Arquitectura y Conectividad

Cuestionario N°5

Profesor Ing. Morales Jorge Elias | https://github.com/JorEl057

Miembros:

- Durigutti, Vittorio | GitHub: https://github.com/vittoriodurigutti
- Zalazar, Joaquín | GitHub: https://github.com/breaakerr
- Marquez, José | Github: https://github.com/marquezjose
- Lujan, Luciano | Github: https://github.com/lucianoilujan
- Velez, Nahuel | Github: https://github.com/Lucasmurua19
- Juncos, Lisandro | Github: https://github.com/Lisandro-05
- Garzón, Joaquín | Github: https://github.com/Joacogarzonn
- Guzmán, Maria |Github: https://github.com/lilenguzman01

Actividad: (Click en las preguntas para link a las respuestas)

1) ¿Qué es un protocolo COAP?, ¿Para qué se usa? Ejemplifique.
2) ¿Qué es un protocolo AMQP?, ¿Para qué se usa? Ejemplifique
3) ¿Qué es un protocolo MODBUS?, ¿Para qué se usa? Ejemplifique
4) ¿Qué es un protocolo HART?, ¿Para qué se usa? Ejemplifique
5) ¿Qué es un protocolo PROFINET?, ¿Para qué se usa? Ejemplifique
6) ¿Qué es un protocolo CANopen?, ¿Para qué se usa? Ejemplifique
7) ¿Qué es un protocolo PROFIBUS-DP/PA?, ¿Para qué se usa? Ejemplifique

1) ¿Qué es un protocolo COAP?, ¿Para qué se usa? Ejemplifique.

¿Qué es un protocolo COAP?

El protocolo COAP (Constrained Application Protocol) es un protocolo de aplicación diseñado para entornos con recursos limitados, como dispositivos IoT y redes de sensores. COAP se utiliza para facilitar la comunicación entre estos dispositivos y las aplicaciones que los controlan.

COAP se basa en la arquitectura cliente-servidor y utiliza el protocolo de transferencia de hipertexto (HTTP) para habilitar la comunicación entre dispositivos y aplicaciones. Este protocolo se desarrolló como una alternativa ligera a HTTP, ya que HTTP estándar puede ser demasiado complejo y pesado para dispositivos con limitaciones de recursos. COAP utiliza el modelo de intercambio de mensajes basado en UDP (User Datagram Protocol) para minimizar el tamaño del paquete y la sobrecarga de protocolo. También incluye mecanismos de seguridad, como autenticación y encriptación, a través de la capa de transporte para proteger la comunicación entre los dispositivos.

Características de CoAP

El encabezado está comprimido, el formato del paquete es simple y el mensaje puede ser muy corto. El mensaje CoAP más pequeño tiene solo cuatro bytes. La capa de transporte utiliza el protocolo UDP para reducir los gastos generales de la red y CoAP admite multidifusión.
Para compensar la falta de fiabilidad de la transmisión UDP, CoAP tiene un mecanismo de retransmisión de mensajes.
CoAP no admite la conexión persistente y no tiene mensajes de heartbeat. Cuando no hay servicio, el dispositivo no necesita enviar mensajes a sistemas externos.
Rendimiento deficiente en tiempo real: el dispositivo debe activarse antes de realizar los servicios.

¿Para qué se usa?

CoAP se usa para que los dispositivos pequeños puedan comunicarse de manera eficiente en redes IoT. Estos dispositivos suelen tener muy poca memoria, poco poder de procesamiento y, a veces, una conexión de internet débil o de baja velocidad. Entonces, necesitan un protocolo liviano y rápido como CoAP.

Ademas, CoAP es más adecuado para dispositivos que requieren el mecanismo de suspensión / activación en escenarios de IoT. El dispositivo puede permanecer en modo de hibernación durante largos períodos de tiempo, lo que ahorra energía de la batería, y una sola batería puede durar 10 años o más. Actualmente, soluciones como medidores de agua

inteligentes, medidores de electricidad inteligente, agricultura inteligente y estacionamiento inteligente utilizan el protocolo CoAP.

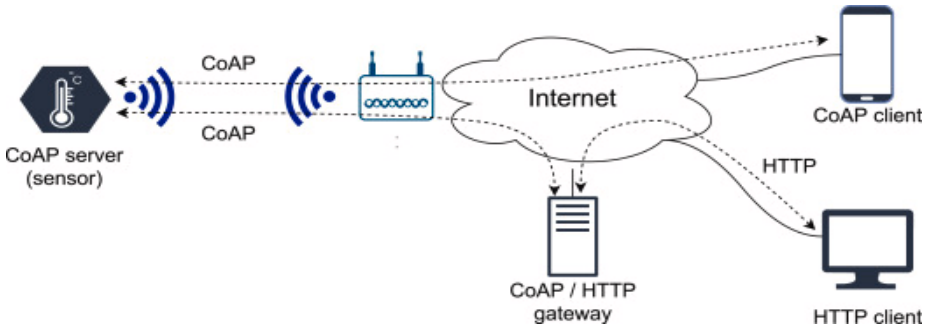
Ejemplo de cómo usar COAP en un escenario de IoT:

Supongamos que tienes un sistema de iluminación inteligente en tu hogar, donde cada luz es un dispositivo IoT compatible con COAP y quieres controlar las luces desde tu teléfono móvil.

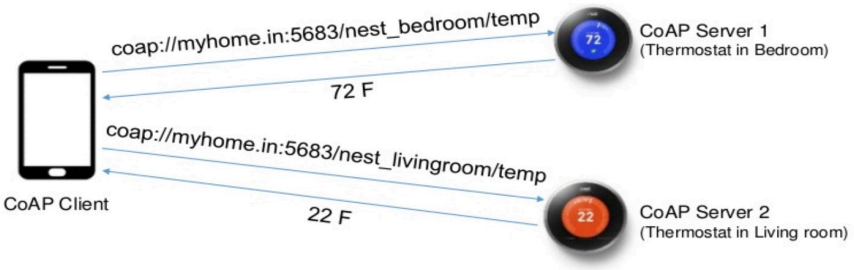
- **Configuración:** Configura cada luz con una dirección IP y un recurso identificador único, por ejemplo, "coap://192.168.1.100/light". Asegúrate de que tu teléfono móvil también esté conectado a la misma red.
- **Descubrimiento de recursos:** Desde tu teléfono móvil, envía una solicitud de descubrimiento COAP a la dirección de broadcast, utilizando el método GET y la opción de mensaje "Well-Known Core". Recibirás una respuesta con los recursos disponibles en la red, incluyendo las direcciones y los identificadores de las luces.
- **Encendido de una luz:** Desde tu teléfono móvil, envía una solicitud COAP a la dirección de la luz específica, utilizando el método PUT y el recurso identificador de la luz. Incluye en el cuerpo del mensaje la información para encender la luz, por ejemplo, {"state": "on"}. La luz recibirá la solicitud y cambiará su estado a encendido.
- **Apagado de una luz:** De manera similar al paso anterior, envía una solicitud COAP a la dirección de la luz específica, utilizando el método PUT y el recurso identificador de la luz. Incluye en el cuerpo del mensaje la información para apagar la luz, por ejemplo, {"state": "off"}. La luz recibirá la solicitud y cambiará su estado a apagado.

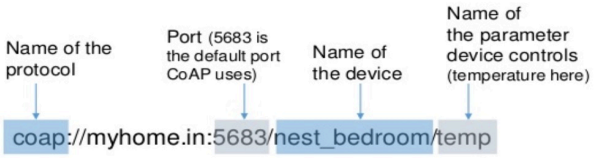
En este ejemplo, COAP se utiliza para enviar solicitudes y recibir respuestas entre tu teléfono móvil y las luces del sistema de iluminación inteligente.

Diagrama de sistema



CoAP – Request Response





2) ¿Qué es un protocolo AMQP?, ¿Para qué se usa? Ejemplifique

El **Advanced Message Queuing Protocol (AMQP)** es un protocolo de comunicación abierto, estándar y orientado a mensajes. Fue diseñado para sistemas de mensajería empresarial, garantizando la interoperabilidad entre diferentes tecnologías y permitiendo el intercambio confiable de mensajes entre aplicaciones distribuidas.

AMQP es utilizado en arquitecturas basadas en microservicios, sistemas de procesamiento distribuido, y aplicaciones que requieren una alta confiabilidad y flexibilidad en la comunicación.

¿Cómo funciona AMQP?

AMQP opera bajo un modelo cliente-servidor, en el que:

Componentes principales

- Productor (Producer): Es la entidad que envía los mensajes al sistema de mensajería.
- Broker: Es el intermediario que administra el enrutamiento y entrega de los mensajes. Ejemplos de brokers que implementan AMQP incluyen RabbitMQ y Apache ActiveMQ.
- Cola (Queue): Es el destino donde se almacenan los mensajes antes de ser consumidos.
- Consumidor (Consumer): Es la entidad que recibe y procesa los mensajes desde las colas.

Flujo de trabajo:

1. **Publicación:** El productor envía mensajes al broker.
2. **Enrutamiento**: El broker analiza las reglas configuradas (como exchanges y bindings) para determinar a qué cola se debe enviar el mensaje.
3. **Almacenamiento**: El mensaje se coloca en la cola designad
4. **Consumo:** El consumidor extrae mensajes de la cola para procesarlos.

¿Cómo funciona AMQP?

1. **Fiabilidad**: AMQP asegura que los mensajes lleguen a su destino y permite confirmaciones de entrega.
2. **Transacciones:** Soporte para transacciones de mensajes, asegurando integridad en operaciones críticas.

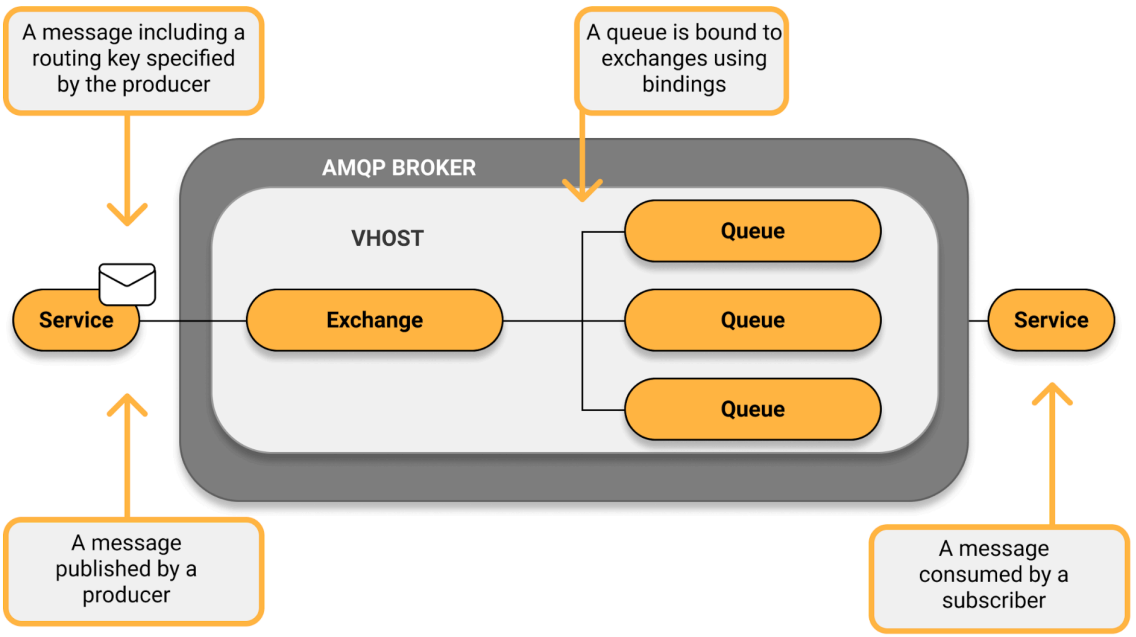
3. **Interoperabilidad**: Funciona en múltiples lenguajes de programación y plataformas.
4. **Modelo basado en cola:** Facilita el procesamiento asincrónico de mensajes.

¿Cómo funciona AMQP?

Supón que tienes un sistema de pedidos:

- Productores: Las aplicaciones que generan nuevos pedidos (como una app de compras).
- Broker RabbitMQ distribuye los pedidos en colas específicas según el tipo de procesamiento necesario (facturación, envíos, etc.).
- Consumidores: Servicios que procesan los pedidos desde las colas, como un módulo de impresión de etiquetas de envío.

¿Cómo funciona AMQP?



3) ¿Qué es un protocolo MODBUS?, ¿Para qué se usa? Ejemplifique

¿Qué es el protocolo Modbus RTU?

Es un protocolo de comunicaciones, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon (ahora Schneider Electric) para su gama de controladores lógicos programables (PLCs).
Debido a que este protocolo fue público, de fácil uso y que requiere poco desarrollo (maneja bloques de datos sin suponer restricciones) se convirtió en un protocolo de comunicaciones estándar en la industria. Es el protocolo de mayor disponibilidad para la conexión de dispositivos electrónicos industriales.
El protocolo Modbus permite el control de una red de dispositivos, por ejemplo un equipo de medición temperatura y humedad puede comunicar los resultados a una PC. Modbus también se usa para la conexión de un PC de supervisión con una unidad remota (RTU) en sistemas de supervisión de adquisición de datos (SCADA).
Existen versiones del protocolo Modbus para puerto serial y Ethernet (Modbus/TCP).

¿Para qué se usa?

Se utiliza para conectar dispositivos electrónicos industriales y permitir la supervisión y el control remoto de procesos. MODBUS es simple, robusto y confiable, lo que lo hace ideal para aplicaciones industriales donde se requiere comunicación maestro/esclavo.

El protocolo MODBUS funciona bajo un esquema **maestro-esclavo** (o cliente-servidor) donde un dispositivo central (el maestro o cliente) se comunica con uno o varios dispositivos periféricos (los esclavos o servidores) para intercambiar información.

Aquí se explican los aspectos clave de su funcionamiento:

1. Arquitectura Maestro-Esclavo:
   - **Maestro:** Es el dispositivo que inicia la comunicación enviando una **consulta** a uno o varios esclavos. Típicamente es un PLC, una PC con software SCADA, o una HMI (Human Machine Interface).

- **Esclavo:** Son los dispositivos que **responden** a las consultas del maestro con la información solicitada o realizando la acción indicada. Pueden ser sensores, actuadores, otros PLCs, medidores, etc.
- **Dirección de Esclavo**: Cada esclavo en una red MODBUS tiene una **dirección única** (entre 1 y 247 para MODBUS Serial). El maestro utiliza esta dirección para especificar a qué esclavo va dirigida la consulta.

2. Modelo de Datos:

MODBUS define cuatro tablas de datos principales que los esclavos utilizan para almacenar información. El maestro puede acceder a estas tablas:

- **Bobinas (Coils)**: Registros de 1 bit para salidas discretas (on/off). Se acceden con los códigos de función 01 (Leer Bobinas) y 05/15 (Escribir Bobinas).
- **Entradas Discretas (Discrete Inputs)**: Registros de 1 bit para entradas discretas (solo lectura). Se acceden con el código de función 02 (Leer Entradas Discretas).
- **Registros de Entrada (Input Registers):** Registros de 16 bits para valores analógicos de entrada (solo lectura). Se acceden con el código de función 04 (Leer Registros de Entrada).
- **Registros de Retención (Holding Registers):** Registros de 16 bits para valores analógicos de salida o parámetros de configuración (lectura/escritura). Se acceden con los códigos de función 03 (Leer Registros de Retención) y 06/16 (Escribir Registros de Retención).

Es importante notar: La documentación de cada dispositivo esclavo indicará cómo se mapean sus datos físicos (temperatura, estado de un motor, etc.) a estas tablas y direcciones MODBUS.

3. Unidad de Datos del Protocolo (PDU):

La PDU es la parte del mensaje MODBUS que contiene la información esencial para la comunicación. Tiene un formato común tanto para la consulta del maestro como para la respuesta del esclavo:

- **Código de Función (Function Code):** Un byte que indica la acción que el maestro quiere que el esclavo realice (lectura o escritura de datos) o el tipo de respuesta del esclavo.
- **Datos (Data)**: Contiene información adicional necesaria para la función, como la dirección inicial del registro a leer o escribir, la cantidad de registros, y los datos a escribir.

4. Unidad de Datos de Aplicación (ADU):
La ADU es la PDU envuelta en información adicional dependiendo del medio de comunicación utilizado:

- **MODBUS Serial (RTU):**
  - Dirección del Esclavo (Slave Address): Un byte que especifica el esclavo destino.
  - PDU (Código de Función + Datos):Verificación de Redundancia Cíclica (CRC - Cyclic Redundancy Check):** Dos bytes para la detección de errores en la transmisión.

- MODBUS TCP/IP:
  - Identificador de Transacción (Transaction Identifier): Dos bytes para identificar pares de solicitud-respuesta.
  - Identificador de Protocolo (Protocol Identifier): Dos bytes (siempre 0 para MODBUS).
  - Longitud (Length):** Dos bytes que indican la longitud de los bytes siguientes.
  - Identificador de Unidad (Unit Identifier):** Un byte (similar a la dirección de esclavo en serial, útil para enrutar a través de gateways).
  - PDU (Código de Función + Datos):**

5. Proceso de Comunicación:

1. El maestro envía una consulta MODBUS:** El maestro construye un mensaje ADU con la dirección del esclavo, el código de función deseado y cualquier dato necesario (dirección de inicio, cantidad de datos, etc.).
2. El mensaje viaja por el medio de comunicación:** Esto puede ser un cable serial (RS-232, RS-485) para MODBUS RTU o una red Ethernet para MODBUS TCP/IP.

3. El esclavo recibe la consulta:** Cada esclavo en la red escucha las consultas. Solo el esclavo cuya dirección coincide con la dirección en la consulta procesa el mensaje.
4. El esclavo procesa la solicitud:** El esclavo realiza la acción solicitada (lee los datos de sus tablas, escribe nuevos valores, etc.).
5. El esclavo envía una respuesta MODBUS (si es necesario):** El esclavo construye un mensaje ADU de respuesta que incluye el código de función (que puede indicar un error), los datos solicitados (si fue una lectura) y cualquier información de verificación de errores.
6. El maestro recibe la respuesta:** El maestro verifica si la respuesta es válida (dirección correcta, sin errores de transmisión) y extrae los datos necesarios.

Ejemplo Simplificado (Lectura de un Registro de Retención):

Imagina un maestro (PLC) queriendo leer el valor de temperatura (almacenado en el registro de retención número 40001, que corresponde a la dirección 0 en la PDU) de un sensor esclavo con dirección 05 (en MODBUS RTU).

- Consulta del Maestro (MODBUS RTU)

[Dirección Esclavo (05)]
[Código Función (03 - Leer Holding Registers)]
[Dirección Inicio (00 00)]
[Cantidad Registros (00 01)]
[CRC (XX XX)]

- Respuesta del Esclavo (si la temperatura es 25 °C, que podría representarse como el valor hexadecimal 0019):

[Dirección Esclavo (05)]
[Código Función (03)]
[Cantidad Bytes (02)]
[Dato Alto (00)]
[Dato Bajo (19)]
[CRC (YY YY)]

¿Cuáles son los beneficios de utilizar el protocolo Modbus RTU/TCP?

- De código abierto, no se requiere pagar por licencia.
- Ampliamente soportado por HMIs o softwares SCADA
- Fácil de usar
- Se pueden integrar varios equipos fácilmente
- Bajo costo de desarrollo
- Conocido ampliamente en la industria

¿En qué lenguaje se programa?

El protocolo MODBUS no define un lenguaje de programación específico, ya que se centra en el formato de mensajes y el protocolo de comunicación. En entornos industriales se puede encontrar como **Ladder (Escalera)** dentro de PLCs.

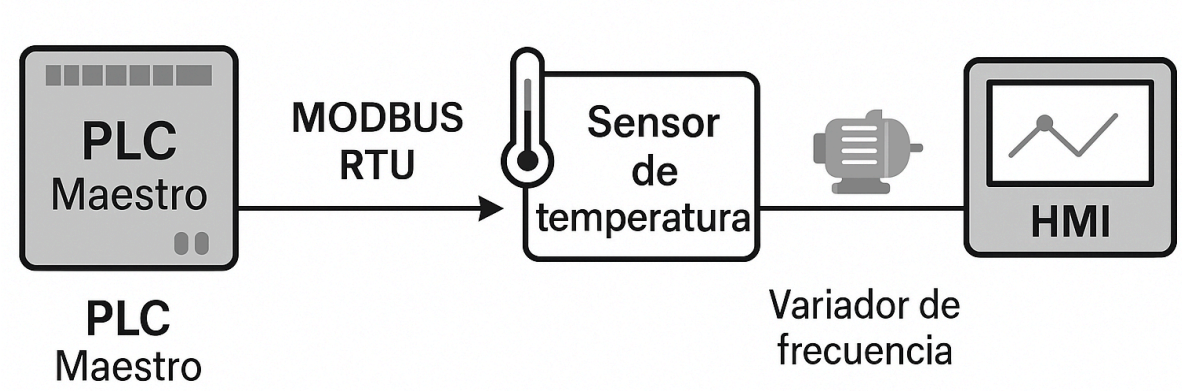
El lenguaje más utilizado actualmente para sistemas embebidos con MODBUS es **C/C++** mientras que **Python** es muy popular para pruebas, herramientas de monitoreo o desarrollo rápido en PC.

Ejemplo de aplicación

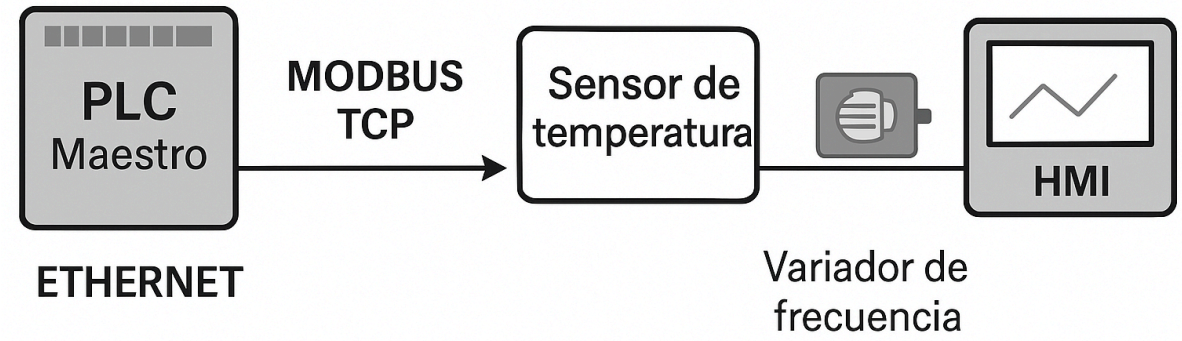
Un PLC maestro se comunica mediante MODBUS RTU sobre RS-485 con los siguientes dispositivos esclavos:

- Un sensor de temperatura
- Un variador de frecuencia (drive)
- Una pantalla HMI

El PLC solicita datos de temperatura, ajusta la velocidad del motor según estos datos y muestra la información en la HMI.



También se puede realizar la comunicación vía ethernet:



4) ¿Qué es un protocolo HART?, ¿Para qué se usa? Ejemplifique

HART *(Highway Addressable Remote Transducer)* es un **protocolo de comunicación industrial** utilizado principalmente en la **automatización de procesos**. Fue desarrollado para permitir que los **instrumentos de medición** como sensores y transmisores puedan comunicarse con un sistema de control (como un PLC o DCS) y además puedan intercambiar información adicional, todo a través de una **única línea de comunicación**.

Lo interesante de HART es que combinan **comunicación digital y analógica** en el **mismo cable**. Esto significa que, por ejemplo, un **sensor de presión** puede seguir enviando una **señal analógica** tradicional de 4-20 mA (que es lo que muchos sistemas usan para leer valores como la presión), pero al mismo tiempo puede estar enviando **datos digitales** adicionales sobre su estado, configuraciones o diagnósticos, todo sin interrumpir la **señal analógica**.

Topologías soportadas

- **Punto a punto** (1 maestro ↔ 1 esclavo)
- **Multidrop** (1 maestro ↔ hasta 15 esclavos)
    - En este modo, cada dispositivo fija su salida a **4 mA**

¿Cómo funciona?

Imaginemos que tenemos un **sensor de temperatura** que está conectado a un sistema de control.

- **Señal analógica:** El sensor envía una señal de 4-20 mA, que representa la temperatura medida.
- **Señal digital (HART):** Al mismo tiempo, el sensor también envía datos digitales sobre su estado, como su modelo, número de serie, posibles fallos, o incluso cambios de configuración.
- **Comunicación bidireccional:** Con HART, además de **leer los valores** de la medición (como la temperatura), podés **enviar comandos** para configurar el sensor, **realizar diagnósticos** o **hacer ajustes**, todo de forma remota.

La señal analógica y la digital viajan a través del mismo cable de manera **superpuesta**. Es decir, las dos señales **no se interfieren entre sí**.

Lenguaje de programación utilizado

El protocolo HART no depende de un lenguaje de programación específico, ya que es un **estándar de comunicación industrial** que define cómo se intercambian datos entre dispositivos (por ejemplo, sensores y controladores). Sin embargo, cuando querés programar un sistema que se comunique mediante HART (como un maestro HART o un sistema SCADA), los lenguajes más usados para implementar esa comunicación son:

- **C / C++:** Muy común para programar dispositivos embebidos que actúan como maestros HART o gateways. Permite control bajo nivel del hardware y del protocolo.
- **Python:** Usado en entornos de prueba, simulación y desarrollo rápido. Existen bibliotecas y scripts que permiten comunicarse con dispositivos HART a través de puertos serie o adaptadores USB.
- **LabVIEW:** Utilizado en automatización y control industrial. Algunas librerías permiten acceder a dispositivos HART si están conectados a través de interfaces compatibles.
- **.NET / Java**: Usados en sistemas de gestión o aplicaciones de escritorio que se comunican con dispositivos HART a través de servidores OPC o gateways.

Ejemplo

Imaginemos que tenemos una **planta de procesamiento químico** con sensores de temperatura en varios puntos. Estos sensores están conectados a un sistema central de control que mide la temperatura para asegurar que los procesos se mantengan dentro de los parámetros correctos.
Con la implementación de HART, los sensores siguen enviando señales análogas que indican la temperatura, pero también están enviando información digital adicional, como el modelo del sensor, posibles fallos o su estado a través de HART.

¿Por qué es importante?

- **Flexibilidad:** Permite la comunicación digital sin cambiar los sistemas de control analógicos existentes, lo que lo hace muy fácil de integrar en instalaciones ya existentes.
- **Diagnósticos avanzados:** Puedes obtener información detallada sobre el estado de los dispositivos, como fallos o condiciones de mantenimiento, algo que no era posible con los sistemas analógicos tradicionales.
- **Configuración remota:** Puedes ajustar parámetros de los instrumentos sin necesidad de ir físicamente al dispositivo.

Desventajas del protocolo

1. **Velocidad de comunicación muy baja**

La velocidad típica es de solo 1200 bps (bits por segundo), lo que lo hace lento para enviar grandes cantidades de datos o múltiples variables.

2. **Limitado a redes punto a punto o multidrop reducidas**

Aunque admite conexiones multidrop (varios dispositivos en una línea), el número de dispositivos es limitado (generalmente hasta 15) y solo se puede obtener una variable a la vez.

3. **Requiere convertidores o pasarelas para integrarse con protocolos digitales modernos**

Para comunicar HART con sistemas Ethernet industriales (como PROFINET o Modbus TCP), es necesario usar dispositivos intermedios (gateways).

4. **Comunicaciones no deterministas**

No garantiza tiempos de respuesta exactos, lo que lo hace menos adecuado para aplicaciones que requieren control en tiempo real.

5. **Obsolescencia tecnológica gradual**

Aunque sigue siendo muy utilizado, está siendo reemplazado por protocolos 100% digitales más rápidos y con mayores capacidades como **Foundation Fieldbus** o **WirelessHART**.

6. **Requiere herramientas específicas para configuración avanzada**

Si bien se puede hacer lectura básica fácilmente, para configuraciones avanzadas se necesita un comunicador HART o software especializado.

Variantes HART

WirelessHART

Es una extensión inalámbrica del protocolo HART, diseñada para operar en redes industriales sin cables. Fue estandarizado en 2007 bajo la norma IEC 62591.

📡 **Características:**

- Usa radiofrecuencia de 2,4 GHz (similar al Wi-Fi)
- Tecnología de red de malla (mesh): cada dispositivo actúa como nodo repetidor
- Comunicación segura con cifrado AES-128
- Totalmente compatible con dispositivos HART existentes (puede integrarse a través de Gateways)

🎯 **Aplicaciones típicas:**

- Lugares donde es difícil o costoso tender cables (plantas químicas, refinerías, áreas clasificadas)

- Monitoreo de variables ambientales: temperatura, presión, humedad
- Instalaciones temporales o móviles
- Sensores en movimiento o que requieren flexibilidad
- Expansiones de planta sin necesidad de cableado adicional

✅ **Ventajas:**
- Reducción de costos de instalación
- Mayor flexibilidad para reconfigurar redes
- Mejor cobertura en áreas remotas o de difícil acceso

HART-IP

Es una variante que permite encapsular mensajes HART para que puedan transmitirse a través de redes Ethernet estándar (TCP/IP).

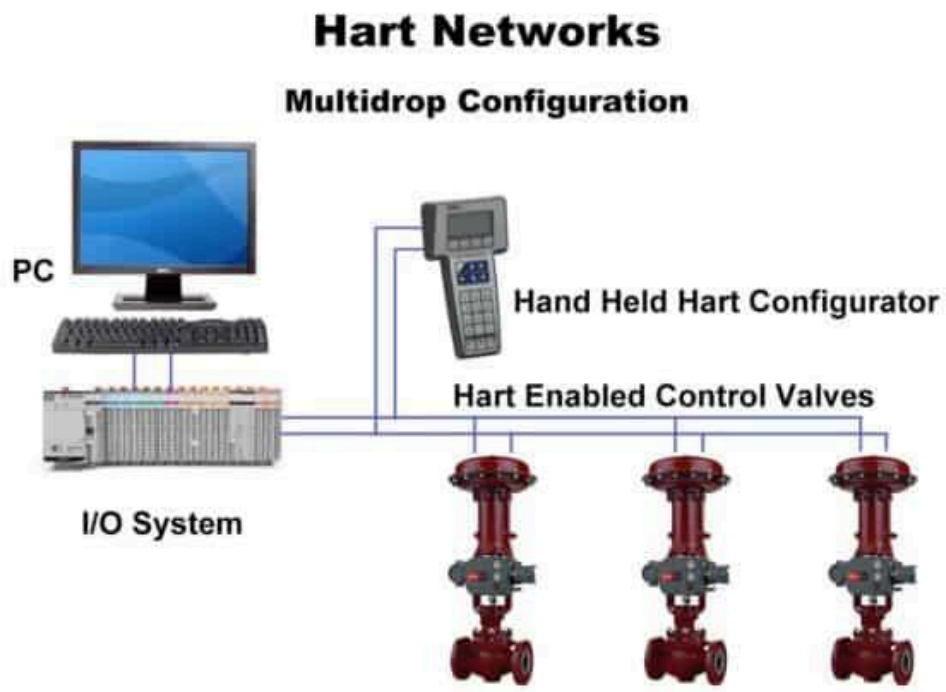
🌐 **Características:**
- Permite comunicación sobre redes Ethernet o incluso Internet
- Mantiene compatibilidad con comandos HART tradicionales
- Se usa para comunicarse con servidores, sistemas SCADA o software de mantenimiento

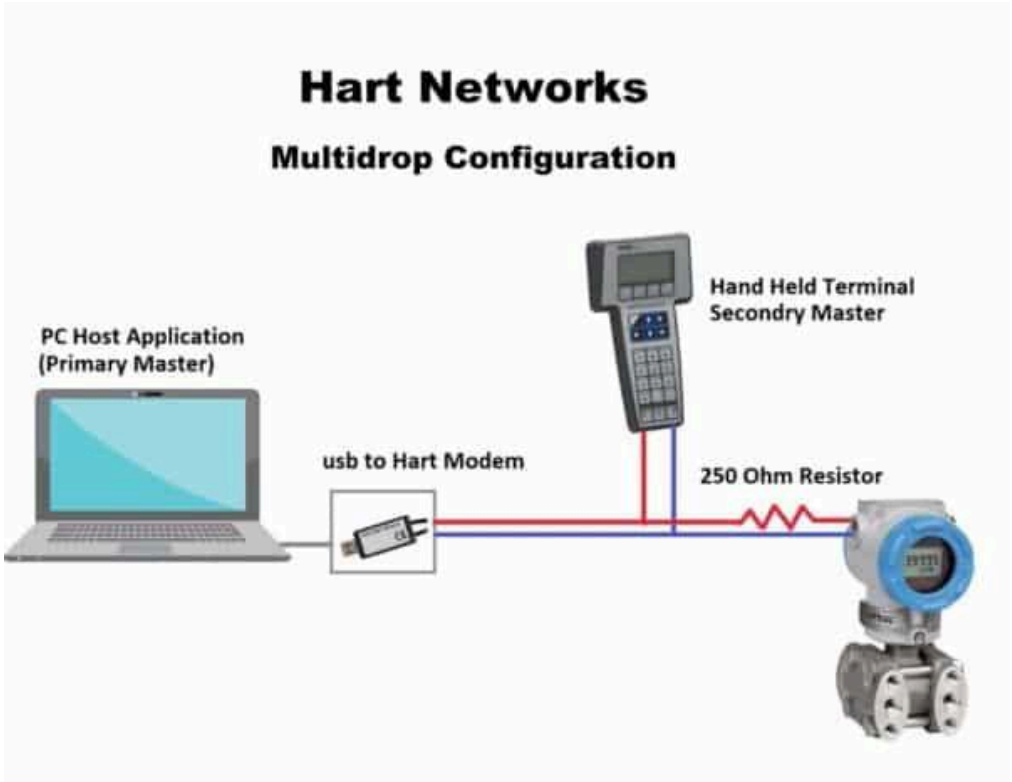
🎯 **Aplicaciones típicas:**
- Sistemas de supervisión centralizados en la nube o en redes LAN industriales
- Integración de dispositivos HART en arquitecturas digitales modernas
- Comunicación entre gateways y sistemas de control (como DCS o PLCs)
- Gestión remota de activos (Asset Management)

✅ **Ventajas:**
- Compatible con infraestructuras Ethernet
- Ideal para monitoreo remoto y diagnóstico avanzado
- Se integra fácilmente con sistemas modernos basados en IP

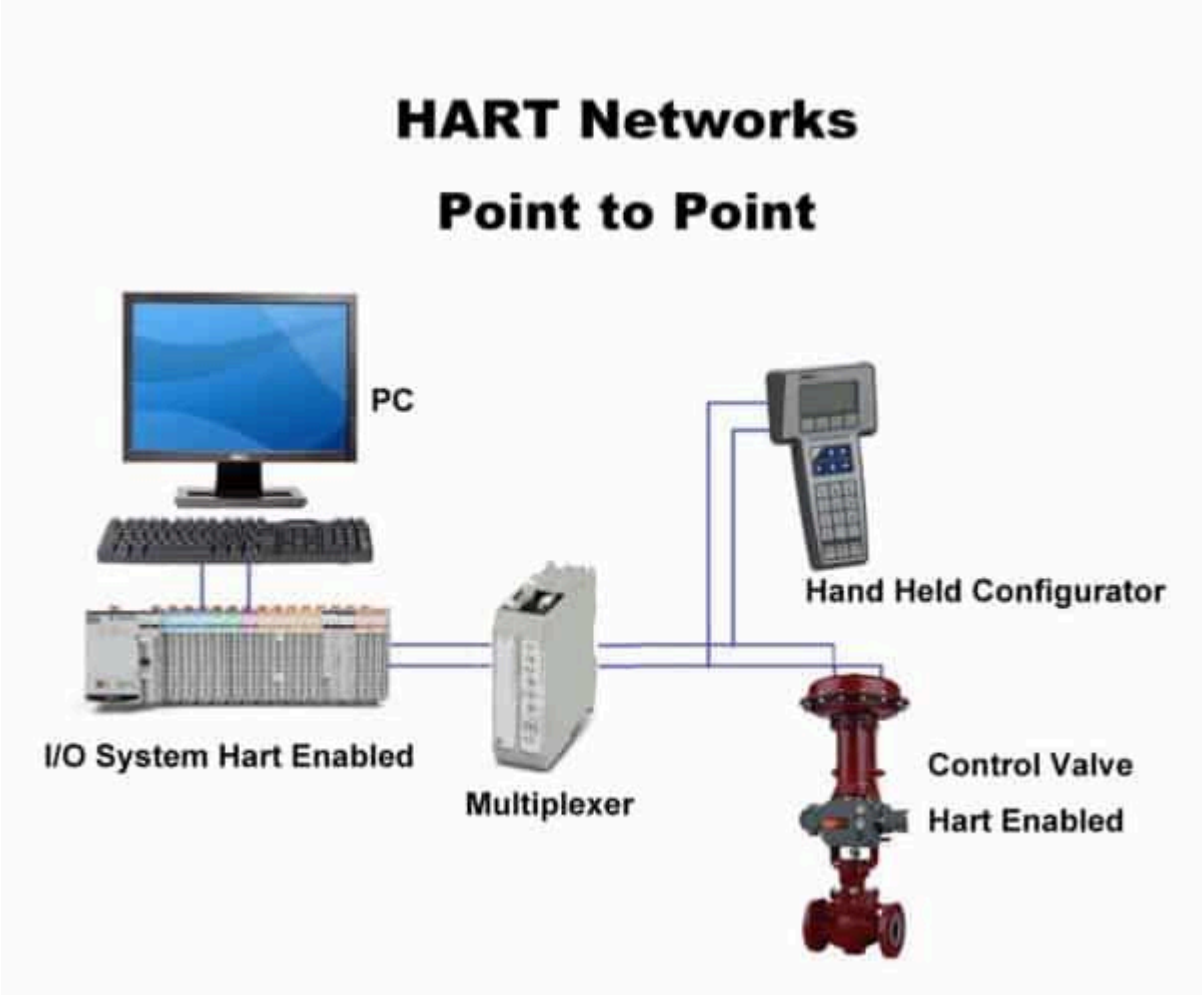
Diagrama de sistema

- Configuración MULTIDROP -





- Configuración Punto a Punto -



5) ¿Qué es un protocolo PROFINET?, ¿Para qué se usa? Ejemplifique

PROFINET (Process Field Network) es un estándar abierto de comunicación industrial basado en Ethernet. Fue desarrollado por la organización **PROFIBUS & PROFINET International (PI)**.
Este protocolo representa una evolución significativa de los sistemas de comunicación de campo tradicionales, adaptándolos a la conectividad moderna a través de Ethernet industrial.

Características fundamentales de PROFINET:

Comunicación basada en Ethernet: Utiliza la infraestructura Ethernet estándar pero con adaptaciones para entornos industriales.

Comunicación en tiempo real: esta ofrece tres niveles de rendimiento:

- TCP/IP estándar (NRT - Non-Real-Time): Para tareas sin requisitos temporales estrictos
- Tiempo real (RT - Real-Time): Para aplicaciones con tiempos de ciclo de 1-10 ms
- Tiempo real isócrono (IRT - Isochronous Real-Time): Para aplicaciones de alta precision con tiempos de ciclo inferiores a 1 ms

Arquitectura flexible: esta permite topologías en estrella, árbol, anillo o línea.

Diagnóstico integrado: tiene capacidades avanzadas para detección y resolución de problemas.

Usos principales de PROFINET

PROFINET se utiliza principalmente para:

Automatización industrial: Para conectar controladores (PLCs), dispositivos de campo, HMIs y otros componentes.
Control de procesos: En industrias como manufactura, automotriz, farmacéutica, alimentaria y química.
Control de movieiento: Para aplicaciones que requieren sincronización precisa de movimientos.
Integración vertical: Permite la comunicación desde el nivel de campo hasta el nivel empresarial.
Industria 4.0: Facilita la implementación de fábricas inteligentes y digitalización industrial.

Ejemplos prácticos de implementación

Línea de producción automotriz
En una línea de ensamblaje automotriz, PROFINET conecta:

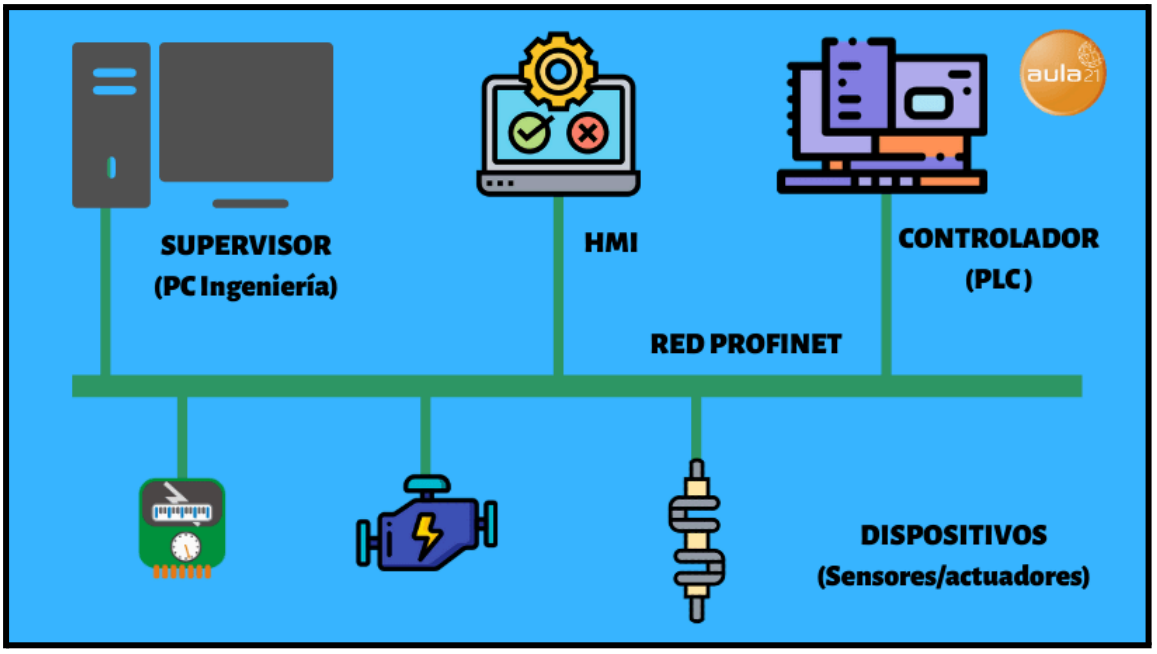
- PLCs que controlan diferentes estaciones de trabajo
- Robots de soldadura y ensamblaje
- Sistemas de visión para control de calidad
- Sensores y actuadores distribuidos
- HMIs para supervisión del proceso

Componentes de PROFINET

PROFINET clasifica los dispositivos en tres tipos: controladores, dispositivos y supervisores.

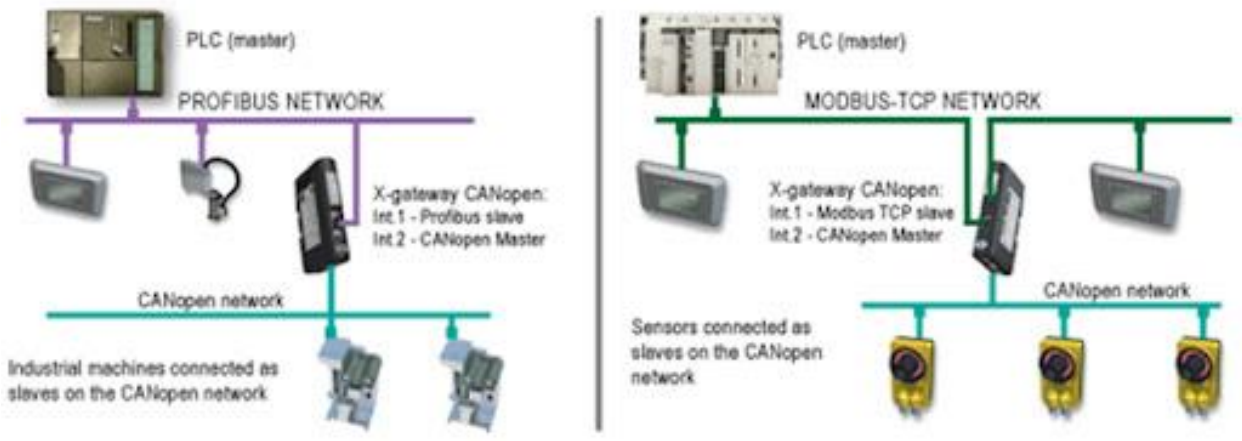
- Los controladores son dispositivos que ejecutan un programa de automatización e intercambian datos con los dispositivos.
- Los dispositivos son sensores/actuadores conectados al controlador a través de Ethernet.
- Los supervisores son un HMI, un PC u otros dispositivos de puesta en marcha, monitoreo o análisis de diagnóstico.

Diagrama Básico de PROFINET



6) ¿Qué es un protocolo CANopen?, ¿Para qué se usa? Ejemplifique

CANopen es una especificación desarrollada por la agrupación CiA (CAN in Automation) sobre la base física y de enlace de datos del bus CAN clásico, ampliable a CAN FD. Diseñado para sistemas embebidos y de automatización con exigencias de determinismo, robustez y trazabilidad, define perfiles de comunicación y de dispositivo que permiten interoperabilidad entre nodos heterogéneos.



Capas del modelo OSI que abarca

- Capa física: Utiliza CAN clásico (hasta 1 Mbit/s) o CAN FD (payload aumentada y tasas superiores).
- Capa de enlace de datos: Arbitraje por CSMA/CR (bitwise), bit-stuffing, CRC de 15 bits, señal de AC
- Capa de aplicación (CiA 301):
  - Perfil de comunicación: Define los servicios (SDO, PDO, NMT, SYNC, Heartbeat, Emergency).
  - Perfiles de dispositivo (CiA 4xx): Estandariza la funcionalidad de diferentes tipos de nodos (E/S, drives, sensores).

Así como otros protocolos de red industriales, se diseñan para hacer uso únicamente de las capas o elementos necesarios para la tarea concreta para la que están desarrollados.

Componentes Clave

Elemento	Descripción técnica
Object Dictionary (OD)	Estructura en memoria (índice–subíndice) que lista todos los parámetros (tipos y valores por defecto).
Service Data Object (SDO)	Servicio acíclico para acceso a cualquier entrada del OD (lectura/escritura puntuales).

Process Data Object (PDO)	Intercambio de datos de proceso (hasta 8 bytes) en modo cíclico o acíclico, con baja latencia.
Network Management (NMT)	Control de estados de nodo: Pre-operational, Operational, Stopped; permite arranque/parada global.
Heartbeat & Life-Guarding	Publicación periódica (1–1000 ms) del “estoy vivo”; detección de fallos de nodo mediante timeout.
Emergency Messages	Mensajes espontáneos ante errores críticos (2 bytes de código de fallo + datos contextuales).

Para que se utiliza?

- Automatización industrial
- Control distribuido de ejes y servomotores.
- Redes de E/S distribuidas en plantas y máquinas.

- Robótica
- Sincronización de múltiples articulaciones usando SYNC y CSP (Cyclic Synchronous PDO).

- Vehículos especializados
- Agricultura y construcción (CANopen FD para mayor throughput y trazabilidad).

- Edificios inteligentes y medicina
- Control HVAC, iluminación, equipos clínicos con exigencias de disponibilidad y registro de fallos.

Ejemplo Práctico: Red de Servomotores

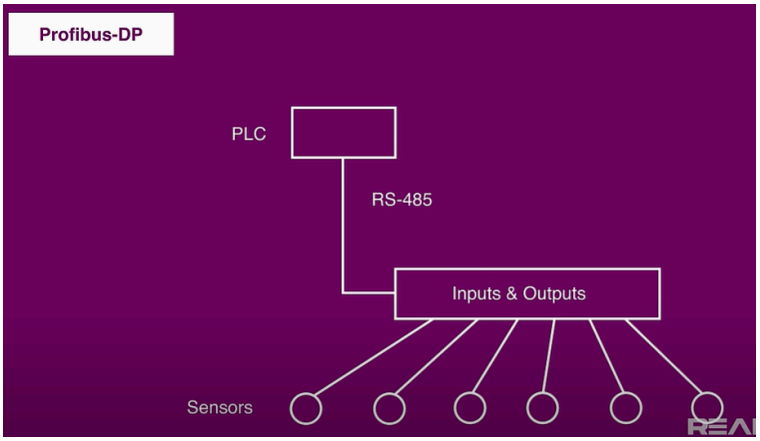
Se considera un sistema con un PLC maestro y cinco servos (nodos 1–5):

1) Inicialización (SDO)
   a) Escritura en OD de cada servo: límite de par (0x6072), offset de encoder (0x6000:0x01).
2) Intercambio de datos en tiempo real (PDO + SYNC)
   a) RPDO1 (maestro→servo): consigna de posición (4 bytes) + modo operativo (1 byte).
   b) TPDO1 (servo→maestro): posición real (4 bytes) + estado (1 byte).
   c) El maestro envía SYNC cada 1 ms; tras recibirlo, cada nodo actualiza/envía su PDO.
3) Gestión de emergencias
   a) Si un servo detecta sobretemperatura envía Emergency (0x3210); el maestro ejecuta NMT “Stopped” y activa alarmas.

7) ¿Qué es un protocolo PROFIBUS-DP/PA?, ¿Para qué se usa? Ejemplifique

PROFIBUS corresponde a un acrónimo de Process Field Bus. Un estándar de comunicación utilizado en automatización, común en entornos industriales. Estas estructuras de conexión cuentan con sensores/actuadores conectado a una sala de control donde se reciben los datos, y son gestionados vía un **Controlador Lógico Programable** (PLC) Este se encuentra definido en la norma (IEC 61158).

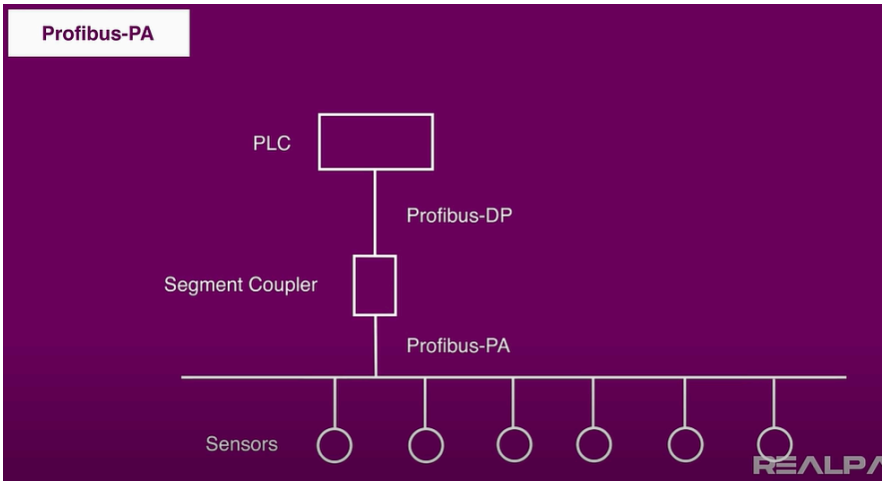
Si imaginamos que dentro de un entorno industrial donde se requieren no solo una enorme cantidad de sensores/actuadores, sino que estos se despliegan a lo largo de grandes distancias, se desarrolló un protocolo para optimizar la comunicación de los mismos vía nodos de I/O (entrada y salida de información) que se encuentren más próximos a los diferentes puntos de sensorización. Y que luego mediante un único cable bus **(RS-485)** conecta con el PLC. Estos nodos funcionan como ****Periféricos Descentralizados (DP)**** De esta forma se reduce drásticamente la cantidad de cable en el espacio físico.



Como evolución del protocolo PROFIBUS-DP. A fin de mejorar el rendimiento y economía del sistema, se utiliza un cable bus al que se conectarán todos los sensores/actuadores. Luego ese table bus se dirigirá hacia un acoplador direccional. Que luego conecta con el PLC mediante el protocolo **PROFIBUS-DP**. Los datos desde los sensores hasta el acoplador se realizan mediante el protocolo **PROFIBUS-PA.**

- Ejemplo de un acoplador direccional

- Ejemplo de estructura red del protocolo **PROFIBUS-PA**.



Arquitectura y relación con la capa OSI:

capas 3–6 no se utilizan

Capa 1: Física

- **DP:** RS-485 diferencial, hasta 12 Mbps.
- **PA:** MBP (Manchester Bus Powered), 31.25 kbps, alimentación sobre el par de datos, apto para zonas peligrosas

Capa 2: Enlace de datos

- Gestión de acceso al medio (token/bus)
- Control de errores (FCS de 16 bits)

- Formatos de trama (maestro a esclavo, esclavo a maestro, broadcast)

Capa 7: Aplicación

- PROFIBUS-DP V0/V1/V2: intercambio de datos cíclicos, diagnóstico, alarmas y acoplamiento de bloques de datos grandes

Ejemplo de uso PROFIBUS-DP
En una línea de montaje automotriz, un PLC maestro envía cada 2 ms valores de salida al módulo de E/S distribuida (esclavo) para:

- Activar solenoides de colocación de tornillos.
- Leer estados de detectores de presencia.
- Si falla un módulo, el PLC detecta el error gracias al diagnóstico cíclico (DP-V1) y genera una alarma en el HMI

Ejemplo de uso PROFIBUS-PA
En una planta química, se instalan transmisores de presión y temperatura en reactores:

- Cada transmisor PA envía mediciones cada 200 ms a la estación de control.
- El bus, intrínsecamente seguro, circula por zonas con atmósfera explosiva.

Característica	PROFIBUS-DP	PROFIBUS-PA
Velocidad	9.6 kbps a 12 Mbps (configurable por segmento)	31.25 kbps (Manchester bit-coding)
Topología	Línea principal (bus line) con terminadores en ambos extremos; repetidores para segmentar	Línea o árbol con segmentos de hasta 1 900 m; repetidores galvánicos para separación
Máster-Esclavo	Uno o varios maestros (PLC), hasta 126 nodos esclavos	No aplica concepto maestro/esclavo tradicional; se usa acoplador DP/PA para integrar con un bus DP maestro/esclavo

Ciclicidad y servicios acíclicos	- **DP-V0**: Sólo datos de proceso cíclicos - **DP-V1**: Añade servicios acíclicos (diagnóstico, parametrización) - **DP-V2**: Transferencia de bloques largos, sincronización de maestros	—
Formato de telegrama	`[BREAK][SYNC][START][ADDR][CT][DATA…][FCS]` • BREAK ≥ 33 µs nivel “marcado” • SYNC: 0xA5 • START: formato (p.ej. 0x10) • ADDR: 0–126 • CT: control (longitud) • DATA: payload • FCS: CRC-16 ($x^{16}+x^{12}+x^5+1$)	—
Alimentación sobre bus	No	Sí: mismo par de cables transporta datos y hasta 320 mA de alimentación
Seguridad intrínseca	No	Sí: permite instalación en zonas ATEX
Integración con DP	—	Mediante acoplador DP/PA para conectar segmentos PA de campo a un bus DP de control

Cuadro Comparativo de Protocolos Industriales y IoT

Protocolo	Tipo de Protocolo	Arquitectura	Medio de Transmisión	Velocidad	Número de Dispositivos	Aplicaciones Típicas	Complejidad	Tiempo Real	Seguridad	Ventajas	Desventajas
CoAP	IoT (RESTful)	Cliente-Servidor	UDP, TCP, SMS, Bluetooth	Baja a Media	Depende de la red IP	IoT restringido, Smart Devices	Simple	No determinístico	DTLS (opcional)	Ligero, eficiente para IoT	Seguridad sin DTLS, adopción
AMQP	Middleware (Mensajería)	Broker-Centrado	TCP	Media a Alta	Prácticamente ilimitado	Mensajería empresarial, sistemas distribuidos	Media a Alta	Configurable (QoS)	TLS/SSL, SASL	Robusto, flexible, confiable	Mayor complejidad, overhead
MODBUS	Industrial	Maestro-Esclavo	Serial, Ethernet	Baja a Alta	Hasta 247 (serial) / Ilimitado (Ethernet)	Automatización general, SCADA	Simple a Media	No determinístico (típicamente)	Limitada (serial) / Variable (Ethernet)	Ampliamente soportado, simple, bajo costo	Baja velocidad (serial), seguridad
HART	Industrial (Híbrido)	Maestro-Esclavo (Punto a Punto)	Bucle de corriente (4-20 mA)	Baja	Punto a punto	Instrumentación de procesos, diagnóstico	Simple	No determinístico	Limitada	Superposición analógica+digital, diagnóstico	Baja velocidad, infraestructura analógica
PROFINET	Industrial	Maestro-Esclavo, Peer-to-Peer	Ethernet	Alta	Prácticamente ilimitado	Automatización de fábricas, control de movimiento	Compleja	Determinístico (con IRT)	Amplias opciones	Alto rendimiento, integración IT, robusto	Mayor complejidad, costo
CANopen	Industrial	Maestro-Esclavo, Peer-to-Peer	CAN Bus	Baja a Media	Hasta 127	Control de maquinaria, robótica	Media	Determinístico (configurable)	Depende de la implementación	Robusto para entornos distribuidos	Complejidad de configuración
PROFIBUS-DP/PA	Industrial	Maestro-Esclavo	Serial (RS-485), Bucle de corriente	Baja a Media	Hasta 126 (DP)	Automatización de fábricas y procesos	Media	Determinístico (con extensiones)	Depende de la implementación	Amplia base instalada, versátil	Requiere terminación, sensibilidad al ruido (DP)