

Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Implementación de un sistema de realidad aumentada para visualización de datos de un entorno de sensores IoT

Autor: Enrique Payán Rodríguez

Tutores: Irene Fondón García

Juan Antonio Becerra González

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020



Trabajo Fin de Grado
Ingeniería de Telecomunicación

Implementación de un sistema de realidad aumentada para visualización de datos de un entorno de sensores IoT

Autor:

Enrique Payán Rodríguez

Tutores:

Irene Fondón García

Profesora Titular de Universidad

Juan Antonio Becerra González

Profesor Sustituto Interino

Dpto. de Teoría de la Señal y Comunicaciones

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Implementación de un sistema de realidad aumentada para visualización de datos de un entorno de sensores IoT

Autor: Enrique Payán Rodríguez

Tutores: Irene Fondón García
Juan Antonio Becerra González

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

EN primer lugar, como no podía ser de otra manera, me gustaría agradecer a mi madre por todo lo que me ha dado y por preocuparse por mí durante toda la vida, especialmente estos últimos años, en los que se ha hecho cargo de tanto y ha sabido sacar fuerzas de donde no las había para salir adelante. Gracias por todas las veces que te has tenido que levantar un poco antes para acercarme a la universidad antes de ir a trabajar y por todas las veces que me has recogido para que no volviera en bus. Gracias por preocuparte de cómo me iban los estudios y por decirme que no me amargara, que fuera a mi ritmo y que al final lo conseguiría. Pues si lees esto significa que lo he conseguido y que ha sido gracias a ti en gran medida.

También me gustaría agradecer al resto de mi familia y amigos que han estado ahí cada vez que les he necesitado y han sabido escucharme. En especial, gracias a esas personas que conocí al principio de esta aventura, que hoy en día considero amigos y espero que así sea por mucho tiempo.

Por otra parte, quería escribir un párrafo de agradecimiento a una persona muy especial durante estos años, una persona a la que le puse el mote de “la de Airbus”. Te estoy muy agradecido por todo lo que has hecho por mí desde el primer año de carrera. Por tu ayuda en cada una de las asignaturas, por esas horas muertas en la escuela en las que tanto hemos reído cuando no estábamos de exámenes y que tanto hemos sufrido cuando quedaban dos semanas para estos, por todas las cosas buenas que me has aportado como persona. En definitiva, sólo puedo decirte que muchas gracias por estar ahí siempre que te he necesitado.

Por último, quería agradecer a mis tutores por darme la oportunidad de realizar este proyecto, el cual me ha resultado apasionante. Gracias a Irene, por proponerme esta idea, que al principio me sonaba como una locura pero que hoy puedo decir que se ha hecho realidad, y gracias a Juan Antonio, por aceptar formar parte de este trabajo y por aguantar tantos correos con cientos de dudas.

Enrique Payán Rodríguez

Sevilla, 2020

Después de la innovación que supuso lo que se conoce como la *World Wide Web*, la evolución de Internet conlleva a que el siguiente paso sea el despliegue de dispositivos por todo el mundo, que sean capaces de conectar e intercambiar datos entre ellos. Esto hará que el internet de las cosas, o IoT, se convierta en una realidad y que, con esta nueva tecnología, se facilite la vida de las personas.

Hoy en día, en un mundo donde casi todas las personas poseen más de un producto inteligente, la importancia de tener todo interconectado coge más peso cuando se puede facilitar la vida a los usuarios de dichos dispositivos. Es por esto por lo que el IoT ha cobrado tanta importancia estos últimos años. La facilidad que otorgan los dispositivos a interconectar los unos con los otros y la eficacia con la que proporcionan datos útiles a las personas ha hecho que, conforme el avance de la tecnología lo permite, los dispositivos nos proporcionen un mejor nivel de vida.

En el presente trabajo se propone una arquitectura ARIoT, que fusiona dos tecnologías: el IoT y la realidad aumentada. Esta arquitectura tiene el potencial de aproximar, de forma dinámica, a usuarios y objetos que conforman la red IoT. La arquitectura ARIoT consta de una red de sensores en la que se implementa un sistema de realidad aumentada para hacer posible la visualización de los datos que recogen dichos sensores, los cuales están configurados para captar datos del entorno.

Abstract

After the innovation that entailed the *World Wide Web*, the Internet's evolution entails that the next step is the deployment of devices around the world that are of connecting and exchanging data between them. This will make the Internet of Things, or IoT, a reality and, with this new technology, people's lives will be easier.

Nowadays, in a world where almost everyone owns more than one smart product, the importance of having everything interconnected becomes more important when it can make life easier for users of such devices. This is why the IoT has acquired significant relevance in recent years. The ease of devices to interconnect with each other and the efficiency with which they provide useful data to people has meant that, as technology advances, devices provide us with a better standard of living.

In the present project an architecture of IoT is proposed, which merges two technologies: IoT and augmented reality. This architecture has the potential to dynamically approximate users and objects that knock the IoT network into shape. The ARIoT architecture consists of a network of sensors, in which an augmented reality system is implemented to make it possible to visualize the data collected by said sensors, which are configured to capture data from the environment.

Índice Abreviado

Agradecimientos	IX
Resumen	XI
Abstract	XIII
Índice Abreviado	XV
Índice	XVII
Índice de Figuras	XIX
Índice de Tablas	XXI
Notación	XXIII
1 Introducción	1
1.1. <i>Objetivos</i>	1
1.2. <i>Alcance</i>	2
1.3. <i>Estructura</i>	2
2 Material utilizado	3
2.1. <i>Hardware</i>	3
2.2. <i>Software</i>	6
3 Un mundo conectado: IoT	9
3.1. <i>Introducción</i>	9
3.2. <i>Arquitectura básica</i>	11
3.3. <i>Protocolos de comunicación</i>	13
3.4. <i>Tecnologías de conectividad</i>	18
4 Arquitectura ARIoT	25
4.1. <i>Componentes de la arquitectura ARIoT</i>	25
4.2. <i>Realidad aumentada: concepto y métodos</i>	27
4.3. <i>Capa Middleware: necesidad y tipos de arquitectura</i>	30
5 Desarrollo práctico del trabajo	35
5.1. <i>Recolección e inserción de datos</i>	35
5.2. <i>Desarrollo de la aplicación móvil</i>	39
5.3. <i>Implementación final del sistema</i>	42
6 Conclusiones y líneas futuras	47
Anexo A Código Python para introducir medidas en la base de datos	49
Anexo B Código PHP para publicar en una página web .PHP los datos de la BD	51
Anexo C Código C# para obtener los datos de la página PHP	53
Referencias	55

Índice

Agradecimientos	IX
Resumen	XI
Abstract	XIII
Índice Abreviado	XV
Índice	XVII
Índice de Figuras	XIX
Índice de Tablas	XXI
Notación	XXIII
1 Introducción	1
1.1. <i>Objetivos</i>	1
1.2. Alcance	2
1.3. Estructura	2
2 Material utilizado	3
2.1. <i>Hardware</i>	3
2.1.1. Raspberry Pi	3
2.1.2. Módulo Sense Hat	5
2.2. <i>Software</i>	6
2.2.1. Unity3D y Vuforia SDK	6
3 Un mundo conectado: IoT	9
3.1. <i>Introducción</i>	9
3.2. <i>Arquitectura básica</i>	11
3.3. <i>Protocolos de comunicación</i>	13
3.3.1. Capa física y de enlace	14
3.3.1.1 IEEE 802.15.4, ZigBee	14
3.3.1.2 IEEE 802.15.1, Bluetooth	14
3.3.1.3 LTE/LTE-A y 5G	15
3.3.2. Capa de red	15
3.3.2.1 RPL	15
3.3.2.2 6LoWPAN	16
3.3.3. Capa de aplicación	16
3.3.3.1 MQTT	17
3.3.3.2 CoAP	17
3.3.3.3 AMQP	17
3.3.3.4 XMMP	17
3.3.3.5 DSS	18
3.4. <i>Tecnologías de conectividad</i>	18
3.4.1. Tecnologías de largo alcance o celulares	19
3.4.1.1 LoRa	20

3.4.1.2	SigFox	20
3.4.1.3	LTE-M	21
3.4.1.4	NB-IoT	21
3.4.2.	Tecnologías de corto alcance o no celulares	21
3.4.2.1	WiFi	22
3.4.2.2	ZigBee	22
3.4.2.3	Bluetooth	23
4	Arquitectura ARIoT	25
4.1.	<i>Componentes de la arquitectura ARIoT</i>	25
4.2.	<i>Realidad aumentada: concepto y métodos</i>	27
4.2.1.	Métodos existentes para la realidad aumentada con el uso de marcadores	28
4.2.2.	Métodos “markerless” para la realidad aumentada	29
4.3.	<i>Capa Middleware: necesidad y tipos de arquitectura</i>	30
4.3.1.	Arquitecturas existentes para el <i>Middleware</i>	31
4.3.2.	El conocimiento del contexto dentro de la capa <i>Middleware</i>	32
5	Desarrollo práctico del trabajo	35
5.1.	<i>Recolección e inserción de datos</i>	35
5.1.1.	Creación del servidor LAMP en la Raspberry Pi controladora	36
5.1.2.	Configuración de la base de datos	37
5.1.2.1	Creación de la base de datos	37
5.1.2.2	Creación de la tabla de datos	37
5.1.3.	Medición e inserción de los datos en la base de datos	38
5.1.3.1	Configuración de la librería EasyDB	39
5.1.3.2	Inserción de datos en la base de datos	39
5.2.	<i>Desarrollo de la aplicación móvil</i>	39
5.2.1.	Vuforia SDK: creación de los marcadores	40
5.2.2.	Vuforia: creación de la base de datos de marcadores	40
5.2.3.	Unity3D: modelado de la interfaz de realidad aumentada	41
5.3.	<i>Implementación final del sistema</i>	42
5.3.1.	Código PHP intermediario	43
5.3.2.	Código para obtener los valores medidos	43
5.3.3.	Pruebas realizadas	44
6	Conclusiones y líneas futuras	47
Anexo A	Código Python para introducir medidas en la base de datos	49
Anexo B	Código PHP para publicar en una página web .PHP los datos de la BD	51
Anexo C	Código C# para obtener los datos de la página PHP	53
	Referencias	55

Índice de Figuras

2.1	Logotipo de Raspberry Pi	3
2.2	Conexionado físico de la RaspBerry Pi 4	4
2.3	Módulo Sense Hat	5
2.4	Módulo SenseHat colocado sobre la RaspBerry Pi 4	6
2.5	Logotipo de Unity3D	6
2.6	Interfaz de Unity3D	7
2.7	Logotipo de Vuforia SDK	8
2.8	Ejemplo de múltiples marcadores de reconocimiento	8
3.1	Internet de las Cosas	9
3.2	Perspectivas del IoT	10
3.3	Arquitecturas del sistema IoT propuestas	11
3.4	Arquitectura SOA	12
3.5	Protocolos del IoT propuestos por los distintos organismos	13
3.6	Quinta Generación de comunicaciones móviles	15
3.7	Escenarios con protocolos de la capa de red	16
3.8	Tecnologías de conectividad según el alcance	18
3.9	Tecnologías de conectividad según si la tecnología es o no celular	18
3.10	Aplicaciones de las redes LPWAN	19
3.11	Tx/Rx en las distintas clases de dispositivos	20
3.12	LTE para aplicaciones IoT en redes LPWAN	21
3.13	Bandas de operación de ZigBee	22
3.14	Topologías disponibles en ZigBee	23
3.15	Ejemplo de <i>Scatternet</i>	24
3.16	Modos de los dispositivos dentro de la <i>piconet</i>	24
4.1	Arquitectura ARIoT propuesta	25
4.2	Capas de la arquitectura ARIoT	26
4.3	Método “ <i>pattern</i> ”	28
4.4	Método “ <i>outline</i> ”	28
4.5	Método “ <i>surface</i> ”	29
4.6	Método “ <i>location</i> ”	29
4.7	Principales sistemas de <i>Middleware</i> existentes para el IoT	31
4.8	Arquitectura <i>Middleware</i> basada en servicios	31
4.9	Arquitectura <i>Middleware</i> basada en la nube	32
4.10	Arquitectura <i>Middleware</i> basada en actuadores	32

5.1	Esquema de recolección e inserción de datos	35
5.2	Entorno de trabajo para la creación del servidor LAMP	36
5.3	Arquitectura de un servidor LAMP	37
5.4	Tabla de temperaturas del nodo 1 del sistema	38
5.5	Interfaz de Sense Hat Emulator	39
5.6	Marcadores utilizados para la aplicación móvil	40
5.7	Base de datos de imágenes de Vuforia	41
5.8	Jerarquía de la escena AR creada	41
5.9	Objetos de la escena AR creada	41
5.10	Esquema de la implementación final del sistema	42
5.11	Nodo 1 del sistema en la aplicación de AR	44
5.12	Nodo 2 del sistema en la aplicación de AR	44
5.13	Diferenciación de los nodos por parte de la aplicación de AR	45

Índice de Tablas

3.1	Comparativa de algunos protocolos de capa física	14
3.2	Comparativa de algunos protocolos de capa de aplicación	16
3.3	Comparativa de las tecnologías LPWAN	19
3.4	Comparativa de las tecnologías de corto alcance	21
3.5	Comparativa de los estándares de Bluetooth	23
5.1	Nodos del sistema	44

Notación

IoT	Internet de las cosas
AR	Realidad aumentada (Augmented Reality)
SO	Objetos inteligentes (Smart Objects)
IDE	Entorno de desarrollo integrado (Integrated Development Environment)
SDK	Kit de desarrollo software (Software Development Kit)
CA	Conocimiento del contexto (Context-Awareness)
GPS	Sistema de posicionamiento Global (Global Positioning System)
BLE	Bluetooth de baja energía (Bluetooth Low Energy)
SQL	Lenguaje de consulta estructurado (Structured Query Language)
IEEE	Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers)
ETSI	Instituto Europeo de Normas de Telecomunicaciones (European Telecommunication Standards Institute)
ITU	Unión Internacional de Telecomunicaciones (International Telecommunication Union)
IETF	Grupo de Trabajo de Ingeniería de Internet (Internet Engineering Task Force)
SIG	Grupo de Interés Especial (Special Interest Group)
M2M	Máquina a Máquina (Machine to Machine)
MAC	Control de Acceso al Medio (Media Access Control)
LR-WPAN	Redes Inalámbricas de Área Personal de Bajo Rango (Low Range – Wireless Personal Area Network)
TDMA	Acceso Múltiple por División en el Tiempo (Time Division Multiple Access)
CDMA/CA	Acceso Múltiple por Detección de Portadora y Prevención de Colisiones (Carrier Sense Multiple Access with Collision Avoidance)
H2H	Humano a Humano (Human to Human)
5G	Quinta Generación de comunicaciones móviles (Fifth Generation)
LTE	Evolución a Largo Plazo (Long Term Evolution)
LTE-A	Cuarta Generación de comunicaciones móviles o Evolución a Largo Plazo Avanzada (Advance Long Term Evolution)
DSSS	Espectro Ensanchado por Secuencia Directa (Direct Sequence Spread Spectrum)
FHSS	Espectro Ensanchado por Saltos de Frecuencia (Frequency Hopping Spread Spectrum)
DL	Enlace descendente (DownLink)
UL	Enlace ascendente (UpLink)
OFDMA	Acceso Múltiple por División de Frecuencias Ortogonales (Orthogonal Frequency-Division Multiple Access)
MIMO	Múltiple entrada, Múltiple Salida (Multiple Input Multiple Output)
ISM	Bandas de radio industriales, científicas y médicas

QPSK	Modulación por Desplazamiento en Fase en Cuadratura (Quadrature Phase Shift Keying)
GMSK	Modulación por Desplazamiento Mínimo Gaussiano (Gaussian Minimum Shift Keying)
ISM	Bandas de radio industriales, científicas y médicas
LLN	Red de baja potencia y pérdidas (Low-Power and Lossy Network)
RPL	Enrutamiento para redes LLN (Routing Protocol for LLNs)
6LoWPAN	IPv6 sobre Redes Inalámbricas de Área Personal (IPv6 Low-power Wireless Personal Area Networks)
E2E	Extremo a Extremo (End-to-End)
MQTT	Message Queue Telemetry Transport
IBM	International Business Machines Corporation
TCP	Protocolo de Control de Transmisión (Transport Control Protocol)
SSL	Secure Sockets Layer
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
QoS	Calidad de Servicio (Quality of Service)
CoAP	Constrained Application Protocol
UDP	Protocolo de Datagramas de Usuario (User Datagram Protocol)
HTTP	Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol)
AMQP	Advanced Message Queuing Protocol
XMPP	Extensible Messaging and Presence Protocol
DDS	Data Distribution Service
OMG	Object Management Group
LPWAN	Red de Área Amplia de Baja Potencia (Low-Power Wide-Area Network)
OWC	Comunicaciones Ópticas Inalámbricas (Optical Wireless Communications)
CSS	Chirp Spread Spectrum
LoRaWAN	Red de Área Extensa de Largo Alcance (Long Range Wide-Area Network)
SINR	Signal-to-Interferencia-plus-Noise Ratio
Tx/Rx	Transmisor/Receptor
DQPSK	Modulación por desplazamiento de fase en cuadratura de polarización dual (Dual-polarization Quadrature Phase Shift Keying)
bps	Bits por segundo (Bits per second)
Hz	Hercio (Hertz)
SC-FDMA	Acceso Múltiple por División en Frecuencia de Portadora Única (Single Carrier Frequency Division Multiple Access)
QAM	Modulación por Amplitud en Cuadratura
WLAN	Red Inalámbrica de Área Local (Wireless Local Area Network)
CSMA/CA	Acceso Múltiple por Detección de Portadores y Prevención de Colisiones (Carrier Sense Multiple Access with Collision Avoidance)
WSN	Red Inalámbrica de Sensores (Wireless Sensor Network)
RSSI	Received Signal Strength Indicator
eSCO	Conexiones Síncronas Extendidas (Extended Synchronous Connections)
EDR	Velocidad de Datos Mejorada (Enhanced Data Rate)
HS	Alta velocidad (High Speed)
IA	Inteligencia Artificial

1 INTRODUCCIÓN AL IoT Y A LA AR

Me encanta la tecnología que me hace crecer.

Sundar Pichai.

En el mundo en el que vivimos, los dispositivos de consumo electrónico han tenido tal evolución que la interacción entre humanos y máquinas es más natural y efectiva [1]. El internet de las cosas se puede definir como una tecnología en la que se realizan comunicaciones máquina a máquina, es decir, una interconexión de dispositivos y objetos a través de la red, interaccionando e intercambiando datos entre ellos. El desembarco del IoT en nuestras vidas supone expandir esta interacción, proporcionando aplicaciones y servicios que permitirán integrar una mejora en la relación entre humanos y dispositivos en nuestro día a día. Es decir, el IoT viene a facilitar nuestra vida diaria incorporando lo que se conocen como objetos inteligentes o SOs (del inglés, *Smart Objects*).

Estos SOs son capaces de comunicarse entre sí y, además, están conectados a internet, lo que supone que pueden obtener datos y compartirlos entre sí para aportar información a los usuarios que se encuentren en el área de operación de dicho SO.

En este contexto, donde el IoT proporciona funcionalidades que hacen que nuestras vidas sean mejores, esta nueva tecnología ha cobrado una gran importancia en los últimos años. Sin embargo, el primer paso para que el usuario acepte la integración de esta nueva tecnología en su vida, es la aceptación de la misma. Es en este sentido donde proporcionar al usuario interfaces amigables se vuelve indispensable. De entre todas las posibles opciones para la interacción natural entre el usuario y los dispositivos inteligentes, una interfaz AR haría posible una percepción muy positiva del IoT. El AR es una tecnología que permite la visualización del mundo real mediante el uso de dispositivos con la adición de información gráfica en la pantalla de los mismos. La integración de la AR con el IoT conlleva que este último se hiciera atractivo para los usuarios, interesándose por esta tecnología y adquiriendo nuevos dispositivos inteligentes compatibles con el IoT.

1.1 Objetivos

El objetivo principal del presente proyecto es la implementación de una red de sensores y la representación de los datos de cada uno de estos, mediante una aplicación móvil de realidad aumentada, de los datos recogidos mediante un sensor, así como la implementación de una red de sensores y la representación de los datos de cada uno de estos. Los datos capturados por los sensores se refieren a la temperatura del entorno. Esta información se introduce en una base de datos para su posterior visualización.

En cuanto a la interacción entre la red de sensores y la realidad aumentada, ésta se puede dividir en tres partes: la recolección e inserción de datos en la base de datos, la lectura y visualización de datos por parte de la aplicación de realidad aumentada y, por último, la diferenciación de nodos de la red e implementación de esta.

Para la primera parte, se hace uso de los sensores, los cuales son dispositivos Raspberry Pi a los que se les añade el módulo Sense Hat para la captación de los datos. Una vez tenemos la medida, se utiliza un script para la inserción del dato de la medida en la base de datos, la cual se trata de MySQL.

Para la segunda parte, se utiliza el programa de Unity3D para el modelado de la interfaz y la funcionalidad de la aplicación móvil y para la realidad aumentada se usa el kit de desarrollo software de Vuforia. Uniendo estas piezas, obtenemos la visualización de los datos.

Por último, para la implementación de la red de sensores se hace uso del propio kit de Vuforia, en el que, mediante la introducción de distintos marcadores, es posible la diferenciación de los distintos nodos de la red y la obtención de la última medida de cada nodo.

1.2 Alcance

Este trabajo propone un alcance que incluirá cuestiones referentes al IoT, como el paradigma de éste, las tecnologías esenciales en las que se basa, las arquitecturas posibles de un sistema IoT y sus protocolos, en el que se integra la componente de AR, y las capas de ésta. Además, se incluye el desarrollo de una interfaz de aplicación móvil que familiarice al usuario con el IoT. Por último, se implementa un sistema de sensores, en el que la componente de AR será capaz de diferenciar a estos mediante reconocimiento de imágenes.

1.3 Estructura

La estructura de la que consta el presente trabajo se puede dividir en un total de 6 capítulos, siendo éste primero una introducción en la que se exponen los objetivos, alcance y la estructura para conseguir el desarrollo del trabajo. El segundo capítulo muestra las herramientas, tanto *Hardware* como *Software*, que se utilizarían para el desarrollo de este.

En el capítulo 3 se muestra una introducción teórica sobre el IoT, que abarca desde el paradigma del mismo, pasando por las distintas arquitecturas que existen para esta tecnología, los protocolos del IoT y terminando con algunas de las tecnologías de conectividad existentes en la actualidad.

El capítulo 4 trata sobre un estudio de la arquitectura ARIoT propuesta, en la que se desarrollará todos los componentes de esta: introducción a la Realidad Aumentada y métodos de esta, capa intermedia o *Middleware* y las tecnologías que hacen posible la interacción de AR con el dispositivo móvil.

Por otro lado, en el capítulo 5 se propone el desarrollo del trabajo seguido para la implementación de la red de sensores, dividiendo dicho desarrollo en 3 partes: recolección e inserción de las medidas en la base de datos, modelado de la interfaz móvil y representación de los datos medidos mediante la misma y, por último, la implementación de la red de sensores.

En último lugar, el capítulo 6 contendrá una conclusión sobre el trabajo realizado y una serie de líneas futuras del mismo.

2 MATERIALES Y MÉTODOS

El verdadero progreso es el que pone a la tecnología al alcance de todos.

Henry Ford.

En este apartado se va a abordar todo el material utilizado, tanto *hardware* como *software*, para el desarrollo del proyecto. Se presentará cada uno de los dispositivos utilizados, así como las herramientas necesarias para que éstos realicen las funciones indispensables para alcanzar la meta del proyecto: implementar un sistema de realidad aumentada para la visualización de datos de una red de sensores IoT.

Se entiende como *hardware*, según la RAE, todo conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático. En el caso de este proyecto, el *hardware* estará formado por todos los dispositivos que, en nuestro caso, serán los sensores del sistema. Estos dispositivos serán los encargados de la recolección de datos que se visualizarán posteriormente mediante realidad aumentada.

Por otra parte, se entiende como *software*, según la RAE, todo el conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. Esto es, en el caso de este proyecto, los programas que nos ayudarán a la inserción de datos en la base de datos y los que se encargarán de compilar y crear la aplicación para los dispositivos móviles, entre otras cosas.

2.1. Hardware

2.1.1 Raspberry Pi

Raspberry Pi es un ordenador funcional de pequeñas dimensiones y de bajo coste, siendo uno de los dispositivos más utilizados en los últimos años para implementar proyectos relacionados con el IoT.

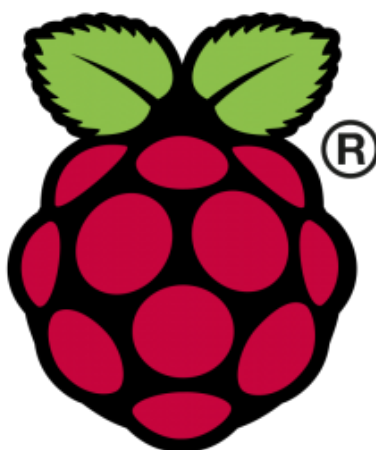


Figura 2.1. Logotipo de Raspberry Pi.

Este dispositivo será el utilizado para hacer el papel de sensor del sistema, por lo que a continuación se comentará su arquitectura física y conexionado.

Arquitectura

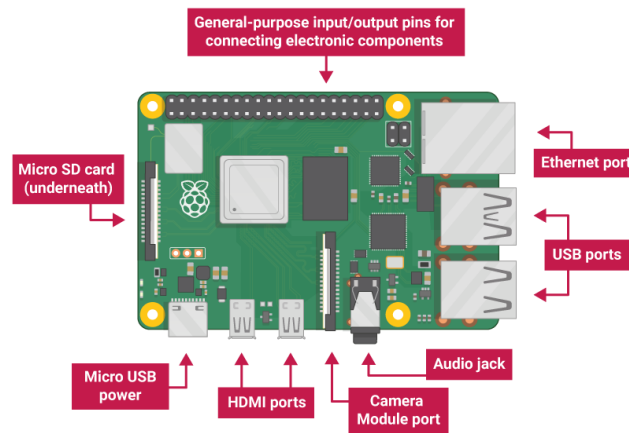


Figura 2.2. Conexión de la Raspberry Pi 4 [2].

La Raspberry posee múltiples puertos y conexiones que hace que sea un dispositivo muy potente a la hora de usarla en un sistema de IoT. Cabe destacar las siguientes conexiones, [2]:

- **USB (Universal Serial Bus):** se trata de unos puertos utilizados para el conexionado de periféricos externos (teclado, ratón, cámara digital, etc.) compatibles. En la versión 4 de la Raspberry existen simultáneamente dos tipos de USB en el dispositivo, USB 2.0 y USB 3.0, mientras que en la versión 3 sólo incluye USB 2.0. Esto hace que la versión más reciente sea una mejor alternativa, debido a que con los USB 3.0 se consiguen una mayor velocidad de transferencia de datos.
- **Puerto Ethernet:** es el puerto que se encarga de que la Raspberry esté conectada a Internet, también conocido como puerto de red. El tipo de conector que se utiliza es el estándar para las conexiones de red, es decir, el conector RJ45.
- **Jack AV 3.5 mm:** es un puerto utilizado tanto para la toma de auriculares como para conexiones para transportar una señal de vídeo a pantallas compatibles.
- **Conector de cámara:** permite el uso de un módulo diseñado específicamente para la Raspberry y que también se conoce como *CSI* o Interfaz Serie para Cámaras.
- **Puertos micro-HDMI (High Definition Multimedia Interface):** transportan señales de audio y vídeo de excelente calidad, de ahí que se denominen como Alta Definición (HD). Al poseer la Raspberry de dos de estos puertos, puede conectarse ambos a la vez, pudiendo utilizar dos pantallas al mismo tiempo.
- **Puerto de carga USB:** conecta a la Raspberry a la fuente eléctrica para que esta funcione y se trata de un USB-C.
- **Conector de tarjeta microSD:** se encarga de almacenar todos los documentos y datos que se almacenan en la Raspberry, es decir, se trata de una tarjeta de memoria. En ella se instalará el sistema operativo escogido por el usuario, sin el cual la Raspberry no podría funcionar.
- **Pines GPIO (general-purpose input/output):** utilizados para la conexión de *Hardware* adicional, como, por ejemplo, iluminación LED, botones, sensores, etc.

Por motivos excepcionales, ocasionados por la COVID-19, no fue posible realizar este proyecto con unos dispositivos físicos. Es por ello por lo que se pasó a utilizar una Raspberry en formato virtual, es decir, sólo se poseía el sistema operativo, mediante el uso del *software* VMWare.

2.1.2 Módulo Sense Hat

Se trata de un módulo que se integra a la Raspberry Pi y que resulta poderoso y multifuncional por la información que es capaz de transmitir.

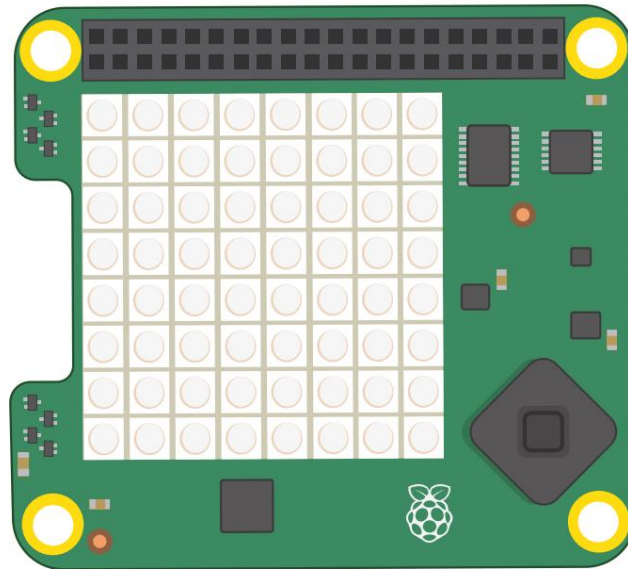


Figura 2.3. Módulo Sense Hat [2].

Además de un panel 8x8 de LEDs RGB programables, incluye un controlador joystick y 6 tipos de sensores, entre los que se encuentran [2]:

- **Sensor de giroscopio:** utilizado para cambiar la velocidad angular, haciendo un seguimiento de la gravedad de la Tierra. Se utiliza para conocer cuando se está rotando el sensor y a que velocidad lo hace, todo esto relativo a la superficie de la Tierra.
- **Acelerómetro:** es parecido al anterior, pero se encarga de medir la aceleración en múltiples direcciones. Combinando este sensor con el anterior, podemos obtener información sobre a que está apuntando el sensor y hacia qué dirección se está moviendo.
- **Magnetómetro:** se encarga de medir la fuerza del campo magnético y también ayuda a realizar un seguimiento de los movimientos del módulo, es decir, midiendo el campo magnético natural de la Tierra, este sensor es capaz de determinar la dirección del norte magnético y saber el movimiento que sufre el módulo. También es utilizado para detectar objetos metálicos y campos magnéticos. Este sensor está colocado en el mismo chip junto al acelerómetro y al giroscopio.
- **Sensor de humedad:** mide la cantidad de vapor de agua existente en el aire, es decir, la humedad relativa. Se trata de una medida que se proporciona en tanto por ciento, desde el 0% hasta el 100%, y que puede ser utilizada para detectar cuando puede llover en el área donde se coloca el sensor.
- **Sensor de presión barométrica:** se conoce como barómetro y mide la presión del aire. Este sensor se encarga de realizar un seguimiento de la altura referente al nivel del mar a la que se encuentra el sensor, dependiendo de la presión de la posición en la que se encuentre.
- **Sensor de temperatura:** mide cómo de frío o caluroso se encuentra el ambiente donde se encuentra dicho sensor. A esta medida también afecta la temperatura a la que se encuentre el módulo.

Como se ha comentado anteriormente, este módulo de sensores es un complemento perfecto para la Raspberry y, por ese motivo, será el utilizado para obtener datos del entorno, en el cual se colocará el sensor. El dato seleccionado para realizar este trabajo ha sido la temperatura, puesto que es una medida al uso para todo tipo de usuarios.

En la siguiente figura, se muestra cómo se complementa el módulo Sense Hat con la Raspberry Pi. Para ello, se atornilla cada esquina del módulo a la Raspberry y quedaría de la siguiente forma:

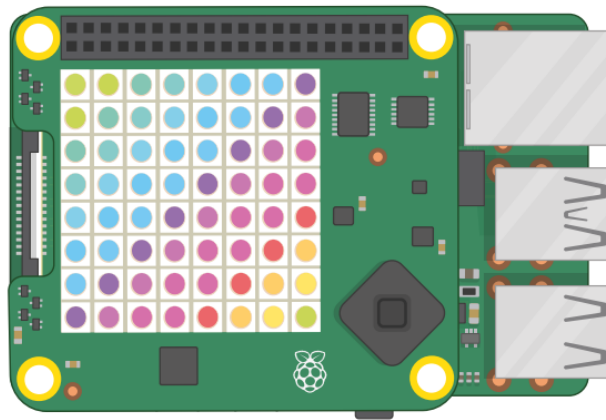


Figura 2.4. Módulo Sense Hat colocado en la Raspberry Pi 4 [2].

Debido a las circunstancias excepcionales, se pasó a utilizar un emulador de este módulo, incluido en el sistema operativo de la Raspberry Pi y conocido como *Sense Hat Emulator*. Este emulador proporciona las mismas funcionalidades y medidas, dando la posibilidad al usuario de establecer los valores de éstas. La interfaz que presenta esta herramienta se mostrará en el capítulo 5, junto al desarrollo práctico del trabajo.

2.2. Software

2.2.1 Unity3D y Vuforia SDK

Unity3D es un motor gráfico de videojuegos multiplataforma desarrollado por *Unity Technologies*, el cual contiene un equipo de desarrollo integrado o *IDE*, que facilita al desarrollador o programador el desarrollo de software, y posee la capacidad de ser desplegado en numerosas plataformas [3].



Figura 2.5. Logotipo de Unity3D.

Como ya se ha comentado, la gran funcionalidad que nos ofrece esta herramienta software es la capacidad de desplegar las aplicaciones que se desarrollen en un gran número de plataformas con facilidad. De esta manera, se puede desarrollar aplicaciones para Windows, OS X, iOS, Android y PlayStation entre otros.

Por otra parte, Unity3D nos permite elegir entre tres lenguajes de programación: JavaScript, C# o Boo. En nuestro caso, para el desarrollo de la aplicación de AR mediante Vuforia SDK, se utilizará el lenguaje C# para los *scripts*, dado que se trata de un lenguaje bien estructurado y que es menos propenso a errores, a diferencia que JavaScript por ejemplo.

La interfaz que nos ofrece esta herramienta se muestra en la siguiente figura:

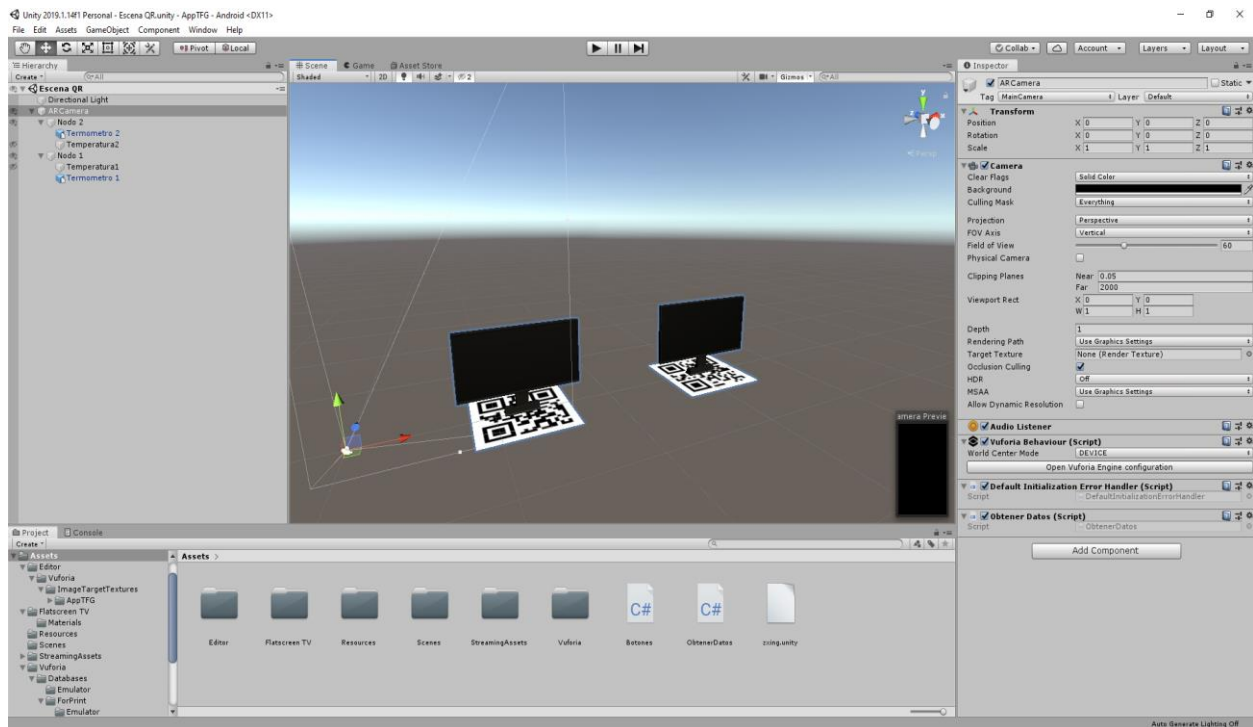


Figura 2.6. Interfaz de Unity3D.

Cabe destacar cinco bloques dentro de la interfaz del programa, los cuales son, según [4]:

1. La vista de escena o **Scene View**, que se trata del espacio en el que se trabajan sobre los objetos de juego de nuestro proyecto o aplicación. Se utiliza principalmente para seleccionar y establecer todos los objetos de juego que queramos que aparezcan en la ventana de la aplicación, así como las cámaras y sus ángulos de visión, la iluminación y todo lo demás que conforma la escena.
2. La vista de juego o **Game View**, en la que, tras haber terminado la escena que se quiera implementar, se puede observar el funcionamiento de la escena en cuestión, es decir, es como una vista previa de lo que se observaría en el dispositivo en el que se implemente la aplicación. Esta ventana es de gran ayuda para corregir posibles errores de implementación, antes de que se lleve a cabo la implementación en el dispositivo final.
3. La ventana de jerarquía o **Hierarchy Window**, en la que encontramos una lista con todos los objetos de juego que se encuentran en la escena seleccionada de manera jerárquica y por orden de creación de estos. Es en esta parte de la interfaz donde aparece el término de jerarquía, es decir, un objeto puede ser “padre” de otro objeto “hijo”, el cual hereda todas las características de su progenitor, desde la rotación hasta el movimiento, por ejemplo. Además, se nos da la posibilidad de ocultar ciertos objetos de la escena sin necesidad de eliminarlos.
4. La ventana de proyecto o **Project Window**, en la que importamos, guardamos y editamos todos los recursos, conocidos como *Assets*, de los que consta la aplicación.
5. La ventana **Inspector**, que contiene toda la información acerca de un objeto de juego o recurso seleccionado, es decir, la configuración y las características que tiene asignado dicho objeto o recurso. Desde esta ventana se permite al usuario poder modificar las propiedades del elemento seleccionado.

Por lo tanto, esta herramienta es de gran utilidad en el presente trabajo para el desarrollo de la aplicación con la que se va a representar, mediante realidad aumentada, los datos recogidos por los sensores que conforman la red. El procedimiento de la representación de los datos recogidos será expuesto en el capítulo 5 del presente documento.

Una vez presentada la herramienta software con la que se modelará la aplicación móvil para representar los datos, es el momento de hablar sobre la otra herramienta esencial para que a dicha aplicación se le puede añadir la realidad aumentada.

Vuforia SDK es un complemento software de desarrollo creado por Qualcomm, el cual ofrece la posibilidad de crear aplicaciones de realidad aumentada utilizando herramientas como Unity3D. Además, posee uno de los mejores algoritmos de reconocimiento de imágenes que pueden encontrarse en la actualidad y es muy poco propenso a errores de detección y seguimiento de dichas imágenes, incluso en condiciones de baja luminosidad.



Figura 2.7. Logotipo de Vuforia SDK.

Este SDK ofrece la oportunidad de utilizar una componente conocida como ARCamera, la cual proporciona a Unity3D el poder usar la cámara de nuestro dispositivo para captar el entorno y representar la AR. Es por esto por lo que, Vuforia nos proporciona una experiencia completa de realidad aumentada, siendo una solución muy poderosa para el desarrollo de aplicaciones de este ámbito.

Por último, con esta herramienta se puede utilizar distintos métodos para el reconocimiento de las imágenes, entre los que están los marcadores de imágenes, los marcadores múltiples y botones virtuales.

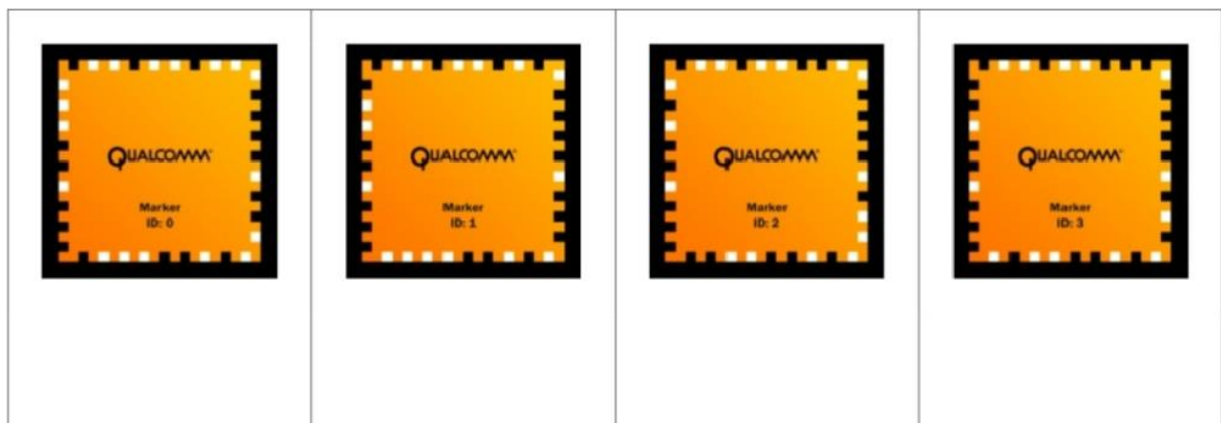


Figura 2.8. Ejemplo de múltiples marcadores de reconocimiento [3].

Antes de finalizar este apartado, aclarar que el método utilizado para llevar a cabo la implementación de la componente AR por parte de la aplicación móvil será el de reconocimiento de múltiples marcadores. Esto es porque la red de sensores que se pretende implementar está formada por más de un sensor, por lo que nos lleva a utilizar más de un marcador.

3 UN MUNDO CONECTADO: IoT

Internet facilita la información adecuada, en el momento adecuado, para el propósito adecuado.

Bill Gates.

El Internet de las Cosas, también conocido como Internet del Todo, es una nueva tecnología hecha para crear una red global donde las máquinas y dispositivos son capaces de interactuar entre sí. Este nuevo paradigma representa una de las evoluciones tecnológicas más importantes desde la creación de Internet como tal. Supone una serie de oportunidades para todo el mundo, proporcionando a las personas la capacidad de interactuar con sensores, actuadores, servicios o, en definitiva, objetos conectados a Internet.

3.1. Introducción al IoT

Actualmente, el concepto de Internet de las Cosas se refiere a una tecnología que permite a personas y cosas estar conectadas en cualquier momento, en cualquier lugar, con cualquier cosa y con cualquiera, idealmente usando cualquier ruta o red y cualquier servicio [13]. Sin embargo, desde que surgió esta nueva tecnología, muchos autores han definido al IoT de diversas formas, todas enfocadas en el mismo fin.

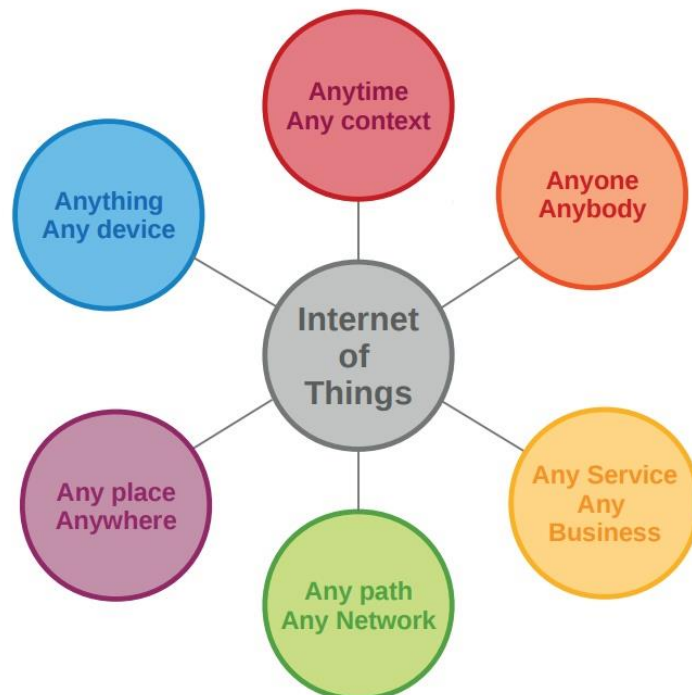


Figura 3.1. Internet de las Cosas [13].

Ejemplo de esto, como se discute en [10], los organismos de estandarización abordan esta nueva tecnología desde el punto de vista de “Orientada a Internet” o mediante la perspectiva de “Orientada a las Cosas”, dependiendo de los intereses específicos de cada uno de estos. Además, la propia definición de IoT implica un gran número de objetos, por lo que el direccionamiento único del objeto y la representación y el almacenamiento de la información se convierten en dos de los problemas que hacen que aparezca otra visión, “Orientada a la semántica”.

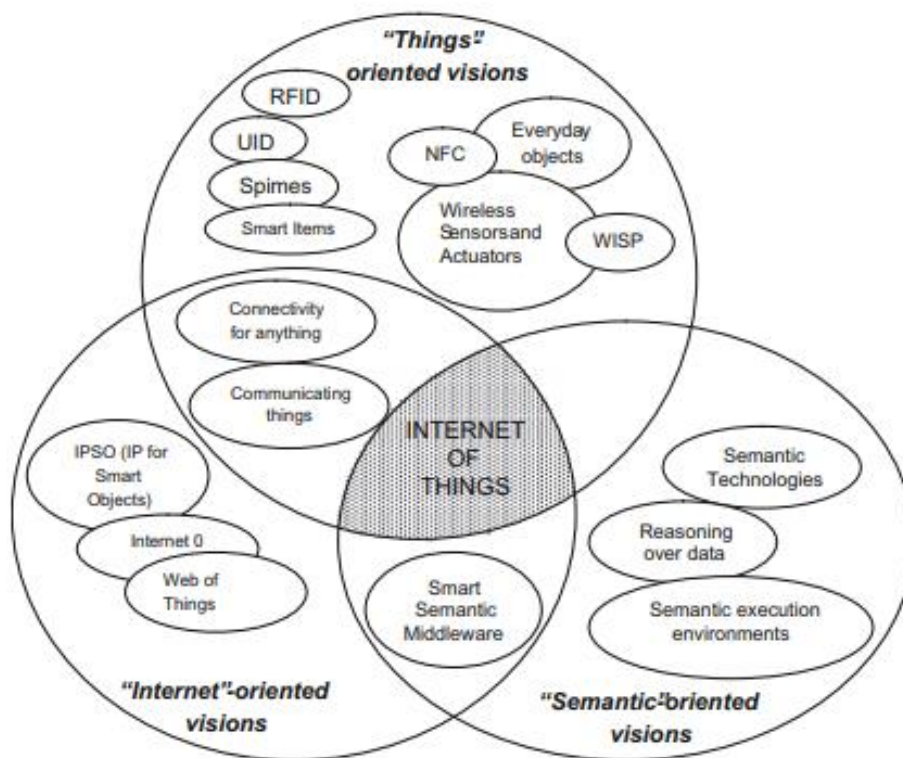


Figura 3.2. Perspectivas del IoT [10].

Como se puede observar en la Figura 3.2, el término de IoT supone la convergencia de todas las perspectivas existentes, cada una de ellas con las distintas tecnologías implicadas.

Por una parte, en la visión de “Orientada a Cosas”, los objetos son rastreados por sensores y tecnologías que utilizan Identificación por Radiofrecuencia (del inglés Radio Frequency Identification) o RFID [11]. La RFID es un tipo de tecnología que se emplea en muchos escenarios para la identificación automática o captura de datos, similar a tecnologías basadas en códigos de barra. El sistema RFID consiste en un lector que envía una señal a una etiqueta, la cual puede ser activa o pasiva, dando lugar a una señal modificada que retransmite datos de vuelta al lector. Los objetos utilizados son identificados de forma única mediante un Código Electrónico de Producto o EPC (Electronic Product Code del inglés) y la colección de datos se realiza a través de un sistema integrado por sensores. Por lo tanto, esta perspectiva del IoT se basa en el uso de redes de sensores que integren RFID o tecnologías basadas sobre RFID, como el NFC.

Por otra parte, la perspectiva “Orientada a Internet” nace de la necesidad de que los objetos o dispositivos físicos se encuentran de alguna manera conectados, esto es, sensores que pueden identificarse mediante una dirección IP e intercambian información entre ellos. Por esto, esta visión del IoT se centra en aumentar la inteligencia de los objetos que se encuentran conectados a Internet. Dichos objetos necesitan conocer, por lo tanto, los protocolos relacionados con Internet, como el direccionamiento IPv4 o el futuro IPv6, y la información enviada por estos debe unificarse mediante un formato único, para poder analizar parámetros y atributos y extraer la información útil.

Por último, en la visión “Orientada a la Semántica” se hace especial énfasis en la idea de que la cantidad de información intercambiada por los distintos objetos es demasiado grande. Por lo tanto, en este punto de vista se proponen métodos para procesar dichos datos de forma coherente, eliminando la parte de dicha información que resulte redundante y haciendo que, tras aplicar métodos de procesamiento de datos, la información que se almacene se convierta en significativa.

Con todo estas perspectivas de definición, se puede decir que el IoT se centra en la integración de objetos o dispositivos inteligentes en una misma arquitectura, donde las “cosas” y la perspectiva orientada a Internet dotan al IoT de un significado orientado a la red mediante el protocolo IP. Además, la inmensa cantidad de dispositivos conectados mediante esta tecnología contribuyen a diversos retos que se solventan utilizando tecnologías conocidas como semánticas, haciendo que el IoT sea una realidad.

3.2. Arquitectura básica del IoT

El modelo de arquitectura para esta nueva tecnología conocida como IoT ha dado lugar a diferentes interpretaciones del mismo, dando lugar, por lo tanto, a distintos modelos de la misma. Tal y como se discute en [12], se pueden llegar a proponer unos cuatro modelos (Figura 3.3) distintos de arquitectura para un sistema IoT: modelo de tres capas, modelo de capa intermedia o Middleware, modelo orientado a servicios o SOA y modelo de cinco capas.

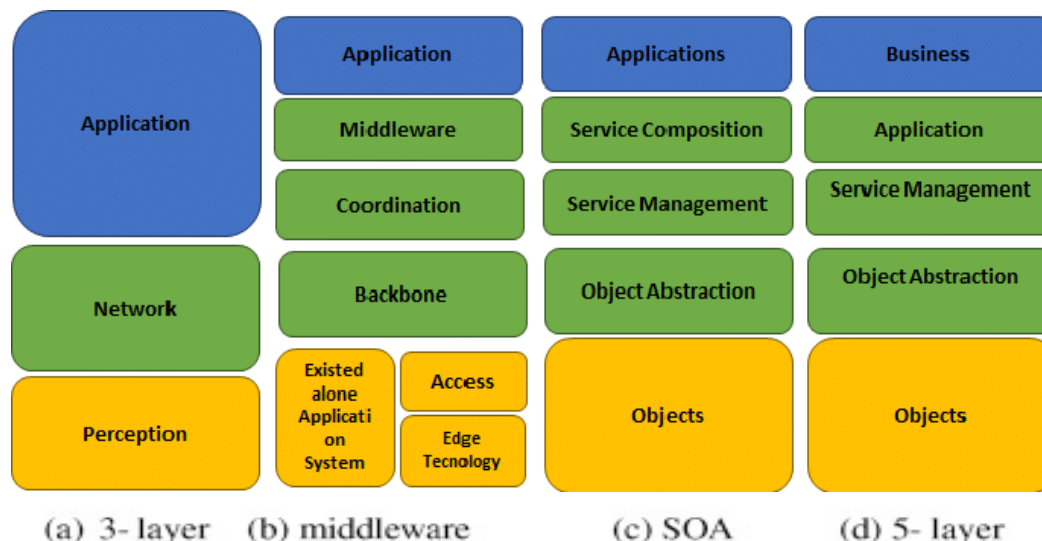


Figura 3.3. Arquitecturas de sistemas IoT propuestas [31].

Empezando por el más básico, el modelo de tres capas está compuesto por la capa de percepción, la capa de red y la capa de aplicación. En el resto de modelos, la capa intermedia experimenta un cambio, dado que pasa a una subdivisión de tres capas, aumentando así la abstracción de la arquitectura. Además, en el segundo modelo también se realiza una subdivisión en tres de la capa de percepción o capa de objetos.

Si bien no hay una arquitectura definida por completo para el IoT, todas estas perspectivas se enfrentan al reto de la abstracción de los objetos y las capacidades de comunicación existentes, proporcionando un conjunto de servicios comunes y la creación de dichos servicios.

Las arquitecturas más utilizadas son la arquitectura de tres capas y la de cinco capas, siendo la primera introducida en las primeras fases de investigación sobre el paradigma del IoT. Esta primera arquitectura consta de tres capas, que son [14]:

- La **capa de percepción**, que se trata de la capa física, es decir, de los sensores y actuadores que recogen la información del medio y se comunican con otros objetos del entorno.
- La **capa de red**, que es la responsable de la conexión con otros dispositivos inteligentes, conectados a la red o servidores. Además, también se encarga de transmitir y procesar los datos de los objetos.
- La **capa de aplicación**, la cual es la responsable de dotar de una serie de servicios específicos al usuario.

Sin embargo, esta arquitectura, aunque define la principal idea del IoT, no es suficiente para la investigación de este, porque no se centra en aspectos como la abstracción del objetos en sí. Es por esto por lo que han ido apareciendo otras arquitecturas, como la arquitectura de cinco capas, en la que, además de las ya mencionadas anteriormente, se añaden:

- La **capa de servicios**, que se encarga de almacenar, analizar y procesar una gran cantidad de información que recibe de la capa inferior. Además, puede administrar y proporcionar servicios a las capas inferiores.
- La **capa de negocio**, que se encarga de gestionar todo el sistema IoT, desde las aplicaciones hasta la privacidad de los usuarios que acceden a dicho sistema.

Por otra parte, la arquitectura orientada a servicios, documentada en [10] y mostrada en la Figura 3.4, está compuesta por 5 capas, donde la capa intermedia está compuesta a su vez por tres subcapas: la capa de abstracción del objeto, la capa de gestión de servicios y la capa de composición del servicio. Sin embargo, las capas de las otras arquitecturas realizan unas funcionalidades muy parecidas a la de esta, exceptuando la de la arquitectura de tres capas como ya se ha comentado.

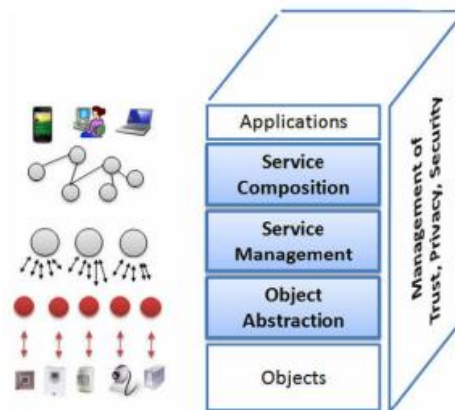


Figura 3.4. Arquitectura SOA [10].

Capa de aplicación

Esta capa se encuentra en lo más alto de la arquitectura y es la que se encarga de proporcionar al usuario todas las funcionalidades del sistema. Además, no es considerada como parte de la capa intermedia o Middleware, pero hace uso de los servicios que le proporciona esta última. Su función es proporcionar al usuario, mediante el uso de protocolos de servicios de Internet y de las tecnologías de composición de servicios, una perfecta integración con el sistema distribuido.

Capa de composición de servicios

Se trata de una de las capas más comunes en una arquitectura SOA, la cual proporciona las funcionalidades para componer servicios ofrecidos por los objetos de la red y construir así aplicaciones específicas. Esta capa es una capa intermedia en la que no hay noción de la existencia de los objetos del sistema, si no que lo único que se conoce son los servicios que se ofrecen a la capa superior.

Una visión importante sobre la creación de los servicios es conocer todas las instancias de los servicios conectados en un momento concreto. La lógica de la creación y la gestión compleja de los mismos se puede expresar en forma de flujos de trabajo, los cuales definen los procesos que interactúan con entidades externas a través de operaciones de servicio web, es decir, la interacción entre la capa de composición de servicios y la capa de aplicación para dar soporte a esta última.

Capa de gestión de servicios

Con esta capa se proporcionan las funciones principales que estarán disponibles para cada objeto y que permitirá la gestión de estos en el escenario IoT.

La gestión de los servicios por parte de esta capa abarca tres partes: descubrimiento de los objetos de forma dinámica, monitorización del estado de dichos objetos y configuración del servicio para cada objeto. Con esta capa se permite, por tanto, la implementación remota de nuevos servicios en tiempo real, satisfaciendo las distintas necesidades de la capa de aplicación. Para ello, se crea una serie de servicios y estos son asociados a cada nodo de la red según las necesidades específicas de los mismos, pudiendo la capa superior componer servicios complejos fusionando servicios de esta capa.

Capa de abstracción del objeto

Una capa de abstracción es una forma de ocultar los detalles de trabajo de un subsistema, permitiendo la interoperabilidad y la independencia del sistema [12]. Esta capa nace de la necesidad de armonizar todo los objetos que se conectan a la red, dado que cada uno proporciona funciones específicas a través de su propio lenguaje. Por tanto, esta capa se encarga de crear un formato común para el envío de datos por parte de los objetos.

Además, existe la necesidad de introducir una capa adicional, que envuelve a esta capa de abstracción, en el caso de que uno de los dispositivos del sistema se encuentre conectado a una red IP. Esta nueva capa consta a su vez de otras dos: capa de interfaz y capa de comunicación. La primera proporciona una interfaz web que recoge todos los métodos disponibles y es responsable de la gestión de todas las operaciones implicadas en la comunicación con el mundo real. Por otra parte, la segunda capa implementa la lógica que se encuentra detrás de los métodos utilizados en el servicio web, traduciendo dichos métodos a un conjunto de comandos que conforman el lenguaje con el que el objeto se comunicará con el mundo real.

Esta capa de abstracción se suele proporcionar a los objetos mediante el uso de un proxy, el cual es el responsable de realizar, con el uso de un socket, la comunicación con el dispositivo y enviar todos los comandos usando los diferentes lenguajes de comunicación. Por lo tanto, es el encargado de realizar la conversión a un lenguaje de servicio web y elaborar la solicitud de comunicación para reducir la complejidad de las operaciones requeridas por el dispositivo final.

Capa de objetos

La capa de objetos, también conocida como capa de percepción, consta de los dispositivos de detección y actuación o sensores, los cuales miden o detectan fenómenos en el entorno que les rodea y se encuentran conectados entre sí, formando así el sistema IoT.

Esta capa, semejante a la capa física del modelo TCP/IP, es la responsable de la generación de datos que, tras ser enviados a través de la red, se almacenarán en sistemas de recopilación de datos, para que un usuario final pueda acceder a estos. También proporciona servicios que hacen que se produzca una diferencia entre el modelo TCP/IP y el modelo de IoT, como son los servicios de identificación de los objetos y los de recopilación de información específica de cada objeto con fines de gestión de los mismos. Además, hace falta, dada la heterogeneidad de los distintos objetos conectados al sistema, una configuración de estos o un mantenimiento continuado para que realicen las funciones que se les pida.

3.3. Protocolos de comunicación del IoT

La estandarización para el desarrollo de protocolos para el IoT ha sido llevada a cabo por organismos como IEEE, ZigBee Alliance, Bluetooth SIG, IETF, ETSI o ITU. Además, muchos de estos organismos se unieron en una plataforma conjunta conocida como oneM2M para el desarrollo de un estándar que se encargue de abordar las necesidades de las aplicaciones y los servicios del IoT.

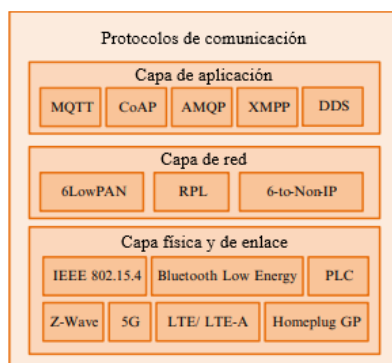


Figura 3.5. Protocolos del IoT propuestos por los distintos organismos [15].

3.3.1 Capa física y de enlace

En estas capas, los protocolos disponibles para las aplicaciones de IoT y M2M son muy variados, diferenciándose los protocolos para comunicaciones inalámbricas, como el estándar IEEE 802.15.1, conocido comúnmente como Bluetooth, o el estándar IEEE 802.15.4, conocido como ZigBee, y los protocolos para comunicaciones móviles, como 5G o LTE/LTE-A [15].

Tabla 3.1. Comparativa de algunos protocolos de capa física [12].

Protocolos de capa física	Técnica de difusión	Banda de Radiofrecuencia (MHz)	Técnica de acceso al medio	Tasa de transmisión (bps)	Máximo n° de nodos
IEEE 802.15.4	DSSS	868/915/2450	TDMA CSMA/CA	20/40/250 k	≈ 65.000
IEEE 802.15.1	FHSS	2400	TDMA	1024 k	5917
LTE/LTE-A	Multiportadora	800/1800/2600	OFDMA	DL 3 G, UL 1.5 G	-
5G	MIMO	3.4 - 3.8	-	+ 500 M	Gran escala

3.3.1.1 IEEE 802.15.4, ZigBee

El estándar IEEE 802.15.4 es una tecnología de red inalámbrica, dirigida a aplicaciones de monitorización remota y de control, desarrollada por un consorcio de empresas llamado ZigBee Alliance. Permite la transmisión de tramas MAC a través del canal físico mediante redes inalámbricas de área personal de baja velocidad o LR-WPAN. Además, proporciona direccionamiento, gestión de los datos y el control del acceso al canal físico para permitir que varios dispositivos utilizados en el sistema IoT (hasta 255 activos) compartan un solo medio.

En cuanto a capa física se refiere, existen dos capas físicas en las que opera este estándar: ZigBee PHY 868/915 MHz y ZigBee PHY 2.450 MHz. La versión que está especialmente pensada para utilizarla en el IoT es el ZigBee 3.0, que opera en la banda de 2.4 GHz y que llega a una tasa de hasta 250 kbps, empleándose una modulación OQPSK con conformado de pulso, lo cual hace que se ocupe muy poco ancho de banda.

El método de acceso al medio seleccionado para este protocolo de capa física es el TDMA CSMA/CA, en el que existe la probabilidad de aumento de colisión de tramas en cuanto se aumenta el número de dispositivos IoT que utilizan el medio. Es por esto último por lo que se lleva a cabo una rutina anticollisiones parecida a la de WiFi.

3.3.1.2 IEEE 802.15.1, Bluetooth

El estándar IEEE 802.15.1 o Bluetooth es, al igual que ZigBee, una tecnología de comunicación inalámbrica de corto alcance. Existen distintas versiones de este estándar, desarrolladas a través de los años desde que apareció la primera versión o Bluetooth 1.0, siendo la utilizada en el IoT la versión 5.0, la cual ofrece una comunicación de bajo consumo.

Este estándar opera en capa física en la banda ISM de 2.4 GHz, la cual se divide en 79 canales de 1MHz, con pequeñas bandas destinadas a intervalos de guarda. Además, se emplea saltos en frecuencia o FHSS y una modulación GMSK.

Por último, el acceso al medio se realiza mediante el método TDMA, pudiendo un canal ser ocupado por distintas fuentes de información. La estructura de la trama consta de intervalos de 625 μ s y las ráfagas de los dispositivos que accedan al medio pueden ser de hasta 5 intervalos consecutivos. Con este protocolo se puede alcanzar una tasa agregada de hasta 1 Mbps.

3.3.1.3 LTE/LTE-A y 5G

Por su parte, LTE y LTE-Advance proporcionan un mayor ancho de banda de comunicación y mayor soporte de movilidad al tratarse de un tipo de comunicación móvil. Además, al ser un estándar enormemente extendido por todo el mundo, se convierte en uno de los candidatos más adecuados para la comunicación M2M en el IoT. Sin embargo, es un tipo de comunicación pensada para transmisión de persona a persona o H2H, lo que hace que implementar un sistema IoT utilizando el LTE o LTE-A sea costoso en cuanto a manejo de datos y a energía consumida.

En cuanto a canales físicos, el más utilizado es el canal RACH o canal común de acceso aleatorio, el cual establece una conexión radio antes de transmitir o recibir los datos del IoT. Sin embargo, como en un sistema IoT hay gran número de dispositivos conectados, se puede producir una sobrecarga de señalización, pudiendo llegar a congestionar el canal mediante el uso de un canal RACH. Por esto, los organismos de estandarización han tenido que trabajar para crear lo que se conoce como LTE Cat0, que hace que el dispositivo que utilice LTE consuma menos, pero sigue habiendo desafíos que abordar para que el LTE se utilice en el IoT.

Por otra parte, las redes 5G se encargan de solventar de alguna forma los problemas de las redes LTE y LTE-A, como son el soporte de una gran cantidad de dispositivos M2M, una velocidad de obtención de datos casi instantánea y una baja latencia.

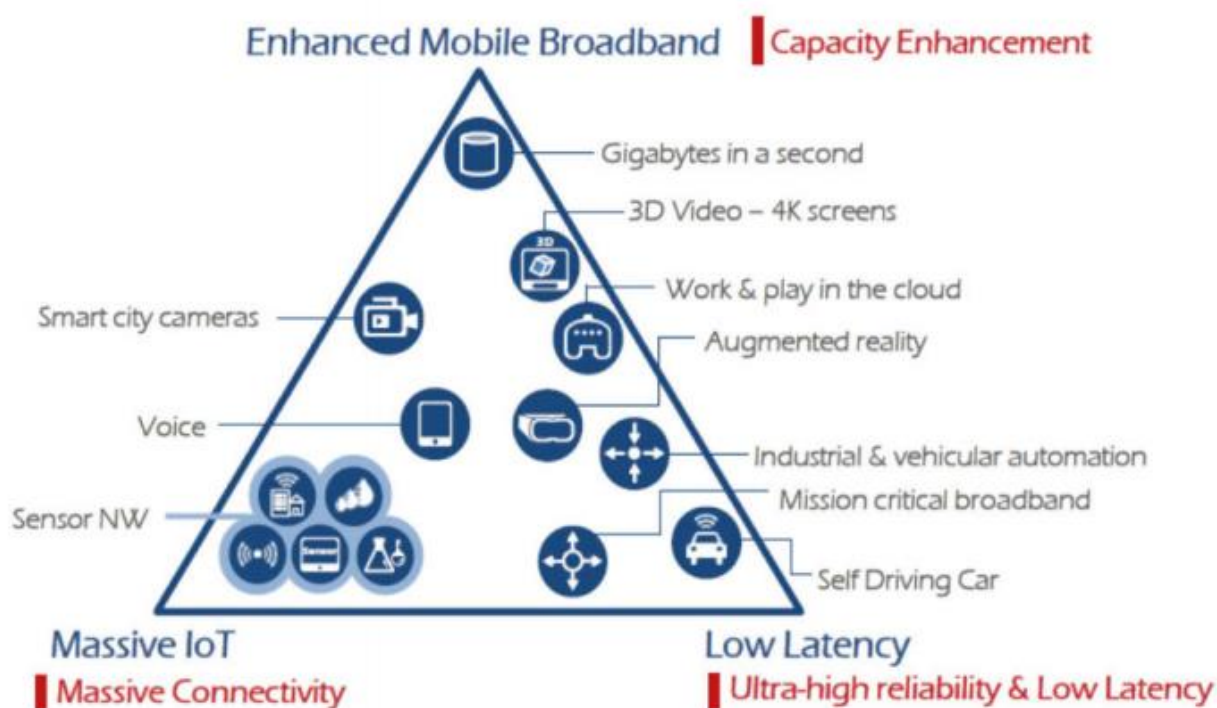


Figura 3.6. Quinta Generación de comunicaciones móviles [16].

3.3.2 Capa de red

3.3.2.1 RPL

En el IoT, el enrutamiento es un problema importante, ya que los dispositivos poseen poca energía para mantenerse activos mediante el uso de baterías. Por esto, el organismo IETF desarrolló el protocolo de enrutamiento RPL, diseñado principalmente para redes inalámbricas, en las que los dispositivos que la forman poseen poca batería, ofreciendo una solución de enrutamiento versátil para proporcionar buen rendimiento en entornos LLN y que se adapta a una amplia cantidad de protocolos de enlace [17].

Este protocolo es un protocolo de enrutamiento por vector de distancia que utiliza IPv6 y que proporciona un mecanismo para admitir tráfico multipunto a punto, punto a multipunto y punto a punto.

3.3.2.2 6LoWPAN

Se trata de un protocolo definido por el IETF y que actúa en la capa de adaptación entre la capa de objetos y la capa de red. Su función es facilitar la transmisión de paquetes IPv6 en pequeñas tramas de capa de enlace de los protocolos de capa física y de enlace.

La razón por la que es un protocolo adecuado para el IoT es que es capaz de conectar una gran cantidad de dispositivos y controlar a estos mediante la red IP. Además, brinda a los objetos de la función que les posibilita el descubrimiento de otros objetos en redes ad-hoc y en redes subenrutadas que utilicen direcciones de capa de enlace en vez de utilizar direcciones IP.

Por lo tanto, es un protocolo que facilita la interoperabilidad entre los objetos de un sistema IoT, proporcionando conectividad E2E a través de Internet. Esto es posible gracias a que se conecta una red que utiliza este protocolo con una red que utilice IPv6 haciendo uso de enrutadores y de Internet. Actualmente, es el protocolo clave para las redes LLN, en las cuales la comunicación se ve restringida por las pérdidas existentes y la baja energía de los dispositivos.

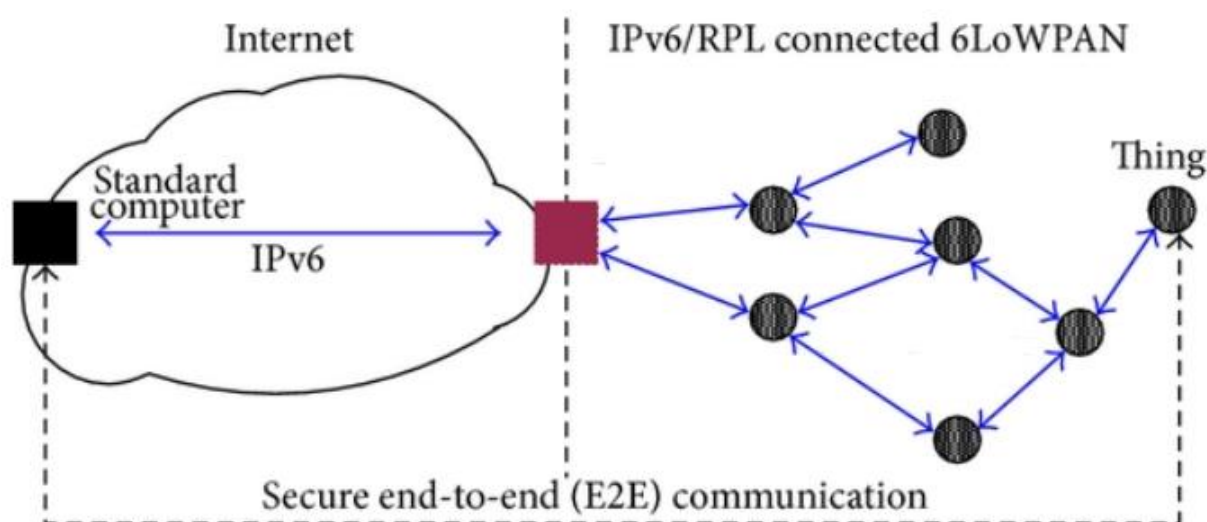


Figura 3.7. Escenario con protocolos de la capa de red.

3.3.3 Capa de aplicación

Tabla 3.2. Comparativa de algunos protocolos de capa de aplicación [19].

Protocolos de capa de aplicación	Protocolo de transporte	QoS	Modelo de la arquitectura	Protocolo de seguridad
MQTT	TCP	SÍ	Publicación/Suscripción	TLS/SSL
CoAP	UDP	SÍ	Petición/Respuesta	DTLS
AMQP	TCP	SÍ	Publicación/Suscripción	TLS/SSL
XMPP	TCP	NO	Publicación/Suscripción Petición/Respuesta	TLS/SSL
DSS	TCP/UDP	SÍ	Publicación/Suscripción	SSL/DTLS

3.3.3.1 MQTT

MQTT, o *Message Queue Telemetry Transport*, es un protocolo de la capa de aplicación que se basa en publicación/suscripción, similar al protocolo de cliente/servidor. Sin embargo, al ser de código abierto y la simplicidad que presenta hace que sólo sea adecuado para entornos restringidos, es decir, entornos de bajo consumo, computación limitada o poco ancho de banda [17].

Es un protocolo desarrollado por IBM y estandarizado por OASIS, el cual es adecuado para las aplicaciones de sistemas IoT y comunicaciones M2M. Además, utiliza TCP como protocolo de capa de transporte, capa que le proporciona seguridad mediante SSL.

Su objetivo es reducir los requisitos de ancho de banda y garantizar la entrega de paquetes, proporcionando para ello una serie de características, como la difusión de mensajes punto a multipunto y la capacidad de establecer comunicaciones entre dispositivos remotos. Pese a estas características, aún hay una que es más importante como es la minimización del tráfico de la red a costa de reducir la sobrecarga de transporte y los cambios de protocolo. Además, es el protocolo que ofrece la mayor QoS, comparado con el resto.

3.3.3.2 CoAP

CoAP, del inglés *Constrained Application Protocol*, es un protocolo utilizado para dispositivos integrados de baja memoria y baja potencia donde utilizar HTTP sería imposible, debido a que HTTP utiliza a TCP como protocolo de transporte, haciendo que se necesiten más recursos y mecanismos más complejos [19].

Este protocolo es del tipo petición/respuesta, similar al de tipo cliente/servidor. Las características más importantes de CoAP son la simplicidad y la admisión de conexiones punto a punto y punto a multipunto por aprovechar el protocolo UDP y tener la capacidad de intercambio de mensajes de forma asíncrona. Sin embargo, presenta una mala latencia, mala entrega de paquetes y es inutilizable en entornos donde los datos usados sean complejos.

Es un protocolo pensado para, al igual que MQTT, entornos restringidos por capacidad o potencia, como redes WPAN. CoAP fue diseñado por el IETF y está pensado principalmente para aplicaciones M2M y para sistemas de automatización para reducir la sobrecarga, mejorar la entrega de paquetes y aumentar la simplicidad, utilizando una interfaz básica con HTTP. A diferencia de MQTT, este último tiene un mejor comportamiento en el caso de redes con alto tráfico, proporcionando mayor rendimiento y menos latencia que CoAP.

3.3.3.3 AMQP

AMQP, siglas de *Advanced Message Queuing Protocol*, es un protocolo que, al igual que MQTT, utiliza un modelo de publicación/suscripción, el cual se utiliza principalmente en plataformas comerciales y de negocios. Utiliza TCP como capa de transporte y el protocolo SSL como parte de seguridad y fue estandarizado por OASIS.

Este protocolo es utilizado en entornos de intercambio de mensajes entre distintos objetos y su característica más importante es la heterogeneidad y la interoperabilidad que ofrece para distintos dispositivos. Además, es un protocolo muy parecido a MQTT, con la diferencia de que AMQP tiene la ventaja de que almacena los datos y los reenvía, asegurando la entrega correcta de la información cuando se producen interrupciones en la red y aumentando la fiabilidad de esta.

3.3.3.4 XMPP

XMPP, de *Extensible Messaging and Presence Protocol*, es un protocolo de mensajes diseñado para intercambios de mensajes en aplicaciones de mensajería, estandarizado por el IETF. Este protocolo se utiliza principalmente en comunicaciones dentro de sistemas IoT para el intercambio de mensajes cortos, dada la baja latencia que proporciona.

Además, es el único protocolo de capa de aplicación que soporta los dos modelos existentes, publicación/suscripción y petición/respuesta, siendo el desarrollador el que elija el tipo de modelo, dependiendo de la aplicación que se quiera llevar a cabo. Sin embargo, no proporciona calidad de servicio y no es práctica para comunicaciones M2M.

3.3.3.5 DDS

DDS, del inglés *Data Distribution Service*, es un protocolo muy importante para entornos IoT en la actualidad, el cual fue diseñado y desarrollado por OMG. Se trata de un protocolo que utiliza el modelo publicación/suscripción y soporta tanto aplicaciones IoT como comunicaciones M2M.

Es un protocolo que soporta muchos criterios de QoS y este se escoge dependiendo del modelo de comunicación escogido para la publicación/suscripción. Además, define dos subcapas: la capa de suscripción y la capa de reconstrucción local de datos. La primera de ella se encarga de entregar al suscriptor el mensaje de información sobre el tema al que se han suscrito y la segunda capa, la cual es opcional, permite la integración del protocolo en la capa de aplicación.

3.4. Tecnologías de conectividad en el IoT

Dado el alto número de dispositivos que se encuentran conectados hoy día, y ya que serán más en el futuro, la necesidad del desarrollo de tecnologías que soporten dicha conectividad es una de las cosas en las que se centran los principales organismos mundiales. Como se discute en [20], las diferentes tecnologías de conectividad que existen en la actualidad se pueden clasificar en términos de alcance como tecnologías de corto alcance o de largo alcance. En el primer grupo encontramos a Bluetooth, WiFi, ZigBee o tecnologías OWC, mientras que en el segundo grupo están el LTE, el 5G o las tecnologías LPWAN.

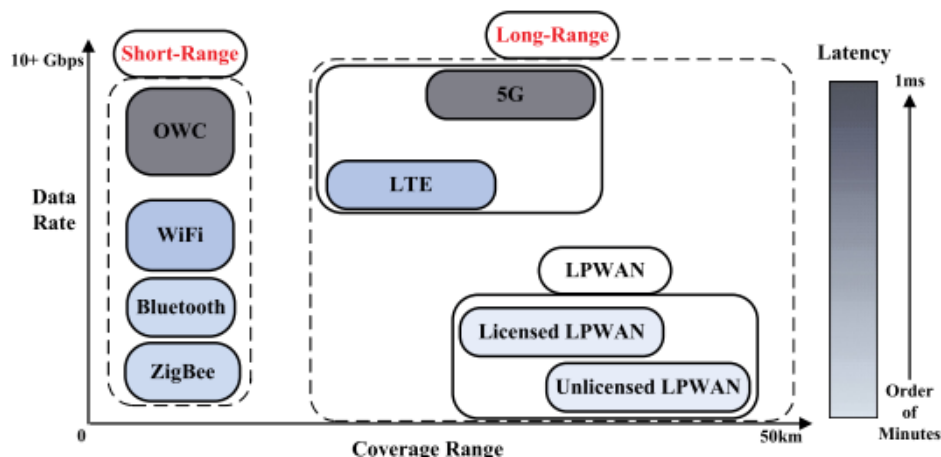


Figura 3.8. Tecnologías de conectividad según el alcance [20].

Sin embargo, en [15], se puede hacer otra clasificación de las tecnologías de conectividad atendiendo a si las comunicaciones son celulares o no. En este caso, las tecnologías celulares son las tecnologías de comunicaciones móviles (3G/4G/5G), LPWAN o LTE-M entre otras, coincidiendo con las tecnologías de largo alcance. Por otra parte, como era de esperar, las no celulares se corresponden con las de corto alcance.

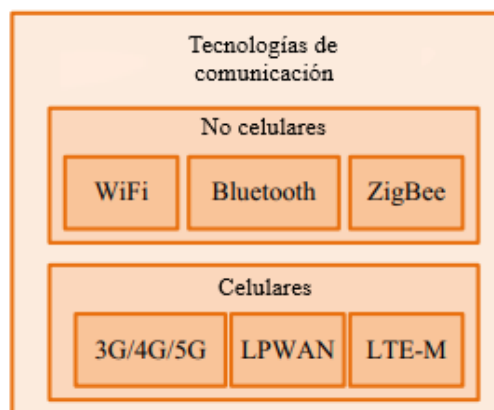


Figura 3.9. Tecnologías de conectividad según si la tecnología es o no celular [15].

3.4.1 Tecnologías de largo alcance o celulares

Como ya se ha comentado, entre estas tecnologías existen las típicas tecnologías enfocadas a las comunicaciones móviles, las enfocadas a resolver los problemas de entornos LPWAN y el LTE-M, que se trata de una versión más básica del LTE. En la presente memoria se va a desarrollar las principales características de las tecnologías LPWAN y las del LTE-M, puesto que hoy día son las más demandadas en el desarrollo de aplicaciones IoT.



Figura 3.10. Aplicaciones de las redes LPWAN [22].

Dentro de las tecnologías LPWAN existen dos subgrupos: las LPWAN con licencia como LTE-M y NB-IoT, las cuales operan en bandas ISM que necesitan licencia para acceder a dichos rangos de frecuencia, y las LPWAN sin licencia como LoRa y SigFox, que operan en bandas de frecuencia libre.

Tabla 3.3. Comparativa de las tecnologías LPWAN [20].

	LoRa	SigFox	LTE-M	NB-IoT
Protocolo RA	ALOHA / ALOHA ranurado	ALOHA	ALOHA ranurado	ALOHA ranurado
Modulación	CSS	GFSK/DBPSK	QPSK/QAM	BPSK/QPSK
Frecuencia	Bandas ISM sin licencia	Bandas ISM sin licencia	Bandas LTE con licencia	Bandas LTE con licencia
Ancho de banda	125/250 kHz	100 Hz	1.4 MHz	200 kHz
Direccionamiento	Semidúplex	Semidúplex	Dúplex/Semi	Semidúplex
Tasa de datos máxima	50 kbps	100 bps	1 Mbps	250 kbps
Tamaño máximo de carga	243 bytes	12 bytes	1000 bits	1000 bits
Cobertura (urbano-rural)	5-20 km	10-50 km	Pocos kms	1-10 km
Inmunidad a interferencia	Alta	Muy Alta	Baja	Baja
Vida de baterías	10 años	10 años	10 años	10 años
Movilidad	Sí	No	Sí	Sí

3.4.1.3 LTE-M

LTE-M es una de las tecnologías que hace uso de una banda especial dentro del espectro radioeléctrico, es decir, hace uso de bandas con licencia. Es una tecnología compatible con las redes celulares actuales en la que los dispositivos se conectan directamente a la red 4G sin necesidad de pasarelas. Además, es una versión simplificada del LTE, orientada a dispositivos de bajo coste y aplicaciones de IoT con bajo consumo de energía. Su área de cobertura no es muy extensa, alcanzando unos cuantos kilómetros.

Utiliza OFDMA en el DL y SC-FDMA multitonos en el UL, con una separación entre portadoras de 15 kHz. La modulación usada es la 16-QAM con una canalización de 1.4 MHz, consiguiendo unas tasas de hasta 1 Mbps, utilizando los dispositivos a una potencia de 20-23 dBm.

3.4.1.4 NB-IoT

NB-IoT es otra de las tecnologías que operan en bandas con licencia, siendo casi similar al LTE-M con la diferencia que este se encuentra ya implementado en el sistema LTE existente desde el Release 13. Esta tecnología utiliza una banda estrecha de 200 kHz, reduciendo así la complejidad del sistema, el coste de los dispositivos, haciendo que se prolongue más la duración de su batería, y proporcionando mayor penetración a la comunicación. Además, las tasas de datos de esta tecnología son de 158.5 kbps en el UL y de 106 kbps en el DL y, en comparación con el LTE-M, esta tiene mayor rango de cobertura.

En cuanto a modulaciones, utiliza OFDMA en el DL, con un espaciado de 15 kHz entre portadoras, y SC-FDMA en el UL, con espaciados de 15 kHz y 3.75 kHz. Por último, en cuanto a los rangos de cobertura, se establece un rango de entorno a 1 km en entornos urbanos y de 10 km en zonas rurales.



Figura 3.12. LTE para aplicaciones IoT en redes LPWAN [16].

3.4.2 Tecnologías de corto alcance o no celulares

Tabla 3.4. Comparativa de las tecnologías de corto alcance [20].

	WiFi	ZigBee	Bluetooth
Protocolos de acceso al medio	CSMA/CA	CSMA/CA	TDMA
Tipo de modulación	BPSK/QPSK/QAM	BPSK/OQPSK	GFSK/DQPSK/DPSK
Máxima tasa de datos	7 Gbps	250 kbps	Hasta 3 Mbps
Cobertura	100-1000* m	100 m	100-240* m
Número de dispositivos	255	255	Más 1000 en redes malladas

* Según la versión seleccionada.

Las tecnologías de corto alcance se utilizan en aplicaciones IoT para mantener conectada un área de no mucho rango. Las principales tecnologías utilizadas para ello son WiFi, ZigBee y Bluetooth. Estos tipo de tecnologías no son capaces de abordar todos los requerimientos que necesitan las aplicaciones IoT en cuanto al rango de la red, capacidad y la eficiencia energética. Sin embargo, pueden permitir las comunicaciones M2M y se pueden desplegar en todo los entornos.

3.4.2.1 WiFi

WiFi es una de las tecnologías más utilizadas en todo el mundo, estandarizada como IEEE 802.11 y utilizada en redes WLAN, implementada en entornos domésticos y de trabajo.

Una de sus características más importantes es que ofrece una gran cobertura y mayores tasas que el resto de las tecnologías de corto alcance. Además, es la única de las tres tecnologías que se exponen que permite la conexión entre sus dispositivos. Esta comunicación entre nodos se realiza a tasas superiores y tienen mucha menor latencia, aunque a todo esto le contrarresta el elevado consumo de potencia.

Existen distintas versiones de WiFi, siendo el estándar IEEE 802.11ac el que mejora el rendimiento, la velocidad y la interferencia, introduciendo MIMO para ello. Sin embargo, aunque las versiones más recientes han servido para poder extender la distancia y disminuir el retardo de la comunicación, el número de nodos que se pueden implementar es reducido.

Por último, el estándar que se lanzó para dar soporte a aplicaciones IoT es el 802.11ah, que reduce el consumo de potencia y da un mayor rango. Opera en subbandas de 1 GHz sin licencia y los anchos de banda utilizados son de 1 o 2 MHz, siendo permitidas bandas de hasta 16 MHz en algunos países. Este último tiene como objetivo dar cobertura de hasta 1 km a bastantes dispositivos con tasas de hasta 300 Mbps [20].

3.4.2.2 ZigBee

ZigBee es una tecnología creada para conectar dispositivos en una red WPAN y fue estandarizado en IEEE 802.15.4 como protocolo de capa física y capa de enlace. Opera en las bandas sin licencia de 868 MHz, 915 MHz y 2.4 GHz, dividiendo dichas bandas en 16 canales de 2 MHz de ancho de banda cada uno. Además, se utiliza modulación DSSS, se puede llegar a conseguir hasta 250 kbps de tasa binaria y se usa CSMA/CA como método de acceso al medio.

868 MHz	COVERAGE	DATA RATE	CHANNELS
	Europe	20 Kbps	1
915 MHz	COVERAGE	DATA RATE	CHANNELS
	Americas	40 Kbps	10
2.4 GHz ISM	COVERAGE	DATA RATE	CHANNELS
	Worldwide	250 Kbps	16

Figura 3.13. Bandas de operación de ZigBee [23].

En cuanto a las limitaciones que conlleva el uso de esta tecnología se encuentran las bajas tasas binarias que es capaz de conseguir (20-250 kbps), el rango de operación no es muy amplio y la capacidad es reducida. En contra, es una tecnología que requiere de baja potencia.

Es usada en redes WSN para aplicaciones de automatización de espacios comerciales y domésticos, monitorización de plantas industriales, de salud, etc. Estas redes de sensores pueden tener tres topologías distintas, todas ellas con un máximo de 255 dispositivos conectados, como son la topología en estrella, la topología mallada y la topología en árbol. Además, los dispositivos comentados pueden ser de tres tipos:

- **Coordinadores**, los cuales son los responsables del control y establecimiento de conexiones en la red. Además, son los responsables de elegir los parámetros de configuración de la red y almacenan información de la red como las claves de seguridad.
- **Encaminadores**, que son los que realizan la funcionalidad de encaminamiento de datos y que actúan como conectores entre distintos nodos de la red, además de poder comunicarse con otros encaminadores.
- **Dispositivos finales**, que sólo tienen la funcionalidad de comunicarse con el nodo padre, ya sea el coordinador o un enrutador, y no pueden enviar mensajes de datos a otros dispositivos.

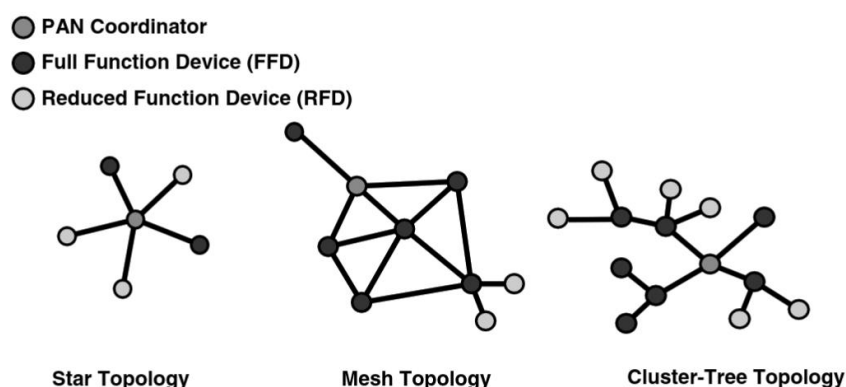


Figura 3.14. Topologías disponibles en ZigBee [23].

En definitiva, ZigBee es una buena alternativa para aquellas aplicaciones IoT de corto alcance que no supongan un gran número de dispositivos y por la baja latencia entre dispositivos que presenta, pero las limitaciones comentadas anteriormente hacen que otras alternativas, como Bluetooth, sean mejores elecciones.

3.4.2.3 Bluetooth

Esta última tecnología es una especificación gestionada por Bluetooth SIG, estandarizada en el IEEE 802.15.1, estándar en el que se definen las características de radiofrecuencia para la capa física y la capa de enlace. Es utilizada en comunicaciones inalámbricas de corto alcance, operando en bandas ISM desde 2.402 a 2.48 GHz, consiguiendo unas tasas de 1 a 3 Mbps.

A lo largo de los años, esta tecnología se ha ido adaptando a las necesidades de los demandantes, desde ofrecer mayor tasas binarias a proporcionar un rango de distancia mayor. Es por esto por lo que, a día de hoy, se han desarrollado una totalidad de 6 versiones distintas de Bluetooth, siendo la última, Bluetooth 5.0, la que está pensada originalmente para aplicaciones IoT, esto es, aplicaciones de continua transmisión de datos de forma segura, con alto grado de fiabilidad y de muy baja potencia. Además, esta última versión proporciona un alcance de 100 metros como máximo y este estándar dota al BLE o Bluetooth 4.0 del doble de tasa.

Tabla 3.5. Comparativa de los estándares de Bluetooth.

Estándar	Incorporaciones
Bluetooth v1.0	Primera versión con problemas de interoperabilidad
Bluetooth v1.1	Se resuelven errores previos y se añaden canales no encriptados y de RSSI
Bluetooth v1.2	Adición FHSS adaptativo y conexiones eSCO, se consiguen tasas de hasta 721 kbps y mejora en la calidad de voz
Bluetooth v2.0 + EDR	Introduce EDR para mejorar tasas hasta los 2.1 Mbps
Bluetooth v2.1 + EDR	Mejora del mecanismo de asociación y de la seguridad
Bluetooth v3.0 + HS	Incluye HS para alcanzar tasas de 24 Mbps de forma teórica y un mecanismo de control de potencia
Bluetooth v4.0	Introduce el término de BLE, proporcionando bajo consumo de energía
Bluetooth v4.1y v4.2	Pequeñas actualizaciones de la versión anterior
Bluetooth v5.0	Enfocada al IoT, dotando del doble de velocidad a la v4.0
Bluetooth v5.1	Mejora la precisión de la localización de los dispositivos

En esta tecnología, la arquitectura de red en la que se encuentran conectados los dispositivos que la forman se conoce como *piconet*. En ella, un dispositivo de todos los que la forman es denominado el maestro, siendo los restantes dispositivos esclavos. Por lo tanto, estamos ante una arquitectura del tipo maestro-esclavo, en la que el maestro fija los saltos de frecuencia y los esclavos tienen que sincronizarse. Además, el esclavo de una *piconet* puede ser maestro en otra, aunque un dispositivo solo puede ser maestro en una de ellas. Es por esto por lo que pueden existir conexiones de *piconets*, las cuales se conocen como *scatternets*.

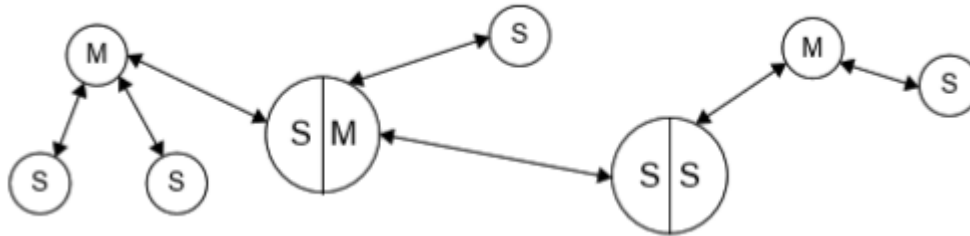


Figura 3.15. Ejemplo de “Scatternet” [24].

Por otra parte, sólo puede haber 7 dispositivos esclavos activos simultáneamente, existiendo, tal y como se recoge en [24], distintos estados por los que pueden pasar los distintos dispositivos:

- Modo **activo**, en el que se puede transmitir en un instante de tiempo prefijado por el dispositivo maestro.
- Modo **sniff**, que se trata de un modo de ahorro de energía en el que, de vez en cuando, escuchan el canal por si hubiera paquetes pendientes de recepción.
- Modo **hold**, en el que los dispositivos (hasta 255) no participan de manera activa durante un tiempo en la *piconet* y después se vuelven a incorporar, requiriéndose un registro previo.
- Modo **parked**, en la que los esclavos mantienen la sincronización con el dispositivo maestro pero no participan de manera activa hasta nuevo aviso.

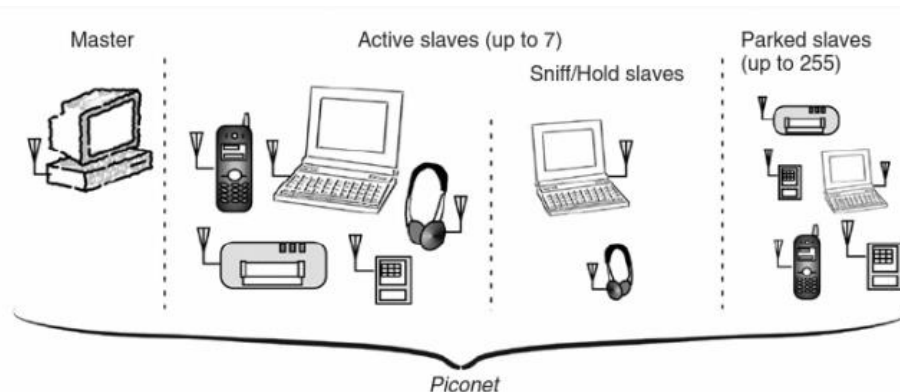


Figura 3.16. Modos de los dispositivos dentro de la piconet [16].

Sin embargo, el principal inconveniente de esta tecnología es el número de dispositivos que pueden conectarse en una red. Por esto, para hacer posible el uso del BLE dentro de aplicaciones IoT de larga escala, se pasó a la implementación de redes malladas, o *mesh networks*, cuya topología utiliza el principio de enrutamiento de inundación, abandonando el concepto de *piconet*, para reenviar los datos de un dispositivo a otro, pudiendo conectar un máximo de 32767 dispositivos.

Con esto, el Bluetooth se ha convertido en la tecnología de corto alcance más utilizada para aplicaciones IoT, siendo usada, por ejemplo, en control y monitorización de edificios inteligentes o en la automatización de servicios domésticos [23].

4 ARQUITECTURA ARIoT

La tecnología es sólo una herramienta. La gente usa las herramientas para mejorar sus vidas.

Tom Clancy.

La introducción de los SOs en nuestra vida diaria está encaminada a facilitar las interacciones que se puede llegar a tener con nuestro entorno y a mejorar la calidad de vida de los usuarios. Conforme la tecnología ha ido evolucionando, la población lo ha hecho de la misma manera, adaptándose a ella para utilizar todas las funcionalidades que nos ofrece.

Con la integración de los SOs en el mundo interconectado, la interacción que los usuarios tienen con la tecnología es más que necesaria. Sin embargo, los usuarios apenas son conscientes de la gran utilidad que les pueden proporcionar los SO en su día a día. Por este motivo se hace necesaria de IoT con interfaces naturales que hagan atractivo y fácil el uso de estos dispositivos. La integración de la realidad aumentada con IoT puede suponer una adaptación completa de los usuarios y los dispositivos inteligentes.

4.1. Componentes de la arquitectura ARIoT

ARIoT es una tecnología que nace de la fusión del IoT y la AR, que se encarga de proporcionar una visualización aumentada de la información que captan los SOs. Esta arquitectura supone una posible solución al problema que pueden llegar a tener los usuarios acerca de la percepción de la inteligencia existente en su entorno. Una interfaz que soporte AR para representar los SOs que se encuentran en un entorno, permitiría obtener una mejor percepción de la inteligencia presente y las funciones que puede aportar la misma. Por esto, se propone una arquitectura de un sistema IoT que haga posible la integración de la AR en un entorno inteligente:

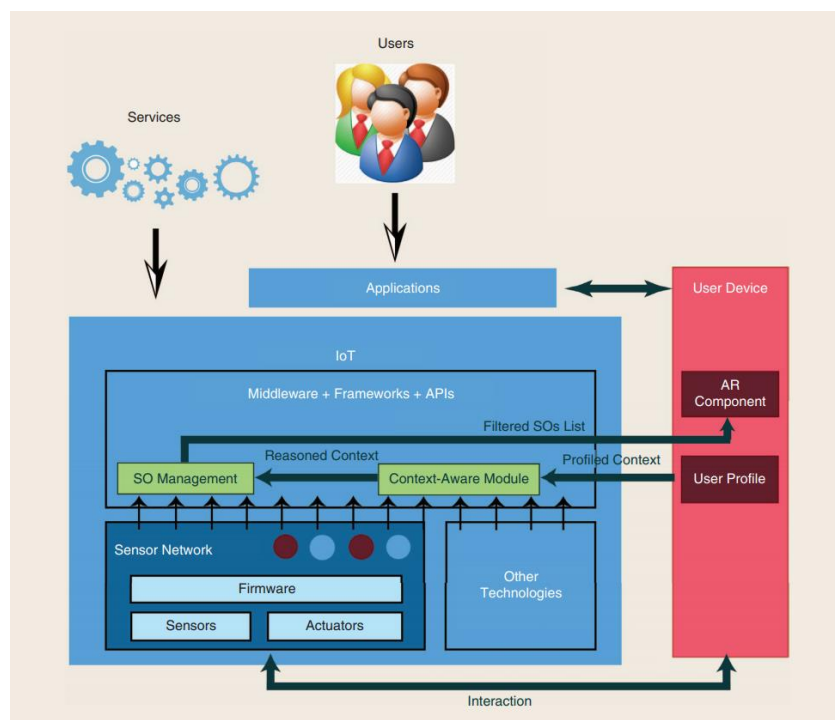


Figura 4.1. Arquitectura ARIoT propuesta [1].

La arquitectura representada en la Figura 4.1 está pensada para unificar a los sensores, al *Middleware* de la parte de IoT y al dispositivo del usuario, con un componente de AR, dentro de un mismo sistema en el que haya una gestión inteligente del SO mediante el uso del conocimiento del contexto o CA (del inglés, *Context-Awareness*). El CA se puede definir como un sistema que proporciona información relevante del entorno a un usuario, dependiendo de las preferencias de éste.

Con esto, la parte del sistema correspondiente al IoT está formada tanto por los sensores que componen al mismo como por el *Middleware*, el cual tiene un papel fundamental en la realización de aplicaciones IoT, dado que incluye una capa de abstracción, necesaria cuando en el mismo sistema existe heterogeneidad entre los distintos dispositivos.

Dentro de la arquitectura, el *Middleware* recibe la información que los sensores mandan, bien utilizando el método “push”, es decir, los sensores transmiten sus medidas basándose en un tiempo fijo o cuando notan un cambio en el entorno, o el método “pull”, en el cual es el propio *Middleware* el que pide los datos a los sensores. Estos datos tienen que estar identificados mediante un identificador descriptivo como un ID, localización o un protocolo de interacción [1]. Por esto, también es preciso que el sistema que controla el *Middleware* sea capaz de identificar los SOs que se encuentren en su entorno, teniendo, por ejemplo, una base de datos en la que se conozcan los dispositivos que entran o salen del entorno inteligente.

Además, dentro del sistema *Middleware* se encuentra el CA, o conocimiento de contexto, el cual es el encargado, en base al perfil del usuario que utilice el sistema, de representar los datos personalizados de los SOs que interesen al mismo. Esto se puede considerar como un proceso de personalización que hace que, al pasar los datos a la componente de AR, se representen los datos que el usuario haya seleccionado o que el propio sistema sepa que le pueden interesar, utilizando inteligencia artificial o IA.

Por otra parte, el componente principal de la representación mediante AR es el dispositivo del usuario, bien sea un *smartphone*, una tableta o unas gafas inteligentes. Este dispositivo es el responsable de la comunicación directa entre el SO y el usuario. En este dispositivo del usuario tendremos dos componentes: la parte relativa al perfil de usuario, que es la que transmite los privilegios e intereses de éste al módulo de CA del *Middleware*, y la componente de AR, que se encarga de la representación aumentada de la información que obtiene el dispositivo por parte de la gestión del SO. De esta manera, el usuario es capaz de percibir la presencia de un dispositivo inteligente en su entorno.

Por lo tanto, teniendo en cuenta toda la arquitectura y todo el proceso que se produce entre la interacción y la representación mediante AR de los datos, se puede decir que la arquitectura ARIoT se puede dividir en tres capas, según [1]:

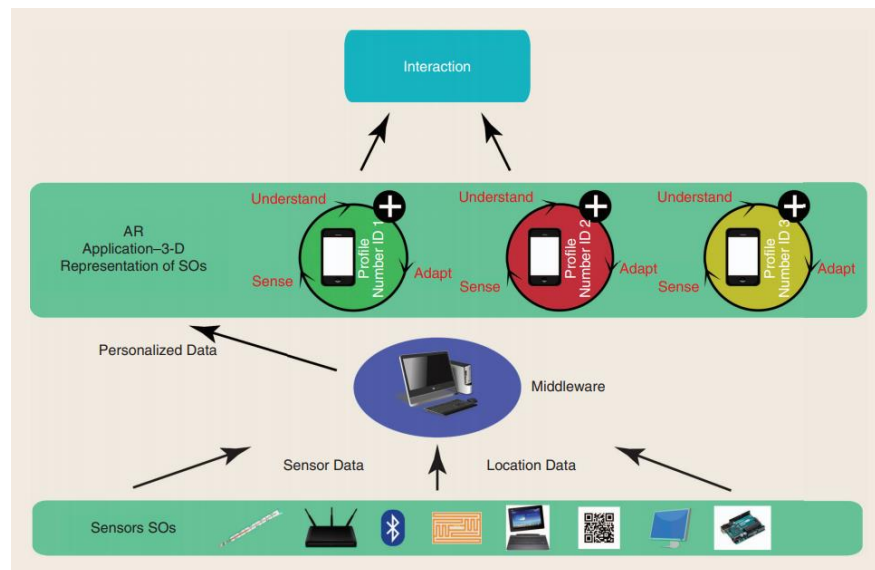


Figura 4.2. Capas de la arquitectura ARIoT [1].

En las capas de la arquitectura se puede observar cómo es dicho procedimiento. Los sensores se encargan de recoger y procesar las medidas al sistema de *Middleware* de la parte de IoT, el cual, mediante el módulo de CA y del perfil de usuario, es capaz de mandar la información a la componente AR, produciéndose así la interacción usuario-SO.

Los siguientes puntos de este capítulo se van a centrar en los aspectos claves de las componentes de la arquitectura, como la AR, *Middleware* y el Context-Awareness.

4.2. Realidad aumentada: concepto y métodos

La realidad aumentada es un término que se utiliza hoy día para describir un conjunto de tecnologías que hacen posible, en tiempo real, la mezcla de contenido generado mediante *software* con elementos reales mostrados en la pantalla del dispositivo de usuario [25]. Es una tecnología que hace el papel de medio para añadir información al mundo físico. Esto permite mostrar información combinando el mundo físico con el mundo virtual en tiempo real, mediante el uso de objetos 3D en la pantalla del dispositivo.

En cuanto a los aspectos principales de la realidad aumentada, en [26] se hace un apunte de una serie de éstos. En primer lugar, tenemos la representación en el mundo físico de información digital, bien sea información sintética, la cual es usada para simulaciones, o real, recogida por sensores o actuadores de su entorno. Esta información puede ser de cualquier tipo, ya sea una fotografía, un objeto 3D, mostrados por pantalla, o la reproducción de un sonido. Lo más importante de este primer aspecto es que el usuario permanece en el mundo físico, es decir, no hay nada que le haga pensar que se encuentra en otro lugar, como ocurre, por ejemplo, con la realidad virtual. Por otra parte, en segundo lugar, otro aspecto importante es que la información que se representa de forma aumentada puede depender de la localización geográfica en la que se encuentre el usuario, obteniendo así una perspectiva diferente para cada persona que use la AR. Por último, otro aspecto importante de la AR es la interacción que se le puede llegar a dar a la experiencia proporcionada al usuario, es decir, la forma en la que puede recibir la información o realizar cambios.

Por otra parte, en cuanto a los componentes principales que son necesarios para obtener una experiencia de realidad aumentada, en [26] se muestran seis a continuación:

- La **aplicación AR**, que se trata del programa que controla los distintos aspectos de la experiencia de AR, la cual se puede usar para distintos contextos del entorno. Esta parte es la que interactúa con los sensores, dispositivos y pantallas que son utilizados en la experiencia.
- El **contenido**, el cual es la clave para el funcionamiento de la aplicación AR y consiste en aspectos de la aplicación como pueden ser los objetos 3D que aparecen por pantalla.
- La **interacción**, que es el componente que permite al usuario ver o percibir la información aumentada en el mundo físico.
- La **tecnología**, la cual es requerida para un correcto funcionamiento de la experiencia y entre la que está la proporcionada por distintos elementos, tales como sensores, la computación el dispositivo para integrar lo virtual en lo físico y los mecanismos de representación por pantalla.
- El **mundo físico**, sin el cual no se podría conseguir una experiencia completa, dado que sin este elemento la AR no se sostendría.
- El **usuario**, que se constituye como el elemento principal porque la AR se concibe para dotar de estímulos visuales o auditivos al participante. Este elemento, por tanto, tiene un rol activo en la experiencia de AR porque sus acciones son las que hacen que el sistema funcione.

Por último, antes de pasar a mencionar los métodos con los que se pueden realizar la AR, se va a comentar el funcionamiento de la realidad aumentada que, según se menciona en [26], se trata de un proceso en dos pasos. En el primero de ellos la aplicación de AR necesita determinar el estado actual del mundo físico y del mundo virtual, como la localización del usuario o la posición de los marcadores utilizados, para así poder representar de la mejor forma posible la información aumentada. El segundo paso se trata de la necesidad por parte de la aplicación de mostrar en la pantalla del dispositivo la fusión del mundo virtual en el mundo físico, ayudándose para ello de tres elementos como son los sensores, el procesador del dispositivo y la pantalla del mismo.

Por su parte, los sensores son los encargados de ayudar a determinar el estado del mundo físico, dentro del entorno en el que se encuentre el usuario. El procesador se encarga de coordinar y analizar los datos proporcionados por los sensores, almacenándolos y recuperándolos para generar la respuesta AR en la pantalla del dispositivo. Por último, la pantalla es la encargada de crear la unión entre ambos mundos, lo cual ayuda a realizar la interacción usuario-SO.

4.2.1 Métodos existentes para la realidad aumentada con el uso marcadores

Los métodos actuales existentes para implementar aplicaciones de AR en un determinado escenario se basan en el reconocimiento de imágenes. Este proceso de reconocimiento se realiza mediante el dispositivo que se utilice, el cual es capaz de procesar dónde y cómo se tiene que llevar a cabo el aumento de la información según el entorno. Como se recoge en [27], existen tres métodos:

- **Pattern:** se trata de un método de realidad aumentada que utiliza un patrón de reconocimiento básico basado en el uso de marcadores. Cuando el dispositivo reconoce al marcador, el sistema introduce en el área de este la información, ya sea en forma de objeto 3D, audio o vídeo.



Figura 4.3. Método "pattern" [27].

- **Outline:** en este método, el dispositivo se encarga de reconocer una parte del cuerpo del usuario, añadiendo a dicha parte un objeto 3D, con el que el usuario es capaz de interactuar. Además, el sistema es capaz de ajustar dicho objeto de acuerdo con los movimientos que realice la persona. Se trata de un método similar al reconocimiento facial, pero con el añadido de la AR.

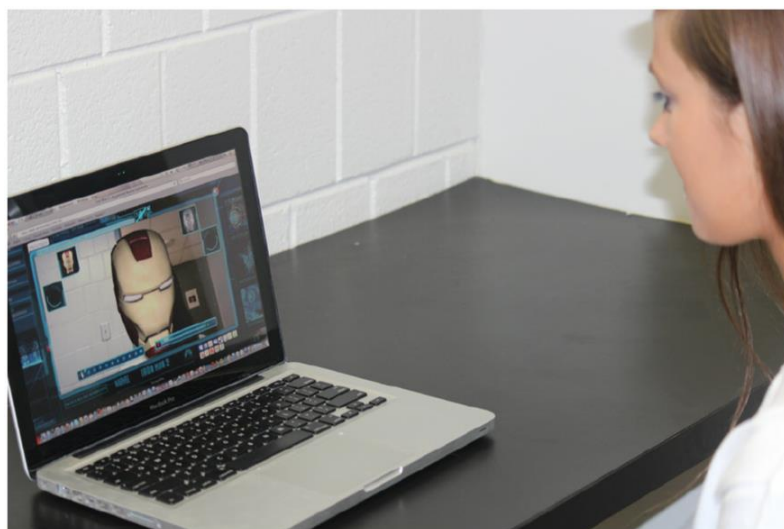


Figura 4.4. Método "outline" [26].

- **Surface:** es un método realizado para proveer al usuario de información en tiempo real usando superficies que respondan al tacto del sujeto. Es una tecnología denominada como "depth-camera", en la que la cámara del dispositivo detecta el rango de los objetos físicos y las superficies para determinar dónde está interactuando el usuario y saber cómo o qué hacer ante esa interacción.



Figura 4.5. Método “surface” [26].

4.2.2 Métodos “markerless” para la realidad aumentada

Hasta ahora los métodos propuestos hacían uso de marcadores para la interacción con realidad aumentada entre el dispositivo del usuario y el objeto del sistema IoT. Teniendo en cuenta que el mayor problema para las aplicaciones de AR es que el sistema tiene que ser capaz de conocer la localización de los objetos, el sistema AR basado en métodos que usan marcadores da lugar a ciertas complicaciones cuando el número de objetos y la dimensión del sistema aumentan.

Aunque los métodos de marcadores son muy fáciles de implementar y proporcionan baja latencia, cuando la escalabilidad aumenta, aparecen los siguientes requerimientos para que el método funcione [1]:

- **Visión directa**, es decir, los marcadores tienen que ser directamente detectados por la cámara del dispositivo para que la AR funcione. Por lo tanto, es necesario una visión limpia y directa de los marcadores para que se produzca la interacción con la AR.
- **Distancia**, debido a que el objeto IoT debe estar cerca del dispositivo con el que se realice el reconocimiento del entorno, si no la visualización AR puede no producirse.
- La necesidad de **muchos marcadores**, dado que, si en un sistema IoT se supone un gran número de objetos, cada objeto tiene que ser identificado por un único marcador.
- Una **etiqueta fija** para cada marcador, la cual no cambia según el estado del objeto y los datos que este facilita, pudiendo presentar sólo un conjunto de información por etiqueta. Esto resulta un inconveniente porque el marcador no puede responder a cambios en el estado del sensor.

Todos estos requerimientos se pueden subsanar mediante la utilización de marcadores dinámicos, aunque estos conllevan un alto coste para la implementación de un sistema de grandes dimensiones. Por esto, se puede plantear otros métodos, conocidos como “markerless”, los cuales también tienen algunos límites.

Por una parte, para los sistemas IoT que se diseñan para un despliegue “outdoor”, el uso de GPS es la mejor opción existente. Esta tecnología, combinada con sensores y otros métodos de seguimiento, como el seguimiento de imágenes, puede hacer que este método mejore la precisión. Al estar pensado para sistemas de exterior, como por ejemplo una *Smart City* como se observa en la Figura 4.6, se necesita un software para conocer los patrones de imágenes y para conexiones constantes con bases de datos. Este método es una forma bastante buena para el rastreo de objetos con una latencia baja, pero puede ocasionar problemas importantes dentro de un sistema con tantos objetos. Un ejemplo de este método es el método *Location*, también conocido como buscador AR. Este método está diseñado para ofrecer información de cualquier cosa o lugar que esté cerca del usuario cuando éste enfoca con su dispositivo al entorno.

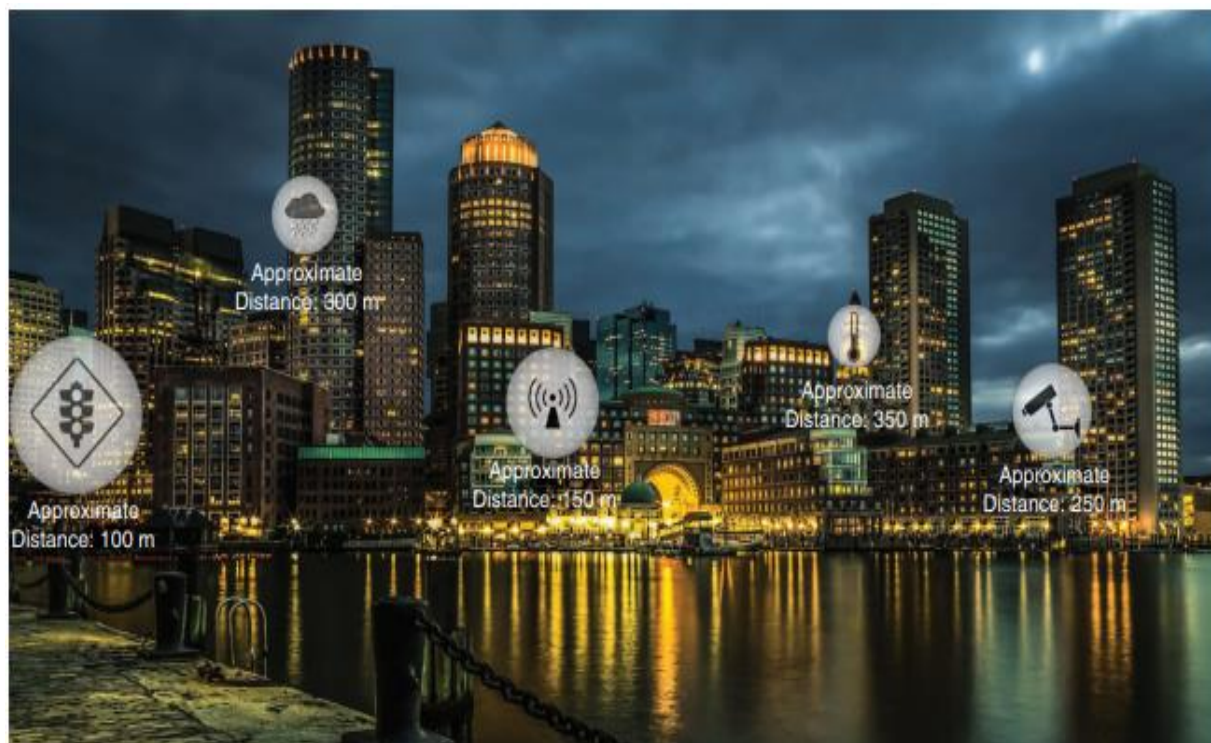


Figura 4.6. Método “location” [1].

Por otra parte, para los sistemas “indoor” se propone la tecnología Bluetooth LE, comentada en el capítulo anterior, la cual es capaz de proporcionar a los usuarios el máximo potencial de los servicios IoT. Este método hace uso de los que se conocen como “balizas BLE”, las cuales proporcionan tiempos cortos referentes al escaneo de la imagen y poca consumición de potencia, además de que es un método poco costoso. Con estas balizas se pueden obtener unos rangos de distancia ventajosos, dado que se pueden configurar, yendo desde los pocos centímetros hasta unos cuantos metros, además de que su ciclo de vida puede llegar a alcanzar los tres años. En definitiva, estos métodos resultan beneficiosos para el usuario porque la localización es crucial cuando este no conoce, mediante el visionado de un objeto, que se encuentra en un entorno inteligente.

4.3. Capa de *Middleware*: necesidad y tipos de arquitecturas

En el anterior capítulo se comentó que, dentro de una arquitectura IoT, se podía encontrar una gran cantidad de dispositivos, cada uno de ellos pudiendo tener distintos protocolos y distintos métodos de interacción con el usuario. Para poder cubrir todas estas cuestiones, es necesario integrar en el sistema IoT una capa de *software*, denominada *Middleware*.

La capa *Middleware* representa una capa intermediaria entre las cosas, denominadas objetos, y la red de Internet. Además, esta capa se encarga de la complejidad del sistema, la cual se presenta cuando se produce una interacción con el dispositivo del usuario. Para un correcto desarrollo de la capa *Middleware* en una arquitectura IoT, hace falta que esta capa recoja la suficiente capacidad para abordar las siguientes cuestiones, según se recoge en [28]:

- **Interoperabilidad:** se trata de una cuestión referente a la heterogeneidad existente entre los dispositivos IoT. Esta interoperabilidad está presente en la abstracción del dispositivo frente a las capas superiores, la cual puede ser según la técnica, definida según la ETSI como la asociación entre *hardware* y *software* para comunicaciones M2M, según la sintáctica, es decir, según el formato de los datos que mandan o reciben los objetos, y según la semántica, la cual se basa en la capacidad de intercambiar datos que los usuarios sean capaces de entender.
- **Gestión y descubrimiento de dispositivos:** los dispositivos (objetos) deben tener un sistema para interactuar con otros, previo inicio de comunicación entre estos. En este sistema, el *Middleware* es el encargado de actualizar la información de enrutamiento y avisar a los objetos, independientemente del protocolo de enrutamiento que utilicen.

- **Seguridad y privacidad:** esta capa debe asegurar la autenticación, confidencialidad e integridad de los datos. Además, debe ser capaz de gestionar diferentes tipos de privilegio entre objetos.
- **Abstracción de la capa de aplicación:** el *Middleware* también se encarga de proveer al usuario de la interfaz que ha sido diseñada para la interacción de este con los objetos.
- **Gestión de datos:** cuestión referente a que dentro de un sistema IoT hay un gran flujo de datos que se intercambian entre objetos, terminales o dispositivos finales y bases de datos.

4.3.1 Arquitecturas existentes para el *Middleware*

Las arquitecturas existentes para el desarrollo del *Middleware* para aplicaciones IoT son tres: basada en servicios, basada en la nube y basada en actuadores. La arquitectura basada en servicios es la adoptada en arquitecturas SOA y permite a los desarrolladores de la misma añadir o desplegar los dispositivos IoT para que aporten servicios a los usuarios. La segunda arquitectura, basada en la nube, limita el número y tipo de dispositivos que se pueden incluir en el sistema, pero permite al usuario conectarse con los objetos, que estos recojan datos y que dichos datos sean interpretados de una forma correcta. En el último tipo, se propone una arquitectura IoT enfocada al “*plug and play*” [29].

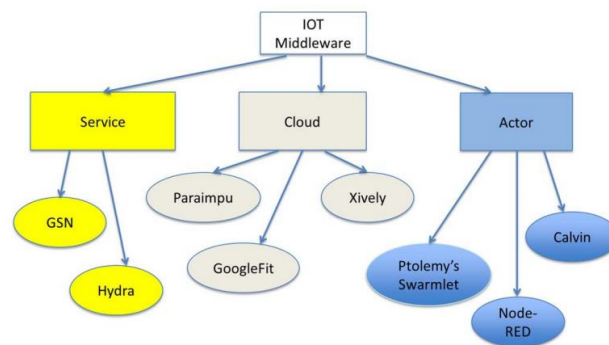


Figura 4.7. Principales sistemas de *Middleware* existentes para el IoT [29].

Las diferencias principales entre las tres arquitecturas son la capacidad de cada una para admitir nuevos dispositivos IoT, los servicios que es capaz de ofrecer y dónde se integra la capa *Middleware* en el sistema.

La arquitectura **basada en servicios** está dividida en tres capas, las cuales consisten en: capa física, constituida por los objetos, capa de virtualización, formada por la infraestructura de servicios y sistemas basados en la nube, y la capa de aplicación. Es en la capa de virtualización donde se lleva a cabo toda la gestión computacional del sistema, como el control de acceso al medio de los objetos, el manejo de los datos o el procesamiento de eventos del sistema. Este tipo de arquitectura se suele desplegar en servidores o en los sistemas de nube dentro del sistema IoT. Además, se puede configurar para la seguridad referente a la protección de los datos que recogen los objetos. Sin embargo, no está diseñada para dispositivos finales, como *smartphones*, y no soporta, por tanto, comunicaciones dispositivo a dispositivo, es decir, está pensada para interacciones entre dispositivo de usuario y objeto IoT.

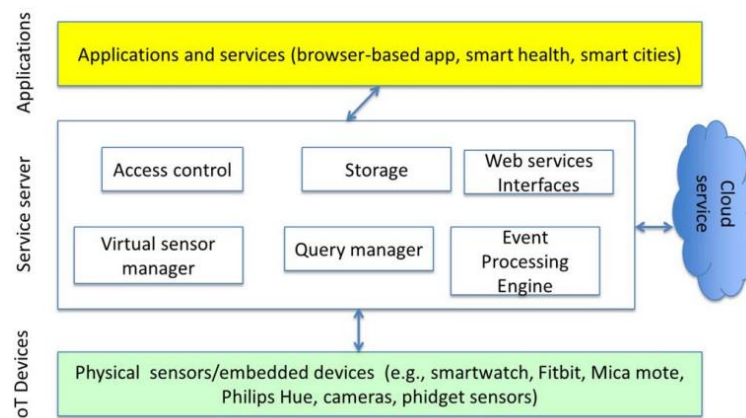


Figura 4.8. Arquitectura *Middleware* basada en servicios [29].

Por otra parte, la arquitectura **basada en la nube** se encuentra limitada a la disponibilidad del sistema de nube escogido. Las funcionalidades de dicho sistema están ligadas a una serie de APIs, las cuales proporcionan una alta eficiencia para el almacenamiento de datos y proveen herramientas para la monitorización de la computación existente en todo el sistema. En este tipo de arquitectura, el usuario no puede decidir cómo ni dónde se almacenan los datos, teniendo estos que confiar en el proveedor del servicio de nube para mantener la privacidad e integridad de sus datos. Además, como pasa en la arquitectura anterior, proporciona una seguridad débil entre los objetos y la capa *Middleware*, dado que esta no se puede integrar en los dispositivos IoT.

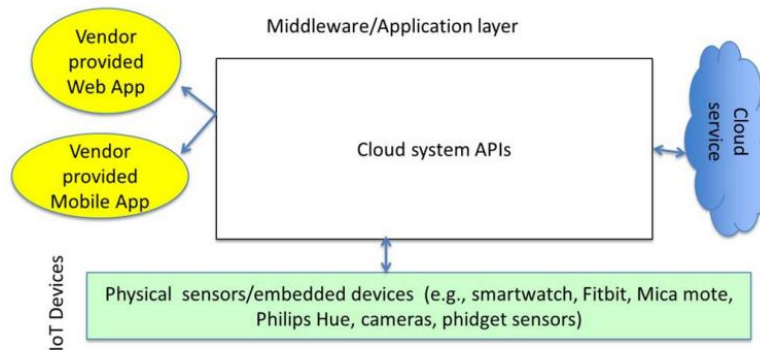


Figura 4.9. Arquitectura Middleware basada en la nube [29].

Por último, la arquitectura **basada en actuadores** se puede dividir en dos capas, que son la capa física, similar a la existente en las otras arquitecturas y que recoge a sensores y objetos, y la capa de acceso al servicio, ambas acompañadas por los servicios de nube. Esta arquitectura está diseñada para ser simple y puede integrarse en todas las capas, es decir, tanto en los dispositivos finales como en los IoT. Además, un factor diferencial de esta arquitectura es que un actuador del sistema *Middleware*, el cual proporciona servicios de almacenamiento de datos, puede obtener dichos datos de la nube en cualquier momento. También cabe recalcar que esta es la arquitectura que presenta mejor latencia y escalabilidad para aplicaciones IoT de larga escala, aunque ofrece interoperabilidad entre dispositivos mediante la adopción de algunos estándares específicos.

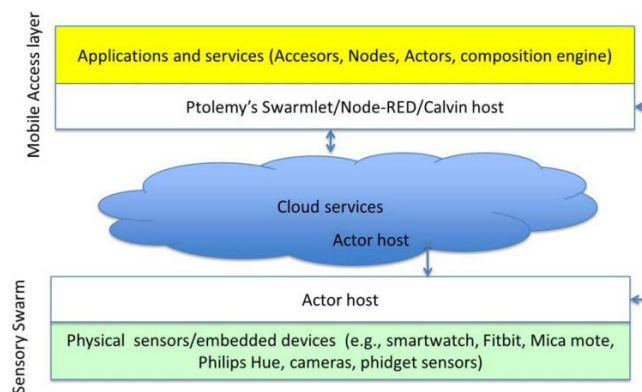


Figura 4.10. Arquitectura Middleware basada en actuadores [29].

4.3.2 El conocimiento del contexto dentro de la capa *Middleware*

La definición de un sistema en el que se hace uso de *Context-Awareness* puede ser como aquel en el que utiliza el contexto, siendo esto la información que puede utilizarse para caracterizar la situación de una entidad, donde la entidad puede ser una persona, lugar u objeto considerado como relevante para la interacción entre usuario y dispositivo IoT, para dar información o cualquier servicio al usuario, dependiendo de las preferencias de este [30].

En la arquitectura ARIoT, el módulo de CA se añade principalmente para ofrecer información específica al usuario según su perfil y sus preferencias. Con esto, el sistema puede seleccionar qué sensores, de todos los que conforman la red, son convenientes para el usuario, dependiendo de la información que estos proporcionen. Por lo tanto, el usuario no visualizará los otros sensores de forma irrelevante, reduciendo así la carga para el dispositivo. Dentro de la arquitectura, el módulo de CA recibe dos tipos de contextos, que son:

- **Contexto sensorial**, referido a la capa de sensores, la cual es la que se encarga de recoger una gran cantidad de datos, teniendo el sistema que organizarlos según el tipo de estos. Este tipo de contexto se encarga de dotar al sistema de una mejor percepción del entorno de cara a ofrecer y satisfacer al usuario de información que este prefiera.
- **Perfil de usuario**, siendo necesario asignar uno a cada usuario que se encuentre utilizando la aplicación IoT, para que el sistema pueda adecuar la experiencia del usuario con la aplicación, basándose en las preferencias de este.

Por lo tanto, este procedimiento puede ser beneficioso para el usuario, dado que este puede llegar a comprender de mejor forma la situación del sistema y cómo funciona el mismo. Sin embargo, aunque suponga una mejor comprensión, la mayor mejora que proporciona añadir el módulo de CA al *Middleware* es que se incrementa el interés del usuario por los nuevos servicios que es capaz de ofrecer el IoT.

5 DESARROLLO PRÁCTICO DEL TRABAJO

La alegría de ver y entender es el más perfecto don de la naturaleza.

Albert Einstein.

EN este capítulo se pretende dar una pautas sobre el desarrollo que se ha llevado a cabo para la implantación de la red de sensores en un entorno AR, para el que se ha utilizado las herramientas mencionadas en el segundo capítulo de este documento.

Este capítulo, como ya se ha mencionado en la estructura, estará dividido en tres partes: la recolección e inserción de datos, el desarrollo de la aplicación de AR y la puesta en marcha del sistema de sensores junto a la aplicación móvil.

5.1. Recolección e inserción de datos

Para llevar a cabo esta parte del trabajo, cabe destacar que ante la falta de medios físicos, como son la Raspberry Pi y el módulo Sense Hat, se pasó a utilizar una Raspberry en formato virtual mediante el uso de la herramienta VMWare Workstation. Esta herramienta nos da la posibilidad de poder tener distintos entornos de trabajo, cada uno de estos corriendo distintos sistemas operativos, lo cual se puede realizar en un mismo equipo.

Con esto, se ha podido realizar toda la parte en la que se ha necesitado el uso de esas herramientas Hardware y la tercera parte de este proyecto. Una vez seleccionado el sistema operativo con el cual arrancar el programa, en nuestro caso es Raspberry Pi Desktop basado en Linux, lo primero que se hizo fue instalar en la Raspberry una base de datos en la que se pudiera guardar las medidas realizadas por Sense Hat. Tras esto, el siguiente paso fue el desarrollo de un script en lenguaje Python, con el que se realizarán medidas por parte de dicho módulo y se insertarán dichas medidas en la base de datos, y otro script, en el mismo lenguaje, para conectar la Raspberry a la base de datos. Por lo tanto, una pequeña aproximación al esquema de esta primera parte del trabajo, en cuanto a elementos se refiere, es la representada en la Figura 5.1:

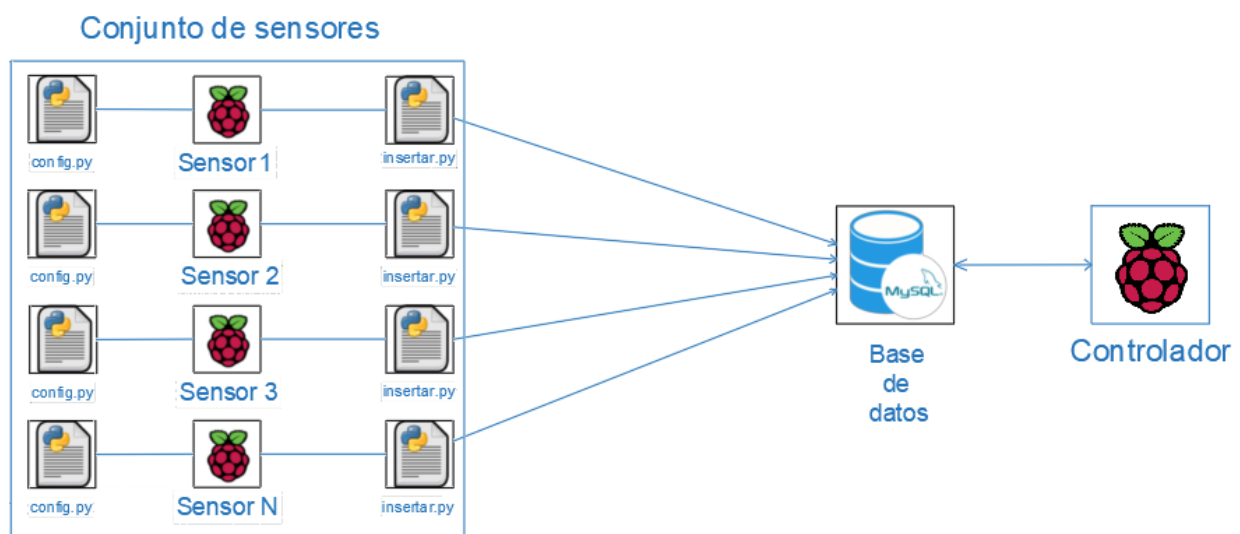


Figura 5.1. Esquema de recolección e inserción de datos.

5.1.1 Creación del servidor LAMP en la Raspberry Pi controladora

Para comenzar esta primera parte del trabajo, partiremos del esquema mostrado en la figura anterior, en la que una Raspberry va a actuar de controladora y en la que se va a tener activo un servidor con el que poder almacenar todas las medidas realizadas por los sensores que conforman la red. Por esta razón, comenzaremos con la instalación de un servidor LAMP en la Raspberry que se considere como central, todo esto realizado mediante un terminal de dicha Raspberry.

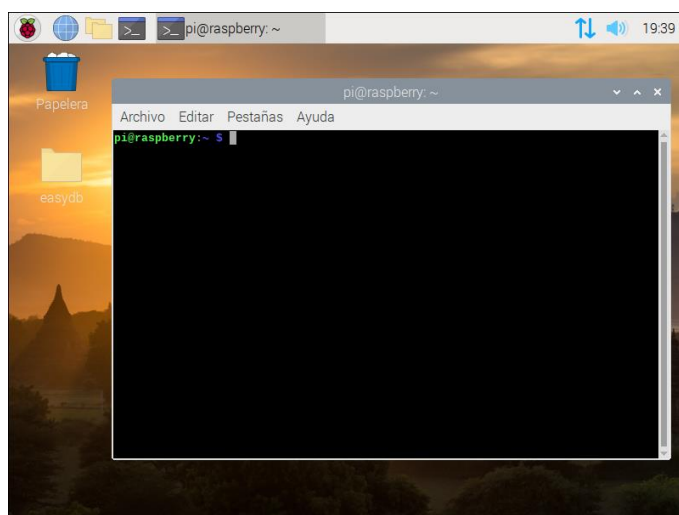


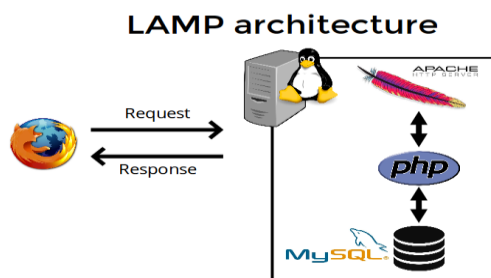
Figura 5.2. Entorno de trabajo para la creación del servidor LAMP.

Este servidor ofrece una solución de muy bajo coste para poder crear web dinámicas, mediante el uso de PHP, ya que con otros lenguajes web no se consigue tanto dinamismo, y la posibilidad de almacenaje de datos utilizando un sistema de gestión de base de datos. Además de las posibilidades que ofrece, este tipo de servidor requiere de una infraestructura especial, formado por un sistema operativo, un servidor web, un gestor de base de datos y uno o más lenguajes de programación.

El funcionamiento de este servidor es muy simple, dado que el sistema operativo sirve de base para el funcionamiento del servidor web, el lenguaje de programación escogido es el que ofrece dinamismo al entorno y este último es el que accede a la base de datos para la representación de los mismos. Además, cada una de estas componentes puede ser sustituida por otras con las mismas características, lo que hace que este servidor esté entre los más usados.

Por otra parte, resulta interesante analizar el significado de cada una de las letras que forman el nombre del servidor [5], para poder así entender su papel realizado dentro de la arquitectura IoT propuesta:

- **L de Linux**, que hace referencia al sistema operativo en el que se va a instalar el servidor, dado que Raspberry Pi Desktop está basado en este sistema operativo. Por lo tanto, esta primera letra sirve para la diferenciación con el resto de sistemas, ya que si hubiese sido una W nos referiríamos a Windows o en el caso de M sería MacOS.
- **A de Apache**, que se trata del servidor web, es decir, la parte más importante. Es la parte que realiza el papel de receptor de peticiones por parte de un sitio web y envía una respuesta a dicha petición.
- **M de MySQL**, es decir, el sistema gestor de base de datos necesaria para almacenar los datos medidos por los sensores. MySQL se define como un sistema de gestión de bases de datos relacionales de código abierto capaz de almacenar datos de distintas aplicaciones, que en nuestro caso serán los distintos nodos de la red. Esta información es almacenada mediante un formato que se puede consultar mediante el lenguaje SQL, que hace referencia a lenguaje de consulta estructurado.
- **P de PHP**, que es el lenguaje de programación de código abierto del que se sirve el servidor web Apache para crear páginas web con procesos dinámicos, como, por ejemplo, extraer datos de una base de datos. Esto nos permitirá añadir contenido a la página web intermediaria donde se encontrarán los datos y que se utilizará principalmente en la última parte de este capítulo, donde la aplicación móvil accederá a dicha página para obtener los datos de la base de datos.



Autor desconocido está bajo licencia [CC BY-SA](#)

Figura 5.3. Arquitectura de un servidor LAMP.

Como el objetivo de este trabajo no es enseñar a crear el servidor LAMP, si el lector necesita más información sobre cómo crearlo, para implementar la arquitectura propuesta en el presente trabajo, puede consultar los pasos a seguir para ese fin en [5].

5.1.2 Configuración de la base de datos

Una vez se haya instalado el servidor LAMP, el siguiente paso para conseguir la implementación del sistema propuesto es crear la base de datos en la que se almacenarán las medidas que recojan los sensores de la red. Posteriormente, se debe crear la tabla que recoja una representación estructurada de los datos para poder acceder a ello más tarde mediante lenguaje SQL.

Por lo tanto, en este apartado se abordará la creación de la base de datos y de la tabla que estructurará dichos datos dentro del servidor. Si el lector desea conocer algún aspecto más sobre el lenguaje SQL, puede consultar [6].

5.1.2.1 Creación de la base de datos

En cuanto a la creación de la base de datos que se utilizará en este trabajo, se ha llevado a cabo mediante consola. Para realizar este paso, primero tendremos que entrar en el sistema gestor de bases de datos MySQL mediante el siguiente comando:

```
sudo mysql -u root -p password
```

Una vez hecho esto, nos encontraremos dentro de MySQL y podremos realizar todo tipo de acciones sobre las bases de datos que se encuentren en el sistema, desde crear nuevas hasta eliminar las ya existentes. Por lo tanto, para crear la base de datos necesaria utilizamos el comando CREATE. Este comando es el comando base del lenguaje SQL para crear un elemento, ya sea base de datos, tablas, índices, etc. Por lo tanto, se usa el siguiente comando, que tiene que utilizarse dentro del menú de MySQL en consola:

```
CREATE DATABASE basedatos;
```

Tras esto, ya se tendría una base de datos dentro del servidor LAMP, en la cual se podría empezar a almacenar los datos que se desee. El siguiente paso consiste en la creación de la tabla que contendrá los datos de los distintos sensores del sistema.

5.1.2.2 Creación de la tabla de datos

Llegado a este punto, el procedimiento para crear la tabla para almacenar los datos medidos es similar a la creación de la base de datos, dado que se utiliza el mismo comando. Sin embargo, la única peculiaridad que tiene este paso es que se debe seleccionar la base de datos en la que crear la tabla, utilizando lo siguiente:

```
USE basedatos;
```

Para esto se ha utilizado un nuevo comando SQL, USE, que sirve para seleccionar una base de datos dentro de MySQL. A partir de aquí, sólo queda crear la tabla, para lo que se utilizará el comando mencionado anteriormente, CREATE:

```
CREATE TABLE `temperatura`.`basedatos` (
  `id` int(0) NOT NULL AUTO_INCREMENT COMMENT 'Identificador de la
    medida',
  `nodeid` int(0) NOT NULL COMMENT 'Identificador del nodo fuente',
  `measdate` datetime(0) NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT
    'Fecha de la medida',
  `temp` double NOT NULL COMMENT 'Temperatura',
  PRIMARY KEY (`id`));
```

Con el anterior comando se ha creado dicha tabla, además de crear las columnas de la misma. Esto se podría haber hecho de forma separada, mediante el comando ALTER TABLE, tal y como se muestra en [6]. Como se puede apreciar, la tabla está formada por cuatro columnas (cuyos datos no pueden ser nulos), las cuales son:

- **“ID”** o Identificador de la medida: hace referencia al número de la medida realizada por el sensor y es de tipo entero (INTEGER). Además se especifica que sea autoincremental porque se va a medir constantemente y, gracias a esta columna, se puede conocer cuantas medidas lleva realizada el sensor.
- **“NodeID”** o Identificador del nodo fuente: se trata del sensor que realiza la medida y es una columna necesaria porque tenemos más de un sensor en la red. También es del tipo entero.
- **“Measdate”** o Fecha de la medida: como su propio nombre indica, se trata de la fecha en la que fue realizada la medida en formato YYYY-MM-DD hh:mm:ss. Es un dato del tipo TIMESTAMP, el cual combina los valores de tiempo y fecha.
- **“Temp”** o Valor de la temperatura medida: se trata de la medida que realiza el sensor y es una columna necesaria porque es el valor fundamental para desarrollar el trabajo. Este último dato es del tipo DOUBLE, dado que la temperatura no es un número entero y proporciona decimales.

id	nodeid	measdate	temp
42	1	2020-04-08 15:27:51	14.3
43	1	2020-04-08 15:28:51	38.5
44	1	2020-04-08 15:29:51	38.5
45	1	2020-04-08 15:30:51	25.6
46	1	2020-04-08 15:31:51	25.6
47	1	2020-04-08 15:32:51	90.3
48	1	2020-04-08 15:33:51	90.3
49	1	2020-04-08 15:34:51	90.3
50	1	2020-04-08 15:35:51	90.5
51	1	2020-04-08 15:46:18	16.4
52	1	2020-04-08 15:47:18	18.3
53	1	2020-04-08 15:48:18	28.9
54	1	2020-04-08 15:49:18	17.6

Figura 5.4. Tabla de temperaturas del nodo 1 del sistema.

Hecho esto, sólo falta el método para introducir los valores medidos por los distintos nodos de la red en la tabla de la base de datos, lo cual se especifica en el siguiente apartado.

5.1.3 Medición e inserción de datos en la base de datos

En esta última parte, en la que se conseguirá introducir las medidas de los sensores en la tabla de temperatura de la base de datos creada, se utilizará como base el programa los códigos expuestos en el [Anexo A](#), ambos escritos en lenguaje Python. Por una parte, se hace uso de la librería EasyDB, con la que se podrá realizar la conexión con la base de datos para hacer cualquier modificación de esta, de una forma muy sencilla. Por otra parte, con insertar.py se puede insertar aquellos valores medidos por los sensores de la red, utilizando lenguaje SQL como se mostrará a continuación.

5.1.3.1 Configuración de la librería EasyDB

Como ya se ha comentado, esta librería facilita al usuario la configuración que este tiene que realizar para conectarse a la base de datos requerida. Esto es posible gracias a que nos facilita todos los ficheros necesarios para ello, teniendo que modificar únicamente el fichero `config.py` de la misma.

Accediendo a dicho anexo se puede observar como ha sido configurado dicho fichero, habiendo tenido que modificar los campos referentes al nombre del usuario de la Raspberry Pi en la que se encuentra la base de datos, la contraseña de esta y el nombre de dicha base de datos a la que nos queremos conectar. Además, nos da la posibilidad de utilizar dos bases de datos al mismo tiempo, aportando mayor funcionalidad.

5.1.3.2 Inserción de datos en la base datos

En este caso, el fichero a utilizar es `insertar.py`, en el cual, por una parte, se importa la librería correspondiente a las conexiones con las bases de datos declaradas en el apartado anterior, esto es, EasyDB. Por otra parte, se importan tanto Sense Hat, para obtener el valor de temperatura medido, como la librería `time`, para establecer un lapso de tiempo entre medida y medida.

Además, entrando un poco en el código para comprender cómo se inserta la medida, se hace uso la siguiente sentencia de lenguaje estructurado SQL:

```
INSERT INTO temperatura(nodeid,temp) VALUES (numNodo,temp)
```

Utilizando los comandos `INSERT INTO`, especificando los datos de la tabla que queremos modificar, y `VALUES`, pasando los valores de dichos datos, se consigue la inserción de los datos medidos por los nodos en la base de datos. Cabe resaltar que el dato referente a la fecha de medida no se tiene que indicar en la sentencia, porque tiene un valor por defecto (`CURRENT_TIMESTAMP`).

Por último comentar que, en este trabajo, ante la falta de medios para el montaje del sistema, se ha optado por utilizar un programa, el cual se encuentra dentro del sistema operativo de la Raspberry Pi, que se trata de *Sense Hat Emulator*. Este programa, cuya interfaz se muestra en la Figura 5.5, aporta al usuario una interfaz en la que modificar todos los parámetros que mide dicho módulo.

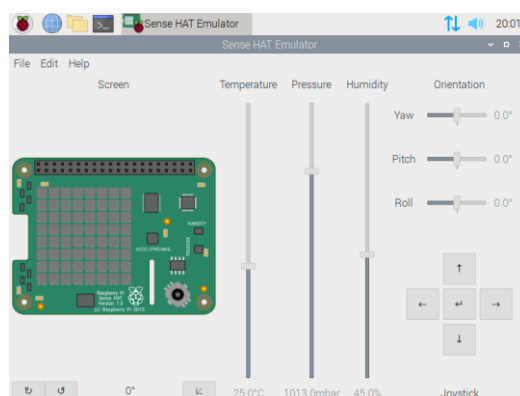


Figura 5.5. Interfaz de Sense Hat Emulator.

Llegados a este punto, ya se tiene una base de datos funcional en la cual se están almacenando los datos recogidos por los sensores del sistema y se pasaría al diseño de la aplicación AR.

5.2. Desarrollo de la aplicación móvil

Tras haber creado la base de datos y comprobar el funcionamiento de la recolección de datos por parte de los sensores del sistema, el siguiente paso es el desarrollo de la aplicación AR, que nos permitirá la visualización adecuada de los mismos. Para esta parte, se utilizarán las dos herramientas software mencionadas en el capítulo 2 de esta memoria, Unity3D y Vuforia SDK. Con la primera se podrá elaborar la interfaz de la aplicación, mientras que la segunda nos ofrece dotar de la realidad aumentada a dicha app.

Es por esto que, en este apartado se tratará de exponer todo el proceso de creación de la app móvil, empezando con Vuforia SDK, para posteriormente rematar este desarrollo con Unity3D.

5.2.1 Vuforia SDK: creación de los marcadores

Como ya se comentó anteriormente, para el reconocimiento de imágenes hay distintos métodos, entre los que están los basados en la utilización de marcadores y los “*markerless*”. Estos marcadores se conocen como rastreables, del inglés “*trackables*”. Los marcadores son una parte fundamental de la AR y pueden ser desde códigos QR hasta imágenes, los cuales hacen que el objeto de realidad aumentada aparezca en la pantalla del dispositivo realizando diferentes acciones. Los métodos existentes son, como ya se ha visto, marcador simple, marcadores múltiples o botones virtuales.

En este trabajo, como también se ha dicho, se ha optado por el reconocimiento de imágenes mediante marcadores múltiples, porque el sistema va a constar de más de un nodo y cada uno de ellos estará ligado a una imagen distinta. La elección de estas imágenes que irán asociadas a los distintos sensores es libre, pudiendo utilizar cualquier imagen que se desee. Por lo tanto, se ha decidido utilizar un código QR para cada sensor, los cuales contienen como información una cadena de texto que hace referencia al nodo de la red en cuestión.

Para crear dichos marcadores se ha utilizado una aplicación web diseñada para ello, de manera que los marcadores usados son los representados en la Figura 5.6.



Figura 5.6. Marcadores utilizados para la aplicación móvil.

5.2.2 Vuforia SDK: creación de la base de datos de marcadores

Para este paso, se usará el gestor de marcadores de Vuforia, que se trata de una herramienta online, proporcionada por Qualcomm, la cual es capaz de crear una base de datos de marcadores para su uso en las aplicaciones de AR. Esta herramienta es accesible mediante el siguiente enlace:

<https://developer.vuforia.com/vui/develop/databases>

Una vez creada la base de datos mencionada, se importan los códigos QR como imágenes. Vuforia permite introducir en las bases distintos tipos de marcadores, como son imágenes simples, cilindros, objetos 3D y cuboides. Además, se puede asignar el nombre del marcador y la dimensión de éste, parámetro importante ya que nos proporciona la escala del marcador en la escena AR. Una vez realizado todo el procedimiento, la base de datos quedaría tal y como se muestra en la Figura 5.7.

AppTFG [Edit Name](#)
Type: Device

Targets (2)

[Add Target](#) [Download Database \(All\)](#)

☐	Target Name	Type	Rating ①	Status ▾	Date Modified
☐	 Nodo2	Single Image	★★★★☆	Active	Jun 16, 2020 19:34
☐	 Nodo1	Single Image	★★★★☆	Active	Jun 16, 2020 19:34

Figura 5.7. Base de datos de marcadores de Vuforia.

Por último, para poder usar esta base de datos en el proyecto de Unity3D, hace falta importarla desde dicha herramienta web. Esto se puede hacer descargando la base de datos mediante el botón de descarga de la figura anterior. Tras esto, lo único que queda para tener lista la aplicación es realizar la disposición de los elementos en la interfaz, lo que se hará mediante Unity3D.

5.2.3 Unity3D: modelado de la interfaz de realidad aumentada

La interfaz de una aplicación móvil creada en Unity3D está conformada por diversas escenas que, mediante la programación de la app, van apareciendo por pantalla dependiendo de las acciones que realice el usuario. En estas escenas se encuentran los objetos que se hayan introducido para dar dinamismo a la aplicación, los cuales, en el caso de una aplicación de AR, son objetos 3D.

Por otra parte, la funcionalidad de una aplicación de AR es muy sencilla, de igual modo que su desarrollo. Esto es, una aplicación de este tipo se ayuda de la cámara del dispositivo donde se instala para representar en el mundo real un objeto de AR cuando la cámara encuentra en el entorno un marcador que esté en la base de datos de marcadores de Vuforia.

En este proyecto, el objeto que se ha querido representar mediante la realidad aumentada ha sido una especie de indicador de temperatura digital, el cual aparece en la pantalla del dispositivo cuando se capta con la cámara uno de los códigos QR que se han creado en apartados anteriores. Este objeto está formado por lo siguiente:

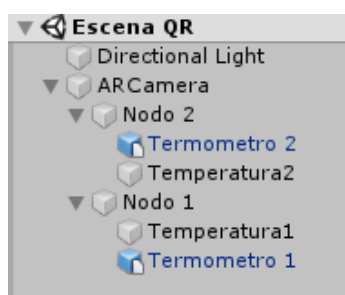


Figura 5.8. Jerarquía de la escena AR creada.

Como se puede apreciar, dicha escena está formada por la cámara de AR, sin la cual no podría representarse nada mediante realidad aumentada, y dentro de la misma están los dos nodos del sistema, que son los marcadores que se han importado de la base de datos de Vuforia. En la descendencia de dichos marcadores nos encontramos con dos objetos, el primero de ellos es un objeto prefabricado, que hace referencia a una especie de modelo 3D de un indicador de temperatura, y una cadena de texto, que se usará para representar la temperatura del nodo.

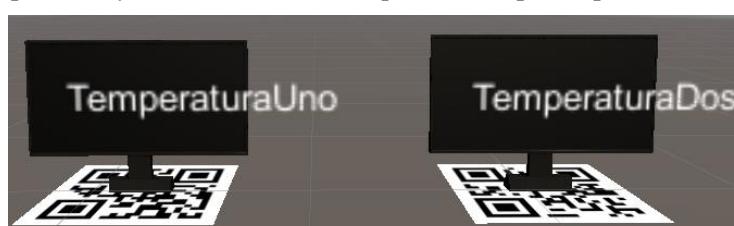


Figura 5.9. Objetos de la escena AR creada.

5.3. Implementación final del sistema

En esta última parte de la explicación práctica del proyecto, se expone la implementación del sistema de AR en el entorno de sensores creado para ello. Una vez realizado los pasos de los anteriores apartados, lo único que falta es hacer que el sistema sea funcional y, para ello, se va a seguir el esquema de funcionamiento mostrado en la Figura 5.10, propuesto para cada uno de los nodos de la red.

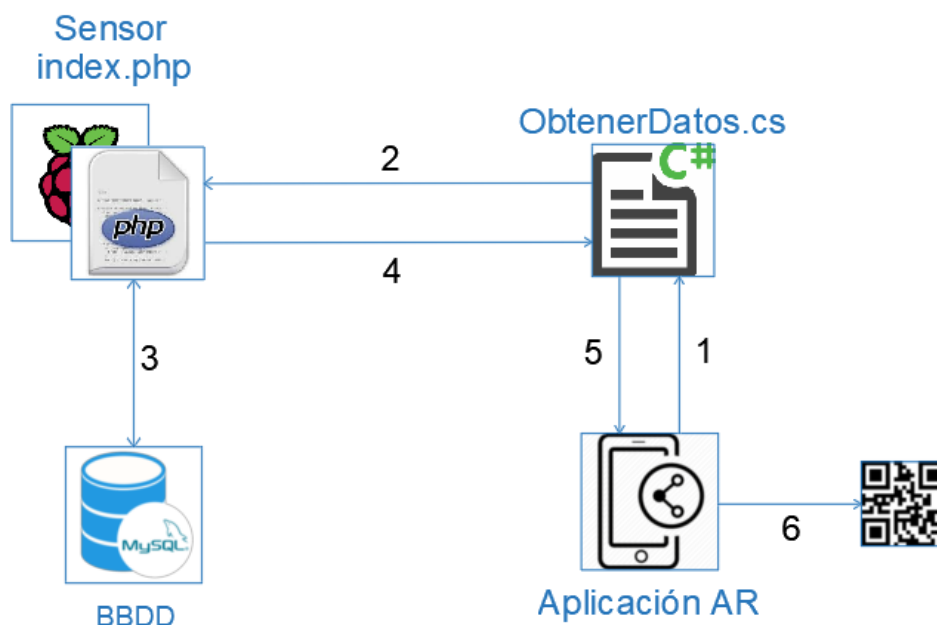


Figura 5.10. Esquema de la implementación final del sistema.

Como se puede observar en el esquema propuesto, los elementos involucrados son el dispositivo móvil, los sensores de la red (se ha reducido el número de los sensores a un único sensor para reducir la complejidad del esquema) con sus respectivos marcadores identificativos, la base de datos y los scripts necesarios para acceder a dichos datos. El proceso de representación de los datos es el siguiente:

1. Al abrir la aplicación, esta hace uso del código que se ha desarrollado junto a la interfaz para acceder a la página web intermediaria de cada uno de los nodos, es decir, habrá una página PHP asociada a cada sensor de la red, y así obtener todas las medidas de dichos nodos.
2. Se realiza una petición a cada sensor del sistema para poder acceder a su página PHP. Esta petición que llega a la Raspberry destino hace que esta de permiso a la aplicación para acceder a su index.php donde se encuentran las medidas.
3. Al permitir el acceso a la página PHP, la configuración de esta última hace que se realice una consulta a la base de datos para rellenar dicha web con una lista, la cual contiene los datos medidos desde que se empieza a medir la temperatura del entorno usando Sense Hat, mediante el script de Python ya explicado anteriormente.
4. Tras acceder a index.php, se obtiene una respuesta en forma de cadena de texto con todas las medidas y es el propio código de la aplicación AR el que extrae de dicha cadena el dato de la última medición realizada por los sensores.
5. La aplicación obtiene los datos extraídos en el punto anterior.
6. Cuando la aplicación reconoce un marcador existente en la base de datos de marcadores de Vuforia, es la misma aplicación la que diferencia a cuál de los nodos de la red pertenece dicho marcador y representa mediante AR la temperatura de ese nodo.

Para finalizar este capítulo, a continuación se exponen los scripts utilizados en el esquema de la implementación final del sistema de AR y unas pruebas realizadas para comprobar que el sistema es verdaderamente funcional.

5.3.1 Código PHP intermediario

Este código, expuesto en el [Anexo B](#), se usa, como ya se ha comentado, para pasar los datos de la base de datos de medidas a una página PHP, para que la aplicación pueda obtener cada una de las medidas realizadas por los nodos de la red. Para el desarrollo de este script se ha utilizado el software SQLite, integrado en la parte PHP del servidor LAMP, que resulta de gran ayuda para almacenar datos en páginas PHP mediante el uso de lenguaje SQL. Todos los métodos y pasos usados para el desarrollo de este script se pueden consultar en [7].

Como se puede apreciar en el script, lo primero que se hace es una conexión a la base de datos, asociando a dicha conexión la consulta que se va a realizar a la misma. Es decir, mediante las siguientes líneas de código se intenta acceder a la base de datos y obtener los datos de la tabla donde se almacenan las medidas:

```
$database = new mysqli("localhost","root","password","database") or
    die('Error:' . mysqli_error());
$consulta = "SELECT * FROM temperatura";
$datos = mysqli_query($database, $consulta) or die('Consulta
    fallida:' . mysqli_error());
```

Por partes, `new mysqli` define la BBDD a la que se quiere conectar, `SELECT`, ya comentado en la creación del servidor LAMP, se trata de la consulta en lenguaje SQL a dicha BBDD y `mysqli_query` es la función de SQLite que ejecuta la conexión y la consulta.

Tras esto, se cuentan las filas que han llegado en la respuesta a la consulta lanzada anteriormente, lo cual se hace mediante la función `mysqli_num_rows`.

Por último, como los datos obtenidos de la anterior consulta son las correspondientes filas de la tabla consultada, se hace uso de un bucle `for` con el que, en cada iteración, se irá posteando el la página PHP la fila correspondiente, utilizando la siguiente declaración “echo”:

```
echo " Temperatura: " . $fila["temp"] . " <br>;
```

Lo expuesto hasta este punto hace que se tenga una página PHP en la que se pueden consultar los datos recogidos por los sensores de la red por todo equipo o aplicación que acceda a esta. Con esto, la aplicación AR ya puede consultar los valores de temperatura medidos, lo cual se explicará en el siguiente apartado.

5.3.2 Código para obtener los valores medidos

En este apartado se pasa a abordar el código del [Anexo C](#), el cual sirve para que la aplicación acceda a la página PHP de cada uno de los nodos y obtenga la última medida de temperatura realizada por estos. Este script, desarrollado usando [8] y [9], tiene tres partes principales: declaración y creación de una lista con las URLs de las páginas PHP de los distintos nodos, obtención de los últimos valores medidos y modificación del objeto de texto de la AR.

Para la primera parte, se utiliza la clase `List<T>` de C# que crea una lista de objetos de tipo `T`, el cual está configurado como tipo `String`, que se accede mediante un índice para recorrer dicha lista en un bucle.

En la segunda parte, una vez definida la lista de URLs, se pasa a una corrutina en la que, mediante el bucle `for`, se accede a cada URL y se obtienen todo el contenido de las páginas PHP. Esto se realiza mediante el método `UnityWebRequest.Get()`. Además, antes de modificar el objeto de texto de la escena de AR, se extrae la última medición calculando para ello la longitud de la respuesta, la cual es una cadena de texto, y haciendo uso del método `Substring()`.

Por último, para modificar el contenido del objeto de AR, se utilizan dos métodos que son `Find()`, para poder encontrar el objeto que se quiere modificar, y `GetComponent<T>`, para poder obtener el componente de tipo `T`, el cual está definido como `TextMesh`, de dicho objeto.

5.3.3 Pruebas realizadas

En este apartado se procede a adjuntar algunas pruebas que se han realizado con el fin de ofrecer una vista funcional de la arquitectura y del sistema de realidad aumentada explicado en este apartado. Para dicho fin, se han implementado un total de dos nodos, cada uno conectado a una página PHP propia. El nodo 1 se ha configurado para que obtenga las medidas de la página “*index.php*”, mientras que el nodo 2 las obtiene de “*index2.php*”. Por otra parte, la última medida realizada por el nodo 1 es de 27.9 °C y la del nodo 2 es de 30.3 °C.

Tabla 5.1. Nodos del sistema.

Nodo	Obtención de los datos	Última medida realizada (°C)
1	index.php	27.9
2	index2.php	30.3

En la Figura 5.11 y 5.12 se muestra la implementación del sistema con la realidad aumentada de cada uno de los nodos. Por otra parte, en la Figura 5.13 se muestra la funcionalidad total del sistema, el cual es capaz de reconocer los marcadores de cada uno de los nodos y mostrar así que la aplicación puede diferenciar a cada nodo, buscando la última medida realizada por cada uno de ellos y representarla adecuadamente.

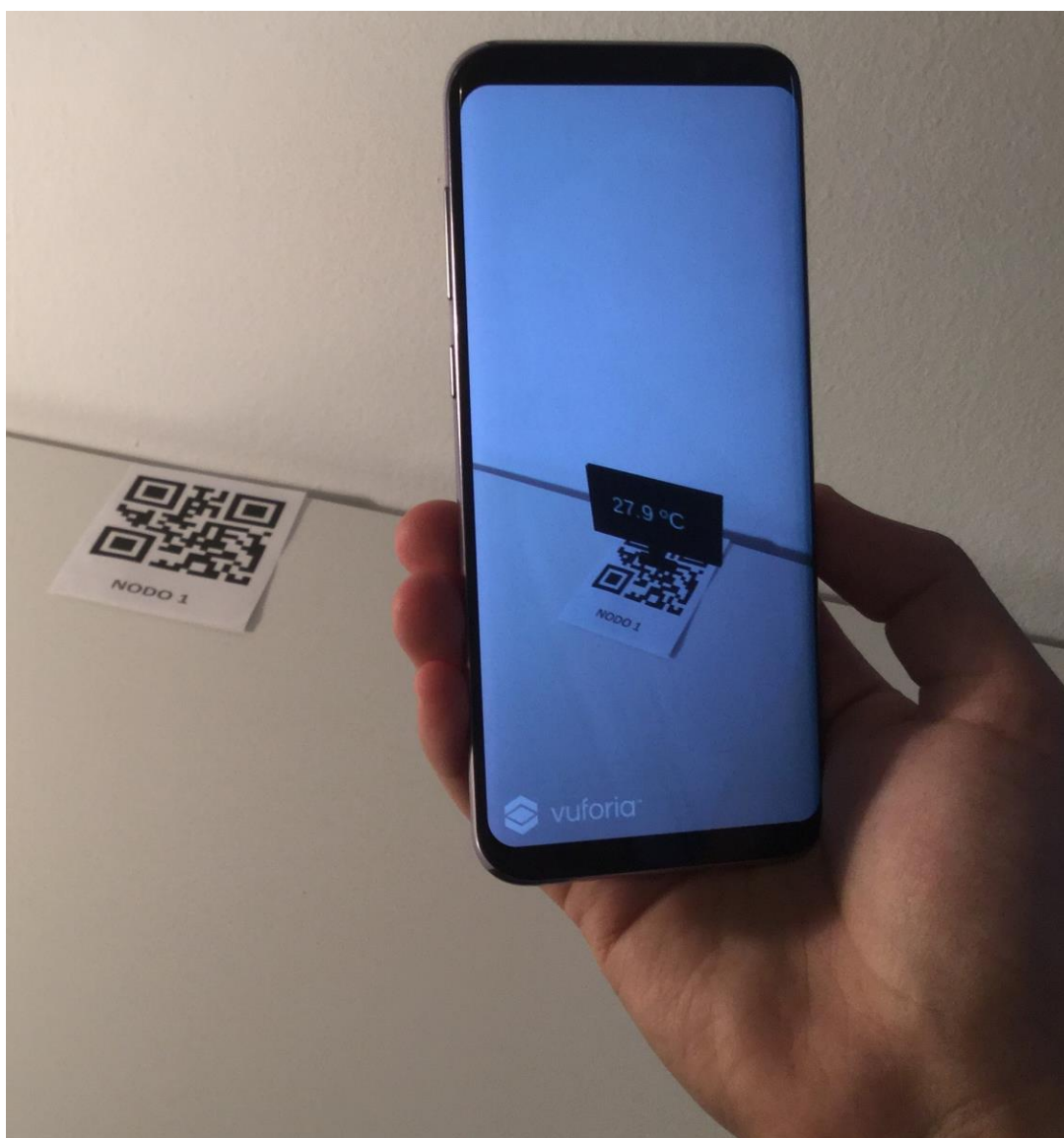


Figura 5.11. Nodo 1 del sistema en la aplicación de AR.

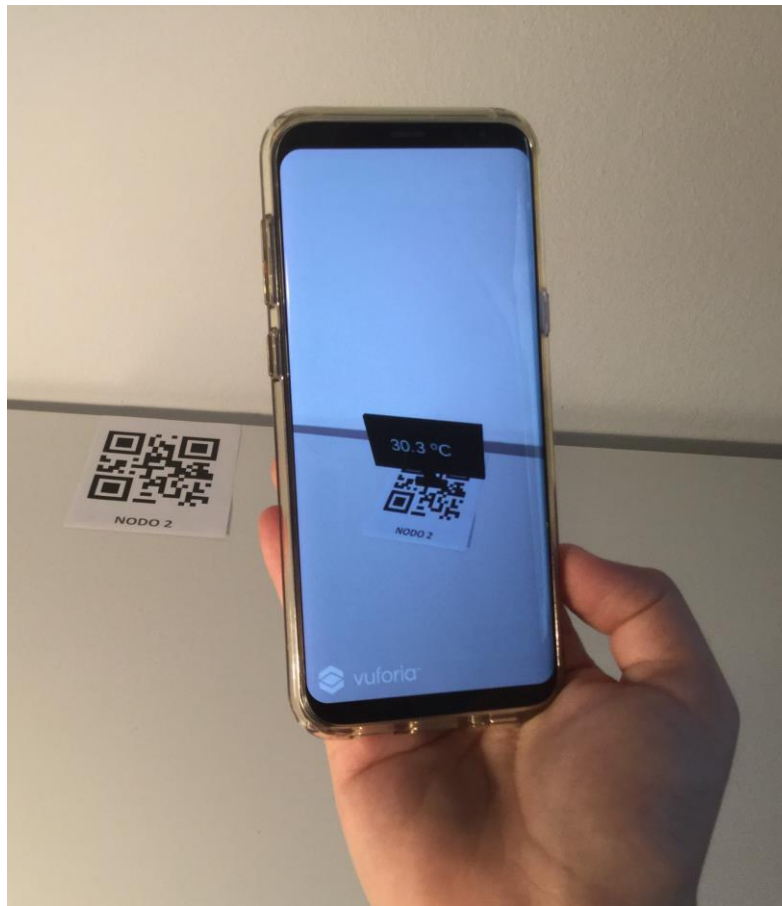


Figura 5.12. Nodo 2 del sistema en la aplicación de AR.



Figura 5.13. Diferenciación de los nodos por parte de la aplicación de AR.

6 CONCLUSIONES Y LÍNEAS FUTURAS

El valor de una idea radica en el uso de la misma.

Thomas A. Edison.

En un mundo en el que los dispositivos inteligentes están en auge, el principal problema de esta nueva tecnología, en la que todo está conectado, mandando y recibiendo información constantemente, es que el usuario no está familiarizado con esta realidad, dado que a veces estos dispositivos no son visibles. Es por esto por lo que, aunque la tecnología avance y los dispositivos sean más accesibles dado su bajo coste, se debe aportar al usuario una experiencia con la que pueda interactuar con los SOs y con la que poco a poco se vaya asentando esta nueva tecnología.

Para intentar solucionar este problema, en el presente trabajo se ha intentado buscar una forma de representar lo que hoy conocemos como SOs, aportando una interfaz de usuario de realidad aumentada con la que poder percibir el entorno inteligente en el que nos encontramos. Aunque no ha sido posible realizar un experimento para evaluar la experiencia de distintos usuarios con la visualización mediante la AR de la red de sensores y la interacción de los mismos con ésta, se ha comprobado que el trabajo llevado a cabo propone una arquitectura funcional y que la realidad aumentada puede llegar a solventar el problema de la percepción de los SOs.

Por otra parte, la localización de los sensores en la red, como ya se ha visto en el contenido teórico, se puede llevar a cabo mediante distintas tecnologías, siendo el GPS para sistemas de exteriores o el BLE para sistemas de interiores las dos con mejores características para llevar a cabo este tipo de sistema en el que se utiliza un smartphone. Sin embargo, el presente trabajo se ha tenido que realizar mediante la utilización de páginas PHP, a las que se accede mediante el reconocimiento de imágenes por parte de Vuforia SDK, lo cual se desvía de la arquitectura ARIoT expuesta, todo ocasionado por la falta de medios para llevar a cabo el proyecto debido a las condiciones ocasionadas por el COVID-19. Por lo tanto, si se decide realizar un sistema utilizando una de las dos tecnologías mencionadas, hay que tener en cuenta problemas en cuanto a la exactitud de posición de las mismas y la eficiencia energética que puede provocar al smartphone.

Para finalizar este capítulo, una de las posibles líneas futuras que puede llevarse a cabo es el diseño de una nueva interfaz para el usuario con más tipos de medidas y datos, utilizando la que se ha realizado en este trabajo como base, en la que se añada como funcionalidad el uso del CA de la capa intermedia de la arquitectura propuesta. Esto es, añadir inteligencia a la aplicación para que, dependiendo del usuario que la utilice, se muestre la información que éste consulta más en su día a día. Otra posible línea futura es la implementación de esta red de sensores en un entorno domótico, en el que el usuario sea capaz, mediante la realidad aumentada de la aplicación móvil desarrollada, de personalizar los parámetros del entorno inteligente, como por ejemplo ser capaz de establecer la temperatura a la que se tenga que adecuar el habitáculo en el que nos encontremos. Por último, también podría realizarse el estudio de cómo se vería afectada la actual arquitectura en el caso de que se añadiese el envío de datos mediante Bluetooth, ya que no ha podido llevarse a cabo y era la idea principal del trabajo.

ANEXO A. CÓDIGO PYTHON PARA INTRODUCIR MEDIDAS EN LA BASE DE DATOS

En este anexo se presenta el código utilizado por la Raspberry para introducir los datos medidos por el emulador de Sense Hat en la base de datos MySQL. Para ello se hace uso de una librería denominada EasyDB, la cual nos facilita el no tener que configurar ninguna conexión de red específica.

A.1. Código para insertar datos

Código A.1. insertar.py

```
#Importamos toda la librería EasyDB que se encargará de realizar un buen
#conexionado a la base de datos.
from easydb import *

#Importamos sense_emu para realizar la medida que se mandará a la
#base de datos.
from sense_emu import SenseHat

#Importamos la librería "time" para no realizar ninguna medida durante
#un periodo de tiempo impuesto por el usuario.
import time

#Definimos una variable tipo entero que se utiliza en el bucle para
#no parar de realizar mediciones.
i=2

#Bucle que no va a dejar de repetirse hasta que no se fuerce su
#detención
while (i<3):
    sensor = SenseHat()
    #Tomamos la medida de la Raspberry a través de SenseHat.
    medida = sensor.temperature

    #Redondeamos dicha medida a un decimal.
    tempe= round(medida, 1)

    db_data = EasyDB("data")

    #Sentencia SQL para insertar valores en la tabla
    db_data.query = 'INSERT INTO temperatura(nodeid,temp) VALUES (1,'+ str(tempe) +')'

    result = db_data.run()

    #Tiempo establecido para parar antes de realizar otra nueva medida
    time.sleep(60)
```


ANEXO B. CÓDIGO PHP PARA PARA PUBLICAR EN UNA PÁGINA WEB .PHP LOS DATOS DE LA BASE DE DATOS

Código B.1. index.php

```
<php

//Definimos la base de datos a la que nos queremos conectar
//Para ello, pasamos como parámetros la dirección de la BD, el usuario, la contraseña
//y el nombre de la BD
$dbase = new mysqli("localhost","root","password","database") or die('Error:' .
mysqli_error());

//Preparamos la sentencia para seleccionar la tabla de medidas
$consulta = "SELECT * FROM temperatura";

//Realizamos la consulta y obtenemos los datos
$datos = mysqli_query($dbase, $consulta) or die('Consulta fallida:' .
mysqli_error());

//Contamos el número de filas de la tabla
$numfilas = mysqli_num_rows($datos);

//Usamos un bucle for para publicar cada una de las medidas realizadas
for($i = 0; $i < $numfilas; $i++)
{
    $fila = mysqli_fetch_array($datos);
    echo " Temperatura: " . $fila["temp"] . " <br>";
}

?>
```


ANEXO C. CÓDIGO C# PARA OBTENER LOS DATOS DE LA PÁGINA PHP

El siguiente código es el utilizado por parte de la aplicación de Android para obtener la última medida realizada por el nodo. La aplicación accede a la página PHP, en la que la Raspberry publica las medidas, y recibe todas las medidas realizadas desde la puesta en marcha del nodo.

Código C.1. ObtenerDatos.cs

```
using JetBrains.Annotations;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class ObtenerDatos : MonoBehaviour
{
    void Start()
    {
        //Urls asociadas a las páginas PHP de las Raspberry (nodos de la red)
        string url1 = "http://192.168.1.173/index.php";
        string url2 = "http://192.168.1.173/index2.php";

        //Lista de urls que se pasa a la corutina para obtener las medidas
        List<String> enlaces = new List<string>();
        enlaces.Add(url1);
        enlaces.Add(url2);

        //Empezamos la corutina para enviar una petición de
        //descarga de datos a la página PHP de la Raspberry
        StartCoroutine(ObtenerTemperaturas(enlaces));
    }

    IEnumerator ObtenerTemperaturas(List<String> lista)
    {
        //Utilizamos un bucle for para recorrer la lista de urls de los distintos nodos
        for (var i=0; i<lista.Count; i++)
        {
            //Accedemos a la web para obtener los datos
            UnityWebRequest www = UnityWebRequest.Get(lista[i]);
            //Esperamos a la respuestas
            yield return www.SendWebRequest();
            //Comprobamos si hay error en la conexión
            if (www.isNetworkError || www.isHttpError)
            {
                Debug.Log(www.error);
            }
        }
    }
}
```

```
else
{
    //Mostramos los resultados
    Debug.Log(www.downloadHandler.text);

    //Obtenemos la longitud de la respuesta de la página web
    int tam = www.downloadHandler.text.Length;

    //Cortamos esa respuesta para obtener el último valor medido
    string temperatura = www.downloadHandler.text.Substring(tam-9,5);

    //Modificamos el texto del objeto 3D para que aparezca la medida
    GameObject.Find("Temperatura"+(i+1)+"").GetComponent<TextMesh>().text
    = temperatura+"°C";
}
}
}
```

REFERENCIAS

- [1] K. Michalakakis, J. Aliprantis and G. Caridakis, "Visualizing the Internet of Things: Naturalizing Human-Computer Interaction by Incorporating AR Features," in IEEE Consumer Electronics Magazine, vol. 7, no. 3, pp. 64-72, May 2018, doi: 10.1109/MCE.2018.2797638.
- [2] Gareth Halfacree, "The Official Raspberry Pi Beginner's Guide: How to use your new computer", 2019, ISBN: 978-1-912047-62-8.
- [3] Habbak, Hassan EL, and Dominic Cushnan. Developing AR Games for iOS and Android, Packt Publishing, Limited, 2013. ProQuest Ebook Central, <http://ebookcentral.proquest.com/lib/uses/detail.action?docID= 1362586>
- [4] <https://learn.unity.com/tutorial/como-usar-la-interfaz-de-unity>
- [5] Membrey, P., & Hows, D. (2013). Learn Raspberry Pi with Linux (1st ed. 2013.). Apress. <https://doi.org/10.1007/978-1-4302-4822-4>
- [6] Godoc, E., & Bisson, A. (n.d.). SQL : Los fundamentos del lenguaje (con ejercicios corregidos) (2a edición). Ediciones ENI.
- [7] Valade, Janet, and Steve Suehring. PHP, MySQL, JavaScript and HTML5 All-In-One for Dummies, John Wiley & Sons, Incorporated, 2013. ProQuest Ebook Central, <https://ebookcentral.proquest.com/lib/uses/detail.action?docID=827043>.
- [8] <https://docs.unity3d.com/es/2019.1/Manual/UnityManual.html>
- [9] <https://docs.unity3d.com/ScriptReference/GameObject.Find.html>
- [10] Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787–2805. <https://doi.org/10.1016/j.comnet.2010.05.010>
- [11] Mehta, R., Sahni, J., & Khanna, K. (2018). Internet of Things: Vision, Applications and Challenges. Procedia Computer Science, 132, 1263–1269. <https://doi.org/10.1016/j.procs.2018.05.042>
- [12] Abhishek Singh, Ashish Payal, and Sourabh Bharti. A walkthrough of the emerging IoT paradigm: Visualizing inside functionalities, key features and open issues. Elsevier, 2019.
- [13] Saint-Exupéry, Antoine de. "Internet of Things Strategic Research Roadmap Antoine De Saint-exupery." (2009).
- [14] Sethi, P., & Sarangi, S. (2017). Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering, 2017(2017), 1–25. <https://doi.org/10.1155/2017/9324035>
- [15] Ejaz, W., & Anpalagan, A. (2018). Internet of Things for Smart Cities: Technologies, Big Data and Security. Springer International Publishing. <https://doi.org/10.1007/978-3-319-95037-2>

- [16] María José Madero Ayora, Diapositivas de la asignatura "Sistemas Emergentes en Comunicaciones", Curso 2019-2020
- [17] R. Sharma, N. Pandey and S. K. Khatri, "Analysis of IoT security at network layer," 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, 2017, pp. 585-590, doi: 10.1109/ICRITO.2017.8342495.
- [18] Yassein, M., Shatnawi, M., & Al-Zoubi, D. (2016). Application layer protocols for the Internet of Things: A survey. 2016 International Conference on Engineering & MIS (ICEMIS), 1–4. <https://doi.org/10.1109/ICEMIS.2016.7745303>
- [19] Makkad Asim. (2017). A Survey on Application Layer Protocols for Internet of Things (IoT). International Journal of Advanced Research in Computer Science, 8(3). <http://search.proquest.com/docview/1901457597/>
- [20] Ding, J., Nemati, M., Ranaweera, C., & Choi, J. (2020). IoT Connectivity Technologies and applications: A Survey. IEEE Access, 8, 1–1. <https://doi.org/10.1109/ACCESS.2020.2985932>
- [21] Queralta, J., Gia, T., Zou, Z., Tenhunen, H., & Westerlund, T. (2019). Comparative Study of LPWAN Technologies on Unlicensed Bands for M2M Communication in the IoT: beyond LoRa and LoRaWAN. Procedia Computer Science, 155, 343–350. <https://doi.org/10.1016/j.procs.2019.08.049>
- [22] Qadir, Q., Rashid, T., Al-Salihi, N., Ismael, B., Kist, A., & Zhang, Z. (2018). Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs. IEEE Access, 6, 77454–77473. <https://doi.org/10.1109/ACCESS.2018.2883151>
- [23] Minoli, D. (2013). Building the internet of things with IPv6 and MIPv6: the evolving world of M2m communications . Wiley. <https://doi.org/10.1002/9781118647059>
- [24] Chandra, P. (2008). Wireless networking. Elsevier/Newnes.
- [25] Mullen, T. (2011). Prototyping augmented reality. Wiley.
- [26] Craig, A. (2013). Understanding augmented reality concepts and applications . Morgan Kaufmann.
- [27] Kipper, G., & Rampolla, J. (2012). Augmented reality an emerging technologies guide to AR (1st ed.). Syngress.
- [28] Pielli, C., Zucchetto, D., Zanella, A., Vangelista, L., & Zorzi, M. (2015). Platforms and Protocols for the Internet of Things. EAI Endorsed Transactions on Internet of Things, 1(1), 150599–10–2015.150599. <https://doi.org/10.4108/eai.26-10-2015.150599>
- [29] Ngu, A., Gutierrez, M., Metsis, V., Nepal, S., & Sheng, Q. (2017). IoT Middleware: A Survey on Issues and Enabling Technologies. IEEE Internet of Things Journal, 4(1), 1–20. <https://doi.org/10.1109/JIOT.2016.2615180>
- [30] Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. IEEE Communications Surveys & Tutorials, 16(1), 414–454. <https://doi.org/10.1109/SURV.2013.042313.00197>
- [31] Wang, Lee. (2017). A Survey of the Internet of Things in Smart City Application in Taipei City. 10.13140/RG.2.2.31635.58401.

