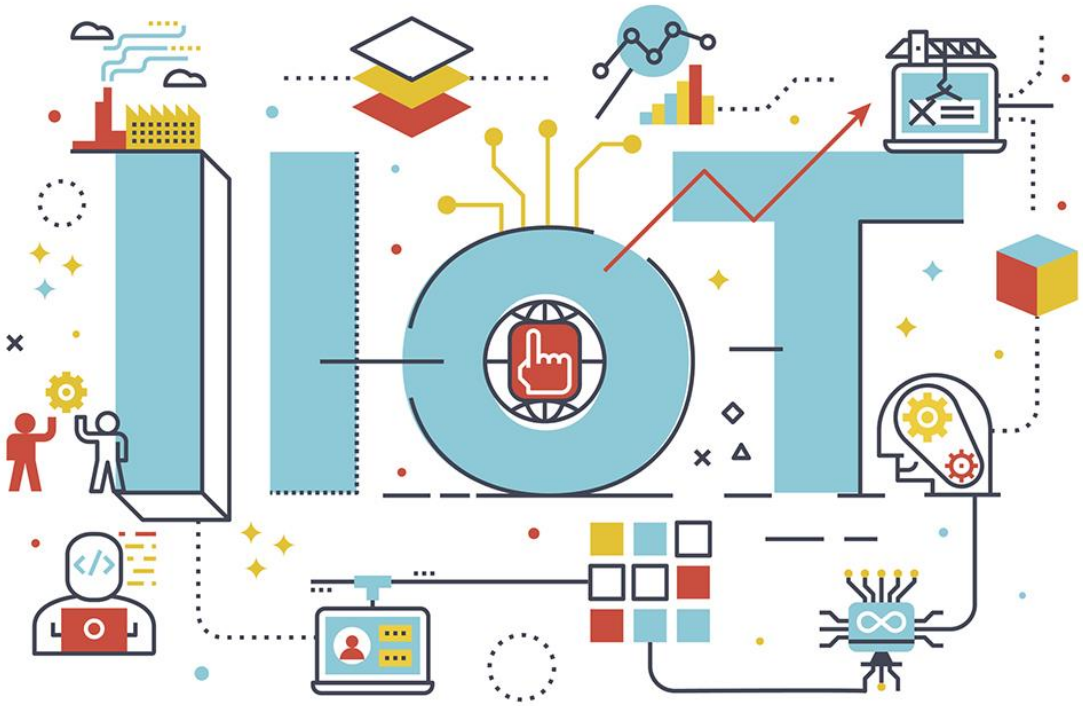
TP 1



ARQUITECTURA Y CONECTIVIDAD

TST-2023

Profesor:

ING. JORGE MORALES

AÑO: 2023

Alumno: Fernando Vexenat

Materia: Arquitectura y Conectividad -TST-2023

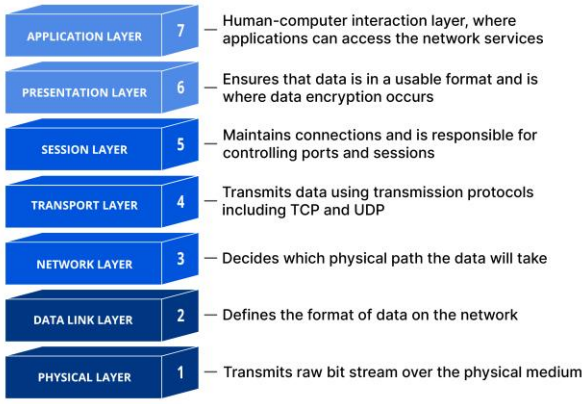
Tecnicatura Superior en Telecomunicaciones

Cuestionario

1) Nombre, describa y grafique las capas OSI.

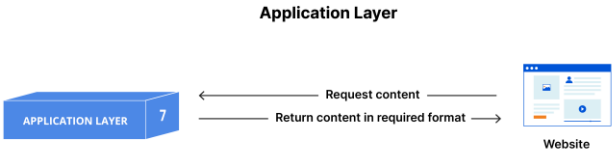
El modelo Open Systems Interconnection (OSI) es un modelo conceptual creado por la Organización Internacional para la Estandarización, el cual permite que diversos sistemas de comunicación se conecten usando protocolos estándar. En otras palabras, el OSI proporciona un estándar para que distintos sistemas de equipos puedan comunicarse entre sí.

El modelo OSI se puede ver como un lenguaje universal para la conexión de las redes de equipos. Se basa en el concepto de dividir un sistema de comunicación en siete capas abstractas, cada una apilada sobre la anterior.



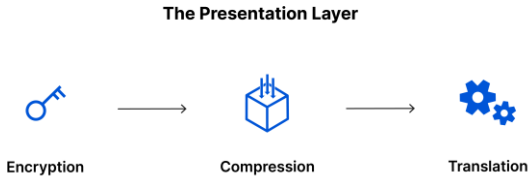
Las siete capas de abstracción del modelo OSI pueden definirse de la siguiente manera, en orden descendente:

7. Capa de aplicación



Esta es la única capa que interactúa directamente con los datos del usuario. Las aplicaciones de software, como navegadores web y clientes de correo electrónico, dependen de la capa de aplicación para iniciar comunicaciones. Sin embargo, debe quedar claro que las aplicaciones de software cliente no forman parte de la capa de aplicación; más bien, la capa de aplicación es responsable de los protocolos y la manipulación de datos de los que depende el software para presentar datos significativos al usuario. Los protocolos de la capa de aplicación incluyen HTTP, así como también SMTP (el Protocolo simple de transferencia por correo electrónico, uno de los protocolos que permiten las comunicaciones por correo electrónico).

6. Capa de presentación



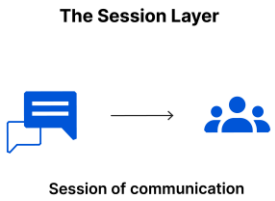
Esta capa es principalmente responsable de preparar los datos para que los pueda usar la capa de aplicación; en otras palabras, la capa 6 hace que los datos se preparen para su consumo por las aplicaciones. La capa de presentación es responsable de la traducción, el cifrado y la compresión de los datos.

Dos dispositivos de comunicación que se conectan entre sí podrían estar usando distintos métodos de codificación, por lo que la capa 6 es la responsable de traducir los datos entrantes en una sintaxis que la capa de aplicación del dispositivo receptor pueda comprender.

Si los dispositivos se comunican a través de una conexión cifrada, la capa 6 es responsable de añadir el cifrado en el extremo del emisor, así como de decodificar el cifrado en el extremo del receptor, para poder presentar a la capa de aplicación datos descifrados y legibles.

Después, la capa de presentación es también la encargada de comprimir los datos que recibe de la capa de aplicación antes de ser enviados a la capa 5. Esto ayuda a mejorar la velocidad y la eficiencia de la comunicación mediante la minimización de la cantidad de datos que serán transferidos.

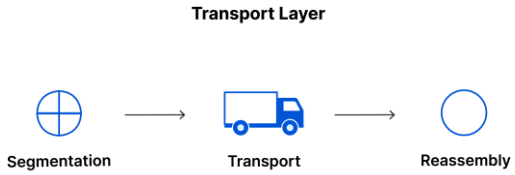
5. Capa de sesión



La capa de sesión es la responsable de la apertura y cierre de comunicaciones entre dos dispositivos. Ese tiempo que transcurre entre la apertura de la comunicación y el cierre de esta se conoce como sesión. La capa de sesión garantiza que la sesión permanezca abierta el tiempo suficiente como para transferir todos los datos que se están intercambiando; tras esto, cerrará sin demora la sesión para evitar desperdicio de recursos.

La capa de sesión también sincroniza la transferencia de datos utilizando puntos de control. Por ejemplo, si un archivo de 100 megabytes está transfiriéndose, la capa de sesión podría fijar un punto de control cada 5 megabytes. En caso de desconexión o caída tras haberse transferido, por ejemplo, 52 megabytes, la sesión podría reiniciarse a partir del último punto de control, con lo cual solo quedarían unos 50 megabytes pendientes de transmisión. Sin esos puntos de control, la transferencia en su totalidad tendría que reiniciarse desde cero.

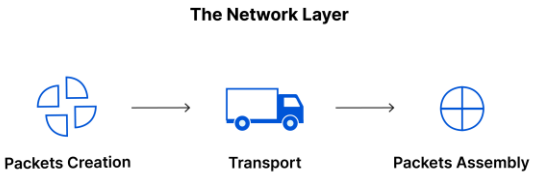
4. Capa de transporte



La capa 4 es la responsable de las comunicaciones de extremo a extremo entre dos dispositivos. Esto implica, antes de proceder a ejecutar el envío a la capa 3, tomar datos de la capa de sesión y fragmentarlos seguidamente en trozos más pequeños llamados segmentos. La capa de transporte del dispositivo receptor es la responsable luego de rearmar tales segmentos y construir con ellos datos que la capa de sesión pueda consumir.

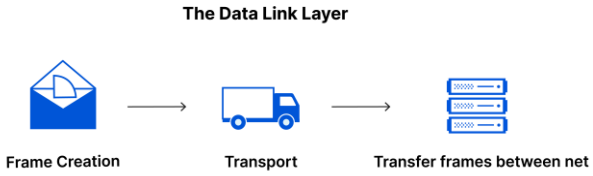
La capa de transporte también es responsable del control de flujo y el control de errores. El control de flujo determina una velocidad óptima de transmisión para garantizar que un emisor con una conexión rápida no abrume a un receptor con una conexión lenta. La capa de transporte realiza un control de errores en el extremo receptor al garantizar que los datos recibidos estén completos y solicitar una retransmisión si no lo están.

3. Capa de red



La capa de red es responsable de facilitar la transferencia de datos entre dos redes diferentes. Si los dispositivos que se comunican se encuentran en la misma red, entonces la capa de red no es necesaria. Esta capa divide los segmentos de la capa de transporte en unidades más pequeñas, llamadas paquetes, en el dispositivo del emisor, y vuelve a juntar estos paquetes en el dispositivo del receptor. La capa de red también busca la mejor ruta física para que los datos lleguen a su destino; esto se conoce como enrutamiento.

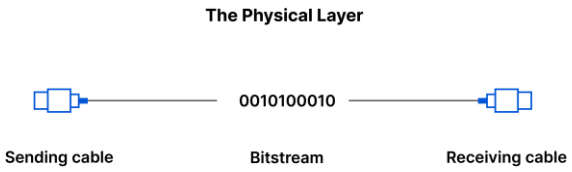
2. Capa de enlace de datos



La capa de enlace de datos es muy similar a la capa de red, excepto que la capa de enlace de datos facilita la transferencia de datos entre dos dispositivos dentro la *misma* red. La capa de enlace de datos toma los paquetes de la capa de red y los divide en partes más pequeñas que se denominan tramas. Al igual que la capa

de red, esta capa también es responsable del control de flujo y el control de errores en las comunicaciones dentro de la red (la capa de transporte solo realiza tareas de control de flujo y de control de errores para las comunicaciones dentro de la red).

1. Capa física



Esta capa incluye el equipo físico implicado en la transferencia de datos, tal como los cables y los conmutadores de red. Esta también es la capa donde los datos se convierten en una secuencia de bits, es decir, una cadena de unos y ceros. La capa física de ambos dispositivos también debe estar de acuerdo en cuanto a una convención de señal para que los 1 puedan distinguirse de los 0 en ambos dispositivos.

2) ¿Cómo se comunican los dispositivos IoT?, Esquematizar y ejemplificar.

Los dispositivos de la Internet de las Cosas (IoT) se comunican de docenas de maneras diferentes, utilizando cientos de protocolos distintos. Esto se debe a que la forma en que se comunican depende de lo que son, de dónde están, de los otros dispositivos y sistemas con los que necesitan hablar y de lo que tienen que decir. No existe un único y mejor protocolo, que es esencialmente el "lenguaje" común utilizado para dirigir los mensajes de un dispositivo IoT a otro. La elección correcta depende siempre de las necesidades específicas de la aplicación.

También hay que tener en cuenta las limitaciones. ¿Cuál es el presupuesto de energía del dispositivo? ¿Cuáles son las limitaciones de coste? ¿Cuáles son los requisitos de tamaño físico, seguridad, tiempo de comercialización, regiones geográficas y mantenimiento remoto? En este artículo examinaremos los componentes integrados de un sistema de comunicación IoT y discutiremos cómo las diferentes necesidades y contextos determinan la mejor solución para cada caso de uso.

Componentes para la comunicación de dispositivos IoT

Aunque los sistemas de IoT tienen muchas arquitecturas diferentes, la mayoría incluyen los siguientes componentes:

1. IoT dispositivo, desde el más pequeño sensor de temperatura hasta un gigantesco robot industrial
2. Comunicaciones locales: el método que utiliza el dispositivo para hablar con los dispositivos vecinos
3. Protocolo de aplicación - el marco que define cómo se transporta el contenido de la información
4. Pasarelas: traducen y retransmiten la información, normalmente enlazando las redes de dispositivos locales con Internet
5. Servidores de red: sistemas que gestionan la aceptación y transmisión de los datos de IoT , normalmente ubicados dentro de los centros de datos en la nube

6. Aplicaciones en la nube: procesan los datos de IoT para convertirlos en información útil y presentarlos a los usuarios.
7. Interfaz de usuario: donde los usuarios ven la información de IoT , la manipulan y emiten órdenes a los dispositivos de IoT

IoT Dispositivos

Cuando hablamos de dispositivos IoT, solemos describir cosas como sensores ambientales, electrodomésticos conectados, rastreadores de vehículos o incluso máquinas de la cadena de montaje.

Aunque un dispositivo IoT es, posiblemente, cualquier aparato electrónico que pueda comunicarse con Internet, no solemos referirnos a teléfonos móviles u ordenadores de uso general.

Conexión de dispositivos inalámbricos

Muchos de estos dispositivos no fueron creados originalmente con capacidades de Internet y deben ser modificados con soluciones posventa para conectarse. Sin embargo, las capacidades de IoT se están diseñando cada vez más en los nuevos dispositivos, donde pueden reducir considerablemente los costes y mejorar la funcionalidad.

Aunque los dispositivos de IoT varían en función de la necesidad para la que fueron creados, casi siempre se incluyen algunos componentes fundamentales. Por ejemplo:

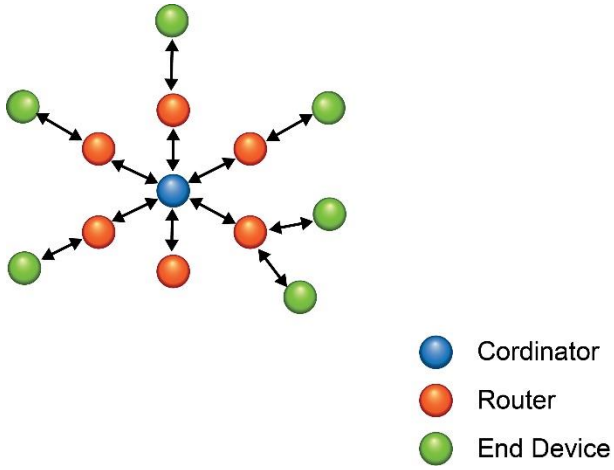
- Suele haber un sensor para detectar sucesos físicos, como el movimiento o una fuga de agua.
- También puede haber actuadores que creen cambios físicos, como encender una luz o cerrar una válvula.
- Estos sensores y actuadores se conectan con uno o más microprocesadores que ejecutan la lógica que impulsa la funcionalidad de IoT .
- Como dispositivo conectado, debe tener al menos un componente de comunicación, ya sea algún tipo de radio o un método de comunicación por cable como Ethernet.
- IoT Los dispositivos suelen funcionar con baterías, lo que hace que la gestión de la energía sea una consideración clave a la hora de seleccionar los equipos, diseñar la funcionalidad y crear estrategias de comunicación.

Todos estos componentes estarán alojados en algún tipo de caja, a menudo bastante pequeña. Dependiendo del entorno, esta caja puede tener que estar sellada y ser hermética, o puede estar muy ventilada para gestionar el calor. Dado que los dispositivos de IoT se instalan a menudo en grandes cantidades, es fundamental acertar con el coste. Cada céntimo cuenta cuando esos céntimos se multiplican por millones.

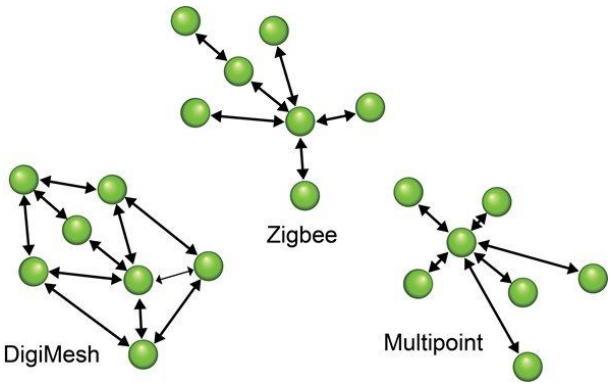
Todos los dispositivos de IoT necesitan comunicarse. Algunos dispositivos sólo envían información; muchos otros envían y reciben. Mientras que algunas comunicaciones con dispositivos pares son directas, las comunicaciones remotas a menudo tendrán que pasar por una pasarela para llegar a su destino. Sea cual sea el destino de los mensajes del dispositivo, todo viaje comienza con un primer paso.

El siguiente gráfico ilustra un modelo de comunicaciones inalámbricas y cómo cada "nodo" de la red inalámbrica desempeña un papel definido. Como se puede ver en este ejemplo, que se denomina "red en estrella", un

módulo inalámbrico inteligente coordina las comunicaciones hacia los dispositivos que actúan como routers y éstos trasladan las comunicaciones a los dispositivos finales.



El escenario cambia para diferentes combinaciones de dispositivos y protocolos inalámbricos. En el siguiente diagrama se puede ver cómo se pueden construir redes que se comporten de diversas maneras con el uso de diferentes protocolos inalámbricos. El mejor protocolo depende de varios factores, como la distancia entre los nodos de comunicación de la red.



El primer paso o "salto" en la comunicación de IoT será por cable o inalámbrico. Las conexiones por cable pueden utilizar un simple protocolo en serie, aunque lo más frecuente es que se emplee un sistema de red como Ethernet, que permite conexiones "directas" del protocolo de Internet (TCP/IP) a un servidor de red o a una aplicación en la nube. Los mensajes que pasan por Internet se enrutan a través de muchos dispositivos diferentes, sin embargo, como arquitectos de IoT, podemos abstraer con seguridad este proceso. Las conexiones por cable son rápidas y fiables, pero a menudo es demasiado caro o poco práctico tender un cableado físico. Naturalmente, para cualquier cosa móvil, los cables están fuera de lugar.

Las comunicaciones inalámbricas de IoT se realizan casi siempre por radio, y hay cientos de protocolos de radio entre los que elegir. Algunos son bastante populares. A continuación, se ofrece una visión general de algunos de los protocolos de comunicación más populares:

- Algunos dispositivos utilizan Wi-Fi, que tiene muchas ventajas siempre que se puedan cumplir sus requisitos de energía y que sus complejas necesidades de procesamiento y aprovisionamiento no

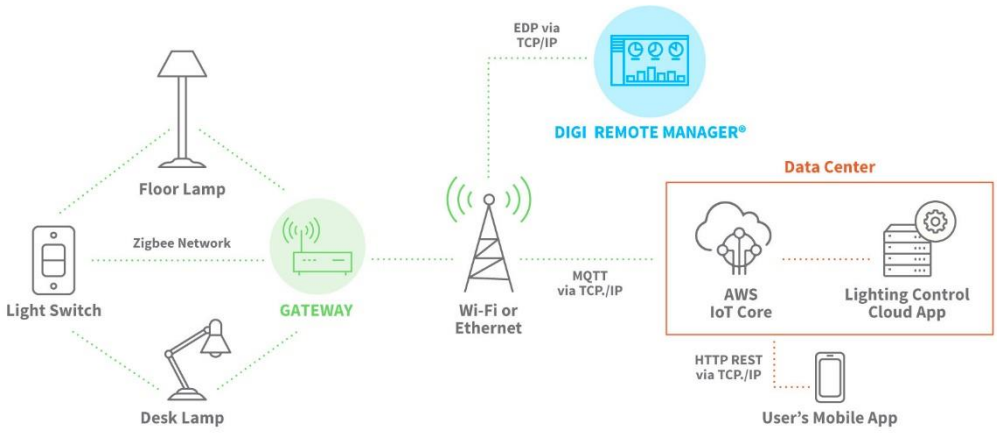
supongan un obstáculo. El Wi-Fi ejecuta TCP/IP de forma nativa, por lo que, una vez configurado, podemos abstraernos de las complejidades de la propia Internet.

- Zigbee y Z-wave son los grandes nombres de las redes de automatización del hogar porque están optimizados para comunicaciones de bajo consumo y bajo ancho de banda, y ambos permiten que los dispositivos del hogar se comuniquen directamente entre sí para mayor velocidad y seguridad. Ninguno de los dos admite directamente el protocolo de Internet, por lo que las comunicaciones fuera del área local suelen enrutarse a través de una pasarela.
- El protocolo LoRaWAN es cada vez más popular también para el bajo ancho de banda IoT . Combina el largo alcance con un ancho de banda muy bajo, soportando kilómetros de alcance en la línea de visión para dispositivos que solo tienen cosas muy pequeñas que decir.
- Bluetooth y su hermana de baja energía BLE son muy populares para los dispositivos sencillos de IoT . Ninguno de los dos puede comunicarse muy lejos, por lo que se utilizará otro dispositivo -a menudo un teléfono móvil- para facilitar la mensajería a larga distancia.
- Las redes celulares pueden ahora acomodar fácilmente los dispositivos de IoT . Los nuevos protocolos celulares, como Cat-M y NB-IoT, permiten que los dispositivos con batería funcionen durante meses sin recargarse, a cambio de un ancho de banda muy limitado.
- Otros protocolos como el 4G LTE y el 5G requieren mucha más potencia, pero también pueden manejar datos más pesados como el vídeo digital.
- También hay muchos protocolos patentados y de un solo fabricante que se adaptan a necesidades de distancia únicas, requisitos especiales de ancho de banda, entornos de radio difíciles y, por supuesto, optimización de costes. No hay un protocolo que los rija a todos. Cada proyecto tendrá su propia solución.

Ejemplo de interruptor de luz

He aquí un ejemplo sencillo de un sistema domótico que utiliza todos estos componentes. Un propietario quiere controlar la lámpara de su comedor mediante un interruptor local, y también poder encender y apagar las luces a distancia. Selecciona un sistema que incluye un interruptor de pared IoT alimentado por pilas. Se comunica directamente con las lámparas mediante el protocolo inalámbrico Zigbee.

Este protocolo incluye un lenguaje especialmente diseñado para la iluminación. Dado que Zigbee es un protocolo de bajo ancho de banda que no utiliza mucha energía, también tiene un alcance limitado. Por lo tanto, para el acceso remoto, el sistema viene con una pequeña pasarela. La pasarela traduce los mensajes Zigbee al protocolo de aplicación MQTT y los pasa a un servidor de red y en la nube que ejecuta la aplicación del sistema de domótica. Esa aplicación en la nube se comunica con una aplicación móvil utilizada por el propietario. Tanto si está en su casa como si está en otro continente, puede ver el estado actual de la luz del comedor y controlarla al instante.



3) Ejemplifique que son dispositivos IoT.

Un dispositivo IoT consiste en un **objeto** al que se le ha dotado de conexión a Internet y cierta inteligencia software, sobre el que se pueden medir parámetros físicos o actuar remotamente y que por tanto permite generar un ecosistema de servicios alrededor del mismo.

Ejemplos de dispositivos IoT serían, por ejemplo:

- Una lavadora sobre la que podamos actuar para cambiar sus modos de lavado a través de un APP.
- Una cafetera que nos dice cuando debemos realizar un mantenimiento rutinario.
- Un coche autónomo.

Los **dispositivos IoT orientados a la captación se les reconoce como *Smart Sensors* (Sensores Inteligentes)** ya que además de obtener el valor del parámetro físico que miden pueden hacer otras muchas funciones.
Además de "inteligencia", estos módulos electrónicos están dotados de comunicación inalámbrica, para posibilitar la interactividad autónoma entre dispositivos incluso estando en movimiento. Las posibilidades son enormes, pues de ese modo, los actuadores de un módulo pueden reaccionar a partir del valor obtenido por los sensores de otros.
Es interesante el uso de sensores y actuadores en la industria para conocer el estado de los procesos con objeto de poder automatizarlos y monitorizarlos analíticamente a lo largo del tiempo. Cuanto mayor sea el número de elementos que intervienen en un proceso mayor será la complejidad para corregirlo y mejorarlo, por eso, la ayuda que nos brindan en este sentido los sensores inteligentes (*Smart Sensors)* es fundamental.

ETIQUETAS INTELIGENTES (SMART TAGS)
A día de hoy las empresas, bien para conocer sus costes productivos en detalle o porque están obligadas a documentarlo, necesitan obtener la **trazabilidad de todo el proceso productivo.** Tenemos por tanto que dar respuesta a preguntas como: ¿Dónde y cuándo se ha producido el producto?, ¿Quién lo ha producido?, ¿Qué materiales han sido utilizados?
Para ello, necesitaremos capturar datos allí donde no es posible hacerlo humanamente y hacer el seguimiento de tiempos, con su traza y características añadidas. Esta tarea, supone un gran reto sobre todo cuando la velocidad de los procesos es alta y se requiere conocer todos los **aspectos relacionados con la fabricación** del producto.
Necesitamos por tanto una tecnología sin contacto (*contactless*) que permita relacionar las señales eléctricas de

las máquinas y los valores de los sensores con el producto para a posteriori conocer cómo fue cada punto del proceso productivo. Para ello disponemos de dos opciones:

Etiquetas impresas

La primera consiste en utilizar una etiqueta pegada o impresa al producto que lo identifica de manera única dentro de un mismo tipo o categoría de producto. Las más conocidas son las etiquetas de códigos de barras y las de códigos QR.

En los puntos de interés del proceso productivo se instalan lectores ópticos que leen el valor de la etiqueta y la envían a un ordenador que almacena en una base de datos la información del proceso (valores de sensores, contadores, energía, etc.) junto a la identificación, pero tanto los códigos QR como los de barras tienen limitaciones importantes a la hora de disponer de toda la trazabilidad del proceso. Una de ellas es que, para realizar la lectura, se requiere contacto visual directo entre la etiqueta y el lector, que no siempre es posible, la otra, es que, además, en este tipo de etiquetas no podemos almacenar información, y, por tanto, la única manera de distinguir entre unidades de producto de un mismo tipo, es mediante la utilización de contadores de piezas e impresoras adicionales.

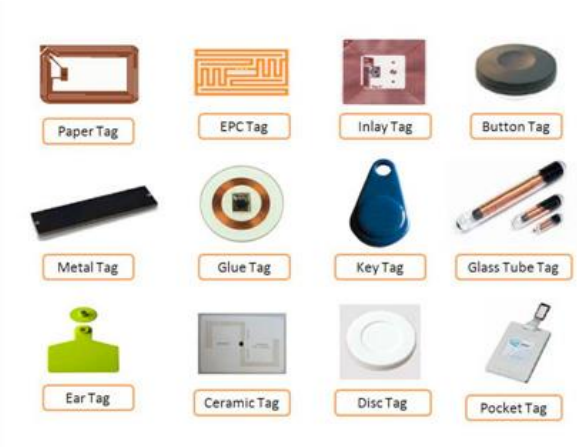
Etiquetas RFID

Otro método cada vez más empleado, debido a las limitaciones de los códigos QR y de barras, es utilizar **tags RFID (*etiquetas de identificación por radiofrecuencia*)**. Estas etiquetas disponen de un minúsculo chip y una antena (ver distintos tipos en la figura siguiente) que permiten escribir información mediante el empleo de un lector/escritor RFID, sin necesidad de existir contacto visual directo entre ambos. Gracias a la memoria del chip RFID es posible almacenar el número de identificación de cada unidad fabricada e incluso almacenar datos del proceso. De esta forma, siempre que queramos, podremos conocer la trazabilidad completa de cualquier unidad fabricada utilizando un lector.

Se suele decir que **la RFID extiende el límite de la tecnología a los objetos**. En los últimos años, debido al abaratamiento de este tipo de etiquetas sus aplicaciones han crecido de forma exponencial siendo usadas principalmente para:

- Control de accesos
- Apertura de vehículos
- Identificación de animales
- Tele-peaje
- Transporte público
- Inventario en tiendas
- Identificación de cajas y palets
- Control de mercancías
- Sistemas de Alarma
- Localización de pacientes en hospitales

El alcance de los tags RFID difiere según sean pasivos (sin batería) cuyo alcance máximo está en torno a los 10 metros o activos (con batería) que pueden superar los 250 metros. Los tag activos además, pueden incorporar sensores y son muy útiles cuando los elementos que queremos localizar se desplazan.



SISTEMAS DE CONTROL EMBEBIDOS (EMBEDDED SYSTEMS)

Un **sistema embebido** es un sistema de computación que ha sido diseñado para realizar funciones dedicadas, normalmente en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general, que están diseñados para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas y por tanto forman parte de multitud de productos en los que se requiere una cierta potencia de proceso como electrodomésticos, vehículos, máquinas, etc.

En la industria estos sistemas se han utilizado tradicionalmente de forma autónoma para controlar, supervisar y ayudar a resolver problemas de todo tipo relacionados con la operativa de equipos, maquinaria o planta como puede ser el de la clasificación de productos atendiendo a su tamaño, forma, color, etc. Su efectividad es tal que podemos decir que hoy día **cualquier decisión que esté basada en lo que observa una persona es reproducible y en muchos casos mejorable por un sistema embebido**.

Algunos ejemplos de estos sistemas con aplicación industrial podrían ser: un lector RFID, un módulo de impresión de códigos de barras, un sistema de visión artificial, un pulsador de emergencia con notificación de alarmas, etc.

Arduino vs Raspberry Pi

Specs	Arduino Uno 	Raspberry Pi Model B+ 
CPU type	Microcontroller	Microprocessor
Operating System	None	Linux (usually Raspbian)
Speed	16 Mhz	700 Mhz
RAM	2KB	512MB
GPU/Display	None	VideoCore IV GPU
Disk	32KB	Depends on SD card
GPIO pins	14 digital pins (includes 6 analog)	26 digital pins
Other connectivity	None	USB, Ethernet, HDMI, audio
Power consumption	0.25W	3.5W

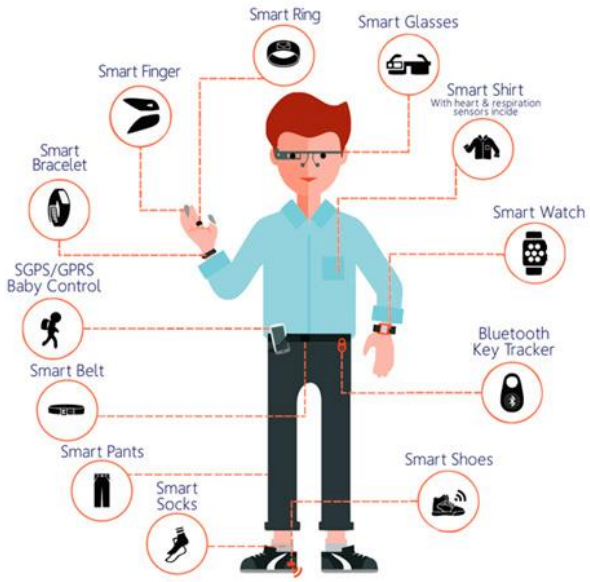
WEARABLES

Wearable hace referencia al conjunto de dispositivos electrónicos que podemos llevar en alguna parte de **nuestro cuerpo** y que son capaces de obtener o mostrar información interactuando de forma continua con el usuario. Para ello disponen de una interfaz de comunicación inalámbrica bidireccional (normalmente bluetooth) que permite conectarlos a Internet a través de una unidad de procesamiento (normalmente un Smartphone).

Relojes inteligentes o smartwatches, zapatillas de deportes con GPS incorporado y pulseras que controlan nuestro estado de salud, son los ejemplos más conocidos de esta tecnología en auge. Por ese motivo, el número de dispositivos no para de crecer, como podemos observar en la figura siguiente, y prácticamente existe un dispositivo para parte del cuerpo.

Existen wearables especialmente concebidos para la industria como cascos, exoesqueletos, botas, guantes, gafas y aunque su aplicación en la industria es bastante reciente, está teniendo mucha aceptación en usos como:

- Prevención de accidentes (detección de caídas, exoesqueletos, etc.)
- Control de actividad (cerebral, pulmonar, constantes vitales, etc.)
- Notificaciones (alertas, averías, mensajes)
- Estudio de comportamientos
- Monitorización de procesos remotamente
- Liberación de las manos en la ejecución de tareas, etc.



4) ¿Qué tecnologías han hecho posible el IoT? Menciona 5 ejemplos.

El Internet de las Cosas, IoT del inglés “Internet of Things”, es el conjunto de tecnologías que permiten que las “cosas” cotidianas estén conectadas a Internet y puedan enviar y recibir datos.
Desde los vehículos autónomos hasta las ciudades inteligentes, pasando por la denominada Industria 4.0, muchas de las innovaciones y nuevos servicios y productos que están apareciendo en el mercado, hacen uso del IoT.
Para potenciar su aplicaciones, el IoT trabaja con otras tecnologías punteras, como Big Data, Inteligencia Artificial, Blockchain, Cloud y Edge Computing, Realidad Aumentada y 5G.

1. Big Data.

Dado que en el IoT genera datos provenientes de miles o millones de sensores, **el Big Data permitirá tomar decisiones automáticas** (a través de actuadores) **o guiadas por las personas** (manejando las aplicaciones informáticas desarrolladas a tal efecto) con objeto de controlar, gobernar y optimizar los servicios IoT desplegados. De hecho, podemos decir que actualmente todos los proyectos de IoT contemplan las tecnologías de Big Data como elemento imprescindible de la solución.

2. Inteligencia Artificial.

Junto con el Big Data, la inteligencia artificial será **necesaria para manejar la gran cantidad de «cosas» conectadas** y para dar sentido a los datos transmitidos desde estos dispositivos.
La inteligencia artificial, las redes neuronales orientadas al aprendizaje profundo y la inteligencia en enjambre (comportamiento colectivo de sistemas descentralizados y autoorganizados) son algunos de los factores innovadores fundamentales que darán tracción la combinación de IA e IoT.

3. Blockchain.

La tecnología Blockchain podría aplicarse para **resolver los problemas de seguridad a los que se enfrentan las soluciones IoT.** A medida que aumentan las conexiones de dispositivos IoT, los puntos de acceso para los hackers también lo hacen. Por ello, la autenticación y la estandarización en cada uno de los elementos IoT son aspectos esenciales para una adopción generalizada.
Entendemos que los esfuerzos para soluciones de Identidad Digital Soberana pueden también aprovecharse en las soluciones IoT, dotando de un identificador único a cada dispositivo y que éste intercambie sólo los datos de identificación y de funcionamiento necesarios en función de qué otro dispositivo o persona esté estableciendo comunicación con él.

4. Cloud y Edge Computing.

Los modelos de “Fog Computing” y de “Edge Computing” surgen para resolver la **problemática de comunicaciones que surge con el IoT.** Un ejemplo ilustrativo del uso de las tres capas sería el de los vehículos autónomos, donde cada vehículo toma decisiones en tiempo real como pueden ser frenar, acelerar o cambiar de carril en función de parámetros internos del coche -EDGE-, del entorno en que se encuentra -FOG- y de los datos generales de tráfico o meteorológicos, así como de nuevos algoritmos generados desde la nube -CLOUD-.
Los sistemas IoT son, por sus características de multitud de dispositivos y multitud de protocolos, claros c**andidatos a ser gestionados desde la nube.** La mayor sofisticación de los dispositivos, y la necesidad de que se comuniquen entre ellos sin intervención humana, está haciendo que se potencie el uso de Edge Computing.
Se está aplicando este modelo en ámbitos como los vehículos conectados y autónomos, ciudades inteligentes y hogares inteligentes.

5. Realidad aumentada.

La tecnología de realidad aumentada se puede usar para **visualizar datos de cientos de sensores simultáneamente**, superponiendo información relevante y procesable sobre el entorno a través de un auricular y de una pantalla semitransparente. Por ejemplo, si se está operando maquinaria pesada, se puede obtener una vista en tiempo real de qué componentes necesitan ser reemplazados.

6. 5G y Mobile IoT.

La tecnología 5G, por sus mejoras en velocidad y latencia respecto al 4G, promete un **salto cualitativo y cuantitativo en aplicaciones** como el coche autónomo o las operaciones quirúrgicas a distancia con robots. Esto hace que aplicaciones que con 4G no podían abordarse, puedan hacerlo ahora.
Por ejemplo, los expertos creen que las redes 5G podrán soportar la enorme cantidad de datos que generarán las ciudades inteligentes. Otro caso donde 5G es imprescindible es en el de los vehículos autónomos. Las innovaciones de 5G podrían permitir que los vehículos autónomos interactúen de manera segura entre sí, con la infraestructura de tráfico e incluso con las carreteras mismas.

5) ¿Qué es la comunicación de datos en serie?, describa su funcionamiento.

6) ¿Cuáles son los protocolos serie más usados?, nombre 3 y descríbalos.

7) ¿Qué es la comunicación de datos en paralelo? Describa su funcionamiento.

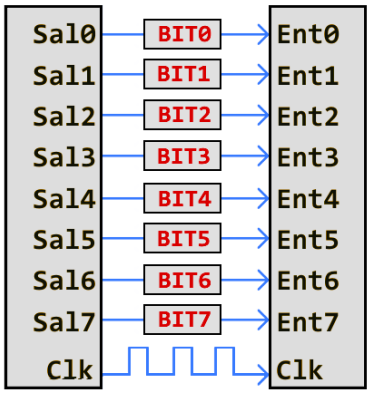
8) ¿Cuáles son los protocolos paralelo más usados?, nombre alguno y descríbalo.

NOTA: LAS 4 PREGUNTAS ANTERIORES SE ENCUENTRAN DESARROLLADAS EN CONJUNTO EN EL TEXTO A CONTINUACION:

La electrónica integrada se conforma con circuitos interconectados (procesadores u otros circuitos integrados) para crear un sistema en el que están repartidas las funciones. Para que esos circuitos individuales intercambien su información, deben compartir un protocolo de comunicación común. Se han definido muchos protocolos de comunicación para lograr este intercambio de datos y, esencialmente, cada uno puede ubicarse en una de dos categorías: **1.** Paralelo o **2.** Serie.

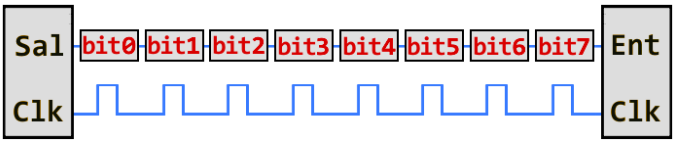
Paralelo versus serie

Las interfaces paralelas transfieren múltiples bits simultáneamente. Por lo general, requieren barras (buses) de datos, que se transmiten a través de ocho, dieciséis o más cables. Los datos se transfieren en amplios oleajes de 1s y 0s.



Un bus de datos de 8 bits, controlado por un reloj,
que transmite un byte por cada pulso de reloj. Se utilizan 9 líneas

En cambio, las **interfaces serie** transmiten sus datos un bit a la vez. Estas interfaces pueden operar con tan solo un cable, por lo general nunca más de cuatro.



Ejemplo de una interfaz serie, transmitiendo un bit cada pulso de reloj. Solo se requieren 2 cables

Piense en las dos interfaces como una fila de automóviles: una interfaz paralela sería una autopista de 8 carriles o más, mientras que una interfaz en serie es más parecida a una carretera de dos carriles. Durante un lapso determinado, la autopista tiene el potencial de llevar a más personas a destino, pero en muchos casos ese sistema sencillo de dos carriles responde a su propósito, y construirlo cuesta una fracción de los fondos.

La comunicación paralela ciertamente tiene sus beneficios. Es rápida, directa y relativamente fácil de implementar. Pero requiere muchas más líneas de entrada/salida (E/S). Si alguna vez ha tenido que traspasar un proyecto de un Arduino Mega a un Arduino UNO básico, sabe que las líneas de E/S en un microprocesador pueden ser preciosas por lo limitadas. Por lo tanto, cada vez más optamos por la comunicación en serie, sacrificando una velocidad potencial para ahorrar pines.

Serie asíncrono (asincrónico)

A lo largo de los años, se han creado docenas de protocolos en serie para satisfacer las necesidades particulares de los sistemas integrados. **USB** (Universal Serial Bus = Bus Serie Universal) y **Ethernet** son dos de las interfaces serie de computación más conocidas en la actualidad. Otras interfaces serie muy comunes son **SPI** (del inglés *Serial Peripheral Interface*), **I²C** (del inglés Inter-Integrated Circuit) y el interfaz serie estándar **TX**/**RX**, del que hablaremos aquí. Estas interfaces serie pueden clasificarse en uno de dos grupos: sincrónico o asincrónico.

Una interfaz serie sincrónica siempre necesita tener una señal de reloj junto a las líneas de datos, por lo que todos los dispositivos en un bus serie sincrónico comparten un pulso común de reloj. Esto hace que en una transferencia en serie más directa, a menudo más rápida, también se requiera al menos un cable adicional entre los dispositivos de comunicación. Entre los ejemplos de interfaces sincrónicas están **SPI** e **I²C**.

Asincrónico significa que los datos se transfieren sin el respaldo de una señal de reloj conectada entre sistemas. Este método de transmisión es ideal para minimizar los cables necesarios, y en consecuencia la cantidad de pines de E/S utilizados, pero implica que debemos poner un poco de esfuerzo adicional en transferir y recibir datos de manera confiable. El protocolo en serie que analizaremos es la forma más común para las transferencias asincrónicas. De hecho, es tan común que cuando la mayoría de la gente dice «en serie», "o serial", están hablando sobre este protocolo.

El protocolo serie sin reloj que analizaremos se usa ampliamente en electrónica integrada. Si está buscando agregar un módulo serie GPS, Bluetooth, XBee, LCD, o muchos otros dispositivos externos a su proyecto, es probable que necesite un interfaz serie.

Reglas de la comunicación serie

El protocolo serie asincrónico tiene una serie de reglas integradas: mecanismos que ayudan a garantizar transferencias de datos sólidas y sin errores. Estos mecanismos, que obtenemos para evitar la señal del reloj externo, son:

- Bits de datos
- Bits de sincronización
- Bits de paridad
- Velocidad en baudios

Teniendo en cuenta la variedad de estos mecanismos de señalización, vemos que no hay una sola manera de enviar datos en serie. El protocolo es altamente configurable. La parte crítica es asegurarse de que ambos dispositivos en una línea serie estén configurados para usar exactamente los mismos protocolos.

Velocidad en baudios

La especificación de velocidad de transmisión indica qué tan rápido se envían los datos a través de una línea serie. Normalmente se expresa en unidades de bits por segundo (bps). Si se invierte la velocidad en baudios, se puede averiguar cuánto tiempo se tarda en transmitir cada bit. Este valor determina durante cuánto tiempo el transmisor mantiene en alto/bajo una línea serie, o a qué velocidad muestrea su línea el dispositivo receptor.

Las velocidades en baudios pueden ser casi cualquier valor dentro de lo que permite el hardware. El único requisito es que ambos dispositivos funcionen a la misma velocidad. Una de las velocidades en baudios más comunes, especialmente para cosas simples donde la velocidad no es crítica, es de 9600 bps. Otras velocidades en baudios «estándar» son 1200, 2400, 4800, 19200, 38400, 57600 y 115200.

Cuanto mayor sea la velocidad en baudios, más rápido se envían/reciben los datos, pero existen límites para la velocidad a la que se pueden transferir los datos. Por lo general, no verá velocidades superiores a 115200, lo que es suficientemente rápido para la mayoría de los microcontroladores. Aumente demasiado y comenzará a

ver errores en el extremo receptor, ya que los pulsos de reloj y los períodos de muestreo no pueden mantenerse.

Estructurando los datos

Cada bloque de datos (generalmente un byte) que se transmite se envía en realidad en un paquete de bits. Los paquetes se crean agregando bits de sincronización y paridad a nuestros datos.

ESTRUCTURA	Inicio	Datos	Paridad	Parada
Tamaño (bits)	1	5-9	0-1	1-2

Algunos símbolos en la estructura del paquete tienen tamaños de bits que son configurables.

Vamos a entrar en los detalles de cada una de estas partes de la estructura del paquete, o bloque.

Bloque de datos

La verdadera sustancia de cada paquete serie es la información que lleva. Ambiguamente llamamos a este bloque de datos un “bloque”, porque su tamaño no está específicamente establecido. En este estándar, la cantidad de datos en cada paquete se puede establecer en valores de 5 a 9 bits. Ciertamente, el tamaño de datos clásico es un byte de 8 bits, pero se usan otros tamaños. Un bloque de datos de 7 bits puede ser más eficiente que 8 si solo está transfiriendo caracteres **ASCII** de 7 bits.

Después de acordar la longitud para un caracter, ambos dispositivos serie también tienen que acordar el formato de sus datos. ¿Se envían los datos desde el bit más significativo (most significative bit = **msb**) al menos significativo, o viceversa? Si no se indica lo contrario, generalmente se puede asumir que los datos se transfieren enviando primero el bit menos significativo (least significative bit = **lsb**).

Bits de sincronización

Los bits de sincronización son dos o tres bits especiales transferidos con cada porción de datos. Son el **bit de inicio** y **el(los) bit(s) de parada**. Tal como indica su nombre, estos bits marcan el principio y el final de un paquete. Siempre hay un único bit de inicio, pero la cantidad de bits de parada se puede configurar en uno o dos (aunque normalmente se deja en uno).

El bit de inicio siempre se indica mediante una línea de datos inactiva que pasa de 1 a 0 (**ALTO** a **BAJO**). Los bits de parada volverán al estado inactivo manteniendo la línea en 1 (**ALTO**).

Bits de paridad

La paridad es una forma de comprobación de errores muy simple y de bajo nivel. Se presenta en dos variantes: impar o par. Para generar el bit de paridad, se suman todos los bits del byte de datos (5 a 9), y el resultado de la suma define si el bit es 1 o 0. Por ejemplo, suponiendo que la paridad se establece en par y se agrega a un byte

de datos como **0b01011101**, que tiene una cantidad impar de 1s (5), el bit de paridad quedaría en 1. Por el contrario, si el modo de paridad se configuró en impar, el bit de paridad sería 0.

La paridad es opcional, y no se usa mucho. Puede ser útil para transmitir a través de medios ruidosos, pero también ralentizará un poco la transferencia de datos y requiere que tanto el transmisor como el receptor implementen el manejo de errores (generalmente, si se detecta error, los datos recibidos con falla deben reenviarse).

Un ejemplo

9600 8N1 – 9600 baudios, **8** bits de datos, **sin** paridad y **1** bit de parada: es uno de los protocolos serie más utilizados. Entonces, ¿cómo se vería un paquete o dos de datos de 9600 8N1?

Un dispositivo que transmita los caracteres ASCII 'O' y 'K' tendría que crear dos paquetes de datos. El valor ASCII de **O** (en mayúsculas) es **79**, que se divide en un valor binario de 8 bits de **01001111**, mientras que el valor binario de **K** es **01001011**. Todo lo que queda es agregar bits de sincronización.

No se establece específicamente, pero la norma más aceptada es que los datos se transfieren enviando primero el bit menos significativo. Observe cómo se envía cada uno de los dos bytes a medida que se lee de derecha (**bit 0**) a izquierda (**bit 7**).

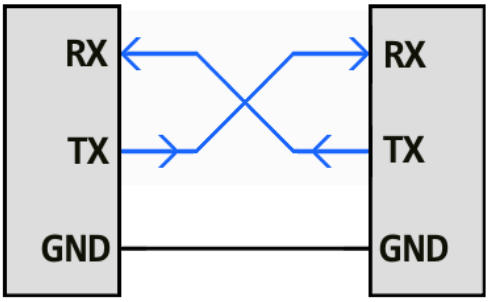
Dado que estamos transfiriendo a 9600 bps, el tiempo empleado en mantener cada uno de esos bits alto o bajo es 1/9600 (bps) o 104 µs por bit.

Por cada byte de datos transmitidos, en realidad como mínimo se envían 10 bits: un bit de inicio, 8 bits de datos y un bit de parada. Entonces, a 9600 bps, en realidad estamos enviando 9600 bits por segundo o 960 (9600/10) bytes por segundo.

Ahora que sabemos cómo construir paquetes serie, podemos pasar a la sección de hardware. Allí veremos cómo esos 1s y 0s, y la velocidad de transmisión, se implementan a un nivel de señal.

Cableado y Hardware

Un bus serie consta de solo dos cables, uno para enviar datos y otro para recibir. Entonces, los dispositivos serie deben tener dos pines serie: el receptor: **RX** y el transmisor: **TX**.



Cableado en serie

Es importante tener en cuenta que esas etiquetas **RX** y **TX** son con respecto al dispositivo en sí. Entonces, el **RX** de un dispositivo debe ir al **TX** del otro y viceversa. Es extraño si uno está acostumbrado a conectar **Vcc** con **Vcc**, **GND** con **GND**, **MOSI** con **MOSI**, etc., pero —pensándolo— tiene sentido. El transmisor debe estar comunicándose con un receptor, no con otro transmisor.

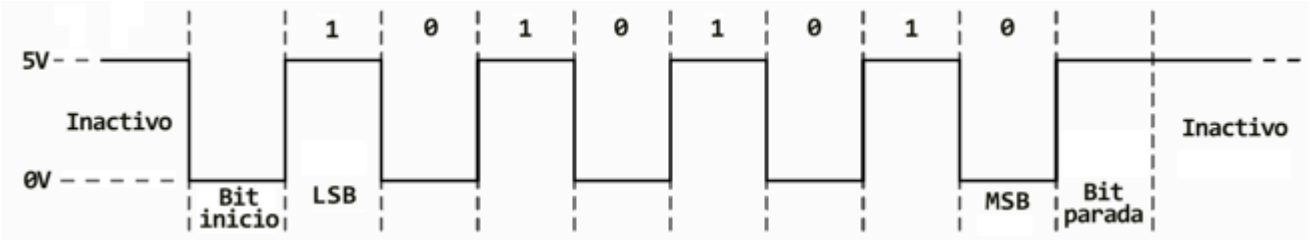
Una interfaz en serie en la que ambos dispositivos pueden enviar y recibir datos es **dúplex completo** (**full-duplex**) o **semidúplex**. Full-duplex significa que ambos dispositivos pueden enviar y recibir simultáneamente. La comunicación **semidúplex** significa que los dispositivos serie deben turnarse para enviar y recibir.

Algunas conexiones serie pueden implementarse con una sola línea entre un dispositivo de transmisión y un dispositivo de recepción. Por ejemplo, los LCD que tienen conexión serie son solo receptores, y realmente no tienen ningún tipo de información para devolver al dispositivo de control. Esto es lo que se conoce como comunicación serie **simplex**. Todo lo que necesita es un solo cable desde la transmisión del dispositivo maestro a la línea **RX** del que recibe.

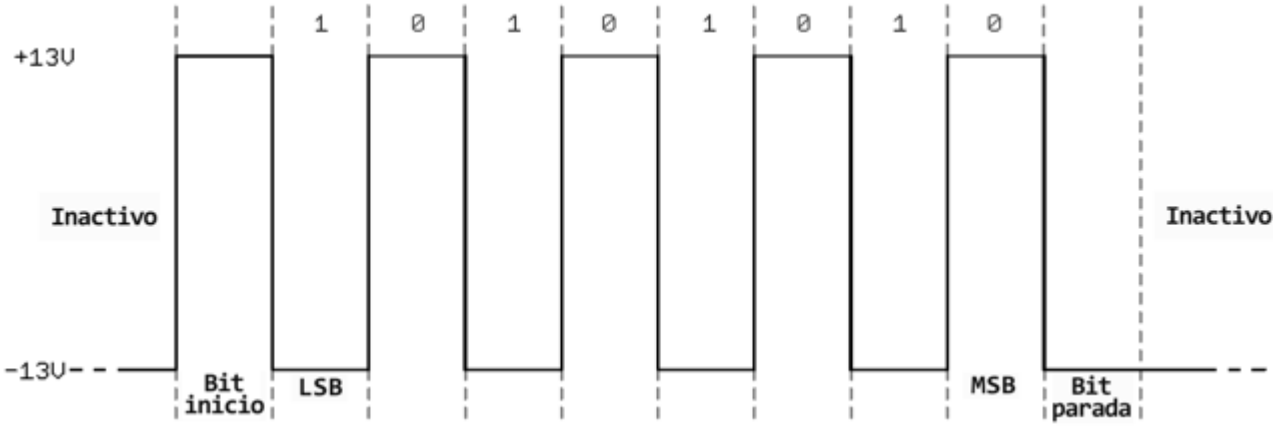
Implementación de hardware

Hasta ahora fue una cobertura de la comunicación serie asíncrona desde un lado conceptual. Sabemos qué cables necesitamos, pero, ¿cómo se implementa realmente la comunicación en serie a nivel de señal? En una variedad de formas, en realidad. Hay todo tipo de estándares para la comunicación en serie. Veamos algunas de las implementaciones de hardware más populares de serie: nivel lógico o **TTL**, y **RS-232**.

Cuando los microcontroladores y otros circuitos integrados de bajo nivel se comunican en serie, generalmente lo hacen a un nivel TTL (Transistor Transistor Logic = Lógica Transistor-Transistor). Las señales serie TTL están en el rango del voltaje que alimenta a un microcontrolador, generalmente de 0V a 3,3V, 0V o 5V. Una señal en el nivel VCC (3,3V, 5V, etc.) indica una línea inactiva, un bit de valor 1 o un **bit de parada**. Una señal de 0V (GND) representa un **bit de inicio** o un bit de datos de valor 0.



El protocolo RS-232, que se puede encontrar en algunas de las computadoras y periféricos más antiguos, es como una interfaz serie TTL puesta cabeza abajo. Las señales RS-232 generalmente oscilan entre -13V y 13V, aunque la especificación permite cualquier cosa desde +/- 3V a +/- 25V. En estas señales, un voltaje bajo (-5V, -13V, etc.) indica la línea inactiva, un **bit de parada** o un **bit de datos** de valor 1. Una señal RS-232 alta significa un bit de inicio o un bit de datos de valor 0. Eso es lo contrario del protocolo TTL.



Entre los dos estándares de señal en serie, el TTL es mucho más fácil de implementar en circuitos integrados. Sin embargo, los niveles de baja tensión son más susceptibles a sufrir pérdidas en las líneas de transmisión largas. El RS-232 o estándares más complejos —como RS-485— son más adecuados para transmisiones en serie de largo alcance.

Cuando conecte dos dispositivos serie juntos, es importante asegurarse de que coincidan los voltajes de su señal. No se puede conectar directamente un dispositivo serie TTL con una línea RS-232. Se deben adaptar esas señales.

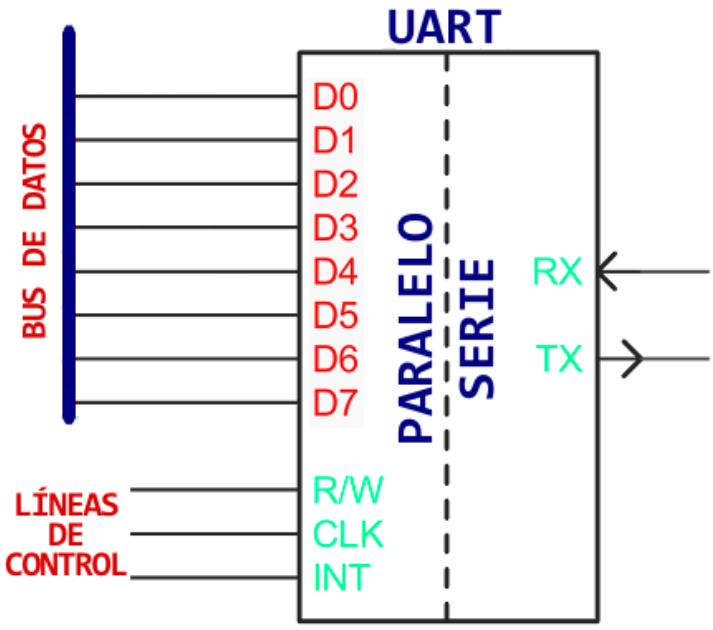
Continuando, exploraremos la herramienta que usan los microcontroladores para convertir sus datos que se encuentran presentes en un bus paralelo desde y hacia una interfaz serial: se llama UART.

UARTs

La última pieza de este armado en serie es encontrar algo para crear los paquetes en serie y controlar las líneas de hardware físico. Esto se concreta con un módulo llamado **UART** (Universal Asynchronous Receiver/Transmiter = Receptor/Transmisor Asíncrono Universal).

Un receptor/transmisor asíncrono universal es un bloque de circuitos responsable de implementar la comunicación en serie. En esencia, este UART actúa como un intermediario entre las interfaces paralelas y seriales. En un extremo del UART hay un bus de ocho o más líneas de datos (más algunos pines de control), en el otro lado están los dos cables serie: **RX** y **TX**.

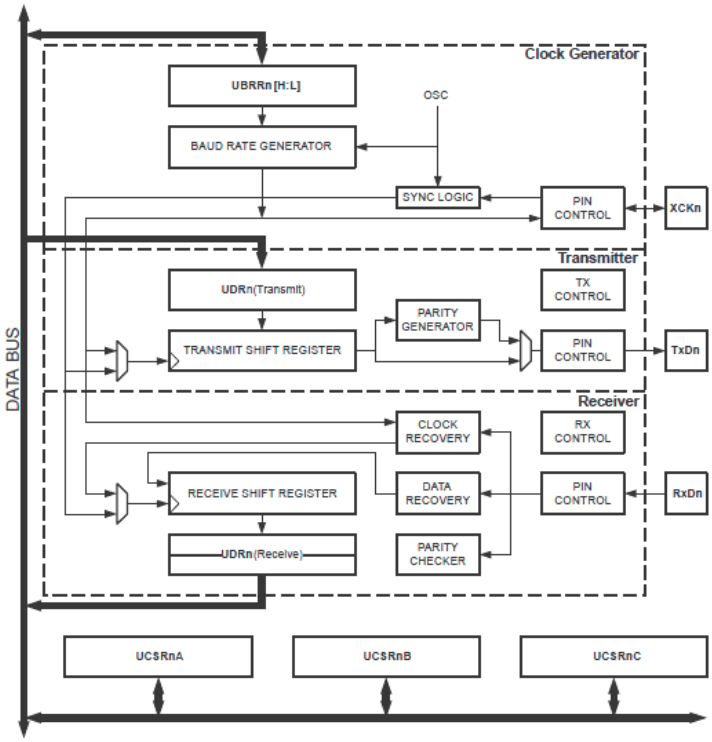
UART simplificado



Los UART existen como circuitos integrados independientes, pero en la actualidad es más común que se encuentren dentro de los microcontroladores. Debemos consultar la hoja de datos de un microcontrolador para ver si tiene algún UART. Algunos no tienen, otros tienen uno, otros tienen varios. Por ejemplo, el Arduino UNO, basado en el «antiguo y fiel» ATmega328, tiene un solo UART, mientras que el Arduino Mega, construido sobre un ATmega2560, tiene cuatro UART.

Como lo indican las letras R y T en el acrónimo, los UART son responsables de enviar y recibir datos en serie. En el lado de transmisión, un UART debe crear el paquete de datos —agregando la sincronización y los bits de paridad— y enviar ese paquete por la línea TX con una sincronización precisa (de acuerdo con la velocidad de transmisión establecida). En el extremo de recepción, el UART tiene que muestrear la línea de RX a velocidades acordes con la velocidad de transmisión que se espera, seleccionar los bits de sincronización y entregar como resultado los datos.

UART interno



Los UART más avanzados pueden enviar los datos que reciben a un archivo de memoria de respaldo, llamado búfer, donde pueden permanecer hasta que el microcontrolador vaya a buscarlos. Los UART generalmente publicarán sus datos almacenados en un búfer con un sistema de “el primero que entra es el primero que sale” (First In First Out = **FIFO**). Los búfer pueden tener apenas unos pocos bits, o pueden ser de gran tamaño, como miles de bytes.

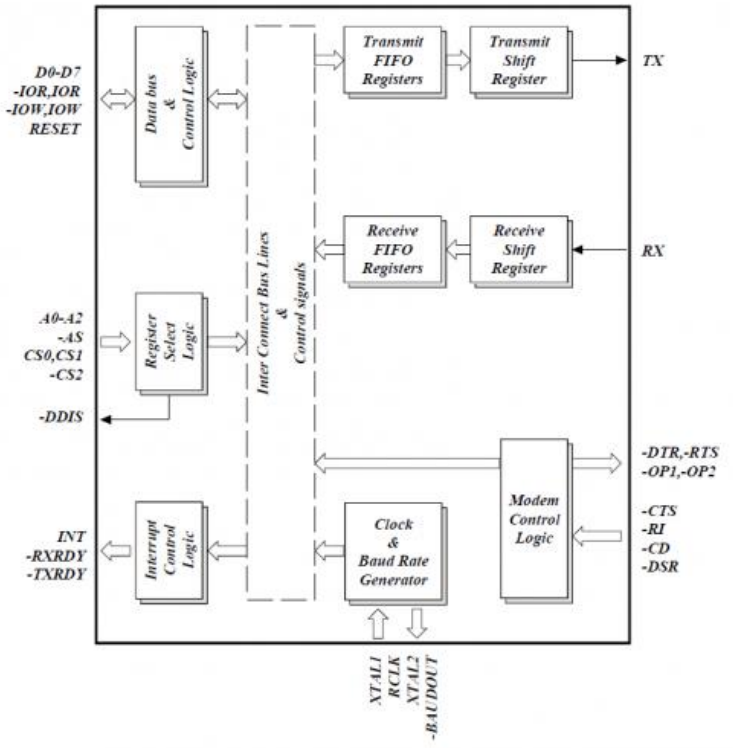


Diagrama de bloques de un UART con FIFO

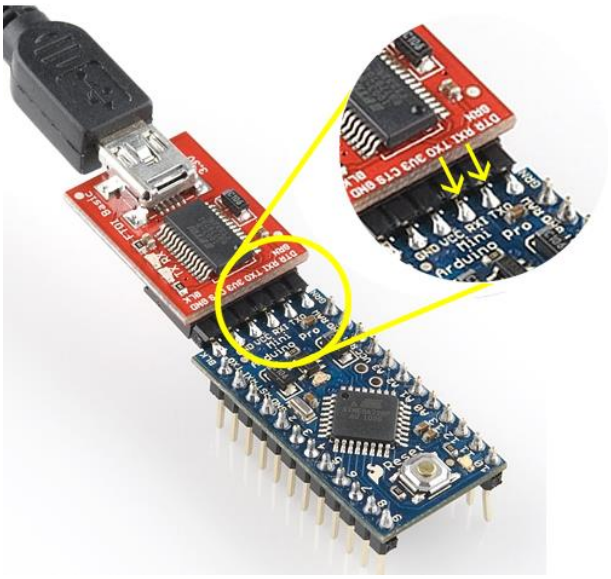
UARTs de software

Si un microcontrolador no tiene un UART, o no tiene suficientes, se puede implementar la interfaz en serie en bits que son controlados directamente por el procesador. Este es el enfoque que tienen las bibliotecas de Arduino como **Software Serial**. El uso de bits es intensivo en el procesador, y no suele ser tan preciso como un UART, pero funciona en caso de necesidad.

Errores comunes

RX a TX, / TX a RX

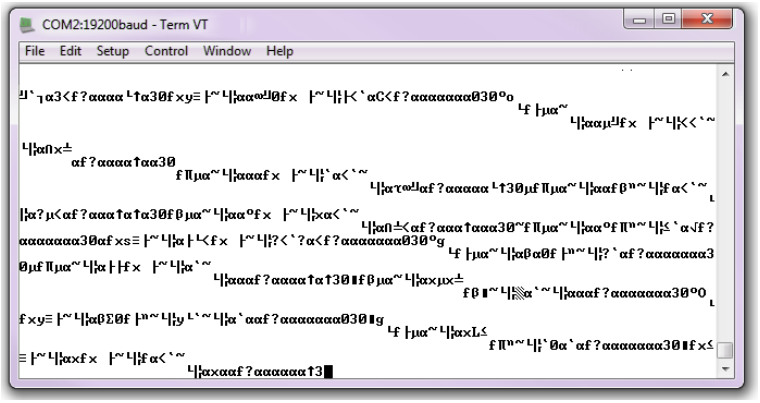
Parece bastante simple, pero es un error que algunos cometen un par de veces. Por mucho que desee que sus etiquetas coincidan, siempre asegúrese de cruzar las líneas RX y TX entre los dispositivos serie.



FTDI Basic programando un Pro Mini. Note que RX y TX están cruzados

Discrepancia en la velocidad de transmisión

Los valores de baudios son como claves en lenguajes de la comunicación en serie. Si dos dispositivos no están hablando a la misma velocidad, los datos pueden ser mal interpretados o completamente perdidos. Si todo lo que el dispositivo receptor ve en su línea de recepción está compuesta de caracteres extraños, llamados “basura” en el ambiente, verifique que coincidan las velocidades en baudios definidas en ambos extremos.



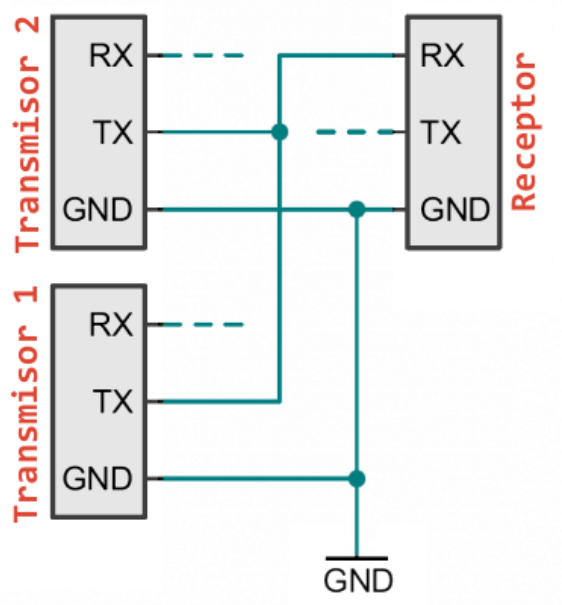
Datos transmitidos a 9600 bps, pero recibidos a 19200 bps. Desajuste de baudios = basura

Contención de transferencia en la línea

La comunicación en serie está diseñada para permitir que solo dos dispositivos se comuniquen a través de un bus en serie. Si más de un dispositivo está intentando transmitir en la misma línea serie, podría encontrarse con una contención de bus.

Por ejemplo, si está conectando un módulo GPS a su Arduino, puede conectar la línea de transmisión de ese módulo a la línea RX de Arduino. Pero ese pin Arduino RX ya está conectado al pin TX del **convertidor de USB a serie** (por ejemplo un chip FTDI) que se usa cada vez que se programa el Arduino, o se usa el Monitor Serie. Esto establece la situación potencial en la que tanto el módulo GPS como el chip FTDI intentan transmitir en la misma línea al mismo tiempo.

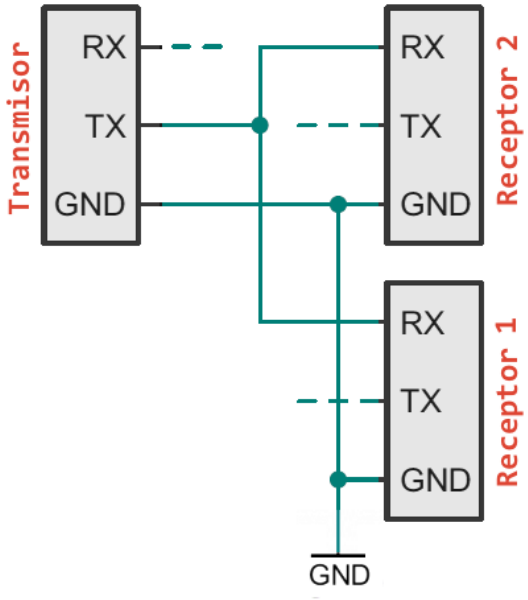
Ejemplo de contención de línea (o bus)



No es bueno que dos dispositivos intenten transmitir datos al mismo tiempo en la misma línea. En el mejor de los casos, ninguno de los dispositivos podrá enviar sus datos. En el peor de los casos, las líneas de transmisión de ambos dispositivos se volverán locas (aunque eso es raro y generalmente están protegidos contra esta situación).

Puede ser seguro conectar varios dispositivos receptores a un solo dispositivo de transmisión. Realmente no cabe dentro de las especificaciones, y probablemente esté mal visto por un ingeniero experimentado, pero funcionará. Por ejemplo, si conectamos un LCD serie a un Arduino, el método más sencillo puede ser conectar la línea RX del módulo LCD a la línea TX del Arduino. El TX de Arduino ya está conectado a la línea RX del programador USB, pero eso deja solo un dispositivo controlando la línea de transmisión.

Implementación segura pero dudosa de un transmisor y dos receptores



La distribución de una línea de transmisión de este tipo puede ser peligrosa desde la perspectiva del firmware, ya que no puede elegir qué dispositivo recibe cual transmisión. La pantalla LCD terminará recibiendo datos que no están destinados a ella, lo que podría ordenarle que pase a un estado desconocido.