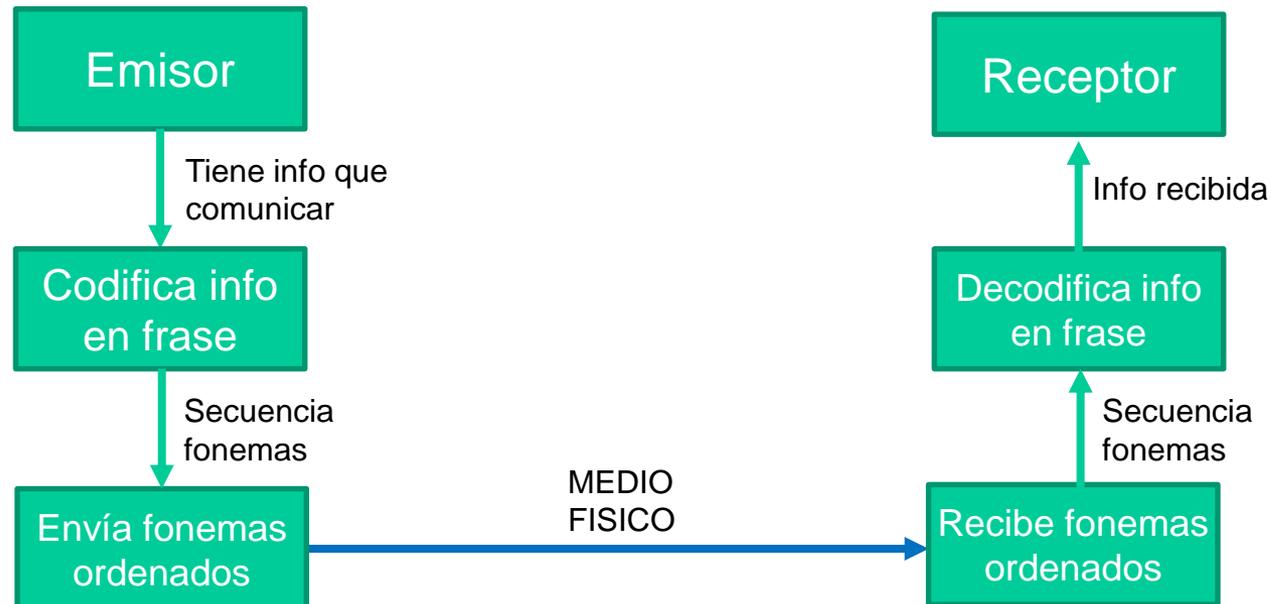


Comunicaciones

□ Comunicación:

- Permite enviar informaciones desde un emisor a un receptor
- Emisor y receptor pueden (y suelen) intercambiar sus papeles
- La información se codifica en tramas: secuencias de datos entendibles por ambos
- Es necesario un protocolo: ¿cuándo enviar/recibir? ¿qué hay que responder? ¿cómo se codifica/decodifica cada trama?

COMUNICACIÓN HUMANA ORAL

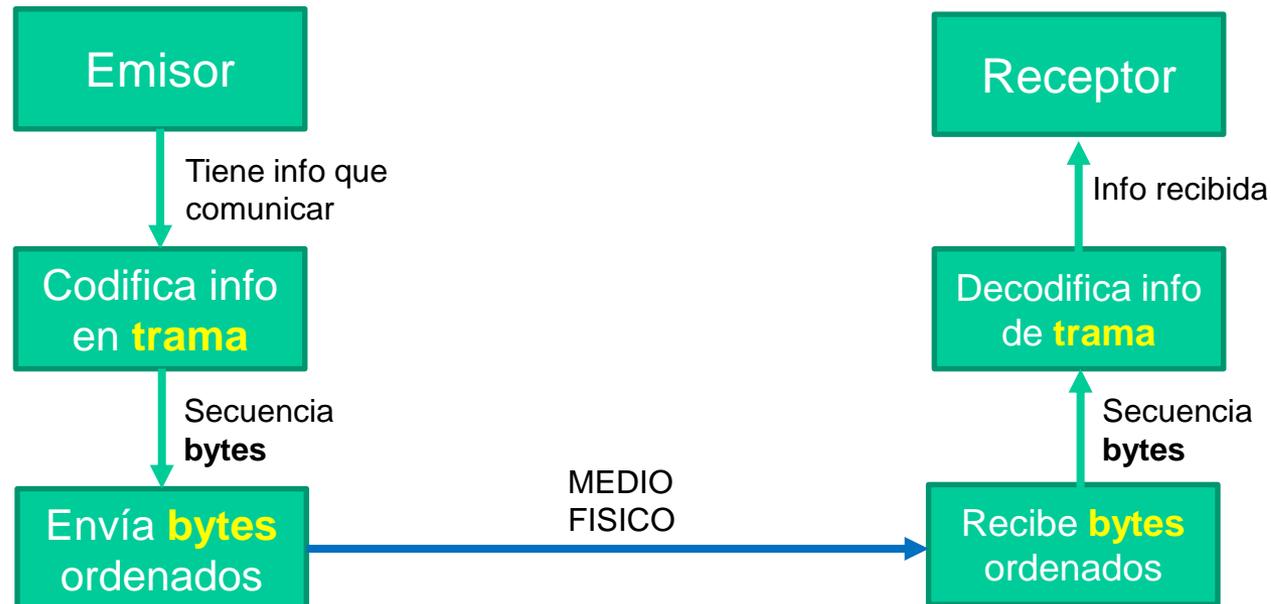


Comunicaciones

□ Comunicación:

- Permite enviar informaciones desde un emisor a un receptor
- Emisor y receptor pueden (y suelen) intercambiar sus papeles
- La información se codifica en tramas: secuencias de datos entendibles por ambos
- Es necesario un protocolo: ¿cuándo enviar/recibir? ¿qué hay que responder? ¿cómo se codifica/decodifica cada trama?

COMUNICACIÓN ENTRE COMPUTADORES





Comunicaciones

□ Comunicaciones:

- Mecanismo(s) de intercambio de información entre 2 ó más computadores conectados entre sí o a través de otros.

□ Terminología:

- **Trama:** unidad de información a transmitir (secuencia de bytes)
- **Medio:** elemento físico por el que circula la información, y características de dicho medio.
- **Interfaz:** conexión del computador al medio físico.
- **Emisor:** dispositivo/programa que genera y envía una trama.
- **Receptor:** dispositivo/programa que debe recibir y procesar una trama.
- **Cliente:** dispositivo/programa que inicia una secuencia de comunicación.
- **Servidor:** dispositivo/programa que espera una comunicación
- **Protocolo:** conjunto de todas las características de la comunicación para que sea viable (interfaz, codificación de tramas, control de errores, secuenciación de tramas, ...)



Comunicaciones

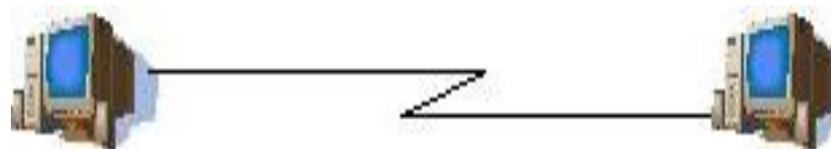
- Punto a punto: sólo hay un emisor y un receptor
 - Serie (RS-232,USB)
 - Paralelo
 - Inalámbrica (Bluetooth)

- En bus: varios emisores y receptores, todos conectados al mismo medio físico
 - Serie (RS-485, I2C, CAN)

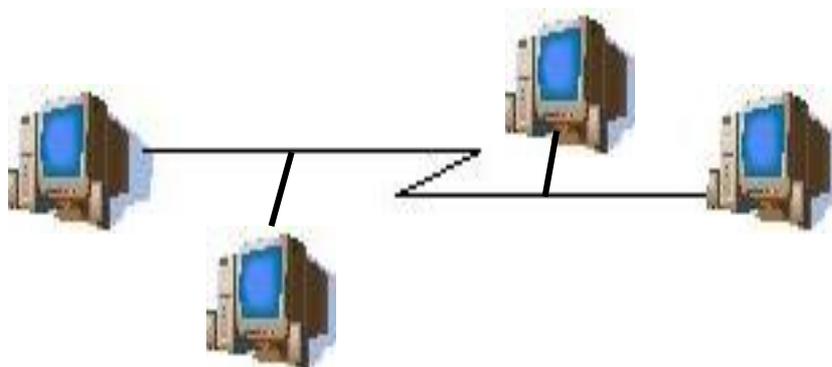
- En red: varios emisores y receptores, no todos conectados al mismo medio físico
 - Red cableada (Ethernet)
 - Red inalámbrica (Wifi)

Comunicaciones

□ Punto a punto



□ En bus



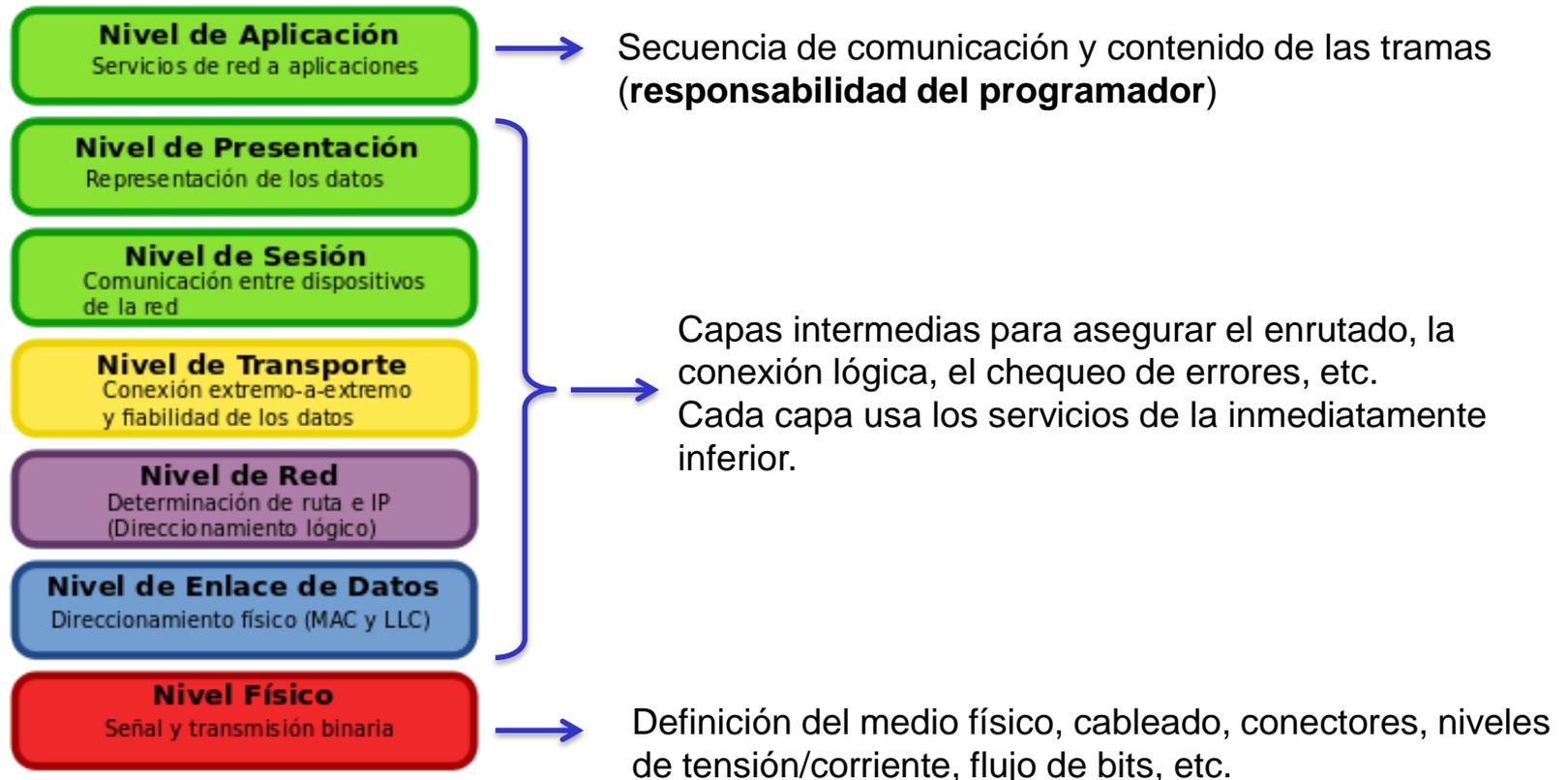
□ En red



Comunicaciones

□ Especificación de comunicaciones: modelo de capas (ISO/OSI)

LA PILA OSI





Comunicaciones

- ❑ La programación de comunicaciones a nivel de capa de aplicación:
 - Codificar en tramas los contenidos a transmitir.
 - Definir la secuencia de comunicación: establecimiento de conexión, envío/recepción, fin de conexión.
 - Identificar al receptor en comunicaciones multi-punto.
 - Establecer protocolo de acceso al medio si no existe en las capas inferiores.
 - Establecer método de detección/corrección de errores si no existe en las capas inferiores.
- ❑ Equipos “pequeños” (sin S.O.): programación del interfaz a nivel de puertos e interrupciones.
- ❑ Equipos “grandes” (con S.O.): programación a nivel de dispositivos de E/S.



Comunicaciones

□ Protocolos estándar:

- HTTP
- MODBUS (RTU/TCP)
- CANopen
- EtherCAT
- ProfiBus
- GigE Vision
- ...

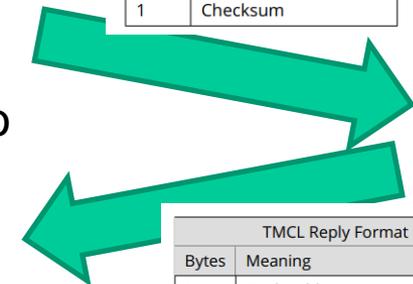
□ Protocolos específicos:

□ Protocolos ad hoc

TMCL Command Format	
Bytes	Meaning
1	Module address
1	Command number
1	Type number
1	Motor or Bank number
4	Value (MSB first!)
1	Checksum

Maestro

Esclavo



TMCL Reply Format	
Bytes	Meaning
1	Reply address
1	Module address
1	Status (e.g. 100 means no error)
1	Command number
4	Value (MSB first!)
1	Checksum



Comunicaciones serie

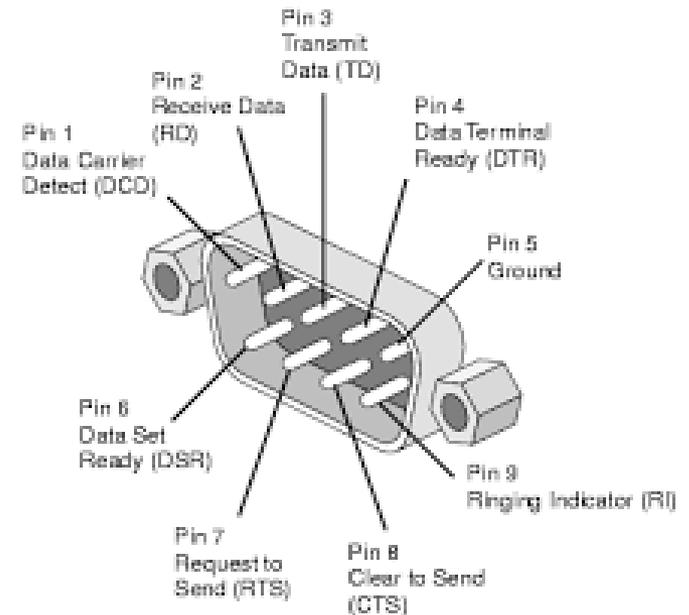
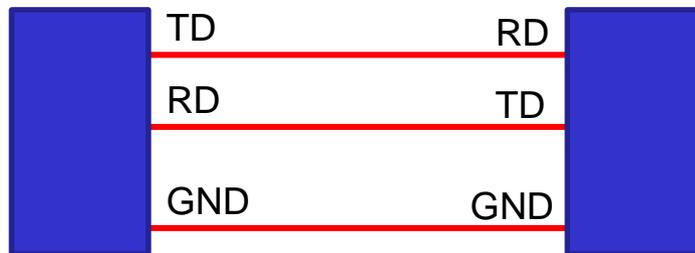
- Protocolos estándar de comunicación serie:
 - UART (RS-232, RS-485): largas distancias (metros o cientos de metros), baja velocidad (menor que 1Mbps)
 - I2C, SPI: pequeñas distancias (cm), alta velocidad (decenas de Mbps)
 - USB
 - Otros (industriales): CAN, ProfiBus, ...

- Características principales:
 - Síncrono/asíncrono (con sin señal de reloj)
 - Velocidad (bits por segundo)
 - Selección de esclavo
 - ...

Comunicaciones serie tipo UART

- ❑ Todos los bits se transmiten por un mismo hilo, uno después de otro
- ❑ Sin niveles intermedios entre la capa de aplicación y el medio físico: todo es responsabilidad del programador
- ❑ Estándares más usados:
 - RS-232: punto a punto
 - RS-485: multi-punto

RS-232: configuración mínima



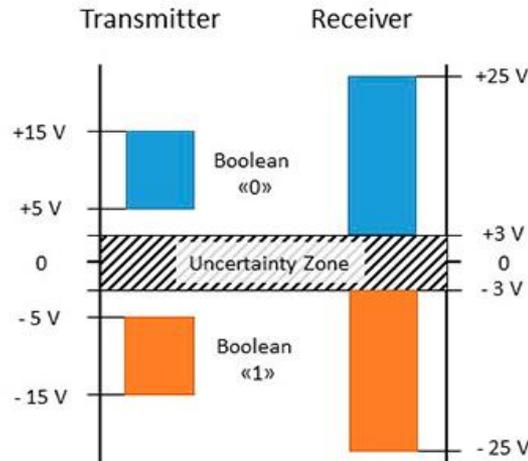
Las señales RS-232 son single-ended (referidas a masa común)



Comunicaciones serie RS-232

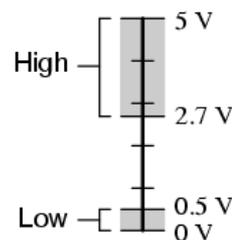
ATENCIÓN: *ambos equipos tienen que usar los mismos niveles de tensión para las señales*

□ Niveles de tensión estándar original

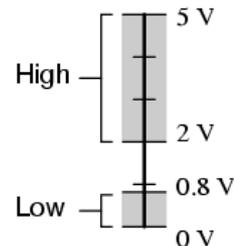


□ Niveles de tensión equipos pequeños (TTL)

Acceptable TTL gate output signal levels

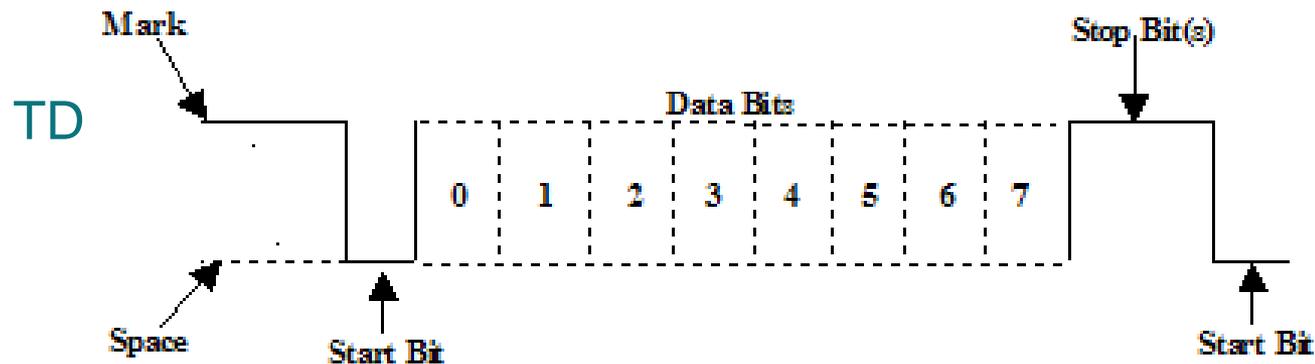


Acceptable TTL gate input signal levels



Comunicaciones serie RS-232

- ❑ La línea de transmisión TD se mantiene a 1 si no hay datos que enviar
- ❑ Cada equipo usa su propio reloj
- ❑ Antes del comienzo de datos, se baja la línea TD a 0 durante un periodo (start)
- ❑ Se envían los bits de un dato a impulsos del reloj
- ❑ Al finalizar un dato, se envía un bit de paridad (opcional) y uno o varios de stop (línea a 1)





Comunicaciones serie RS-232

- Los programas de los dos equipos conectados tienen que utilizar los mismos parámetros de comunicación:
 - Baudrate: velocidad de reloj (bits/segundo)
 - ✦ Valores normalizados: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600
 - Número de bits de datos: 5 a 8
 - Uso de paridad: par (even), impar (odd), ninguna (none)
 - Número de bits de stop: 1, 1.5 ó 2
 - Uso de líneas auxiliares para sincronización hardware: RTS/CTS (Request To Send/Clear To Send)
 - Uso de sincronización software: bytes especiales XON/XOFF



Comunicaciones serie RS-232

- ❑ Aunque es posible programar la comunicación con el estándar C (FILE* fid, fopen(), fclose(), ...):
 - Este juego de funciones no admite cambiar los parámetros de la comunicación
 - No existe la posibilidad de controlar las líneas auxiliares

- ❑ Más habitual con funciones no estándar:
 - Windows:
 - HANDLE fid, CreateFile(), ReadFile(), WriteFile()
 - Linux:
 - int fid, open(), read(), write()
 - Funciones auxiliares para control de comunicación: ioctl(), fcntl(), tcgetattr(), tcsetattr(), tcflush()



Comunicaciones serie RS-232

- Ejemplo con comentarios:
Comunicación serie con Arduino Uno

<http://isa.uniovi.es/~ialvarez/Curso/infindycom/ejemplos/SerialPort.pdf>

Arduino Uno:

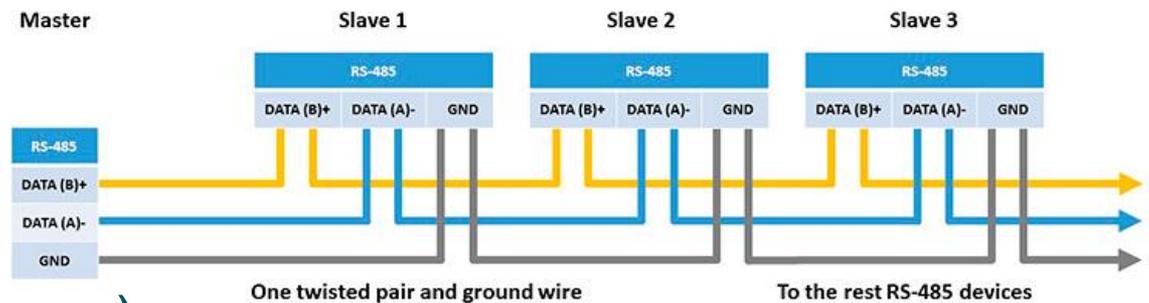
Microcontrolador programable en C/C++ con Arduino IDE (ojo, el main() está “escondido”)

```
main()
{
  setup();    // Función de usuario
  while (1)
    loop();  // Función de usuario
}
```

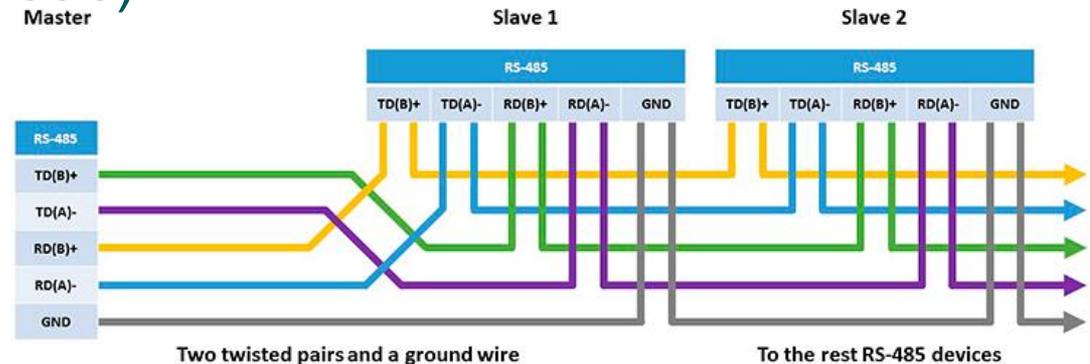


Comunicaciones en bus RS-485

- ❑ Estrategia maestro/esclavos
- ❑ Niveles de tensión diferencial (no single-ended)
- ❑ Half dúplex (dos líneas)



- ❑ Full dúplex (4 líneas)



- ❑ Necesarias resistencias de terminación (120Ω)



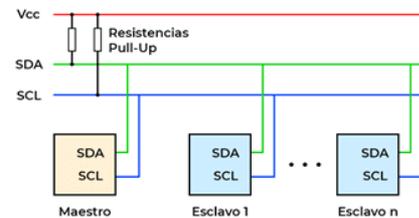
Comunicaciones en bus RS-485

- ❑ El maestro siempre inicia la comunicación
- ❑ Los esclavos deben tener un identificador único (número del 1 al 255)
- ❑ El 1^{er} byte transmitido por el maestro indica el id del esclavo que debe responder
- ❑ Una vez recibida la respuesta del esclavo, el maestro puede comenzar una nueva secuencia de comunicación

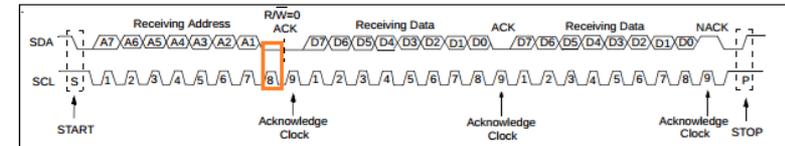
Comunicaciones serie I2C y SPI

- ❑ Para pequeñas distancias (del orden de cm) y alta velocidad (decenas de Mbps)
- ❑ Habituales con sensores y accionadores con electrónica integrada

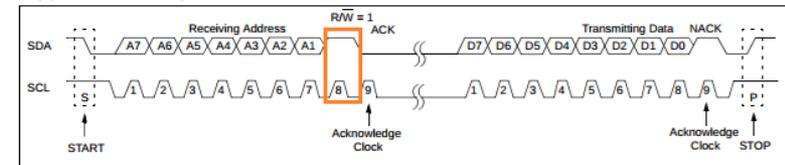
I2C



Operación de escritura I2C

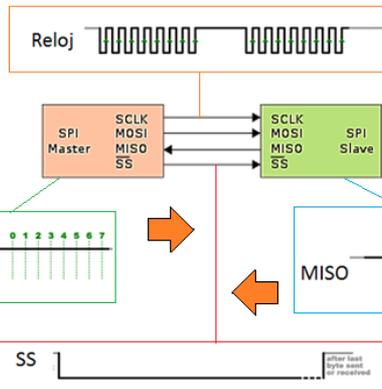


Operación de lectura I2C

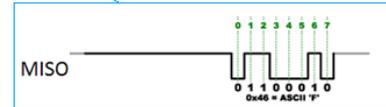


SPI

El dato a enviar es la letra S en ASCII



El reloj se activa para que envíe datos el maestro y el esclavo



El esclavo responde

Se pone en bajo el habilitador todo el tiempo que dure la comunicación



Comunicaciones en red TCP/IP

- ❑ **Redes de ordenadores:** grupo de ordenadores autónomos interconectados
 - Interconexión de dos o más ordenadores: situación en la que éstos son capaces de intercambiar información.
 - Autónomos: excluye aquellos casos donde existe una clara relación maestro/esclavo.

- ❑ Utilidades de las redes de Ordenadores para las empresas:
 - Compartición de recursos
 - Mayor fiabilidad
 - Reducción de costes
 - Mayor flexibilidad
 - Simplificación y agilización de comunicación

- ❑ Utilidades de las redes de Ordenadores para el público general:
 - Acceso a información remota
 - Una nueva forma de comunicación personal
 - Nuevas formas de entretenimiento

- ❑ Utilidades de las redes de Ordenadores para la industria:
 - Implementación del control distribuido



Comunicaciones en red TCP/IP

- ❑ Sistema de comunicación: conjunto de hardware y software que permite la comunicación entre estaciones.
- ❑ Funciones del sistema de comunicación:
 - Identificar las estaciones que conforman la red
 - Establecimiento de conexiones y multiplexación de canales
 - Control de errores durante la comunicación
 - Fragmentación y reconstrucción de los mensajes
 - Compactación de mensajes
 - Manejo de congestiones y control del flujo de la información
 - Sincronización
 - Establecimiento de distintos niveles de prioridad



Comunicaciones en red TCP/IP

□ Modelo de capas ISO/OSI

Modelo de referencia OSI Suite o Conjunto de protocolos de TCP/IP

Nivel	Función	Protocolo					
1	Aplicación	Telnet	FTP	TFTP	SMTP	DNS	
2	Presentación						
3	Sesión	TCP		UDP			
4	Transporte						
5	Red	IP	ICMP		RIP	OSPF	EGP
					ARP	RARP	
6	Enlace de datos	Ethernet	Token Ring		Otros medios		
7	Físico						

HTTP
SMTP
POP3
...



Comunicaciones en red TCP/IP

- ❑ Protocolo de red IP (Internet Protocol):
 - Identificación de un nodo en la red: **dirección IP**, formada por 4 bytes
 - Identificación de un programa usuario del nodo: **nº de puerto**

- ❑ Protocolo de transporte con conexión: TCP
 - Conexión punto a punto (dirección IP + nº puerto)
 - Conexión -> envío de datos -> cierre conexión
 - Conexión segura: el protocolo asegura que no hay pérdida de mensajes, el troceado, la reconstrucción y el orden.

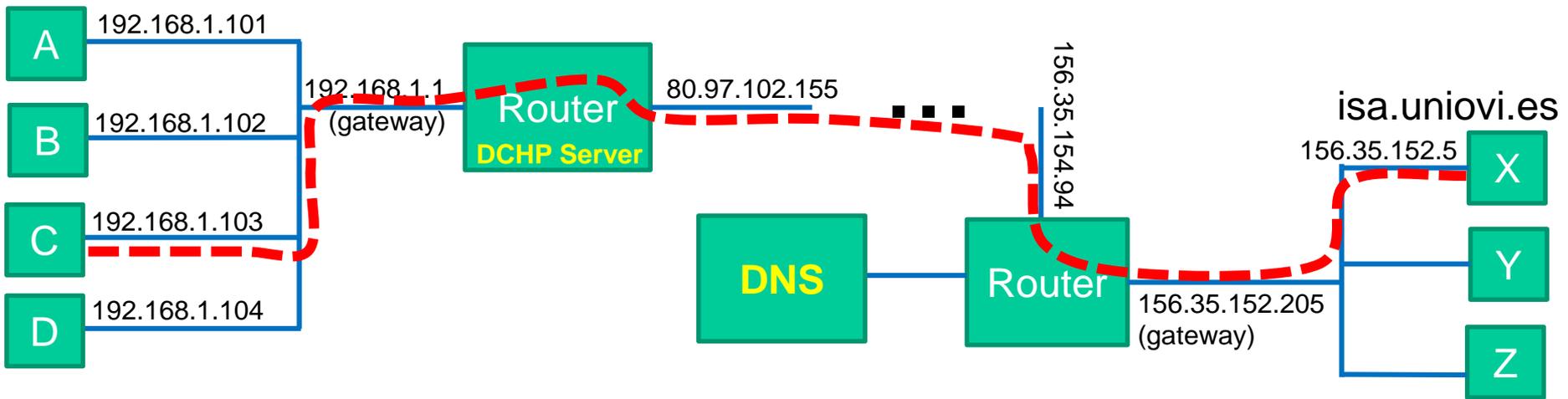
- ❑ Protocolo de transporte sin conexión: UDP
 - No hay conexión
 - Envío y recepción de datos a cualquier punto (dirección IP + nº puerto)
 - Envío y recepción no seguros



Comunicaciones en red TCP/IP

□ Protocolo de encaminamiento: IP

- Permite hacer llegar los paquetes a su destino a través de routers o encaminadores
- Dirección IP compuesta por un valor de 32 bits (se suele indicar como 4 números de 8 bits: x.x.x.x)
 - Estática: fija, para equipos que siempre se encuentran en el mismo lugar
 - Dinámica (DHCP): variable, para equipos que se conectan a routers variados (se precisa tener acceso a un servidor DHCP, normalmente el propio router)
- Servidor de nombres (DNS) como “agenda” para las direcciones IP



Comunicaciones en red TCP/IP

- Algunos equipos “especiales” en el protocolo IP
 - Servidor DNS: convierte nombres estilo texto (isa.uniovi.es) en direcciones ip (156.35.152.5)
 - Servidor DHCP: asigna direcciones IP dinámicas para equipos sin dirección fija (estática)

```
Símbolo del sistema
C:\Users\ialvarez>ipconfig /all

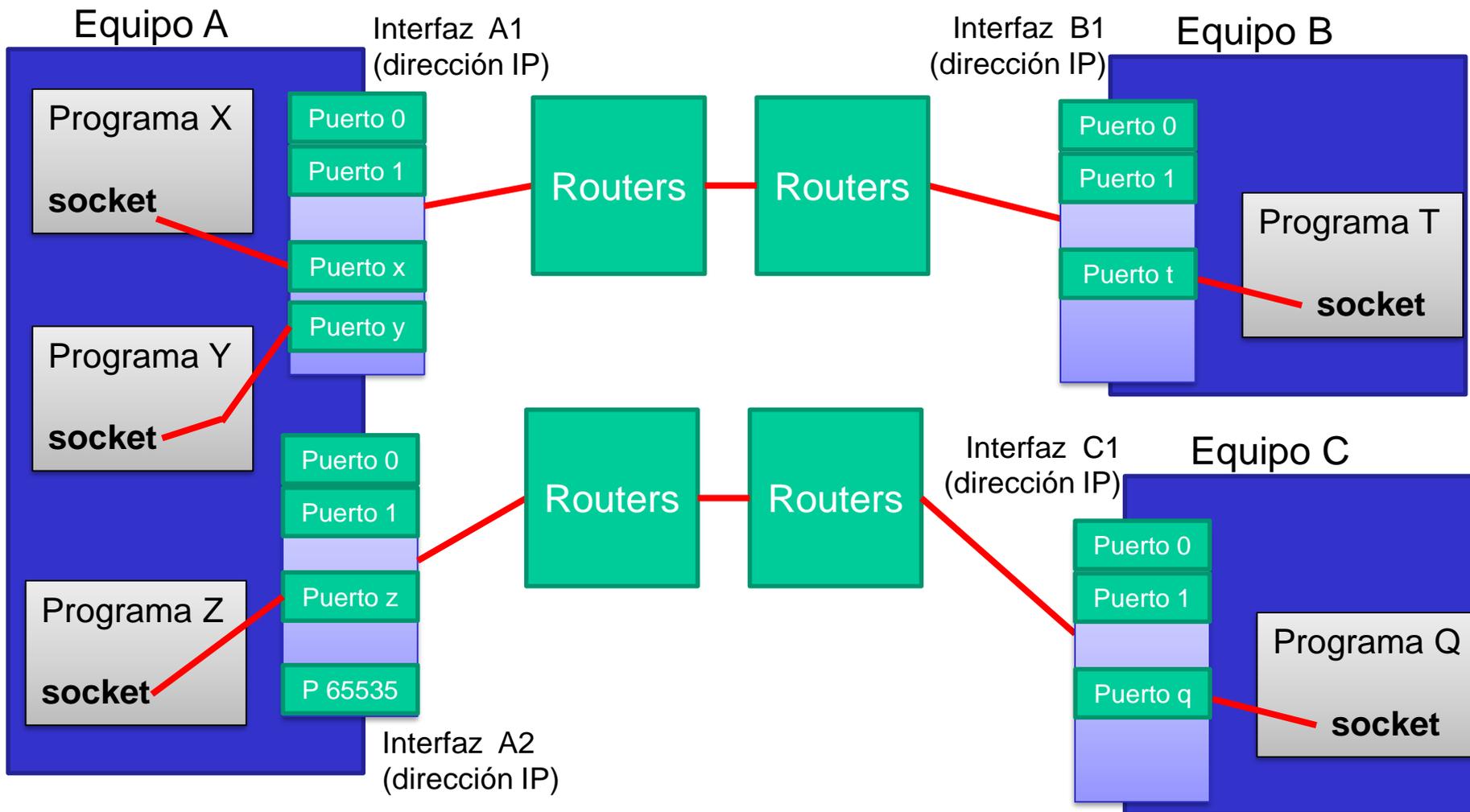
Configuración IP de Windows

Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . . :
    Descripción . . . . . : Intel(R) Dual Band Wireless-AC 8260
    Dirección física. . . . . : 1C-4D-70-0B-8D-44
    DHCP habilitado . . . . . : sí
    Configuración automática habilitada . . . : sí
    Vínculo: dirección IPv6 local. . . : fe80::e402:db9f:b964:d35d%14(Preferido)
    Dirección IPv4. . . . . : 192.168.1.38(Preferido)
    Máscara de subred . . . . . : 255.255.255.0
    Concesión obtenida. . . . . : sábado, 1 de diciembre de 2018 6:01:40
    La concesión expira . . . . . : domingo, 2 de diciembre de 2018 6:25:26
    Puerta de enlace predeterminada . . . . . : 192.168.1.1
    Servidor DHCP . . . . . : 192.168.1.1
    IAID DHCPv6 . . . . . : 236735856
    DUID de cliente DHCPv6. . . . . : 00-01-00-01-16-0D-49-10-7B-44-D9-B3-7A
    Servidores DNS. . . . . : 80.58.61.250
                                80.58.61.254
    NetBIOS sobre TCP/IP. . . . . : habilitado
```



Comunicaciones en red TCP/IP





Puertos IP

□ Puertos estándar:

- 20 FTP data (File Transfer Protocol)
- 21 FTP (File Transfer Protocol)
- 22 SSH (Secure Shell)
- 23 Telnet
- 25 SMTP (Send Mail Transfer Protocol)
- 43 whois
- 53 DNS (Domain Name Service)
- 68 DHCP (Dynamic Host Control Protocol)
- 79 Finger
- 80 HTTP (HyperText Transfer Protocol)
- 110 POP3 (Post Office Protocol, version 3)
- 115 SFTP (Secure File Transfer Protocol)
- 119 NNTP (Network New Transfer Protocol)
- 123 NTP (Network Time Protocol)
- 137 NetBIOS-ns
- 138 NetBIOS-dgm
- 139 NetBIOS
- 143 IMAP (Internet Message Access Protocol)
- 161 SNMP (Simple Network Management Protocol)
- 194 IRC (Internet Relay Chat)
- 220 IMAP3 (Internet Message Access Protocol 3)
- 389 LDAP (Lightweight Directory Access Protocol)
- 443 SSL (Secure Socket Layer)
- 445 SMB (NetBIOS over TCP)
- 666 Doom
- 993 SIMAP (Secure Internet Message Access Protocol)
- 995 SPOP (Secure Post Office Protocol)

□ Aplicaciones de usuario:

49151 a 65535

- **Ports 0–1023** – system or well-known ports.
- **Ports 1024–49151** – user or registered ports.
- **Ports >49151** – dynamic / private ports.



Comunicaciones en red TCP/IP

- ❑ **socket**: identificación de una conexión punto a punto entre 2 dispositivos conectados en red (dirección IP + puerto).

- ❑ Incluir `<socket.h>` (en Windows, “`winsoc2.h`”)

- ❑ Declaración de identificador:

```
int sock; (en Windows, SOCKET sock)
```

- ❑ 1^{er} paso: crear id. válido de socket (tipo `SOCK_STREAM` para TCP, tipo `SOCK_DGRAM` para UDP):

```
sock=socket(AF_INET,SOCK_STREAM,0);
```

- ❑ 2^o paso: asignar puerto local al socket:

```
struct sockaddr_in add_local;
```

```
... Rellenar campos de add_local: dirección IP, puerto  
(network order), protocolo
```

```
err = bind(sock,(struct sockaddr*) &add_local, sizeof(struct  
sockaddr_in));
```

- ❑ Si no ha habido errores, el socket ya puede ser utilizado para:
 - Conectar + enviar/recibir + desconectar (TCP)
 - Enviar/recibir (UDP)



Comunicaciones en red TCP/IP

□ Programación de sockets sin conexión (UDP)

□ Enviar datos:

```
ssize_t sendto(int sockfd, const void *buf, size_t len,  
              int flags, const struct sockaddr *dest_addr,  
              socklen_t addrlen);  
ssize_t send (int sockfd, const void *buf, size_t len, int flags);
```

□ Recibir datos:

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);  
  
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,  
                struct sockaddr *src_addr, socklen_t *addrlen);
```

□ Cerrar el socket:

```
close(sock);
```

- No hay seguridad de que los datos hayan llegado aunque `err==0`
- Si se envían varios mensajes, no hay seguridad del orden de llegada (pueden seguir rutas distintas)
- Si el mensaje es demasiado grande, es responsabilidad del remitente trocearlo y enviarlo, y del receptor componer los trozos.

Programación sockets TCP/IP

- ❑ Establecimiento de conexión: modelo cliente/servidor.
 - Cliente: solicitante de una conexión (ej. navegador web)
 - Servidor: receptor de una solicitud de conexión (ej. servidor de páginas web).
- ❑ Hasta que no se establezca una conexión no se pueden enviar y recibir datos
- ❑ Se asegura la llegada de los mensajes y su orden
- ❑ Se trocean y recomponen automáticamente los mensajes grandes.

LADO CLIENTE

- ❑ Solicitud de conexión:


```
struct sockaddr_in addr_servidor;
```

... Rellenar campos de addr_servidor ...

```
err=connect(sock, (struct sockaddr*)
&addr_servidor, sizeof(struct
sockaddr_in));
```
- ❑ Si no hay error, se ha establecido la conexión: se pueden enviar y recibir datos con send() y recv()
- ❑ Cerrar conexión con close()

LADO SERVIDOR

- ❑ Admitir solicitudes de conexión: listen()


```
err=listen(sock, n_conex_max);
```
- ❑ Esperar por una conexión: accept()


```
int conectado;
struct sockaddr_in addr_cliente;
int len=sizeof(struct sockaddr_in);

conectado=accept(sock, (struct
sockaddr*) &addr_cliente, &len);
```
- ❑ Si no hay error, se ha establecido la conexión con el socket 'conectado': se pueden enviar y recibir datos con send() y recv().
- ❑ Cerrar conexión con close(conectado);



Programación sockets TCP/IP

□ Más detalles y ejemplos:

- Ejemplo de cliente TCP para comunicar con un servidor de páginas web:

http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm

- Crea socket tipo SOCK_STREAM
- Bind a dirección local
- Conecta con servidor 156.35.152.5
- Envía/recibe datos según protocolo http
([https://es.wikipedia.org/wiki/Protocolo de transferencia de hipertexto](https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto))

- Documentación y ejemplos de pareja cliente/servidor TCP :

<http://isa.uniovi.es/~ialvarez/Curso/descargas/ProgramarWinsockQt.pdf>

- Para enviar/recibir datos según protocolo propio