La teoria che……

Manuale semiserio di Robotica

Realizzato quasi per scherzo per:

www.roboitalia.com

Dispensa n° 3

Comunicare con il Robot (e non solo)

Dovrebbe essere evidente che un Robot ha la necessità di comunicare, se non lo fa con l'uomo, lo farà con un PC o quantomeno con i suoi sensori, il microprocessore per prendere decisioni deve infatti interfacciarsi con il mondo esterno.
Detto questo parliamo dei principi della comunicazione.
Quando due corrispondenti devo comunicare tra loro occorre che si intendano correttamente (scontato? …..se lo dite voi).
Perché si intendano occorre stabilire delle regole di comunicazione comuni: chiamiamo PROTOCOLLO di comunicazione l'insieme delle regole che appunto regolano lo scambio di informazioni tra i corrispondenti.
Tanto per fare un esempio, pensiamo ad una telefonata fra due persone; bisogna sapere il numero del corrispondente, ci sono delle regole per trovarlo consultando l'elenco telefonico.
Bisogna eseguire la chiamata telefonica: ci sono delle regole precise per effettuarla, (alzare la cornetta, attendere il segnale, comporre il numero, …).
Una volta stabilita la connessione, occorre avere un linguaggio comune per intendersi, ad esempio l'italiano.
C'è una regola per rispondere: …pronto chi parla? … sono XXX… non ho capito, può ripetere? … XXX … ho capito, cosa desidera? Ecc. ecc.
C'è poi il contenuto della telefonata che, per essere inteso, ha di solito bisogno di seguire le regole del normale modo di pensare.
Insomma un sacco di convenzioni.
Osserviamo però che la comunicazione avviene per livelli: c'è il mezzo trasmissivo (cavo, radio,…), il tipo di segnali che si inviano attraverso il mezzo, il modo in cui si usano i segnali per formare un messaggio e il loro significato, il modo per assicurarsi che il messaggio sia arrivato al corrispondente e correttamente interpretato, in caso contrario ripeterlo.
E' convenzione allora dividere le regole, cioè il protocollo, in livelli che rispecchino i vari aspetti della comunicazione.
L'ISO ha introdotto sette livelli di "intesa" e relative "regole".
Non in tutti i sistemi sono necessari tutti e sette, noi affronteremo i 4 che normalmente sono i più usati.

1. Livello FISICO: comprende le specifiche fisiche della connessione: mezzi fisici, tipo dei segnali trasmessi. A questo livello si parla di segnali elettrici, cavi/fili di connessione, connettori, tipi di interfaccia, velocità di trasmissione,…

2. Livello LINEA: comprende le specifiche di come i segnali descritti al livello 1 sono organizzati in simboli, i simboli in messaggi e definisce le regole per ottenere un efficace scambio di messaggi fra i corrispondenti. A questo livello si parla di codici, caratteri di controllo, blocchi di dati, pacchetti,… Spesso è quel livello che si intende quando si parla semplicemente di protocollo.

3. Livello RETE: comprende le specifiche che regolano lo scambio di messaggi quando esiste più di un solo possibile corrispondente e questi sono collegati

tra loro con una rete. A questo livello si parla di punto punto, multipunto, indirizzi,… Spesso i livelli 2 e 3 vengono trattati insieme.

4. Livello APPLICAZIONE: riguarda lo scambio dei messaggi fra i programmi che "girano" sulle macchine che stanno comunicando tra loro attraverso i livelli 1, 2 e 3. I tipi di messaggi e le regole sono strettamente connessi all'applicazione, per esempio nel caso di un Robot il programma sarà interessato a gestire il movimento e le regole saranno adeguate a questo.

Faremo un'analisi pratica di ogni livello (se vi siete stancati ditelo ☺)

Livello FISICO: Collegamento via cavo
E' possibile mettere in comunicazione due dispositivi attraverso una connessione elettrica ad esempio attraverso la classica RS232 (la seriale del PC).
La RS232 specifica i livelli elettrici dei segnali da usare, il tipo di connettore, una serie di fili per trasmettere e ricevere serialmente dei segnali digitali, e una serie di fili per controllare un MODEM; il modem altro non è se non un apparecchio in grado di trasformare questi segnali digitali in analogici e viceversa, questo al solo scopo di aumentare in modo indefinito la portata (distanza), nel caso si colleghino tra loro delle macchine con distanze inferiori ai 15m possiamo fare a meno del modem e in questo caso potremo fare uso di solo 3 fili (modo chiamato "senza handshake").
Di questi tre fili, due vengono usati per convogliare i segnali da e per l'interlocutore e uno serve come riferimento comune per i livelli dei segnali.
I tre fili sono chiamati TXD (trasmissione dati), RXD (ricezione dati) e GND (ground o massa); ora nel caso in cui si connettono due apparecchiature tipo PC, è Chiaro che il TXD di uno deve essere collegato al RXD dell'altro e viceversa mentre i due GND vanno collegati tra loro. Per quanto riguarda la RS232 vi invito a consultare anche la rete, ci sono tantissime informazioni a riguardo.
L'insieme dei tre fili costituisce il canale di comunicazione.
Siccome su un filo viaggiano i segnali in trasmissione e sull'altro quelli in ricezione, la comunicazione può avvenire contemporaneamente: questo tipo di collegamento viene di solito chiamato Full Duplex.
Questa è una caratteristica della linea, può essere però che a livelli successivi vi siano restrizioni diverse e quindi la comunicazione avvenga in modo Half Duplex (è anche il caso normale di una telefonata, uno parla e l'altro ascolta).
I dati vengono trasmessi serialmente cioè un bit dopo l'altro.
Visto che ogni bit può assumere solo due valori (0 e 1) si usano due livelli elettrici di tensione, un livello corrisponde a zero e l'altro a uno. Per l'interfaccia standard RS232 il livello 1 corrisponde a –12V e il livello 0 a +12V.
Nel nostro canale inviamo un bit alla volta, questi bit però normalmente vengono raggruppati in caratteri. Un carattere è un insieme di bit a cui è associato un preciso significato; ASCII è un tipo di regola che da un significato a gruppi di bit.
Significato del gruppo di bit a parte, deve esserci la possibilità di riconoscere un carattere dal successivo o dal precedente, un metodo usato è quello di

marcare l'inizio e la fine di un carattere con uno START bit e uno STOP bit: questo permette di lasciar scorrere un tempo arbitrario tra due caratteri. Ad ogni carattere è possibile poi aggiungere altri bit di controllo ad esempio la "parità", questo tipo di controllo è un po' in disuso in quanto si preferisce fare un controllo a livello linea.
All'interno di un carattere devo poter riconoscere i singoli bit e per fare questo è sufficiente sapere quanto dura un bit, già e quanto dura un bit? Eccola qua la velocità di trasmissione, ogni volta che vi connettete alla rete, il vostro modem e quello del provider negoziano per un po' per mettersi d'accordo sulla velocità di trasmissione, ipotizziamo di avere 300bit al secondo (baud) un bit non potrà che durare 1/300 secondi, è quindi questo il modo in cui il sistema che riceve distingue un bit dall'altro.
Parlando di velocità è necessario sapere cosa e quanto posso trasmettere in un determinato tempo, sarebbe infatti stupido interrogare il Robot ogni millesimo di secondo se per farlo devo inviare comandi a 300baud. Se ipotizziamo di trasmettere 8 bit per carattere più 2 (Start e Stop) e il comando da inviare è lungo 10 caratteri (da 8+2bit cadauno) avremo che al massimo e solo teoricamente ci servirà un terzo di secondo per comunicare il comando, se a questo aggiungiamo che il Robot dovrà elaborare quanto ricevuto, confermare e quindi cominciare ad eseguire, avremo un idea dei limiti che abbiamo (diversi ovviamente se trasmettiamo a 56kbaud); abbiamo detto anche teoricamente in quanto abbiamo ipotizzato che la trasmissione avvenga senza errori e quindi senza bisogno di ripetere e, non abbiamo detto che il tempo tra un carattere e il successivo è arbitrario e dipende dal sistema in trasmissione, ecco perché tra l'altro questo sistema viene detto asincrono.

Livello FISICO (2): Collegamento via Radio
Avere un Robot con un cordone ombelicale ne limita molto le possibilità, ecco che volendo il collegamento tra PC e Robot può essere fatto via etere, in rete vi sono alcuni prodotti già pronti, non sono altro che dei modem dotati di ricetrasmittente.
Per utilizzare dei modem e per farlo con velocità decenti è necessario prevedere di utilizzare quei collegamenti specifici che avevamo tralasciato prima, al di là delle caratteristiche di questi segnali (ampia la documentazione in rete) questi servono per dare tempo e modo al modem di fare la conversione.
In realtà nel caso di comunicazione radio esistono due interfacciamenti ma il canale in radio frequenza è normalmente trasparente all'utente e quindi non ne parliamo (felici è? ☺), diremo solo che a meno di utilizzare moduli complessi che lavorano su frequenze distinte potremmo trovarci nella situazione che ricezione e trasmissione non sono possibili in contemporanea e quindi avremo un sistema Half Duplex.
Abbiamo detto che in caso di interfaccia RS232 Modem è doveroso utilizzare altri segnali, anche qui diremo poche cose, in particolare si deve sapere che il PC viene chiamato DTE e il modem DCE e che nei DCE i terminali TXD e RXD sono riferiti al DTE (in pratica il TXD del PC va collegato al TXD del modem e così anche per RXD).

Livello LINEA: Protocollo di linea
Il livello di linea o LOGICO è quello che si incarica di presiedere al buon invio di un pacchetto di dati, o messaggio fra i due corrispondenti.
Questo livello include le specifiche della forma in cui il nostro messaggio verrà inviato, normalmente chiamato pacchetto, le tecniche di rilevazione degli errori di ricezione e la loro correzione. E' importante sottolineare che a questo livello non importa il contenuto o il significato del messaggio ricevuto o trasmesso che invece riguarda il livello successivo (livello applicazione). Ci si preoccupa soltanto del corretto inoltro del messaggio.
A questo livello si deve dare una struttura al messaggio, ipotizziamo di avere anche una rete (ne parleremo tra un attimo), un'ipotetica struttura potrebbe essere la seguente:

Comunicazione PC→Robot

- Numero caratteri da trasmettere
- Identificativo Slave
- Messaggio
- Codice controllo errori

Comunicazione Robot→PC

- Identificativo Slave
- Messaggio
- Codice controllo errori

E' solo un'idea anche se valida, notate che nella comunicazione Robot→PC non è stato previsto il numero caratteri da trasmettere, non è un errore, semplicemente in una struttura Master/Slave di cui parleremo si può pensare che le risposte siano preordinate e quindi si sappia a priori in base alla domanda quanti siano i caratteri di risposta, in questo caso è inutile sprecare tempi e risorse HW e SW.

Livello RETE:
Anche semplicemente due PIC costituiscono una RETE e se invio il comando a quella sbagliata… ok è chiaro.
Il modo migliore e universalmente usato è quello di dare un indirizzo o identificativo che manco a dirlo è numerico tipo 01, 02, 03 ecc. inoltre si deve stabilire chi comanda.
Nel caso di un PC che gestisce via rete un Robot potrebbe comandare proprio il PC (Master) e gli altri PIC essere quindi Slave; in un caso simile è sempre il Master ad interrogare e gli Slave possono solo rispondere, ecco perché sarebbe di fatto inutile l'invio del numero di caratteri da parte dello Slave (se chiedo allo Slave "dammi lo stato motore", lo Slave può solo rispondere "marcia" o "fermo" cosa che si tradurrà con un solo bit a 1 o 0.

Livello APPLICAZIONE

Qui dobbiamo decidere cosa dire, è chiaro che il contenuto del messaggio varia di volta in volta e la decisione è solo di chi fa il SW.
Dovendo gestire dei motori o sensori, il cosa dirci sarà riferito alle grandezze in gioco, in generale è chiaro che le informazioni di tipo digitale (on/off) potranno essere gestite in un solo bit e quindi visto che non possiamo trasmettere meno di un carattere per volta (normalmente 8 bit → 1Byte) possiamo pensare di raggruppare almeno 8 segnali digitali (che però dovranno far capo ad un unico Slave).
Facciamo un esempio: abbiamo un PIC che gestisce due motori i quali hanno ognuno due finecorsa (uno avanti e l'altro indietro), il Master potrebbe chiedere "dammi lo stato motori" e lo Slave rispondendo con un solo Byte tipo: 01000010 (bello, che significa?), leggiamolo in verticale

0	bit non usato
1	Motore A in marcia (sarebbe a zero se fermo)
0	Finecorsa Avanti motore A non raggiunto
0	Finecorsa Indietro motore A non raggiunto
0	bit non usato
0	Motore B fermo
1	Finecorsa Avanti motore B raggiunto
0	Finecorsa Indietro motore B non raggiunto

Ecco qua, in un solo Byte abbiamo un sacco di informazioni pronte ad essere lette dal nostro SW (anche questa è solo un ipotesi).
Diverso è il caso di grandezze analogiche o di contatori (encoder), in questo caso sarà necessario usare uno o più caratteri per avere la nostra risposta, il concetto comunque non cambia, a questo livello si devono organizzare i contenuti dei messaggi veri e propri, volendo gestire un motore, potrei pensare di inviare un comando del tipo:

PC→Robot: Accelerazione, velocità, spazio, verso
Robot→PC: Ricevuto
PC→Robot: Start
Qui il PC attende il tempo di esecuzione e quindi:
PC→Robot: Dammi stato motori
Robot→PC: un byte come sopra, Passi encoder

Qui il PIC risponde con un numero di passi encoder ma, potrebbe lui stesso fare il calcolo dell'errore eventualmente commesso rispetto allo spazio richiesto.

Abbiamo accennato al controllo errori indicando un codice di controllo a livello linea, questo può avvenire anche a livello SW infatti quel codice può essere ad esempio uno XOR dei caratteri da trasmettere, ricevuto il messaggio, si esegue uno XOR dei caratteri (meno l'ultimo) e lo si confronta con l'ultimo, se sono uguali si risponde "ricevuto" diversamente "Ripetere".

Detto questo è facile capire come sia difficile far comunicare due apparati.

Spesso non si tiene nella giusta considerazione ogni livello e quindi nascono delle incongruità che sballano il tutto. Procedendo per livelli e sistemandoli uno alla volta avrete risultati migliori! Sarà poi anche più facile nel caso di errori cercare di capire a che livello siano e quindi ad esempio verificare che la velocità di comunicazione sia impostata uguale per tutti gli interlocutori prima di perdersi nelle righe del programma alla ricerca di un improbabile errore.

TIPS:

Abbiamo detto che è facile estrarre informazioni dalla ricezione di:
01000010, ecco come si fa…
E' vero che riceviamo i bit ma poi a livello applicazione il tutto viene ricevuto come un carattere, così come sarà necessario specificare un carattere per trasmettere una determinata sequenza di bit.
Per ogni linguaggio di programmazione esistono specifici comandi per convertire un binario in carattere e viceversa quindi quando scriviamo 01000010 sappiate che esiste un solo carattere equivalente.
Bene ora passiamo all'estrazione del solo bit che ci interessa:
Ci interessa sapere se il motore B è in marcia o no.
Se facciamo fare al SW un operazione del tipo:

01000010 (dato ricevuto)
AND
00000100 (maschera motore B)

possiamo avere solo due risultati o 00000000 come in questo caso in cui il motore è fermo oppure 00000100 nel caso di motore in marcia, diventa facile costruire quindi una struttura del programma che dica: "se risultato=0 allora…" "se risultato=4 (0000100) allora…".
In pratica per ogni bit da estrarre dovremo avere la sua "maschera" che sarà composta da tutti zeri meno il bit interessato che sarà a 1.

Come sempre domande, risposte e integrazioni da parte di tutti sul sito di:

www.roboitalia.com