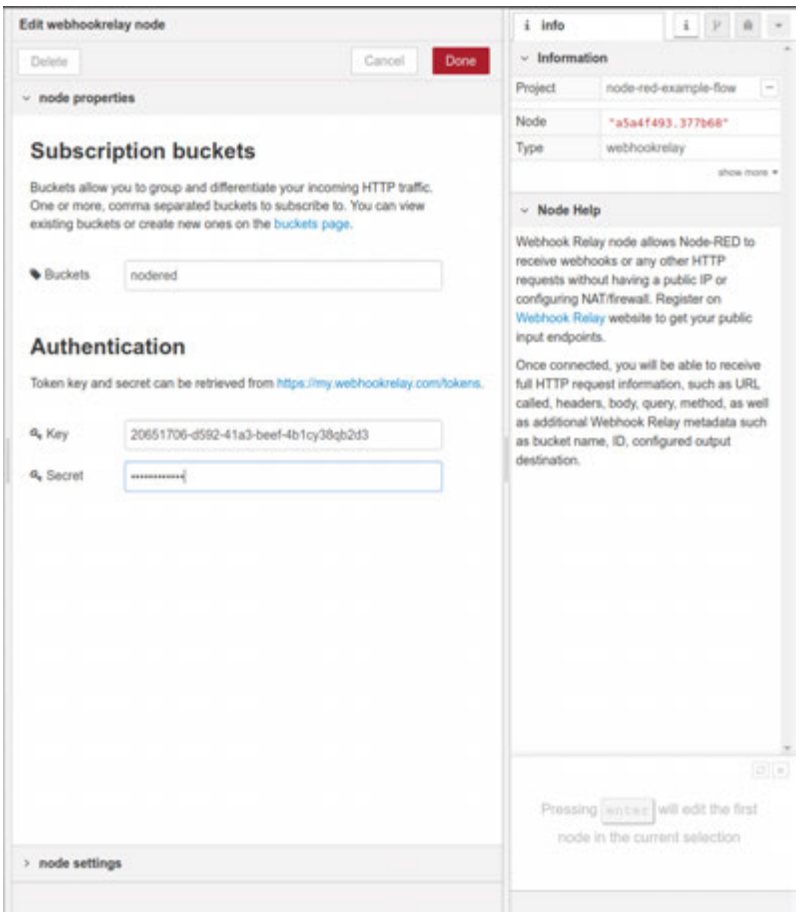

Fig. 32 Configuración de nodo de Webhookrelay

Se recoge este dato, que viene en el body(cuerpo) del mensaje y según el contenido, se usa para encender/apagar la bombilla o emitir el valor que se ha pedido por voz (que vienen del esquema correspondiente mediante un nodo que les permite cruzar flujos). La función reconocimiento voz lo único que hace es extraer el valor correspondiente para emitirlo, pues la orden y el valor tienen que venir en un mismo vector.

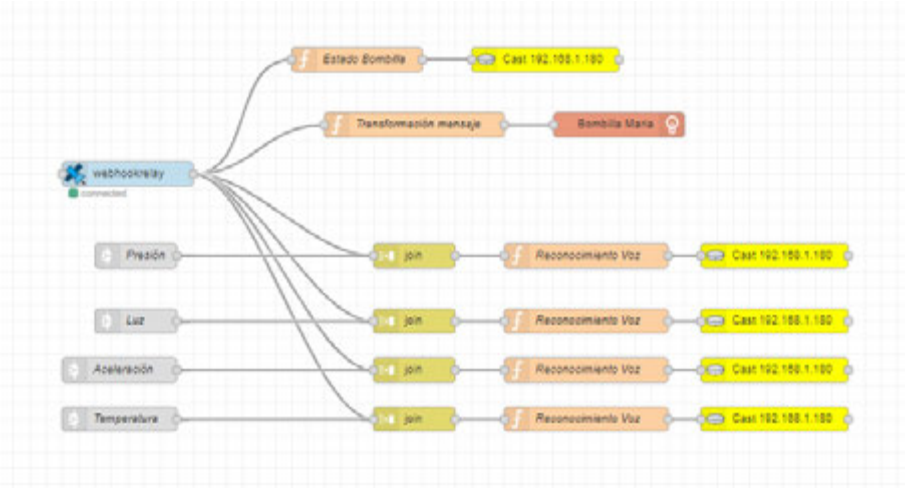

Fig. 33 Nodos de reconocimiento de voz

Por último (figura 34), esta rama guarda los datos del sensor en Watson IBM (es necesario anteriormente hacerse una cuenta en Watson IBM, crear un Gateway y un dispositivo y pasar los tokens al nodo [31]) para posteriormente ser mostrados a través del dashboard del propio Watson IBM [32] (figura 27). La estructura es parecida a la del dashboard de node-red, a la izquierda el histórico de medidas y a la derecha la medida actual.

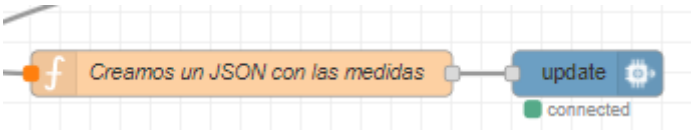

Fig. 34 Guardado de los datos en Watson.

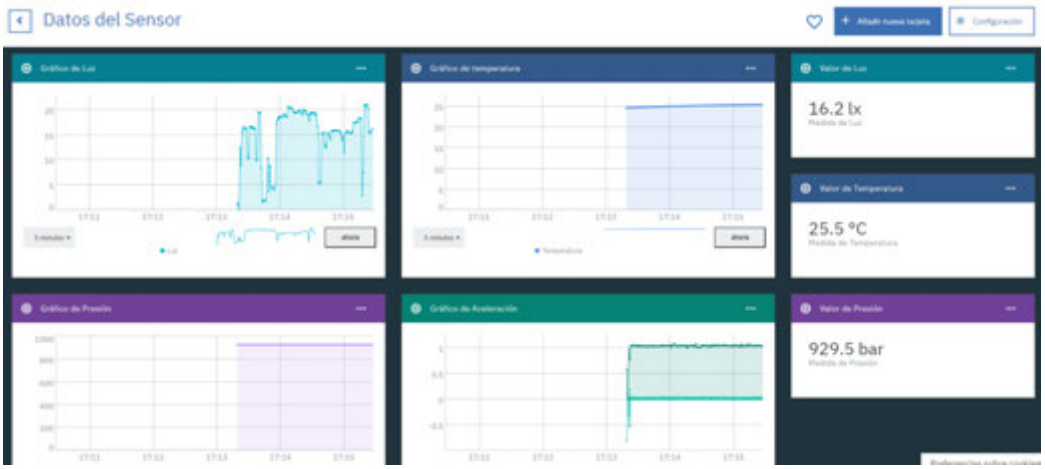

Fig. 35 Dashboard en Watson IBM.

6 Pruebas y comparación de las dos Arquitecturas

En esta parte del trabajo, se mostrarán las diferentes pruebas hechas al sistema y compararemos las dos arquitecturas que existen.

6.1 Pruebas

La primera prueba que realizar es comprobar que la Raspberry detecta y se conecta al sensor, para eso se ejecuta el script de Python con el siguiente comando

sudo Python -u /home/pi/sensortag.py

y se comprueba si el sensor devuelve medidas a través de la línea de comandos como podemos ver en la figura 36

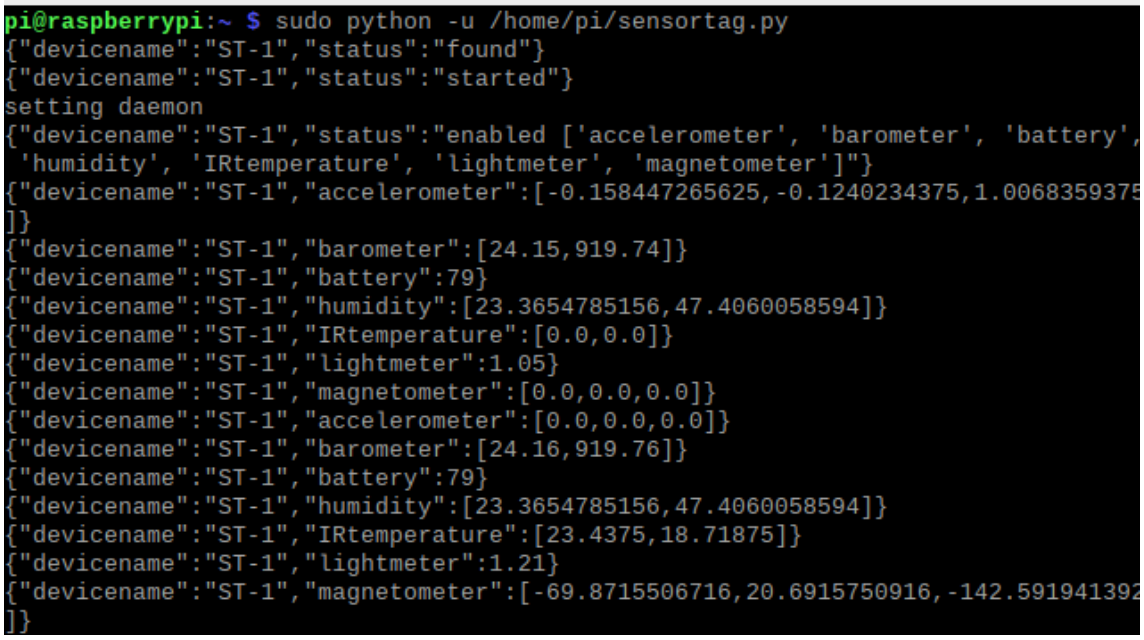

Fig. 36 Prueba de funcionamiento del sensor.

Lo siguiente es comprobar el funcionamiento de node-red, para ello se añade el comando anteriormente probado a un nodo y se comprueba mediante un nodo de debug si los datos llegan al flujo, se ve en la figura 37.

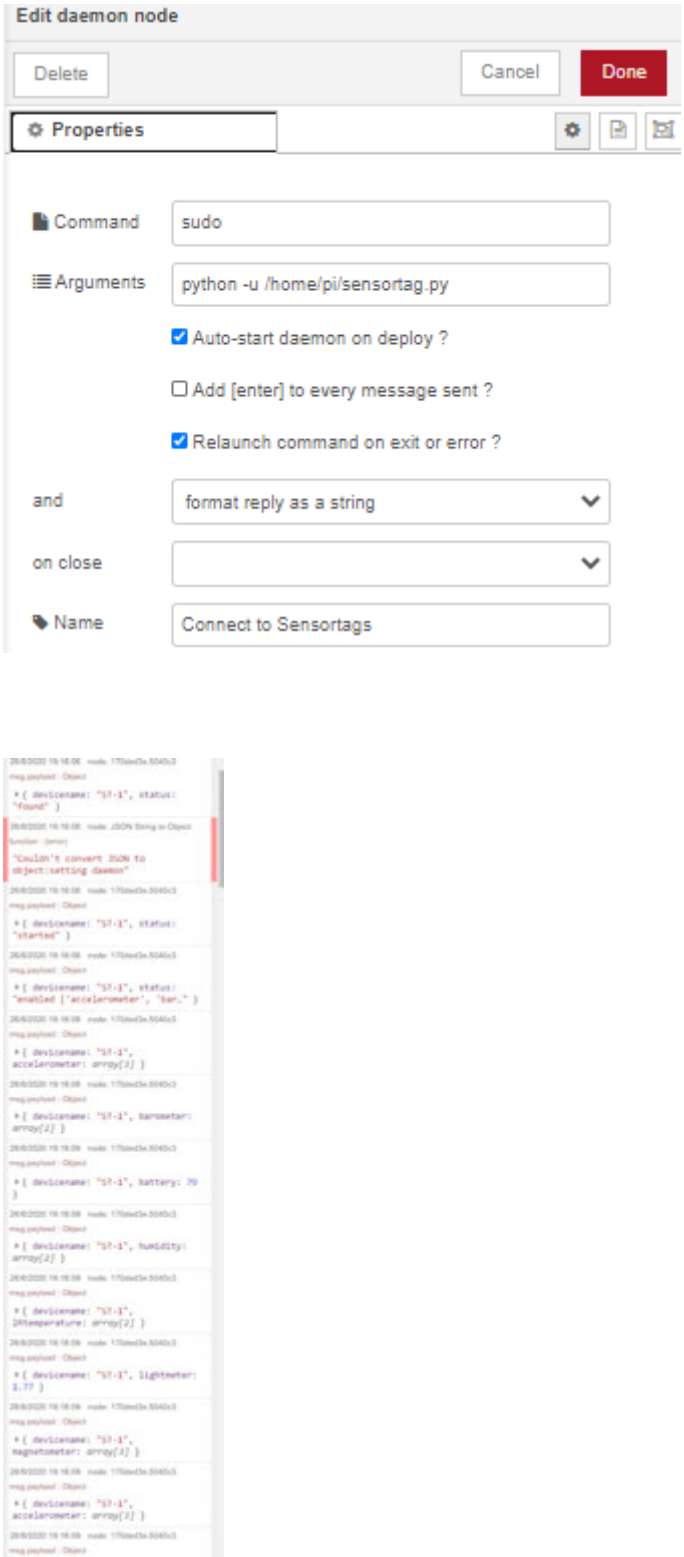

Fig. 37 Prueba de recogida de datos.

Si los datos llegan, se van añadiendo diferentes nodos y se comprueba su funcionamiento.

Se añade el nodo de Trädfri y se comprueba si se enciende o apaga la bombilla usando un nodo de inyección con los strings “on” y “off”. Luego se comprueba si cambia el brillo inyectando diferentes valores para este (figura 38).

Fig. 38 Prueba de nodo Trädfri.

Luego se prueba el dashboard, añadiendo los nodos de dashboard y mandando los datos individuales, si en el dashboard empieza a crear gráficas, es que funciona. Un ejemplo de estas gráficas está en la figura 31.

Después, se hace lo mismo con Watson IBM, se puede ver que funciona bien en la figura 35.

Para comprobar el funcionamiento del Google Nest es muy sencillo, se inyectan datos a los nodos y si los emiten por el altavoz, funcionan. Lo mismo para la detección de voz, una vez conectados se comprueba que al recibir una orden emite la acción correspondiente.

Por último, se comprueba el funcionamiento de mongodb añadiendo sus nodos y luego usando el comando:

mongo

Para acceder al control de la base de datos y comprobar que los datos están en su base de datos correspondiente, para ello usamos:

use (la base de datos que se quiere usar)

db.(colección).find()

Para mostrar todos los valores de la base de datos y comprobar que se van añadiendo, al estar en línea de comandos no son muy legibles, pero se puede comprobar que se van añadiendo (figura 39).

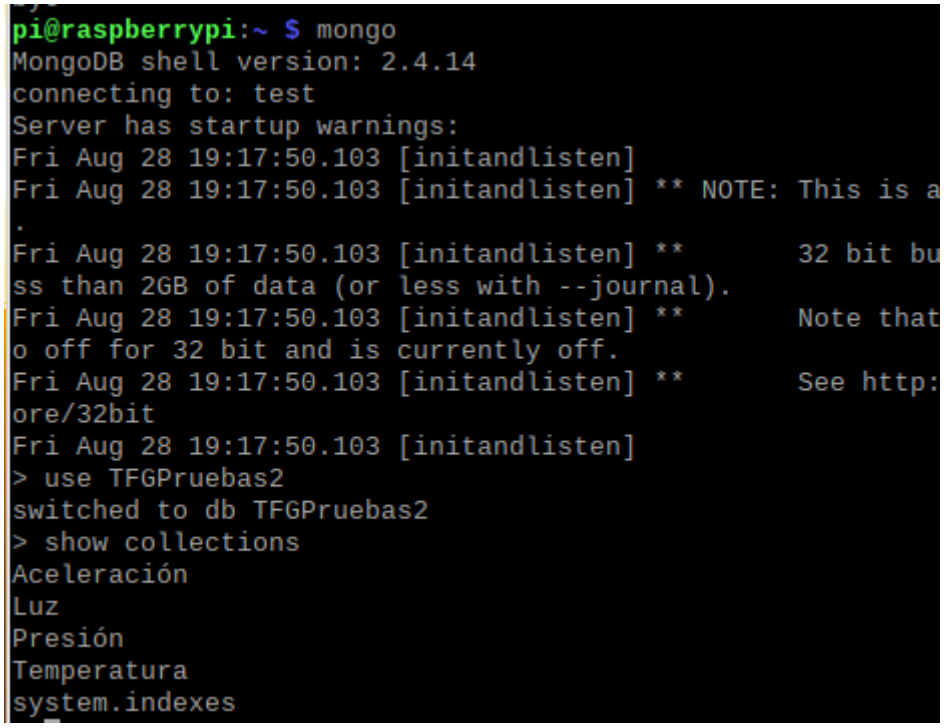

Fig. 39 Pruebas en Mongodb.

6.2 Seguridad

En este apartado se explicará la seguridad de distintos métodos de comunicación:

COAP – Tiene 4 modos de seguridad [33]:

1. **Nosec** – Manda los mensajes sin DTLS
2. **PreSharedKey** – Manda los mensajes con DTLS, tiene una lista de claves compartidas y cada una puede comunicarse con unos nodos muy concretos.
3. **RawPublicKey** – Manda los mensajes con DTLS y usa una clave asimétrica sin certificado.
4. **Certificate** – Usa DTLS y requiere un certificado X.509 para validar el mensaje

Es vulnerable a los ataques de DDOS (Denegación de servicio, hacer muchas peticiones al mismo tiempo para intentar saturar el sistema) pero en este caso es bastante improbable que ocurra uno

DTLS – Está basado en TLS y usa UDP [34], por lo que puede que los paquetes no estén en orden y haya que reorganizarlos cuando llegue al destino. En nuestro caso lo usa el gateway de Ikea para comunicarse con los dispositivos de entrada y salida. Al conectarse con un dispositivo nuevo, hace un "handshake" (manda un saludo al GW, el GW se lo devuelve al cliente y el cliente lo confirma) para confirmar que es la persona correcta la que pide acceso y si es así, le deja interactuar con las bombillas (figura 32).

Parece ser que había una vulnerabilidad de MITM (Man In The Middle, interceptar el mensaje entre el cliente y el GW haciéndose pasar por uno de ellos o teniendo acceso al canal de envío) pero en la siguiente versión se solucionó insertando más comprobaciones de usuario.

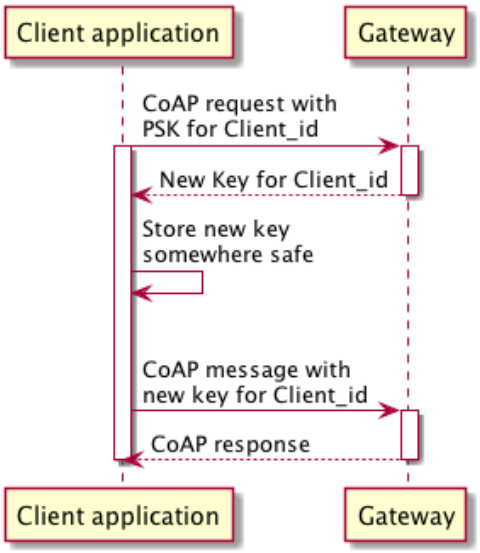

Fig. 40 Handshake de DTLS. Fuente: [34].

BLE – Es de las partes más vulnerables, si hacen un ataque MITM podrían hacerse pasar por la Raspberry y el sensor quedaría inutilizado. Puesto que BLE no tiene autentificación de usuarios, cualquiera que intente conectarse justo cuando el sensor manda el mensaje de conexión, podría hacerlo. La única solución es que al conectar el sensor comprobar que todo está en orden y funciona.

6.3 Escalabilidad

El sistema se puede escalar fácilmente, se pueden integrar varios sensores del mismo tipo fácilmente o incluso de otro tipo para tener más lecturas de otros valores.

El script usado para conectarse al CC2650 soporta la conexión de varios a la vez así que no habría problema. De cara a las salidas también es posible conectar todos los dispositivos necesarios, desde un termostato que mantenga la temperatura constante a un sistema que controle la altura de las persianas en base a la luz que detecte el dispositivo.

6.4 Versatilidad

Como se puede ver en lo comentado anteriormente, el sistema es bastante versátil, de hecho, con la estructura de node-red es posible cambiar las entradas y las salidas para prácticamente cualquier dispositivo de IoT, por ejemplo, un sistema automático de riego. En este caso de uso, el único cambio que habría que hacer es tener un sensor que detecte la humedad de la tierra, hacer prácticamente lo mismo que hace ahora node-red y la salida ser una señal de apertura del sistema de riego en vez de una bombilla encendiéndose. Lo mismo podría hacerse con cualquier estructura IoT.

6.5 Comparación de las dos arquitecturas

Para las dos soluciones desarrolladas: una basada en una arquitectura local y otra en la nube, a continuación, se comentarán diferentes características y las diferencias entre las dos arquitecturas.

Respecto a la privacidad, la opción local es más segura, pues no es necesario conectar el sistema a internet y por tanto sólo sería accesible por los usuarios de la red, a diferencia del sistema basado en la nube, cuya información podría ser interceptada de camino a la nube o incluso haber una brecha de seguridad en la plataforma de Watson IBM.

El acceso al sistema a través de internet no está implementado en ninguna de las dos arquitecturas, pero su implementación sería diferente según la arquitectura:

La opción local necesitaría de un sistema capaz de conectarse a la red local a través de internet, para ello existen alternativas gratuitas como no-ip [35] o hamachi [36], la primera basada en DNS dinámicas lee la IP de la red y da un hostname de acceso que se adapta aunque la IP cambie sus valores, por lo que sería posible acceder a la red desde internet usando únicamente el hostname que nos provee no-ip. La segunda, crea una vpn para que cualquier persona que tenga las credenciales del sistema, se pueda conectar a la red y acceder a la Raspberry y por tanto al node-red dashboard.

Respecto a la opción basada en la nube sería más fácil de implementar, Watson IBM cuenta con un sistema para mandar peticiones a través de internet a node-red, que las recogerá y transmitirá las órdenes donde sea necesario.

Por último, respecto a la persistencia de los datos, lo mejor sería una combinación de ambas arquitecturas para tener redundancia. Los datos se guardan de manera local en la Raspberry y luego se vuelven a guardar en la base de datos de Watson IBM como prevención si fallase una de las partes.

7 Planificación y presupuesto

7.1 Planificación

Este proyecto se desarrolló desde el 27 de enero hasta el 7 de septiembre, la distribución del tiempo en cada tarea es la siguiente:

Tareas	Inicio	Fin	Días
Documentación	27-ene.	25-feb	29
Diseño	06-mar	30-mar	24
Desarrollo	08-abr	28-jul	111
Pruebas	30-jul	15-ago	16
Desarrollo Memoria	16-ago	07-sep	22
Total	27-ene	07-sep	224

Tabla 15. Tiempo de desarrollo de las diferentes tareas.

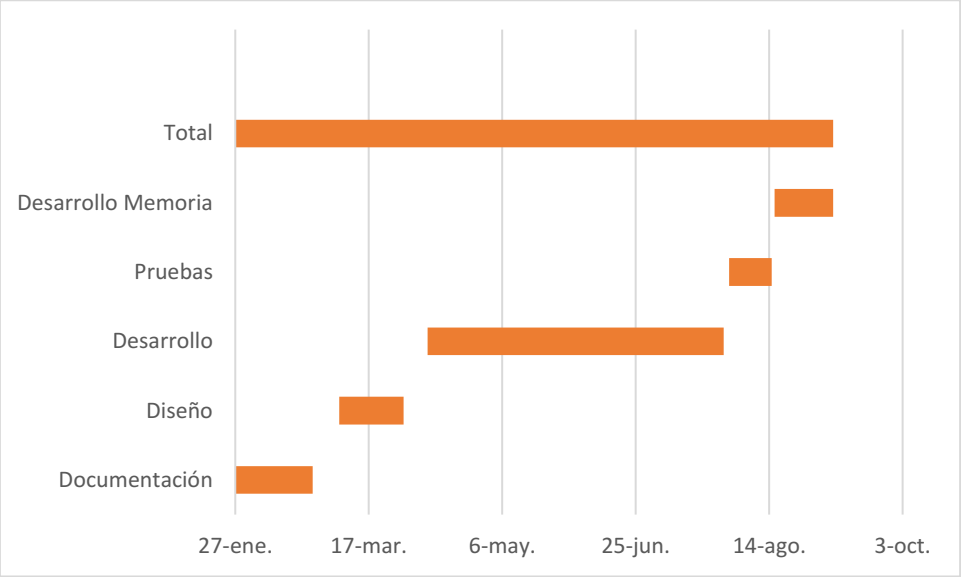

Tabla 16. Diagrama de Gantt de las diferentes tareas.

7.2 Coste laboral

Teniendo en cuenta que este proyecto se valora en 12 créditos ECTS, se puede valorar el trabajo del alumno en las horas aproximadas que representa un crédito (25 – 30 horas). El sueldo respectivo a estas horas será de unos 20 €/h, teniendo en cuenta los sueldos de un ingeniero en prácticas.

Respecto a la docente, se contará como estimación una tutoría a la semana de una hora con un sueldo de 26 €/h.

Los gastos quedarían entonces así:

Alumno		Profesora	
Créditos ECTS	12	Fecha Inicio	27/01/2020
Media horas por ECTS	27	Fecha Fin	7/09/2020
Estimación de sueldo	20 €/h	Semanas/Hora	32/1
		Estimación de sueldo	26 €/h
Total	6480 €	Total	832 €
Total		7312 €	

Tabla 17. Costes laborales.

7.3 Costes Materiales

Los costes de construir el sistema son los siguientes, incluido IVA:

Gateway Trädfri	30 €
Windows 10	Cedido por la universidad
Paquete Office	Cedido por la universidad
Bombilla Trädfri	15 €
Raspberry Pi	59,99 €
Router	67 €
SensorTag CC2650	60 €
Google Nest	59 €
Total	290,99 €

Tabla 18. Costes materiales.

7.4 Costes Totales

La suma de ambos costes da un total de 7602,99 € para construir todo el sistema. A esto habría que añadirles a los sueldos, el 21% de IVA, por lo que sale una suma de: **9138,51 €**.

8 Conclusiones y trabajos futuros

Como se puede observar, el IoT es un sector en alza que cada vez se usa de manera más habitual en domótica, por lo que el sistema desarrollado en este TFG sirve de ejemplo para ver cómo funciona la tecnología, comparar los sistemas de almacenamiento de datos para su posterior análisis, los distintos dashboards dónde mostrarlo y la integración entre diferentes sistemas domóticos.

El propósito final del trabajo es comprender el funcionamiento de las diferentes herramientas usadas:

Node-red permite una gran versatilidad, puede conectar prácticamente cualquier dispositivo IoT o función en distintos lenguajes de programación y permite mostrar los datos muy fácilmente.

Mongodb es un programa para hacer bases de datos muy simple de usar, solo indicando la base de datos y la colección ya genera una nueva.

Por último, IFTTT y Webhookrelay son una forma muy sencilla de implementar órdenes simples en dispositivos IoT, muy útil para tener frases personalizadas en dispositivos de reconocimiento de voz.

El trabajo podría ampliarse añadiendo más sensores y más actuadores para tener un sistema más complejo que controlara múltiples habitaciones. También podría disponer de un sistema de control a través de internet, que sólo con la iluminación no tiene mucho sentido, pero con la inclusión de termostatos sí sería útil.

Referencias

[1] Tech Native, «Tech Native,» 23 Abril 2019. [En línea]. Available: https://www.technative.io/iot-trends-2020-and-beyond/. [Último acceso: 13 Mayo 2020].

[2] A. Banafa, «OpenMind BBVA,» 2019 Diciembre 16. [En línea]. Available: https://www.bbvaopenmind.com/tecnologia/mundo-digital/diez-tendencias-del-internet-de-las-cosas-en-2020/. [Último acceso: 13 Mayo 2020].

[3] It Pro Team, «ItPro,» 26 Junio 2018. [En línea]. Available: https://www.itpro.co.uk/cloud-computing/28077/where-is-the-iot-being-deployed. [Último acceso: 10 Mayo 2020].

[4] Capgemini, «Capgemini,» Marzo 2018. [En línea]. Available: https://www.capgemini.com/wp-content/uploads/2018/03/dti-research-report_iot-in-operations_web.pdf.

[5] Telefónica, «Telefónica,» 2019. [En línea]. Available: https://www.ituser.es/whitepapers/content-download/a01a0957-18bf-44e8-ada8-6b02ffdeaad1/informe-things-matters-2019-cast.pdf.

[6] A. Ene, «Tech The Lead,» 19 Agosto 2020. [En línea]. Available: https://techthelead.com/recent-alexa-hack-jeopardized-echo-users-network-and-data/.

[7] Samsung, «Samsung,» 2020. [En línea]. Available: https://www.samsung.com/mx/apps/smartthings/. [Último acceso: 11 Agosto 2020].

[8] Philips , «Philips,» 2019. [En línea]. Available: https://www.philips-hue.com/en-us/p/hue-motion-sensor/046677473389. [Último acceso: 11 Agosto 2020].

[9] Perfect Home Solutions, «Perfect Home Solutions,» 2020. [En línea]. Available: http://perfecthomesolutions.net/. [Último acceso: 11 Agosto 2020].

[10] Bosch, «Bosch,» 2020. [En línea]. Available: https://www.bosch-

smarthome.com/uk/en/categories/smart-system-solutions/security/presence-simulation. [Último acceso: 11 Agosto 2020].

[11] Bosch, «Bosch,» Mayo 2019. [En línea]. Available: https://community.home-assistant.io/t/bosch-smart-home/115864. [Último acceso: 12 Agosto 2020].

[12] Fundación Raspberry Pi, «Raspberry Pi,» 24 Junio 2019. [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Último acceso: 22 Abril 2020].

[13] Texas Instruments, «Texas Instruments,» Marzo 2015. [En línea]. Available: https://www.ti.com/lit/ug/tidu862/tidu862.pdf?ts=1597946047971&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTIDC-CC2650STK-SENSORTAG. [Último acceso: 15 Agosto 2020].

[14] Domótica en Casa, «Domótica en Casa,» 2018. [En línea]. Available: https://domoticaencasa.es/ikea-tradfri/. [Último acceso: 20 Julio 2020].

[15] Node red, «Node red,» Julio 2020. [En línea]. Available: https://nodered.org/#features. [Último acceso: 12 Agosto 2020].

[16] SAP SE, «Node red,» 2019. [En línea]. Available: https://flows.nodered.org/node/node-red-contrib-saprfc. [Último acceso: 10 Agosto 2020].

[17] Node Red, «Node Red,» 26 Julio 2016. [En línea]. Available: https://flows.nodered.org/node/node-red-dashboard. [Último acceso: 23 Abril 2020].

[18] Mongodb, «Mongodb,» [En línea]. Available: https://www.mongodb.com/es.

[19] Wikipedia, «Wikipedia,» Agosto 2020. [En línea]. Available: https://en.wikipedia.org/wiki/Python_(programming_language). [Último acceso: Agosto 2020].

[20] IFTTT, «IFTTT,» [En línea]. Available: https://ifttt.com/.

[21] Wikipedia, «Wikipedia,» Agosto 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Ethernet#:~:text=Ethernet%20(pronunciado%20

%2F%CB%88i%CB%90%CE%B8%C9%99rn%C9%9Bt%2F%20en,(ether%2C%20en%20ingl%C3%A9s).. [Último acceso: 13 Agosto 2020].

[22] Wikipedia, «Wikipedia,» Agosto 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth. [Último acceso: 13 Agosto 2020].

[23] Paessler, «Paessler,» 2019. [En línea]. Available: https://www.paessler.com/it-explained/mqtt#:~:text=MQTT%20stands%20for%20Message%20Queuing,networks%20with%20limited%20bandwidth%20resources.. [Último acceso: 2020 Julio 2020].

[24] Wikipedia, «Wikipedia,» 20 Julio 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Zigbee. [Último acceso: 8 Agosto 2020].

[25] Y. FM, «Xataka,» 9 Enero 2019. [En línea]. Available: https://www.xataka.com/basics/conectividad-zigbee-amazon-echo-plus-que-como-funciona-otros-dispositivos-compatibles. [Último acceso: 8 Agosto 2020].

[26] I. Barnard, «IBM,» 13 Agosto 2019. [En línea]. Available: https://developer.ibm.com/recipes/tutorials/connecting-multiple-ti-sensortags-to-iot-platfom-using-a-raspberry-pi-gateway/.

[27] Fundación Raspberry Pi, «Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/.

[28] Etcher, «Etcher,» [En línea]. Available: https://www.balena.io/etcher/.

[29] Webhookrelay, «Webhookrelay,» [En línea]. Available: https://webhookrelay.com/.

[30] Webhookrelay, «Webhookrelay,» [En línea]. Available: https://webhookrelay.com/v1/installation/node-red.

[31] IBM IoTF, «IBM,» 20 Mayo 2019. [En línea]. Available: https://developer.ibm.com/recipes/tutorials/how-to-register-gateways-in-ibm-watson-iot-platform/.

[32] IBM, «IBM,» [En línea]. Available: https://ibmcloud-watson-

day.mybluemix.net/files/Lab3_IoT.v2.2.pdf.

[33] Wikipedia, «Wikipedia,» 27 Julio 2020. [En línea]. Available: https://en.wikipedia.org/wiki/Constrained_Application_Protocol. [Último acceso: 10 Agosto 2020].

[34] E. Lupander, «Callista,» 15 Marzo 2019. [En línea]. Available: https://callistaenterprise.se/blogg/teknik/2019/03/15/a-quick-home-automation/. [Último acceso: 17 Agosto 2020].

[35] No-Ip, «No-IP,» [En línea]. Available: https://www.noip.com/support/faq/frequently-asked-questions/#:~:text=What%20Does%20No%2DIP%20Do%3F&text=Our%20Free%20Dynamic%20DNS%20service%20takes%20your%20Dynamic%20IP%20address,with%20the%20current%20IP%20address..

[36] Hamachi, «Hamachi,» [En línea]. Available: https://www.vpn.net/.

[37] J. Boe, «Texas Instruments,» 3 Junio 2015. [En línea]. Available: https://e2e.ti.com/blogs_/b/process/archive/2015/06/03/create-develop-prototype-3d-print-your-iot-idea-with-simplelink-sensortag . [Último acceso: 13 Agosto 2020].

English Summary

Introduction

Motivation for the essay

Internet of Things (IoT) is a way to connect several kinds of devices to the internet to send data or receive orders. The data collection is made by sensors that send that information to another place over the internet, the orders are sent by that place to be played by actuators.

This technology is becoming more and more popular over time, partly because sensors are becoming cheaper and can be attached to any device and partly because the internet connection is way cheaper and faster than the past, so sending the data from several sensors all over the world is now viable.

With all these considerations, it's estimated that in 2020 there'll be over 30000 million of IoT devices, this growth means the future of IoT will depend on these factors:

- AI (Artificial Intelligence) will be more and more necessary, having tons of data from several sensors will need an AI to classify and analyze it.
- 5G will allow more data to be sent over the internet because it will let more quantity of data to be sent faster than today.
- Edge Computing will be more popular, the processing cost of a network will be lessened by decentralizing the place where all the data goes, having processors near the sensors (the edge) to process part of the data before sending it to the central processor.
- A new security standard will be necessary because of all the data going a central piece of the system.

IoT will flourish in new sectors due to the simplicity and the cheap cost of it, some examples are:

- Domotics: IoT is becoming more popular in the households all over the world, either being integrated with home appliances or by having voice-activated helpers such as the ones made by Google or Amazon.
- Health: sensors benefit this sector because they can be used to track the status of different patients without making them go to the hospital in person.
- Industry: IoT can be used to collect data from several sensors and track if the machinery is working correctly or the activity of the workers so their productivity can be measured.
- Security: in this sector there's improvements to be made, but plenty of households or small businesses are using it to have internet connected cameras to protect their property.
- Other uses: there's plenty of other uses, such as smart cities, having the traffic lights automated or sensors to check if something is wrong with the water/electricity supply. Car companies are using it too to have voice-activated functions or self-driving cars.

Regulatory Framework

Because IoT receives and stores a lot of data about households or industry, many security and privacy standards are to be met so companies don't use data wrongly or strangers get access to this data without the owner's permission.

Objectives

The objective of this project is to compare 2 IoT options which store, treat and visualize data from sensors and create an according action by several actuators. The options are:

Local: it doesn't need internet, a Texas Instrument CC2650 is connected to a Raspberry Pi by BLE and node-red is used to treat these data, store them, show them in a dashboard and depending of the value of the data, make different things such as turning a light on if the sensor detects low lights or sending an alert with Google Nest if the sensor has been moved/detects high temperatures. The dashboard apart from showing data has a manual control for the light.

Cloud: this option is the local version but cloud-based, the data go from the raspberry pi to the Watson IoT cloud platform so they can be monitored over the internet. A control over the internet for the system could be implemented but hasn't been implemented yet.

Both options can get temperature alerts and voice commands over the Google Nest to change the bulb state or get the sensor data, but they need internet to work.

The primary objective with these options is making a low-cost, robust, secure and scalable system to monitor the temperature/light/pressure/humidity of a house in real time and control the lightning of a house.

Results

This section will describe the different tests made to check the system is working properly and a comparison of the two different architectures.

Tests

The first test is checking the raspberry pi is detecting the sensor, this command will run a script written on python

sudo python -u /home/pi/sensortag.py

if the command works, the Raspberry OS terminal should show something like this:

```
pi@raspberrypi:~ $ sudo python -u /home/pi/sensortag.py
{"devicename":"ST-1","status":"found"}
{"devicename":"ST-1","status":"started"}
setting daemon
{"devicename":"ST-1","status":"enabled ['accelerometer', 'barometer', 'battery',
 'humidity', 'IRtemperature', 'lightmeter', 'magnetometer']"}
{"devicename":"ST-1","accelerometer":[-0.158447265625,-0.1240234375,1.006835937
]}
{"devicename":"ST-1","barometer":[24.15,919.74]}
{"devicename":"ST-1","battery":79}
{"devicename":"ST-1","humidity":[23.3654785156,47.4060058594]}
{"devicename":"ST-1","IRtemperature":[0.0,0.0]}
{"devicename":"ST-1","lightmeter":1.05}
{"devicename":"ST-1","magnetometer":[0.0,0.0,0.0]}
{"devicename":"ST-1","accelerometer":[0.0,0.0,0.0]}
{"devicename":"ST-1","barometer":[24.16,919.76]}
{"devicename":"ST-1","battery":79}
{"devicename":"ST-1","humidity":[23.3654785156,47.4060058594]}
{"devicename":"ST-1","IRtemperature":[23.4375,18.71875]}
{"devicename":"ST-1","lightmeter":1.21}
{"devicename":"ST-1","magnetometer":[-69.8715506716,20.6915750916,-142.591941392
]}
```

Fig. 41 Sensor connection test.

The next step is to check if node-red is working, to do that, the script previously used is attached to a node and a debug node will check if data are coming across the flow.

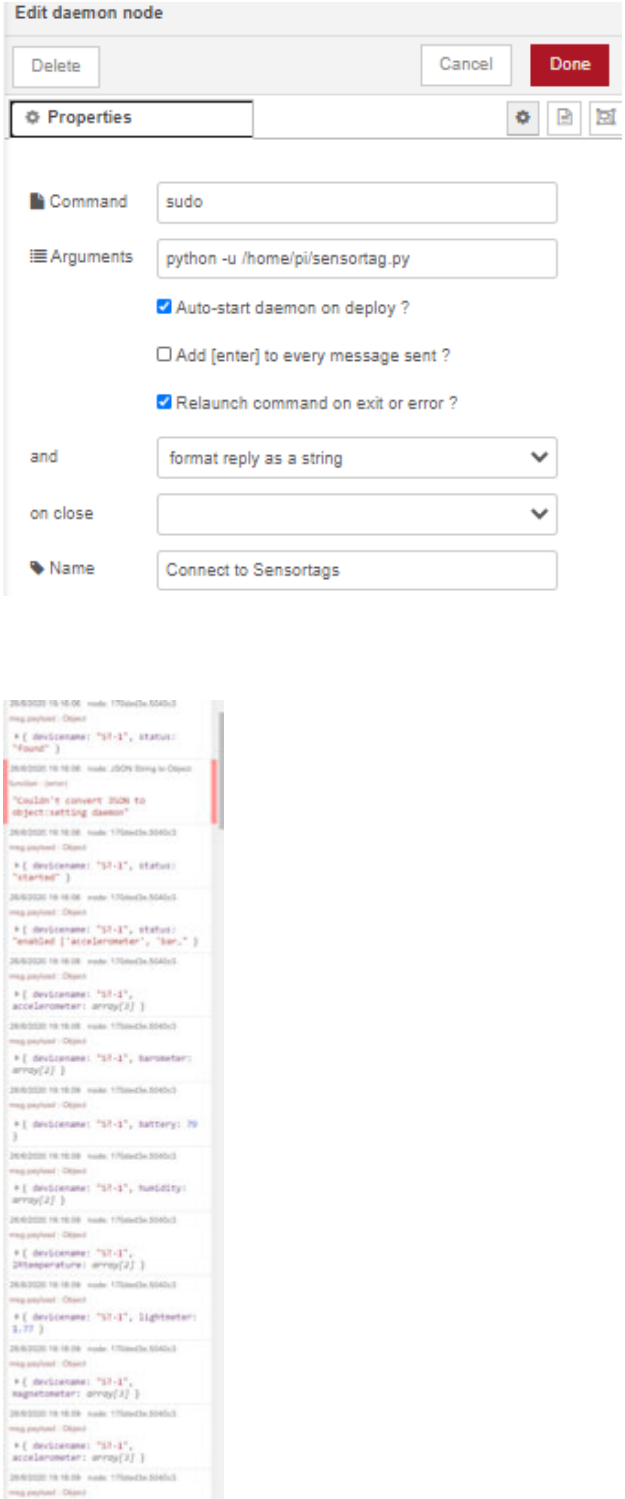

Fig. 42 Sensor connection to node-red test.

If the data come across the flow, more nodes are added (and checked if they are working properly).

Then, the Trädfri nodes are added, the way to test them is to inject into them the strings "on" or "off" and checking if the light turns on or not.

To check the dashboard nodes, the only thing needed is to open the dashboard and see if the different graphics/buttons appear on it and work as intended.

The tests on Watson IBM are pretty much the same, the nodes are attached and then the web is checked to see if the data appears there.

To ensure the Google nest is working properly, there's two steps, first the speaker is checked, any word is injected into its node, if the Google Nest emits the word, then it's working. Then, the voice control is tested by trying to say the voice command, if the Google Nest answers with the right answer, it's working.

The last test is the mongodb working properly, the nodes are added as usual and then we use these commands:

mongo

use (the database)

db.(collection).find()

if the data appears in its collection, then it's working.

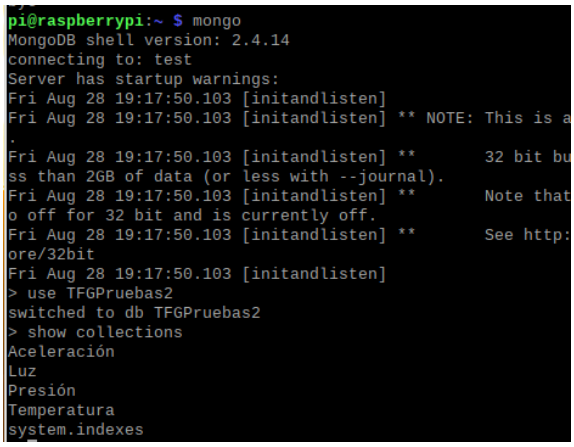

Fig. 43 Mongo start test.

Security

COAP – It has 4 security modes:

1. Nosec – Messages are sent without DTLS.
2. PresharedKey – Messages are sent with DTLS, it has a public key list and they can be shared to some nodes.
3. RawPublicKey – Messages are sent with DTLS and an asymmetric key without a certificate.
4. Certificate – Messages are sent with DTLS and need a X.509 certificate to validate the message.

It's vulnerable to DDOS (Distributed Denial of Service) but in this case it's not a probable worry.

DTLS – It's based on TLS and uses UDP, that's why the packets may be reordered in the destination. In this case, the Ikea Gateway uses it to communicate with the IO devices. When a new device is connected, it does a handshake with to confirm the right person is connecting and if so, it lets them use the lightbulbs.

It had a Man In The Middle vulnerability, but it has been solved in the last version

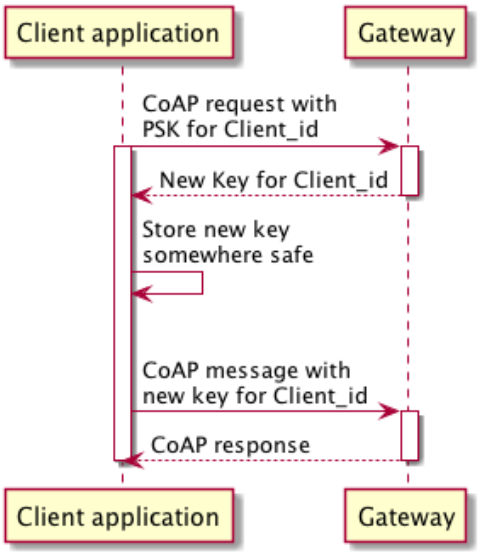

Fig. 44 DTLS Handshake. Source: [34].

BLE – Perhaps it's one of the most vulnerable parts of this project, BLE does not check the identity of the person connecting to the system, so if a person connected when the sensor just turned on he could steal the data off that sensor, the only solution then is to check that the system has connected correctly when it starts

Scalability

The system is very easily scalable, the number of sensors can be increased (of the same type or different sensors) or more actuators added. The script supports more CC2650 connecting and node-red supports the connection of different sensors/actuators.

Versatility

As said before, the system can be versatile, adding new features easily with the use of node-red, which supports almost any kind IoT dispositive. We could even take out the sensors and the lightbulbs and the structure of the new system built would be very similar

Comparison between the 2 architectures

The two architectures are a local-based one and a cloud-based one, now the different characteristics of them will be listed.

Regarding privacy, local-based it's more secure because it doesn't need internet connection, so only the network users can access it. The cloud-based one could have its information breached on the way to the cloud or in the cloud database.

Internet control its not implemented in either of them, but the implementation would be different for each one:

> The local-based would need a system to connect to it over the internet, there's some free alternatives such as no-ip or Hamachi, the first one is based on the Dynamic DNS, reads the network IP and gives an access hostname that adapts to

the IP even if it changes its value, making it accessible from the internet with that hostname. The second one creates a VPN so anyone with the login credentials can access the network and the dashboard.

The cloud-based option it's easier to implement, because Watson IBM has a system to send requests over the internet, node-red can get and use them.

Lastly, to get data persistency the best option is to mix both architectures, saving a copy in the local option and another one in the cloud, so if any of them fails, there's another one.

Conclusions and future work

As you can see, IoT is a growing sector that's being used more and more in domotics, so the system built in this essay is an example to show how the technology works and compare 2 different storage/dashboard systems and how different devices work in node-red.

The end of this essay is to understand how the different tools used work. Node-red allows great versatility, almost anything can be attached to it and made work, it doesn't matter if sensors, actuators or code in any programming language. Mongodb is an easy to use database. IFTTT and Webhookrelay are an easy way to customize the orders of an IoT voice recognition device.

We could expand the work by using more sensors/actuators to control several rooms, it could also have an internet control, so it can be used remotely, although for a lightbulb it's kind of a nonsense.