
UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

# Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW

Rubén Gómez Villegas

Mes, 2024


UNIVERSIDAD DE CASTILLA-LA MANCHA

TRABAJO FIN DE GRADO

Departamento de Tecnologías y Sistemas de Información

Tecnología Específica de Sistemas de Información

# Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW

Autor: Rubén Gómez Villegas

Tutor Académico: Rubén Cantarero Navarro

Cotutor Académico: Ana Rubio Ruiz

Mes , 2024

*D*edicado a mi familia y a todos
aquellos ...

# Declaración de Autoría

Yo, Rubén Gómez Villegas con DNI 02318379W declaro que soy el único autor del trabajo fin de grado titulado “Protocolo de mensajería ligero inspirado en MQTT para redes ESP-NOW” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Talavera de la Reina, a…..

Fdo: ……

# Resumen

Esta plantilla puede modificarse para adaptarse a las particularidades de cada Proyecto, tanto en contenido como en formato, siempre y cuando se respete las directrices básicas indicadas en la guía de estilo y formato para la elaboración de TFG del Grado en Ingeniería Informática de la Facultad de Ciencias Sociales y Tecnologías de la Información de Talavera de la Reina.

# Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# Agradecimientos

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

# Índice general

# Índice de figuras

# Índice de Tablas

# Índice de Listados

# Acrónimos

**ACK** Acknowledgement, Acuse de recibo

**AES** Advanced Encryption Standard, Estándar de Cifrado Avanzado

**AIoT** Artificial Intelligence of Things, Inteligencia Artificial de las Cosas

**AMQP** Advanced Message Queuing Protocol, Protocolo Avanzado de Colas de Mensajes

**ARPANET** Advanced Research Projects Agency Network, Red de la Agencia de Proyectos de Investigación Avanzada

**API** Application Programming Interface, Interfaz de Programación de Aplicaciones

**BLE** Bluetooth Low Energy, Bluetooth de Baja Energía

**CBCF** Congressional Black Caucus Foundation, Fundación del Caucus Negro del Congreso de los Estados Unidos

**CoAP** Constrained Application Protocol, Protocolo de Aplicación Restringida

**CPU** Central Processing Unit, Unidad Central de Procesamiento

**DDS** Data Distribution Service, Servicio de Distribución de Datos

**DTLS** Data Transport Layer Security, Protocolo de Seguridad de la Capa de Transporte

**ECC** Elliptic Curve Cryptography, Criptografía de Curva Elíptica

**ECDH** Elliptic-curve Diffie–Hellman

**ETS** Erlang Term Storage, Almacenamiento de términos de Erlang

**GHz** Gigahercios

**GPIO** General Purpose Input/Output, Entrada/Salida de Propósito General

**HTML** HyperText Markup Language, Lenguaje de Marcado de Hipertexto

**HTTP** HyperText Transfer Protocol, Protocolo de Transferencia de Hipertexto

**HTTPS** HyperText Transfer Protocol Secure, Protocolo Seguro de Transferencia de Hipertexto

**ID** Identificador, Identifier

**IDE** Integrated Development Environment, Entorno de Desarrollo Integrado

**IEEE** Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos

**IoT** Internet of Things, Internet de las Cosas

**IP** Internet Protocol, Protocolo de Internet

**ITU** International Telecommunication Union, Unión Internacional de Telecomunicaciones

**Kbps** Kilobits por segundo

**KiB** Kibibyte

**LAN** Local Area Network, Red de Área Local

**mA** Miliamperios

**MAC** Media Access Control, Control de Acceso a Medios

**MAN** Metropolitan Area Network, Red de Área Metropolitana

**Mbps** Megabits por segundo

**MHz** Megahercios

**MIB** Management Information Base, Base de Información Gestionada

**MiB** Mebibyte

**MQTT** Message Queue Telemetry Transport, Transporte de Telemetría en Colas de Mensajes

**MTU** Maximum Transmission Unit, Unidad de Transmisión Máxima

**mW** Milivatios

**NACK** Negative Acknowledgement, Acuse de recibo Negativo

**OASIS** Organization for the Advancement of Structured Information Standards, Organización para el Avance de Estándares de Información Estructurada

**OSI** Open Systems Interconnection, Interconexión de Sistemas Abiertos

**QoS** Quality of Service, Calidad de Servicio

**RAM** Random Access Memory, Memoria de Acceso Aleatorio

**REST** Representational State Transfer, Transferencia de Estado Representacional

**RFID** Radio Frequency Identification, Identificación por Radiofrecuencia

**RISC** Reduced Instruction Set Computer, Computador con Conjunto de Instrucciones Reducido

**RSA** Rivest–Shamir–Adleman

**SHA** Secure Hash Algorithm, Algoritmo de Hash Seguro

**SoC** System on a Chip, Sistema en un Chip

**SNMP** Simple Network Management Protocol, Protocolo Simple de Gestión de Redes

**SSL** Secure Sockets Layer, Capa de Puertos Seguros

**TCP** Transmission Control Protocol, Protocolo de Control de Transmisión

**TLS** Transport Layer Security, Seguridad de la Capa de Transporte

**TFG** Trabajo Final de Grado

**UDP** User Datagram Protocol, Protocolo de Datagramas de Usuario

**V** Voltios

**VoIP** Voice over Internet Protocol, Voz sobre Protocolo de Internet

**WEP** Wired Equivalent Privacy, Privacidad Equivalente a Cableado

**WPA** Wi-Fi Protected Access, Acceso Wi-Fi Protegido

**XML** Extensible Markup Language, Lenguaje de Marcado Extensible

**XMPP** Extensible Messaging and Presence Protocol, Protocolo Extensible de Mensajería y Comunicación de Presencia

**µs** Microsegundos

**ºC** Grados centígrados

# Capítulo 1

# Introducción

El capítulo de Introducción debe describir el problema que se pretende resolver con el desarrollo del Trabajo Fin de Grado (TFG). Debe dar respuesta al qué sin especificar cómo se va a realizar, para lo cual se usarán el resto de los capítulos del documento. El lector de este documento debe tener claro el alcance del proyecto habiendo leído únicamente el capítulo de Introducción.

## 1.1. Motivación

Esta sección aborda la motivación del trabajo. Se trata de señalar la necesidad que lo origina, su actualidad y pertinencia. Puede incluir también un estado de la cuestión (o estado del arte) en la que se revisen estudios o desarrollos previos y en qué medida sirven de base al trabajo que se presenta.

En este capítulo debería introducirse el contexto disciplinar y tecnológico en el que se desarrolla el trabajo de modo que pueda entenderse con facilidad el ámbito y alcance del TFG. Puesto que un TFG no tiene que ser necesariamente un trabajo con aportes novedosos u originales, solo es necesario la inclusión de estado del arte cuando este contribuya a aclarar aspectos clave del TFG o se desee justificar la originalidad del trabajo realizado. Si la sección estado del arte es muy extensa, considera la opción de introducirla como un capítulo independiente.

## 1.2. Redacción de la Memoria

Durante la realización de la memoria del TFG es importante tener presente respetar la guía de estilo de la institución. Por tanto, el empleo de plantillas para un sistema de procesamiento de textos (por ejemplo, Word o LaTeX) puede requerir su adaptación cuando la plantilla mencionada no haya sido suministrada en la institución a la que se dirige el trabajo.

Para redactar un trabajo académico de modo efectivo se deben tener presentes una serie de normas que

ayuden a conseguir un resultado final que sea claro y de fácil lectura.

A la hora de redactar el texto se debe poner especial atención en no cometer plagio y respetar los derechos de propiedad intelectual. En particular merece gran atención la inclusión de gráficos e imágenes procedentes de Internet que no sean de elaboración propia. En este sentido se recomienda consultar el manual de la Universidad de Cantabria [1] en el que se explica de modo conciso cómo incluir imágenes en un trabajo académico.

## 1.3. Estructura del Documento

Este capítulo suele incluir una sección que indica la estructura (capítulos y anexos) del documento y el contenido de cada una de las partes en que se divide. Por tanto, las secciones que suelen acompañar este capítulo son:

- Motivación. Responde a la pregunta sobre la necesidad o pertinencia del trabajo.
- Objetivo. Determina de modo claro el propósito del trabajo descrito que puede desglosarse en subobjetivos cuando el objetivo principal se puede descomponer en módulos o componentes. Es muy importante definir el objetivo de modo apropiado. El Capítulo Objetivos de esta guía explica cómo definir el objetivo.
- Antecedentes o Contexto disciplinar/tecnológico. También puede denominarse Estado del Arte cuando se trata de comentar trabajos relacionados que han abordado la cuestión u objetivo que se plantea.
- Estructura del documento. Resumen de los capítulos y anexos que integran el documento.

## 1.4. Abreviaturas y Acrónimos

Un TFG que utiliza muchas abreviaturas y acrónimos puede añadir esta sección dónde se muestra el conjunto de abreviaturas y acrónimos y su significado.

## 1.5. Objetivos

Para hacer un planteamiento apropiado de los objetivos se recomienda utilizar la Guía para la elaboración de propuestas de TFG en la que se explica cómo definir correctamente los objetivos de un TFG.

### 1.5.1. Objetivo General

Introduce y motiva la problemática (i.e ¿cuál es el problema que se plantea y por qué es interesante su resolución?).

[1] Guía de Imágenes: https://web.unican.es/buc/Documents/Formacion/guia_imagenes.pdf

Debe concretar y exponer detalladamente el problema a resolver, el entorno de trabajo, la situación y qué se pretende obtener. También puede contemplar las limitaciones y condicionantes a considerar para la resolución del problema (lenguaje de construcción, equipo físico, equipo lógico de base o de apoyo, etc.). Si se considera necesario, esta sección puede titularse Objetivos del TFG e hipótesis de trabajo. En este caso, se añadirán las hipótesis de trabajo que el/la estudiante pretende demostrar con su TFG.

Una de las tareas más complicadas al proponer un TFG es plantear su Objetivo. La dificultad deriva de la falta de consenso respecto de lo que se entiende por objetivo en un trabajo de esta naturaleza.

En primer lugar, se debe distinguir entre dos tipos de objetivo:

- La finalidad específica del TFG que se plantea para resolver una problemática concreta aplicando los métodos y herramientas adquiridos durante la formación académica. Por ejemplo, Desarrollo de una aplicación software para gestionar reservas hoteleras on-line.
- El propósito académico que la realización de un TFG tiene en la formación de un graduado. Por ejemplo, la adquisición de competencias específicas de la intensificación cursada.

En el ámbito de la memoria del TFG se tiene que definir el primer tipo de objetivo, mientras que el segundo tipo es el que se añade en el Capítulo de Conclusiones y que justifica las competencias específicas de la intensificación alcanzadas y/o reforzadas con la realización del trabajo.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria del trabajo. Así en el caso de estudios y validación de hipótesis el apartado de resultados y conclusiones debería incluir los resultados de experimentación y los comentarios de cómo dichos resultados validan o refutan la hipótesis planteada.

### 1.5.2. Objetivos Específicos

Generalmente, el objetivo general puede ser descompuesto en varios objetivos más específicos que se pretenden alcanzar. En esta sección se enumeran y describen cada uno de ellos.

Junto con la definición de estos objetivos se puede especificar los requisitos que debe satisfacer la solución aportada. Estos requisitos especifican características que debe poseer la solución y restricciones que acotan su alcance. En el caso de un trabajo cuyo objetivo es el desarrollo de un artefacto los requisitos pueden ser funcionales y no funcionales.

Al redactar el objetivo de un TFG se debe evitar confundir los medios con el fin. Así es habitual encontrarse con objetivos definidos en términos de las acciones (verbos) o tareas que será preciso realizar para llegar al verdadero objetivo. Sin embargo, a la hora de planificar el desarrollo del trabajo si es apropiado descomponer todo el trabajo en hitos y estos en tareas para facilitar dicha planificación.

La categoría del objetivo planteado justifica modificaciones en la organización genérica de la memoria

del trabajo. Así en el caso de estudios y validación de hipótesis el apartado de resultados y conclusiones debería incluir los resultados de experimentación y los comentarios de cómo dichos resultados validan o refutan la hipótesis planteada.

# Capítulo 2

# Estado del arte

## 2.1. Internet de las Cosas

Dado que una de las principales motivaciones de este Trabajo de Fin de Grado es el aporte que puede ofrecer a la facilidad de implementación del Internet de las Cosas, es importante conocer el joven concepto que nos rodea en la actualidad.

El siglo XX dio lugar al desarrollo de una cantidad de inventos que permitieron una revolución y un avance ágil en la sociedad, inventos nacidos de ideas creativas de sus desarrolladores y que en la actualidad son indispensables debido al frecuente uso y la manera en la que facilita la vida humana. Entre estos inventos se encuentra una de las herramientas más importantes y que mejoró la comunicación, el Internet. Nacido en 1969, ha permitido enlazar a personas con personas y tener acceso a información, pero hoy en día, no solo las personas están conectadas a Internet, también millones de objetos cuyas funciones son dependientes de la red.

*Internet de las Cosas* o *Internet of Things* es un término cuyo origen está disputado. El primer uso de este término data del discurso que realizó en septiembre de 1985 Peter T. Lewis, cofundador de la primera compañía de telefonía móvil de Estados Unidos, Cellular One. En tal discurso realizado en la Conferencia Legislativa Anual de la CBCF en Washington D.C., Lewis comentó de manera acertada “Predigo que no sólo los humanos, sino también las máquinas y otras cosas se comunicarán interactivamente a través de Internet.” **(TODO: mencionar al podcast)**. Por otro lado, el mismo término fue acuñado en 1999 cuando Kevin Ashton, directivo de Procter & Gamble, tuvo la iniciativa de investigar el uso de etiquetas de identificación por radiofrecuencia (RFID) y otros sensores en los productos de la cadena de suministro, y para presentar el proyecto tuvo que idear un título llamativo para la presentación. Esta presentación le permitió encontrar financiación además de cofundar y dirigir el laboratorio Auto-ID Center del Instituto de Tecnología de Massachusetts, en el cuál construyó la base del Internet de las cosas.

Es esencial tener una clara comprensión del significado del Internet de las Cosas, en mayor parte por la confusión inherente al término en sí y por las aplicaciones cotidianas de esta tecnología. A primera vista, "Internet de las Cosas" podría dar la impresión de ser un término moderno para referirse a "conectar a Internet algo para controlarlo", una definición bastante simple para alguien que por ejemplo simplemente controla las luces de su hogar desde su teléfono móvil. Es tanta la confusión que no existe una definición formal única, debido a que hay sistemas que implementan IoT y no cumplen todas las definiciones, por lo tanto, no se ha llegado a un consenso para formalizar una definición única, y a la hora de intentar conocer su significado se encuentran una disparidad de definiciones. En 2012, la ITU (Unión Internacional de Telecomunicaciones) recomendó una aclaración del término, definiendo el Internet de las Cosas como una infraestructura global que permite ofrecer servicios avanzados a todo tipo de aplicaciones conectando objetos entre sí e interoperando tecnologías de la información y comunicación, aprovechando capacidades de identificación, obtención de datos, procesamiento y comunicación y cumpliendo con requisitos de seguridad y privacidad. Sin embargo, por lo general, podemos entender que el Internet de las Cosas trata de dotar de capacidades de comunicación además de procesamiento, captura y/o análisis de datos a distintos tipos de entes, como dispositivos físicos, objetos, edificaciones, terrenos, sistemas, hardware, software, e incluso contextos y situaciones, ya sea añadiéndoles dispositivos o integrando las capacidades en los propios objetos. Estos entes pueden estar compuestos de sensores, que recopilan datos, o actuadores, que controlan otros objetos, y a través de redes, privadas o públicas, pueden intercambiar información con otros dispositivos, recopilar la información en un mismo dispositivo y transferir órdenes, además que existe la posibilidad de formar una agrupación de dispositivos para identificarlos como un único sistema que trata los datos. Estas redes de conexión de dispositivos IoT pueden utilizar la información transferida para poder automatizar sus comportamientos, pero el nodo final de la conexión suelen ser las personas, ya que utilizan los objetos como fuentes de información, para a su vez utilizar los datos recibidos como descubrimiento de oportunidades de negocio y nuevos servicios, monitorización y evaluación del estado y el comportamiento de los objetos y el entorno, y la toma de decisiones sobre los propios objetos manipulándolos remotamente y programándolos. Dependiendo del entorno en que se implemente, estos sistemas deben cumplir requisitos de seguridad y privacidad, evitando manipulaciones y accesos indebidos o incluso daños en los propios objetos y en el entorno que los rodea.

Un ejemplo concreto de la aplicación de Internet de las Cosas es un sistema de riego por aspersión inteligente. Como se puede comprobar en la Figura 2.1, existe un jardín dividido en zonas, cada una con un dispositivo compuesto por sensores de temperatura y humedad y actuadores que activan y desactivan los aspersores que riegan la vegetación. Estos dispositivos están conectados y se comunican con un único gateway o puerta de enlace, teniendo la capacidad de recibir información de los dispositivos y mandar órdenes a estos. El gateway está comunicado a través de Internet con un servidor compuesto por una base de datos, donde se almacenan los datos históricos y registros, y un servicio que le permite ser controlado desde otro dispositivo conectado a Internet en cualquier sitio y momento. Este último dispositivo, denominado cliente,

puede ser un teléfono móvil o un ordenador, y puede utilizarse para ver el estado del jardín y controlar los dispositivos manualmente desde una interfaz. La siguiente arquitectura, abstraída completamente de las limitaciones y los problemas que puede ofrecer su implementación, podrá dar lugar a dos casos de uso:

- El usuario quiere activar los aspersores. El usuario, desde la interfaz el dispositivo cliente como puede ser un botón, enviará la orden de activar los aspersores al servicio, que a su vez se lo enviará al gateway. El gateway enviará una orden compatible a los dispositivos instalados en el jardín, haciendo que los aspersores comiencen a funcionar. Esta secuencia de ejecución será parecida si el usuario desea desactivar los aspersores.
- Los aspersores funcionan cuando la temperatura es muy elevada y la humedad es baja. El usuario previamente, desde la interfaz del cliente, ha establecido que los aspersores funcionen de manera automática cuando, por ejemplo, el ambiente supere una temperatura de 42ºC y la humedad sea considerada baja. Estos parámetros los recibe el servidor y se los pasa al gateway, el cual los recordará. A partir de ese momento, el gateway irá recibiendo de los sensores de cada zona lecturas de temperatura y humedad, y las comparará con los parámetros establecidos. En el caso de que se superen la temperatura y humedad, se activarán los aspersores de la zona. Una vez activados, se mantendrá la lectura de temperatura y humedad, y cuando las lecturas sean inferiores, se desactivarán los aspersores. Además, con cada activación de los aspersores, el gateway notificará al servicio, que a su vez notifica al usuario, especificando la zona activada.

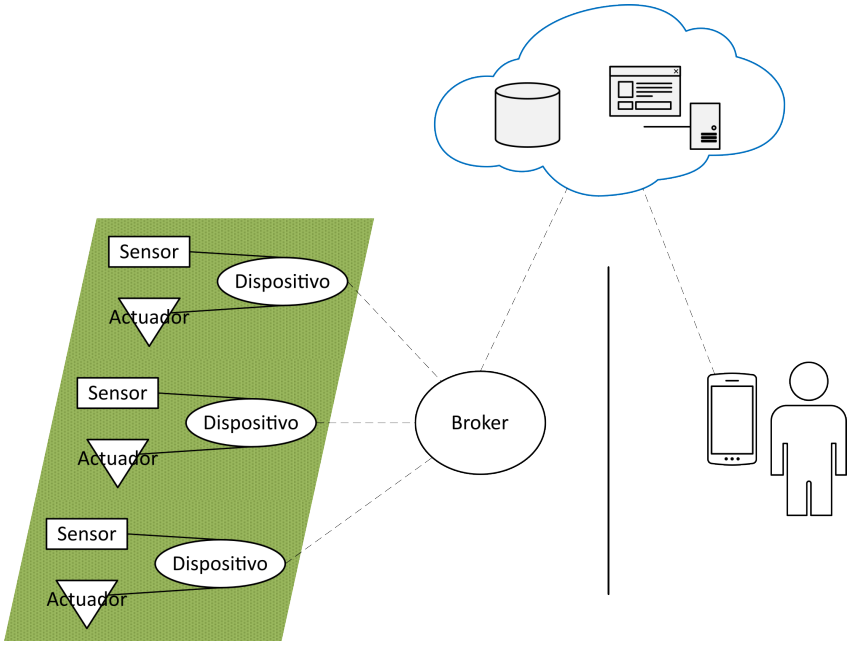


Figura 2.1: Esquema de un sistema de riego por aspersión utilizando dispositivos IoT

Cabe recalcar que el mencionado ejemplo puede volverse más complejo, por ejemplo añadiendo comprobaciones de previsión meteorológica en los próximos días, sensores de luz, control del caudal del agua,

pero igualmente cumple con los requisitos para ser un sistema que implementa el Internet de las Cosas, ya que dota a un jardín de capacidad de comunicación, captura y análisis de datos, y se comunica con otros dispositivos y con personas para informar del estado y recibir órdenes.

### 2.1.1. Factores, limitaciones y desafíos a tener en cuenta

Al desarrollar una arquitectura del Internet de las Cosas, es necesario tener en cuenta una serie de factores que afectan tanto a los dispositivos como a la comunicación que puedan establecer entre ellos, ya que no todos los dispositivos funcionan de la misma manera en cualquier ambiente y entorno.

Las principales consideraciones se encuentran en los dispositivos, ya que cada uno tiene sus características y se deben aplicar soluciones que puedan adaptarse a cada uno de ellos, además de depender entre sí. En los dispositivos, uno de los mayores problemas es la alimentación, ya que pueden alimentarse estando conectados constantemente o mediante una batería. En ambos casos habrá que tener en cuenta el consumo, ya que el coste de los recursos es importante, pero en el caso de la batería tiene aún mayor importancia al tener una capacidad limitada y, a su vez, limitar la vida útil del dispositivo. Si el dispositivo se encuentra en un objeto móvil, o si de por sí es un objeto móvil, también es algo a tener en cuenta, ya que, en este caso, el rango de movilidad condiciona el tipo de alimentación. No es lo mismo un dispositivo que puede moverse unos pocos centímetros de un enchufe que se encuentra en la pared, que un dispositivo diseñado para recorrer kilómetros, el cual deberá estar conectado a una batería o a un panel solar. En caso de que la batería sea un factor demasiado limitante en el funcionamiento de la arquitectura, está la posibilidad de utilizar baterías más grandes y de mayor capacidad, suponiendo también un peso y un tamaño adicional a los ya existentes en el dispositivo, limitando también el entorno en el que utilizarlos. Un dispositivo de grandes dimensiones no podrá pasar desapercibido si eso fuese un requisito, o un dispositivo de gran densidad será más difícil de implementar en zonas donde sea necesario que flote o se necesite sujetar a un objeto endeble.

El Internet de las Cosas se basa en la transmisión de datos, por lo que habría que tener en cuenta la frecuencia de transmisión y el tamaño de los datos. Una transmisión constante supondrá un mayor consumo que una transmisión que ocurra en intervalos de tiempo más espaciados. Lo mismo ocurre con los datos, ya que, si es necesario transmitir grandes cantidades de datos, también se requiere procesarlos, y en ambos casos da lugar a un mayor consumo de energía.

El entorno y la manera de desplegar los dispositivos suponen una serie de limitaciones, ya que si el entorno es abierto y los dispositivos están expuestos, pueden afectarles distintas condiciones ambientales, como la lluvia, la humedad, las vibraciones o las altas temperaturas, o pueden estar desplegados en zonas con conectividad limitada en términos de velocidad y disponibilidad. En el caso de montar la arquitectura con muchos dispositivos o en una zona en la que se encuentren una gran cantidad de estos, pueden ocurrir interferencias electromagnéticas que afecten la comunicación.

Por otra parte, de manera más técnica, los dispositivos tienen sus propias limitaciones en cuanto al funcionamiento. Mayormente, el Internet de las Cosas se centra en la interconexión de dispositivos heterogéneos, por lo tanto, no todos tendrán las mismas capacidades. Es necesario encontrar dispositivos que puedan comunicarse al menos con otro mediante una tecnología de comunicación establecida y compatible con ambos (teniendo en cuenta además la velocidad y alcance de transmisión), calcular y establecer una cantidad de dispositivos desplegados y que la topología de conexión los admita, y comprobar cuán adecuado es la velocidad y capacidad de procesamiento en los dispositivos; todos estos factores en cuanto a los requisitos del despliegue. También es responsabilidad del desarrollador del software de los dispositivos tener en cuenta las capacidades y las limitaciones de estos. No va a ser lo mismo programar el funcionamiento de un dispositivo siempre conectado a la corriente y con recursos elevados que un dispositivo portátil y con menor capacidad de procesamiento, por lo que el desarrollador debe aplicar diversos métodos para, por ejemplo, suspender el funcionamiento del dispositivo cuando no está ejerciendo ninguna función útil y activarlo de nuevo cuando sea necesario, u optimizar el código para minimizar el consumo de recursos.

Tratando la arquitectura IoT como un ente, es necesario conocer cómo hacer funcionar los dispositivos que la componen y cuál es el fin de la arquitectura. Dependiendo del uso que se les vaya a dar a los dispositivos, habrá que tener en cuenta la topología de conexión entre estos, la necesidad de escalabilidad en caso de necesitar ampliar el área de recolección de datos o para mejorar el procesamiento, la gestión de dispositivos para actualizarlos, reconfigurarlos, localizarlos o desconectarlos, y la recolección de datos.

Y por último, como riesgo, un tema muy importante es la ciberseguridad. Los dispositivos IoT pueden recopilar datos personales y sensibles que se transmiten constantemente por Internet, y los dispositivos utilizados para el Internet de las Cosas no suelen disponer de altas capacidades de procesamiento para establecer controles de seguridad, o por su general coste bajo los fabricantes de los dispositivos han podido descuidar aspectos como la seguridad en el cifrado o en controles de acceso y centrarse en optimizar los recursos para una mayor potencia. Por lo tanto, dependiendo de la sensibilidad de los datos, podría ser necesario implementar mecanismos de autenticación, cifrado, gestión del control de acceso y políticas, entre otros, que aseguren la confidencialidad, integridad y disponibilidad.

El Internet de las Cosas admite su implementación en una gran cantidad de situaciones y entornos. Pero antes de empezar a desplegar dispositivos es necesario realizar un análisis de requisitos y de posibles limitaciones en el entorno, un listado de soluciones a esas limitaciones y un diseño previo, además de pruebas y evaluaciones, resultando así en una arquitectura eficiente que aporte los resultados esperados.

{ “IoT fundamentals: Networking technologies, protocols, and use cases for the internet of things”. David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Jerome Henry, Robert Barton https://www.redeweb.com/articulos/sensores/los-sensores-inalambricos-iot-y-el-problema-de-la-corta-duracion-de-las-baterias https://neurona-ba.com/iot-a-tu-alcance/ https://www.ittrends.es/infraestructura/2021/12/los-problemas-de-conectividad-estan-dificultando-los-despliegues-de-iot https://deingenierias.com/curso/iot/1-4-desafios-y-limitaciones-de-

iot/ https://www.chakray.com/es/5-requesitos-de-una-arquitectura-iot/ https://www.incibe.es/incibe-cert/blog/riesgos-y-retos-ciberseguridad-y-privacidad-iot }

### 2.1.2. Ventajas y desventajas

El Internet de las Cosas es un concepto en desarrollo que actualmente presenta, al igual que cualquier concepto, sus ventajas y desventajas.

Las siguientes son las ventajas que puede ofrecer el uso del Internet de las Cosas:

- Capacidad de conectar dispositivos a la red para enviar y transmitir datos de forma rápida y en tiempo real.
- Mejora la recopilación de datos, haciéndolos accesibles en cualquier momento y lugar para ser utilizados cuando el usuario lo necesite. Este atributo también se aplica al control de los dispositivos.
- Implementación en una gran variedad de entornos para distintos usos, aunque no siempre de la misma manera, ya que se debe adaptar a las necesidades específicas.
- Habilidad de monitorización de procesos, controlando el consumo de recursos y habilitando la posibilidad de mejorar y optimizar su uso. Esto resulta en procesos más sostenibles y una reducción de costos en cuanto a recursos energéticos y personal humano, optimizando las actividades de los trabajadores en términos de tiempo y productividad.
- Automatización de tareas, reduciendo procesos manuales o tareas repetitivas. Además, se puede incluir el añadido de inteligencia a los dispositivos para que tomen decisiones por sí mismos a partir de los datos, minimizando el esfuerzo humano y limitando las funciones del especialista en IoT a la configuración, administración y soporte del sistema. Esta ventaja también conlleva una mejora en la eficiencia del sistema y, a su vez, en los procesos, recursos, costes, tiempo y productividad.
- Posibilidad de mejorar la calidad de vida del usuario, dependiendo del entorno en el que se aplique. Por ejemplo, con la ayuda del Internet de las Cosas, un paciente puede tener un seguimiento continuo y preciso de sus constantes vitales que permita prever cualquier anomalía. En una ciudad inteligente que monitorea el consumo y las calles gracias al IoT, los habitantes pagarán menos por el mantenimiento de esta y se podrán identificar zonas donde sean necesarias modificaciones urbanísticas, como añadir nuevas señales de tráfico para evitar accidentes. En un vehículo equipado con sensores y actuadores, facilita la evitación de accidentes y se pueden realizar avisos necesarios en caso de accidente.
- Ofrecer experiencias personalizadas y atractivas a clientes, proporcionando ofertas personalizadas y modificando la distribución de los productos en una tienda a partir de los datos recopilados sobre el comportamiento de los clientes utilizando IoT. Estos datos pueden incluir, por ejemplo, la manera en que se mueven por los pasillos y en qué productos se detienen más, resultando en un análisis profundo de tendencias y comparación con lecturas pasadas, útil para aplicar cambios y aumentar la

lealtad de los clientes.

- Tomar decisiones mejor informadas a partir de los datos generados y procesados en el sistema acerca de un negocio, tanto de manera interna como externa, ayudando a agregar valor al negocio de manera más rápida, impulsar la modernización y, por ejemplo, reducir el tiempo de comercialización de nuevos productos o servicios.

En cuanto a las desventajas, se pueden encontrar las siguientes:

- La implementación del IoT conlleva varias limitaciones mencionadas en el apartado **TODO: REFERENCIAR APARTADO ANTERIOR**, como la dependencia energética, la tecnología a utilizar y la seguridad, siendo esta última la que puede traer más riesgos según aumenta su popularidad. Además de lo mencionado anteriormente, el conjunto de dispositivos IoT se conecta a través de redes que pueden tener vulnerabilidades frente a infiltraciones y otras amenazas. Por ejemplo, el acceso no autorizado a una cámara puede ser peligroso. Los dispositivos recopilan y transmiten grandes cantidades de datos, los cuales también pueden ser afectados por las mismas amenazas, lo que puede resultar grave en caso de tratarse de datos sensibles sobre usuarios o empresas. La ciberseguridad debe tratarse con alta prioridad, y los dispositivos IoT no siempre se incluyen en la estrategia de seguridad de las empresas. Una correcta seguridad implica que los dispositivos estén protegidos contra manipulaciones físicas y ataques en la red, software y hardware, y un adecuado cumplimiento del reglamento general de protección de datos. A este punto se puede aplicar la privacidad, ya que las empresas deben implementar las medidas necesarias al tratar con datos personales.
- Siguiendo la desventaja anterior, las regulaciones legales de privacidad, protección de datos y ciberseguridad pueden variar entre países, siendo necesario conocer y aplicar la legislación de la zona en la que se va a implementar el sistema.
- Los dispositivos que conforman el Internet de las Cosas suelen ser heterogéneos y de distintos fabricantes, que utilizan diferentes estándares y protocolos. Sumado al hecho de no existir un estándar internacional de compatibilidad para IoT, esto dificulta la interoperabilidad y la comunicación máquina a máquina entre dispositivos, requiriendo configuraciones, conexiones de hardware específicas y el uso de dispositivos adicionales, y dando lugar a fuentes de datos complicadas de analizar e integrar a un sistema.
- Un sistema IoT implica una complejidad que debe ser gestionada. Pese a que los dispositivos de manera independiente puedan realizar tareas sencillas, el sistema está compuesto por una gran cantidad y variedad de ellos, por no mencionar las distintas tecnologías que utilizan cada uno (como lenguajes de programación o métodos de transporte de datos). Esto requiere una curva de aprendizaje en la implementación, desarrollar una estrategia sobre cómo y por qué implementarlos, un desarrollo para llevar a cabo la implementación, y un mantenimiento obligatorio en caso de errores o mal funcionamiento del software. A su vez, requiere una inversión en la adquisición de los dispo-

sitivos y en personal capacitado para ofrecer soporte al sistema, por lo que sería adecuado planificar un presupuesto previo antes de desplegar dispositivos.

- Los datos producidos deben ser adecuados y el usuario u organización debe estar preparada para tratarlos y analizarlos, y, debido a la gran cantidad de estos, implica un reto significativo al analizarlos y extraer información, especialmente en casos de falta de herramientas o experiencia en análisis.
- Los dispositivos IoT requieren métodos fiables para transportar datos, ya sea utilizando redes fiables o implementando mecanismos de acuses de recibo y verificación de mensajes. Si los datos se corrompen o son interceptados y modificados, el sistema puede dejar de funcionar correctamente.
- Existe la posibilidad de aplicar escalabilidad a la arquitectura IoT, algo que se debe considerar para evitar un rediseño completo de las conexiones y los dispositivos. Aunque la capacidad de escalabilidad es una ventaja para soportar grandes volúmenes de datos y mejorar el procesamiento, puede volverse una desventaja si se aplica incorrectamente, resultando en peor rendimiento.
- El aumento de la brecha tecnológica que generan estos sistemas impacta en la sociedad y limita el grupo social que tiene la capacidad de comprender su funcionamiento y sabe acceder y hacer uso de esta tecnología correctamente. Esto implica una necesidad de educación hacia los usuarios.

{ https://es.wikipedia.org/wiki/Internet_de_las_cosas https://www.ibm.com/es-es/topics/internet-of-things https://blog.ubisolutions.net/es/iot-tecnologias-casos-de-uso-ventajas-y-limitaciones-guia-completa https://www.tokioschool.com/noticias/ventajas-desventajas-internet-cosas/ https://kryptonsolid.com/principales-ventajas-y-desventajas-de-iot-en-los-negocios/ }

### 2.1.3. Primeros ejemplos de IoT

El primer dispositivo IoT que se conoce fue una máquina de Coca-Cola conectada a ARPANET a principios de la década de los 80 en la Universidad Carnegie Mellon de Pittsburgh, Pensilvania. Desde los años 70, el departamento de Ciencias de la Computación se encargaba de mantener la máquina de refrescos, que era cargada por alumnos en horarios erráticos y cuyos precios eran los más baratos (unos 10 centavos menos) no solo del campus sino también de la zona de Pittsburgh, lo que provocó una gran popularidad en ventas. A mediados de la misma década, debido a la expansión del departamento, se tuvieron que desplazar las oficinas lejos de la máquina que se encontraba en la tercera planta del edificio, lo que provocaba molestias a los programadores al ir en busca de su dosis de cafeína, ya que al bajar a la máquina y gastar el dinero que tanto les había costado conseguir, se encontraban con que o bien estaba vacía o que la Coca-Cola conseguida estaba recién cargada y, por lo tanto, desagradablemente caliente. Cansados de este problema, un grupo de personas se reunió para idear una solución. Instalaron microinterruptores en la máquina de Coca-Cola para poder detectar cuántas botellas había en cada una de las 6 columnas disponibles. Estos interruptores estaban conectados, a través de una interfaz de red instalada en la máquina, al ordenador principal del departamento, el PDP-10 (denominado CMUA). Para este ordenador, diseñaron un programa,

que apoyándose en un servidor programado para registrar la hora de la última transacción de cada columna, mostraba uno de los siguientes datos por cada columna de la máquina de Coca-Cola:

- "EMPTY", si la columna estaba vacía.
- El tiempo que llevaba la nueva botella cargada, por ejemplo, "1h 3m".
- "COLD", si la columna contenía una botella fría, es decir, que llevaba 3 horas cargada.

Para completar la idea, permitieron el acceso a esta información desde fuera del CMUA. Modificaron el servidor Finger ya existente en el CMUA, el cual era utilizado para obtener información sobre los usuarios conectados al sistema, para añadirle el usuario "coke", para que cada vez que se hiciese la petición a Finger con dicho usuario se ejecutase el programa de estado de la máquina de Coca-Cola. Debido a que las peticiones Finger formaban parte de los protocolos estándares de ARPANET, se podía descubrir el estado de la máquina de Coca-Cola tanto desde cualquier ordenador conectado en la Universidad Carnegie Mellon como desde cualquier parte del mundo conectada a la red, simplemente utilizando el comando `finger coke@cmua`.

Esta innovación se mantuvo en pie durante más de una década, pasando por una reescritura del código cuando se tuvo que migrar del PDP-10 al más moderno Unix Vaxen a principios de los 80, hasta que Coca-Cola descontinuó las botellas antiguas y las nuevas no eran compatibles con la máquina de refrescos existente, lo que provocó que fuera sustituida por completo.

La máquina de Coca-Cola de los 80 inspiró a que en 1992 un estudiante y un miembro del departamento de Ciencias de la Computación de la misma universidad instalasen a la nueva maquina y la de M&M que se encontraba cercana aisladores ópticos caseros para identificar la presencia del producto, y las conectasen a Internet para poder consultar el estado ejecutando un programa llamado "jf" ("junk food", comida basura).

Incluir imágenes: - Xerox Alto, ordenador fabricado en 1973, usado en la CMU para ver el estado de la maquina
- Máquina de cocacola https://engineercom.wpenginepowered.com/wp-content/uploads/2016/02/IIot1.png
- PDP-10

Posteriormente apareció otro notable objeto conectado a un Internet más similar al actual pero con menos personas (rondando las 3 millones), y fue una tostadora en 1990. El origen de este objeto ocurrió en la edición de 1989 de Interop, una conferencia anual que reunía a profesionales de las tecnologías de la información, ingenieros de redes, desarrolladores y proveedores de tecnología, en la que se podía poner a prueba la interoperabilidad de dispositivos informáticos, electrónicos y productos relacionados con Internet. En dicha edición, el organizador Dan Lynch retó a John Romkey a poner un dispositivo en línea y presentarlo en la siguiente edición. Romkey aceptó, ya que no desconocía la tecnología de la época que le permitiría conseguir dicho reto. John Romkey fue cocreador de la primera pila de protocolos de Internet TCP/IP, PC/IP del Instituto Tecnológico de Massachusetts (MIT) para MS-DOS en el IBM PC, además de

proveer pilas TCP/IP a través de la empresa FTP Software, la cual fundó.

Por aquella época, Romkey trabajaba en el Protocolo Simple de Gestión de Redes (SNMP), junto a Karl Auerbach y David Bridgham, de Epilogue Technology, y Simon Hackett, de Internode. SNMP es un protocolo sencillo que permite a los usuarios leer y escribir variables en un agente remoto y que hasta ese momento se había utilizado únicamente para gestionar dispositivos como routers, para inspeccionar recuentos de paquetes y tablas de enrutamiento, borrar contadores y habilitar o deshabilitar interfaces de red. John Romkey quería aprovechar la oportunidad para demostrar que, dado que SNMP podía controlar dispositivos físicos, también podría ampliar su alcance para gestionar salas completas que albergaran equipos de red y ordenadores, y consideró que controlar una tostadora sería un buen punto de partida. La tostadora, al ser un electrodoméstico sencillo, si su implementación del protocolo funcionaba y podía demostrarlo, el público podría entender que la automatización es posible y se podría llevar a niveles industriales. También, la implementación de Internet en la tostadora podría plantear problemas de seguridad a tener en cuenta, ya que habría que gestionar quién puede tostar el pan y quién puede consultar los hábitos de uso del electrodoméstico, además de tener el factor ridículo de poner una tostadora en Internet.

El primer paso que tomaron fue buscar una tostadora fácil de controlar físicamente, ya que la mayoría requería de bajar una palanca para también bajar el pan, algo fácil e intuitivo para el ser humano pero difícil de controlar por ordenadores sin hacer uso de la robótica. Con la ayuda de Two Guys and a Vax, una empresa amiga de Epilogue, descubrieron una tostadora Sunbeam Radiant Control, la cual no necesitaba ninguna palanca. Si la tostadora tiene energía y un pan insertado, el pan se baja automáticamente y se empieza a tostar, y finaliza cuando el pan se ha tostado hasta el punto deseado o se corta la corriente. Ese modelo era ideal para ser controlado por ordenador, ya que era más fácil de controlar la alimentación de energía que llegaba al dispositivo que una palanca mecánica de una tostadora.
Con la tostadora ideal elegida, el siguiente paso fue construir hardware capaz de controlar la energía. A principios de la década de los 90, la sociedad se encontraba en un entorno pre-web y no existían las placas de desarrollo actuales como Arduino, ni Wi-Fi, y mucho menos dispositivos conectables de manera inalámbrica a través de Wi-Fi, por lo que este paso era obligatorio. El primer intento de John Romkey fue crear un relé simple controlable a través del puerto paralelo de un ordenador portátil, el cual funcionó brevemente, pero terminó tostando el puerto del portátil debido a que el relé consumía más energía de la que el puerto podía suministrar. El segundo intento, mucho más exitoso y resultando en una manera fácil de controlar la tostadora mediante software, fue usar un interruptor de menor consumo conectado directamente al puerto paralelo del portátil para controlar un relé más grande que, a su vez, controlaba la alimentación de la tostadora.

El último paso fue la implementación del software para controlar la tostadora por Internet, es decir, programar la parte SNMP del proyecto que indicaba cómo de oscuro tostar el pan y el tipo de pan insertado. El lenguaje de SNMP permite especificar que controlar en un agente remoto, ya que cada dispositivo ges-

tionado puede tener su propia base de información gestionada (MIB por sus siglas en inglés). Dentro del MIB se pueden especificar objetos, y por cada uno el tipo de objeto, el acceso permitido, el estado y la descripción, y en el caso de la tostadora, se habían especificado en el código como objetos el fabricante, modelo, los controles de la tostadora para subir y bajar la rebanada (o iniciar y terminar el tostado), cómo de hecha se deseaba la tostada y el tipo de pan. Como ejemplo, la siguiente porción de código controlaba el inicio del tostado:

```
toasterControl OBJECT-TYPE
    SYNTAX INTEGER {
        up(1),
        down(2)
    }
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION"
        This variable controls the current state of the toaster.
        To begin toasting, set it to down(2).
        To abort toasting (maybe in the event of an emergency), set it to up(2)".
    :: = {toaster 3}
```

(TODO: REFERENCIAR codigo tomado del articulo)

Gracias a los objetos especificados, se podía identificar qué tostadora se estaba controlando, además de establecer el inicio, el tipo de pan y el nivel de tostado final del pan al modificar los objetos, todo a través del agente SNMP conectado al hardware que controlaba la alimentación de la tostadora, permitiendo un control remoto del electrodoméstico. Por ejemplo, si un usuario a través de SNMP modificaba la variable adecuada para indicar que quiere iniciar la tostadora, la tostadora recibía alimentación e iniciaba el proceso de tostado. La tostadora de Internet fue demostrada finalmente en el expositor de Epilogue en la convención de Interop 1990, mostrando un uso alternativo de SNMP, fuera de la gestión de routers, y aprovechando la ocasión para buscar licenciar su implementación SNMP. Como curiosidad, no se podía realizar la demostración de forma legal, ya que tostar pan se consideraba preparación de alimentos, algo permitido solo al sindicato, y para evitar este problema, Epilogue Technology negoció con el sindicato y acordaron tostar repetidamente la misma rebanada de pan para que el público no se la comiera. Además, en la posterior edición de 1991, el hardware de la tostadora evolucionó para añadir un pequeño brazo robótico de LEGO, también controlado por Internet, capaz de tomar una rebanada de pan e insertarla en la tostadora, consiguiendo automatizar el sistema por completo. En la actualidad, John Romkey aún posee esta tostadora de Internet, la cual sigue activa de vez en cuando en Portland, Oregón.

- TODO: Incluir imágenes tostadora (sacarla de url)

Como último ejemplo, también de la misma época de Internet, está la cafetera del laboratorio de informática de la Universidad de Cambridge. Desde los inicios de la computación, la Universidad de Cambridge ha jugado un papel fundamental, contando incluso los primeros ordenadores útiles de uso general, la EDSAC y la EDSAC 2. Además, ha albergado pioneros en ciencias de la computación, incluyendo al creador de las subrutinas, desarrolladores de los primeros procesadores ARM y de varios lenguajes de programación. Esto da lugar a que, por el entorno y por el talento que se encontraba en el campus, era natural que surgieran proyectos de informática. Uno de estos fue la máquina de café conectada a Internet, cuya primera versión fue creada por Quentin Stafford-Fraser y Paul Jardetzky. En 1991, mientras se implantaba la World Wide Web en el Centro Europeo de Investigación Nuclear (CERN), la cual no tenía la popularidad ni la presencia actual, los autores del objeto eran investigadores en el grupo de sistemas del campus, y se encontraban junto a entre 10 y 13 compañeros trabajando la mayoría en una sala del antiguo laboratorio de informática conocida como la Sala Trojan, en cambio otros estaban dispersos por la universidad. Debido a que eran académicos con recursos limitados, solo disponían de una única máquina de café de goteo-filtro para todos, que se encontraba en el pasillo junto a la salida de la Sala Trojan. Aunque la cafeína era esencial para mantener la investigación en informática (de hecho, por la necesidad las jarras de café no duraban mucho), el café de esta máquina era poco inspirador y considerado horrible, pero tolerable si se podía conseguir recién hecho, algo que pocas personas tenían suerte de disfrutar, como Stafford-Fraser que se sentaba al lado de la máquina, mientras otras personas que trabajaban en plantas distintas tenían que recorrer todo el edificio solo para encontrar los posos en el fondo de la jarra, café de muy mala calidad o descubrir que les tocaba rellenar la máquina. Este problema causó cierta angustia en el grupo, por lo que decidieron solucionarlo mediante la tecnología.

En el laboratorio se disponía de cámaras de video económicas excedentes resultado de un proyecto, así como de varios racks de ordenadores simples utilizados para pruebas de redes. Así, en noviembre de 1991, en un lluvioso día, prepararon el sistema XCoffee. Primero fijaron una de las cámaras a un soporte de laboratorio, instalándola para que apuntara a la jarra de la cafetera ubicada en el pasillo. Los cables conectados a la cámara se extendieron por debajo del suelo hasta una tarjeta capturadora de fotogramas de vídeo, instalada en un ordenador Acorn Archimedes situado en el rack de la Sala Trojan. El software de este sistema constaba de dos partes: la parte servidor, escrita por Paul Jardetzky, que capturaba imágenes de la cafetera cada pocos segundos y en diversas resoluciones; y la parte cliente, escrita por Quentin Stafford-Fraser, ejecutable por cualquier usuario conectado a través de Ethernet en la misma red local que el servidor, y que mostraba una imagen obtenida del servidor del tamaño de un icono en una ventana en la esquina de la pantalla. Este sistema operaba utilizando el protocolo X Window System, donde XCoffee empleaba llamadas a procedimientos remotos en modo de transferencia asíncrona a través de Ethernet para obtener las imágenes de la cafetera que se mostraban al usuario, con una resolución de 768x576 píxeles y 8 bits de

profundidad, en escala de grises y actualizada tres veces por minuto, una configuración no molesta ya que la jarra se llenaba despacio y para distinguir el tono del café bastaba con comprobar como de gris estaba la imagen.

Teniendo este sistema, los investigadores del departamento, desde la comodidad de sus oficinas, podían comprobar la cafetera, su luz de encendido, el nivel y la oscuridad del café, y si consideraban oportuno bajaban a por su dosis de cafeína, solucionando finalmente el esfuerzo físico de comprobar la cafetera y evitando la angustia emocional de llegar y encontrarla vacía. Esta solución solo ayudaba a quienes estaban conectados a la red informática interna del laboratorio de Cambridge, mientras que a los que no y no tenían la posibilidad de ejecutar el software de la cámara de la cafetera, como el Dr. Martyn Johnson, el problema les seguía afectando. Influenciado por esta limitación y por la evolución de las páginas web, que pasaron en marzo de 1993 de ser páginas simples de texto que solo se les podía cambiar los colores y la fuente, a adquirir la capacidad de incluir imágenes en el código HTML y mostrarlas, junto a Daniel Gordon, ambos crearon la primera webcam de la historia.

El 22 de noviembre de 1993, Gordon y Johnson, tras estudiar las capacidades de la web y analizar el código del servidor, y considerando que la tarea sería relativamente sencilla, tomaron el sistema de XCoffee y lo desplegaron a la World Wide Web. Johson desarrolló un script, cuya primera versión únicamente ocupaba unas 12 líneas de código, que, junto a algunas modificaciones del software original, hacía que, tras recibir una solicitud HTTP, el servidor web desplegado solicitase el fotograma más reciente capturado por la tarjeta capturadora, y generara y sirviese la página que mostraba esta imagen, en otras palabras, una página con una imagen distinta cada vez. Esta implementación sirvió como prueba para el desarrollo de páginas web dinámicas que, en lugar de contener imágenes estáticas, presentaran imágenes que cambiasen constantemente. Además, ampliaron el acceso para poder acceder no solo desde la red local, sino también desde Internet.

La posibilidad de consultar la cámara directamente desde el navegador web, sin necesidad de ejecutar software ni utilizar protocolos de red especiales, tuvo como efecto secundario, que, aunque no fuese muy útil para personas a varios kilómetros de distancia de la Sala Trojan, todo el mundo podía ver el estado de la cafetera, así ocurrió que cientos de miles de personas han mirado la cafetera, ganando notoriedad internacional y convirtiéndola en la cafetera más famosa del mundo. Esta fama llevó a la instalación de una lámpara apuntando a la jarra para permitir consultar su estado incluso por la noche.

Tras una década en funcionamiento y habiendo pasado por varias cámaras y cafeteras, el grupo de investigación de encontraba frente a un nuevo desafío. Con el software volviéndose cada vez más difícil de mantener y la necesidad de trasladar el laboratorio a las nuevas instalaciones en el Edificio William Gates, se planteaba la urgencia de pasar página. Este software de investigación demostraba no tener la calidad óptima, como suele ocurrir con otro software del mismo estilo, y a su vez la imposibilidad de migrar el sistema a las nuevas máquinas del laboratorio complicaba aún más la situación. Finalmente, tomaron una

decisión que tuvo gran cobertura mediática y no fue apoyada en las protestas nostálgicas de los aficionados a las webcams de todo el mundo: apagar la cámara y la cafetera. El 22 de agosto de 2001, a las 09:54 UTC, se observó en Internet la última imagen capturada por las cámaras que apuntaban a la máquina de café de la Sala Trojan, que muestra los dedos de Daniel Gordon, Martyn Johnson y Quentin Stafford-Fraser pulsando el interruptor de apagado del Acorn Archimedes que había capturado las imágenes durante estos años. En el mismo mes, una de las cuatro o cinco últimas máquinas de café de la Sala Trojan vistas en Internet, la que más tiempo estuvo en servicio, una Krups ProAroma 305, se puso en subasta en Internet a través del portal de subastas y comercio electrónico eBay con el fin de recaudar fondos para las instalaciones de café del nuevo laboratorio. Fue adquirida por 3.350 libras, siendo el mayor postor el sitio web alemán de noticias Der Spiegel. Posteriormente, la cafetera fue restaurada de manera gratuita por los empleados de Krups, y fue encendida de nuevo en la oficina de redacción de la revista. Desde el verano de 2016, la cafetera se encuentra en préstamo permanente en el museo de informática Heinz Nixdorf Museums Forum de Paderborn, Alemania.

https://www.nougir.com/index.php/blog-3/item/13-que-es-iot-o-internet-de-las-cosas-y-sus-aplicaciones https://informationmatters.net/internet-of-things-definitions/ (DOCUMENTO) https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11559&lang=es (LIBRO) https://www.taylorfrancis.com/chapters/oa-edit/10.1201/9781003337584-3/internet-things-cognitive-transformation-technology-research-trends-applications-ovidiu-vermesan-markus-eisenhauer-harald-sundmaeker-patrick-guillemin-martin-serrano-elias-tragos-javier-vali %C3 %B1o-arthur-van-derwees-alex-gluhak-roy-bahr (LIBRO) https://books.google.es/books/about/La_R (DOCUMENTO) https://cdn.ihs.com/www/pdf/IoT_ebook.pdf (DOCUMENTO) https://www.researchgate.net/publication/35254 https://www.microsoft.com/insidetrack/blog/transforming-microsoft-buildings-with-iot-technology-and-indoor-mapping/ (LIBRO) https://www.se.com/us/en/download/document/998-20233517/ (DOCUMENTO) https://www.researchgate.net/publication/320135661_Efficient_IoT-based_Sensor_BIG_Data_Collection-Processing_and_Analysis_in_Smart_Buildings (DOCUMENTO) https://www2.deloitte.com/content/dam/Deloitte/nl/Documents estate/deloitte-nl-fsi-real-estate-smart-buildings-how-iot-technology-aims-to-add-value-for-real-estate-companies.pdf https://innovayaccion.com/blog/aplicando-el-internet-de-las-cosas-a-las-empresas-2 (DOCUMENTO) https://ieeexplore.ieee.org/document/7879243 https://www.intel.com/content/www/us/en/internet-of-things/overview.html https://www.fundacionbankinter.org/ftf-informes/internet-de-las-cosas/ https://www.statista.com/statistic number-of-connected-devices-worldwide/ (VIDEO) https://www.youtube.com/watch?v=RnasX1bFBh8 (LIBRO) https://www.amazon.com/IoT-Fundamentals-Networking-Technologies-Protocols/dp/1587144565 (LIBRO) https://www.ra-ma.es/libro/internet-de-las-cosas_93304/ https://es.wikipedia.org/wiki/Internet_de_las_cosas https://www.eexcellence.es/expertos/kevin-ashton-un-tecnologo-visionario https://www.redhat.com/es/topics/internet-of-things/what-is-iot https://at3w.com/blog/iot-internet-of-things-tecnologia-proteccion-contra-rayo-tomas-tierra/ https://www.itop.es/blog/item/iot-origen-importancia-en-el-presente-y-perspectiva-de-futuro.html https://www.linkedin.com/pulse/el-origen-del-internet-de-las-cosas-iot-comnet-s-a-/

https://www.ui1.es/blog-ui1/historia-y-origen-del-iot
https://blog.avast.com/es/kevin-ashton-named-the-internet-of-things
https://www.datacenterdynamics.com/en/podcasts/zero-downtime/episode-18-the-origin-of-the-internet-of-things-with-peter-lewis/ https://swifttalk.net/2021/10/06/the-concept-of-iot-internet-of-things/
https://www.cs.cmu.edu/~coke/ https://www.cs.cmu.edu/~coke/history_long.txt https://www.ibm.com/docs/es/aix/7.1?topic=protoc
namefinger-protocol https://www.livinginternet.com/i/ia_myths_toast.htm https://en.wikipedia.org/wiki/John_Romkey
https://ieeexplore.ieee.org/document/7786805 Romkey, J. (2017). Toast of the IoT: The 1990 Interop Internet Toaster. IEEE Consumer Electronics Magazine, 6(1), 116–119. doi:10.1109/mce.2016.2614740
https://romkey.com/about/ https://blog.avast.com/the-internets-first-smart-device https://en.wikipedia.org/wiki/Trojan_Room_coffe
https://www.cl.cam.ac.uk/coffee/qsf/coffee.html https://www.cl.cam.ac.uk/coffee/coffee.html https://www.youtube.com/watch?v=u
https://www.bbc.com/news/technology-20439301 https://www.historyofinformation.com/detail.php?id=1507
https://quentinsf.com/coffeepot/metcalfe/ https://www.cl.cam.ac.uk/coffee/qsf/switchoff.html https://owl.museum-digital.de/object/3761

### 2.1.4. Tipos de dispositivos y redes

Como se ha mencionado anteriormente, el IoT se basa en la transmisión de datos entre dispositivos. Estos datos pueden ser información recibida del entorno u órdenes de actuación, que salen o llegan a un dispositivo ubicado en un extremo de la red de conexión y que tiene conectado uno o ambos de los siguientes tipos de objetos:

- Sensores: Miden una propiedad física y la representan digitalmente, siendo útiles para detectar cambios en el entorno. Tienen mayor precisión y capacidad de percepción que los órganos sensoriales humanos, y pueden integrarse en cualquier objeto físico para interpretar su entorno. Actualmente están experimentado una proliferación debido a su disminución en tamaño y coste. Estos se pueden categorizar a partir de ciertos criterios:
  - Activos, si requieren una fuente de energía externa, o pasivos, si no requieren alimentación adicional.
  - Invasivos, si forman parte del entorno que miden, o no invasivos, si miden externamente.
  - Absoluto o relativo, dependiendo de si las lecturas se realizan en función de otro valor.
  - Según su ámbito de aplicación, por ejemplo, el sector agrario o el médico.
  - Según el mecanismo utilizado para medir la información, por ejemplo, óptico o termoeléctrico.
  - Según lo que miden, por ejemplo, posición, fuerza o luz.
- Actuadores: Dispositivos que interpretan una señal eléctrica y desencadenan un efecto físico tras interpretarla, siendo útiles para producir cambios en el entorno. Si un sensor se puede comparar con los órganos sensoriales humanos, un actuador se puede comparar con las acciones que pueden ejecutar las extremidades. Dependiendo de cómo actúen, se pueden categorizar por:

  - Según el tipo de movimiento que producen, por ejemplo, lineal o rotativo.
  - Según su potencia, por ejemplo, alta o baja.
  - Binarios, si solo producen dos estados (por ejemplo, encendido o apagado), o continuos, si tienen un rango de posiciones o estados (por ejemplo, un motor y sus grados).
  - Según su ámbito de aplicación.
  - Según el tipo de energía.

En una red del Internet de las Cosas, tanto sensores como actuadores son el corazón del sistema, ya que permiten interactuar con el mundo físico. Pero a esta lista también se le agregan los objetos inteligentes. Estos están dotados de procesador, sensores y/o actuadores, un método de comunicación y fuente de energía, y son objetos cotidianos que integran todo lo necesario para interactuar con el entorno.

Todos estos dispositivos heterogéneos están interconectados en una red que detecta, mide y actúa en relación con el entorno. Sin embargo, es necesario establecer un diseño previo de la red para poder integrar estas capacidades correctamente en una red sin pérdida de datos, tolerante a fallos y efectiva, según varios criterios:

- Topología y jerarquía de red:
  - Si van a estar todos los dispositivos interconectados entre sí (topología punto a punto).
  - Si un dispositivo central se comunica con el resto (topología en estrella).
  - Si hay agrupaciones de dispositivos y estas se comunican entre sí por un único nodo (topología en malla).
- Distancia entre dispositivos y tipo de red: Cada tecnología de comunicación tiene un alcance máximo y podría no ser compatible con los dispositivos a conectar. Es necesario elegir una que pertenezca a la clasificación de red más adecuada de las siguientes:
  - PAN, Red de Área Personal: Conecta dispositivos cercanos al usuario, con un alcance de hasta máximo 10 metros. Por ejemplo, Bluetooth.
  - LAN, Red de Área Local: Conecta dispositivos dentro de una pequeña zona, como un edificio, con un alcance de hasta 100 metros. Por ejemplo, Wi-Fi 802.11.
  - MAN, Red de Área Metropolitana: Conecta dispositivos dentro de un área grande, como una ciudad o un grupo de edificios, con un alcance máximo de 50 kilómetros.
  - WAN, Red de Área Amplia: Conecta grupos de dispositivos a grandes distancias. Por ejemplo, LoRaWAN.

  Cabe remarcar que las distancias mencionadas no las cumplen todos los protocolos, sino que son valores estimados del alcance del tipo de red.
- Patrones de comunicación: Si los objetos se activan tras un evento determinado o en intervalos periódicos.
- Procesamiento de datos: Si se procesan los datos en el mismo dispositivo que los recopila, si hay

un dispositivo central que agrupa varios, agrega los datos recibidos de estos y los procesa, o si se recopilan y procesan en un servidor o en la nube los datos de todo el sistema.

/*TODO: insertar imagen topologias*/

{ https://www.ibm.com/es-es/topics/internet-of-things (LIBRO, el de los apuntes) “IoT fundamentals: Networking technologies, protocols, and use cases for the internet of things”. David Hanes, Gonzalo Salgueiro, Patrick Grossetete, Jerome Henry, Robert Barton. https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-personal-area-network/ https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-lan/ https://www.cloudflare.com/learning/network-layer/what-is-a-metropolitan-area-network/ https://www.cloudflare.com/learning/network-layer/what-is-a-wan/ https://www.gadae.com/blog/tipos-de-redes-informaticas-segun-su-alcance/ }

### 2.1.5. Usos y el valor de los datos

Hoy en día vivimos en un mundo interconectado, en el cual estamos rodeados de muchos dispositivos, gran parte de ellos interconectados como sistemas IoT. La información es crucial en la actualidad, especialmente a nivel empresarial, ya que permite crear nuevas posibilidades, modelos, interacciones y soluciones únicas, entender y anticiparse a tendencias, identificar nuevas oportunidades, encontrar ventajas competitivas y problemas internos, entre otras aplicaciones beneficiosas para la sociedad. Esta información se obtiene a partir del procesamiento y análisis de colecciones de datos, recopilados de diversas fuentes potenciales, como es el caso del IoT. El Internet de las Cosas se encarga de capturar información del entorno e interactuar con el mismo, siendo un factor fundamental en la transformación digital de los sectores laborales. Sin embargo, disponer de varios dispositivos interconectados durante largos periodos de tiempo en ambientes variados puede generar cantidades masivas de datos, interpretables como Big Data y difíciles de analizar manualmente. Por lo tanto, un método muy común es incorporar la inteligencia artificial al ecosistema IoT, utilizándola para analizar y extraer información valiosa de los datos. Las técnicas de inteligencia artificial suelen ser eficientes y requieren poca o nula interacción humana. Entre los beneficios que ofrecen se encuentran la identificación de patrones, la realización de predicciones y la toma de decisiones automatizadas basadas en los datos.

El uso de IoT es aplicable a una gran variedad de entornos y puede traer beneficios de manera general, como monitorear la localización, el consumo y los errores que puedan ocurrir en el equipamiento (por ejemplo, el médico o el utilizado en obras), administrar el inventario fácilmente (el de un supermercado, una farmacia) y reducir y optimizar el trabajo del personal. De manera más concreta, los beneficios pueden tener un gran impacto en los siguientes ámbitos y sectores:

- Sanidad: La implantación de sistemas IoT en el sector de la salud puede ayudar a reducir costes de atención y mejorar la calidad de vida de los pacientes. Se pueden utilizar sensores para monitorizar

los signos vitales (como la frecuencia cardíaca, la presión arterial, la saturación de oxígeno o la temperatura) y dificultades físicas del paciente sin necesidad de ingresarlos en un hospital, evitando la sobrepoblación y la falta de camas en los hospitales. Junto a esta monitorización remota, se agrega la recopilación y análisis de los mismos datos, útiles para detectar patrones e identificar posibles problemas de salud, y ofrecer de forma instantánea un amplio contexto a los especialistas sanitarios para que puedan tomar decisiones y hacer recomendaciones de forma adecuada sin necesidad de invertir tiempo en buscar datos del paciente ni tomarle notas. Además, se pueden utilizar dispositivos fáciles de manejar por los pacientes para recibir avisos textuales, de audio o video por parte de los doctores, y en cuanto a las medicaciones se pueden integrar etiquetas inteligentes para poder monitorear la localización, obtener información sobre la expiración, comprobar si el medicamento es auténtico y las dosis recomendadas.

- Agricultura y ganadería: No solo es posible aplicar el Internet de las Cosas a zonas urbanas, sino también a zonas donde la conectividad es más compleja de implementar, por lo que normalmente en este sector se suelen utilizar dispositivos de bajo consumo conectados a baterías o a placas solares. La tecnología puede ayudar a analizar las condiciones del suelo, como la temperatura, humedad y nutrientes mediante el uso de sensores, y actuar en consecuencia de forma manual o automatizada, por ejemplo, regando de manera controlada, y así reduciendo el desperdicio de recursos. Es útil para analizar y prever patrones climáticos y determinar cuándo sería adecuado realizar la cosecha, así como para monitorizar el crecimiento de los cultivos, y se puede utilizar para determinar la necesidad de utilizar fertilizantes o insecticidas. También se pueden utilizar sensores para localizar al ganado, registrar información, analizar constantes vitales, monitorizar la salud o identificarlo, y para mantener una alta calidad en los productos provenientes de animales, por ejemplo, mediante sensores se podría monitorizar la temperatura y el movimiento de la leche recolectada en un tanque.

- Transporte, conducción y logística: El Internet de las Cosas también es compatible con vehículos en movimiento, siendo capaz de ofrecer un transporte inteligente. Un vehículo repleto de sensores puede tener la capacidad de ofrecer conducción asistida o automatizada mediante reconocimiento de señales, frenado en caso de emergencia y detección de peatones, evitando accidentes. También pueden informar sobre el estado del vehículo y monitorizar su rendimiento, para prever problemas y darle un mejor mantenimiento. También puede tener la capacidad de reducir el tiempo de conducción y mejorar la eficiencia del combustible al calcular rutas óptimas y asistir en la búsqueda de aparcamiento. Lo mencionado anteriormente no solo se puede aplicar a la conducción privada, sino también al transporte público y a los envíos de mercancías, mejorando los tiempos de entrega y agregando la capacidad de rastreo de los envíos mediante etiquetas inteligentes y sensores. En el caso de transportar productos perecederos, se puede mejorar la calidad de los mismos al monitorizar con sensores y mantener la temperatura y humedad del remolque que los transporta.

- Fabricación: Monitorizar los sitios donde se fabrican los productos mejora la eficiencia en la fabricación. Pueden implementarse sensores en las máquinas para monitorizar su rendimiento y detectar fallos, así como en los centros de fabricación para controlar la temperatura y humedad, asegurando condiciones óptimas para la producción de productos sensibles. Gracias a todos los datos recopilados, es posible optimizar procesos de producción e identificar áreas donde sería necesario más personal, sin necesidad de estar presencialmente en el centro de fabricación. Además, mediante la automatización, se podrían realizar mejores controles de calidad a los productos terminados y optimizar otros procesos de producción.

- Comercio: IoT sirve para mejorar tanto la fabricación como la venta, trayendo beneficio a todo el ciclo de vida de un producto. Esta mejora influye incluso en la propia decisión que el cliente debe tomar respecto a un producto, ya que se pueden crear ofertas, optimizar la colocación de los productos y mejorar la experiencia del cliente mediante sensores en la tienda, que rastreen el tráfico dentro de la misma y analicen el comportamiento de los clientes. Los clientes también se ven beneficiados, ya que al tener sensores en los productos o en las estanterías, pueden guiarse mediante algún dispositivo específico al producto que desean, y con terminales de autopago no necesitan esperar largas colas, haciendo que las compras sean eficientes. Además, los productos de las tiendas pueden ser rastreados y analizar toda su trazabilidad, desde el primer momento en el que llegaron al almacén hasta cuando salieron por la puerta a modo de venta.

- Ciudades y edificios inteligentes: El Internet de las Cosas puede traer facilidades incluso a la vida cotidiana, ya sea en un hogar, en la oficina del trabajo o en una ciudad por completo. Mediante sensores y cámaras, puede ser posible tener una mejor vigilancia en las calles, haciendo más eficientes las acciones de los equipos de emergencia, además de analizar infracciones, accidentes y monitorear el tráfico, con la intención de evitar congestiones y realizar modificaciones en las calles y carreteras. Además, en una ciudad inteligente, se podrían tener aparcamientos repletos de sensores para hacer eficiente la búsqueda de aparcamiento para los usuarios y poder analizar el uso de las plazas, útil por si fuera necesario añadir más. Siendo la contaminación uno de los factores más influyentes en la salud de las personas, en una ciudad inteligente se podría mejorar, instalando sensores para la contaminación acústica y del aire, o incluso en las plantas de tratamiento del agua. Incluso en zonas de mayor atracción turística es posible instalar sensores que guíen a un sitio u objeto en específico, para, por ejemplo, facilitar la llegada a un asiento en un estadio o encontrar una pieza de arte en un museo. Todos los datos recopilados de varios sensores pueden dar lugar a implantar soluciones inteligentes en calles y edificios para facilitar la vida a sus ciudadanos. De manera más específica, en un edificio o sitio cerrado se podrían equipar dispositivos que detecten la presencia y así tomar decisiones en cuanto a la intensidad de la iluminación o el control de la calefacción, permitiendo además controlarlos remotamente. También se pueden aplicar para controlar, gestionar y automatizar

entornos habitables, por ejemplo, la habitación de un niño pequeño o la casa de una persona anciana, tipos de personas más vulnerables a riesgos.

El Internet de las Cosas no solo se reduce a estos sectores, es prácticamente aplicable a todos, como la educación, la construcción o el turismo, y a cualquier fase, desde el diseño hasta ofrecer y consumir el servicio o producto, gracias a la popularización de esta tecnología y al aumento de dispositivos que aparecen a nuestro alrededor.

TODO: { https://www.campusbigdata.com/big-data-blog/item/101-relacion-iot-con-big-data https://www.ibm.com/es-es/topics/internet-of-things https://www.nougir.com/index.php/blog-3/item/13-que-es-iot-o-internet-de-las-cosas-y-sus-aplicaciones https://innovayaccion.com/blog/aplicando-el-internet-de-las-cosas-a-las-empresas-2 https://cdn.ihs.com/www/pdf/IoT_ebook.pdf https://www2.deloitte.com/content/dam/Deloitte/nl/Documents/real-estate/deloitte-nl-fsi-real-estate-smart-buildings-how-iot-technology-aims-to-add-value-for-real-estate-companies.pdf https://www.researchgate.net/publication/325373920_Internet_of_things_IoT_a_survey_on_architecture_enablin
}

## 2.2. Protocolos de mensajería usados en IoT

El uso más común de IoT es desplegar una arquitectura compuesta por varios dispositivos IoT. En mayor parte, estos dispositivos se designarán simplemente como dispositivos IoT, ya sean sensores o actuadores, mientras que habrá pocos dispositivos (al menos uno) con el rol de centro de mensajería. Dependiendo del uso que se le dé a la arquitectura, estos objetos se comunicarán, y dependiendo de quién sea el emisor y el receptor, dan lugar a estos escenarios:

- Un mensaje ha llegado al centro de mensajería IoT. Este mensaje es procesado y se actúa en consecuencia, por ejemplo, enviando la información necesaria a ciertos dispositivos IoT o almacenándola a una base de datos.
- Un dispositivo IoT ha generado datos. Estos datos son procesados y luego enviados al centro de mensajería.
- Un dispositivo IoT ha recibido datos del centro de mensajería. Estos datos son procesados y se actúa en consecuencia.

Estos dispositivos, por sí solos, no son capaces de intercambiar esos grandes volúmenes de datos que tratan. Por ello, tras escoger una tecnología para conectar los dispositivos entre sí, es esencial en el desarrollo de aplicaciones IoT disponer de un protocolo de mensajería.

Un protocolo de comunicación permite que los dispositivos se comuniquen y transmitan mensajes entre los dispositivos IoT y el centro de mensajería. Además, proporciona cierta fiabilidad a la comunicación, ya que permite que los mensajes lleguen y sus datos sean entendidos y procesados correctamente. Esta comunicación ocurre sobre TCP, o incluso sobre abstracciones de mayor nivel como HTTPS.

La elección del protocolo se basa en cómo se adecua al escenario en el que se quiere implementar, considerando requisitos a tener en cuenta como la ubicación, las limitaciones físicas, el consumo, la batería y el coste. Por lo general, no cualquier protocolo de comunicación es apropiado. Los protocolos que se mencionan en este apartado se adecuan a la mayoría de escenarios IoT debido a su rapidez y su fácil implementación, y es posible escoger aquel que se adapte mejor a los requisitos.

### 2.2.1. MQTT

El protocolo Message Queue Telemetry Transport es uno de los más populares en el ámbito del Internet de las Cosas. Diseñado para ser ligero y adecuado para redes con ancho de banda limitado y dispositivos con pocos recursos, este estándar del comité técnico OASIS permite el transporte bidireccional de mensajes con datos entre múltiples dispositivos.

MQTT utiliza el patrón de comunicación publicación-suscripción. En este patrón, los publicadores categorizan los mensajes, y los suscriptores recibirán mensajes de las categorías de su interés, a diferencia de la comunicación tradicional en la que los publicadores envían los mensajes directamente a los suscriptores. En el caso de MQTT, el patrón está basado en temas o topics, siendo posible que los suscriptores se interesen por uno o varios. Este patrón permite utilizar una red única para transmitir datos entre dispositivos y servidores, habilitando el control remoto de una gran cantidad de dispositivos a través de Internet.

En una red MQTT, se definen dos roles principales: el broker o intermediario de mensajes y los clientes. El broker MQTT es un servidor comparable a una oficina de correos, que recibe todos los mensajes publicados por los clientes y los dirige a los clientes de destino apropiados. Por otra parte, un cliente es cualquier dispositivo conectado al broker a través de una red, y puede producir y recibir datos al publicar y suscribirse respectivamente. Este mecanismo es útil para compartir datos, controlar y gestionar dispositivos. Por ejemplo, un dispositivo cliente puede publicar datos de sensores y además recibir información de configuración o comandos de control. La enrutación de mensajes que realizada por el broker proporciona transparencia y desacoplamiento en el espacio, ya que el publicador no necesita conocer ni la cantidad ni las direcciones de los suscriptores, y los suscriptores no necesitan conocer al publicador, ambos interactúan únicamente con el broker.

Los mensajes están organizados en una jerarquía de temas o topics. Al publicar un mensaje, se publica en un tema específico, y en el caso de querer publicar a varios se deben realizar varias publicaciones. En cambio, un suscriptor puede suscribirse a un tema específico o a varios simultáneamente y recibirá una copia de todos los mensajes compatibles con los temas suscritos. La manera de indicar varios temas es mediante el uso de los siguientes caracteres comodín:

- Comodín de un nivel ‘+’: coincide con un nivel de tema y puede utilizarse más de una vez en la especificación del tema.

- Comodín de varios niveles '#': coincide con cualquier número de niveles y debe ser el último carácter en la especificación del tema.

Como ejemplo de uso de los temas, cuando se publica un mensaje en el tema "edificioA/sensor1/temperatura", el broker enviará una copia del mensaje los clientes suscritos a los temas "edificioA/sensor1/temperatura", "edificioA/+/temperatura" y "edificioA/#", pero no a un cliente suscrito a "edificioB" o a "edificioA/+/humedad".

La transmisión de mensajes se realiza de forma asíncrona, sin detener la ejecución de ambos componentes a la hora de publicar o recibir, y se puede realizar una comunicación uno a muchos (un publicador y varios suscriptores), muchos a uno (varios publicadores y un suscriptor), uno a uno (un publicador y un suscriptor, menos común) y muchos a muchos (varios publicadores y varios suscriptores).

En el caso de que el broker reciba una publicación de un tema en el cual no hay nadie suscrito, el broker por defecto descarta el mensaje. Es posible activar la retención de mensajes configurando un campo en el mensaje para evitar esto, consiguiendo así que el broker almacene el último mensaje retenido de cada tema y lo distribuya inmediatamente a cualquier nuevo cliente suscrito, permitiendo así que el suscriptor reciba el valor más reciente en lugar de esperar a una nueva publicación, y además añadiendo soporte a una comunicación desacoplada en el tiempo, donde publicadores y suscriptores no necesitan estar conectados simultáneamente.

El protocolo soporta un mecanismo de limpieza de sesión. Por defecto, un cliente tras desconectarse y volverse a conectar no recibe los mensajes publicados durante su desconexión y el broker olvida las suscripciones del mismo cliente. Pero al desactivar dicho mecanismo, el broker mantiene tanto las relaciones de suscripción como los mensajes offline, enviándolos al cliente al momento de reconectarse, lo cual es útil para dispositivos que se conectan y desconectan constantemente, común en redes IoT. Además, MQTT enfrenta la inestabilidad de la red con un mecanismo Keep Alive que, al transcurrir un prolongado periodo sin interacción, ocurre un ping entre el cliente y el broker para evitar la desconexión. Si el ping falla y se identifica el cliente como desconectado, aplicará un mecanismo Last Will, que publica un último mensaje a un tema específico debido a una desconexión anormal, en el caso de estar configurado.

MQTT dispone de 14 tipos de mensajes diferentes, la mayoría utilizados para mecanismos internos y flujos de mensajes:

- CONNECT: establece una conexión con el broker, y si está configurado, se debe proporcionar un usuario y contraseña.
- DISCONNECT: finaliza una sesión MQTT enviando este mensaje para cerrar la conexión. Esta desconexión se denomina "graceful shutdown" o "apagado elegante", porque está la posibilidad de conectarse al broker con la misma sesión y reanudar el progreso.
- PINGREQ/PINGRESP: una operación de ping utilizada para saber si está viva la conexión y mante-

nerla.

- PUBLISH: contiene un mensaje para publicarlo en un tema específico.
- SUBSCRIBE: utilizado por los clientes para suscribirse a un tema específico y recibir las actualizaciones de este.
- UNSUBSCRIBE: mensaje que utiliza un cliente para indicar la pérdida de interés y anular la suscripción a un tema específico
- LWT: este mensaje "last will and testament" (última voluntad y testamento) se configura en un cliente para publicarse automáticamente si ocurre una desconexión inesperada. El broker mantiene un temporizador, y si comprueba que recientemente el cliente no ha publicado ni ha mandado un PINGREQ, se publica el mensaje LWT especificado notificando así a los suscriptores.

El diseño de MQTT se basa en la simplicidad y en minimizar el ancho de banda, dejando la interpretación de los mensajes en manos del desarrollador. Los mensajes transmitidos a través de la red tienen la posibilidad de configurar el QoS o calidad de servicio por cada tema, asociados con distintas garantías de entrega y cómo se entregan los mensajes. Aunque MQTT depende de TCP, el cual tiene su propia garantía de entrega, históricamente los niveles QoS eran necesarios para evitar la pérdida de datos en redes antiguas y poco fiables, una preocupación válida para las redes móviles actuales. Estos son los siguientes tipos de QoS:

- QoS 0, a lo sumo una vez: los mensajes se envían y no se tiene en cuenta si llegan o no. Está la posibilidad de la pérdida de mensajes y no se hacen retransmisiones.
- QoS 1, al menos una vez: el receptor recibe el mensaje por lo menos una vez. El receptor debe enviar un acuse de recibo al emisor en cuanto reciba el mensaje, y si este ACK nunca llega (ya sea debido a que el mensaje nunca llegó o que el ACK se perdió), el emisor retransmitirá el mensaje, pudiendo producirse mensajes duplicados.
- QoS 2, exactamente una vez: asegura que el mensaje llegue exactamente una vez, manejado mediante la sobrecarga en la comunicación y el envío de una serie de acuses de recibo, y es la mejor opción cuando no se acepta ni la pérdida ni la duplicidad de mensajes.

La transmisión de datos se realiza principalmente sobre la capa TCP/IP, pero existe la posibilidad de operar encima de otros protocolos de red que ofrezcan conexiones ordenadas, sin pérdidas y bidireccionales, y se transmiten en un tamaño reducido de paquetes de datos, estructurado por los siguientes campos:

- Cabecera fija, en la que se especifica el tipo de mensaje, si el mensaje es un duplicado, el QoS, si es un mensaje que retener y el tamaño del paquete.
- Cabecera variable, no siempre presente en los mensajes, y puede transportar información adicional de control.
- Payload o carga útil.

Por defecto, este protocolo envía credenciales y mensajes en texto plano sin medidas de seguridad, pero admite utilizar conexiones TLS/SSL protegidas por certificado, nombre de usuario y contraseña para cifrar y proteger la información transferida contra la intercepción, modificación o falsificación. Además, un broker MQTT tiene soporte para conectar dispositivos IoT a escala masiva, un factor tenido en cuenta a la hora de su diseño y que resulta en una alta capacidad de concurrencia, rendimiento y escalabilidad, características útiles en una red IoT. Implementaciones como EMQX y HiveMQ han alcanzado hitos notables, con 100 y 200 millones de conexiones respectivamente, y un pico de 1 millón de mensajes gestionados por segundo. A esta escalabilidad se le puede sumar la capacidad de implementar múltiples brokers, para así compartir la carga de clientes y tratar la redundancia y la copia de seguridad de los mensajes en caso de fallo.

### 2.2.2. AMQP

Advanced Message Queuing Protocol es un protocolo binario avanzado que opera sobre la capa de aplicación, cuyo estándar abierto permite desarrollar mensajería y patrones de comunicación entre dispositivos. Facilita la comunicación entre servicios definiendo el formato de los datos enviados a través de la red como un flujo de bytes, así como la creación de mensajes, el encolamiento y enrutamiento de los mensajes producidos, y la manera de entregarlos a los consumidores.

Este protocolo se basa en el concepto de publicar y consumir de colas de mensajes a través de una conexión fiable, persistente y orientada al envío de flujos de datos. Además, es compatible con el envío de múltiples flujos de datos simultáneos mediante múltiples canales en una única conexión. Permite hacer uso de estas colas de mensajes mediante distintos modelos de mensajería, como la entrega directa punto a punto y el modelo publicación-suscripción, y, pese a no ser un protocolo diseñado originalmente para su uso en el Internet de las Cosas, funciona muy bien en este ámbito y en la gran variedad de escenarios de comunicación y mensajería posibles.

En AMQP se definen las dos siguientes entidades principales que interactúan entre sí:

- Cliente: del tipo suscriptor o publicador (o consumidor y productor, respectivamente), se conecta a un broker a través de credenciales y, en caso de estar autorizado, puede recibir o publicar mensajes.
- Broker: servidor de mensajería al que los clientes se conectan y que se encarga de distribuir los mensajes. Internamente, posee exchanges o intercambiadores, donde se conectan los productores de mensajes, y colas, vinculadas a los exchanges dependiendo de varios criterios y a las que se conectan los consumidores para extraer los mensajes producidos.

De manera sencilla, se puede resumir el funcionamiento de este protocolo como un modelo en el que los mensajes son publicados y enviados a exchanges, los cuales enrutan los mensajes a las colas apropiadas según reglas o bindings, y los consumidores reciben los mensajes a través de las mismas colas.

Los exchages, además de recibir mensajes de los productores, se encargan de enviar los datos a las colas

apropiadas, ya sean a una o a varias dependiendo del exchange y la clave de enrutamiento o routing key con la que se publica el mensaje. A este funcionamiento se le puede asociar la analogía del funcionamiento del envío de emails, ya que estos se envían a direcciones "routing key@exchange", siendo la clave de enrutamiento la dirección de correo y el exchange el servidor. En AMQP existen cuatro tipos de exchanges:

- Topic, que permite una comunicación publicación-suscripción: los mensajes se enrutan a las colas adecuadas en función de la clave de enrutamiento y el patrón de vinculación al exchange.
- Direct, que permite una comunicación punto a punto: los mensajes se reciben en un exchange con una clave de enrutamiento específica y son enrutados directamente a una cola creada con la misma clave. Es posible que una cola tenga varias claves, conectándose a varios exchange simultáneamente, al igual que varias colas pueden compartir una misma clave de enrutamiento, enviando los mensajes a varios destinatarios.
- Fanout, que permite una comunicación parecida a broadcast: los mensajes son enrutados a todas las colas vinculadas al exchange.
- Cabeceras, que permite una comunicación publicación-suscripción: los mensajes se encaminan a las colas dependiendo de la coincidencia de las cabeceras de los mensajes.

Las colas son fragmentos de memoria creados por el cliente suscriptor identificadas unívocamente mediante un nombre, definido previamente por el cliente o automáticamente por el broker. Intrínsicamente, AMQP garantiza la recepción y procesamiento de mensajes, ya que dispone de un mecanismo de ACK o acuse de recibo que permite confirmar la recepción y procesamiento del mensaje. En el caso de no recibir el ACK de un mensaje por parte de un consumidor, ya que perdió la conexión o no se procesó correctamente, el broker encola de nuevo el mensaje para reintentar las operaciones. Junto a este mecanismo, también está la posibilidad de rechazar mensajes mediante un mensaje NACK o acuse de recibo negativo, útil para indicar que no se ha procesado bien el mensaje y que el broker puede borrar el mensaje de esa cola. Además, las colas admiten persistencia, para mantener la existencia de la cola incluso luego de que ocurra un reinicio en el broker y que el suscriptor no deba conectarse nuevamente.

Para relacionar colas y exchanges se utilizan bindings o vinculaciones para especificar el flujo de los mensajes. Tras crear el exchange y la cola, se indica al broker la vinculación de ambas mediante un binding, especificando una clave de enrutamiento que, según el tipo de exchange, enruta adecuadamente las colas. El exchange entregará como máximo una copia de mensaje a una cola si corresponden las propiedades del mensaje con las propiedades del binding. Con los bindings es posible vincular varias colas a un mismo exchange, al igual que una cola a varios exchanges.

Con los tres conceptos mencionados previamente, exchange, binding y cola, se produce una transparencia en la comunicación entre los productores y los consumidores, ya que ambos están aislados entre sí y desconocen su existencia y ubicación.

Este protocolo especifica el comportamiento del servidor y de los clientes conectados, junto al formato de los datos enviados a través de la red, por lo que al crear implementaciones se deben ajustar a este mismo formato de datos, permitiendo interoperar independientemente del lenguaje o del proveedor. Los mensajes se transmiten sobre la capa TCP, y disponen de propiedades como las siguientes:

- Clave de enrutamiento.
- Modo de entrega, si es un mensaje persistente.
- Prioridad del mensaje en un rango entre 0 y 9.
- Vencimiento, indicado en milisegundos, es el tiempo que el broker debe esperar antes de descartar el mensaje si no fue consumido.
- Contador de intentos de entrega.
- Anotaciones de mensaje.
- Propiedades, como el ID del mensaje, usuario y tiempo de creación.
- Payload o carga útil.
- Footer, que contiene detalles del mensaje como hashes o firmas.

AMQP es extenso en cuanto a funciones, soportando seguridad mediante el cifrado extremo a extremo, multitud de propiedades de mensajes y modos de entrega, fiabilidad de entrega, persistencia de mensajes, enrutado complejo y estrategias de distribución de mensajes, escalabilidad y definición de topologías, entre otros. Sin embargo, pese a su idoneidad para sistemas distribuidos, es un protocolo que requiere un alto consumo de recursos como energía y memoria.

### 2.2.3. XMPP

Otra manera de comunicar varios dispositivos IoT se logra mediante el Extensible Messaging and Presence Protocol, anteriormente conocido como Jabber. Este protocolo se basa en la transmisión de datos estructurados en formato XML dentro de una red de arquitectura cliente-servidor, en la cual los dispositivos están identificados por un Jabber ID, cuyo formato es similar al de una dirección de correo electrónico (por ejemplo, “abc@example.com”). En esta red, el cliente establece una conexión TCP/IP con el servidor, la cual permanece abierta. Posteriormente, el cliente se autentica con el servidor, y tras una autenticación exitosa, se habilita la posibilidad de enviar y recibir mensajes.

Una característica notable de XMPP es que no existe un servidor central que gestione todas las comunicaciones; el intercambio de mensajes entre clientes y servidores está descentralizado, permitiendo a los usuarios y organizaciones operar su propio servidor XMPP. Esto no restringe a los usuarios a conectarse únicamente con otros usuarios en el mismo servidor, ya que, al ser un estándar abierto regido por la Internet Engineering Task Force, los desarrolladores disponen de un protocolo bien documentado y fiable. De este modo, es posible interoperar entre diferentes implementaciones de XMPP a través de Internet, independientemente del proveedor. En el caso de querer comunicarse con otro servidor, ambos servidores XMPP

se intercambian la información necesaria, habilitando un modelo federado.

Este protocolo está diseñado para ofrecer mensajería instantánea o casi en tiempo real a través de la red, sin importar la distancia entre los dispositivos, uno de los problemas más comunes en IoT. Además, permite obtener información de presencia sobre los usuarios conectados y mantener una lista de contactos para cada usuario. XMPP también admite extensibilidad, permitiendo a los desarrolladores añadir características y funcionalidades personalizadas, ofreciendo más allá de la mensajería tradicional y adaptando XMPP a necesidades específicas de aplicaciones, como la transmisión de señales VoIP, video, ficheros, chat grupal, conferencias multiusuario, suscripción de presencia para conocer cuándo alguien está conectado a la red, y comunicación publicación-suscripción para recibir actualizaciones sobre temas específicos de interés.

En los mensajes XMPP se utiliza el formato stanzas XML para estructurar y transportar los datos. Existen 3 tipos principales de stanzas:

- Stanza de mensaje (`message`): utilizado para enviar mensajes instantáneos entre clientes. Contiene los campos remitente, destinatario, cuerpo del mensaje y otros metadatos opcionales. Después de recibir el mensaje, el servidor utiliza el campo de destinatario para enrutar el propio mensaje. Ejemplo de uso de esta stanza:

```
<message from='abc@example.com'
    to='xyz@example.com'
    type='chat'>
    <body>Hemos tenido una velada encantadora.</body>
</message>
```

- Stanza de presencia (`presence`): permite a las entidades conocer el estado y la disponibilidad online/offline de otros cliente. También puede transportar información adicional, como la actividad del cliente o su ubicación. Cuando un cliente se conecta o desconecta del servidor, envía una stanza de presencia para notificar a otros clientes de su lista de contactos. Ejemplo de uso de esta stanza:

```
<presence from="abc@example.com">
    <show>away</show>
    <status>Paro para comer.</status>
    <priority>5</priority>
</presence>
```

- Stanza de IQ o info/query (`iq`): se usa para consultar al servidor, gestionar suscripciones o intercambiar datos estructurados entre clientes y servidores. Funciona de manera similar a los métodos HTTP GET y POST, siguiendo un patrón de petición-respuesta, en el cual un cliente envía una petición al

servidor y este responde con la información solicitada o con una confirmación. Ejemplo de uso de esta stanza:

```
<iq to="user@example.com" type="get" id="314">
   <query xmlns="http://jabber.org/protocol/disco#items" />
</iq>
```

El protocolo XMPP es altamente escalable debido a su capacidad de manejar multitud de conexiones y mensajes simultáneos. Además, al ser descentralizado, permite implementar fácilmente más servidores para gestionar el aumento de usuarios y altos picos de uso. En cuanto a seguridad, XMPP es compatible con cifrado de extremo a extremo mediante TLS o SSL, garantizando así la confidencialidad de los mensajes. Por último, cuenta con una amplia comunidad de usuarios, diversas implementaciones y guías que facilitan a los desarrolladores la creación de aplicaciones que integren este protocolo.

### 2.2.4. DDS

El Data Distribution Service es un estándar de middleware y API máquina-máquina que facilita la comunicación y el intercambio de datos. Fue desarrollado por el Object Management Group con el fin de responder a las necesidades específicas de aplicaciones que requieren intercambios de datos fiables y de alto rendimiento en sistemas distribuidos en tiempo real, sin dejar de lado la eficiencia. Su arquitectura se basa en un modelo de publicación-suscripción sin servidor, sino que los dispositivos se conectan entre sí, y donde los datos son publicados en un dominio y los suscriptores se conectan a este para recibir la información que les interesa.

Este middleware es la capa de software que se encuentra entre el sistema operativo y las aplicaciones, abstrayendo la comunicación entre ambos y, por tanto, los detalles del tranporte de red y de los datos a bajo nivel. Permite que los distintos componentes de un sistema compartan y comuniquen datos, proporcionando el formato, el protocolo de transporte, la fiabilidad, la calidad de sevicio y la seguridad, y simplificando así el desarrollo. En el caso de DDS, se centra completamente en los datos, asegurando un buen transporte e incluyendo información contextual de los mismos, lo que lo hace ideal para el Internet de las Cosas aplicado en entornos industriales.

DDS funciona con el principal concepto de espacio de datos global, un almacén de datos que a ojos del programador parece una memoria local accedida mediante APIs. Sin embargo, este espacio es solo una ilusión, ya que no existe un lugar donde residan todos los dato, es un concepto que se refiere a una colección de distintos almacenes locales en cada nodo por los cuales se reparten los datos. El programador piensa que está enviando mensajes a un almacén global, pero en realidad DDS envía mensajes a los almacenes locales apropiados.

Este espacio global, a pesar de ser tan característico, no pierde compatibilidad, ya que el middleware es in-

dependiente del lenguaje de programación y la plataforma, posibilitando la interoperabilidad entre distintos sistemas y aplicaciones y una implementación que se adapte a las necesidades sin afectar a las comunicaciones entre dispositivos. Tampoco pierde efectividad, ya que su velocidad, baja latencia, baja sobrecarga en la comunicación, optimización del transporte y capacidad de transmitir millones de mensajes a multitud de receptores instantáneamente, lo convierten en una tecnología ideal para sistemas de alto rendimiento en tiempo real.

Al ser descentralizado, es decir, al no requerir un nodo central, el servicio DDS es mucho más eficiente y eficaz, ya que los mensajes no deben atravesar un dispositivo intermediario, ejecutando la comunicación punto a punto, y resultando en una mayor velocidad. Además, dispone de un servicio de descubrimiento dinámico, haciendo que las aplicaciones DDS sean extensibles y escalables. La aplicación no necesita conocer ni configurar los puntos finales de los dispositivos para la comunincación, ya que estos se descubren automáticamente en ejecución, y es capaz de identificar si el punto final se utiliza para publicar datos, para suscribirse o para ambos, el tipo de dato publicado o suscrito, y las características específicas de la comunicación. En cuanto a los participantes, admite tanto a los localizados en una misma máquina como a los distribuidos en distintas máquinas, y es compatible con miles de dispositivos, manteniendo una alta velocidad.

DDS soporta principalmente el modelo de publicación-suscripción, intercambiando información basada en un tema o topic identificado por su nombre. En este modelo, cualquier nodo conectado puede publicar mensajes con el tema especificado o suscribirse a un tema, y DDS se encarga de que los datos se entreguen a los suscriptores correctos en el momento adecuado mediante comunicación peer-to-peer. Al suscribirse, es posible especificar filtros de tiempo y subconjuntos de datos para obtener solo los requeridos, y tiene la capacidad de detectar cambios para que los suscriptores reciban actualizaciones adecuadas de los datos. Por otro lado, al publicar, es DDS quien gestiona la complejidad de la transmisión y se encarga de almacenar los datos de manera segura. También ofrece compatibilidad con RPC o llamadas a procedimientos remotos.

El middleware es independiente del transporte, y puede funcionar sobre UDP, TCP, memoria compartida, entre otros. Entre las características opcionales que ofrece, como el filtrado de grandes datos, se encuentra la gestión de calidad de servicio o QoS, donde se pueden especificar requisitos de rendimiento y confiabilidad, como la latencia, el ancho de banda, la prioridad, la disponibilidad de los datos, el uso de recursos y la sincronización. Además, incluye mecanismos de seguridad que proporcionan autenticación, encriptación, confidencialidad, control de acceso e integridad en la distribución de información.

### 2.2.5. CoAP

El Constrained Application Control es un protocolo de la capa de aplicación que permite a dispositivos con recursos limitados, como los que se encuentran en una red IoT, comunicarse entre sí. Funciona en un marco cliente-servidor, en el cual el cliente realiza una solicitud a un punto de comunicación del dispositivo

servidor, y este responde, permitiendo la interoperabilidad entre los dispositivos uno a uno.

Este protocolo opera sobre el protocolo de transporte UDP, que, a diferencia de TCP, no requiere que los dispositivos establezcan una conexión de datos previa al envío de datos. Esto trae tanto consecuencias positivas como negativas. La consecuencia negativa radica en la poca fiabilidad en la comunicación de base, ya que el protocolo UDP no garantiza la entrega de los mensajes, sino que esta garantía se gestiona desde la implementación de CoAP. Es posible establecer acuses de recibo (ACK), de manera que, por cada mensaje enviado, el dispositivo espera un acuse de recibo y, en caso de no recibirlo en un tiempo determinado, el mensaje se retransmite. La consecuencia positiva del uso de UDP es la posibilidad de funcionar en redes con pérdidas o inestables, adecuado para redes IoT, ya que suelen operar en entornos de red difíciles, y la rapidez en la comunicación, pues no requiere una conexión de datos previa, enviando directamente el mensaje.

Esta comunicación utiliza una arquitectura RESTful, en la cual los datos y las funcionalidades se consideran recursos a los que se accede mediante una interfaz estándar y uniforme. Estos recursos se acceden y se interactúa con ellos mediante métodos HTTP estándar (GET, POST, PUT, DELETE, que realizan las funciones “obtener”, “crear”, “actualizar” y “eliminar” recursos, respectivamente), permitiendo una interoperabilidad sencilla entre distintos tipos de dispositivos y facilitando a los desarrolladores la creación de aplicaciones que usan protocolo. No es necesario que los recursos de la red sean conocidos por el dispositivo que vaya a utilizarlos, ya que CoAP implementa un mecanismo de descubrimiento integrado, útil en redes IoT en las que los dispositivos constantemente se conectan y desconectan. Esta función de descubrimiento trata de consultar un recurso conocido “núcleo” en la red, el cual provee la lista de los recursos de los dispositivos en la red. Es decir, si un dispositivo IoT quiere interactuar con los recursos de otro, puede consultar al núcleo y comprobar qué recursos hay disponibles y cómo puede interactuar con ellos.

El intercambio de mensajes CoAP entre dispositivos es asíncrono, lo que significa que un dispositivo puede enviar una solicitud a otro y continuar ejecutando otras tareas mientras que la respuesta puede recibirla en cualquier momento. Esto se logra mediante un id en los mensajes, permitiendo al dispositivo relacionar peticiones con respuestas, asegurando un alto nivel de fiabilidad en el intercambio de mensajes. Esta comunicación asíncrona es crucial en redes IoT, ya que los dispositivos pueden no estar siempre conectados o disponibles para responder en el momento de la solicitud.

CoAP se basa en el intercambio de mensajes compactos codificados en un formato binario simple. El tamaño de estos mensajes no puede superar al necesario para encapsularlos dentro de un datagrama IP, y tienen distintos campos:

- Versión de CoAP.
- Tipo de mensaje.
- Longitud del Token.

- Código, en formato “c.dd”, siendo “c” la clase indicando si es una solicitud, una respuesta satisfactoria, un error del cliente o un error del servidor, y “dd” el detalle.
- ID de mensaje.
- Token, usado para correlacionar solicitudes y respuestas.
- Opciones.
- Payload o carga útil.

Los distintos tipos de mensajes que se pueden transmitir son los siguientes:

- Mensajes confirmables (CON): utilizados cuando se necesita asegurar que el mensaje llegue al destinatario. Contienen un temporizador y un mecanismo de retroceso. Al transmitir una petición CON, se espera recibir un mensaje ACK con el mismo ID de la petición o una respuesta en un mensaje CON y con un ID distinto.
- Mensajes no confirmables (NON): son mensajes menos fiables, usados para enviar información no crítica, que no requieren un acuse de recibo (ACK). En el caso de enviarse una solicitud como un mensaje NON, la respuesta también se recibirá como un mensaje NON (en el caso que el servidor tenga la información necesaria para responder).
- Mensajes de acuse de recibo (ACK): son transmitidos para reconocer que ha llegado un mensaje confirmable específico identificado por su ID de transacción. Estos mensajes pueden tener su propio payload y algunas opciones para detallar la recepción.
- Mensaje de reinicio (RST): cuando al receptor le falta información para procesar una solicitud, transmite un mensaje RST. Esto ocurre cuando el receptor se ha reiniciado y no ha persistido adecuadamente la petición recibida anteriormente, o cuando cancela una transacción.

Además es un protocolo diseñado para requerir poca energía en la transferencia (tiene bajo consumo de recursos), y permite transferir tanto datos como archivos, utilizar el protocolo DTLS para aumentar la seguridad de las transferencias, y extender la implementación del protocolo con funcionalidades adicionales. Por el contrario, es un protocolo menos maduro y menos adoptado que sus alternativas, resultando en una menor cantidad de recursos, guías y herramientas, además de una compatibilidad reducida con otros dispositivos IoT.

https://webbylab.com/blog/mqtt-vs-other-iot-messaging-protocols-detailed-comparison/ https://www.techtarget.com/iotagenda/tip/
12-most-commonly-used-IoT-protocols-and-standards https://build5nines.com/top-iot-messaging-
protocols/ https://www.a3logics.com/blog/iot-messaging-protocols/ https://en.wikipedia.org/wiki/MQTT
https://en.wikipedia.org/wiki/Publish %E2 %80 %93subscribe_pattern https://www.emqx.com/en/blog/what-
is-the-mqtt-protocol https://www.gotoiot.com/pages/articles/mqtt_intro/index.html https://dzone.com/refcardz/getting-
started-with-mqtt https://www.hivemq.com/resources/achieving-200-mil-concurrent-connections-with-
hivemq/ http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ https://en.wikipedia.org/wiki/Advanced_Messag
https://www.gotoiot.com/pages/articles/amqp_intro/index.html https://www.emqx.com/en/blog/mqtt-vs-

amqp-for-iot-communications#what-is-amqp http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#section-message-format https://en.wikipedia.org/wiki/XMPP https://www.pubnub.com/guides/xmpp/ https://blazeclan.com/blog/xmpp-for-dummies-part-3-stanzas-in-detail/ https://slixmpp.readthedocs.io/en/latest/api/stanza/presen https://slixmpp.readthedocs.io/en/latest/api/stanza/iq.html https://www.dds-foundation.org/omg-dds-standard/ https://www.utpl.edu.ec/proyectomiddleware/?q=tutorial-dds https://www.emqx.com/en/blog/coap-protocol0 https://www.gotoiot.com/pages/articles/coap_intro/index.html (DOCUMENTO) https://datatracker.ietf.org/doc/html/rf

### 2.2.6. Comparación de todos los protocolos

A partir de la información de los apartados anteriores, se pueden recopilar las diferencias y similitudes en los siguientes puntos:

1. *MQTT*:
   - **Descripción corta:** Basado en colas de mensajes.
   - **Patrón de comunicación:** Publicación-suscripción.
   - **Necesita intermediario:** Sí, el broker.
   - **Protocolo de transporte:** TCP.
   - **Ventajas:** Muy utilizado, ligero, eficiente, útil en dispositivos y redes de recursos limitados, soporta distintas calidades de servicio.
   - **Desventajas:** Encriptación y enrutación limitadas, bajo soporte para tipos de datos complejos.
   - **Casos de uso:** Telemetría, mensajería ligera, automatización industrial, monitorización del entorno, hogares inteligentes, soluciones energéticas.
2. *AMQP*:
   - **Descripción corta:** Basado en colas de mensajes y exchanges.
   - **Patrón de comunicación:** Publicación-suscripción.
   - **Necesita intermediario:** Sí, el broker.
   - **Protocolo de transporte:** TCP.
   - **Ventajas:** Alto rendimiento, seguro, ampliamente usado, parecido a MQTT.
   - **Desventajas:** Alto consumo de recursos, difícil aprendizaje.
   - **Casos de uso:** Servicios financieros, procesamiento de transacciones, envío de datos en tiempo real, comunicación segura entre entidades.
3. *XMPP*:
   - **Descripción corta:** Comunicación de datos y presencia mediante mensajes XML.
   - **Patrón de comunicación:** Cliente-servidor.
   - **Necesita intermediario:** Sí, el servidor.
   - **Protocolo de transporte:** TCP.
   - **Ventajas:** Robusto, extensible, muy ampliable de funciones.

- **Desventajas:** No optimizado para entornos limitados, complejo de implementar.
- **Casos de uso:** Mensajería instantánea, redes sociales, plataformas de colaboración.

4. *DDS*:
   - **Descripción corta:** Comunicación sin servidor.
   - **Patrón de comunicación:** Publicación-suscripción.
   - **Necesita intermediario:** No.
   - **Protocolo de transporte:** TCP y UDP, compatible con otros.
   - **Ventajas:** Alto rendimiento, fácil de escalar, comunicación centrada en datos en tiempo real, soporte de tipos de datos complejos, configurable la calidad de servicio.
   - **Desventajas:** Complejo de implementar, altos recursos de dispositivos y de ancho de banda.
   - **Casos de uso:** Sistemas de tiempo real, aplicaciones de misión crítica, automatización industrial, automoción, gestión de cadenas de suministro, sistemas distribuidos de gran escala.
5. *CoAP*:
   - **Descripción corta:** Basada en recursos compartidos y accesibles mediante REST.
   - **Patrón de comunicación:** Cliente-servidor.
   - **Necesita intermediario:** No.
   - **Protocolo de transporte:** UDP.
   - **Ventajas:** Eficiente en redes y dispositivos de recursos limitados, soporte nativo para tecnologías web.
   - **Desventajas:** Bajo soporte de clientes concurrentes.
   - **Casos de uso:** Dispositivos de recursos limitados, automatización en hogares.

{ https://webbylab.com/blog/mqtt-vs-other-iot-messaging-protocols-detailed-comparison/ https://www.a3logics.com/blog/iot-messaging-protocols/

}

## 2.3. Espressif y sus dispositivos

Espressif Systems es una multinacional pública de semiconductores sin fábrica fundada en 2008, que opera como líder mundial en el ámbito del Internet de las Cosas y está comprometida en proporcionar a millones de usuarios algunos de los mejores dispositivos y plataformas de software de la industria, junto con una variedad de soluciones IoT seguras.

La empresa se identifica como una empresa compuesta por especialistas, ingenieros y científicos dedicados al desarrollo de soluciones de vanguardia de bajo consumo que aprovechan la comunicación inalámbrica, el IoT y la inteligencia artificial de las cosas (AIoT). Estas soluciones se caracterizan por su seguridad, robustez, eficiencia energética, versatilidad, asequibilidad y enfoque código abierto.

Con el surgimiento de la inteligencia artificial y la evolución del IoT, la demanda de productos con conectividad inalámbrica segura ha ido creciendo considerablemente, y Espressif Systems ha respondido a este desafío desarrollando soluciones adaptadas a las necesidades del mercado. Espressif emplea los nodos de tecnología avanzada, la informática de bajo consumo, la comunicación inalámbrica, así como la tecnología de malla, para crear conjuntos de chips y módulos de alto rendimiento, que son más inteligentes, adaptables y versátiles.

El compromiso de esta empresa china con el código abierto se refleja en su oferta de una variedad de frameworks y herramientas de desarrollo para construir aplicaciones en diferentes ámbitos, como IoT, audio, malla, conectividad de dispositivos, reconocimiento facial y asistentes de voz. Gracias a las tecnologías y soluciones abiertas de Espressif, permiten acercar el IoT a sus clientes, comerciales y no comerciales, y que desarrolladores de todos los ámbitos puedan utilizarlas a nivel mundial y construir sus propios dispositivos inteligentes conectados y soluciones a algunos problemas del presente. Con este acercamiento a sus clientes, estos últimos pueden efectuar una conectividad inalámbrica de sus productos optimizando el rendimiento de estos y en un reducido tiempo de desarrollo. A su vez, apoyan activamente proyectos de código abierto en la comunidad de makers, creyendo en la accesibilidad de la tecnología como motor para el desarrollo de la sociedad AIoT del futuro.

Además de su enfoque en el código abierto y en la tecnología accesible, Espressif se destaca por el compromiso con la sostenibilidad, la responsabilidad social corporativa y por la inversión en investigación y desarrollo de tecnologías ecológicas. A lo largo de su trayectoria, ha respaldado activamente soluciones que reducen el consumo de energía y el desperdicio de materiales.

Los productos de Espressif se utilizan ampliamente en productos como electrodomésticos, altavoces, bombillas, cámaras, tabletas, TV boxes, terminales de pago y otros dispositivos de electrónica de consumo, y son especialmente útiles en contextos donde es necesario el Internet de las Cosas, comunicación inalámbrica e Inteligencia Artificial. Espressif es conocido por sus populares series de chips, módulos y placas de desarrollo ESP8266, ESP32 y ESP32-S, ESP32-C y ESP32-H, los cuales se analizarán en los siguientes apartados. { https://www.espressif.com/en/company/about-espressif https://www.eurotronix.com/es/fabricantes/espressif https://www.digikey.es/es/supplier-centers/espressif-systems }

### 2.3.1. ESP8266

El ESP8266 es un SoC o sistema en un chip diseñado para dispositivos móviles, electrónica portátil y aplicaciones del Internet de las Cosas. Lanzado en agosto de 2014, integra un procesador mononúcleo Tensilica L106 con una arquitectura RISC de 32 bits de bajo consumo y una velocidad de reloj de entre 80 y 160 MHz.

Presenta una arquitectura de ahorro de energía, permitiendo establecer el chip en modo activo, reposo y reposo profundo, lo cual es útil para que los dispositivos diseñados para alimentarse por batería funcionen durante mucho más tiempo.

En cuanto a memoria, no dispone de una memoria flash para almacenar programas, la cual debe ser proporcionada por el módulo que implemente este chip y puede tener un tamaño máximo de 16 MiB. Integra una RAM para instrucciones de 32 KiB, una caché de instrucciones 32 KiB, 80 KiB para almacenar datos del usuario y 16 KiB para datos del sistema de ETS.

Su bajo voltaje operativo oscila entre 2,5 y 3,6 V, con una corriente de operación alrededor de los 80 mA. Cuenta con la capacidad de funcionar en entornos industriales gracias a su amplio rango de temperatura de operación, que va de -40 y 125 ºC.

Admite distintos tipos de protocolos, como IPv4, TCP, UDP y HTTP. Es un dispositivo certificado para funcionar por Wi-Fi y compatible con los protocolos 802.11 b/g/n en una frecuencia de 2,4 GHz. Tiene la capacidad de actuar como cliente en redes protegidas por claves WEP, WPA y WPA2, además de poder actuar como punto de acceso inalámbrico.

También integra en sus dimensiones compactas 16 pines GPIO para conectar dispositivos de entrada y salida, un conversor analógico de 10 bits, conmutadores de antena, un amplificador de potencia y de recepción, un balun de radiofrecuencia y módulos de gestión de potencia.

Este sistema admite varios IDEs y lenguajes de programación, como C y C++, utilizando Arduino IDE o PlatformIO; MicroPython, utilizando Mu Editor, Thonny IDE o Pymakr; y Lua, utilizando LuaLoader.

{ https://www.espressif.com/en/products/socs/esp8266 https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf https://en.wikipedia.org/wiki/ESP8266 https://www.luisllamas.es/esp8266/ https://programarfacil.com/podcast/esp8266-wifi-coste-arduino https://randomnerdtutorials.com/micropython-ides-esp32-esp8266/ https://www.danielmartingonzalez.com/es/usando-lua-en-esp8266-nodemcu-con-lualoader-y-esplorer/ }

### 2.3.2. ESP32

El ESP32 es el SoC sucesor del ESP8266. Igual de apto para electrónica portátil e Internet de las Cosas, comparte muchas características y añade mejoras que lo convierten en un sistema muy superior.

Integra un procesador Tensilica Xtensa LX6 de doble núcleo (o de uno, dependiendo de la variante utilizada) cuya frecuencia de reloj oscila entre los 160 y 240 MHz, que trabaja en conjunto con un coprocesador de ultra baja energía.

La memoria experimenta un significativo aumento, con un total de 520 KiB de SRAM, 448 KiB de memoria de solo lectura, 32 KiB de caché y hasta 4 MiB de memoria de almacenamiento (dependiendo del modelo).

Este chip, lanzado en septiembre de 2016, añade en comparación con el ESP8266 una mejora de potencia, soporte de Bluetooth 4.2 y BLE, sensor de temperatura, sensor hall, sensor táctil, reloj de tiempo real, más pines GPIO (hasta 34) y varios modos de energía.

Además, incorpora arranque seguro, encriptado de la flash y soporte de aceleración por hardware para los algoritmos y protocolos de cifrado y encriptación AES, SHA-2, RSA, ECC y el generador de números aleatorios.

El ESP32 tiene la posibilidad de funcionar como un sistema autónomo o como parte de un puente e interconexiones, y tiene la capacidad de interactuar con otros sistemas para proveer funcionalidad Wi-Fi y Bluetooth a través de sus interfaces.

Desde el lanzamiento del ESP32 original, han ido apareciendo variantes con distintas capacidades y procesadores, pero gran parte del código del ESP32 es compatible. Estas variantes son:

- ESP32-S2: Enfocado en el consumo, integra un procesador mononúcleo LX7, reduce la memoria disponible y no tiene soporte de Bluetooth.
- ESP32-S3: Utiliza el mismo procesador que el anterior, contiene más memoria y da soporte a Bluetooth 5 y BLE, enfocado al soporte de inteligencia artificial.
- ESP32-C3: Contiene un procesador RISC-V mononúcleo y admite Bluetooth 5 y BLE, enfocado en la seguridad.
- ESP32-C6: Centrado en la conectividad, la principal diferencia con el anterior es el soporte de Bluetooth 5.3, Wi-Fi 6 (802.11ax) y conectividad de radio (802.15.4) compatible con los protocolos Thread, Zigbee y Matter.
- ESP32-C2: Incorpora un procesador RISC-V mononúcleo y admite Bluetooth 5 y BLE. Es un chip de pequeñas dimensiones que mantiene una conectividad robusta y estándares de seguridad.
- ESP32-C5: Es la versión más reciente con mayor velocidad de reloj y capacidad de memoria, y es el primero que soporta Wi-Fi 6 a 5 GHz. Su enfoque en la conectividad también proviene de la capacidad de conexión con Bluetooth 5.2.

{ https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf https://www.espressif.com/en/products/soc https://en.wikipedia.org/wiki/ESP32 https://www.luisllamas.es/esp32/ https://www.espboards.dev/blog/esp32-soc-options/ }

## 2.4. ESP-NOW

El Internet de las Cosas se forma a partir de una conectividad entre objetos, donde surge la necesidad de un protocolo que equilibre las necesidades de latencia, uso de energía, capacidad de transmisión, confiabilidad y seguridad en la transmisión de datos. Son factores determinantes en el futuro desarrollo de esta tecnología, y aparecen como candidatos un gran número de protocolos, destinados tanto para la comunicación en

área local, como Wi-Fi, para áreas amplias, como LoRa, y para transmisión a corta distancia, como RFID. Recientemente, al grupo de protocolos se ha añadido ESP-NOW, el cual, pese a sus limitaciones, tiene características notables y es adecuado para IoT.

ESP-NOW es un protocolo de comunicación inalámbrica diseñado por Espressif para su uso entre sus dispositivos, como los ESP8266 y ESP32. Con el objetivo de sustituir a Wi-Fi y a otras tecnologías, ESP-NOW es capaz de realizar transmisiones de información y control rápidas, estables y con un bajo consumo de recursos de CPU y memoria flash entre dispositivos inteligentes, sin necesidad de un enrutador.

Se caracteriza por la rapidez de la transmisión, lograda mediante la evitación de establecer una conexión previa entre dispositivos. Permite a su vez poner a disposición los dispositivos para transmitir datos y recibir órdenes instantáneamente tras el encendido. Además, este protocolo está basado en la capa de enlace de datos y es capaz de omitir las capas de red, transporte, sesión, presentación y aplicación del modelo OSI, reduciendo el consumo de energía (mejorando la autonomía en dispositivos con batería) y el retardo en la recepción y en el procesamiento de mensajes debido a la nula necesidad de cabeceras de paquete o desempaquetadores de cada capa. En redes congestionadas, es una característica beneficiosa, ya que brinda la capacidad de respuestas rápidas que reducen el retraso causado por la pérdida de paquetes. En el modelo utilizado en ESP-NOW en comparación con el modelo OSI estándar de la figura 2.2 se puede observar la notoria ausencia de las 5 capas mencionadas anteriormente.

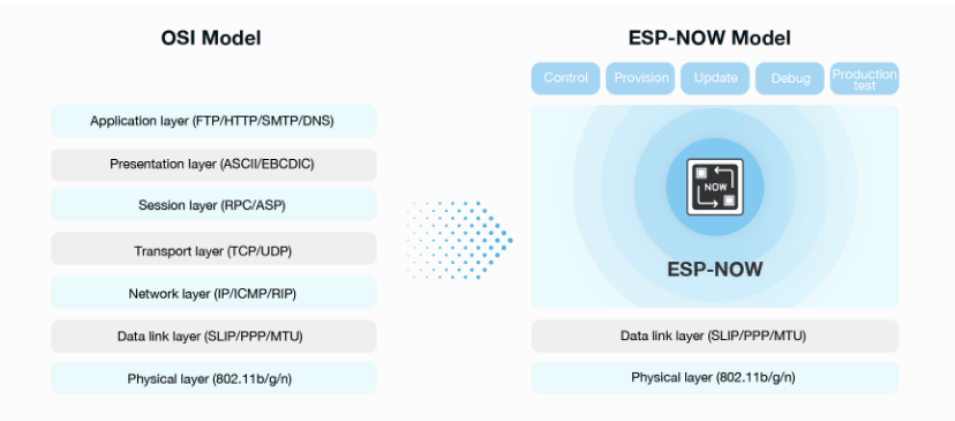


Figura 2.2: Comparación de las capas del modelo OSI con las del modelo ESP-NOW **TODO: REFERENCIAR**

Pese al objetivo de ESP-NOW de reemplazar Wi-Fi, en los dispositivos de Espressif es capaz de coexistir simultáneamente junto a Wi-Fi y Bluetooth. Esto es útil para utilizar un dispositivo como hotspot o gateway y exportar los datos intercambiados entre ESP-NOW hacia otras redes.

Tiene la capacidad de transmitir datos de máquina a máquina, unicast, o broadcast, a todos los conectados, y únicamente para establecer la comunicación es necesario la dirección MAC de destino y establecer un canal de transmisión. No obstante, dispone de una cantidad limitada de dispositivos con los que se puede

emparejar. En general, el número de dispotivos emparejados no puede exceder de 20, y la cantidad de estos con los que se puede establecer una comunicación cifrada es configurable. Por defecto, este valor es 7, y admite un valor máximo de 17. Esta limitación puede ser un inconveniente en caso de necesitar una gran cantidad de dispositivos interconectados, pero una solución sería formar jerarquías de dispositivos.

Para el envío de datos, permite establecer una función de callback que será llamada instantáneamente tras el envío para poder gestionarlo. Esto puede llegar a ser útil debido a los posibles fallos que puedan ocurrir, por ejemplo, si el dispositivo de destino no existe, los canales de transmisión establecidos en ambos dispositivos no son los mismos o la trama de acción se pierde por interferencias. De la misma manera, se puede establecer una función de callback que sea llamada al recibir datos para poder tratarlos. Cabe remarcar que el protocolo no garantiza que se reciban los datos correctamente, pero existe la posibilidad de establecer ACKs para confirmar la correcta recepción y procesamiento de los datos, además de números de secuencia para afrontar la duplicidad.

ESP-NOW utiliza tramas de acción específicas del proveedor para encapsular y transmitir datos de una longitud máxima de 250 bytes, con un alcance de transmisión de entre 100 y 500 metros, dependiendo de las condiciones atmosféricas, y con una tasa de velocidad de bits de 1 megabit por segundo. Esto es beneficioso para la comunicación a larga distancia debido a su gran alcance en dispositivos al aire libre o incluso separados por paredes o pisos. Sin embargo, limita su uso debido a la pequeña carga útil que puede transmitir, por lo que en otros casos podría ser mejor utilizar otras tecnologías como Wi-Fi. ESP-NOW utiliza tramas de un tamaño entre 43 y 293 bytes, cuyo formato está compuesto por los siguientes campos:

- Cabecera MAC, distinta de una cabecera MAC común debido a su funcionamiento sin conexión.
- Código de categoría, establecido a 127 para indicar la categoría específica del proveedor.
- Identificador de la organización único, que son los 3 primeros bytes de la dirección MAC aplicada por Espressif.
- Valores aleatorios, utilizados para prevenir ataques de retransmisión.
- Contenido específico del proveedor, que ocupa entre 7 y 257 bytes y contiene los siguientes campos específicos del proveedor:
    - ID del elemento, establecido a 221 para indicar que se trata del elemento específico del proveedor.
    - Longitud total del resto de campos.
    - Identificador de la organización, igual que el mencionado antes, los 3 primeros bytes de la dirección MAC.
    - Tipo, con valor 4 para indicar ESP-NOW.
    - Versión de ESP-NOW.
    - Cuerpo, que contiene los datos de ESP-NOW y puede ocupar entre 0 y 250 bytes.
- Frame Check Sequence, utilizado para verificar la integridad de la información recibida.

Existe la posibilidad de asegurar la transmisión de datos a través de ESP-NOW utilizando algoritmos de encriptación ECDH y AES y el método CBC-MAC Protocol (CCMP) que protegen las tramas de acción. El funcionamiento de estos se realiza mediante dos tipos de claves en los dispositivos: una Clave Maestra Principal (PMK) y varias Claves Maestras Locales (LMK) que corresponden a cada dispositivo emparejado. La PMK se utiliza para cifrar la LMK, y la LMK del dispositivo emparejado se utiliza para cifrar la trama de acción específica del proveedor. Esto está limitado a comunicaciónes entre pares, ya que no se admite el cifrado de tramas utilizadas para la multidifusión.

En cuanto a la gestión de dispositivos, puede utilizarse como un protocolo que ayude al aprovisionamiento de datos y configuraciones a dispositivos, depurarlos y actualizar su firmware.

ESP-NOW no necesita ningún procedimiento especial aparte de la implementación para poder utilizarse con fines comerciales. En la actualidad, se encuentra ampliamente utilizado en electrodomésticos inteligentes, iluminación inteligente, control remoto, sensores y otros.

{ (imagen, fuente): https://www.espressif.com/sites/all/themes/espressif/images/esp-now/model-en-mobile.png https://www.espressif.com/en/solutions/low-power-solutions/esp-now https://docs.espressif.com/projects/espressif-esp-faq/en/latest/application-solution/esp-now.html https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html https://emariete.com/esp8266-esp32-espnow/ }

## 2.4.1. Comparaciones con otras tecnologías

### 2.4.1.1. Wi-Fi

“Wi-Fi” es el nombre que otorga la Wi-Fi Alliance a este protocolo de red inalámbrica basado en los estándares IEEE 802.11. Este protocolo es ampliamente utilizado para enlazar dispositivos en redes LAN y proveer de acceso a Internet utilizando ondas de radio de 2,4 o 5 GHz (dependiendo de la versión) para transmitir la información, cumpliendo con la misión de ser una alternativa al envío de datos a través de cables.

En una red que utiliza Wi-Fi, se pueden encontrar una variedad de dispositivos cliente, que son los que aprovechan las características de la red, y dispositivos que distribuyen la red. Estos últimos consisten en routers o enrutadores, que brindan la conexión a Internet a los dispositivos y enrutan los mensajes enviados a través de la red; puntos de acceso, que transmiten la señal inalámbrica y es donde se conectan los dispositivos introduciendo las credenciales de la red; y repetidores, utilizados para extender el área de cobertura de una red. Entre los dispositivos, tanto clientes como los distribuidores de red, se utilizan adaptadores de red inalámbrica, que convierten los datos en una señal de radio y viceversa.

Es una de las tecnologías más utilizadas a nivel mundial y, en la mayoría de hogares y establecimientos, se encuentran disponibles estas redes, además de los dispositivos que se fabrican con un Certificado Wi-Fi otorgado por la Wi-Fi Alliance tras superar las pruebas homologadas de interoperabilidad. Esta popularidad

es beneficiosa para el Internet de las Cosas, ya que ofrece una capa de compatibilidad con una amplia gama de dispositivos sin necesidad de antenas, adaptadores de red ni otro tipo de hardware adicional. Además, Wi-Fi no es una tecnología nueva, tiene un sólido legado de interoperabilidad, y permite enviar información entre dispositivos con baja latencia.

Entre sus características adicionales se encuentra la topología flexible, que permite implementar Wi-Fi conectando los dispositivos de distintas maneras; la seguridad, ya que es posible tener redes protegidas con contraseñas cifradas mediante distintos protocolos (WEP, WPA y WPA2), el coste de implementación, que en comparación con la instalación de una red cableada, resulta más económico; y la capacidad de llegar a donde los cables no pueden llegar.

Existen distintos estándares de Wi-Fi que definen cómo actúa la red, y que cambian cada pocos años trayendo mejoras en el alcance, la velocidad y la conectividad. Por lo general, los dispositivos certificados para un estándar son intercomunicables con los certificados para otro estándar Wi-Fi siempre que compartan la misma banda de frecuencia, por lo que no es una preocupación tener todos los dispositivos con la versión más reciente. Los estándares Wi-Fi se muestran en la tabla 2.1.

Tabla 2.1: Lista de estándares Wi-Fi *TODO: REFERENCIAR*

| Nombre comercial | Estándar IEEE | Año | Frecuencia (GHz) | Velocidad máxima | Rango (metros) |
|---|---|---|---|---|---|
| Wi-Fi 1 | 802.11a | 1999 | 5 | 54 Mbps | 120 |
| Wi-Fi 2 | 802.11b | 1999 | 2,4 | 11 Mbps | 140 |
| Wi-Fi 3 | 802.11g | 2003 | 2,4 | 54 Mbps | 140 |
| Wi-Fi 4 | 802.11n | 2009 | 2,4 y 5 | 450 Mbps | 250 |
| Wi-Fi 5 | 802.11ac | 2014 | 5 | 2,3 Gbps | 35 |
| WiGig | 802.11ad | 2016 | 60 | 7 Gbps | 100 |
| Wi-Fi 6 | 802.11ax | 2019 | 2,4, 5 y 6 | 9,6 Gbps | 240 |
| Wi-Fi 7 | 802.11be | 2024 | 2,4, 5 y 6 | 46 Gbps | (Por determinar) |

La popularidad de Wi-Fi abarca una gran variedad de dispositivos soportados, como teléfonos inteligentes, ordenadores, televisores inteligentes, impresoras e incluso placas de desarrollo como ESP8266 y ESP32.

{ https://en.wikipedia.org/wiki/Wi-Fi https://es.wikipedia.org/wiki/Wifi https://emariete.com/esp8266-esp32-espnow/ https://www.wi-fi.org/discover-wi-fi/internet-things https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/ https://www.proofpoint.com/es/threat-reference/wifi#:~:text=Wifi %2C %20que %20es %20una %20cont

```
Fuentes de tabla
```

```
https://www.intel.la/content/www/xl/es/support/articles/000005725/wireless/
    ↪ legacy-intel-wireless-products.html
https://www.monolitic.com/cual-es-la-mejor-tecnologia-wifi-para-desarrollos-iot/
https://www.makeuseof.com/tag/understanding-common-wifi-standards-technology-
    ↪ explained/
https://www.netspotapp.com/es/blog/wifi-standards/
https://www.xataka.com/nuevo/nuevo-wifi-7-informacion
https://www.geckoandfly.com/10041/wireless-wifi-802-11-abgn-router-range-and-
    ↪ distance-comparison/
https://www.business.com/articles/what-is-802-11-ax-wi-fi/
```

}

#### 2.4.1.2. Bluetooth

Bluetooth es un estándar de tecnología que facilita el intercambio de datos entre dispositivos a través de distancias cortas, con un máximo de 10 metros.

Desde su introducción en 1998, ha pasado por múltiples revisiones, siendo las más notorias de la última década: - Versión 4.0 hasta 4.2: Aumentó la velocidad de transferencia de datos a 24 Mbps y el alcance hasta 100 metros, y añadió soporte al protocolo IPv6. Además, introdujo BLE o Bluetooth Low Energy, una nueva variante de esta tecnología. - Versión 5.0 hasta 5.2: Aumentó la velocidad de transferencia de datos a 50 Mbps y el alcance hasta 200 metros, y realizó mejoras en la transmisión de audio y en el consumo de energía.

El Bluetooth Low Energy mencionado anteriormente fue diseñado para operaciones de bajo consumo de energía, y capaz de soportar tipologías de comunicación (punto a punto, difusión y malla), a diferencia del clásico que solo admite punto a punto. Mientras que el Bluetooth clásico se usa para transferir datos y sonido, BLE es capaz de utilizarse para analizar con alta precisión la presencia, distancia y dirección del dispositivo.

Bluetooth utiliza ondas de radio UHF en las bandas ISM sin licencia de 2,4 GHz, y se utiliza como alternativa a las conexiones por cable para intercambiar ficheros y conectar transmisores de audio. Gracias a su bajo consumo de energía, seguridad, capacidad contra interferecias, compatibilidad con varios sistemas operativos y facilidad de implementación, esta tecnología se convierte en una buena opción para la implementación del Internet de las Cosas. Además, tiene la capacidad de agregar capas de cifrado, autenticación y verificación, y de construir PAN entre dispositivos al interconectar varios entre sí. Es común encontrarlo en pulseras y relojes inteligentes, teléfonos inteligentes, ordenadores, reproductores de música, altavoces, auriculares, y placas de desarrollo como ESP32.

{ https://www.bluetooth.com/learn-about-bluetooth/tech-overview/ https://en.wikipedia.org/wiki/Bluetooth https://www.xatakahome.com/curiosidades/bluetooth-su-evolucion-estas-diferencias-distintas-versiones https://www.mokosmart.com/es/what-is-bluetooth-iot-and-why-choose-it/ }

#### 2.4.1.3. Comparaciones entre Wi-Fi, Bluetooth y ESP-NOW

/*TODO: insertar logos*/

Es de alta relevancia comparar estos protocolos, ya que son los más populares en el ámbito del Internet de las Cosas y son compatibles con las placas ESP32 que incorporan el reciente e innovador ESP-NOW. En particular, ESP32 es compatible con Bluetooth 4.2 y con Wi-Fi b, g y n de 2,4 GHz, por lo que a lo largo de este apartado se comparan estas versiones. Estos tres protocolos son similares en varios aspectos, ya que utilizan ondas de radio para transmitir datos de forma inalámbrica a una amplia gama de dispositivos, de manera rápida y fiable. Esto puede resultar en una decisión compleja para elegir entre los tres, aunque hay escenarios en los que no es necesario elegir, ya que en un ESP32 pueden trabajar en conjunto.

De manera más resumida, estas son las características teóricas que ofrecen los protocolos:

1. *ESP-NOW*:
   - **Alcance**: 220 metros.
   - **Cantidad de dispositivos conectables**: 20 a cada nodo.
   - **Unidad de Transmisión Máxima (MTU)**: 250 bytes.
   - **Velocidad de transmisión**: 1 Mbps.
   - **Uso**: IoT y comunicación entre dispositivos de Espressif.
2. *Wi-Fi b/g/n (2,4 GHz)*:
   - **Alcance**: 250 metros.
   - **Cantidad de dispositivos conectables**: Depende de la configuración de la red y la asignación de direcciones IP.
   - **Unidad de Transmisión Máxima (MTU)**: 1460 bytes, configurado en la librería de red de ESP32.
   - **Velocidad de transmisión**: 54 Mbps.
   - **Uso**: Conexión a internet, acceso a dispositivos e IoT.
3. *Bluetooth 4.2*:
   - **Alcance**: 50 metros.
   - **Cantidad de dispositivos conectables**: 7 a cada nodo.
   - **Unidad de Transmisión Máxima (MTU)**: 251 bytes.
   - **Velocidad de transmisión**: 1 Mbps.
   - **Uso**: Audio, dispositivos personales e IoT.

/*TODO: referenciar*/

Para detallar las comparaciones aplicando los protocolos en escenarios de uso real, se han tomado los datos de distintas pruebas realizadas y detalladas en una publicación de 2021 llevada a cabo por Dania Eridani, Adian Fatchur Rochim y Faiz Noerdiyan Cesara, miembros del Departamento de Ingeniería Informática de la Universidad de Diponegoro, en Indonesia.

En esta publicación se realizaron pruebas de rango, velocidad, latencia, consumo y resistencia a barreras. Para ello, se utilizaron una placa ESP32 Development Board, una ESP32-CAM y una ESP32U, además de una antena externa de 2.4GHz conectada a la última placa mencionada y utilizada en ciertas pruebas. En la mayoría de las pruebas, estas placas estaban conectadas a 1 metro de distancia entre ellas en un mismo circuito en el cual, al pulsar un botón, iniciaban simultáneamente la transmisión y recepción de datos.

Tabla 2.2: Resultados de las pruebas comparativas realizadas entre los protocolos *TODO: REFERENCIAR*

| Prueba realizada | ESP-NOW | Wi-Fi | Bluetooth |
|---|---|---|---|
| *Rango máximo (metros), más valor es mejor:* | | | |
| · con antena interna | **185** | 84 | 15 |
| · con antena externa | **220** | 88 | 25 |
| *Velocidad de transmisión, más valor es mejor:* | | | |
| · de 200 bytes (Kbps) | 588 | **2048** | 938 |
| · del MTU (Mbps) | 0,63 | **24,86** | 0,98 |
| *Latencia (µs), menos valor es mejor:* | | | |
| · con 100 bytes de datos | **1869** | 3435 | 8514 |
| · con 50 bytes de datos | **1435** | 3388 | 6460 |
| · con 10 bytes de datos | **1133** | 3367 | 6200 |
| · con 1 byte de datos | **1059** | 3359 | 6048 |
| *Consumo (mW), menos valor es mejor:* | | | |
| · solo conectado, como receptor | 489 | 214 | **141** |
| · solo conectado, como emisor | 449 | 465 | **212** |
| · transfiriendo datos, como receptor | 511 | 477 | **338** |
| · transfiriendo datos, como emisor | 1042 | 538 | **441** |

La primera prueba llevada a cabo fue la de rango máximo, exponiendo las placas en un área residencial con una condición de propagación de alcance visual o line of sight, en la que las ondas viajan en un camino directo desde el emisor al receptor. La prueba se basaba en un emparejamiento correcto entre las placas y la transmisión y recepción de 1 byte, confirmando una transmisión correcta. En la tabla 2.2 se pueden

observar las distancias máximas obtenidas, demostrando que el uso de una antena externa correctamente instalada es capaz de aumentar el rango en un 18,9 %, 4,7 % y 66,6 % en los protocolos ESP-NOW, Wi-Fi y Bluetooth, respectivamente. Además, ESP-NOW es el protocolo que soporta una mayor distancia entre dos dispositivos interconectados.

La segunda prueba se centró en la capacidad de transmisión de los datos, utilizando código programado en Arduino IDE para ESP-NOW, Bluetooth Serial y Wi-Fi UDP. Se realizó en dos tandas, la primera utilizando paquetes de 200 bytes, y la segunda utilizando paquetes con un tamaño igual al MTU de cada protocolo. En el caso de ESP-NOW y Wi-Fi, se requiere de un retardo entre paquetes para prevenir errores de transmisión en ESP-NOW y para alcanzar la velocidad máxima en Wi-Fi. Tras 30 pruebas ejecutadas, cuyos resultados se muestran en la tabla 2.2, se puede concluir que Wi-Fi es el protocolo con mayor capacidad para transmitir datos rápidamente.

Para la prueba de latencia, se realizaron mediciones para cuatro tamaños distintos de paquetes. Los resultados, mostrados en la tabla 2.2, demuestran que la variación del tamaño es directamente proporcional a la latencia: a mayor tamaño, mayor latencia. Se observa que ESP-NOW tiene la menor latencia, 1ms para 1 byte, tres veces menos que Wi-Fi y seis veces menos que Bluetooth, siendo la mejor opción en cuanto a esta propiedad de la conexión. Debido a la baja compatibilidad de dispositivos que admiten ESP-NOW, una buena alternativa sería utilizar Wi-Fi para la transmisión de datos con poco retardo.

El consumo, una propiedad importante en el Internet de las Cosas, fue medido de cuatro maneras distintas, mezclando los dispositivos en modo de transmisor y receptor, y en modo solo conectado o transmitiendo datos. En la tabla 2.2 se puede observar que Wi-Fi y ESP-NOW tienen un consumo similar al estar conectados como transmisor, mientras que Bluetooth consume menos tanto como transmisor como receptor. En el caso de transmitir datos, ESP-NOW es el que más consume ya que necesita activar internamente Wi-Fi para funcionar. La prueba demuestra que, en el caso de implementar una batería, la mejor solución puede ser Bluetooth, aunque hay diversas maneras de mejorar el código que se utilice en la placa ESP32 para mejorar la eficiencia energética, como añadir paradas y suspensión.

Finalmente, se evaluó una prueba para comprobar la capacidad y potencia de atravesar barreras con los distintos protocolos, y cómo estas reducen la señal. Para ello, se instaló un dispositivo transmisor a 10 centímetros de una barrera, y el receptor a 5-10 metros de la misma barrera, que es de distintos materiales: madera contrachapada, cristal, metal y muro. La prueba se realiza con distintos materiales, ya que cada uno tiene distintos valores que demuestran la capacidad de absorber, bloquear y reflejar señales. Los resultados demuestran que la madera y el cristal no afectan gravemente a la señal en comparación con la transmisión al aire libre, que el metal afecta de manera más notoria a los 10 metros y que el Bluetooth sin una antena externa no es capaz de atravesar muros a 10 metros.

Las conclusiones a las que se llega en esta publicación se pueden representar en la figura 2.3, que muestra

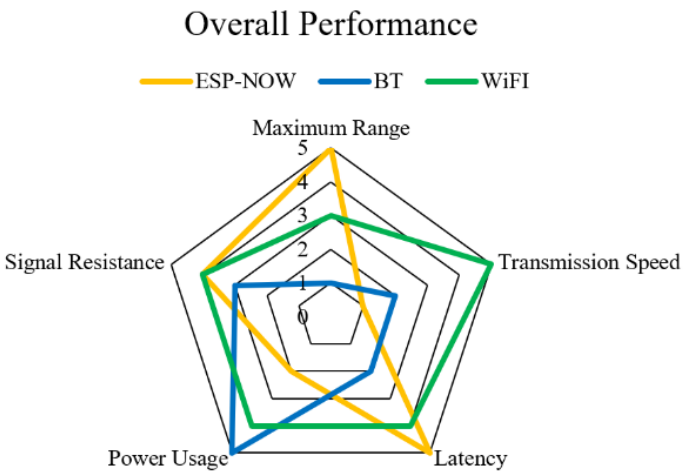


Figura 2.3: Comparación del rendimiento entre protocolos en distintos aspectos **TODO: REFERENCIAR**

que:

- La mayor ventaja de Bluetooth se encuentra en el consumo, ya que su rendimiento en el resto de los aspectos es muy deficiente.
- ESP-NOW es el mejor candidato cuando se requieren rangos elevados, una señal de comunicación resistente y mínima latencia en velocidades de datos muy pequeñas, pero su uso es poco eficiente de energía.
- Wi-Fi es un protocolo muy equilibrado, y destaca por su velocidad.

{ https://es.wikipedia.org/wiki/Bluetooth_(especificaci %C3 %B3n) https://docs.espressif.com/projects/esp-faq/en/latest/application-solution/esp-now.html https://www.amarinfotech.com/differences-comparisons-bluetooth-5-vs-4-2.html https://www.symmetryelectronics.com/blog/bluetooth-5-versus-bluetooth-4-2-what-s-the-difference/ https://docs.arduino.cc/tutorials/nano-esp32/esp-now/ https://github.com/espressif/arduino-esp32/blob/master/libraries/Network/src/NetworkUdp.cpp https://www.electronicdesign.com/technologies/communications/article/v42-creating-faster-more-secure-power-efficient-designspart-1

```
(documento)https://ieeexplore.ieee.org/document/9573246
}
```

## 2.5. Ejemplo de modelo tradicional publicador-broker-suscriptor con MQTT y ESP32

Tras introducir una serie de conceptos sobre el Internet de las cosas, en este apartado se detallará el ejemplo visto en el apartado de Internet de las Cosas, en la figura 2.1, que trata de un sistema de riego por aspersión.

La primera parte consiste en identificar los dispositivos a utilizar. Los sensores y actuadores deben estar conectados a un dispositivo capaz de interactuar con ellos, como puede ser una placa ESP32.

El broker utilizará una comunicación basada en colas para recibir todos los datos, integrando así una implementación del protocolo de mensajería MQTT, como puede ser Mosquitto, que únicamente se puede llevar a cabo si el broker es un ordenador, como un ordenador portátil o un single board computer. Para este ejemplo, se puede suponer que el broker será una placa Raspberry Pi.

Teniendo las placas ESP32 y la Raspberry Pi con Mosquitto, es necesario establecer la comunicación entre ambos. Una de las opciones a evaluar que sea compatible con ambas placas es Bluetooth, pero no es adecuado para una comunicación a larga distancia, ni para transmitir datos en tiempo casi real debido a su su baja velocidad, y tampoco es compatible con Mosquitto. Por lo tanto, se debe utilizar Wi-Fi para que las placas ESP32 publiquen los datos que generen y se suscriban a las órdenes y configuraciones mediante las colas adecuadas de Mosquitto. En este caso, se deberá desplegar una serie de puntos de acceso y routers para formar una red LAN o MAN que dote de Internet a todos los dispositivos.

Finalmente, se configuran las placas ESP32 para que, con la ayuda de alguna librería de código, puedan interactuar con las colas de Mosquitto, y también el broker para poder interactuar con el servidor.

La implementación de este sistema de riego presenta una serie de inconvenientes. En cuanto a inversión de dinero y tiempo, implica el despliegue de numerosos puntos de acceso y routers, así como su configuración y mantenimiento. Además, se debe asegurar una buena señal al aire libre, evaluar la zona donde se instalarán y proveer más baterías en los dispositivos que formen esta red.

Todos los dispositivos están conectados en la misma red, por lo que un tráfico alto de datos o una interrupción en el servicio puede provocar una congestión de la red y un mal funcionamiento de la misma.

Finalmente, en cuanto a seguridad, a todos los dispositivos de esta red se les otorga acceso a Internet, por lo que deben estar preparados para no sufrir un ciberataque que pueda invalidar el sistema por completo y todos los dispositivos conectados. Esto supone una mayor complejidad en el despliegue y mayor mantenimiento para evitar un riesgo significativo en la red.

Aunque la implementación de este sistema de riego se puede realizar de distintas maneras, en este ejemplo se intenta demostrar la complejidad que supone un despliegue limitado por utilizar MQTT y ESP32 en la actualidad. Pese a que, de manera independiente, ambos tienen grandes ventajas, su combinación supone una complejidad difícil de evitar.

/*TODO: añadir un nuevo esquema*/

{ https://www.prometec.net/esp32-mqtt/ (libreria de codigo) }

# Capítulo 3

# Metodología

En este apartado se deben indicar las metodologías empleadas para planificación y desarrollo del TFG, así como explicar de modo claro y conciso cómo se han aplicado dichas metodologías.

## 3.1. Guía Rápida de las Metodologías de Desarrollo del Software

A continuación, se incluye una guía rápida que puede ser de gran utilidad en la elaboración de este capítulo.

### 3.1.1. Proceso de Desarrollo de Software

El proceso de desarrollo de software se denomina también ciclo de vida del desarrollo del software (SDLC, Software Development Life-Cycle) y cubre las siguientes actividades:

- Obtención y análisis de requisitos (requirements analysis). Es la definición de la funcionalidad del software a desarrollar. Suele requerir entrevistas entre los ingenieros de software y el cliente para obtener el ‘qué’ y ‘cómo’. Permite obtener una especificación funcional del software.
- Diseño (SW design). Consiste en la definición de la arquitectura, los componentes, las interfaces y otras características del sistema o sus componentes.
- Implementación (SW construction and coding). Es el proceso de codificación del software en un lenguaje de programación. Constituye la fase en que tiene lugar el desarrollo de software.
- Pruebas (testing and verification). Verificación del correcto funcionamiento del software para detectar fallos lo antes posible. Persigue la obtención de software de calidad. Consisten en pruebas de caja negra y caja blanca. Las primeras comprueban que la funcionalidad es la esperada y para ello se verifica que, ante un conjunto amplio de entradas, la salida es correcta. Con las segundas se comprueba la robustez del código sometiéndolo a pruebas cuya finalidad es provocar fallos de software.

Esta fase también incorpora las pruebas de integración en las que se verifica la interoperabilidad del sistema con otros existentes.

- Documentación (documentation). Persigue facilitar la mejora continua del software y su mantenimiento.
- Despliegue (deployment). Consiste en la instalación del software en un entorno de producción y puesta en marcha para explotación. En ocasiones implica una fase de entrenamiento de los usuarios del software.
- Mantenimiento (maintenance). Su propósito es la resolución de problemas, mejora y adaptación del software en explotación.

### 3.1.2. Metodologías de Desarrollo Software

Las metodologías son el modo en que las fases del proceso software se organizan e interaccionan para conseguir que dicho proceso sea reproducible y predecible para aumentar la productividad y la calidad del software.

Una metodología es una colección de:

- Procedimientos: indican cómo hacer cada tarea y en qué momento,
- Herramientas: ayudas para la realización de cada tarea, y
- Ayudas documentales.

Cada metodología es apropiada para un tipo de proyecto dependiendo de sus características técnicas, organizativas y del equipo de trabajo. En los entornos empresariales es obligado, a veces, el uso de una metodología concreta (p. ej. para participar en concursos públicos). El estándar internacional ISO/IEC 12270 describe el método para seleccionar, implementar y monitorear el ciclo de vida del software.

Mientras que unas intentan sistematizar y formalizar las tareas de diseño, otras aplican técnicas de gestión de proyectos para dicha tarea. Las metodologías de desarrollo se pueden agrupar dentro de varios enfoques según se señala a continuación.

- Metodología de Análisis y Diseño de Sistemas Estructurados (SSADM, Structured Systems Analysis and Design Methodology). Es uno de los paradigmas más antiguos. En esta metodología se emplea un modelo de desarrollo en cascada (waterfall). Las fases de desarrollo tienen lugar de modo secuencial. Una fase comienza cuando termina la anterior. Es un método clásico poco flexible y adaptable a cambios en los requisitos. Hace hincapié en la planificación derivada de una exhaustiva definición y análisis de los requisitos. Son metodologías que no lidian bien con la flexibilidad requerida en los proyectos de desarrollo software. Derivan de los procesos en ingeniería tradicionales y están

enfocadas a la reducción del riesgo. Emplea tres técnicas clave:

- Modelado lógico de datos (Logical Data Modelling),
- Modelado de flujo de datos (Data Flow Modelling), y
- Modelado de Entidades y Eventos (Entity EventModelling).

- Metodología de Diseño Orientado a Objetos (OOD, Object-Oriented Design). Está muy ligado a la OOP (Programación Orientada a Objetos) en que se persigue la reutilización. A diferencia del anterior, en este paradigma los datos y los procesos se combinan en una única entidad denominada objetos (o clases). Esta orientación pretende que los sistemas sean más modulares para mejorar la eficiencia, calidad del análisis y el diseño. Emplea extensivamente el Lenguaje Unificado de Modelado (UML) para especificar, visualizar, construir y documentar los artefactos de los sistemas software y también el modelo de negocio. UML proporciona una serie diagramas de básicos para modelar un sistema:

  - Diagrama de Clases (Class Diagram). Muestra los objetos del sistema y sus relaciones.
  - Diagrama de Caso de Uso (Use Case Diagram). Plasma la funcionalidad del sistema y quién interacciona con él.
  - Diagrama de secuencia (Sequence Diagram). Muestra los eventos que se producen en el sistema y como este reacciona ante ellos.
  - Modelo de Datos (Data Model).

- Desarrollo Rápido de Aplicaciones (RAD, Rapid Application Developmnent). Su filosofía es sacrificar calidad a cambio de poner en producción el sistema rápidamente con la funcionalidad esencial. Los procesos de especificación, diseño e implementación son simultáneos. No se realiza una especificación detallada y se reduce la documentación de diseño. El sistema se diseña en una serie de pasos, los usuarios evalúan cada etapa en la que proponen cambios y nuevas mejoras. Las interfaces de usuario se desarrollan habitualmente mediante sistemas interactivos de desarrollo. En vez de seguir un modelo de desarrollo en cascada sigue un modelo en espiral (Boehm). La clave de este modelo es el desarrollo continuo que ayuda a minimizar los riesgos. Los desarrolladores deben definir las características de mayor prioridad. Este tipo de desarrollo se basa en la creación de prototipos y realimentación obtenida de los clientes para definir e implementar más características hasta alcanzar un sistema aceptable para despliegue.

- Metodologías Ágiles. "[…] envuelven un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo. Cada itera-

ción del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. Teniendo gran importancia el concepto de"Finalizado" (Done), ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de "software que funciona" (sin errores). Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. […]"}[1]

## 3.2. Proceso de Testing

Se debe indicar qué tipo de pruebas se han realizado, por ejemplo las siguientes:

- Pruebas modulares (pruebas unitarias). Su propósito es hacer pruebas sobre un módulo tan pronto como sea posible. Las pruebas unitarias que comprueban el correcto funcionamiento de una unidad de código. Dicha unidad elemental de código consistiría en cada función o procedimiento, en el caso de programación estructurada y cada clase, para la programación orientada a objetos. Las características de una prueba unitaria de calidad son: automatizable (sin intervención manual), completa, reutilizable, independiente y profesional.
- Pruebas de integración. Pruebas de varios módulos en conjunto para comprobar su interoperabilidad.
- Pruebas de caja negra.
- Beta testing.
- Pruebas de sistema y aceptación.
- …

## 3.3. Marco Tecnológico

En esta sección se enumeran las tecnologías y herramientas utilizadas en la elaboración del TFG. A continuación, se citan algunos ejemplos.

### 3.3.1. Herramientas CASE (Computer Aided Software Engineering)

Las herramientas CASE están destinadas a facilitar una o varias de las tareas implicadas en el ciclo de vida del desarrollo de software. Se pueden dividir en la siguientes categorías:

- Modelado y análisis de negocio.
- Desarrollo. Facilitan las fases de diseño y construcción.
- Verificación y validación.
- Gestión de configuraciones.
- Métricas y medidas.

[1] Fuente: Wikipedia

- Gestión de proyecto. Gestión de planes, asignación de tareas, planificación, etc.

### 3.3.2. IDE (Integrated Development Environment)

- Notepad++: https://notepad-plus-plus.org/
- Visual Studio Code: https://code.visualstudio.com/
- Atom: https://atom.io/
- GNU Emacs: https://www.gnu.org/s/emacs/
- NetBeans: https://netbeans.org/
- Eclipse: https://eclipse.org/
- QtCreator: https://www.qt.io/ide/
- jEdit: http://www.jedit.org/

### 3.3.3. Depuración

- GNU Debugger: https://www.gnu.org/s/gdb/

### 3.3.4. Repositorios y control de versiones

- Git: https://git-scm.com/
- Mercurial: [https://www.mercurial-scm.org/
- Github: https://github.com/
- Bitbucket: https://bitbucket.org/
- SourceTree: https://www.sourcetreeapp.com/

### 3.3.5. Documentación

- LaTeX: https://www.latex-project.org/
- Markdown: https://markdown.es/
- Doxygen: https://www.doxygen.nl/
- DocGen: http://mtmacdonald.github.io/docgen/docs/index.html
- Pandoc: http://pandoc.org/

### 3.3.6. Gestión y Planificación de Proyectos

- Trello: https://trello.com/
- Jira: https://es.atlassian.com/software/jira
- Asana: https://asana.com/
- Slack: https://slack.com/

- Basecamp: https://basecamp.com/
- Teamwork Proyects: https://www.teamwork.com/project-management-software
- Zoho Projects: https://www.zoho.com/projects/

# Capítulo 4

# Resultados

En los que se describen cómo se ha aplicado el método de trabajo para el caso concreto del TFG, incluyendo aquellos elementos (modelos, diagramas, especificaciones, etc.) más importantes y relevantes que se quieran hacer notar.

## 4.1. Resultados del TFG

Este apartado debe explicar cómo el empleo de la metodología permite satisfacer tanto el objetivo principal como los específicos planteados en el TFG así como los requisitos exigidos (según exposición en capítulo Objetivos).

# Capítulo 5

# Conclusiones

En este capítulo se debe incluir el juicio crítico y discusión sobre los resultados obtenidos. Si es pertinente deberá incluir información sobre trabajos derivados como publicaciones o ponencias, así como trabajos futuros, solo si estos están planificados en el momento en que se redacta el texto. Incluirá obligatoriamente la justificación de las competencias de la tecnología específica cursada por el estudiante que se han adquirido durante el desarrollo del TFG

## 5.1. Revisión de los Objetivos

En esta sección se deberá revisar en qué grado se han completado los objetivos fijados al principio del proyecto. Se deberá también indicar las posibles desviaciones de los objetivos fijados, así como de la planificación, y tratar de justificar tales desviaciones.

## 5.2. Presupuesto

Si el TFG consiste en el desarrollo e implementación de un prototipo, la memoria debe incluir el coste del prototipo considerando tanto el hardware como los recursos humanos necesarios para su desarrollo.

Cuando se tiene en cuenta la puesta en marcha de un proyecto de ingeniería, la planificación y presupuesto que se realizan de modo previo a su ejecución son críticos para gestionar los recursos que permitan alcanzar los objetivos de calidad, temporales y económicos previstos para el proyecto. Es muy importante que todas las justificaciones aportadas se sustenten no solo en juicios de valor sino en evidencias tangibles como: historiales de actividad, repositorios de código y documentación, porciones de código, trazas de ejecución, capturas de pantalla, demos, etc.

## 5.3. Competencias Específicas de Intensificación Adquiridas y/o Reforzadas

Se deberán listar aquellas competencias de la intensificación que hayan sido adquiridas y/o reforzadas con el desarrollo de este TFG, incluyendo su justificación.

# Bibliografía

[1] BibTeX format description. https://www.bibtex.org/Format/.

[2] PlantUML de un vistazo. https://plantuml.com/es/.

# Anexo A. Uso de la Plantilla

A continuación se presenta la estructura y utilización de esta plantilla.

## A.1. Configuración

La configuración de la plantilla se realiza a través del fichero *config.yaml*. Este es un fichero *yaml*, cuyos campos se describen en la Tabla A.1.

Tabla A.1: Valores del fichero *config.yaml*

| clave | valor |
|---|---|
| Cite | Citas bibliográficas a incluir en la bibliografía no referenciadas en el texto |
| Cotutor | Nombre y apellidos del co-tutor académico |
| Csl | Fichero csl con el formato de las referencias |
| Department | Departamento del tutor académico |
| Language | Lenguaje de la plantilla [english\|spanish] |
| Month | Mes de defensa del TFG |
| Name | Nombre del autor del TFG |
| Technology | Tecnología específica |
| Title | Título del TFG |
| Tutor | Nombre del tutor académico |
| Year | Año de defensa del TFG |

Un ejemplo de configuración de este fichero se muestra en el Listado A.1.

Listado A.1: Ejemplo de fichero de configuración config.yaml

```
1 Cite: @Gutwin2010GoneBut, @Rekimoto1997PickDrop
2 Cotutor: John Deere
```

```
3  Csl: input/resources/csl/acm-sig-proceedings.csl
4  Department: Ciencias de la Computación
5  Language: spanish
6  Month: Agosto
7  Name: Johny Anston
8  Technology: Sistemas de Información
9  Title: Una Aplicación para Resolver Problemas Genéricos
10 Tutor: Adam Smith
11 Year: 2023
```

## A.2. Estructura de Directorios

La plantilla tiene la estructura mostrada en la Figura A.1.

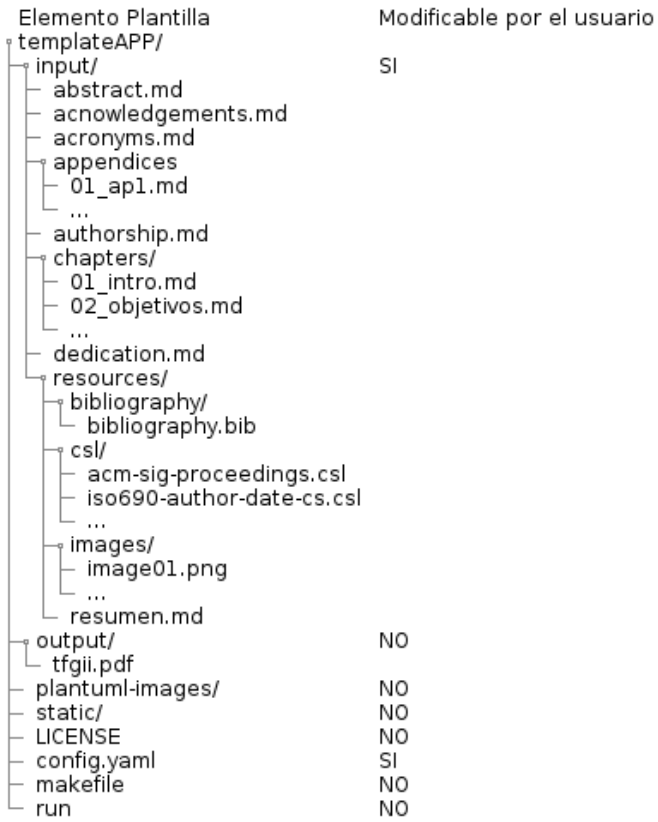


Figura A.1: Estructura de directorios de la plantilla del TFG

El documento generado por la plantilla es **output/tfgii.pdf**. Este documento se sobreescribe cada vez que se compila la plantilla. Es este documento, compilar se refiere a procesar los elementos de la plantilla para

generar el documento *pdf*.

Las carpetas y ficheros modificables por el usuario y que contienen la información propia del TFG son las siguientes (marcadas como SI en la figura A.1):

- input
- config.yaml

El resto de ficheros y directorios no deben de ser modificados por el usuario para el correcto funcionamiento de la plantilla.

### A.2.1. Carpeta input

La carpeta *input* contiene los ficheros que corresponden con las partes del TFG.

En primer lugar están los ficheros correspondientes al **abstract**, **acknowledgements**, **acronyms**, **authorship** y **dedication**. En estos, a excepción del **acronyms**, sólo hay que introducir el texto correspondiente.

El fichero **acronyms** tiene un formato yaml. En el, hay que introducir los pares *acrónimo* y *valor* separados por ":" y un espacio. Un ejemplo sería el del Listado A.2.

Listado A.2: Ejemplo de definición de acrónimos

```
1 HTML: Lenguaje de marcas de hypertexto
2 HCI: Human Computer Interaction
```

También se encuentran en el directorio *input* los directorios **appendices** y **chapters**. Contienen, respectivamente, los apéndices y los capítulos del *TFG*, que corresponden con ficheros en *markdown*. En estos ficheros hay que tener en cuenta lo siguiente:

- El fichero debe comenzar con un encabezado de primer nivel
- Los ficheros se ordenan en el documento según su orden alfabético
- Sólo se tienen en cuenta los ficheros que comienzan con dos dígitos numéricos, p.ej. 01
- Si un fichero no comienza con dos dígitos numéricos, es ignorado y no se incluye a la hora de generar el documento

### A.2.2. Carpeta resources

El directorio *resources* contiene recursos de utilidad para la generación del documento, como imágenes o bibliografía.

La bibliografía se recoge en el fichero **bibliography/bibliography.bib**. Es un fichero en formato *bibtex* [1] que contiene los elementos bibliográficos. Por ejemplo, en el Listado A.3 se encuentra definido el

elementos bibliográfico *bibtex*.

Listado A.3: Ejemplo de definición de elemento bibliográfico

```
@misc{bibtex,
  title = {BibTeX Format Description},
  howpublished = {\url{https://www.bibtex.org/Format/}},
  note = {Accessed: 2023-09-28}
}
```

El directorio **csl** contiene ficheros con diferentes formatos de bibliografía. Finalmente, el directorio **images** contiene las imágenes a utilizar en el documento.

## A.3. Elementos del Documento

Si bien la plantilla permite utilizar cualquier elemento de *Markdown* y *pandoc*, hay una serie de elementos que pueden ser de especial utilidad de cara a la realización de un TFG:

- Referencias bibliográficas
- Figuras
- Tablas
- Listados de código
- Notas al pie de página
- Acrónimos
- PlantUML
- Referencias dentro del Documento

### A.3.1. Citas Bibliográficas

La plantilla utiliza la bibliografía que se recoge en el fichero correspondiente. Para incluir una cita a un elemento bibliográfico, hay que añadir en el documento su referencia precedida del símbolo @. Por ejemplo, para añadir la cita al elemento biliográfico *bibtex* se haría de la forma indicada en el Listado A.4.

Listado A.4: Ejemplo de cita de elemento bibliográfico

```
Es un fichero en formato *bibtex* @bibtex.
```

### A.3.2. Figuras

Para incluir figuras lo hacemos añadiendo una imagen en la cual especificamos el elemento *label*, que sirve para referenciarla. Por ejemplo, el código del Listaqdo A.5 produce como resultado la Figura A.2.

Listado A.5: Ejemplo de definición de una figura

```
1 ![Logo de HTML5\label{anexo:ejemploFiguraResultado}](HTML5_sticker.png){width
    ↪ =50%}
```

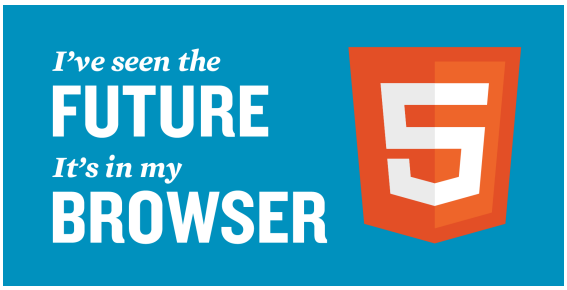


Figura A.2: Logo de HTML5

### A.3.3. Tablas

Para definir una tabla se utiliza la sintaxis mostrada en el Listado de Código A.6, cuyo resultado se muestra en la Tabla A.2.

Listado A.6: Ejemplo de definición de una tabla

```
1 | Right | Left | Default | Center |
2 |------:|:-----|---------|:------:|
3 | 12 | 12 | 12 | 12 |
4 | 123 | 123 | 123 | 123 |
5 | 1 | 1 | 1 | 1 |
```

Tabla A.2: Tabla de ejemplo

| Right | Left | Default | Center |
|---:|:---|---|:---:|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

### A.3.4. Notas al Pie de Página

Para la definición de notas al pie de página se utiliza el código mostrado en el Listado A.7, cuyo resultado se muestra al pie de esta página[1].

[1] Ejemplo de nota al pie de página.

Listado A.7: Ejemplo de definición de una nota a pie de página

```
cuyo resultado se muestra al pie de esta página[^anexo:ejemploNotaPie].

[^anexo:ejemploNotaPie]: Ejemplo de nota al pie de página.
```

### A.3.5. Acrónimos

Para hacer referencia a un acrónimo, por ejemplo HTML, hay que utilizar el código mostrado en el Listado A.8. Previamente, el acrónimo debe de haber sido definido, tal y como se muestra en el Listado A.2.

Listado A.8: Ejemplo de definición de referencia a un acrónimo

```
Para hacer referencia a un acrónimo, por ejemplo [HTML](#HTML), hay ...
```

### A.3.6. PlantUML

Esta plantilla permite la inserción de diagramas en *PlantUML* [2]. En el Listado A.9 se muestra un ejemplo básico de un diagrama de objetos en *PlantUML*, cuyo resultado se muestra en la Figura A.3.

Listado A.9: Ejemplo de código plantuml básico

```
@startuml
skinparam dpi 300
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
@enduml
```

También se puede dar formato, colores y formas, a los diagramas generados con PlantUML, tal y como se muestra en el Listado A.10. El resultado se muestra en la Figura A.4.

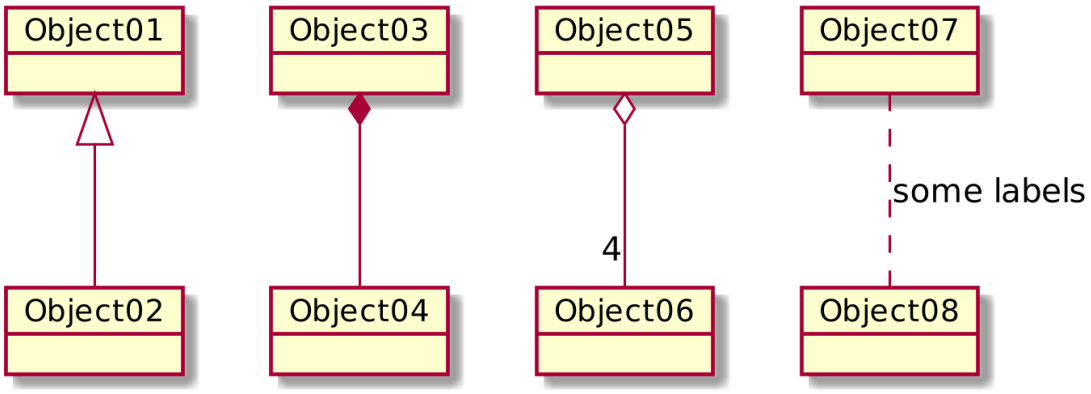


Figura A.3: Ejemplo de plantuml básico

Listado A.10: Ejemplo de código plantuml con formato

```
@startmindmap
skinparam dpi 300
<style>
mindmapDiagram {
  node {
    BackgroundColor white
    FontColor #B30033
  }
  :depth(1) {
    BackgroundColor white
    FontColor #B30033
  }
  .uclm {
    BackgroundColor white
    FontColor #B30033
  }
  .active {
    BackgroundColor #B30033
    FontColor white
  }
}
</style>
* Requisitos
** Requisitos de usuario <<active>>
```

```
*** Requisitos de dominio <<active>>
*** Requisitos de negocio <<active>>
*** Requisitos de usuario final <<active>>
** Requisitos de sistema
** Requisitos de SW/HW

@endmindmap
```

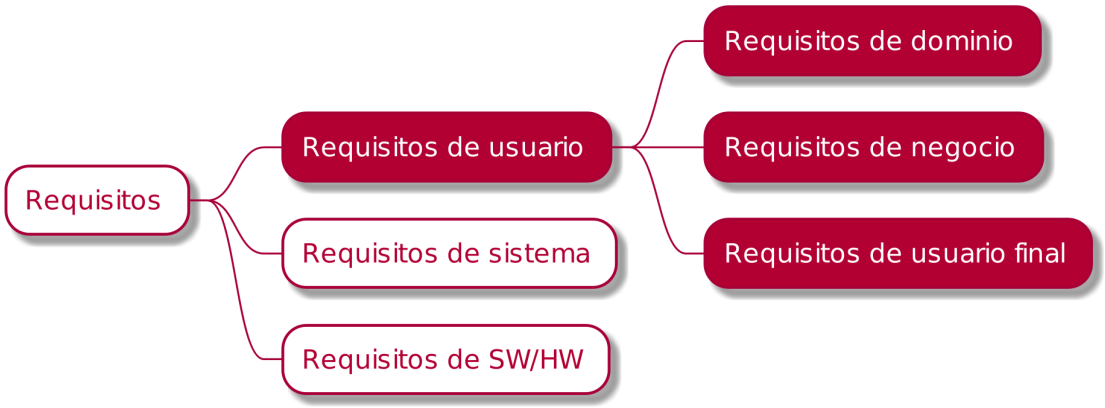


Figura A.4: Ejemplo de plantuml con formato

### A.3.7. Referencias Dentro del Documento

Para hacer una referencia a un capítulo dentro del documento se indica entre llaves el texto del enlace, seguido entre paréntesis del nombre del capítulo precedido del carácter *'#'*. El nombre del capítulo se indica en minúsculas y se sustituyen los espacios por el carácter *'-'*. Por ejemplo, si se quiere referenciar al capítulo *"Estructura de Directorios"* se utilizará el siguiente código mostrado en la Figura A.11.

Listado A.11: Referencia dentro del documento

```
Referencia a la [estructura de directorios](#estructura-de-directorios)
```

## A.4. Creación del Documento

La creación del documento se puede realizar utilizando un contenedor en *Docker* o de forma nativa en *Linux*[2]. La Figura A.5 muestra la estructura del directorio raíz de la plantilla.

A continuación, se explica cómo funciona cada una de estas aproximaciones, así como sus requisitos.

[2]Se ha probado que funciona correctamente en la distribución *Ubuntu 22.04.1*.

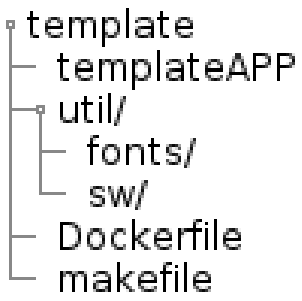


Figura A.5: Estructura de directorios raíz de la plantilla

### A.4.1. Ubuntu

Ejecutar 'make' en el directorio raíz de la plantilla.

Hay que tener instalado el siguiente software[3]:

- texlive-latex-base
- texlive-lang-spanish
- texlive-xetex
- make
- pandoc-plantuml-filter

También hay que instalar las fuentes adecuadas. Para ello crear el directorio (si no existe), copiar las fuentes y actualizarlas según el código de la Figura A.12.

Listado A.12: Instalación de las fuentes para el documento

```
$ sudo mkdir -p /usr/share/fonts/truetype/msttcorefonts/
$ sudo cp util/fonts/* /usr/share/fonts/truetype/msttcorefonts/
$ sudo fc-cache -f -v
$ sudo texhash
```

Finalmente, hay que instalar pandoc, pero en su versión 2.19.2-1. Para ello, ejecutar la orden de la Figura A.13 en el directorio raíz de la plantilla. Si se utiliza cualquier otra versión es muy probable que se tengan problemas, por ejemplo a la hora de generar imágenes con *plantuml*.

Listado A.13: Instalación de Pandoc en su versión 2.19.2-1

```
$ sudo dpkg -i util/sw/pandoc-2.19.2-1-amd64.deb
```

[3] Para instalar el software en Ubuntu hay que ejecutar *apt install nombre-paquete*

### A.4.2. Docker

Ejecutar ‘make docker’ en el directorio raíz de la plantilla.

En el caso de utilizar Docker, el único prerequisito es tenerlo instalado[4].

[4] https://docs.docker.com/engine/install/ubuntu/

# Anexo B. Título del Anexo II

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh

sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

## B.1. Una sección

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu

eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.