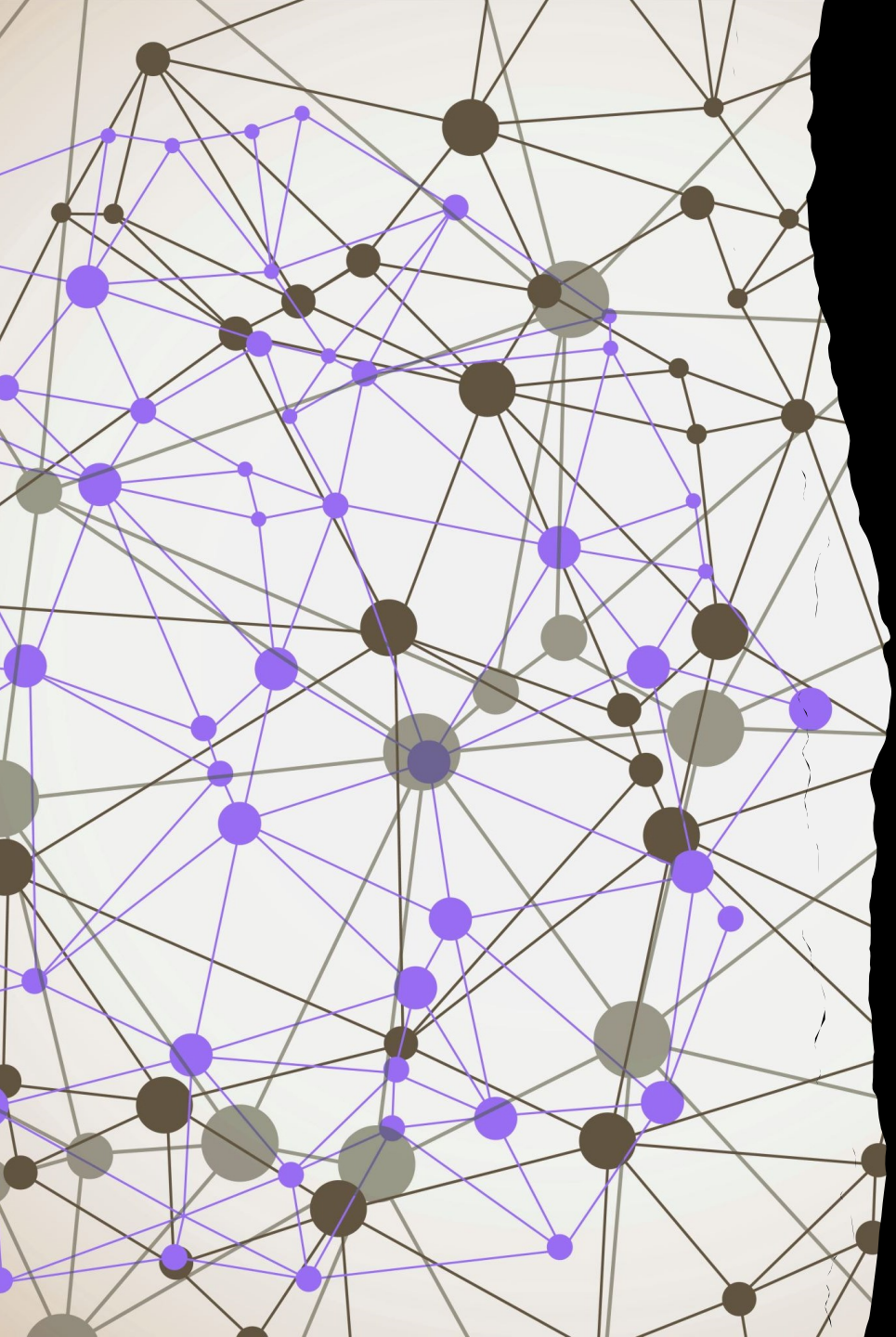


COMUNICAZIONE TRA DISPOSITIVI

MODBUS e protocolli CAN veicolari

OBIETTIVI DEL SEMINARIO

- Panoramica su alcuni protocolli di comunicazione tra dispositivi connessi via filo a lunga distanza:
  - Ethernet TCP/IP
    - Modbus Master/Slave
  - Linee seriali
    - RS485 Modbus Master/Slave
  - Altri protocolli basati su CAN layer 2.0 e altri supporti
- Considerazioni finali e scelte progettuali

MODBUS

- E’ un protocollo di comunicazione inizialmente basato su linee seriali sviluppato nel 1979 da Modicon (azienda ora parte del gruppo Schneider Electric) per mettere in comunicazione i propri controllori logici programmabili (PLC). È diventato uno standard de facto nella comunicazione di tipo industriale, ed attualmente è uno dei protocolli di connessione più diffusi al mondo fra i dispositivi elettronici industriali
- Esistono due versioni del protocollo:
  - su porta seriale (RS485 di default, ma anche RS232)
  - su Ethernet.
- E’ un protocollo di tipo master-slave, con un master e uno o più slave, il sito di riferimento è:
  - https://www.modbus.org/

MODBUS LA COMUNICAZIONE RS485

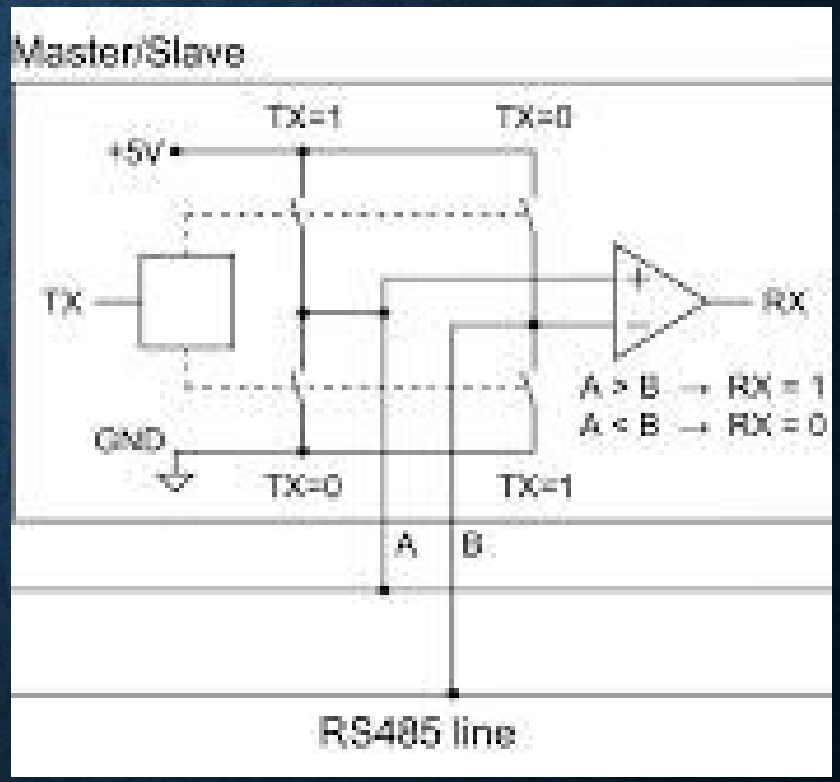
- Prendiamo in considerazione la comunicazione seriale RS485, la più diffusa, al punto tale che si trovano chip in grado di interfacciarsi direttamente alle UART dei microcontrollori e fornire i pin per la realizzazione della linea.
- RS485 è una specifica ISO/OSI a livello fisico di una connessione seriale a due fili, half-duplex e multipunto. Lo standard specifica un sistema di gestione del segnale in forma differenziale: la differenza tra la tensione presente sui due fili costituisce il dato in transito. Una polarità indica un livello logico 1, quella inversa indica il livello logico 0. La differenza di potenziale deve essere di almeno 0,2 V per un'operazione valida, ma qualsiasi tensione compresa tra +12 V e −7 V permette il corretto funzionamento del ricevitore.
- Cercando su Internet si ha una ampia documentazione sulle specifiche e accessori correlati:
  - Transceiver
  - Adattatori RS232->RS485, RS232->RS422
  - ….
- Potrebbe essere resa full duplex con una linea a 4 fili, ma per il protocollo Modbus non serve, in quanto il protocollo è half duplex, cioè il master prima trasmette, poi si mette in ricezione, lo slave selezionato nel messaggio trasmetterà la risposta, poi tornerà in ricezione. Uno slave non selezionato riceverà il messaggio, ma non essendo a lui diretto provvederà a scartarlo.

MODBUS LA COMUNICAZIONE RS485(1)

- SCEGLIERE LA VELOCITÀ CORRETTA
- La **capacità parassita** della linea di trasmissione aumenta all'aumentare della lunghezza della linea, limitando la massima velocità utilizzabile.
- Una legge empirica fornisce i seguenti valori:
  - 115200 bps: fino a 85m
  - 57600 bps: fino a 170m
  - 38400 bps: fino a 250m
  - 19200 bps: fino a 500m
  - 9600 bps: fino a 1000m
  - 8400 bps: fino a 1200m
- I valori possibili non sono così scarsi, date le dimensioni di un impianto industriale.

MODBUS LA COMUNICAZIONE RS485(2)

- Uno schema logico del transceiver è mostrato qui accanto
- Tra i fili A e B va inserita una resistenza, detta di terminazione, di 120 Ω



MODBUS LA COMUNICAZIONE ETH/TCP

- Nel corso del tempo il Modbus è stato arricchito dal supporto di rete fornito dal TCP/IP, in particolare su reti filari basate su Ethernet, ma non sono da escludere dispositivi che possano implementare il TCP su altri tipi di supporto, ma, essendo il protocollo orientato ai sistemi SCADA e/o controlli industriali, che sono, notoriamente, elettricamente sporchi, utilizzare connessioni a filo resistenti ai disturbi elettromagnetici (Ethernet su cavi RJ45 è il migliore) è fortemente raccomandato.
- Vedremo più avanti che il protocollo prevede un controllo di integrità di ogni singolo messaggio e, in caso di errore, si ha messaggio invalido e il recovery è a carico del programmatore:
  - Se c'è un errore di ricezione nello slave, se era indirizzato a lui risponderà con errore altrimenti non risponderà ed il master andrà in Time Out
  - Se c'è un errore di ricezione dal master, questo riceverà:
    - Framing error se è presente un problema di discriminazione degli 1/0
    - Un CRC16 error se non c'è match fra calcolato e ricevuto.

MODBUS LA STRUTTURA DEL MESSAGGIO

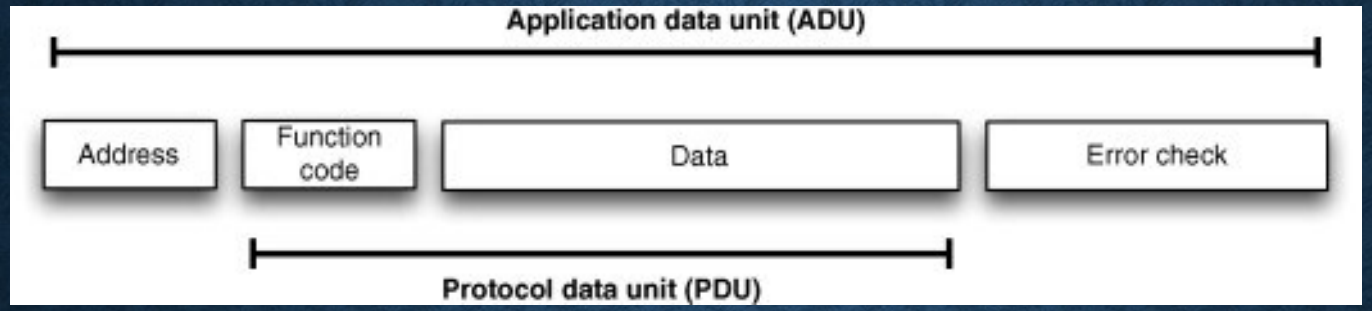
- Nel Modbus si distinguono due tipologie la RTU (remote terminal unit) e la ASCII
- Nella trattazione che seguirà tratteremo la modalità RTU, la più utilizzata il campo industriale.

	Modbus/ASCII		**Modbus/RTU**	
Characters	ASCII **0**...**9** and **A**..**F**		Binary **0**...**255**	
Error check	**LRC** Longitudinal Redundancy Check		**CRC** Cyclic Redundancy Check	
Frame start	character ':'		3.5 chars silence	
Frame end	characters **CR**/**LF**		3.5 chars silence	
Gaps in message	1 sec		1.5 times char length	
Start bit	**1**		**1**	
Data bits	**7**		**8**	
Parity	even/odd	none	even/odd	None
Stop bits	**1**	**2**	**1**	**2**

MODBUS LA COMUNICAZIONE ETH/TCP (1)

- Abbiamo già trattato gli aspetti della comunicazione TCP in altri seminari di questo Ordine ed in particolare il meccanismo di comunicazione basato sulle socket in cui si ha:
  - Un processo server, in ascolto su una porta, che entra in esecuzione a fronte della chiamata di un client
  - Un processo client che si connette ad un server
- E' intuibile, in questo caso, che i dispositivi slave fungeranno da server TCP ed i master da client TCP. E' da tenere presente che il lato full-duplex della connessione è utilizzato solo da TCP, mentre lo scambio messaggi sarà sempre half-duplex per il tipo di comunicazione «input after prompt», cioè lo slave risponderà se e solo se riceverà un messaggio dal master
- E' da sottolineare che nel caso di Modbus TCP non è prevista l'unicità del master (TCP client), un server TCP risponde all'IP chiamante e se si vuole evitare la ricezione da particolari IP occorrerà implementare un filtro sull'IP inviante da parte dello slave; rammentiamo che ogni nodo TCP avrà il suo IP e IP duplicati non sono ammessi.
- E' , altresì, da ricordare che il protocollo TCP non prevede un time-out nelle primitive di scambio (get, put, red e write) e pertanto, si dovrà necessariamente dare una risposta negativa per non mandare in hang-out il protocollo.

MODBUS LA STRUTTURA DEL MESSAGGIO(1)



- La struttura ADU si riferisce ai messaggi via RS485, la PDU ai messaggi via TCP
- Come si può notare nei messaggi PDU non è presente il campo Address, perché TCP, usando l'IP univoco sa individuare lo slave attraverso esso e non è presente il campo di error check, perché anche in questo caso l'integrità è assicurata dal TCP

MODBUS LA STRUTTURA DEL MESSAGGIO(2)

- Per le lunghezze dei singoli campi abbiamo:
  - Address 1 byte
  - Function code 1 byte
  - Dati N bytes
  - Error Check 2 bytes (CRC 16)
- I tipo di dato che sono trattati dal Modbus sono:
  - Binari 1 bit
  - Binari 16 bit

MODBUS LA STRUTTURA DEL MESSAGGIO(3)

- Le possibilità offerte dal Modbus sono
  - Scrittura/lettura di un registro ad 1 bit
    - Read/Write Coil (assimilabile ad un interruttore bistabile)
    - Read discrete input
  - Scrittura/lettura di un registro a 16 bit
    - Read/Write Holding register
    - Read 16 bit input
- Esistono poi comandi multipli, cioè in grado di manipolare più registri, ma vedremo più avanti nel dettaglio dei Function code

MODBUS LA VISIONE CONTROLLISTICA

- Nella slide precedente abbiamo dato un descrizione puramente informatica delle possibilità di comunicazione offerte dal Modbus, ma da ingegneri controllisti dovremmo darne un senso più orientato ai controlli che all'informatica.
- In ambito controllistico ha molto più senso parlare di comandi (Coil e Holding) e di input dell'esito del comando (discrete input 1 o 16 bit)
- Prendiamo in considerazione il coil, ma il ragionamento è equivalente per un Holding, che può essere sia scritto che letto, ma se deve rappresentare un comando (avvia un motore, commuta un relé,…) tale valore non può essere riletto per avere un feedback del comando prevedendo che lo slave lo modifichi, ma bisognerà prevedere un input register ad 1 bit che segnali l'esito del comando; qualora non ci dovesse essere congruenza (dopo un ragionevole tempo) sarà opportuno che il SW, che utilizza il Modbus, predisponga un allarme.
- La rilettura di un setpoint, coil o holding che sia, serve solo per rilevare la grandezza di errore con l'input associato, per poter dare attuazione alla regolazione (si pensi ad un regolatore PID).

MODBUS LA VISIONE CONTROLLISTICA(1)

- Alla luce della slide precedente, dobbiamo vedere il sistema Modbus, come un modo comunicativo per avere una mappa, sul master, di cosa avviene nel campo (field nel gergo controllistico) .
- Per attuare il precedente scenario, il progettista del SW dovrà prevedere:
  - una lettura ciclica dei registri di input
  - L'elaborazione dei dati ricevuti (attenzione, il Modbus non prevede l'invio su variazione, per cui il programmatore dovrà prevedere in autonomia)
  - Una scrittura di eventuali registri di output
- La ciclicità di questo processo sul master: input, elaborazione, output può essere anche a loop senza wait, dipende dal contesto e dal numero degli slave; potrebbe anche presentarsi il caso che l'elapsed time di repolling di uno stesso slave sia troppo lungo per il sistema che stiamo progettando ed in questo caso, nulla vieta di progettare un sistema multi thread con più linee RS485, una per thread, o di progettare sempre un multi thread, una per ciascuno slave presente.

MODBUS LA VISIONE CONTROLLISTICA(2)

- Dal punto di vista di uno slave, questi dovrà avere un SW che preveda la ricezione asincrona di un messaggio Modbus a fronte del quale elaborerà nuovi comandi e invii le risposte di conseguenza.
- Potrebbe capitare che, a fronte di un comando o di un malfunzionamento interno, lo slave non sia in grado di funzionare correttamente e/o trovarsi in uno stato di errore interno: Modbus, nel suo protocollo di scambio, non prevede segnalazioni di errore, per cui, qualora occorra, il progettista potrà optare per un input register segnalante lo stato, l'opzione di non rispondere e scatenare un timeout potrebbe essere accettabile solo per le linee RS485. Nelle ultime specifiche del protocollo è previsto un comando di richiesta stato (code 08x), ma dobbiamo sempre rammentare che, se non si invia tale messaggio, lo slave non ha modo di segnalare spontaneamente lo stato di errore.

MODBUS I MESSAGGI (PROTOCOLLO)

- L' elenco dei codici del protocollo è il seguente:
  - 01 (0x01) Read Coils
    02 (0x02) Read Discrete Inputs
    03 (0x03) Read Holding Registers
    04 (0x04) Read Input Registers
    05 (0x05) Write Single Coil
    06 (0x06) Write Single Register
    08 (0x08) Diagnostics (Serial Line only)
    11 (0x0B) Get Comm Event Counter (Serial Line only)
    15 (0x0F) Write Multiple Coils
    16 (0x10) Write Multiple Registers
    17 (0x11) Report Server ID (Serial Line only)
    22 (0x16) Mask Write Register
    23 (0x17) Read/Write Multiple Registers
    43 / 14 (0x2B / 0x0E) Read Device Identification
  - I codici in arancione sono codici ammissibili, ma non sempre implementati, mentre quelli in bianco sono obbligatori.
  - Negli esempi successivi useremo il formato messaggio corrispondente alla modalità ethernet, rammentando che nel caso di RS485, va aggiunto in testa il byte di indirizzo e in coda l'error checking
  - Faremo riferimento ai codici Hex, in quanto il protocollo ASCII implica messaggi più lunghi ed pochissimo usato.

MODBUS PROTOCOLLO COD 01X

- Il codice 01x (read coils) legge da 1 a 2000 coil, invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	01x
1	2	First coil addr.	0000x ->FFFFx
3	2	Input count	0001x ->07d0x

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	01x
1	2	Byte Count	(Coil Count+7)/8
3	N	Coil data	…

- Oppure un codice di errore, c he può dipendere da vari fattori, ma che in questa sede omettiamo, lasciando al lettore l'approfondimento su internet.

MODBUS PROTOCOLLO COD 02X

- Il codice 02x (read discrete inputs) legge da 1 a 2000 input discreti (bit), invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	02x
1	2	First inp. address	0000x ->FFFFx
3	2	Coil count	0001x ->07d0x

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	02x
1	2	Byte Count	(Inp. Count+7)/8
3	N	Input data	…

MODBUS PROTOCOLLO COD 03X

- Il codice 03x (read holding registers) legge da 1 a 125 holding register (16 bit), invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	03x
1	2	First inp. address	0000x ->FFFFx
3	2	Reg. count	0001x ->007dx

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	03x
1	2	Byte Count	Reg. Count*2
3	N	Reg. data	…

MODBUS PROTOCOLLO COD 04X

- Il codice 04x (read input registers) legge da 1 a 125 input register (16 bit), invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	04x
1	2	First inp. address	0000x ->FFFFx
3	2	Reg. count	0001x ->007dx

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	04x
1	2	Byte Count	Reg. Count*2
3	N	Reg. data	…

MODBUS PROTOCOLLO COD 05X

- Il codice 05x (write single coil) scrive un singolo coil verso il device, invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	05x
1	2	First coil address	0000x ->FFFFx
3	2	Coil value	0000x or FF00x

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	05x
1	2	First coil address	0000x ->FFFFx
3	2	Reg. data	0000x or FF00x

MODBUS PROTOCOLLO COD 06X

- Il codice 06x (write single register) scrive un singolo register verso il device, invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	06x
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. value	0000x-> FFFFx

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	06x
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. data	0000x or FFFFx

MODBUS PROTOCOLLO COD 0FX

- Il codice 0Fx (write multiple coil) scrive da 1 a 1968 coils verso il device, invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	0Fx
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. count	0000x-> 07B0x
4	1	Byte count	(Reg.Count+7)/8
5	Byte Count	Coil values	…

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	0Fx
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. count	0000x -> 07B0x

MODBUS PROTOCOLLO COD 10X

- Il codice 10x (write multiple register) da 1 a 123 registri verso il device, invierà:

Offset	Len (bytes)	Desc	Values
0	1	Function code	10x
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. count	0000x-> 007Bx
4	1	Byte count	00x-> 7Bx
5	N words	Register values	…

- Riceverà un messaggio:

Offset	Len (bytes)	Desc	Values
0	1	Function code	10x
1	2	First reg. address	0000x ->FFFFx
3	2	Reg. count	0000x -> 007Bx

MODBUS PROTOCOLLO ALTRO

- Nella letteratura specifica del Modbus si possono trovare i formati specifici per i comandi che abbiamo saltato, nonché il formato completo dei messaggi, sia per l'RS484 che per il formato TCP.
- Il paragrafo precedente ci pone una domanda tipica per lo sviluppatore:
  - Libreria SI
    - Molto del lavoro di impacchettamento e di invio/ricezione viene semplificato, come viene semplificato il controllo del CRC per l'RS485
  - Libreria NO
    - Occorrerà prepararsi tutto il necessario, con particolare attenzione per il CRC16 dell'RS485, poiché la matrice iniziale di configurazione è predefinita e va rispettata.
- Il mondo Modbus fornisce librerie per ogni genere di environment, esistono librerie DLL per il mondo Windows e .so/.shl per il mondo Linux, nonché per il mondo embedded, con particolare riferimento a librerie esclusivamente per gli slave, che è la casistica più diffusa per chi sviluppa embedded con Modbus.
- Sempre a proposito di librerie disponibili c'è da considerare che l'offerta, oltre ad essere varia per piattaforme, è molto varia anche in termini prezzi, opensource e/o freeware.

MODBUS DEBUGGING & TUNING

- Quando ci si accinge a sviluppare un dispositivo con protocollo Modbus, non sempre conviene e/o si ha a disposizione l'impianto completo, anzi è sempre meglio effettuare debugging e tuning in modalità standalone servendosi di un emulatore, la cui disponibilità presenta la stessa casistica delle librerie.
- L'utilizzo degli emulatori, oltre a permettere il debugging di tutte le combinazioni di dati e comandi possibili, ha a disposizione un tool di spy, cioè è possibile vedere il buffer completo di invio e di ricezione.

MODBUS USO DEI REGISTRI

- A questo punto dobbiamo considerare alcune caratteristiche, per meglio utilizzare questo bus di campo ed ottenere il massimo della flessibilità; teniamo presente che i registri sono individuati da un indice 16 bit unsigned e abbiamo comandi di scrittura/lettura multipli, cioè si può comandare e leggere più registri contemporaneamente purché siano contigui.
- Questa considerazione ci invita a numerare i registri, tenendo presente che può convenire raggrupparli per sfruttare le operazion a registri multipli, ad es. se si hanno più led da pilotare, con un solo write mutiple coil si possono gestire le accensioni con un comando solo.
- Un'altra considerazione da tenere molto presente è la temporizzazione (dispositivi master) per cui potrà essere possibile un invio personalizzato a istanti diversi; per gli slave, ovviamente, ricevuto il messaggio si processeranno tutti i ricevuti. Normalmente in uno slave, la libreria crea una tabella di indici<->valori che viene riempita/reinviata ad ogni ricezione e mette a disposizione funzioni di GET/PUT dei valori nei registri, poi un'unica routine di «process» provvederà a gestire l'I/O.

MODBUS USO DEI REGISTRI (1)

- Dobbiamo a questo punto fare presente che, in un sistema Modbus, potrebbero coesistere dispositivi slave acquistati a libero commercio con dispositivi progettati e realizzati custom.
- Ovviamente i dispositivi acquistati potrebbero non avere flessibilità nella numerazione dei registri, ma essere preconfigurati e, spesso, con indicizzazione non contigua, rendendo difficile l'utilizzo dei messaggi di lettura/scrittura multipli per poter arrivare ad un'unica richiesta verso quel dispositivo.
- L'ottimo per un sistema Modbus è che il master possa pollare con il minimo di richieste tutta la periferia collegata.

ALTRI PROTOCOLLI SU CAN

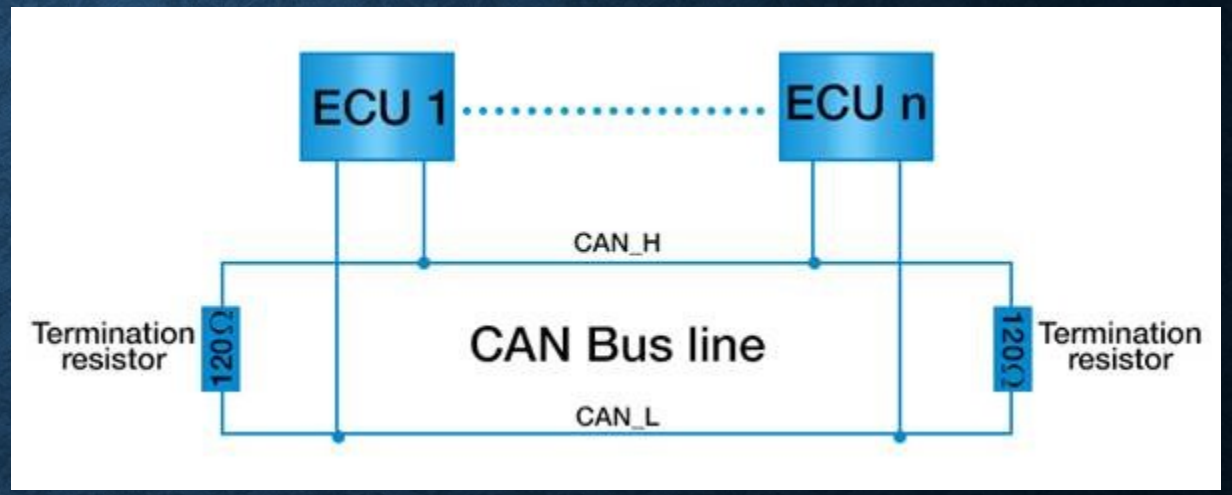
- Generalmente, quando si parla di CAN bus ci si riferisce al diffusissimo CANopen, riguardo al quale è già stata fatta ampia trattazione in un precedente seminario, tuttavia è da tenere presente che il Canopen è utilizzato sia nei veicoli, che in ambiente industriale; in ambito veicolare, ove stiamo per presentare una panoramica, il CANopen, quando presente, è inserito nel network attraverso un apposito bridge, convertitore di protocollo, che non può essere un dispositivo «dumb», ma corredato di apposito SW per convertire i dati dei PDO verso il protocollo veicolare.
- Il CAN bus è nato in casa Bosch proprio per la comunicazione dei dispositivi ad uso automotive. Qualsiasi moderno veicolo ha a bordo una o più linee CAN che interconnettono le varie ECU presenti
- La ragione di questa diffusione del CAN sui veicoli è praticamente imperniata sulla riduzione dei cablaggi per lo scambio di informazioni e sulla robustezza del protocollo.

CAN BUS

- Ideato negli anni ’80 in Germania dalla Robert Bosh GmbH, con lo scopo di collegare varie unità di controllo elettronico come sensori e centraline per veicoli.
- Nato principalmente per l’utilizzo in ambiente automotive, il protocollo CAN-bus è stato espressamente progettato per funzionare in ambienti fortemente disturbati dalla presenza di onde elettromagnetiche e, attualmente, è utilizzato in molte applicazioni industriali, dove è richiesto un alto livello di immunità ai disturbi.
- Il protocollo di comunicazione CAN è standardizzato ISO 11898-1 (2015)

CAN BUS (1)

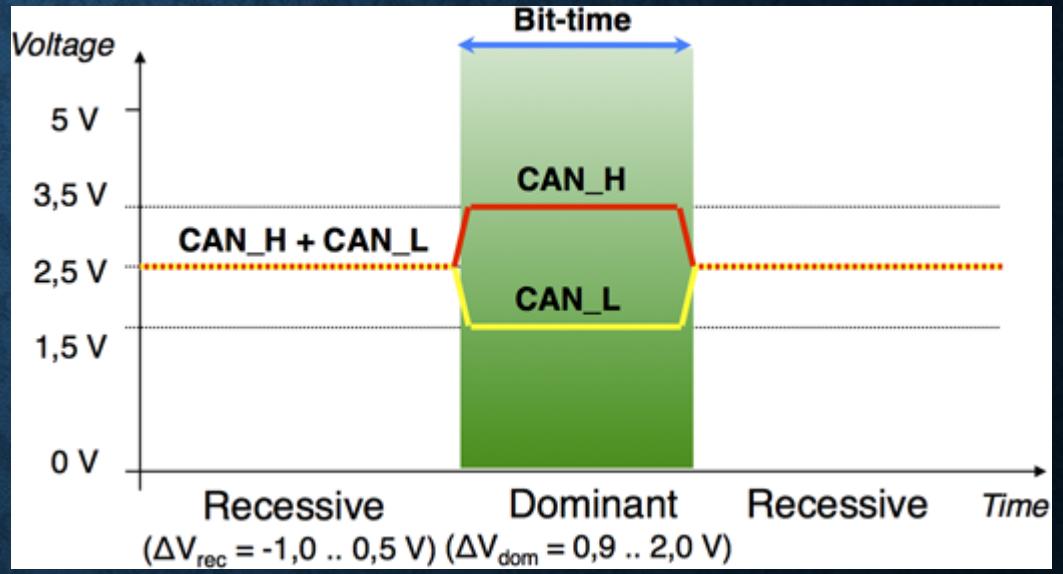
- A livello fisico il can bus utilizza sostanzialmente due fili Can-H e Can-L, il terzo filo, di massa o riferimento per gli altri due, potrebbe non essere necessario in quanto affidato ad una massa metallica comune ai dispositivi in comunicazione.



- Il filo tratteggiato è la massa, che potrebbe, non esistere
- E' necessaria una resistenza di terminazione da 120 Ohm tra i due fili Hi e Low

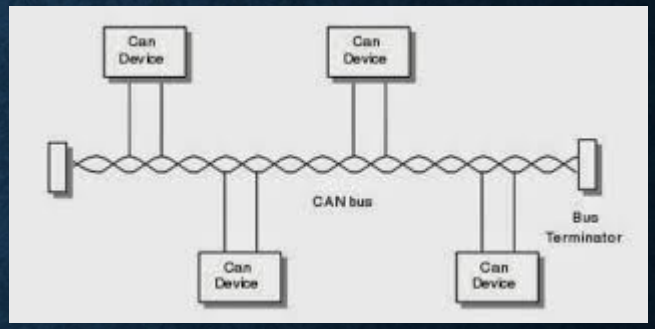
CAN BUS (2)

- L'immagine a lato ci fa comprendere come il CAN Bus codifica i bit da trasmettere, quando abbiamo il massimo divario si trasmette 0, nell'altro caso 1.
- Si intuisce che la velocità di campionamento dei segnali determina la velocità di trasmissione



CAN BUS(3)

- Fisicamente le line CAN vengono poste in opera ritorcendo i due conduttori principali
- Dato il modo di trasmettere (slide precedente) si capisce che una qualsiasi fonte di disturbi elettromagnetici è ininfluente perché provoca lo stesso disturbo ad entrambi i conduttori, per cui il sistema, che lavora sulle differenze di tensione tra i due segnali, è esente disturbi.
- La immunità ai disturbi elettromagnetici è uno dei plus del CAN che è stato studiato e progettato per l'uso automotive, ma poi si è esteso anche nel mondo industriale.



CAN BUS (4)

- Abbiamo già puntualizzato che per un corretto riconoscimento dei dati trasmessi occorre fissare un clock uguale per tutti i dispositivi; abbiamo un vasto range di velocità possibili, le più comuni sono 250 Kb, 500Kb e 1Mb, ma attenzione dobbiamo sempre ricordarci dello sviluppo in serie di Fourier, il dato elementare è sempre formato da onde quadre in banda base

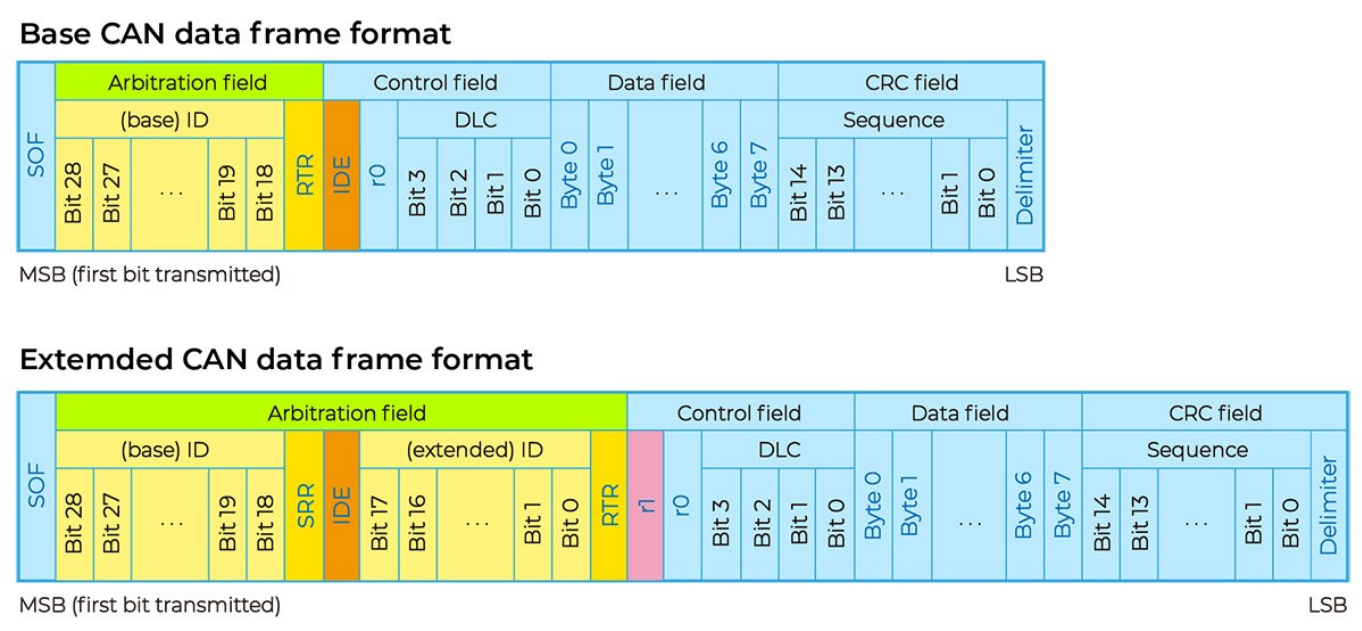
Bit Rate (kb/s)	Bus Length (m)
1000	30
500	100
250	250
125	500
62.5	1000

- Qualora fosse necessario collegare apparati diversi, non supportanti la velocità base di tutto il sistema, si possono prevedere appositi bridge.

CAN BUS(5)

- Vediamo ora di passare ai livelli superiori della trasmissione, dato che ora la rappresentazione dei bit e le velocità di trasmissione sono state fissate.
- Nel tempo si è arrivati alla definizione del Layer 2.0(A o B) che è ormai lo standard di base della trasmissione:
  - A -> 11 bit ID
  - B -> 29 bit ID
- La trasmissione avviene per frame e poiché a un solo nodo alla volta è consentito trasmettere, la comunicazione è del tipo CSMA/CD ovvero C(arrier)S(ense)M(ultiple)A(ccess)/C(ollision)D(etect)

CAN BUS (IL FRAME 2.0)



- Qui sopra le due possibilità di frame, con ID a 11 bit o 29 bit, non si entra nei dettagli, perché esula dall'obiettivo del seminario, tuttavia è bene notare che lo spazio dei dati vero e proprio è da 1 a 8 byte, e che anche l'ID serve a determinare cosa rappresentino i byte di dato.
- Notare che ogni frame ha in coda un CRC, per cui, oltre all'immunità elettromagnetica, si ha un controllo di integrità sul singolo frame
- Per tutti i riferimenti riguardo al CAN BUS il sito di riferimento è: https://www.can-cia.org/

IL PROTOCOLLO SAE J1939

- Citiamo questo protocollo della S(ociety of) A(utomotive) E(ngineers) International perché è «de facto» presente su tutto il parco veicoli a cui è dedicato e precisamente:
  - Truck (camion)
  - Bus
  - Agriculture (trattori)
- Il protocollo si basa sul Layer 2.0 B a 29 bit ID e l'ID ha la funzione di specificare il tipo di messaggio, infatti i messaggi sono di due tipi:
  - Da ECU a ECU
  - Broadcast

IL PROTOCOLLO SAE J1939(1)

- La struttura dei 29 bit dell'ID è qui a lato.
- Notiamo che ci sono tre bit per impostare una priorità
- Un bit R (generalmente non usato e viene calssificato reserved)
- Un data page (DP), che permette di raddoppiare i contenuti
- Un source address che identifica l'ECU inviante
- Un PGN diviso in due parti:
  - PDU Format
  - PDU Specific

IL PROTOCOLLO SAE J1939(2)

- PDU Format
  - Se compreso tra 0 e 239 (0xEF bit 7 off) il messaggio è di tipo ECU -> ECU e PDU Specific è l'address del destinatario
  - Se compreso tra 240 e 255 (bit 7 on) il messaggio è di tipo Broadcast e PDU specific indica un gruppo di dati
- Non staremo qui ad elencare tutti i messaggi possibili perché con 16 bit di identificativi la lista è assai lunga e non sempre i veicoli li implementano tutti,
- Ma approfittiamo per indicare che la diagnostica standardizzata, per il protocollo J1939, OBD, EOBD,EOBD2 utilizza due PGN di tipo ECU ->ECU per veicolare il protocollo diagnostico verso le ECU presenti; siamo in pratica nella situazione in cui un protocollo ne contiene un altro.

LA DIAGNOSTICA VEICOLARE (OBD)

- Come accennato nelle slide precedenti, all'interno dei network veicolari, ci sono messaggi riservati alla diagnostica, non potendo trattare esaustivamente l'argomento diamo un cenno di come i costruttori seguono le direttive ISO sull'argomento.
- Ogni network CAN veicolare è contraddistinto dalla velocità oltre che dal Layer 2.0 che può essere A o B.
- Per accedere alla diagnostica occorre, dunque, effettuare 4 tentativi di connessione dati derivanti dalla combinazione di due velocità ammissibili (nda) e di due lunghezze di ID. Ovviamente la prima combinazione che otterrà una risposta decodificabile, segnerà velocità e Len ID di tutta la successiva comunicazione.
- In commercio esistono degli OBD link che si inseriscono sulla presa per la diagnostica, si occupano di arbitrare la giusta combinazione di comunicazione e di codificare/decodificare il protocollo restituendo/inviando il messaggio completo (l'ID viene restituita come 32 bit).

IL BUS LIN

- LIN, che sta per Local Interconnect Network, è un sistema di comunicazione utilizzato in campo automotive in contemporanea al CANbus.
- Il LIN ha preso piede nel contesto automotive per diminuire il numero dei nodi CAN attivi, allo scopo di clusterizzare le informazioni.
- Per meglio comprendere il problema possiamo immaginare che in un gruppo di componenti il veicolo, ad esempio un gruppo ottico posteriore, sia opportuno effettuare della diagnostica ad es. lampade (led o tradizionali) non funzionanti, si può pensare di mettere dei sensori di corrente, uno per lampada, raggruppati in una scheda di monitoraggio e che questa sia poi connessa al CAN.
- Il sistema, anche se sembrerebbe più complesso, ha il vantaggio di avere un monitoraggio sulla parte con tempi e frequenze diverse dal resto del veicolo e di interessare il CAN solo se necessario, parallelamente il CAN può inviare comandi digitali al sottosistema che provvederà in autonomia (es. accendi le luci di stop).
- Il LIN è generalmente connesso al CAN per mezzo di un bridge.

IL BUS LIN(1)

- Il bus LIN usa un solo filo per trasmettere/ricevere e si configura come master/slave
- La velocità massima è circa 20Kbit/s (19.200) e si possono avere sino a 16 slave
- La tensione di trasmissione in gioco è di 12V, tipica dell'automotive
- A parte il master, che è sempre attivo, specie come bridge verso il CAN, gli slave possono essere dei lower power systems, cioè microcontrollori che hanno la capacità di porsi in uno stato di bassissimo consumo elettrico e di risvegliarsi a fronte di un evento (il LIN master inizia la comunicazione inviando un break sulla linea ed i microcontrollori hanno UART in grado di rivelare un break sulla linea).
- Per saperne di più, internet è un ottimo mezzo, ma approfondire esula dal presente seminario, menzioniamo il LIN per la sua diffusione come bus ausiliario del CAN.

IL BUS LIN(2)

- I sottocomponenti del veicolo in cui si usa il LIN sono, generalmente:
  - **Comfort:** Sensori di temperatura, insolazione tetto, luminosità, umidità
  - **Powertrain:** Sensori di posizione, velocità, pressione
  - **Aria condizionata:** Motori, pannello di comando
  - **Portiere:** specchietti esterni, finestrini, controllo sedili, chiusura porte
  - **Altro:** tergicristalli, sensori di pioggia, sensori di abbagliamento, …

IL PROTOCOLLO NMEA2000

- NMEA sta per National Marine Electronics Association
- E' un protocollo, basato su CAN, che è divenuto uno standard per le imbarcazioni da diporto ed è standardizzato come IEC 61162-3.
- Si basa sul Layer 2.0B (29 bit ID) e sul SAE J1939, a cui sono stati aggiunti PGN specifici per la nautica.
- Nel J1939 gli indirizzi dei device sono fissi (es. 01 ECU motore, 03 Cambio automatico, EE tachigrafo, …) mentre i device NMEA utilizzano un arbitraggio dell'indirizzo, per altro previsto anche dal J1939, ma mai utilizzato, rendendo i dispositivi plug-n-play
- Anche per questo protocollo ci sono messaggi from->to e messaggi broadcast

IL PROTOCOLLO NMEA2000(1)

- Dispositivi tipici della nautica possono essere:
  - Ricevitori GPS
  - Pilota automatico
  - Anemometro
  - Profondimetro e ecoscandaglio
  - Bussola e strumenti di navigazione (velocità relativa all'acqua, velocità assoluta GPS, …)
- Un altro pregio è la facilità di ripetizione dei dispositivi sia di comando, che di segnalazione. In molte imbarcazioni la plancia di comando può essere in cabina, sul ponte, sul flying bridge ed un cablaggio basato su CAN bus semplifica la connessione.

CONCLUSIONI

- Con questo terzo seminario abbiamo dato una panoramica sul mondo della comunicazione tra dispositivi, una panoramica tutt'altro che esaustiva, infatti esistono anche altri protocolli e media per comunicare.
- Abbiamo escluso i sistemi senza fili, che necessiterebbero una trattazione a parte data la varietà:
  - Wi-Fi
  - Bluetooth
  - Zig Bee
  - …
- Mentre ci siamo soffermati sui problemi della comunicazione wired, comunicazione molto più difficile da «disturbare» e quindi assai usata nelle comunicazioni che necessitano una «sicurezza» di scambio.
- Quello che non è stato trattato è di minore importanza in termini di diffusione, tuttavia può sempre essere trattato o compreso attraverso Internet.

GRAZIE PER L'ATTENZIONE

Francesco Maria Rietti