

# **Herramientas y plataformas para Internet de las Cosas**

**Internet de las cosas en el contexto de Big  
Data**

Máster Interuniversitario en Big Data: Tecnologías de  
Análisis de Datos Masivos  
Universidade de Santiago de Compostela (USC)

# Índice

- BLOQUE I: Plataformas para gestión de la información
- BLOQUE II: Dispositivos y soluciones para recogida de información de sensores
- BLOQUE III: Los dispositivos móviles como sensores de datos

# **BLOQUE I: Plataformas para gestión de la información**

# Plataformas IoT

- Conjunto de **hardware y software** sobre la que otras aplicaciones pueden funcionar
- La plataforma permite desplegar y ejecutar **aplicaciones**
- Las plataformas de aplicaciones IoT proporcionan un amplio conjunto **herramientas** (funcionalidades independientes que se pueden utilizar para construir aplicaciones IoT)
- Plataforma en la **nube**

# Cloud computing: Introducción

- Paradigma basado en el concepto de **provisión dinámica y bajo demanda** de:
  - Computación (capacidad)
  - Almacenamiento (capacidad)
  - Servicios de red
  - Infraestructuras IT en general
- Recursos disponibles a través de una red de acceso (normalmente Internet)
- Recursos disponibles sólo cuando se necesitan

# Cloud computing: perspectiva de usuario

- Acceso a **infraestructuras IT de altas capacidades** (computación, almacenamiento) sin grandes inversiones
- **Accesible** fácilmente no necesarios grandes conocimientos
- Usuarios **sólo pagan** por lo que realmente *necesitan*
  - Cantidad de recursos
  - Tiempo de uso
- Acceso a los servicios de forma **ubicua**

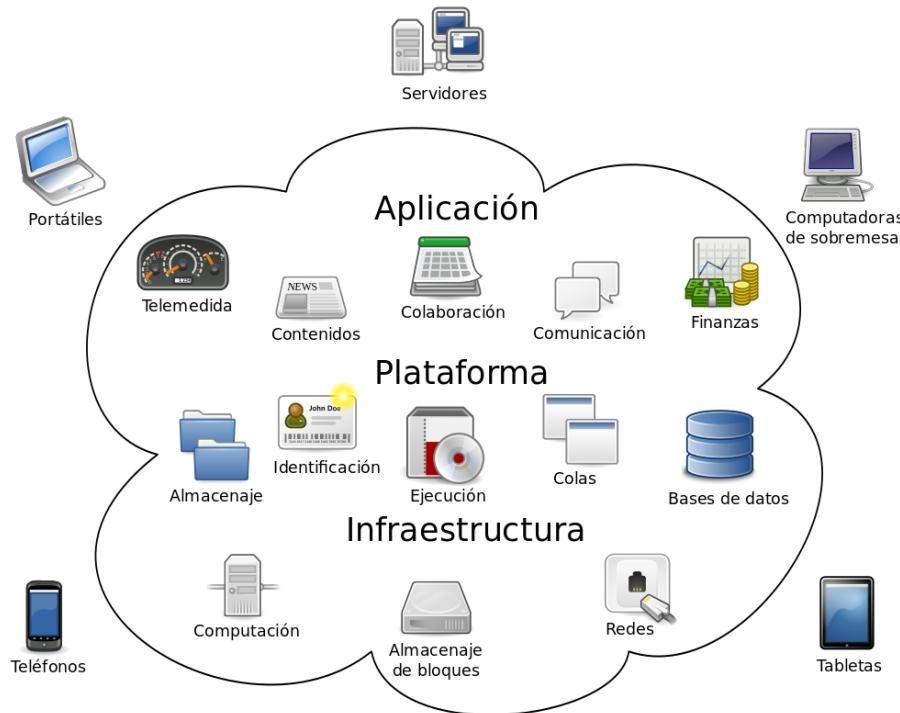
# Cloud computing: perspectiva empresarial

- **Empresas grandes** pueden **delegar** actividades no relacionadas con su actividad principal
- **Empresas pequeñas** pueden abordar nuevos proyectos sin grandes inversiones económicas
- Los **desarrolladores** pueden centrarse en la lógica de negocio y en sus tareas de desarrollo, dejando de lado las tareas de despliegue y gestión de la arquitectura IT

# Cloud computing

- **Virtualización** de infraestructuras y dispositivos
- **Modelos de servicios** flexibles y dinámicos:
  - IaaS (Infrastructure as a Service)
    - Virtualización de Hardware, computación o almacenamiento
  - PaaS (Platform as a Service)
    - Herramientas e interfaces para desarrollo software
  - SaaS (Software as a Service)
    - Aplicaciones finales

# Cloud computing



*“Cloud Computing es un modelo para habilitar acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios. Este modelo de nube promueve la disponibilidad y está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.”*



**National Institute of Standards and Technology (NIST)**

# Cloud computing: características

- 1. Auto-servicio bajo demanda:** sin tener que comunicarse personalmente con el proveedor de servicios.
- 2. Escalabilidad y elasticidad:** añadir o eliminar recursos de cómputo.
- 3. Conjunto de recursos compartidos:** los recursos sirven a múltiples consumidores. El consumidor no posee control o conocimiento sobre la ubicación exacta de los recursos.

hay normativas, por ejemplo para datos medicos

# Cloud computing: características

- 4. Acceso a través de Internet:** los servicios son entregados a través de internet utilizando mecanismos y protocolos estándar.
- 5. Modelo de pago por uso:** los servicios son monitorizados a través de métricas que permiten el establecimiento de diferentes modelos de pago.

# Cloud computing: pilares

- 1. Sistemas distribuidos:** permiten la gestión de grandes despliegues descentralizados, aumentando la escalabilidad de los sistemas
- 2. Virtualización:** concepto clave que ha permitido el desarrollo flexible de plataformas software sobre un hardware común:
  - Mejora en la utilización de los recursos hardware
  - Ahorro de espacio físico
  - Ahorro energético y de costes de gestión
  - Necesidad de arquitecturas flexibles
- 3. WEB 2.0:** habilita el acceso fácil a estos servicios para los usuarios finales. La mayoría de estos servicios puedes ser gestionados vía web

# Cloud computing: virtualización

- Simulación de **máquinas virtuales**.
- Se lleva a cabo en una plataforma de hardware mediante un **software "host"**(que es un programa de control) que simula un entorno computacional (máquina virtual) para su **software "guest"**
- Este *guest*, que generalmente es un SO completo, corre como si estuviera instalado en una plataforma HW autónoma.

# Cloud computing: virtualización

- Muchas máquinas virtuales son simuladas en una máquina física dada.
- Para que el sistema operativo guest funcione, la simulación debe ser lo suficientemente grande como para soportar todas las interfaces externas de los sistemas guest, las cuales pueden incluir los drivers de hardware.

# Cloud computing: virtualización

- **Host:** anfitrión en un entorno virtualizado.  
Como ejemplo concreto podemos pensar en un **equipo físico** con un sistema operativo donde se crean máquinas virtuales.
- **Guest:** el recurso virtual acogido en el host.  
Típicamente la maquina virtual.
- **Hipervisor o VMM(Virtual Machine Monitor/Manager):** SW donde reside la emulación y gestión de las maquinas virtuales.

SW = software

# Cloud computing: virtualización

- **Hipervisor o VMM (virtual machine monitor):**
  - Capa entre el SO y el HW. Proporciona los servicios necesarios para que puedan ejecutarse varios SO sin conflictos. (todo de forma anónima)
  - Gestiona las colas y las instrucciones de/hacia el HW.
  - Se implementa como código embebido en el firmware del sistema o como una capa software.

# Tipos de hipervisor

- **Hipervisor Tipo 1 o Hipervisor Nativo (bare-metal):**

bare metal porque se ejecuta directamente sobre el HW

- Se ejecuta directamente sobre el HW del host
- No necesita un SO de base importante !!! lo hace más ligero
- Acceso directo a los recursos HW también importante !!
- Ejemplos: VMware ESXi, Citrix XenServer and Microsoft Hyper-V hypervisor

# Tipos de hipervisor

- **Hipervisor Tipo 1 (Pros & Cons):**

eficientes porque ejecutan directo en el HW y tienen acceso directo a los recursos

- + Muy eficientes (acceso directo a la CPU, memoria, red, almacenamiento)

- + Seguros porque no hay SO de por medio

- Suelen necesitar una máquina dedicada para gestionar varias MV y controlar el acceso a los recursos HW del host

(no virtualizada)

# Tipos de hipervisor

las que usaremos en las prácticas

- **Hipervisor Tipo 2 o *hosted hypervisor*:**
  - Se ejecuta en el SO del host
  - No se ejecutan directamente sobre el HW sino como una *aplicación* en un determinado host
  - El hipervisor solicita al SO que haga las llamadas al HW.
  - El host prioriza sus propias funciones
  - Ejemplos: PCs y analistas de seguridad

# Tipos de hipervisor

- **Hipervisor Tipo 2 (Pros & Cons):**
  - + Acceso rápido y sencillo al SO del guest
  - + Incluyen herramientas adicionales para el guest
  - No hay acceso directo a los recursos HW, por lo que son menos eficientes que los Tipo 1.
  - Menor seguridad ya que si se accede al SO del host, también se puede hacerlo al del guest.

# Tipos de hipervisor

Criteria	Type 1 hypervisor	Type 2 hypervisor
AKA	Bare-metal or Native	Hosted
Definition	Runs directly on the system with VMs running on them	Runs on a conventional Operating System
Virtualization	Hardware Virtualization	OS Virtualization
Operation	Guest OS and applications run on the hypervisor	Runs as an application on the host OS
Scalability	Better Scalability	Not so much, because of its reliance on the underlying OS.
Setup/Installation	Simple, as long as you have the necessary hardware support	Lot simpler setup, as you already have an Operating System.
System Independence	Has direct access to hardware along with virtual machines it hosts	Are not allowed to directly access the host hardware and its resources
Speed	Faster	Slower because of the system's dependency
Performance	Higher-performance as there's no middle layer	Comparatively has reduced performance rate as it runs with extra overhead
Security	More Secure	Less Secure, as any problem in the base operating system affects the entire system including the protected Hypervisor
Examples	<ul style="list-style-type: none"><li>VMware ESXi</li><li>Microsoft Hyper-V</li><li>Citrix XenServer</li></ul>	<ul style="list-style-type: none"><li>VMware Workstation Player</li><li>Microsoft Virtual PC</li><li>Sun's VirtualBox</li></ul>

Source: <https://www.hitechnectar.com/blogs/hypervisor-type-1-vs-type-2/>

# Tipos de virtualización

no hay uno asociado a hip de tipo 1 y otro a tipo 2

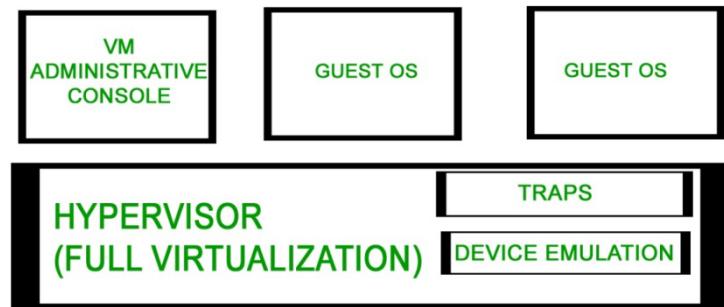
- **Paravirtualización:** relacionado sobre todo con los hipervisores de tipo 1
  - El HW no es simulado
  - El SO del guest **se modifica y recompila** antes de su instalación el la MV, lo que mejora su rendimiento ya que de esta forma se comunica directamente con el hipervisor.
  - Eliminar la necesidad de realizar traducciones continuas entre los recursos físicos y virtuales

# Tipos de virtualización

- **Para-virtualización (Pros & Cons):**
  - + Sencillo
  - + Buen rendimiento
  - Requiere la modificación del SO del guest

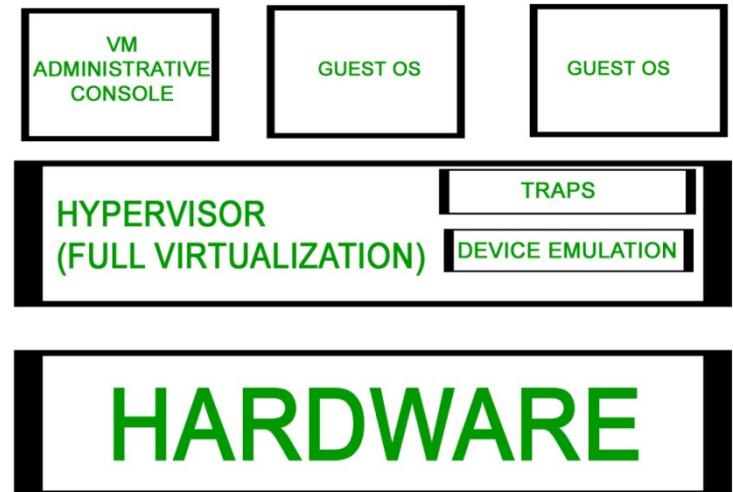
# Tipos de virtualización

- **Virtualización completa**
  - Similar a la paravirtualización  
puede permitir tambien acceso al HW
  - Puede emular el HW subyacente de ser necesario
  - El hipervisor gestiona las operaciones máquina, las emula en SW y devuelve los códigos de status de forma consistente a como lo haría el HW real.
  - No se modifica el SO del guest



# Tipos de virtualización

- **Virtualización completa (Pros & Cons)**
  - + No se modifica el SO
  - Complejo
  - Lento (emulación)



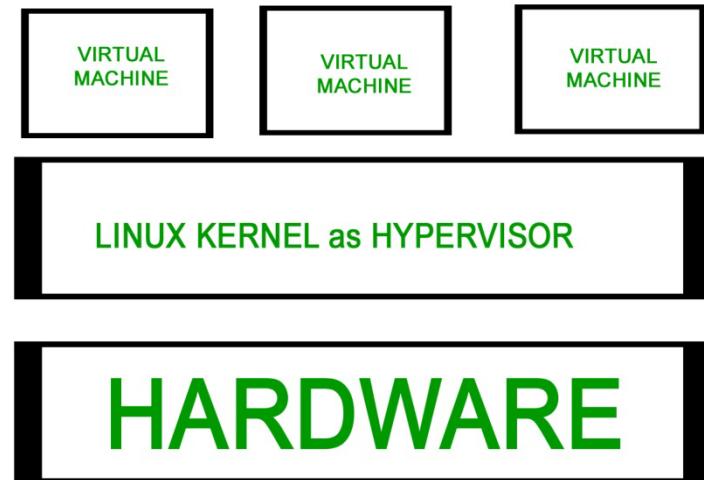
# Tipos de virtualización

estrategias intermedias, las anteriores son las dos grandes

- **Virtualización asistida por HW**
  - Gran parte de las operaciones de emulación de E/S e instrucciones de estatus ejecutadas en el SO del guest se apoyan en extensiones HW.
  - Permite usar SO no modificados
  - Ejemplos: V Pacifica and Intel VT Vanderpool proporcionan HW para dar soporte a la virtualización.

# Tipos de virtualización

- **Virtualización a nivel de núcleo:**
- En vez de un hipervisor, ejecuta un *kernel* Linux y trata a la MV asociada como un proceso de usuario del host
- Necesita ayuda del procesador
- Driver específico para la comunicación kernel/MV



# Modelos de servicio: IaaS

mas bajo nivel

- **IaaS (infraestructura como servicio)**

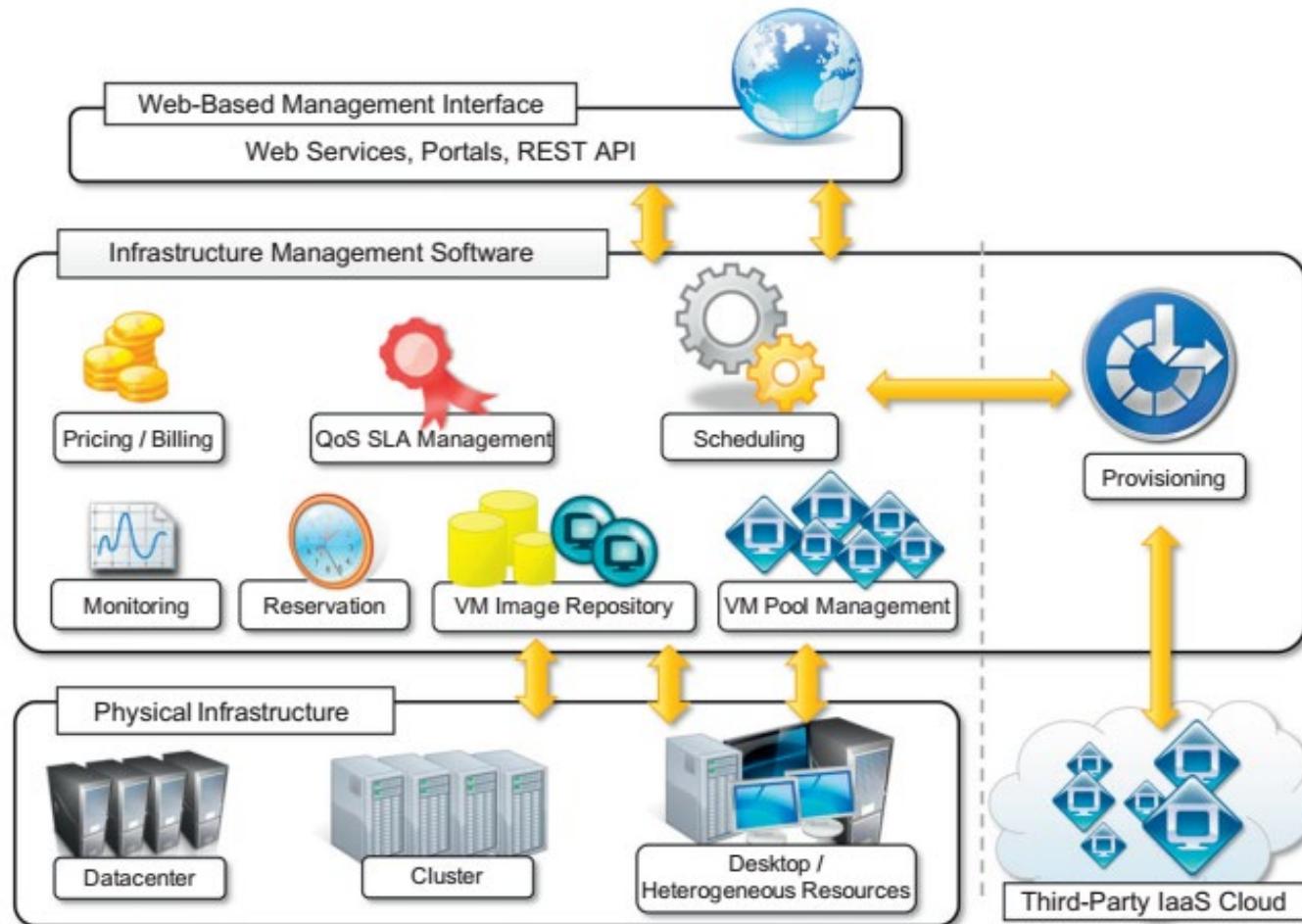
Proporciona la *infraestructura* informática, física u otros recursos como máquinas virtuales, imágenes de disco, almacenamiento basado en archivos, *firewalls*,平衡adores de carga, direcciones IP, redes de área local virtuales, etc.



# Modelos de servicio: IaaS

- El proveedor ofrece una infraestructura escalable y compartida a diferentes clientes
- Los clientes pueden obtener servicios de infraestructura computacional bajo demanda.
- El cliente puede ajustar la infraestructura a sus necesidades.
- Ejs: web hosting, computación alto rendimiento, infraestructura de pruebas, desarrollo y producción, almacenamiento en la nube

# Modelos de servicio: IaaS



Ejemplos: Amazon EC2, OpenSource: OpenNebula, OpenStack, Vmware cloud, etc.

# Modelos de servicio: PaaS

- **PaaS (plataforma como servicio)**

Proporciona las *plataformas* de cómputo que normalmente incluye sistema operativo, lenguaje de programación del entorno de ejecución, base de datos, servidor web, etc.

En este caso los desarrolladores se encargan de desarrollar las propias aplicaciones que se ejecutan en la nube.

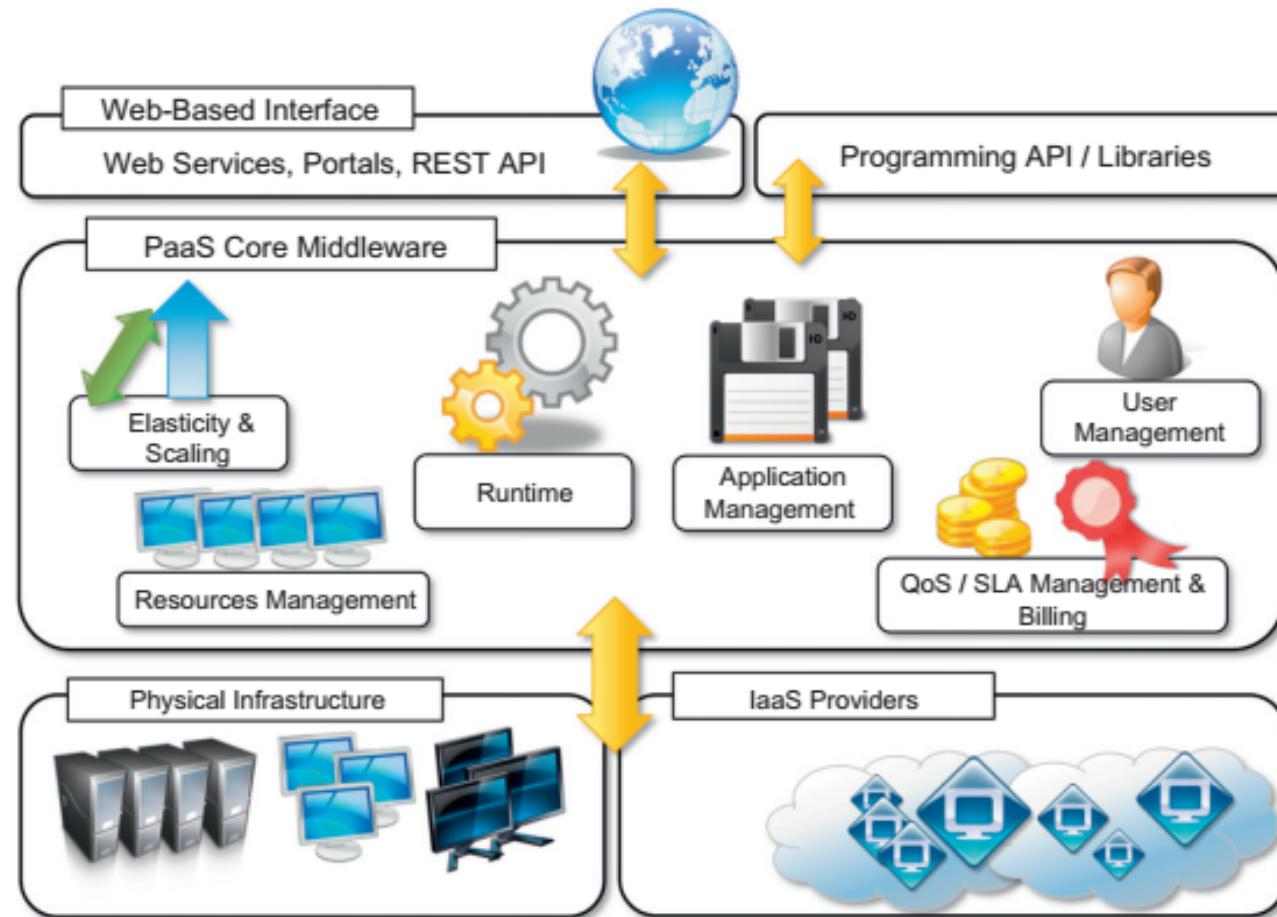


# Modelos de servicio: PaaS

capa intermedia

- El proveedor de servicios ofrece el middleware requerido para el desarrollo y la ejecución de aplicaciones.
- No hay que preocuparse de la infraestructura.
- Se proveen APIs y Frameworks para el desarrollo de sistemas
- Ejemplos de uso: plataformas de desarrollo en la nube, entornos de programación y compilación on-line

# Modelos de servicio (PaaS)



Ejemplos: Google AppEngine, Microsoft Azure, IBM Cloud (Bluemix)

# Modelos de servicio: SaaS

mas alto nivel

- **SaaS (software como servicio)**

Se trata de cualquier *servicio* basado en la web. En este tipo de servicios se accede a través del navegador sin atender al software. Todo el desarrollo, mantenimiento, actualizaciones, copias de seguridad es responsabilidad del proveedor. El usuario paga por el uso, por la infraestructura necesaria.

# Modelos de servicio: SaaS

- El software es desarrollado y gestionado remotamente por un proveedor de servicios.
- Todo está en la nube. Los clientes no tienen que realizar operaciones en su infraestructura.
- El pago se realiza por suscripción (mensual, anual, etc.) o a través de métricas de uso.
- Ejemplos: redes sociales, editores on-line, aplicaciones científicas, etc.

# Modelos de servicio: SaaS

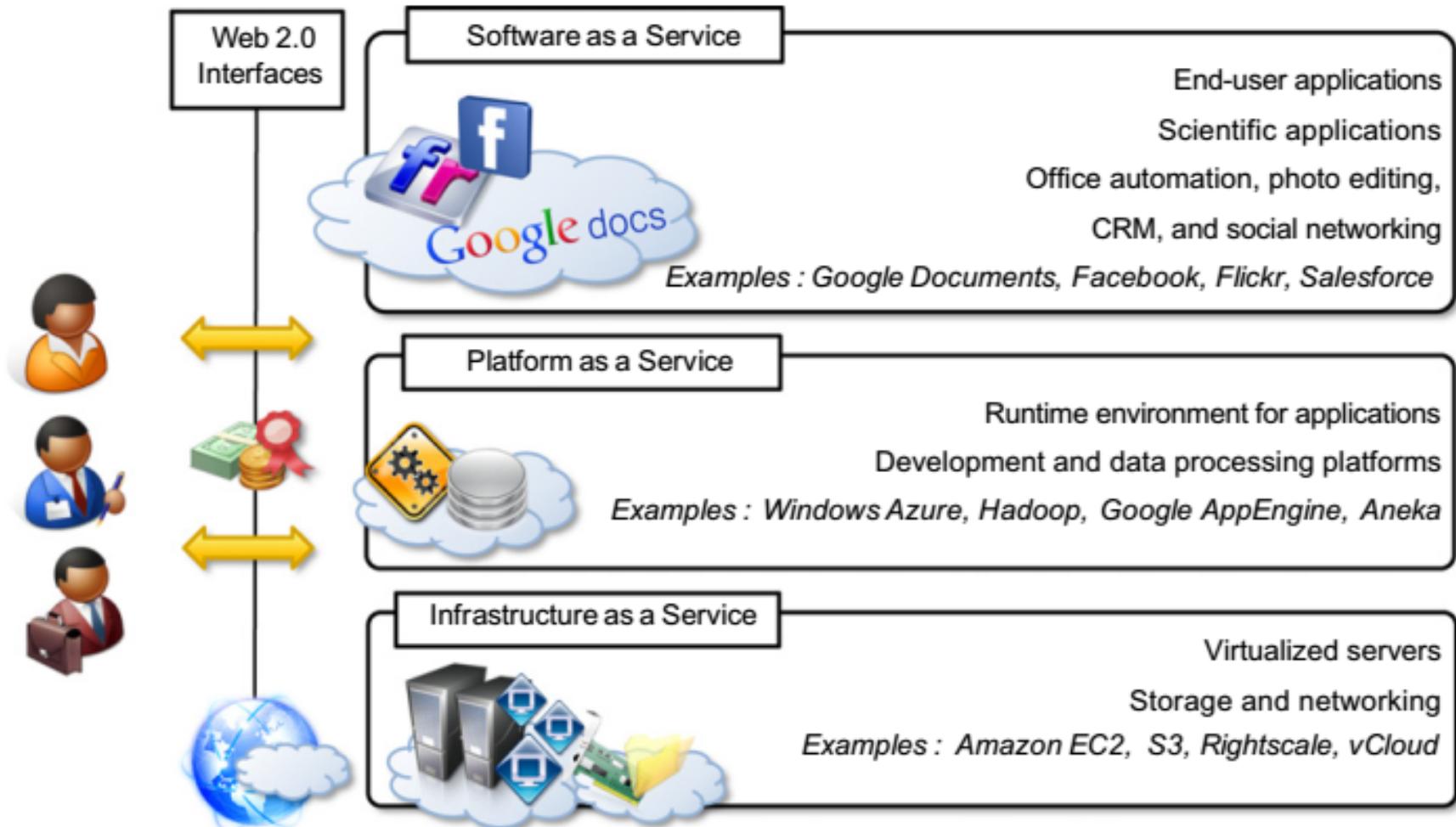


Google™ Apps

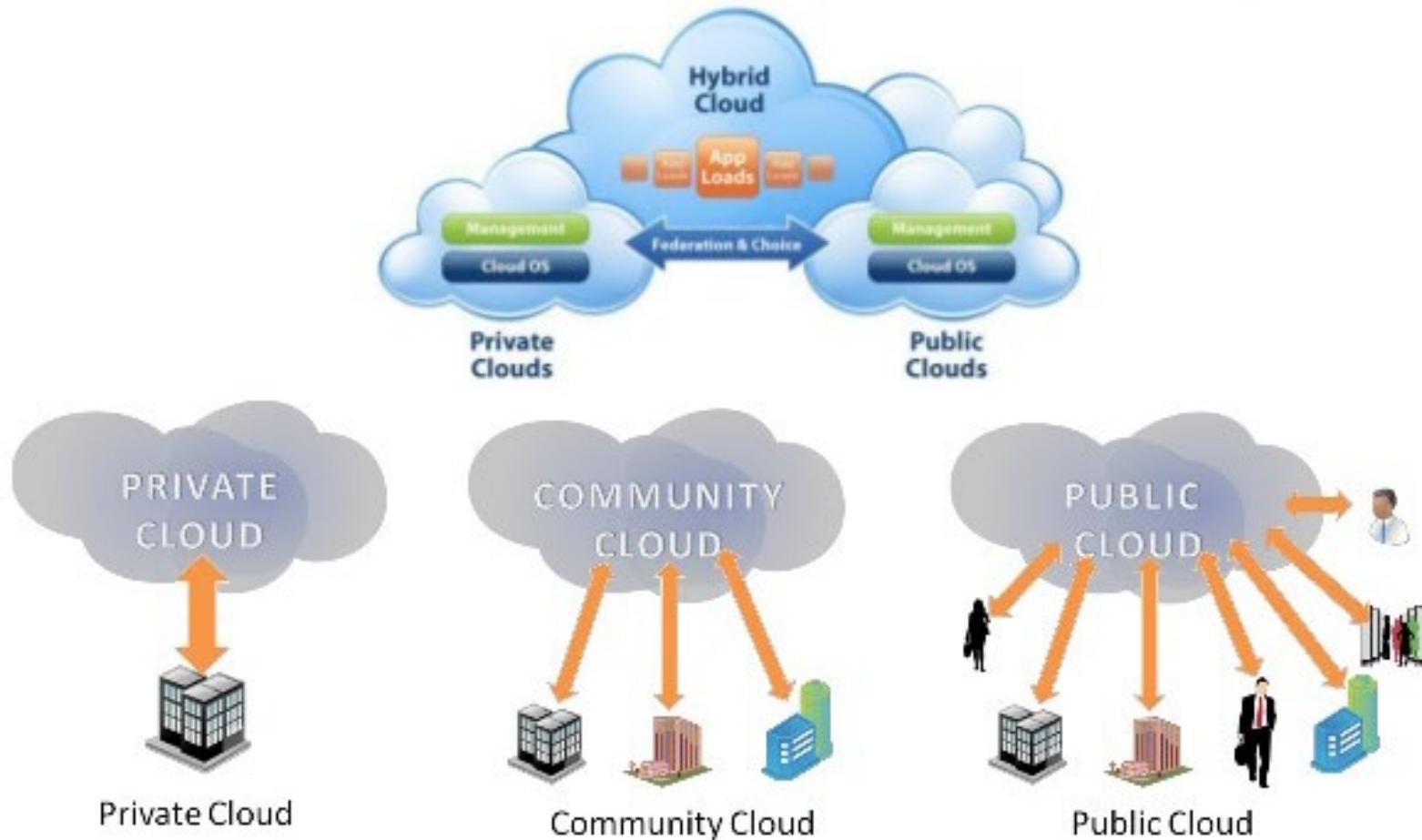


Ejemplos:  
Google apps  
Web-mail  
Overleaf  
Office 365

# Modelos de servicio



# Modelos de despliegue



# Nube pública

- Grandes infraestructuras para soportar un gran número de usuarios.
- **Disponible al público** en cualquier momento.
- Cualquier organización puede hacer uso de los servicios de forma **rápida**.
- Formadas por multitud de servidores geográficamente distribuidos
- **Multi-usuario:** necesario el aislamiento de recursos

# Nube privada

- Privacidad de los datos
- La infraestructura es operada por una sola organización
- Servicios están disponibles a **usuarios internos** de la organización.
- Sólo departamentos internos pueden hacer uso de los servicios de una manera rápida
- Necesidad de desarrollar estándares de utilización

# Nube comunitaria

- La infraestructura soporta los requerimientos de varias organizaciones.
- Los servicios están disponibles a los usuarios de las organizaciones participantes.
- Sólo las organizaciones participantes pueden hacer uso de los servicios de una manera rápida (coste medio de configuración inicial).
- Las organizaciones participantes pueden aprovechar los recursos de las demás.

# Nube híbrida

- Combinación de los modelos anteriores.
- Infraestructura operada por organizaciones externas (Public Cloud) e internas (Private Cloud).
- Las organizaciones pueden mantener su infraestructura privada y, de requerir más recursos, utilizan infraestructuras públicas.
- Los datos sensibles pueden mantenerse en la red privada.

# Cloud computing: ventajas

- ✓ **Bajo coste.** No hay necesidad de adquirir un hardware y un software determinado
- ✓ **Bajos requerimientos en los dispositivos de acceso,** ya que se accede a los servicios a través de interfaz web.
- ✓ **Mejora del rendimiento.** Los ordenadores que utilicen aplicaciones en la nube estarán menos sobrecargados porque tienen un menor procesado a nivel local.
- ✓ **Flexibilidad:** El servicio de nube se paga de acuerdo a la demanda.
- ✓ **Movilidad:** La información al quedar alojada en la nube pueden ser consultada por el usuario desde cualquier lugar

# Cloud computing: ventajas

- ✓ **Mejor utilización de recursos hardware.** Los elementos que forman la nube son explotados de forma más exhaustiva.
- ✓ **Reducción del consumo energético** al haber menos equipamiento hardware por separado
- ✓ **Menor tiempo de despliegue** de aplicaciones
- ✓ **Facilidad para agregar y eliminar servicios**, al estar todos éstos virtualizados
- ✓ Acceso a toda la **información en tiempo real**
- ✓ Las **actualizaciones de software** son **instantáneas y transparentes** al usuario final
- ✓ Capacidad de **almacenamiento casi ilimitada**

# Cloud computing: desventajas

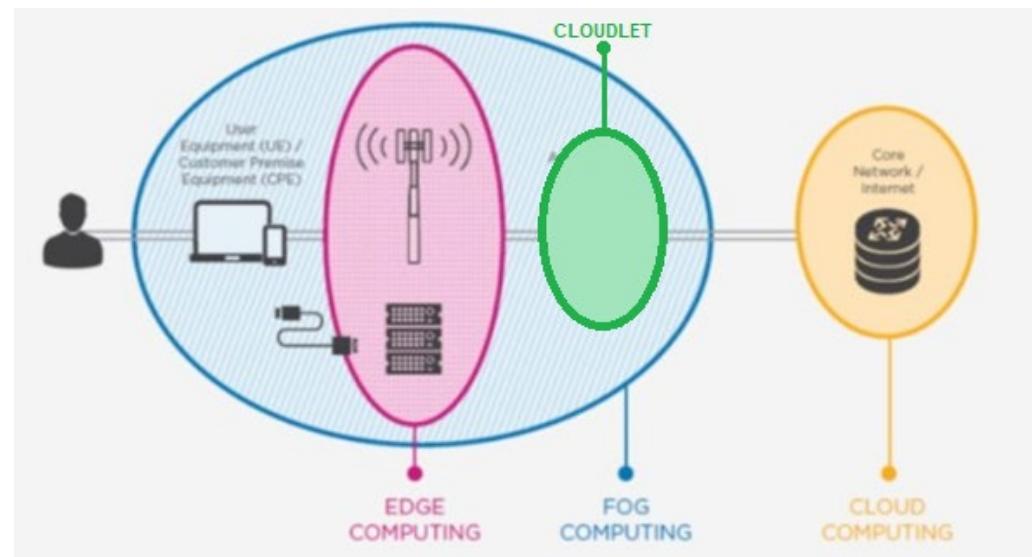
- ✓ **Recursos compartidos:** QoS (Quality of Service)
- ✓ **Falta de estándares entre plataformas:** dificultad de migración
- ✓ **Software limitado** al ofrecido por la plataforma
- ✓ Muy **dependientes** del acceso a la red
- ✓ Riesgos de **seguridad** y **privacidad** de los datos
- ✓ **Altas latencias** en la recuperación de los datos
- ✓ **Servicio centralizado** en la nube

# Cloud computing: alternativas

- Nuevas aplicaciones con nuevos requisitos:
  - Baja latencia
  - Alto ancho de banda
  - Servicios contextualizados y dependientes de la localización
- Los servicios cloud presentan problemas para manejar estos requerimientos:
  - Camino muy largo desde el usuario final hasta los servidores (**alta latencia**)
  - Posible colapso del backbone
  - Dificultad para proveer servicios contextualizados por el gran volumen y precisión de la información necesaria para esta contextualización

# Cloud computing: alternativas

- Tres alternativas:
  - Cloudlets
  - Fog Computing
  - Mobile Edge Computing (MEC)
- Aproximan la computación y el almacenamiento al usuario



# Cloudlets

- Propone instalar ***datacenters dedicados*** más próximos al usuario final
- Capa de procesado entre los dispositivos finales y el cloud: requiere de hardware y software específico: por ejemplo OpenStack++
- No ha tenido gran repercusión por la complejidad del despliegue

# Fog Computing

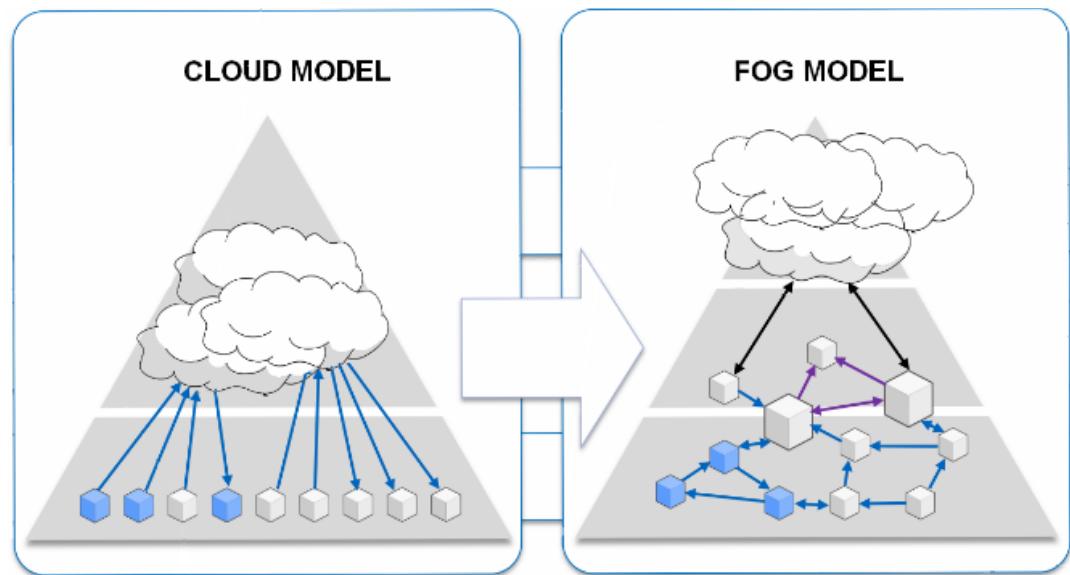
- Propone situar la nube más cerca de los usuarios finales
- En un principio, el fog se situaba en los dispositivos finales (móviles, portátiles, dispositivos IoT con capacidad de procesado, etc.)
- Actualmente, el fog se entiende como toda la infraestructura comprendida entre los dispositivos finales y la red troncal

# Fog Computing

- Los datos, procesamiento y aplicaciones se concentran en los dispositivos de usuario en lugar de existir casi en su totalidad en la nube.
- Computación distribuida de proximidad. Cada uno de los dispositivos conectados a la red puede procesar los datos y sólo transmitir un resumen al siguiente nivel, o hacerlo sólo en determinados casos.

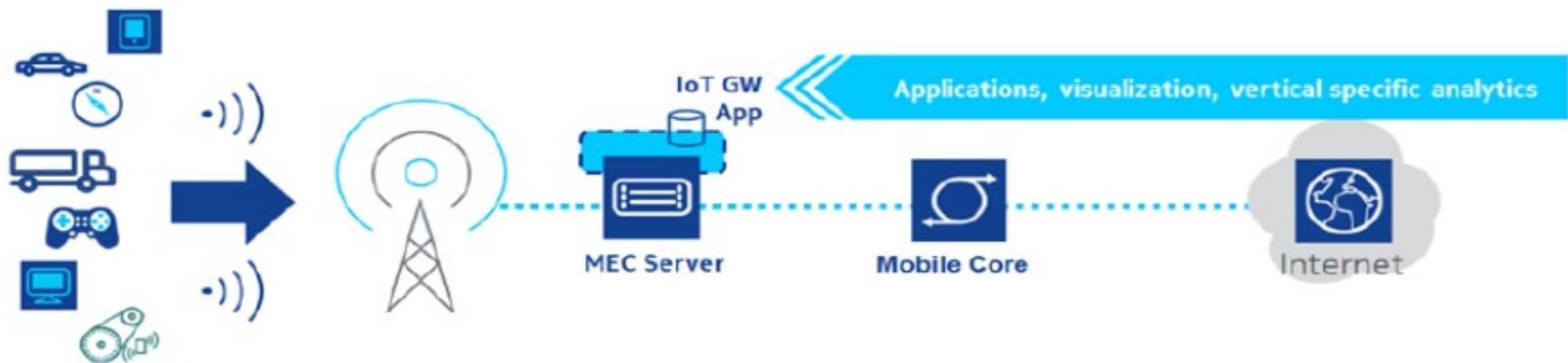
# Fog Computing

- ✓ Baja latencia y conciencia de la ubicación
- ✓ Heterogeneidad
- ✓ Movilidad
- ✓ Alta cantidad de nodos
- ✓ Acceso inalámbrico predominante
- ✓ Amplia distribución geográfica
- ✓ Fuerte presencia de streaming y aplicaciones en tiempo real



# Mobile Edge Computing (MEC)

- Procesamiento en quasi-tiempo real de grandes volúmenes de datos
- Elementos de procesado y almacenamiento situados **exclusivamente** en las estaciones base
- Ligado al desarrollo de 5G, por la baja latencia necesaria (<1ms)



# Mobile Edge Computing (MEC)

- Características de funcionamiento similares a la de *fog computing* con una menor heterogeneidad de los dispositivos involucrados
- Acceso a las métricas de nivel físico de los nodos conectados a la estación base, lo que permite tomar decisiones en la configuración de las comunicaciones
- Otra acepción: Multi-Access Edge Computing
  - En 5G se consideran múltiples tecnologías de acceso simultáneas
  - 5G-RAN. Nodos RAN - desde torretas o antenas a pequeñas infraestructuras. Cloud-RAN (C-RAN): Centros de datos locales que sirven a varios nodos

# Mobile Edge Computing (MEC)

- Seguridad y videovigilancia
- Realidad virtual y aumentada, gaming
- Vehículo autónomos
- Redes 5G

**MEC = Baja latencia + alto ancho de banda**  
**MEC =  $\Sigma$  (movilidad + servicios cloud + edge computing)**

# Fog & MEC vs Cloud

<b>FOG &amp; MEC</b>	<b>CLOUD</b>
Almacenamiento y procesamiento local	Almacenamiento y procesamiento lejano
Servicios alojados en los dispositivos cercanos al borde de la red	Los servicios se alojan en servidores virtuales
Compatible con aplicaciones de IoT que demandan tiempo real o latencia predecible	Compatible con los servidores de hardware, aplicaciones y casi cualquier tipo de datos
Se basa en el principio de aislamiento de datos del usuario que se alojan en el borde de la red	Todos los datos se centralizan en uno o más centros de datos

# Plataformas IoT



# Google Cloud Platform



- Capacidades de aprendizaje automático para cualquier necesidad de IoT.
- Información empresarial en tiempo real para dispositivos dispersos por todo el mundo.
- Capacidades de IA.
- Brinda soporte para una amplia gama de sistemas operativos integrados.
- Inteligencia de ubicación.

# Salesforce IoT Cloud



- Gestión de relaciones con los clientes para todo tipo de empresas
- Proporciona datos reales sobre el uso y el rendimiento del producto.
- Datos de cualquier dispositivo.
- Con la API RESTful, puede importar datos de cualquier fuente.
- Vista de tráfico en tiempo real.



# ThingWorx

- Industrial IoT (IIoT)
- Análisis predictivo de datos
- Supervisión y mantenimiento remotos
- Acceso a los datos de aplicaciones desde servidores web locales
- Aplicaciones en la nube y entornos híbridos.



# IBM Watson IoT

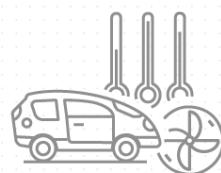
- Analítica basada en IA
- REST y API en tiempo real
- Conexión MQTT y HTTP
- Soluciones flexibles
- Seguridad
- Captura datos en tiempo real



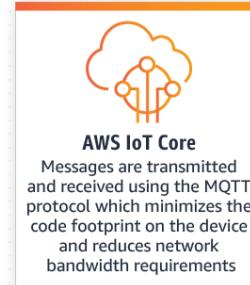
AWS IoT

# Amazon AWS IoT Core

- Procesa una gran cantidad de mensajes
- Plataforma confiable
- Compatible con otros servicios de AWS
- Acceso seguro a los dispositivos



**Devices publish & subscribe**  
Billions of devices can publish and subscribe to messages



**Devices communicate**  
AWS IoT Core enables devices to communicate with AWS services and each other



# Microsoft Azure IoT

- Plataforma abierta
- Principiantes y expertos
- Azure IoT Hub:
  - comunicación entre la aplicación de IoT y los dispositivos que ésta administra.
- Azure Digital Twins
  - modelos digitales para la mejora de productos y optimización de procesos



# Oracle IoT

- Permite conectar un dispositivo a JavaScript, Android, iOS, Java y C POSIX.
- Virtualización de dispositivos, mensajería de alta velocidad y administración de terminales
- Análisis, procesamiento y enriquecimiento de datos.
- API REST para la integración con aplicaciones de Oracle y dispositivos IoT.

# Plataformas IoT

	Google	IBM	ThingWorx	Azure	Thing Speak	Zetta	Yaler	Amazon	Axeda
Escalabilidad	✓	✓	✓	✓	✓	✗	✓	✓	✓
Seguridad y privacidad	✓	✓	✓	✓	✗	✗	✗	✓	✓
Plug&Play	✓	✓	✓	✓	✗	✗	✗	✓	✓
Soporte millones de dispositivos	✓	✓	✓	✓	✗	✗	✗	✓	✓
Datos en tiempo real	✓	✓	✓	✓	✓	✗	✗	✓	✓
Almacenamiento de datos	✓	✓	✓	✓	✗	✗	✗	✓	✓
Soporte técnico	✓	✓	✓	✓	✓	✗	✓	✓	✓
Tipo de solución	PaaS	PaaS	Complete IoT	PaaS	Data Analytics	API to devices	-	IaaS	SaaS

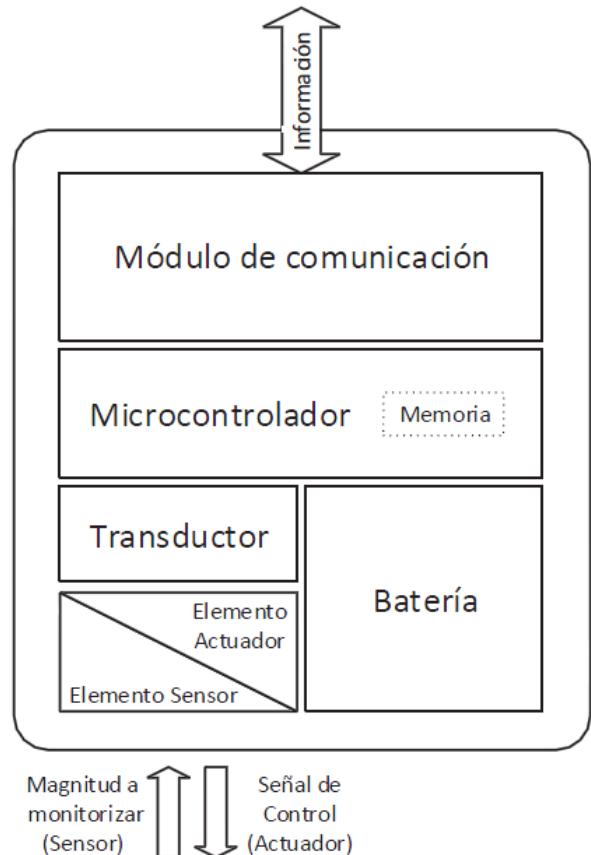
Fuente: Bhumi Nakhuva and Prof. Tushar Champaneria, Department of Computer Science and Engineering, L. D. College of Engineering, India.

International Journal of Computer Science & Engineering Survey (IJCSES) Vol.6, No.6, December 2015

## **BLOQUE II: Dispositivos y soluciones para recogida de información de sensores**

# Arquitectura funcional de un nodo sensor/actuador

- La arquitectura de un elemento sensor o actuador se puede dividir en tres bloques funcionales:
  - Bloque sensor/actuador
  - Bloque de procesado
  - Bloque de comunicación

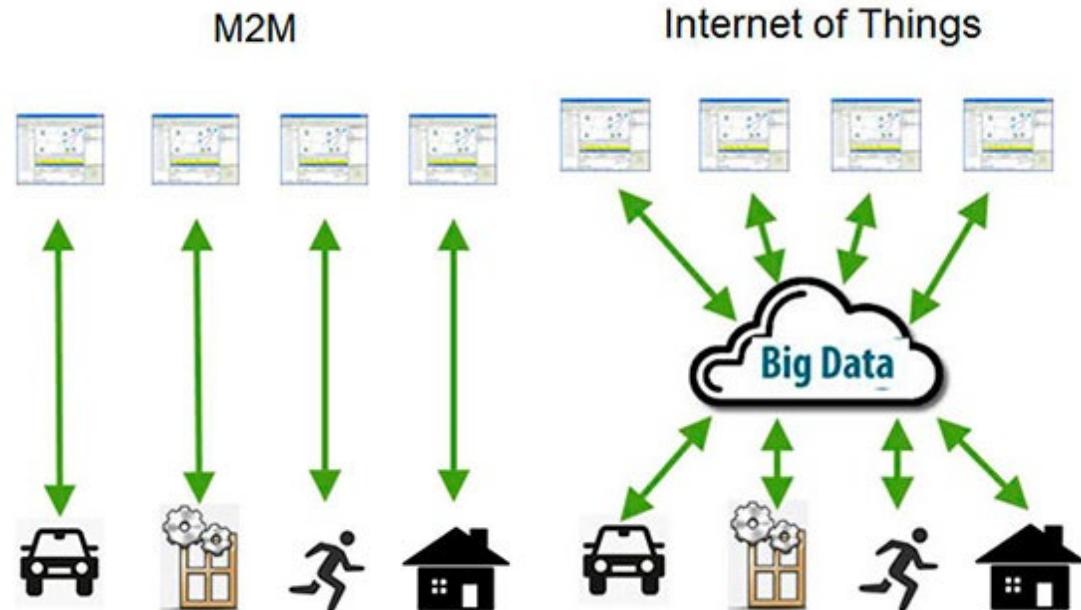


# M2M

- M2M (Machine-to-Machine): tecnologías que permiten la comunicación entre dispositivos
- No señala una tecnología específica ni en el ámbito de las redes, ni en el de la información o las comunicaciones. Se utiliza para describir cualquier tecnología que permita a diferentes dispositivos conectados intercambiar información a través de redes fijas o móviles de corto o largo alcance, y ejecutar acciones sin la intervención manual de seres humanos.

# M2M vs IoT

- En M2M los dispositivos solo envían la información que recolectan
- En IoT envían, reciben, **procesan** información y toman acciones con (o sin) la ayuda de soluciones Big Data.



	 M2M	 IoT
Se aplica en	Máquinas	Sensores
Basado en	Hardware	Software
Aplicaciones	Verticales	Horizontales
Tipo de instalación	Sistemas cerrados	Grandes redes
Tipo de comunicación	Máquina-a-máquina	Máquina-a-máquina, humanos-a-máquinas y máquinas-a-humanos
Tipo de protocolo	No IP	IP
¿Usa la nube?	Puede, pero no es imprescindible	Sí
Singularidad	Comunicación punto-a-punto con hardware embébido	Redes IP
Dirección de la comunicación	Normalmente, unidireccional	Bidireccional
Propósito principal	Monitorización y control	Múltiples aplicaciones
Modo de operación	Desencadena respuestas basadas en una acción	Puede desencadenar respuestas, pero no es imprescindible
Capacidad de integración	Opciones limitadas (los dispositivos deben tener estándares complementarios)	Opciones ilimitadas vía software
Tipos de datos	Estructurados	Estructurados y no estructurados
Necesita conexión a internet	No	Sí

# Gateways en una arquitectura IoT (I)

- Los sensores y dispositivos tienen **capacidades de interconexión limitadas**, utilizando protocolos de bajo consumo (Bluetooth Low Energy, Zigbee, LoRaWAN, Sigfox...) para conectarse a redes de área local (LAN) o residencial (HAN). Es decir, no tienen acceso directo a Internet

Es necesario un **Gateway** para coordinar la red de sensores y actuar de proxy con la red donde la plataforma IoT expone sus interfaces para recibir y proporcionar información.

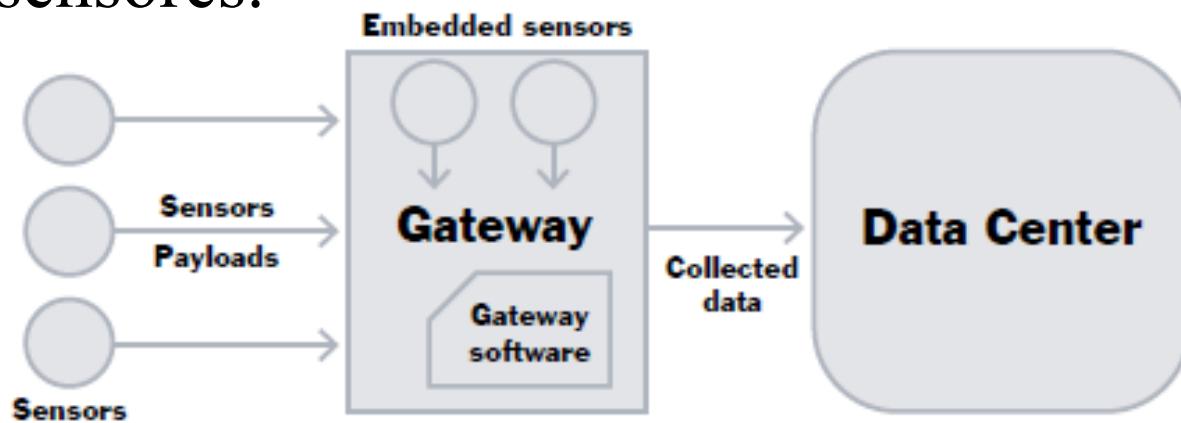
# Gateways en una arquitectura IoT (II)

- El tratamiento de la información en bruto procedente de sensores y dispositivos, directamente en la plataforma IoT, puede ser en ocasiones ineficiente en términos de rendimiento y ancho de banda.

**Es necesario disponer de Gateways con capacidad de almacenamiento temporal de la información (*caching*) y procesamiento de la misma. De este modo es posible filtrar y agregar la información antes de enviarla a la plataforma, así como garantizar que no se pierde información en caso de interrupción temporal de las comunicaciones.**

# Gateways en una arquitectura IoT (III)

- El Gateway centraliza las operaciones de monitorización de todos los dispositivos conectados a su red de sensores.



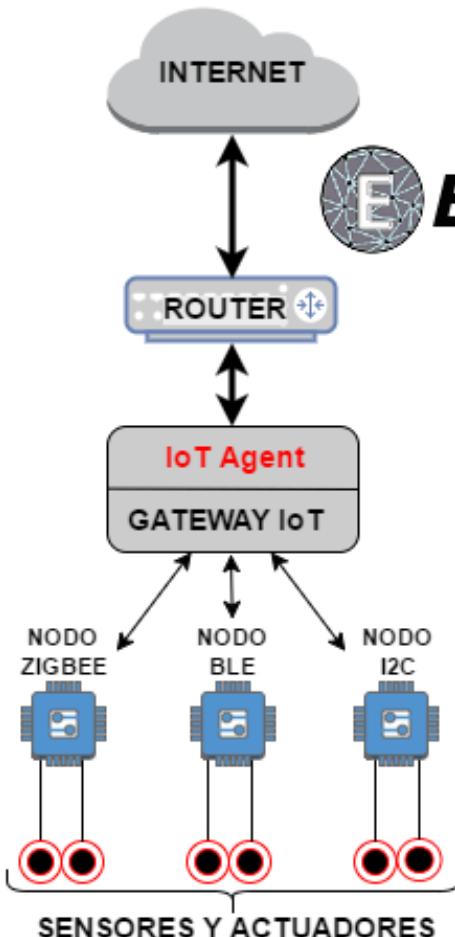
- Recuperación ante fallos: de la alimentación, de comunicaciones, en sensores o dispositivos...
- Soporte para actualizaciones automáticas
- Soporte para configuración

# Diferencia entre Gateway y rooter

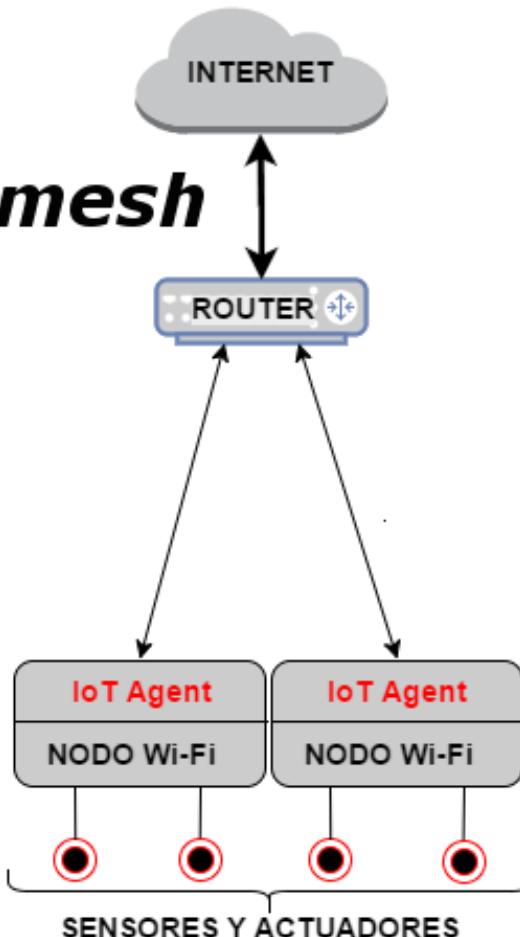
- Un *router* es un dispositivo de red que **integra dos o más redes**, a la vez que controla el tráfico de datos sobre la red externa global (Internet), permitiendo, por ejemplo, el control sobre los puertos de entrada y salida, y asegurando que los paquetes de datos viajen de manera correcta. En términos de redes, opera en la capa 3 (capa de red).
- Un *gateway* es cualquier **punto de conexión o nodo en una red** que provee acceso a otra a través de él mismo. Aunque puede utilizarse de la misma forma que el router para conducir el tráfico de una red, lo más habitual es usarlo como conexión de salida externa para comunicar entornos, protocolos y arquitecturas diferentes.
- La diferencia fundamental radica en que se emplean los gateways para gestionar el tráfico entre redes diferentes en cuanto a protocolos y arquitecturas, a diferencia de los routers que gestionan el tráfico entre redes similares.

# Gateways: Tipos de conexión

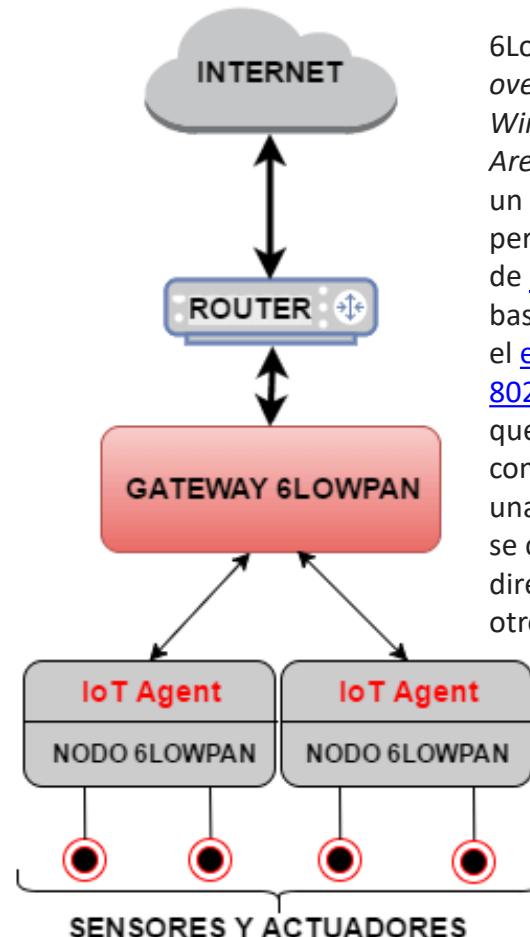
ESCENARIO 1



ESCENARIO 2



ESCENARIO 3



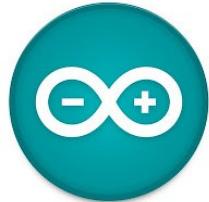
6LoWPAN ( *IPv6 over Low power Wireless Personal Area Networks* ) es un estándar que permite el uso de IPv6 en redes basadas en el [estándar IEEE 802.15.4](#). Permite que dispositivos como los nodos de una red inalámbrica se comuniquen directamente con otros dispositivos IP.

\* Fuente: Ermesh

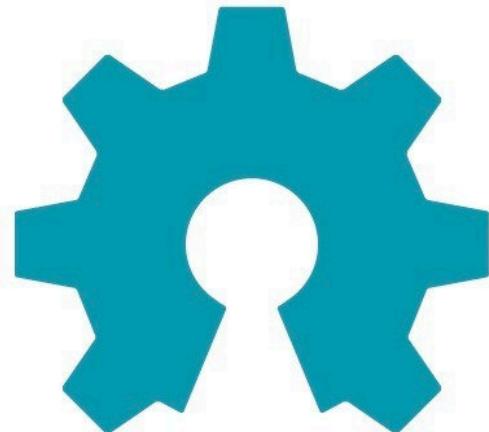
# Dispositivos para IoT: Arduino



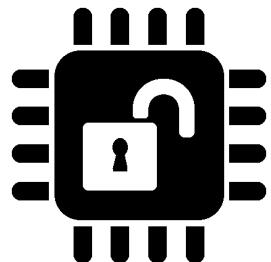
- Plataforma de creación de prototipos electrónicos de código abierto.
- El **hardware** consiste en una placa con un **microcontrolador y puertos de entrada/salida**.
- El **software** consiste en un entorno de desarrollo que implementa el lenguaje de programación **Processing/Wiring**.
- Ejecución de proyectos sin necesidad de conectar a un computador.



# Arduino



open source  
hardware

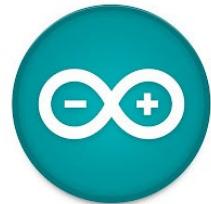


Al ser **Open-Hardware**, tanto su diseño, como su distribución es libre.

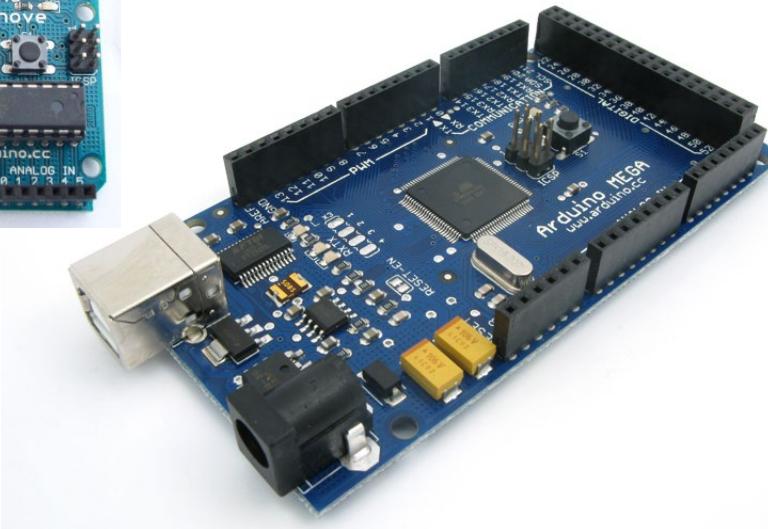
Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin necesidad de adquirir ninguna licencia

# Arduino

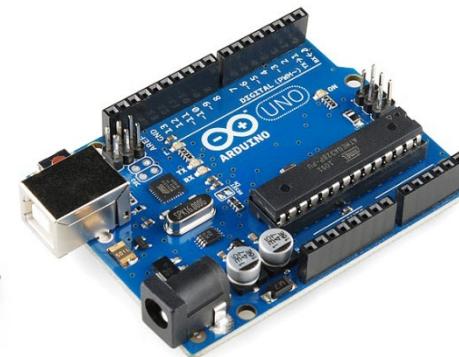
From 35€



Duemilanove



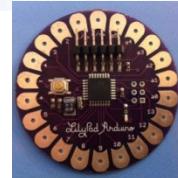
Mega



Uno



Nano



Lilypad



Mini

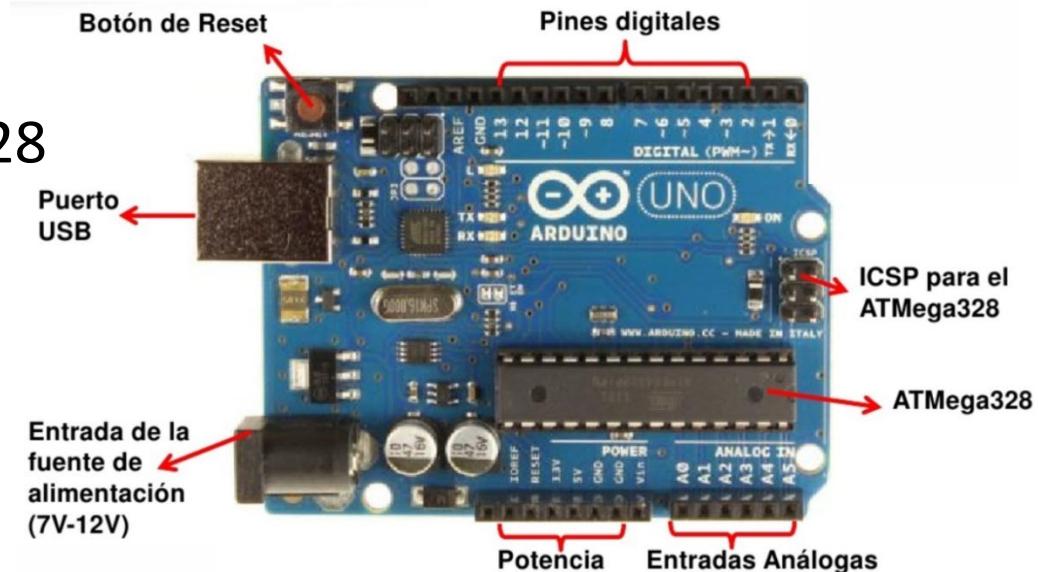


Leonardo



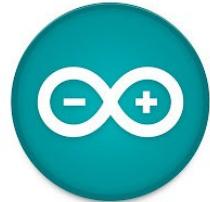
# Arduino Uno

- Microcontrolador: ATMega328
- Voltaje de operación: 5V DC
- Alimentación: 7 – 12V DC
- Pines digitales I/O: 14
- Entradas analógicas: 6
- Bootloader: SW alojado en la memoria flash que nos permite programar Arduino a través del puerto serie (USB) sin necesidad de usar un programador externo.
- Entrada ICSP (In Chip Serial Programmer): graba desde el PC al microcontrolador sin usar el puerto USB.



# Arduino Uno

- ATMega328:
  - Microcontrolador de 8-bits de bajo consumo
  - Arquitectura RISC: menos instrucciones (131), más rápido
  - 32Kbytes de memoria flash de programa
  - Bajo consumo. 6 modos en reposo: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby
    - Active mode: 1.5mA at 3V - 4MHz
    - Power-down mode: 1 $\mu$ A at 3V

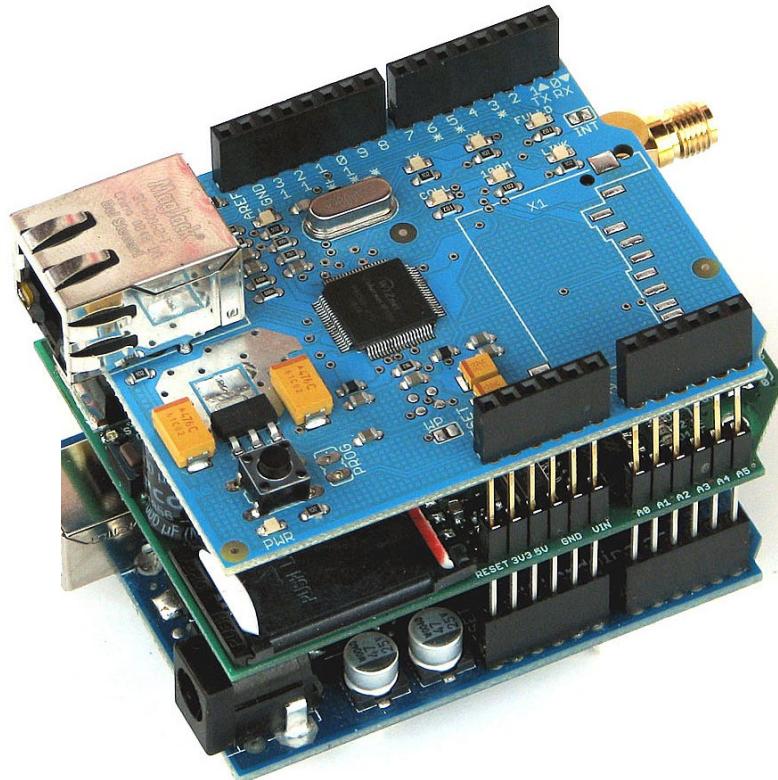


# Arduino Shields

- Expansión de funcionalidad



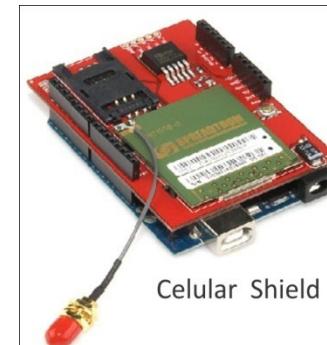
Motor Shield



GPS shield



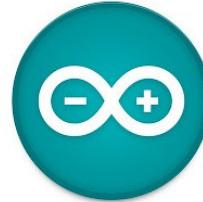
Ethernet shield



Cellular Shield



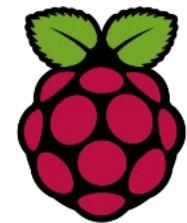
# Arduino para IoT



- **Arduino MKR1000**, diseñado específicamente para IoT.
- Microcontrolador **SAMD21 + ARM Cortex M0 de 32-bits** ( $9\mu\text{A}/\text{MHz}$ ) con 256Kb de memoria FLASH y 32Kb de SRAM.
- **Chip Wifi WINC1500** integrado de bajo consumo con el stack TCP/IP e IEEE 802.11 así como el modo cliente o servidor, DNS y DHCP, así como un **chip de encriptación por hardware ECC508** que permite realizar comunicaciones seguras.
- Conexión para batería LiPo de 3.7V con cargador USB integrado → dispositivo totalmente autónomo

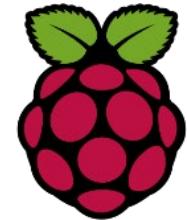
# Familia MKR Arduino para IoT

- Basada en microcontrolador SMAD21
- Menor factor de forma.
- Productos:
  - [Arduino MKR1000 WIFI](#)
  - [Arduino MKR WiFi 1010](#) : módulo ESP32 de U-BLOX para acelerar aplicaciones
  - [Arduino MKR FOX 1200](#): conectividad Sigfox.
  - [Arduino MKR WAN 1300](#): conectividad LoRA
  - [Arduino MKR NB 1500](#): Narrow Band IoT



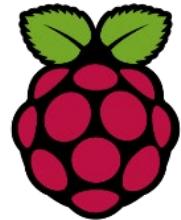
# Raspberry Pi

- Raspberry Pi comenzó a fabricarse en 2012 como ordenador de bajo coste para facilitar la enseñanza de la informática en los colegios.
- Placa base de pequeñas dimensiones en la que se aloja un chip Broadcom BCM2835 con procesador ARM hasta a 1 GHz de velocidad, GPU VideoCore IV y hasta 512 MB de memoria RAM.
- Se puede añadir teclado, ratón o cualquier dispositivo USB y conectarlo a un monitor. Disco duro conectado de forma externa, bien por USB o con una tarjeta SD.



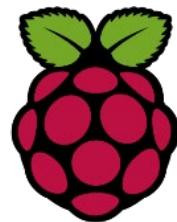
# Raspberry Pi: conexión

- **microUSB:** sistema de alimentación de la Raspberry.
- **GPIO:** 40 puertos, 26 se pueden usar como E/S.
- **HDMI:** conexión a TV u otro monitor. También se puede acceder en remoto desde otro PC, smartphone, etc.
- **USB:** La versión 2 B cuenta con 4 puertos USB
- **microSD.**
- **Ethernet:** para conexión de red.
- **Display DSI:** pequeñas pantallas táctiles que podemos acoplar a la Raspberry.
- **Audio 3,5 mm.**



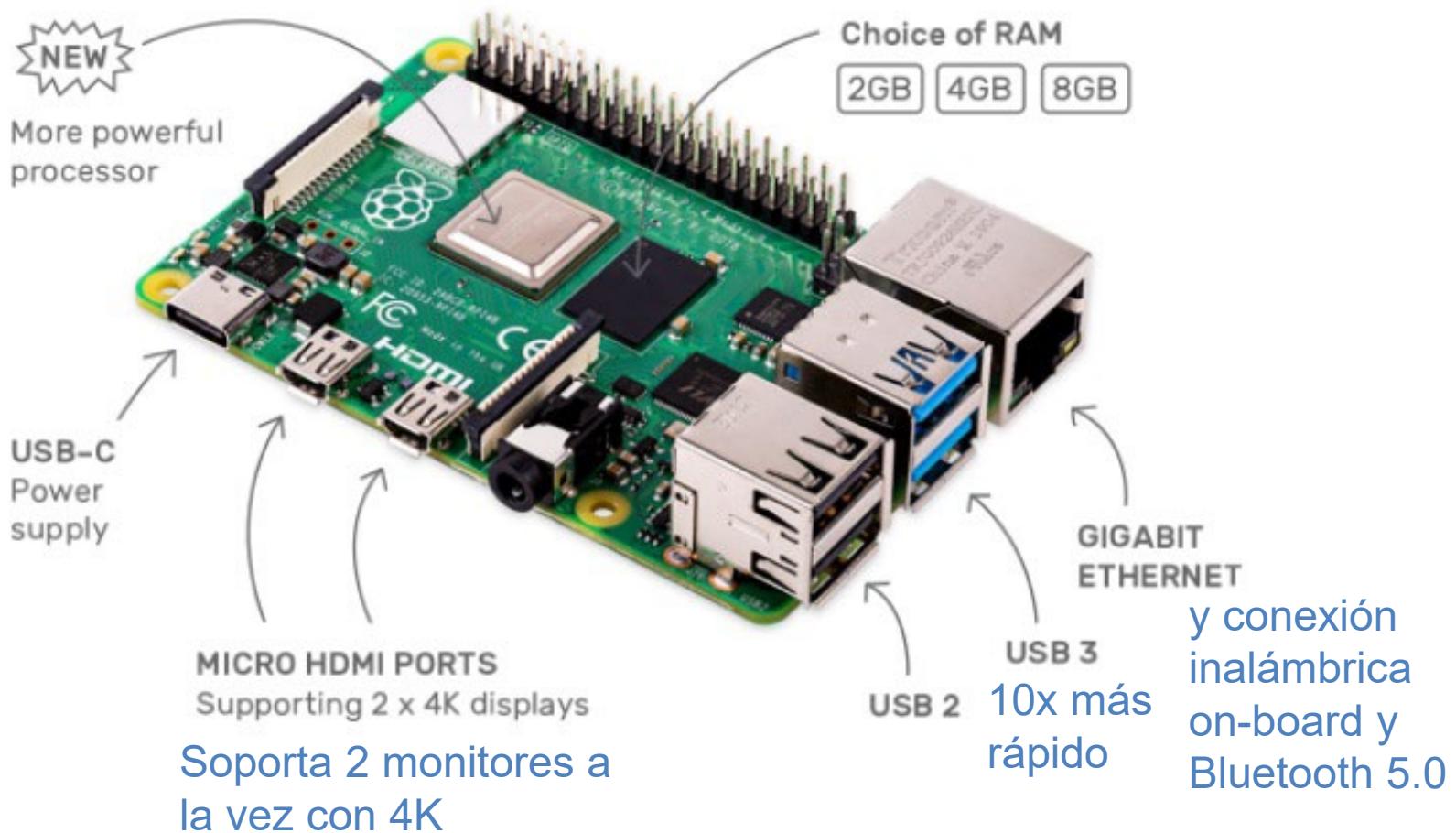
# Raspberry Pi: conexión





# Raspberry Pi 4

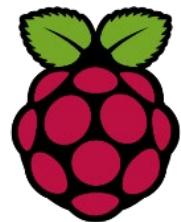
From \$35



Además es más silenciosa y tiene menor consumo

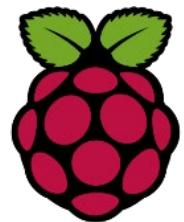
# Raspberry Pi 4: procesador

- ARM Cortex A72
- Arquitectura ARM big.LITTLE: combina núcleos potentes y lentos: 4 núcleos @ 2.5 GHz por núcleo + 4 núcleos A53 de bajo consumo
- Proceso de fabricación en 16 nm FinFet+ → 64b con un consumo 75% menor que en 32b
- Caché CoreLink CCI-500
- Tarjeta gráfica Mali-T880



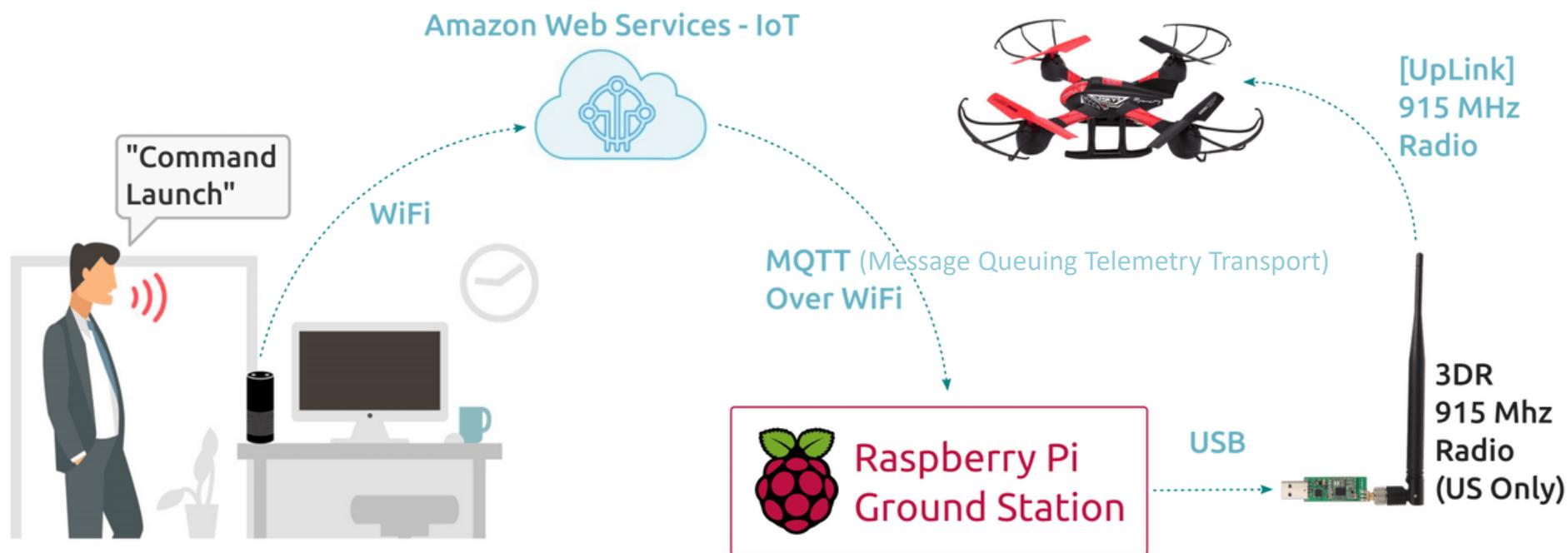
# Raspberry Pi: Casos de uso

- Mini PC de escritorio
- Servidor de impresión WiFi, servidor web o FTP
- Sistema de almacenamiento NAS (Network-attached storage)
- Convertir TV en Smart TV
- Sistema de música en streaming casero
- Controlador para robótica
- Hogar domótico
- Vigilancia y videoseguridad



# Raspberry Pi: Casos de uso

- Dron controlado por voz (Amazon Alexa)



\* <https://www.hackster.io/veggiebenz/voice-controlled-drone-with-raspi-amazon-echo-and-3dr-iris-c9fd2a>

# BeagleBone

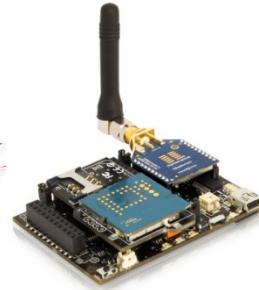
From 115€

- **BeagleBone** es un **ordenador** de placa reducida basado en **Linux**.
- Diseñado por Texas Instruments con una filosofía **open-source y de bajo coste**.
- Utilizada para aplicaciones de visión artificial
- **BeagleBone AI**: TI C66x digital-signal-processor (DSP) cores y embedded-vision-engine (EVE) cores. OpenCL API para machine learning con herramientas pre-instaladas.

# Arduino, Raspberry Pi, BeagleBone

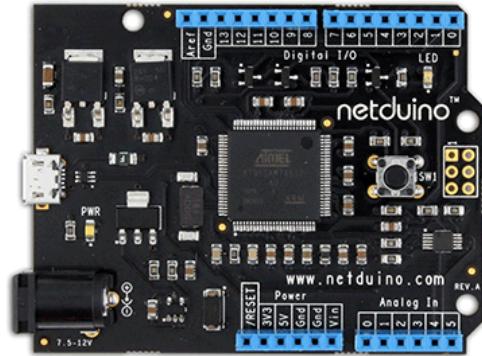
Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	\$29.95	\$35	\$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

# Libelium WaspMote



- Plataforma modular Open Source, optimizada en cuanto a consumo y aplicaciones. Pensada para la creación de grandes redes de sensores, con gran autonomía energética.
- Mismo entorno de desarrollo que Arduino.
- Protocolo de comunicación Zigbee
- Módulos para dotarle de diferentes medios de comunicación adicionales (Bluetooth, GPS, y GPRS).

# Netduino



- Potentes y flexibles.
- Implementación de robots y pequeños autómatas con **lenguajes de alto nivel**. El SO es el Microsoft .NET\* Micro Framework.
- Programables en C# (versión C++ de Microsoft para .Net)
- Netduino Plus incluye 22 GPIO con SPI, I2C, 4 UART (1 RTS / CTS), 6 PWM y 6 canales de ADC para interactuar con los interruptores, sensores, LEDs, dispositivos de serie, y mucho más.

\*.NET es una plataforma de código abierto para crear aplicaciones de escritorio, web y móviles que se pueden ejecutar de forma nativa en cualquier sistema operativo



# meadow



- Tiene el poder de RaspberryPi, el factor de computación de un Arduino y la capacidad de administración de una aplicación móvil.
- Diseñado para ejecutarse en una variedad de microcontroladores. La primera placa se basa en la MCU STM32F7 (STMicroelectronics), 2xpotente, 0,5xconsumo.
- WiFi, BLE, 32 MB de RAM, 32 MB de Flash. 25 puertos GPIO, PWM, I2C, SPI, CAN, UART y cargador de batería LiPo integrado, todo ello en el factor de forma Adafruit Feather.



# meadow



- Incluye soporte para actualizaciones seguras por aire (OTA), lo que permite la gestión remota de las instalaciones de campo de IoT → clave para las implementaciones empresariales de IoT.
- IA en IoT (AIoT): Meadow permite ejecutar visión artificial a través de TensorFlow y otros paquetes de inteligencia artificial de alto nivel localmente en chip.
- **Código abierto** (como el de Netduino). Desde el código nativo, a las bibliotecas de clases hasta la extensión de Visual Studio.



# Meadow (Pros & Cons)



- + Compatible con el factor de forma [Adafruit feather](#)
- + Incluye un conector de batería y cargador integrado:  
soluciones autónomas
- + Código en C# usando Visual Studio
- ~50\$, mucho más caro que Arduino y Raspberry Pi
- Competencia de la plataforma OA .NET [Nano Framework](#): esfuerzo para portar .NET a varios chips STM, TI y [Espressif ESP32](#) (módulos Wi-Fi+Bluetooth/BLE para IoT).



# Jetson Nano

- *“Bringing the Power of Modern AI to Millions of Devices”*
- Aplicaciones en visión por computador y procesado de voz

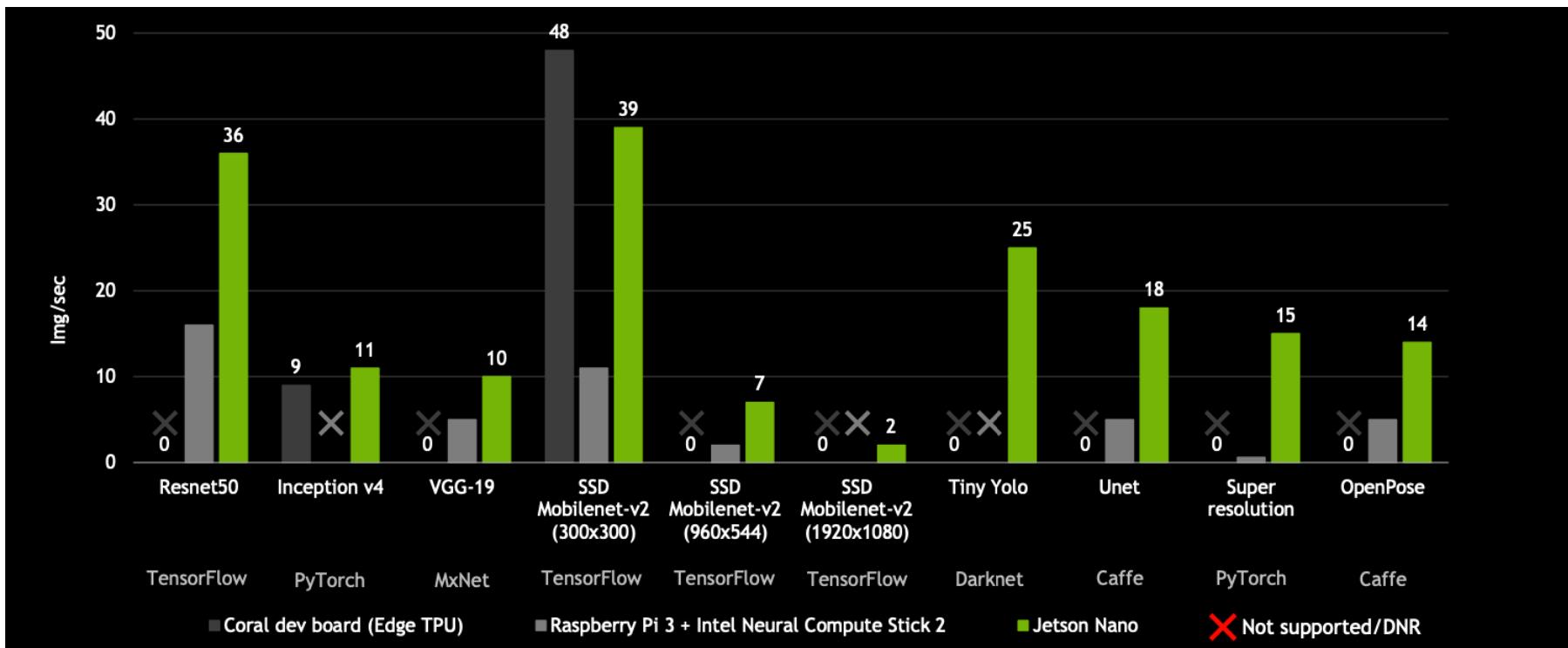
Developer Kit Technical Specifications	
<b>GPU</b>	128-core NVIDIA Maxwell™
<b>CPU</b>	Quad-core ARM® A57 @ 1.43 GHz
<b>Memory</b>	2 GB 64-bit LPDDR4 25.6 GB/s
<b>Storage</b>	microSD (Card not included)
<b>Video Encoder</b>	4Kp30   4x 1080p30   9x 720p30 (H.264/H.265)
<b>Video Decoder</b>	4Kp60   2x 4Kp30   8x 1080p30   18x 720p30 (H.264/H.265)
<b>Connectivity</b>	Gigabit Ethernet, 802.11ac wireless*
<b>Camera</b>	1x MIPI CSI-2 connector
<b>Display</b>	HDMI
<b>USB</b>	1x USB 3.0 Type A, 2x USB 2.0 Type A, 1x USB 2.0 Micro-B
<b>Others</b>	40-pin header [GPIO, I2C, I2S, SPI, UART] 12-pin header [Power and related signals, UART] 4-pin fan header*
<b>Mechanical</b>	100 mm x 80 mm x 29 mm



# Jetson Nano

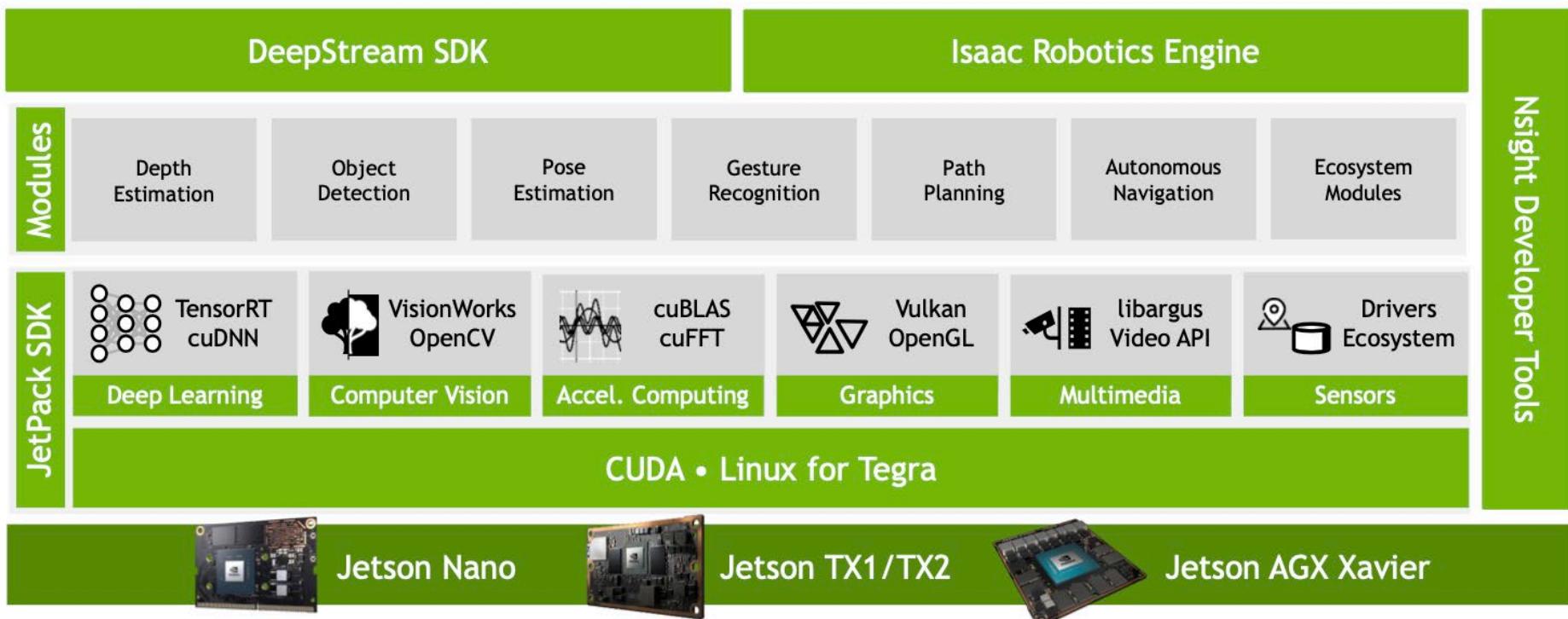
- Permite ejecutar múltiples redes neuronales en paralelo.
- Incluye soporte NVIDIA JetPack con software CUDA-X
- Soporte para tecnologías cloud
- Frameworks específicos para análisis de video, salud, genómica y robótica.

# Jetson Nano





# Otros productos NVIDIA



## **BLOQUE III: Los dispositivos móviles como sensores de datos**

# Los dispositivos móviles como sensores de datos

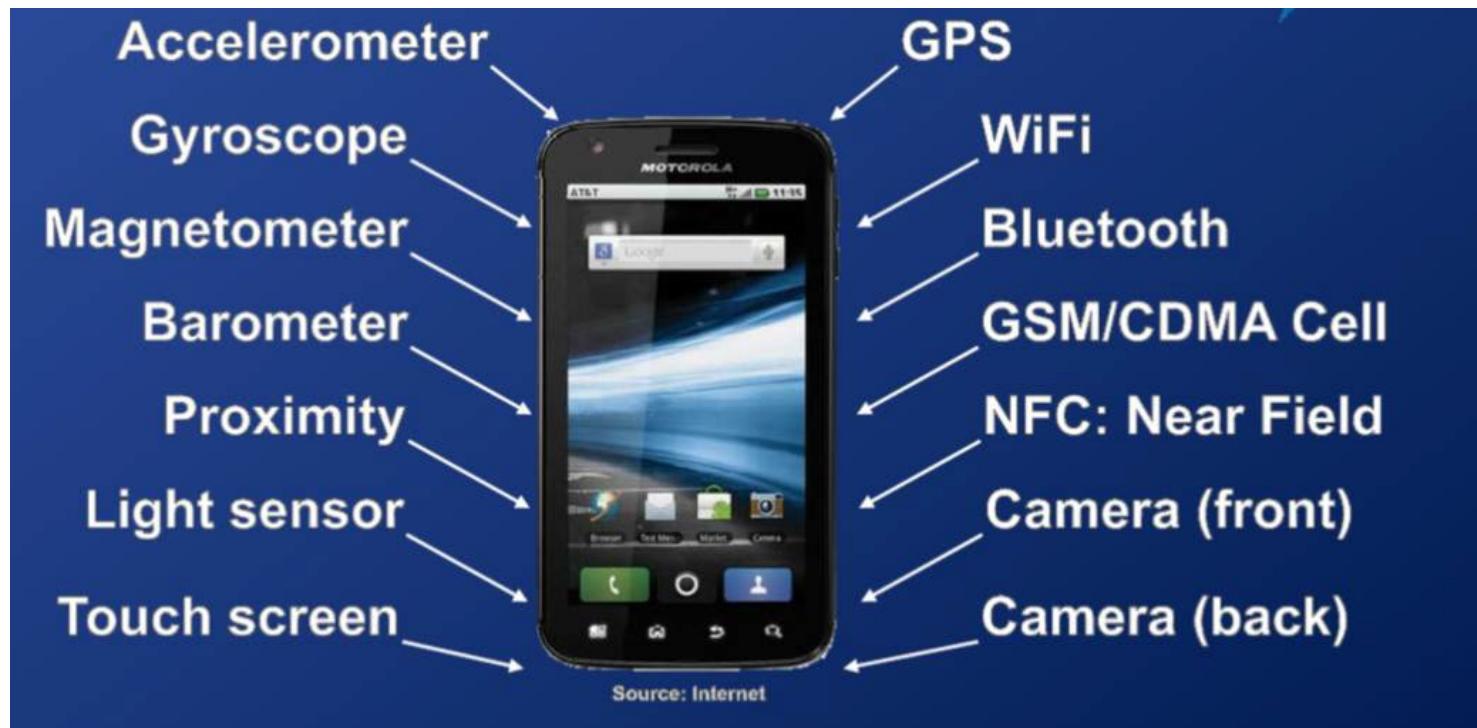
- Características generales de los dispositivos móviles
- Smartphones como sensores de datos
- WEARABLES como sensores de datos
- Código QR (Quick Response)
- Tecnología RFID & NFC



# Características generales

- Aparatos pequeños.
- Capacidad de procesamiento.
- Conexión permanente o intermitente a una red.
- Memoria (RAM, tarjetas MicroSD, flash, etc.).
- Uso individual
- Alta capacidad de interacción mediante la pantalla o el teclado

# Smartphones como sensores de datos



# Smartphones como sensores de datos

- **Giroscopio:** permite girar la pantalla.
- **Acelerómetro:** mide la aceleración con respecto a la fuerza de la gravedad, es decir, detecta el movimiento y la orientación.
- **Magnetómetro:** detecta campos magnéticos. Se usa en las aplicaciones de la brújula para señalar el polo norte o para detectar metales.
- **Podómetro:** mide los pasos que da el usuario

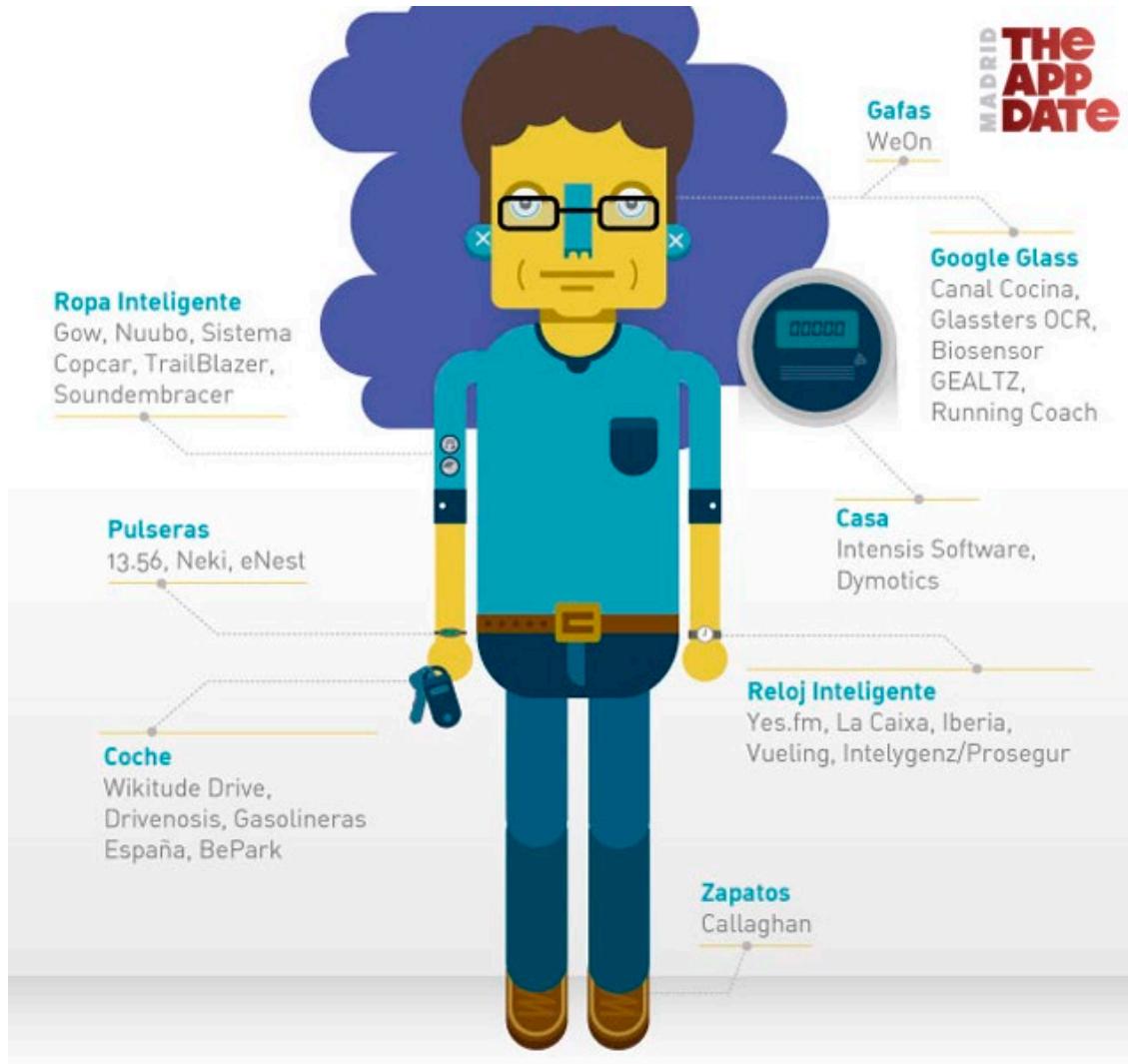
# Smartphones como sensores de datos

- **Sensor de proximidad:** se sitúa cerca del auricular del teléfono para reconocer cuándo el usuario pega la oreja al hablar por el móvil.
- **Sensor de luz:** mide la luz ambiental y ajusta automáticamente el brillo de la pantalla.
- **Termómetro:** controla la temperatura interior del dispositivo y la batería para evitar daños

# Smartphones como sensores de datos

- **Sensor de huellas dactilares:** desbloqueo de pantalla o para pagar con el móvil.
- **Pulsómetro:** mide el ritmo cardiaco a través de los vasos sanguíneos de los dedos.
- **Sensor de humedad del aire:** se utiliza, p.e., en la aplicaciones de salud para saber si el usuario está o no en su 'zona de confort'.

# Wearables como sensores de datos



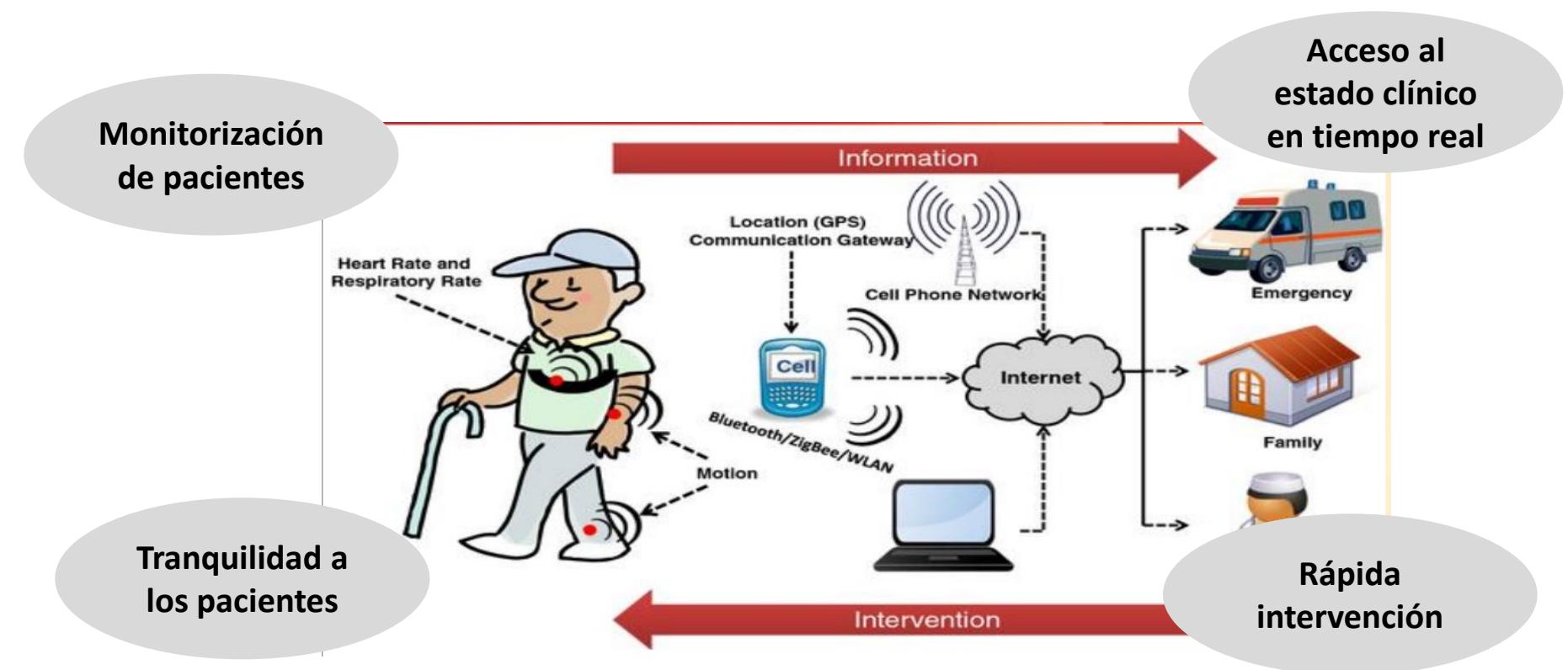
# Wearables como sensores de datos

- Desmonopolizar la atención constante del usuario.
- No debe restringir la movilidad del usuario mientras este realiza otras tareas.
- Observable por el usuario: el medio de salida es perceptible por el portador.
- Controlable por el usuario: Puede tomar control en el momento que lo deseé.

# Wearables como sensores de datos

## Wearables de salud:

**E-health**, alude a la práctica de cuidados sanitarios apoyada en tecnologías de la información y las comunicaciones (TIC). Las estadísticas recogidas por estas prendas electrónicas WEARABLES permiten conocer determinados parámetros sanitarios de la persona que las lleva.



# Smartglasses



- Tecnología inteligente para trabajar con fines correctivos, combatir anomalías oculares y proteger la vista contra los rayos ultravioleta.
- Incluyen cámaras y reproducen imágenes directamente en el campo de visión, siendo capaces de reconocer rostros, interactuar con sitios web, enviar información a redes sociales y mostrar gráficas computarizadas en 3D.

# Smartglasses



- GoogleGlass: se utilizan únicamente con la voz. Permiten estar conectados pudiendo sacar fotos, vídeos o realizar llamadas desde las propias gafas. Aplicaciones en la educación, el turismo, la medicina o el marketing.
- Alternativas: Smart Eyeglass (Sony) que apuesta por la tecnología de realidad aumentada. También Epson y otras marcas han optado por proponer sus propias alternativas.

# Smartwatches



- Incluyen funciones como podómetros, medidores de ritmo cardiaco, GPS, control de los ciclos del sueño y muchas de las funciones de un smartphone.
- Idóneos para **monitorizar** el entrenamiento deportivo, como guías de viaje o para controlar las notificaciones que recibamos sin necesidad de visualizar directamente el Smartphone.

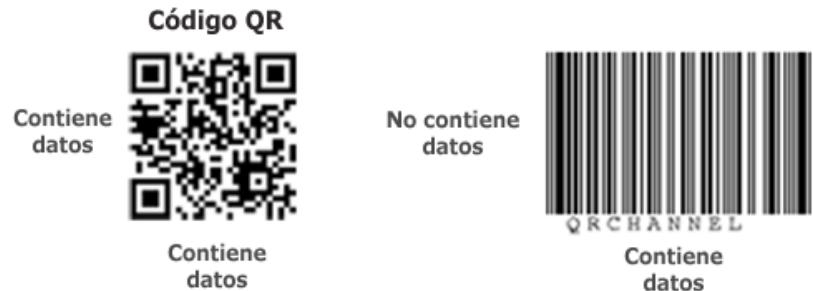
# Fitness trackers



- Su **tamaño pequeño** y su aspecto moderno hacen que sus **funcionalidades** se vean **reducidas**, hasta incluso sólo una funcionalidad.
- Las **pulseras fitness** pueden ser un elemento motivador para gente que lleva un estilo de vida sedentario.

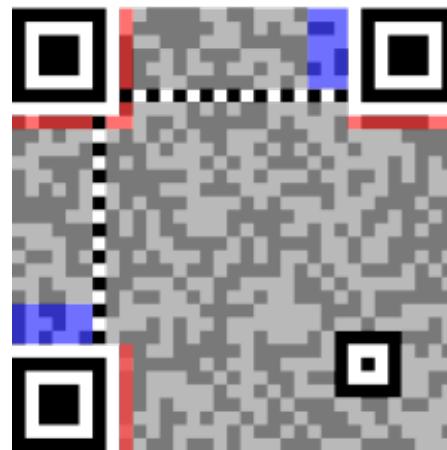
# Código QR

Matriz en dos dimensiones formada por una serie de cuadrados negros sobre fondo blanco. Esta matriz es leída por un lector específico y de forma inmediata nos lleva a una aplicación en Internet ya sea un mapa de localización, un correo electrónico, una página web o un perfil en una red social.



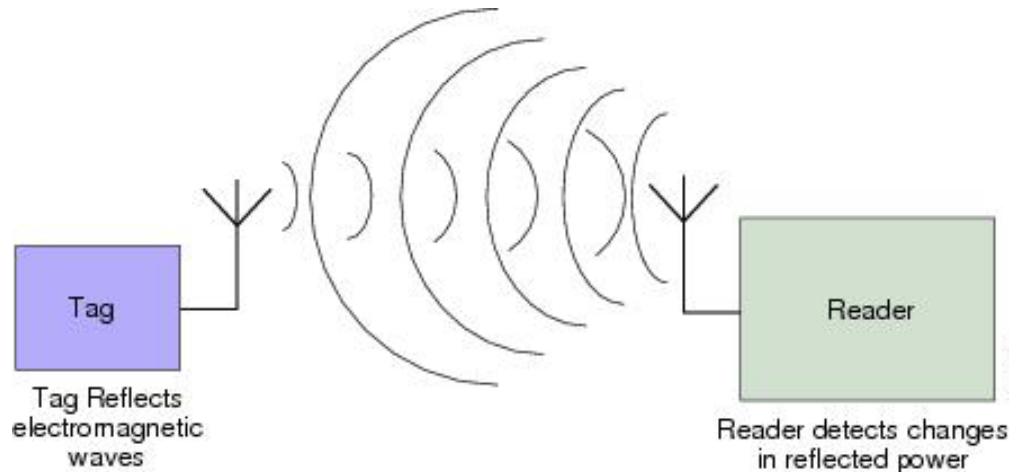
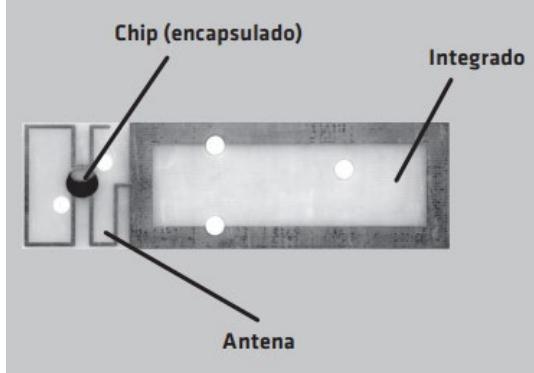
# Código QR

- Compuestos por tres cuadrados en las esquinas que permiten detectar al lector la posición del código QR y una serie de cuadrados dispersos que codifican el alineamiento y la sincronización.
- Se pueden incorporar imágenes personalizada a nuestro código QR dando un aspecto más artístico y personal.
- Son libres y cualquiera puede crear sus propios QR(  
<https://www.qrcode.es/es/generador-qr-code/>)



- 1. Información de la Versión
- 2. Información del Formato
- 3. Corrección de Errores y Datos
- 4. Patrones Requeridos
  - 4.1. Posición
  - 4.2. Alineamiento
  - 4.3. Sincronización

# Tecnología RFID



Los **tags** RFID constan de dos elementos: un chip y una antena.

**RFID pasivo**



**RFID activo**



## Definición

La identificación por radiofrecuencia o **RFID** (*Radio Frequency IDentification*), es una tecnología de identificación remota e inalámbrica en la cual un dispositivo lector o **reader**, se comunica a través de una antena con un **transponder** (también conocido como tag o etiqueta) mediante ondas de radio.

# Tecnología RFID



- El alcance de lectura varía de unos cuantos centímetros a decenas de metros, en función de la frecuencia que se utilice, de la potencia y de la sensibilidad direccional de la antena. La tecnología HF tiene un alcance máximo de lectura de unos tres metros. La tecnología UHF proporciona un alcance de lectura de 20 metros o más.
- La presencia de metales y líquidos puede causar interferencias que afecten a la lectura/escritura.
- Los chips RFID son difíciles de hackear.
- Numerosas normas que garantizan la diversidad de frecuencias y aplicaciones.

# Tecnología RFID e IoT

