Módulo 1: Fundamentos IoT

ÍNDICE

1.- ¿Qué es IoT?

1.1 Introducción: conectando objetos cotidianos

Internet de las cosas (o *Internet Of Things, IoT*) es una expresión para definir la conexión, a través de Internet, de dispositivos o *cosas*, que son objetos cotidianos, más allá de los dispositivos informáticos que ya contaban con conexión a Internet (ordenadores, smartphones, etc). En muchos casos son de bajo coste, bajo consumo y con una capacidad de proceso generalmente baja. El concepto fue introducido por Kevin Ashton (1999), investigador en campos como RFID, sensores y otros, si bien comenzó a extenderse unos diez años después.

IoT trata de objetos que podemos llevar con nosotros, o están presentes en casa, en el trabajo o en la calle, con conexión a Internet. Las aplicaciones son múltiples, más allá de la clásica automatización doméstica (domótica, hogar digital…) o industrial: desde conectar cualquier dispositivo o electrodoméstico en el hogar hasta la comunicación entre vehículos, pasando por dispositivos que el usuario lleva consigo a diario (*wearables*) hasta muchos otros.

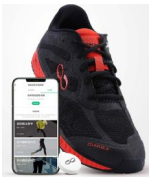
Los dispositivos conectados (cosas, *things*) proporcionan información del entorno a través de sensores o actúan sobre el entorno a través de actuadores. Un sistema que utilice IoT se diferencia de una red informática clásica en:

- requisitos de conexión: puertas de enlace entre redes diferentes, etc
- software diverso en los dispositivos, que cuentan o no con sistema operativo
- hardware diverso

Los problemas fundamentales que surgen son: la **seguridad** y privacidad, y el **crecimiento** exponencial de dispositivos conectados a Internet, con el riesgo de colapsar la red.

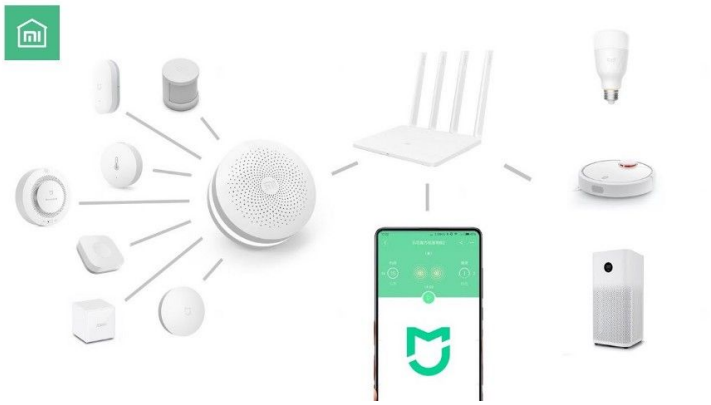
Algunos ejemplos de aplicaciones IoT comerciales:

- **Wearables**: dispositivos como las pulseras de actividad y otros relojes inteligentes pueden monitorizar el pulso, actividad física (movimiento), enviando la información a una app para determinar la regularidad, calidad y horas de sueño, pulso, actividad física diaria y estados de reposo, etc. Algunos ejemplos son las pulseras de actividad y otros relojes inteligentes, que almacenan pasos, distancia, recorrido seguido (GPS), miden pulso… Ejemplos conocidos son Apple Watch o Fitbit, marcas deportivas como Garmin, Polar, y genéricas Huawei o Xioami (Mi Band, Amazfit). El calzado deportivo inteligente almacena pasos, distancia, etc, otros pueden ser gafas de realidad virtual o realidad aumentada, y tejidos con diversos sensores que pueden cambiar de color o realizar otras funciones ante cambios en la sudoración, temperatura, etc.

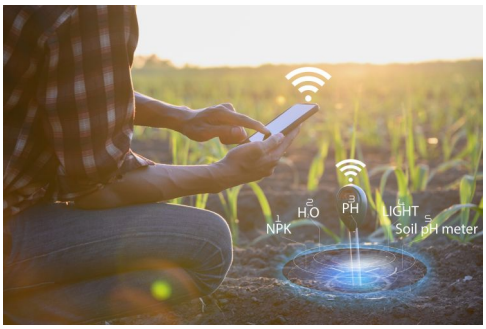


- Dispositivos relacionados con **salud, deporte y bienestar**: supervisan y monitorizan actividad física sin ser *wearables*, como básculas que monitorizan peso, índice de masa corporal y otros, mediante el uso de diversos sensores; desde la app se monitoriza tanto los valores actuales como el histórico, tendencias, etc. Las marcas de *wearables* suelen fabricar estos dispositivos para complementar sus sistemas.

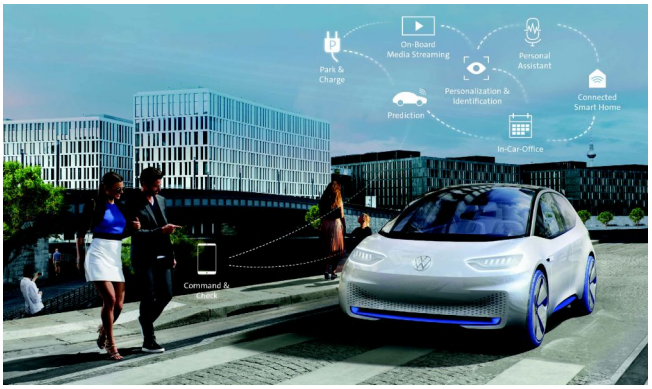
- **Domótica, seguridad** y **hogar digital**: aportan bajo coste y funciones avanzadas más allá de las tradicionales soluciones domóticas o de seguridad (alarmas). Algunos dispositivos o *cosas* son: sensores de temperatura y humedad, vibración, de apertura de puertas y ventanas, y actuadores como humidificadores y deshumidificadores, lámparas (bombillas) regulables tanto en intensidad como en temperatura de color, electroválvulas, sirenas y otros avisadores acústicos y enchufes o relés a los que sea posible conectar cualquier equipo. Los robots de limpieza (iRobot, LG, Conga…) permiten conocer desde la app el progreso de limpieza, visualizando la ubicación del dispositivo sobre el plano de la vivienda. Otros sensores (nutrientes, humedad en suelo, etc.) permiten aplicaciones en jardinería. Los electrodomésticos (lavadoras, frigoríficos, etc.) incorporan cada vez más a menudo conexión a la nube. Los sistemas de seguridad y videovigilancia, como cámaras, detectores de movimiento, de gas, inundación etc también se consideran *cosas* de un sistema IoT cuando cuentan con conexión a la nube, directa o indirecta. Grandes fabricantes de automatización como Siemens, Schneider Electric, etc., así como de dispositivos de bajo coste como Xiaomi (su plataforma *Mi Home* está muy extendida) cuentan con sistemas IoT comerciales, como evolución de los sistemas domóticos. La ventaja sobre los sistemas tradicionales, además del bajo coste, es la ausencia en muchos casos de un sistema central que tome decisiones, dejando esta tarea a la plataforma en la nube.



- Aplicaciones **agrícolas** y ganaderas: el bajo coste y facilidad de interconexión de los sistemas IoT permite automatizar riegos y otros procesos agrarios que antes solo eran posibles con caros autómatas (PLC), y la toma de decisiones puede ser realmente compleja al poder almacenar gran cantidad de información de diversos sensores en la plataforma en la nube.



- **Vehículos**: además de la monitorización y seguimiento de vehículos, ya sea de flotas en grandes empresas o de particulares, los vehículos que están apareciendo no solo permiten el acceso remoto a multitud de funciones (carga en el caso de eléctricos, estado de alarmas, etc.) sino la propia interconexión entre ellos (M2M o *Machine to Machine*): es ya una realidad la proliferación de vehículos conectados, que evitan accidentes sin intervención humana, con técnicas como detección de cambio involuntario de carril, prevención de colisión (controles de velocidad adaptativos), aparcamiento autónomo, detección de fatiga y otros. No solo automóviles, sino motocicletas, bicicletas y patinetes eléctricos cuentan con conexión para monitorizar los trayectos (vehículos compartidos), su estado de carga, alarmas, etc.



- Aplicaciones **industriales** e inmóticas: términos como Industria 4.0 o ciudades inteligentes están relacionados con la extensión de la interconexión y análisis de datos a cualquier dispositivo en la industria (por pequeño y sencillo que sea), sector terciario (hostelería, oficina...) y por extensión a toda una población (telegestión de contadores de agua y luz, medidas de la calidad del aire y agua, etc). La gestión remota de tráfico (semáforos, etc.), mapas de ruido, iluminación, etc., permite no solo la automatización sino la supervisión y el mantenimiento predictivo de cualquier sistema.

1.2 Componentes en IoT

El objetivo es conectar *cosas* para obtener y analizar información o actuar sobre el entorno, para ello será necesario distinguir entre:

- **dispositivo** (thing, o *cosa*): incluye el hardware y software que interactúa con el entorno de manera directa, y puede conectarse con otros dispositivos o con la nube de forma directa o indirecta. Son sensores y actuadores, como sensores de temperatura, movimiento, o pulso cardíaco, o actuadores como lámparas, motores, etc:

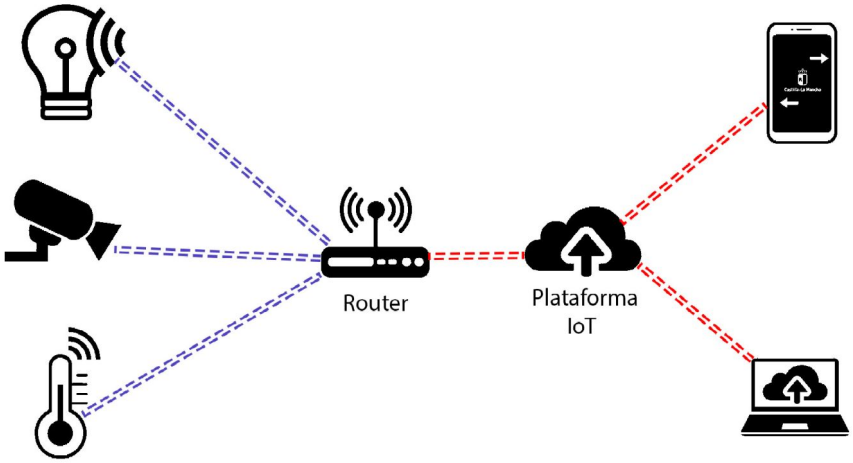
- **puerta de enlace** (*gateway*): cuando los dispositivos no se conectan de forma directa a la red necesitan de otro dispositivo que se lo permita; este dispositivo puede encargarse tanto de gestionar la propia red de *cosas* como de conectar esa red a Internet: por tanto une redes de *cosas* con redes informáticas. Puede también encargarse de procesar la información antes de enviarla a la nube

- **plataforma en la nube** (*cloud platform*): es el servicio donde se almacenan y procesan los datos recogidos por los dispositivos, y permiten al usuario acceder a estos dispositivos, a través de una web, de una app en su ordenador, smartphone o cualquier dispositivo conectado:

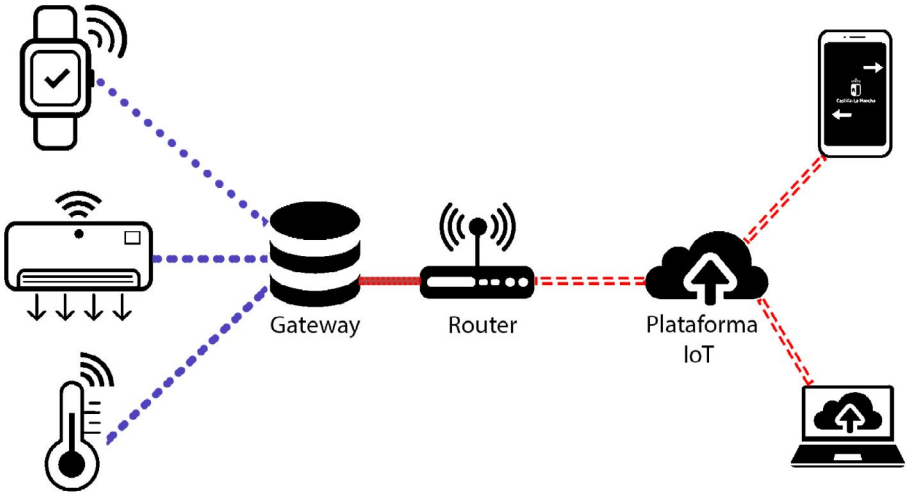
1.3 Conexión a Internet

Las redes entre máquinas más allá de los ordenadores ya estaban desarrolladas hace unos años: desde hace décadas hay redes RS-485, Profibus y otras en la industria que interconectan dispositivos. Sistemas como Zigbee, Bluetooth y otros permiten la conexión entre sensores y controladores en redes de área personal (PAN, frente a LAN): muchos sensores de estaciones meteorológicas domésticas y otros anteriores a IoT ya usaban estos sistemas. Pero para que consideremos a unos dispositivos como *cosas* de un sistema IoT es necesaria la conexión no solo entre ellos (red de área personal o local) sino a Internet. La conexión de las *cosas* a *Internet* puede ser:

- **directa**: los dispositivos pueden conectarse directamente y usan protocolos TCP/IP. En realidad la conexión siempre se realiza a través de alguien (por ejemplo una conexión wifi necesita un router que conecte a su vez a Internet a través de una conexión WAN), pero se considera este caso cuando el dispositivo es capaz de comunicarse vía TCP/IP:



- **indirecta**: necesitan puerta de enlace (*gateway*) al usar otras redes (Bluetooth, Zigbee) que no tienen conectividad directa a Internet (TCP/IP):



La conexión a Internet suele ser inalámbrica (aunque puede ser también cableada) a través de:

- **Wifi**: el dispositivo incorpora adaptador inalámbrico que permite la conexión a la red
- **Datos** 3G/4G/5G: incorpora un adaptador GSM, es necesaria una tarjeta SIM
- Redes de **baja potencia**: es la gran novedad en IoT: redes como *LoRa* o *SigFox* permiten conectar dispositivos con muy bajo consumo (alimentados con baterías) a redes de baja velocidad y área extensa en entornos donde no existe una red wifi o el acceso a datos es complicado. En algunos casos no precisan de una puerta de enlace (no es similar a una red Zibgee u otras) pues el proveedor de la red actúa como pasarela entre redes. En otros casos sí es necesaria, pero puede estar a varios kilómetros del dipositivo, lo que simplifica la conexión gateway-Internet.

2.- Arquitectura de sistemas IoT

2.1 Requisitos hardware

Los dispositivos o *cosas* emplean un hardware que debe estar lo más adaptado a la tarea que se pretende realizar. Con la proliferación de la cultura *Maker* y del *Do It Yourself* (DIT, o hazlo tú mismo) en el mundo de la electrónica, placas de desarrollo como Arduino se han extendido tanto en el ámbito de la enseñanza como en otros. Debido a su uso en entornos educativos hay multitud de información disponible, tanto de montajes como de aplicaciones.

Son muchas las variantes de placas **Arduino** que cuentan con conexión a Internet vía WiFi o de otro tipo, de manera nativa o con alguna placa adicional. Además de derivados de Arduino, son empleadas en proyectos IoT placas como Beaglebone, **Raspberry Pi** y otras. Placas de muy bajo coste, como **NodeMCU** y derivadas de la ESP32, también cuentan con conexión WiFi de manera nativa. Para elegir la placa adecuada al proyecto IoT hay que tener en cuenta:

- **coste**: en IoT es más importante que en otros sistemas interconectados: un sensor de temperatura, de apertura de puerta, etc., no puede costar lo mismo que un ordenador
- entradas y salidas (**E/S**): sensores y actuadores pueden conectarse a entradas o salidas digitales, analógicas, salidas PWM, o conexiones tipo I2C, SPI, etc. El tipo de E/S depende de sensores y actuadores, tanto su número como el tipo (entradas analógicas o digitales…). En entradas analógicas se puede requerir más o menos precisión. En las salidas es importante saber de antemano por ejemplo la intensidad requerida para mover un actuador (motor, etc.) y otras cuestiones
- **coste energético**: el consumo es crítico en aplicaciones IoT pues es habitual que se trate de dispositivos con baterías que trabajan de forma desatendida durante mucho tiempo. El bajo consumo es uno de los requisitos de cualquier dispositivo IoT.

- **conectividad**: no es lo mismo que el dispositivo tenga una conexión directa a Internet, trabajando con protocolos TCP/IP (a través de un router, o con conexión directa a redes de datos móviles), que se trate de otro tipo de red (ZigBee, etc) lo que obligaría a usar una puerta de enlace. Las prestaciones, fiabilidad y estabilidad del sistema dependen de esta conectividad

- **tamaño**: en IoT es un tema fundamental, pues en dispositivos *wearables* y otras aplicaciones es necesario que todo el sistema sea lo más compacto posible. Placas como *NodeMCU*, *Wemos D1*, *Arduino Nano 33IoT* o *Rapsberry Pi Zero W* son ideales por sus reducidas dimensiones. Es conveniente que la conexión con el exterior (si no se emplea un *Gateway*) sea directa, vía wifi o datos, sin empleo de placas externas, pues aumentaría el tamaño y complejidad del sistema y por supuesto reduciría la fiabilidad del mismo al aumentarse las interconexiones

- **sistema operativo**: a veces trabajar sobre un sistema operativo puede ser una ventaja, si se van a instalar diversas apps para controlar nuestro sistema: sería el caso de *Raspberry Pi* y otras. Si por el contrario se pretende la máxima simplicidad y control directo sobre las entradas y salidas es preferible trabajar sin sistema operativo, como es el caso de *Arduino*, *NodeMCU*, etc, donde el entorno de desarrollo (IDE) compila la aplicación para su uso en la placa sin la intermediación de un sistema operativo

- **información disponible**: según la disponibilidad de librerías, aplicaciones, ejemplos… será más conveniente trabajar con una u otra placa

- **otras** características: consumo, tensión de alimentación (5V, 3,3V…), entorno/s de desarrollo (IDE) disponible/s, etc.

2.2 Interconexión entre placas y sensores/actuadores

Los diversos sensores, actuadores y otros periféricos de nuestro sistema IoT (como relojes a tiempo real, memorias, etc.) pueden conectarse a la placa mediante:

- E/S **digitales**: se suelen denominar GPI/O (*General Purpose Input/Output*), con dos niveles (alto y bajo, 1 y 0) con niveles de tensión predefinidos (5V, 3,3V, etc.). A veces es necesario aislar estos pines de las entradas y salidas, por seguridad. Podemos encontrar optoacopladores en las entradas, relés en las salidas, o bien la combinación optoacoplador más transistor de potencia (conocida como *relé de estado sólido*)...

- E/S **analógicas**: pueden tener más o menos resolución (en bits) lo que indica lo preciso que es su convertidor analógico/digital o digital/analógico. En el caso de sensores convencionales (temperatura, etc.) es normal trabajar con 8, 10 o 12 bits. Una placa suele incluir varias entradas analógicas, si bien no es tan común contar con salidas analógicas

- Salidas **PWM** (*Pulse-Width Modulation*, modulación por ancho de pulsos): se genera una onda rectangular de frecuencia relativamente alta, en la que el ancho de pulso es proporcional a la tensión analógica que se desea obtener. Algunos fabricantes las denominan 'salidas analógicas' pero en realidad necesitarían un filtrado previo para considerarse como tales. Son empleadas para regular la velocidad de motores de corriente continua, controlar brillo de LEDs, etc.

- **I2C** (*Inter-Integrated Circuit*) es un bus serie síncrono (con reloj enviado por un hilo adicional), en el que hay un dispositivo maestro y varios esclavos que interactúan. Se emplea en el control de pantallas LCD, relojes a tiempo real, memorias y otros. Hay otras variantes como SMBus, y otros buses como SPI, etc.

- puerto **serie** (UART): permite la comunicación entre dispositivos de forma asíncrona (sin transmisión del reloj por un hilo adicional). Su uso es habitual en la conexión a placas Bluetooth, WiFi, y otros transceptores inalámbricos (por ejemplo en bandas ISM, como 433 o 898 MHz, como es *LoRa*, tecnología diseñada para cubrir grandes distancias con baja potencia). Con las modificaciones necesarias un puerto serie se puede convertir en USB, RS-232, RS-485, etc.

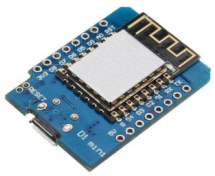
2.3 Algunas plataformas hardware para IoT

Entre la selección, descartamos la clásica *Arduino UNO R3* por no contar con conexión wifi sin la ayuda de una placa externa. Mediante una placa *ESP01* (basada en ESP8266) se podría expandir, pero el coste de ambas placas sería superior a casi todas las placas propuestas, además de que la interconexión de ambas y los requisitos de alimentación (5V Arduino, 3,3V ESP01) reducen la fiabilidad y aumentan tamaño y consumo.

NodeMCU está basada en el chip ESP8266, el mismo que en la placa ESP01 y otras. Incorpora conexión USB, al estilo de Arduino, a diferencia de la ESP01, que requiere adaptador especial. En vez de usar ESP01 como una ampliación de Arduino UNO, es posible usar directamente NodeMCU sin más. Al igual que Arduino, se trata de un proyecto de código abierto, tanto en software como en hardware, por lo que hay mucha información disponible. Las ventajas de NodeMCU son su bajo coste (desde unos 3€), conectividad WiFi y posibilidad de desarrollo bajo el entorno de desarrollo (IDE) de Arduino y otros, con lo que es fácil adaptar un proyecto desde Arduino, aunque NodeMCU trabaja a 3,3V en vez de a 5V como hace Arduino. Hay distintas versiones, todas con WiFi de 2.4 GHz (802.11 b/g/n), una entrada analógica (desventaja frente a Arduino UNO R3, que cuenta con 6) de 10 bits, y 13 pines digitales (D0 a D12) de entrada/salida:

NodeMCU

Wemos D1 mini es una placa basada también en ESP8266, de prestaciones similares a NodeMCU, aunque algo más compacto, y cuenta con distintas variantes:



Wemos D1 mini

Arduino Nano 33 IoT es una placa muy completa, pero de mayor coste; disponible desde finales de 2019, cuenta con WiFi y un sistema de comunicaciones seguras, Bluetooth, una IMU de 6 ejes, y un consumo muy reducido. Además, permite utilizar *Arduino IoT Cloud*, plataforma con un número por ahora limitado de placas, al menos en su versión gratuita. Esta plataforma cuenta con un asistente que permite poner en marcha un sistema IoT en pocos minutos:

Arduino Nano 33 IoT

Lilygo TTGO es un sistema de placas basadas en ESP32 con múltiples variantes: conexión LoRa de gran alcance, receptor GPS, datos GPRS (usa tarjeta SIM)... Casi todas sus versiones cuentan con WiFi y Bluetooth. Muchas cuentan con pantallas OLED y cargadores integrados de baterías LiPo, con lo que permiten construir un sistema IoT con muy pocos accesorios externos:

Lilygo TTGO T-Beam ESP32

M5Stack es un sistema de módulos interconectables (lo denominan *apilables*). Tanto las placas como los sensores y actuadores cuentan con una carcasa, a diferencia del resto de placas. Esto facilita la construcción de un sistema real, al no tener que incluir algún recinto o carcasa para el proyecto. Hay múltiples variantes, con distintos sensores y actuadores, así como versiones con cámara especializadas en reconocimiento de imágenes y otras aplicaciones. Un modelo destacado es **M5StickC**, pequeño módulo basado en ESP32, pantalla OLED a color, batería LiPo integrada, IMU, buzzer, transmisor IR, WiFi, Bluetooth, etc:

M5StickC, Atom y StickV, todas de M5Stack

3.- Plataformas en la nube

En IoT el esquema clásico para el almacenamiento de la información es el de una plataforma en la nube (o *nube* sin más), que centraliza toda la información. Actúa de intermediario entre la aplicación del usuario (una app en un smartphone o PC, una web…) y los dispositivos (o puerta de enlace, si es necesaria). Realiza múltiples funciones además del almacenamiento de datos: proporciona análisis, estadísticas y realiza multitud de procesos. Es habitual que realice también el registro, autenticación y autorización de los dispositivos (si tienen conexión directa) o de la puerta de enlace (si no la tienen). La conexión entre las *cosas* (dispositivos) y esta *nube* se realiza mediante diversas API (interfaces de programación de aplicaciones).

Algunas plataformas comerciales, de pago, son: **Google Cloud** IoT, **AWS** IoT (Amazon Web Services), **IBM** Watson IoT. Otras, gratuitas, para uso con placas de desarrollo, son: **ThinkSpeak** (de MathWorks, creadores de MathLab), **Adafruit IO**, **Arduino IoT Cloud**, o **Thinger.io**. Hay apps que permiten una gestión de dispositvos *IoT* de manera sencilla, como **Linear MQTT Dashboard**. Una plataforma usada en domótica es **Domoticz** (suele basarse en Raspberry Pi). La plataforma *IFTTT Maker* (*If This Then That*) permite integrar servicios de diversas plataformas. Otras como *Alexa* o *Siri* pueden interactuar con cualquier sistema IoT. Hay una relación entre estas plataformas en la nube, el *big data* (gestión de datos masivos) y el aprendizaje automático. Lo interesante es que la capacidad de un sistema IoT para *aprender* no depende de la capacidad de proceso de sus dispositivos, sino de la propia nube, con lo que se abren infinitas posibilidades.

4.- Protocolo MQTT

MQTT (*Message Queue Telemetry Transport*) es uno de los protocolos usados en IoT. Aunque no es el único, su conocimiento básico ayuda a entender el intercambio de información entre un dispositivo y la plataforma en la nube. Se trata de un estándar OASIS/ISO para el envío de telemetría (datos recabados por los dispositivos) en tiempo real y recepción de forma inmediata mensajes enviados desde la nube a un dispositivo para actualizar su estado. Es sencillo y *ligero* al estar orientado a la comunicación entre máquinas (M2M). Es usado por Google Cloud, Amazon Web Services (AWS), Adafruit IO, IBM Cloud e incluso por Facebook Messenger. En este momento su última versión es la 5.0, de marzo de 2019.

Normalmente funciona bajo TCP/IP, aunque puede funcionar bajo cualquier otro sistema, siempre que haya comunicación bidireccional. Usa por defecto el puerto 1883 para comunicaciones sin cifrar y el 8883 para cifradas.

Utiliza la técnica de *publicación/suscripción*, permitiendo comunicación bidireccional y acuse de recibo de los mensajes consumiendo muy poco ancho de banda.

MQTT no encripta los datos de manera nativa, pero incorpora cierto nivel de *seguridad* al permitir cifrado SSL/TLS, aunque esto requiere una mayor potencia de cálculo en el procesador y aumenta el ancho de banda en la red. También incorpora gestión de la *Calidad del Servicio* (QoS) en tres posibles grados: sin acuse de recibo (QoS0), con lo que el mensaje podría no sea entregado, con reenvío hasta tener acuse de recibo, pero puede llegar duplicado (QoS1) o garantizar que solo llegará una copia del mensaje al destinatario (QoS2).

Usa una topología (forma de conexión, a nivel lógico) en *estrella*, donde cada *cosa* o dispositivo o *cliente* se conecta en un extremo, y en el centro de esta estrella estaría el servidor o *broker*, a través del cual pasan todos los mensajes: los mensajes se envían a través del *broker* a los *clientes* que se hayan suscrito a una publicación concreta.

Cada cliente se suscribe a un *topic* o tema. El dispositivo emisor envía el mensaje con un *topic* y el broker lo reenvía a quien está suscrito al mismo. El mensaje puede ser cualquier dato, como información capturada por un sensor, etc. Así, cuando un dispositivo o *cosa* lee datos de un sensor, se puede considerar que es un dispositivo *cliente* que *publica* un mensaje. Cuando tenemos un actuador, se puede considerar que es un cliente *suscrito* a un tema.

Cada cliente inicia la conexión enviando un mensaje *CONNECT* con su autenticación (usuario, contraseña, etc). El broker responde con un mensaje *CONNACK* indicando si acepta o no la conexión. El cliente que envía datos (normalmente de sensores) entonces envia mensajes *PUBLISH* que contienen el *topic* y los datos (*payload*). El cliente receptor (normalmente asociado a actuadores) se suscribe mediante *SUBSCRIBE* y *UNSUBSCRIBE*, a los que el broker responde con *SUBACK* o *UNSUBACK*. La conexión finaliza con un mensaje *DISCONNECT*.

La estructura de los mensajes es la siguiente:

- cabecera: de 2 a 5 bytes, un byte para uno de los códigos de control vistos (*CONNECT, PUBLISH, SUBSCRIBE*...), y de 1 a 4 bytes para la longitud del mensaje
- cabecera adicional: opcional, con un número indeterminado de bytes
- datos (payload): de 0 a 256 MB, incluye el contenido del mensaje, como datos de sensores, información a actuadores, estado del sistema, etc.

En cada aplicación se diseña un formato de datos, no está predefinido en el estándar. Cuando se envía un mensaje, el receptor no tiene por qué estar conectado en ese momento, y el broker se encargará de entregarlo cuando se conecte. Para mantener la conexión, el cliente manda periódicamente un mensaje *PINGREQ* y espera la respuesta del broker (*PINGRESP*).

En cada topic se indica el destinatario, en una serie de niveles, finalizando con el dato del sensor o actuador correspondiente, por ejemplo:

/granja/nave1/sensorhumo

Se puede emplear el símbolo **#** para indicar cualquier dispositivo bajo un nivel; por ejemplo para suscribirnos a todos los topics (dispositivos) en la *nave 3*:

/granja/nave3/#

En cambio el símbolo ***+*** se usa para indicar cualquier zona dentro de un mismo nivel; por ejemplo para suscribirnos a todos los sensores de luz de la *granja*:

/granja/+/sensorluz

El mensaje puede tener entre 2 bytes y 256 MB, si bien suele ocupar unos pocos kB. Un *topic* lanza un evento como actuador o como sensor. Los actuadores estarán suscritos a topics y actuarán dependiendo de ellos, y los sensores publicarán en topics la información que han capturado del exterior. El uso de MQTT no obliga al uso de una plataforma en la nube, pudiendo estar el broker en el interior de una red. Tanto si está en el interior como en el exterior, es posible recibir datos en un navegador web mediante *websockets*, permitiendo al navegador visualizar directamente datos MQTT.

5.- Enlaces de interés

https://cloud.google.com/solutions/iot-overview?hl=es-419

https://www.internetsociety.org/es/resources/doc/2015/iot-overview

http://www.bcendon.com/el-origen-del-iot/

https://es.wikipedia.org/wiki/Internet_de_las_cosas

https://es.wikipedia.org/wiki/IFTTT

https://en.wikipedia.org/wiki/MQTT

https://www.link-labs.com/blog/sigfox-vs-lora

https://programarfacil.com/esp8266/mqtt-esp8266-raspberry-pi/

https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html

6.- Créditos y licencias

Autor: Antonio Cano Morcillo. Publicado por el Centro Regional de Formación del Profesorado de Castilla-La Mancha.

http://centroformacionprofesorado.castillalamancha.es/comunidad/crfp