# Adquisición de información del mundo real

El Internet de las Cosas (IoT, por sus siglas en inglés) permite que personas, dispositivos y sistemas permanezcan constantemente conectados mediante Internet, con acceso a información en tiempo real desde prácticamente cualquier lugar del mundo. Sin embargo, para que esto sea posible, es necesario comprender cómo se traduce la información del mundo físico al entorno digital, de modo que pueda ser procesada tanto por humanos como por sistemas computacionales avanzados.

Se vive en una era donde los dispositivos interconectados no sólo reciben órdenes, sino que también se ajustan automáticamente a rutinas y contextos. Entender su funcionamiento es fundamental para aprovechar su potencial. Por ejemplo, al apagar la alarma del despertador por la mañana, podría activarse automáticamente la cafetera, considerando incluso recomendaciones personalizadas para mantener una dieta saludable. Otro caso sería que un reloj inteligente detecte la ubicación y avise al estar cerca de casa, anticipando tareas o activando funciones del hogar inteligente. Las posibilidades son casi ilimitadas: el IoT puede transformar la vida cotidiana, adaptándose a las actividades y promoviendo bienestar y seguridad.

Para que todas estas interacciones sean posibles, es imprescindible medir constantemente los cambios que ocurren en el entorno físico. Esto se logra mediante sensores, dispositivos capaces de registrar variaciones en su entorno —como temperatura, humedad, movimiento o la presencia de gases— y enviar esos datos a sistemas más complejos para su procesamiento. Estos datos no sólo se almacenan, sino que también se ordenan, analizan y transforman para generar respuestas útiles y automatizadas. Más específicamente, un sensor puede medir una variable de forma análoga, o un valor discreto (niveles de temperatura, humedad, etc.). También existen sensores que miden una variable dicotómica del tipo si / no (¿hay humedad? Si / no, ¿hay alta temperatura? Si / no, etc.).

El proceso de captura de estos datos comienza en la **capa de Percepción** dentro de la arquitectura del IoT. Esta capa está compuesta por sensores y actuadores: los sensores transforman señales físicas (como calor, luz o presión) en señales eléctricas, mientras que los actuadores hacen lo inverso, ejecutando acciones físicas a partir de señales digitales. Gracias a esta capa, los sistemas informáticos pueden detectar y comprender el estado del entorno que los rodea.

Una vez que la información ha sido capturada, interviene la **capa de Red**, que tiene como misión transportar los datos generados por los sensores hacia los centros de procesamiento y análisis. Esta transmisión puede realizarse mediante redes inalámbricas (como Wi-Fi o redes celulares) o mediante conexiones cableadas (como Ethernet). La red convierte la información en formatos digitales adecuados para su intercambio y procesamiento, asegurando su disponibilidad en tiempo real.

Con el crecimiento exponencial de dispositivos inteligentes, también ha aumentado la cantidad de datos generados. Este flujo de información es tan grande que muchas veces no resulta eficiente procesarlo localmente, en el mismo lugar físico donde se ubican los sensores. Por esta razón, muchos de estos dispositivos ya vienen integrados con capacidades de conectividad a Internet, permitiendo que los datos se envíen directamente a servidores remotos para su análisis y almacenamiento. Además, existen dispositivos intermediarios —como Gateways o pasarelas IoT— que se encargan de gestionar y dirigir este flujo de información, controlando la comunicación entre sensores y sistemas centrales.

Chips de silicio miniaturizados son diseñados con nuevas capacidades en factores de forma cada vez más pequeños y con mejor rendimiento en procesado y eficiencia. Los costes están bajando, siguiendo la Ley de Moore. El coste del ancho de banda también ha disminuido y de igual modo los costes de procesado, permitiendo que más dispositivos estén no únicamente conectados, sino que sean lo suficientemente inteligentes como para saber qué hacer con todos los nuevos datos que se generan o reciben.

Capacidades como el conocimiento del entorno y la comunicación entre máquinas son consideradas de alta prioridad para la IoT. Prioridades adicionales son la integración de memoria y la potencia de procesado, la capacidad de resistir a entornos severos, y una seguridad asequible. Por otra parte, el desarrollo de núcleos de procesadores/microcontroladores de ultra bajo consumo de energía diseñados específicamente para dispositivos móviles de la IoT y una nueva clase de sistemas inteligentes céntricos de la IoT simples y asequibles serán un factor clave. Las soluciones a este respecto van desde máquinas de estado finito micro programadas al uso de microcontroladores. La elección es un equilibrio entre la flexibilidad, la capacidad de programación, el área de silicio y el consumo de energía. Los dispositivos requieren alguna forma de almacenamiento no volátil (EEPROM/FRAM/Polímero), independientemente de si se trata de una memoria de máscara, programable una sola vez, o bien regrabable eléctricamente. La memoria no volátil regrabable es claramente la preferida para lograr un alto rendimiento durante la fase de producción y al mismo tiempo servir como memoria de usuario, pudiendo ser programada y almacenar datos del sensor.

En conjunto, estas tecnologías permiten que el IoT funcione como una red viva, capaz de percibir el mundo físico, interpretarlo y responder en tiempo real. Gracias a esta interacción constante entre sensores, redes y sistemas computacionales, el IoT tiene el poder de transformar radicalmente la manera en que se vive, trabaja y relaciona con el entorno.

## Ley de Moore

En 1975, Gordon Moore (cofundador en 1968 de la compañía Intel) afirmó que “el número de transistores por centímetro cuadrado en un circuito integrado se duplicaría aproximadamente cada 18 a 24 meses”. Lo anterior puede resumirse en que cada dos años hay en el mercado computadoras más pequeñas, de menor consumo y a más bajo costo, pero muchísimo más potentes que sus predecesores. Computadoras sobre las que hoy corren aplicaciones que abarcan todos los sectores de la economía, que simplifican muchas operaciones y permiten volver simple lo complejo, automatizar lo que antes era manual y hacer más cosas con menos recursos.

No se puede afirmar que es solo debido al crecimiento exponencial del número de transistores en un microprocesador que la productividad ha aumentado de forma significativa en algunos países (nuevamente, sobre todo en los desarrollados), pero ha sido un gran factor de potenciación de las economías que han basado sus procesos empresariales en las TIC.

# Protocolos de información entre periféricos y dispositivos.

## I2C

I2C es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. También es conocido como IIC o TWI – Two Wire Interface.

El protocolo I2C es uno de los más utilizados para comunicarse con sensores digitales, ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos, dentro de la misma trama, entre otras ventajas.

La conexión de tantos dispositivos al mismo bus es una de las principales ventajas. Además si se compara a I2C con otro protocolo serial, como Serial TTL, este incluye más bits en su trama de comunicación que permite enviar mensajes más completos y detallados.

Los mensajes que se envían mediante un puerto I2C, incluye además del byte de información, una dirección tanto del registro como del sensor. Para la información que se envía siempre existe una confirmación de recepción por parte del dispositivo. Por esta razón se diferencia a los distintos elementos involucrados en este tipo de comunicación.

### Esquema de comunicación

Siempre que se habla de una comunicación oral, se entiende que es entre dos o más personas. Como consecuencia se puede también indicar que en una comunicación digital existen distintos dispositivos o elementos. En el caso de I2C se diferencian dos elementos básicos, un MAESTRO y un ESCLAVO. La figura muestra una conexión típica de tres dispositivos, el bus consiste de dos líneas llamadas, Serial DAta – SDA y Serial CLock – SCL. Es decir, Datos Seriales y Reloj Serial. En particular al bus se le conectan dos resistencias en arreglo pull-up, de entre 2.2K y 10K.

Diagrama

El contenido generado por IA puede ser incorrecto.

El MAESTRO I2C se encarga de controlar al cable de reloj, por sus siglas en inglés llamada SCL – Serial CLock. Además el MAESTRO se encarga de iniciar y parar la comunicación. La información binaria serial se envía sólo por la línea o cable de datos seriales, en inglés se llama SDA – Serial DAta. Dos Maestros no pueden hacer uso de un mismo puerto I2C. Puede funcionar de dos maneras, como maestro-transmisor o maestro-receptor.

Los modos de comunicación en I2C se refieren a las distintas tramas que pueden formarse en el bus. Estas tramas o modos dependen de por ejemplo, si se quiere leer al sensor esclavo, o si se lo quiere configurar. Existen principalmente dos modos de comunicación:

1. Maestro-Transmisor y Esclavo-Receptor. Este modo se usa cuando se desea configurar un registro del esclavo I2C.
2. Maestro-Receptor Y Esclavo-Transmisor. Se usa cuando se quiere leer información del sensor I2C.

Sus funciones principales son:

* Iniciar la comunicación – S
* Enviar 7 bits de dirección – ADDR
* Generar 1 bit de Lectura o Escritura – R/W
* Enviar 8 bits de dirección de memoria
* Transmitir 8 bits de datos –
* Confirmar la recepción de datos – ACK – ACKnowledged
* Generar confirmación de No-recepción, NACK – No-ACKnowledged
* Finalizar la comunicación

El ESCLAVO I2C, generalmente suele ser un sensor. Este elemento suministra de la información de interés al MAESTRO. Puede actuar de dos formas: esclavo-transmisor o esclavo-receptor. Un dispositivo I2C esclavo, no puede generar a la señal SCL. Sus funciones principales son:

* Enviar información en paquetes de 8 bits.
* Enviar confirmaciones de recepción, llamadas ACK

### Bits de la trama del puerto

El protocolo de comunicación I2C se refiere al conjunto de bits que son necesarios para enviar uno o varios bytes de información. En lo particular, para este protocolo existen los siguientes bits importantes:

* Inicio o Start – S
* Parada – P
* Confirmación – ACK
* NoConfirmación – NACK
* Lectura-/Escritura – L/W
* 7 bits para la dirección del dispositivo esclavo/maestro
* 8 bits de dirección ( para algunos sensores pueden ser 16 bits)
* 8 bits de datos

El conjunto de estos bits y su orden va formando distintas tramas de comunicación. Existen distintos modos de comunicación dependiendo del arreglo de estos bits. Tanto el maestro como el esclavo pueden o no generar los bits anteriores, según los modos de comunicación.

El puerto I2C está disponible si las dos líneas, SDA y SCL están en un nivel lógico alto.

## RS232

La información que se transfiere entre el equipo de procesamiento de datos y los periféricos es en forma de datos digitales que se transmiten en modo paralelo o en serie. Las comunicaciones paralelas se utilizan principalmente para conexiones entre instrumentos de prueba o computadoras e impresoras, mientras que la serie se usa a menudo entre computadoras y otros periféricos.

La transmisión del protocolo RS232 implica el envío de datos un bit a la vez, a través de una sola línea de comunicaciones. En contraste, las comunicaciones paralelas requieren al menos tantas líneas como bits en una palabra que se está transmitiendo (para una palabra de 8 bits, se necesita un mínimo de 8 líneas). La transmisión serial RS232 es beneficiosa para las comunicaciones de larga distancia, mientras que el paralelo está diseñado para distancias cortas o cuando se requieren velocidades de transmisión muy altas.

Oficialmente, RS-232 se define como la "Interfaz entre el equipo terminal de datos y el equipo de comunicaciones de datos que utiliza el intercambio de datos binarios en serie". Esta definición define el equipo de terminal de datos (DTE) como la computadora, mientras que el equipo de comunicaciones de datos (DCE) es el módem. Un cable de módem tiene conexiones pin a pin y está diseñado para conectar un dispositivo DTE a un dispositivo DCE.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

RS232 es una forma de comunicación en serie, lo que significa que la información se envía entre dispositivos un bit a la vez. Es una forma de comunicación asincrónica, lo que significa que los dispositivos de envío y recepción pueden operar en diferentes momentos.

El estándar RS232 define los detalles de cómo se envían y reciben los datos entre dispositivos. Las principales características del estándar RS232 son las siguientes:

* Niveles de voltaje: La comunicación RS232 usa señales de voltaje para representar datos binarios. Un voltaje entre -3 V y -15 V representa un "1" binario (también conocido como "Marca"), mientras que un voltaje entre +3 V y +15 V representa un "0" binario (también conocido como "Espacio").
* Transmisión de datos: RS232 admite comunicación dúplex completo, lo que significa que los datos se pueden enviar y recibir en ambas direcciones al mismo tiempo.
* enmarcado: Cada paquete de datos enviado a través de RS232 contiene un bit de inicio, de 5 a 9 bits de datos, un bit de paridad opcional para la detección de errores y uno o dos bits de parada. Esta estructura se denomina "marco".
* Tasa de baudios: La tasa de baudios es el número de cambios de señal por segundo que se envían o reciben a través de la línea. En RS232, la velocidad en baudios generalmente se especifica en bits por segundo (bps).
* Líneas de control: RS232 utiliza varias líneas de control para gestionar el flujo de datos entre el transmisor y el receptor. Estos incluyen líneas como "Terminal de datos listo" (DTR), "Conjunto de datos listo" (DSR), "Solicitud de envío" (RTS) y "Listo para enviar" (CTS).
* Comprobación de paridad: Este es un método para detectar errores en los datos transmitidos. La verificación de paridad agrega un bit adicional (el bit de paridad) a cada palabra de datos (generalmente un byte), que se establece para garantizar que el número total de 1 bits en la palabra (incluido el bit de paridad) sea siempre par o impar, según dependiendo de si se usa paridad par o impar.

### Ventajas sobre otros protocolos

Hay muchos otros protocolos de comunicación disponibles, como RS422, RS485 y USB . Si bien estos protocolos más nuevos pueden ser superiores a RS232 en muchos aspectos (como tasas de transferencia más altas o mejor inmunidad al ruido), RS232 tiene algunas ventajas clave:

* **Sencillez**: En comparación con otros protocolos, RS232 es relativamente fácil de implementar y comprender. Esto puede hacerlo atractivo para los desarrolladores e ingenieros que necesitan una solución de comunicación de datos fiable y rápida.
* **Compatibilidad**: RS232 sigue siendo ampliamente compatible tanto en dispositivos más antiguos como en algunos más nuevos. Esto significa que a menudo es la mejor o la única opción para la comunicación con ciertos tipos de hardware.
* **Comunicación dúplex completo**: A diferencia de RS485, que es semidúplex, RS232 admite comunicación dúplex completo. Esto significa que los datos se pueden enviar y recibir simultáneamente en ambas direcciones.
* **Robustez**: Aunque RS232 puede ser más sensible al ruido que RS422 o RS485, es conocido por su confiabilidad y solidez en una amplia variedad de aplicaciones y entornos.
* **Costo**: El hardware necesario para establecer una comunicación RS232, como cables y conectores, suele ser más económico que el de otros protocolos como USB , RS422 o RS485.
* **Disponibilidad de recursos**: Debido a la larga historia de RS232, existen numerosas herramientas y recursos disponibles para ayudar en la implementación y solución de problemas de este protocolo.

## SPI

La interfaz de periféricos en serie (SPI) es un bus de interfaz utilizado habitualmente para enviar datos entre microcontroladores y pequeños periféricos como registros de desplazamiento, sensores y tarjetas SD. Utiliza líneas de reloj y datos separadas, junto con una línea de selección para elegir el dispositivo con el que se desea hablar.

Un puerto serie común, del tipo con líneas TX y RX, se llama “asíncrono” (no síncrono) porque no hay control sobre cuándo se envían los datos ni ninguna garantía de que ambos lados estén funcionando precisamente a la misma velocidad. Dado que los ordenadores se basan normalmente en que todo está sincronizado con un único “reloj” (el cristal principal conectado a un ordenador que lo controla todo), esto puede ser un problema cuando dos sistemas con relojes ligeramente diferentes intentan comunicarse entre sí.

Para solucionar este problema, las conexiones seriales asíncronas añaden bits de inicio y parada adicionales a cada byte, lo que ayuda al receptor a sincronizarse con los datos a medida que llegan. Ambas partes deben acordar de antemano la velocidad de transmisión (por ejemplo, 9600 bits por segundo). Las ligeras diferencias en la velocidad de transmisión no suponen un problema porque el receptor vuelve a sincronizar al principio de cada byte.

La comunicación serial asíncrona funciona bien, pero agrega una gran sobrecarga tanto en los bits de inicio y parada adicionales enviados con cada byte, como en el complejo hardware necesario para enviar y recibir datos. Y si ambos lados no están configurados a la misma velocidad, los datos recibidos serán basura. Esto se debe a que el receptor está muestreando los bits en momentos muy específicos (las flechas en el diagrama anterior). Si el receptor está mirando en los momentos equivocados, verá los bits equivocados.

SPI se presenta como solución. Es un bus de datos “síncrono”, lo que significa que utiliza líneas separadas para los datos y un “reloj” que mantiene ambos lados en perfecta sincronía. El reloj es una señal oscilante que indica al receptor exactamente cuándo debe muestrear los bits de la línea de datos. Puede ser el flanco ascendente (de bajo a alto) o descendente (de alto a bajo) de la señal de reloj; la hoja de datos especificará cuál de ellos debe utilizarse. Cuando el receptor detecte ese flanco, mirará inmediatamente a la línea de datos para leer el siguiente bit (ver las flechas en el siguiente diagrama). Dado que el reloj se envía junto con los datos, no es importante especificar la velocidad, aunque los dispositivos tendrán una velocidad máxima a la que pueden funcionar.

Diagrama

El contenido generado por IA puede ser incorrecto.

Una de las razones por las que SPI es tan popular es que el hardware receptor puede ser un simple registro de desplazamiento. Se trata de una pieza de hardware mucho más sencilla que el UART (receptor/transmisor asíncrono universal) que requiere la serie asíncrona.

### Funcionamiento

En SPI, sólo un lado genera la señal de reloj (normalmente llamada CLK o SCK para Serial ClocK). El lado que genera el reloj se llama “controlador”, y el otro lado se llama “periférico”. Siempre hay un solo controlador, pero puede haber múltiples periféricos.

Cuando se envían datos desde el controlador a un periférico, se envían por una línea de datos llamada COPI, por “Controller Out / Peripheral In”. Si el periférico necesita enviar una respuesta al controlador, el controlador continuará generando un número preestablecido de ciclos de reloj, y el periférico pondrá los datos en una tercera línea de datos llamada CIPO, para “Controller In / Peripheral Out”.

Como el controlador siempre genera la señal de reloj, debe saber de antemano cuándo un periférico necesita devolver datos y cuántos datos serán devueltos. Esto es muy diferente a la serie asíncrona, donde se pueden enviar cantidades aleatorias de datos en cualquier dirección y en cualquier momento. En la práctica esto no es un problema, ya que SPI se usa generalmente para hablar con sensores que tienen una estructura de comandos muy específica. Por ejemplo, si se envía el comando para “leer datos” a un dispositivo, se sabe que el dispositivo siempre enviará dos bytes a cambio.

SPI es “full duplex” (tiene líneas de envío y recepción separadas), y, por lo tanto, en ciertas situaciones, puede transmitir y recibir datos al mismo tiempo.

Diagrama

El contenido generado por IA puede ser incorrecto.

### Múltiples periféricos

Hay dos formas de conectar múltiples periféricos a un bus SPI:

1. En general, cada periférico necesitará una línea CS independiente. Para hablar con un periférico en particular, se hace que la línea CS de ese periférico sea baja y se mantiene el resto alta (que dos periféricos se activen al mismo tiempo resultaría en que intenten hablar en la misma línea CIPO resultando en datos confusos). Muchos periféricos requerirán muchas líneas CS; si no quedan salidas, hay chips decodificadores binarios que pueden multiplicar las salidas CS.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

1. Por otro lado, algunas partes prefieren estar conectadas en cadena, con la CIPO (salida) de una que va a la COPI (entrada) de la siguiente. En este caso, una sola línea CS va a todos los periféricos. Una vez enviados todos los datos, la línea CS se eleva, lo que hace que todos los chips se activen simultáneamente. Esto se utiliza a menudo para los registros de desplazamiento en cadena y los controladores de LED direccionables.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Para esta disposición, los datos se desbordan de un periférico a otro, por lo que, para enviar datos a cualquier periférico, se tiene que transmitir suficientes datos para llegar a todos ellos. Además, considerar que el primer dato que se transmita acabará en el último periférico.

Este tipo de diseño se utiliza normalmente en situaciones de sólo salida, como la conducción de LEDs donde no se necesita recibir ningún dato de vuelta. En estos casos se puede dejar la línea CIPO del controlador desconectada. Sin embargo, si los datos necesitan ser devueltos al controlador, se puede hacer cerrando el bucle de la cadena (cable azul en el diagrama anterior). Si se hace esto, los datos de retorno del periférico 1 tendrán que pasar por todos los periféricos antes de volver al controlador, así que hay que asegurarse de enviar suficientes comandos de recepción para obtener los datos que necesitas.

## MQTT

MQTT (Transporte de Telemetría en Cola de Mensajes) es un protocolo de mensajería ligero, basado en publicación-suscripción, diseñado para dispositivos con recursos limitados y redes con bajo ancho de banda, alta latencia o poco fiables. Se utiliza ampliamente en aplicaciones del Internet de las Cosas (IoT), proporcionando una comunicación eficiente entre sensores, actuadores y otros dispositivos.

MQTT se ha consolidado como uno de los mejores protocolos de IoT gracias a sus características y capacidades únicas, adaptadas a las necesidades específicas de los sistemas IoT. Algunas de las principales razones incluyen:

* Ligereza: Los dispositivos IoT suelen tener limitaciones en cuanto a potencia de procesamiento, memoria y consumo energético. La mínima sobrecarga y el pequeño tamaño de paquete de MQTT lo hacen ideal para estos dispositivos, ya que consume menos recursos, lo que permite una comunicación eficiente incluso con capacidades limitadas.
* Confiabilidad: Las redes IoT pueden experimentar alta latencia o conexiones inestables. La compatibilidad de MQTT con diferentes niveles de QoS, reconocimiento de sesión y conexiones persistentes garantiza una entrega de mensajes fiable incluso en condiciones adversas, lo que lo hace ideal para aplicaciones IoT.
* Comunicaciones seguras: La seguridad es crucial en las redes IoT, ya que a menudo transmiten datos confidenciales. MQTT admite el cifrado de seguridad de la capa de transporte (TLS) y capa de sockets seguros (SSL), lo que garantiza la confidencialidad de los datos durante la transmisión. Además, proporciona mecanismos de autenticación y autorización mediante credenciales de usuario/contraseña o certificados de cliente, lo que protege el acceso a la red y sus recursos.
* Bidireccionalidad: El modelo de publicación-suscripción de MQTT permite una comunicación bidireccional fluida entre dispositivos. Los clientes pueden publicar mensajes en temas y suscribirse para recibir mensajes sobre temas específicos, lo que facilita el intercambio de datos eficaz en diversos ecosistemas de IoT sin necesidad de acoplamiento directo entre dispositivos. Este modelo también simplifica la integración de nuevos dispositivos, garantizando una fácil escalabilidad.
* Sesiones continuas con estado: MQTT permite a los clientes mantener sesiones con estado con el broker, lo que permite que el sistema recuerde las suscripciones y los mensajes no entregados incluso después de una desconexión. Los clientes también pueden especificar un intervalo de mantenimiento de la conexión, que solicita al broker que compruebe periódicamente el estado de la misma. Si se pierde la conexión, el broker almacena los mensajes no entregados (según el nivel de QoS) e intenta entregarlos cuando el cliente se reconecta. Esta función garantiza una comunicación fiable y reduce el riesgo de pérdida de datos debido a una conectividad intermitente.
* Compatibilidad con dispositivos IoT a gran escala: Los sistemas IoT suelen implicar una gran cantidad de dispositivos, lo que requiere un protocolo capaz de gestionar implementaciones a gran escala. La ligereza de MQTT, su bajo consumo de ancho de banda y la eficiencia en el uso de recursos lo hacen ideal para aplicaciones IoT a gran escala. El patrón de publicación-suscripción permite a MQTT escalar eficazmente, ya que desacopla el emisor del receptor, reduciendo el tráfico de red y el uso de recursos. Además, la compatibilidad del protocolo con diferentes niveles de QoS permite personalizar la entrega de mensajes según los requisitos de la aplicación, garantizando un rendimiento óptimo en diversos escenarios.
* Compatibilidad con lenguajes: Los sistemas IoT suelen incluir dispositivos y aplicaciones desarrollados con diversos lenguajes de programación. La amplia compatibilidad de lenguajes de MQTT facilita la integración con múltiples plataformas y tecnologías, lo que fomenta una comunicación fluida y la interoperabilidad en diversos ecosistemas IoT. Funcionamiento

MQTT funciona a través de un patrón de publicación-suscripción donde un cliente MQTT publica mensajes en un tema específico o se suscribe a un tema para recibir mensajes, todo administrado por un agente MQTT que garantiza la entrega de mensajes de acuerdo con los niveles de QoS (calidad de servicio) especificados.

Cualquier aplicación o dispositivo que ejecute la biblioteca cliente MQTT es un cliente MQTT. Por ejemplo, una aplicación de mensajería instantánea que usa MQTT es un cliente, varios sensores que usan MQTT para reportar datos son un cliente, y varias herramientas de prueba MQTT también lo son.

El Broker MQTT gestiona las solicitudes de conexión, desconexión, suscripción y baja de clientes, así como el enrutamiento de mensajes. Un potente broker MQTT puede soportar conexiones masivas y un rendimiento de mensajes de millones de bits, lo que ayuda a los proveedores de servicios de IoT a centrarse en el negocio y a crear rápidamente una aplicación MQTT fiable.

El patrón de publicación-suscripción se diferencia del patrón cliente-servidor en que separa al cliente que envía mensajes (publicador) del cliente que los recibe (suscriptor). Los publicadores y suscriptores no necesitan establecer una conexión directa, y el Broker MQTT se encarga de enrutar y distribuir todos los mensajes.

El siguiente diagrama muestra el proceso de publicación/suscripción de MQTT. El sensor de temperatura se conecta al servidor MQTT como cliente y publica datos de temperatura en un tema (p. ej., Temperature). El servidor recibe el mensaje y lo reenvía al cliente suscrito al Temperature tema.

Diagrama

El contenido generado por IA puede ser incorrecto.

El protocolo MQTT enruta los mensajes según el tema. El tema define la jerarquía mediante una barra diagonal /, similar a las rutas URL. El tema MQTT admite los siguientes comodines: +y #.

+: indica un único nivel de comodines, como a/+coincidencia a/xo a/y.

#: indica múltiples niveles de comodines, como por a/#ejemplo a/x, a/b/c/d.

El flujo de trabajo, en general, es el siguiente:

Los clientes inician una conexión con el bróker mediante TCP/IP, con cifrado TLS/SSL opcional para una comunicación segura. Los clientes proporcionan credenciales de autenticación y especifican una sesión limpia o persistente.

Los clientes publican mensajes sobre temas específicos o se suscriben a ellos para recibirlos. Los clientes que publican envían mensajes al broker, mientras que los que se suscriben expresan su interés en recibir mensajes sobre temas específicos.

El broker recibe los mensajes publicados y los reenvía a todos los clientes suscritos a los temas relevantes. Garantiza la entrega fiable de mensajes según el nivel de calidad de servicio (QoS) especificado y gestiona el almacenamiento de mensajes para clientes desconectados según el tipo de sesión.

# Bibliografía

* <https://proyectodescartes.org/iCartesiLibri/PDF/IoT.pdf>
* <https://psm.fei.stuba.sk/pages/95/LM08_F_ES.pdf>
* <https://revistascientificas.cuc.edu.co/ingecuc/article/view/490/399>
* <https://hetpro-store.com/TUTORIALES/i2c/?srsltid=AfmBOornhA6ftB1Xj8EkWafkRtM2aeb4woBk0M77iuM9kB-ZKGf8on-Z>
* <https://books.google.com.ar/books?hl=es&lr=&id=vnnEDwAAQBAJ&oi=fnd&pg=PA12&dq=internet+de+las+cosas+protocolos+de+informacion+entre+perifericos+y+dispositivos&ots=oSpCUVgotl&sig=Dhed5Xt5T_PGXgrKxUJIoV0Ooqc&redir_esc=y#v=onepage&q&f=false>
* <https://www.tenveo-video-conference-es.com/info/how-rs232-serial-communication-works-33863320.html>
* <https://opencircuit.es/blog/rs232-protocol-de-gids-voor-beginners>
* <https://cursos.mcielectronics.cl/2022/08/23/serial-peripheral-interface-spi/>
* <https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt>