Corso di Sistemi Elettronici per Automazione e Robotica
LM Ingegneria dell'Automazione e Robotica
Prof. S. Saponara

Appunti sui protocolli di comunicazione nei sistemi di automazione e automotive (draft 1.0)

1. Introduzione: comunicazione in sistemi di automazione/automotive
Approccio tradizionale
Fieldbus Network
Protocolli di comunicazioni e terminologia

2. Interfaccia RS-232
Segnale RS-232
Problema della sincronizzazione
Connessione RS-232 tra PC e periferica
UART

3. CAN (Controller Area Network)
Introduzione al CAN
Caratteristiche del protocollo CAN
Formati dei messaggi
Rilevamento degli errori
Auto diagnosi dei nodi
Livello fisico del CAN
Implementazione dei nodi CAN (Basic CAN e Full CAN)
Uso di bus CAN nel settore automotive

Riferimenti

1. Introduzione: comunicazione in sistemi di automazione/automotive

Approccio tradizionale
Sistemi di controllo tradizionali in ambito industriale erano realizzati connettendo i dispositivi periferici che controllano le varie parti di un processo direttamente ad una unità centrale, dotata di elevata potenza di calcolo, tramite collegamenti punto-punto. Le stazioni periferiche fornivano informazioni all'unità centrale che eseguiva tutte le operazioni di controllo sull'intero sistema e poi ridistribuiva i compiti ad ognuno dei dispositivi periferici. Questo tipo di connessione comporta un elevato numero di cablaggi, spese di installazione e manutenzione, nonché una bassa flessibilità nell'upgrade del sistema. In particolare nel settore automotive l'esigenza di far comunicare i molti dispositivi elettronici presenti all'interno delle automobili (e.g. Antilock Braking System, Engine Control, Air Conditioning Control, chiusura centralizzata etc.) e la complessità di questi avrebbe portato ad un aumento insostenibile di collegamenti dedicati ed una duplicazione dei sensori necessari a più dispositivi, con conseguente aumento dei costi di produzione e soprattutto notevole ingombro fisico.

Fieldbus Network
L'evoluzione delle tecnologie microelettroniche e la conseguente disponibilità di microprocessori con prestazioni elevate e costi contenuti ha consentito di decentralizzare il controllo permettendo ai nodi periferici di poter eseguire per proprio conto le operazioni di cui necessitano. Si è passati così ad uno scenario in cui tutti i componenti del sistema (microprocessori, risorse di memorizzazione volatile e non, convertitori AD e DA, sensori e attuatori con interfacce digitali, etc.) sono connessi tra loro sfruttando un bus e diventando a tutti gli effetti nodi di una unica rete (Fieldbus Network).

A differenza di altri tipi di rete per telecomunicazioni e/o scambio dati tra computer l'obiettivo principale delle reti Fieldbus non è massimizzare la quantità di dati trasferiti/sec (throughput). L'obiettivo è lo scambio di messaggi di comando e stato ovvero messaggi di dimensioni modeste (qualche bytes) con protocolli semplici, cablaggi ridotti, minimizzazione degli errori, capacità di diagnostica, facilità di installazione, interoperabilità e intercambiabilità dei dispositivi connessi.

Protocolli di comunicazioni e terminologia
Tra le tecnologie, intese come protocolli di comunicazione, più diffuse impiegati in reti Fieldbus (http://www.fieldbus.com.au/techinfo.htm) troviamo la RS-232, la RS-485 e il CAN (Controller Area Network) che implementano protocolli di tipo seriale asincrono.
Seriale significa che i bit che costituiscono l'informazione sono trasmessi uno alla volta, in successione, su un solo "filo" di connessione. Questo termine è in genere contrapposto a "**parallelo**": in questo caso i dati sono trasmessi contemporaneamente su più fili (e.g. bus PCI nei computer è un bus parallelo a 32 bit). Da notare che una trasmissione seriale non è a priori più lenta di una parallela: se da un lato su di un

filo possono passare meno informazioni che su 32 questo viene bilanciato dalla difficoltà di controllare lo skew (disallineamento temporale tra i vari segnali) dei molti trasmettitori in un bus parallelo. Per esempio in una fibra ottica o in un cavo FireWire (standard seriali) le informazioni transitano ad una velocità paragonabile a quella di un bus PCI parallelo. In ogni caso la scelta nei Fieldbus di adottare standard seriali è legata alla necessità di semplificare i cablaggi, minimizzare errori di trasmissione/ricezione dei messaggi, alla richiesta di bit-rate non elevati (max. centinaia di Kbits/s).
Il termine **asincrono** indica che i dati sono trasmessi senza l'aggiunta di un segnale di clock, cioè di un segnale comune per sincronizzare la trasmissione con la ricezione; sia il trasmettitore che il ricevitore sono comunque dotati di un clock locale per poter interpretare i dati. La sincronizzazione dei due clock è necessaria ed è fatta in corrispondenza della prima transizione sulla linea dei dati.
Se la trasmissione dati è bidirezionale ma non avviene contemporaneamente nelle due direzioni si parla di comunicazione **Half-duplex**: un dispositivo (ricevitore, Rx) ascolta e l'altro (trasmettitore, Tx) emette segnali. Quando è necessario si scambiano i ruoli. **Full-duplex** indica che la trasmissione è bidirezionale e contemporanea. In questo caso sono necessari due fili oppure qualche altro sistema (divisione di frequenza, divisione di codice) per distinguere i due messaggi contemporanei nelle due direzioni. Se la trasmissione è sempre in un solo verso, si parla di **Simplex**. Tipicamente i vari nodi che costituiscono la rete del sistema di controllo fungono sia da trasmettitori che ricevitori (protocollo duplex).
Se nel bus uno solo dei nodi svolge la funzione di gestore della rete (controlla e gestisce l'accesso al bus) si parla di protocollo **master/slave** in cui il predetto nodo funge da master mentre gli altri nodi sono detti slave e sono semplici utilizzatori del canale di comunicazione (possono ricevere o trasmettere informazione ma in funzione del controllo del bus fatto dal master). Una rete master/slave in cui ci sono più dispositivi che fungono da master si dice **multi-master**. In tal caso è necessario un **sistema di arbitraggio** per i risolvere i conflitti che nascono quando più master richiedono il controllo del canale di comunicazione. Tipicamente in queste reti il nodo (master) che prende il controllo del bus, quando inizia una comunicazione, specifica tramite un indirizzo a quale nodo della rete è destinata l'informazione (e.g. bus dati e indirizzi nell'architettura interna di un calcolatore descritta nelle slides di Lezione 16). Esistono anche protocolli detti **producer/consumer** in cui qualsiasi nodo può acquisire momentaneamente il controllo del bus ed iniziare una trasmissione (funge da producer) mentre gli altri nodi si attiveranno in questa fase in ricezione (consumer). Il nodo della rete che trasmette non specifica a quale nodo della rete è destinata l'informazione (ovvero non viene specificato nessun indirizzo) ma specifica da quale interfaccia vengono prodotti i dati. Il CAN è un protocollo di comunicazione di questo tipo.

2. Interfaccia RS-232

Segnale RS-232
Figura 1 mostra l'andamento del segnale (trama) che, in una comunicazione tra due dispositivi con protocollo RS-232 a 9600 bits/s 8n2 rappresentante il valore binario 00110000.

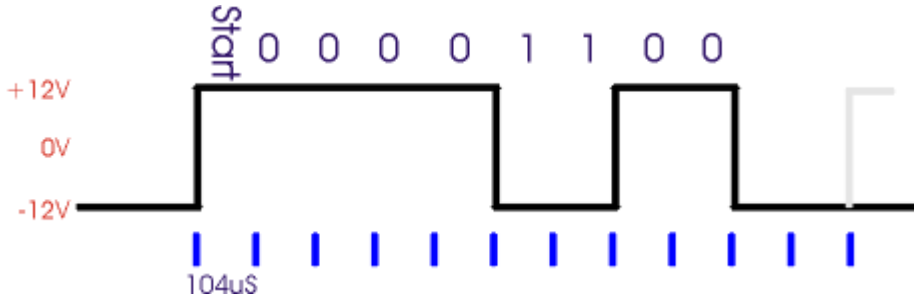

Figura 1: Esempio segnale RS-232

Il segnale, così come nel CAN, utilizza per i bit una codifica **NRZ** (Non-Return-Zero): il segnale è un treno di impulsi rettangolari di durata T fissata dal bit-rate (T=1/9600=104 µs nell'esempio) e con ampiezza caratterizzata da un valore "alto" pari a circa +12V ed un valore "basso" pari a –12V. Ogni impulso rappresenta un bit. In particolare nella RS-232 un valore alto di tensione rappresenta lo zero logico ed uno basso un uno logico. La linea si trova inizialmente nello stato di riposo con un livello basso di tensione (nessun dato in transito); la prima transizione da livello basso di tensione a livello alto indica l'inizio della trasmissione ("bit di start" della durata di 104 µs). Segue il bit meno significativo (LSB), dopo altri 104 µs il secondo bit, e così via, per otto volte, fino al bit più significativo (MSB). Segue infine un periodo di riposo della linea di almeno 208 µs, cioè due bit di stop e quindi (eventualmente) inizia un nuovo pacchetto di bit. Pertanto nell'esempio considerato vengono trasmessi 11 bit di cui solo 8 rappresentano l'informazione effettivamente utile. Da notare che con la codifica NRZ il numero di commutazioni è inferiore al numero di bit (in Figura 1 abbiamo 4 commutazioni a fronte di 11 bit trasmessi)

Esistono delle varianti: Se la trasmissione è più veloce o più lenta, la durata degli impulsi varia (a 1200 Kbits/s le transizioni avvengono a multipli di 0,833 ms). Lo standard originale prevede una velocità fino a 20 Kbits/s. Uno standard successivo (RS-562) ha portato il limite a 64 Kbits/s con i due standard compatibili a bassa velocità. Le interfacce seriali RS-232 nei normali PC in genere superano i 100 Kbits/s. Il numero di bit dei dati trasmessi per ogni trama può variare da 5 a 9; e' possibile aggiungere un bit di parità per incrementare la robustezza della comunicazione; al termine della comunicazione la linea rimane nello stato di riposo per almeno 1 o 2 bit. Il formato del pacchetto trasmesso è indicato da una sigla composta da numeri e cifre, per esempio 8n2 dove la prima cifra indica quanti bit di dati sono trasmessi (8), la prima lettera il tipo di parità (nessuna), la seconda cifra il numero di bit di stop (2).

Come detto oltre ai bit dei dati viene inserito un bit di parità (opzionale) per verificare la correttezza del dato ricevuto. Esistono cinque tipi di parità:

- None: nessun tipo di parità, cioè nessun bit aggiunto
- Pari (even): il numero di 1 (incluso il bit di parità) è sempre pari
- Dispari (odd): il numero di 0 (incluso il bit di parità) è sempre dispari
- Mark: il bit di parità vale sempre 1
- Space: il bit di parità vale sempre 0

La tensione di uscita di un trasmettitore RS-232 deve essere compresa in valore assoluto tra 5 e 25 V (valore ridotto a 13 V in alcune revisioni dello standard). A volte le tensioni in uscita sono diminuite a +/- 6V anziché i 12 V dell'esempio per permettere minori emissioni elettromagnetiche. Il ricevitore deve funzionare correttamente con tensioni di ingresso comprese, in valore assoluto, tra 3 V e 25 V. Molti ricevitori commerciali considerano semplicemente una tensione di soglia al valore di +2V (sopra viene riconosciuto un segnale alto, sotto uno basso). Per adattare i segnali utilizzati da circuiti digitali con livelli non direttamente compatibili con la standard RS-232 esistono appositi traslatori di livello che hanno il compito di fornire sia in trasmissione che in ricezione gli opportuni livelli pur non modificando la forma del segnale trasmesso.

Problema della sincronizzazione
Data la possibilità di implementare diverse varianti dello stesso protocollo di comunicazione sia trasmettitore che ricevitore devono accordarsi sul modo di trasmettere i dati prima di iniziare la trasmissione vera e propria. Inoltre e' importante garantire il rigoroso rispetto della durata dei singoli bit: infatti non è presente alcun segnale di clock comune tra trasmettitore e ricevitore e l'unico elemento di sincronizzazione è dato dal fronte di salita del bit di start. Poiché il campionamento in ricezione è effettuato di norma al centro di ciascun bit l'errore massimo ammesso è, teoricamente, pari alla durata di mezzo bit. Naturalmente questo limite non tiene conto della possibile difficoltà di riconoscere con precisione il fronte del bit di start (soprattutto su grandi distanze ed in ambiente rumoroso) e della presenza di interferenze intersimboliche tra bit adiacenti.

Connessione RS-232 tra PC e periferica
Sul PC sono disponibili due tipi di connettori RS-232 per connessione con periferiche (strumentazione, attuatori o sensori con interfaccia digitale, altri PC) identici dal punto di vista funzionale: uno a 9 pin (DB9) e uno a 25 pin (DB25 ormai scomparso dalle generazioni di PC odierne). Per ricevere e trasmettere un segnale RS-232 bastano in teoria tre fili: ricezione, trasmissione e massa. Gli altri fili sono utilizzati per l'handshake tra trasmettitore e ricevitore ovvero per sincronizzare la comunicazione. In particolare sono presenti due coppie di fili:

DTR/DSR: Quando il PC è collegato per la prima volta, pone alto DTR. La periferica risponde ponendo alto DSR

RTS/CTS: quando il PC inizia la trasmissione pone RTS alto, la periferica segnala che è pronta a iniziare la comunicazione ponendo CTS alto. Per interrompere la trasmissione la periferica pone CTS basso.

UART

Da notare che i processori lavorano con bus paralleli: da 8-bit tipici di microcontrollori per controllo industriale, fino ai 128 bit di processori Very Long Instruction Word dedicati per elaborazioni di segnali multimediali, con 32-bit valore tipico per processori general purpose. Per trasformare il segnale parallelo proveniente dal processore in segnale seriale esistono dei chip detti UART (Universal Asyncrhonous Receiver/Transmitter). In genere vengono gestite dall'hardware tutte le funzioni a basso livello necessarie quali inserimento dei bit di start e di stop, generazione o riconoscimento del bit di parità, generazione di interrupt e spesso è presente un buffer con logica FIFO (First In First Out) che permette di ricevere ed inviare dati anche quando il processore è impegnato. Nei moderni microcontrollori il modulo che fa da UART è spesso integrato sullo stesso chip del core di processamento e pertanto il microcontrollore può comunicare secondo protocolli sia seriali che paralleli.

3. CAN (Controller Area Network)

Introduzione al CAN

CAN (Controller Area Network) è un bus seriale di comunicazione dati progettato per applicazioni real-time: consente a controllori, sensori e attuatori di comunicare l'uno con l'altro ad una velocità fino a 1Mbit/s, tipicamente 500 Kbits/s nelle attuali implementazioni (data-rate molto maggiore delle velocità descritte per la RS-232), offrendo anche:

– bassi costi di progettazione e implementazione
– funzionamento in ambienti ostili
– facilità di configurazione e modifica
– rilevamento automatico degli errori di trasmissione.

Nato originariamente per l'industria automobilistica (CAN è stato sviluppato dalla Bosch nel 1986 su richiesta della Mercedes), si è diffuso presto nell'automazione industriale per le sue caratteristiche di robustezza ed affidabilità. Oggi sono disponibili chip di diverse aziende che implementano il protocollo CAN (Philips, Intel, Motorola, Siemens etc.) e sono disponibili diversi integrati che implementano su un unico chip un controllore CAN insieme al core di processamento e una UART.

Caratteristiche del protocollo CAN

Il nucleo del protocollo CAN è nel livello data link ovvero nella politica di accesso al mezzo di trasmissione. L'architettura di rete è molto flessibile in quanto possono essere supportate connessioni punto-punto, master/slave o multi-master. Tra le caratteristiche del protocollo abbiamo:

Assenza di indirizzi mittente/destinatario e multicast

I pacchetti trasmessi da un modulo CAN non contengono indirizzi di alcun genere, al loro posto troviamo un identificatore del contenuto del messaggio (e.g. giri al minuto, temperatura motore, temperatura abitacolo ...) unico sull'intera rete. Pertanto, in accordo ad un approccio producer/consumer, il nodo della rete che trasmette non specifica a quale nodo della rete è destinata l'informazione ma specifica da quale interfaccia vengono prodotti i dati. Un nodo ricevitore può verificare il contenuto del messaggio e filtrare i soli pacchetti a cui è interessato (message filtering), ignorando gli altri. Questo modo di operare in cui l'informazione del nodo trasmettitore è accessibile a tutti gli altri nodi della rete è detto Multicast.
Ad esempio in Figura 2 viene rappresentata la situazione di un nodo (il nodo 2) che trasmette dati riguardanti e.g. i giri al minuto del motore e mentre il tachimetro (nodo 1) e l'Engine Management System (nodo 4) accettano il pacchetto, l'unità di gestione dell'aria condizionata (nodo 3) lo ignora.
Questo approccio di comunicazione (basato sul contenuto dei messaggi trasmessi sulla rete e non sugli indirizzi dei nodi della rete) permette un alto grado di flessibilità e modularità del sistema, consentendo che nuovi nodi che sono solo ricevitori e che hanno bisogno dei soli dati esistenti possano essere aggiunti senza alcuna modifica nè all'hardware nè al software.
Nel suo formato standard il CAN supporta identificatori del messaggio a 11 bits (che consentono la codifica di 2048 tipi di messaggi differenti).

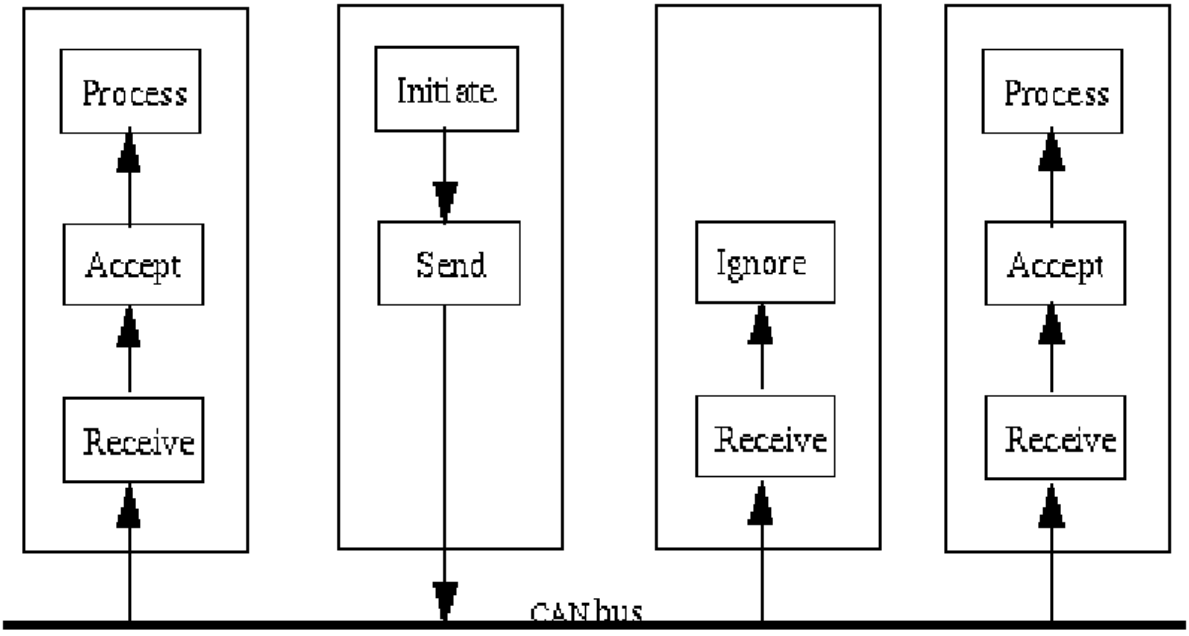

Figura 2: Esempio di message filtering, l'informazione del nodo 2 è ricevuta dai nodi 1, 3 e 4 ma viene ignorata dal nodo 3 e utilizzata solo dai nodi 1 e 4.

Politica di arbitraggio
Ovviamente in una rete in cui vi sono più nodi in grado di generare informazione nasce il problema dell'arbitraggio dell'accesso al bus. Quando il canale (bus) è libero ogni unità connessa può cominciare a trasmettere secondo una politica di trasmissione detta *non-destructive bitwise arbitration*. Ovvero due o più stazioni che iniziano a trasmettere competono per l'accesso al bus con un valore di priorità

determinato proprio dall'identificatore. Quello con il valore numerico più basso vince la competizione per il canale. In pratica l'ID del messaggio definisce una priorità in trasmissione. Il nodo che deve trasmettere il messaggio con priorità maggiore (ID minore: lo 0 vince sull' 1. Al livello logico 0 corrisponde un livello di segnale detto DOMINANT, al livello logico 1 corrisponde un livello di segnale detto RECESSIVE. Una eventuale sovrapposizione di segnali DOMINANT e RECESSIVE sul bus viene interpretato come uno 0 logico) può trasmettere non appena è libero il bus.
In Figura 3 è riportato un esempio di conflitto sull'accesso al bus tra tre nodi che tentano di trasmettere allo stesso istante: in particolare vengono riportati i segnali che tentano di trasmettere ed il segnale effettivo che si legge sul bus. In particolare per i primi tre cicli non c'e' conflitto. Al quarto ciclo il nodo 1 perde la competizione e smette di trasmettere perché cerca di imporre un uno logico (recessivo) mentre i nodi 2 e 3 impongono lo zero logico (dominante). In corrispondenza dell'ottavo ciclo il nodo 3 perde la competizione e smette di trasmettere mentre il nodo 2 continua a trasmettere come se fosse stato solo sul bus. Da notare che i nodi perdenti diventano subito ricevitori e non tenteranno una ritrasmissione non prima che il bus sia diventato libero. Questa politica garantisce il determinismo dell'accesso al bus, e l'assenza di periodi di inattività del canale. Anche in questo caso la politica di arbitraggio è basata sul contenuto dei messaggi trasmessi sulla rete e non sugli indirizzi dei nodi della rete.

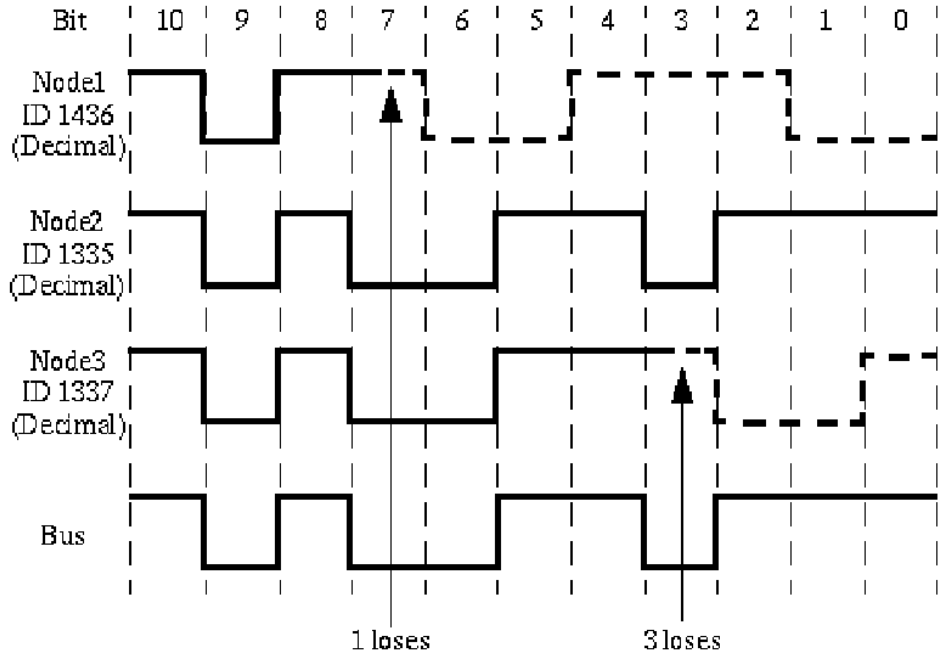

Figura 3: Esempio di arbitraggio nell'accesso al bus CAN

Formati dei messaggi

CAN prevede 4 tipi di messaggi:
1. Data Frame (trasporta dati da un nodo sorgente ad un nodo ricevente);
2. Remote Frame (viene inviato da un nodo per ottenere un DATA FRAME con lo stesso ID);
3. Error Frame (trasmesso da qualsiasi stazione che rilevi un errore sul bus);
4. Overload Frame (serve ad aggiungere ritardi extra)

Ci sono due formati per i Data Frame e Remote Frame:
- Standard CAN (Versione 2.0 A)
- Extended CAN (Versione 2.0 B)

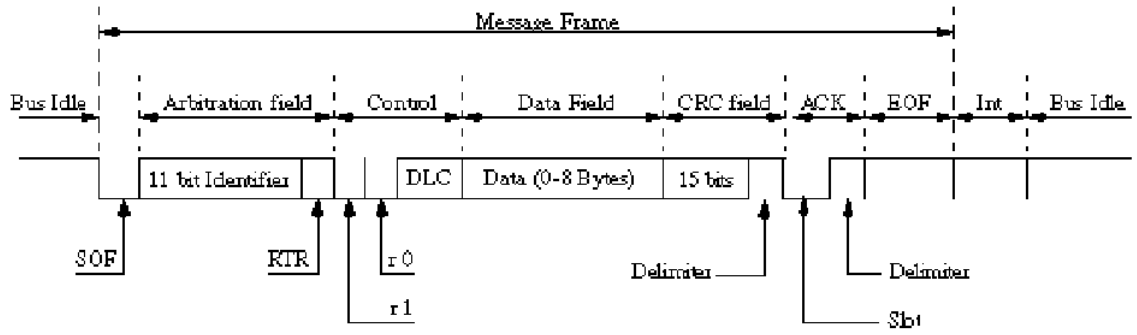

Figura 4: Struttura messaggio nel protocollo CAN 2.0

Con riferimento a Figura 4 lo Standard CAN 2.0 A prevede che il Data Frame sia strutturato come segue:
- Un campo Start Of Frame (SOF). È un bit dominante (0 logico) che indica l'inizio di un message frame. Il rilevamento di un bit dominante quando la linea è in pausa (Bus Idle, sulla linea c'e' un bit 1 recessivo) è quindi interpretato come un SOF. Difatti l'SOF in Figura 3 ha la stessa funzione dello start bit della RS-232 in Figura 1.
- Un Arbitration Field, contenente 11 bit di identificatore e il Remote Transmission Request (RTR) bit. Quest'ultimo bit quando è settato a 0 indica che il frame è un Data Frame, se settato a 1 indica che è un Remote Frame. Il Remote Frame è una richiesta da parte di un nodo del Data Frame corrispondente (avente lo stesso identificatore) e non presenta un campo dati. Ovvero un nodo della rete può richiedere dei dati di un tipo specificato (da un certo ID) inviando un Remote Frame (che essendo una richiesta non porta informazione e quindi ha il campo dati Data Field nullo). Il nodo che può fornirli li invia in un Data Frame che ha lo stesso ID del Remote Frame.
- Un Control Field contenente 6 bit: 2 bit riservati per usi futuri e 4 bit di Data Lenght Code (DLC). Questo indica il numero dei byte nel Data Field seguente, esso può variare da 0 a 8 byte (2^4=16, ma si utilizzano le sole combinazioni "0000" → 1000").
- Data Field rappresenta il contenuto informativo vero e proprio.
- Il CRC Field contenente 15 bit di cyclic redundancy check code e un bit recessive come delimitatore. Il CRC difatti rappresenta una evoluzione del concetto di bit di parità ovvero dell'inserzione di bit ridondanti atti ad aumentare la capacità di individuazione di errori nella comunicazione.
- Il campo Ack di 2 bit; il primo è lo Slot Ack che è trasmesso come recessive, ma è sovrascritto con un bit dominante da ogni stazione che riceve correttamente il messaggio; il secondo bit è recessive e svolge il compito di delimitatore.
- Il campo End Of Frame (EOF) che consiste di 7 recessive bit.

Per separare tra loro frames in sequenza vi è l'Interframe Space che non è un periodo di tempo tra due frame, bensì un insieme di bit ovvero un pacchetto speciale inviato

dall'ultimo nodo che ha trasmesso (Overload Frames ed Error Frames non sono preceduti dall'Interframe Space). Il pacchetto ha due (tre) campi:
- Intermission (Int che consta di 3 bits settati a 1; quando sul bus passano i bits di Intermission nessuna stazione può trasmettere Data o Remote Frame: l'unica azione ammessa è la segnalazione di condizioni di overload);
- Bus Idle (di lunghezza arbitraria);
- Suspend Transmission (solo per stazioni in condizioni Error Passive; tale campo, se presente, è posto tra gli altri due. Una stazione Error Passive, dopo l'invio di un messaggio, trasmette l'Intermission seguito da 8 bits a 1 prima di trasmettere un nuovo frame o di liberare il bus. Se durante tale periodo un'altra stazione comincia a trasmettere, questa si arresta).
Da notare che la struttura di un Remote Frame e' simile a quella del Data Frame ma Data Field non c'è e RTR bit vale 1 invece di 0.
Lo standard CAN 2.0 B prevede un identificatore di 29 bit, per offrire compatibilità con altri protocolli di comunicazione seriale usati in USA. In particolare l'identificatore è diviso in Base ID lungo 11 bit per garantire la compatibilità con la versione A e in Extension ID di 18 bit. Esistono tre tipi di controllori CAN: i 2.0A che sono capaci di spedire solo messaggi di formato Standard, restituendo errore nel caso ricevano in formato Extended; i 2.0B passive che sono in grado di spedire solo in formato Standard, ma possono ricevere in formato Extended; i 2.0B che possono funzionare in entrambe le modalità.

Rilevamento degli errori
Il processo di segnalazione degli errori in una rete CAN si articola nelle seguenti fasi:
- Un controller CAN rileva un errore (in trasmissione o in ricezione)
- Un Error Frame viene immediatamente trasmesso
- Il messaggio incriminato viene ignorato da tutti i nodi
- Viene aggiornato lo stato del controller CAN
- Il messaggio viene ritrasmesso, eventualmente competendo con altri.

Un errore può essere rilevato in 5 modi, 3 dei quali a livello del messaggio e 2 a livello del singolo bit:
- Bit Stuffing Error; normalmente un nodo in trasmissione inserisce dopo 5 bit consecutivi della stesa polarità un bit di polarità opposta; ciò è chiamato bit stuffing. Un nodo che riceve più di 5 bit consecutivi di segno uguale rileverà un errore di questo tipo.
- Bit Error; un nodo in trasmissione ascolta sempre il bus per verificare la corrispondenza con ciò che sta trasmettendo: se esso ascolta un bit diverso dal suo verrà segnalato un errore.
- Checksum Error; ogni nodo ricevente ricalcola il CRC in base a ciò che ha ricevuto, e se non corrisponde a quello inviato dal mittente viene segnalato un errore.
- Frame Error; viene segnalato questo tipo di errore quando vengono violati alcuni campi fissi del pacchetto (bit che devono essere spediti sempre dello stesso tipo).
- Acknowledgement Error; se il trasmettitore non rileva alcun riscontro al frame appena inviato.

Un Error Frame è costituito da un Error Flag ed un Error Delimiter.
L'Error Flag è lungo 6 bit dello stesso segno e viola volontariamente la regola dello bit stuffing in modo che tutte le altre stazioni rilevino un errore e spediscano anch'esse un Error Flag. Per questo motivo il campo Error Flag nel pacchetto è di lunghezza variabile (max 12 bit) dato dalla sovrapposizione di tutti gli Error Flag spediti. A seguito c'è un Error Delimiter costituito da 8 bit recessive.

Simile all'Error Frame è l'Overload Frame poiché anch'esso consiste di un Overload Flag e di un Overload Delimiter. Esso viene generato quando un nodo ricevitore ha bisogno di più tempo per processare i dati correnti prima che altri vengano ricevuti. Un nodo trasmetterà l'overload flag subito dopo l'EOF (End of Frame in Figura 4): in questo modo tutti gli altri nodi diagnosticheranno una condizione di overload e spediranno anch'esse un overload flag. A seguito della sovrapposizione dei flag ci saranno 8 bit recessive di delimiter. Un Overload Frame non richiede la ritrasmissione del frame che ha causato la condizione di overload.

Auto diagnosi dei nodi
CAN offre anche un meccanismo di auto isolamento dei guasti con la possibilità di distinguere tra condizioni di guasto transitorie (e.g. dovute a sbalzi di voltaggio, condizioni esterne di disturbo), e guasti permanenti (e.g. dovuti a cattive connessioni, cavi rotti). Ogni nodo CAN ha due registri di error count: uno di trasmissione e uno di ricezione. Essi sono inizialmente settati a 0 e vengono incrementati ogni qualvolta si presenta una situazione di errore (+1 per un errore in ricezione, +8 per un errore in trasmissione). A sua volta ogni nodo della rete CAN può trovarsi in tre stati (vedi Figura 5):
- Error Active. Nessuno dei due contatori ha superato il valore di 127. Il nodo è nel pieno delle sue funzionalità e decrementa di 1 i contatori ogni volta che riceve un messaggio andato a buon fine. Quando è in questo stato il nodo che rileva un errore spedisce un Error Flag costituito da 6 bit dominanti in modo da interrompere sempre la trasmissione.
- Error Passive. Almeno uno dei due contatori ha superato 127. Il nodo è ancora in grado di eseguire tutte le sue funzioni, ma è probabile che esso presenti dei disturbi o condizioni di guasto. Per questo quando esso rileva un errore spedisce un Error Flag di 6 bit recessive che vengono interpretati come errore solo se nessuna stazione sta spedendo un suo proprio messaggio (i bit recessive contrariamente vengono sovrascritti).
- Bus Off. Se uno dei contatori supera 255 il nodo si stacca dal bus e non partecipa alla comunicazione lasciando gli altri nodi nella possibilità di continuare a scambiarsi informazioni (autoisolamento). Se ciò accade certamente la stazione presenta un problema permanente che necessita di un intervento esterno per ripristinare il perfetto funzionamento. Alcune implementazioni consentono al nodo di tornare Error Active dopo che esso abbia ricevuto 128 messaggi andati a buon fine, altre necessitano di un reset hardware.

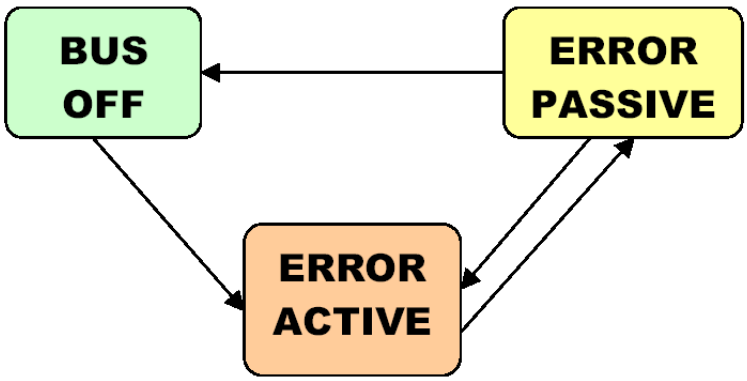

Figura 5: Stati di un nodo CAN

L'alta affidabilità del CAN e il suo successo in applicazioni in cui la sicurezza è un fattore critico (automazione, trasporti, biomedicale) è legata alla capacità di identificare dati corrotti da guasti di trasmissione. La probabilità residua d'errore è una misura statistica e specifica la probabilità che un messaggio sia corrotto ma non diagnosticato tale da nessun nodo della rete. E' stato calcolato che su un bus CAN a 1Mbit/s utilizzato al 50%, con lunghezze medie di messaggi di 80 bits si avrebbe una probabilità residua di non individuare un messaggio corrotto di circa 410^{-7} messaggi/ora (a 8 ore al giorno per 365 giorni l'anno si avrebbe un guasto non diagnosticato ogni 1000 anni).

Livello fisico del CAN

Il livello fisico del CAN è stato standardizzato in accordo con la direttiva ISO 11898. Il cavo trasmissivo su cui vengono trasmesse le informazioni in accordo al protocollo seriale fino ad ora analizzato è costituito da una coppia di fili intrecciati (chiamati CAN_H e CAN_L) pilotati in modo differenziale. Se l'informazione è trasmessa in modo differenziale ovvero $V= V_{CAN\_H} -V_{CAN\_L}$ tutti i disturbi ΔV che si manifestano sulle due linee nello stesso modo scompaiono essendo $V= (V_{CAN\_H} + \Delta V) – (V_{CAN\_L} + \Delta V)= V_{CAN\_H}-V_{CAN\_L}$. Il cavo può essere sia schermato sia non schermato. L'impedenza che termina il cavo deve essere di 120 Ω. La lunghezza massima del bus dipende dalla velocità di trasmissione usata in accordo ai valori in Figura 6 (bit-rate maggiori sono supportati su connessioni di lunghezza inferiore). In ogni caso il CAN è un bus per connessioni con data-rate massimi di 1 Mbits/s.

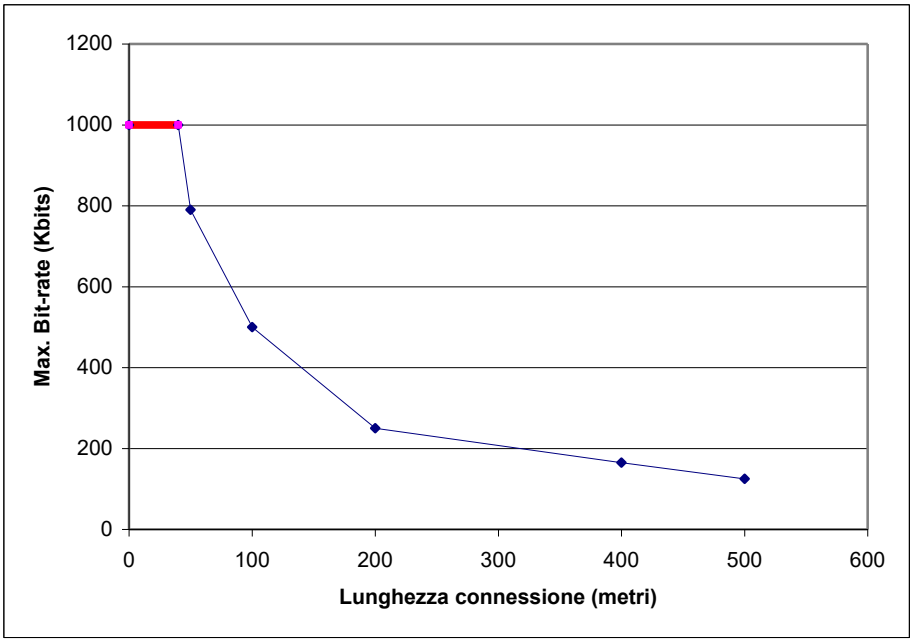

Figura 6: Velocità di trasmissione in funzione della lunghezza del bus

Per ridurre la potenza dissipata esiste per i nodi di una rete CAN la possibilità di entrare in una fase di sleep mode (nessuna attività interna, disconnessione dai bus drivers). Il nodo esce dallo sleep mode tramite un segnale di wake-up causato da qualche attività sul bus o da condizioni interne del sistema. Per svegliare nodi in sleep mode esiste un messaggio dedicato di wake up: "11111101111" che deve essere inviato con l'ID più piccolo possibile (infatti più piccolo l'ID del messaggio , maggiore è la sua priorità nel caso di conflitti sul bus).

Implementazione di nodi CAN

Il mercato offre un ampia gamma di dispositivi CAN, più di 50 prodotti diversi (da più di 15 aziende diverse). Nel 2000 sono stati venduti oltre 100 milioni di controllori CAN. In molti casi il controllore CAN è integrato sullo stesso chip del microprocessore.

Ci sono due principali strategie di implementazione per i controller CAN: le sostanziali differenze tra le due sono su come vengono filtrati i messaggi, su come vengono bufferizzati e su come avvengono le risposte ai Remote Frames.

Basic CAN

Il Basic CAN è un controller più economico. Esso possiede dei buffer di ricezione e trasmissione gestiti con politica First-In-First-Out: un messaggio può essere ricevuto su un buffer mentre il microcontrollore sta leggendo su un altro buffer; se arriva un messaggio quando tutti i buffer sono pieni, il più vecchio viene conservato, il che significa che possono andare perse delle informazioni nell'eventualità che il microcontrollore non sia abbastanza veloce. Un messaggio viene inviato scrivendolo in un buffer di trasmissione.

I messaggi a cui il nodo è interessato sono filtrati usando due registri che operano sull'identificatore del messaggio: se l'identificatore supera il controllo di tali

maschere viene registrato in un buffer. Il filtraggio finale viene fatto a livello software con un extra carico per il microcontrollore.
Il Basic CAN non supporta la risposta automatica ai Remote Frames, ma l'applicazione software dovrà gestirli, garantendo la correttezza della risposta (si ha così un carico extra di lavoro per il processore).

Full CAN
Il Full CAN è un controller più performante e più costoso del Basic. Ha un insieme di buffer chiamati mailboxes i quali al momento dell'inizializzazione del sistema sono settati in trasmissione o in ricezione e ad ognuno viene assegnato un identificatore; questo significa che ad ogni messaggio compete il proprio buffer specifico.
Quando viene ricevuto un messaggio vengono controllati tutti i buffer di ricezione cercando quello avente l'identificatore in questione: se viene trovato significa che il messaggio ha un contenuto rilevante per il nodo e viene memorizzato, altrimenti viene scartato in quanto non interessante. In questo modo il filtraggio avviene a livello hardware senza che se ne occupi il processore.
Lo stesso in trasmissione, il messaggio viene memorizzato nel buffer che gli compete; in questo modo può essere attuata anche una politica di selezione del messaggio da trasmettere, favorendo il più prioritario e non una semplica FIFO come nel Basic.
Quando viene ricevuto un Remote Frame, il controller CAN verifica se esiste un buffer di trasmissione con lo stesso identificatore: in caso affermativo, il Data Frame corrispondente viene subito inviato senza chiamare in causa il microcontrollore e snellendo di conseguenza i suoi compiti.

Uso del CAN nel settore automotive
Il CAN è largamente usato nel settore automotive per i suoi particolari vantaggi:
- bassi costi di progettazione e implementazione
- operatività in condizioni critiche
- facilità di configurazione e modifica
- rilevamento automatico degli errori
- diagnosi centralizzata dei guasti
- ottima affidabilità e sicurezza.

La maggior parte delle case automobilistiche fa uso di reti CAN. A cominciare dalla Daimler-Mercedes, per continuare con Audi, BMW, Renault, Opel, Saab, Volkswagen, Volvo, etc. Figura 6 riporta un esempio di rete CAN in ambito automotive. All'interno dell'auto vi sono diversi tipi di rete: una rete CAN high-speed per la gestione dei sotto-sistemi con stringenti richieste di real-time (gestione del motore, del sistema di trasmissione, sistema di frenata etc.) e una rete CAN low-speed per i sotto-sistemi con specifiche non stringenti di real-time quali chiusura centralizzata, controllo dei sedili, dei finestrini, aria condizionata, luci, etc. Un terzo tipo di rete connette tutti gli apparati di intrattenimento presenti all'interno dell'abitacolo come la radio, il compact disc, il cellulare, il sistema di navigazione satellitare (in Figura 6 questa rete è connessa alla rete CAN tramite il multimedia gateway). In accordo ad una strategia gerarchica di connessione, ogni nodo (magari costituito da più dispositivi) interfacciato alla rete CAN può a sua volta presentare un

bus locale (Local Interconnect Network, LIN sub-bus in Figura 6). In tecnologie attuali la divisione tra linee CAN high-speed e low-speed è per velocità dei 125 Kbits/s. In ogni caso la high-speed arriva al massimo ai 500 Kbits/s. Figura 7 riporta un esempio di rete di comunicazione completa in una automobile (in-vehicle network composta da CAN high-speed e low-speed, LIN locali e rete per dispositivi multimediali) .

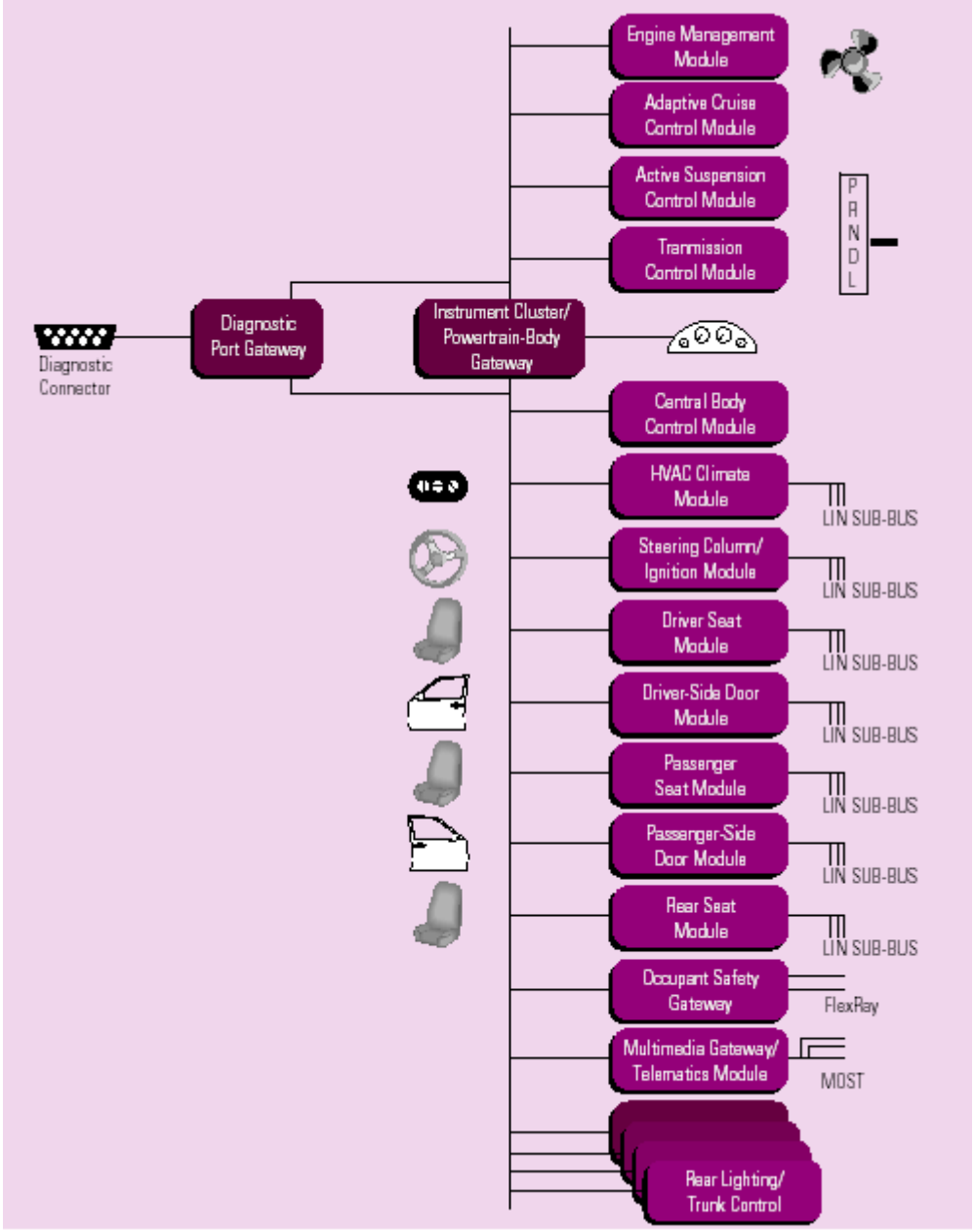

Figura 6: Esempio di rete CAN nel settore automotive

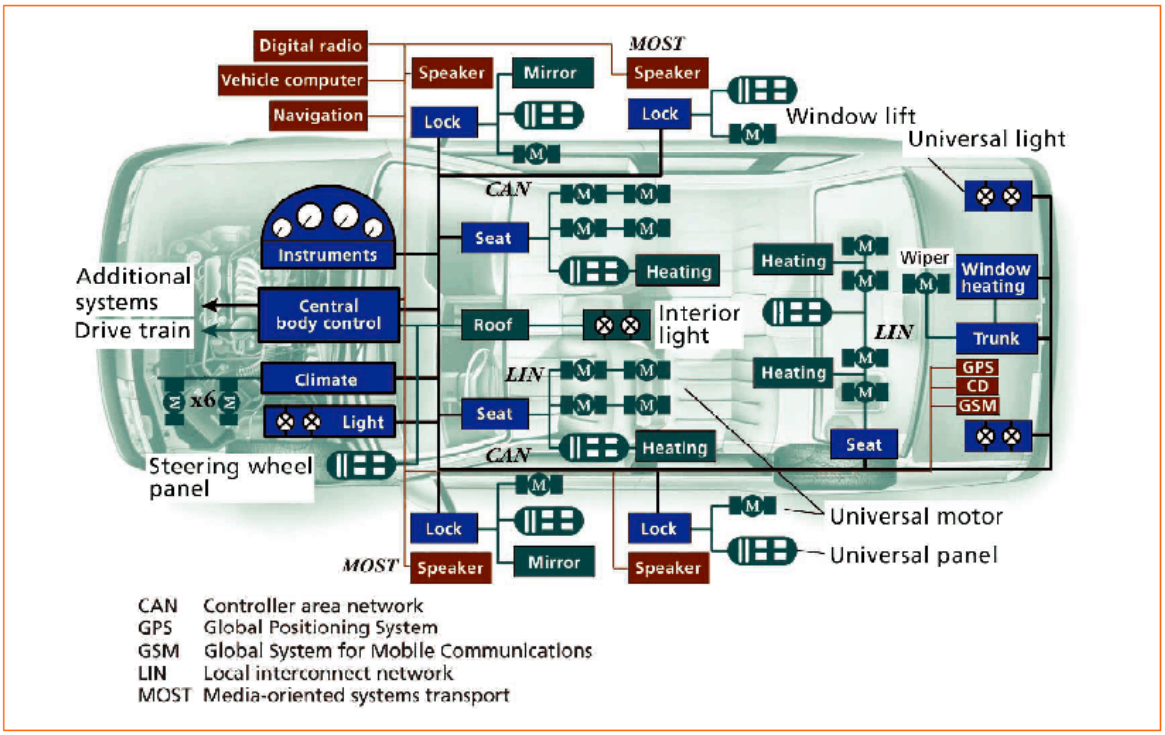

Figura 7: Rete di comunicazione per automotive

Altre applicazioni del CAN le troviamo nell'elettronica marina, nel controllo degli ascensori, nelle macchine agricole. Moltissime aziende manufatturiere ricorrono al CAN. Il settore tessile è stato uno dei pionieri ed ha aperto la porte al CAN per entrare nelle linee di produzione. Discorso a parte merita il settore dell'industria medica, dove il CAN è apprezzato moltissimo per le sue funzionalità riguardanti la sicurezza e l'affidabilità. Apparecchiature mediche quali macchine per raggi X fanno uso del CAN, così come altre apparecchiature speciali come i telescopi.

Riferimenti

CAN specification, v2.0 A e B, http://www.can.bosch.com/docu/can2spec.pdf

Fieldbus specialists, http://www.fieldbus.com.au/techinfo.htm

Motorola, Automotive area network controller applications, http://e-www.motorola.com/webapp/sps/site/application.jsp?nodeId=04J7lVNZnLNnms#blockDiagram

CAN homepage of Bosch, http://www.can.bosch.com/

CAN in Automation user group, http://www.can-cia.org/

G. Leen, D. Heffernan, Expanding Automotive Electronic Systems, IEEE Computer, Volume: 35 Issue: 1, January 2002, Pages: 88-93, http://www.cs.umd.edu/class/spring2002/cmsc818m/doc/0220/expanding.pdf

G. Tognini, Studio, progetto e realizzazione di un analizzatore di traffico su rete CAN, Tesi di lurea, Università di Pisa, http://etd.adm.unipi.it/theses/available/etd-11172003-195702/