Il telecomando a infrarossi

La funzione di un qualsiasi telecomando è quella di comandare qualcosa a distanza, nel tuo caso il robot. Il prefisso tele-, infatti, significa 'lontano' (si pensi ai termini televisione, telefono, telecinesi). Esistono diversi tipi di telecomando, a seconda della tecnologia impiegata per comunicare con l'apparecchio da controllare: ci sono, ad esempio, telecomandi a cavo, a radiofrequenza, a infrarossi ecc. Il tipo più comune è quello a infrarossi (diffusissimo, in quanto usato per comandare televisori, cancelli, HI-FI ecc.), categoria cui appartiene anche quello che ti è stato fornito. Perché un telecomando possa comunicare con l'apparecchio da controllare è necessario che quest'ultimo sia dotato di un sistema ricevente (anche solo di tipo hardware, oppure misto hardware e software), in grado non solo di recepire i segnali inviati dal telecomando, ma anche e soprattutto di decodificarli. Per gestire i propri comandi, infatti, ogni telecomando impiega una codifica specifica, che deve essere 'capita' e interpretata dall'apparecchio ricevente.

Dopo averlo alimentato con due pile classe AA, sperimenta subito il telecomando scaricando i programmi dal sito

Nel caso del sistema ricevente del tuo robot, la parte hardware è composta da uno dei due sensori a infrarossi che hai già montato sulla scheda madre; per quanto riguarda invece la decodifica dei segnali, dovrai operare a livello software. Ricordati che, prima di poter utilizzare il telecomando, dovrai alimentarlo con due pile classe AA. Se presto imparerai come programmarlo, fin da ora puoi invece scaricare dal sito www.robot.deagostini.it alcuni programmi eseguibili immediatamente, creati per te dagli esperti: tali programmi sfruttano la particolare dicitura dei tasti, dettagliata in didascalia, che facilita il controllo di alcune funzioni preferenziali (ad esempio il movimento dei motori e l'esecuzione dei programmi).

Le funzioni dei tasti del telecomando: STOP arresta l'esecuzione del programma in corso; i tasti ms, md e M controllano, rispettivamente, il motore sinistro, il motore destro e i due in coppia; i tasti P controlleranno i movimenti della pinza robotica segnalati dalle frecce (apri, chiudi, alza e abbassa); prg manda in esecuzione il programma successivo (contenuto nella memoria del robot); 1, 2, 3... 17 non sono dedicati (potrai attribuire loro le funzioni più opportune).

Le fasi di programmazione

Con l'istruzione goto, abbiamo visto come sia possibile alterare l'esecuzione sequenziale di un programma, saltando ad altre porzioni del codice contrassegnate da un'etichetta. Ora introdurremo due nuove istruzioni di salto, if...then e gosub, che avremo modo di incontrare molto spesso nei programmi di navigazione che andremo a costruire.

L'istruzione if... then (dall'inglese, 'se... allora') è una cosiddetta istruzione di test o di salto condizionato: consente infatti di saltare a un'etichetta, solo se una determinata condizione risulta vera. La sintassi completa dell'istruzione è: *if condizione then etichetta*. Questo significa che solo se la verifica (il test) di quanto contenuto in *condizione* risulta vera, allora l'istruzione rimanda all'esecuzione della parte di codice contrassegnata da *etichetta* (come avverrebbe con l'istruzione goto). Dunque, *condizione* deve essere un'espressione di tipo particolare che, valutata durante l'esecuzione del programma, deve poter generare come risultato *true* ('vero') o *false* ('falso'): nel primo caso viene eseguito il salto, nel secondo invece l'esecuzione procede normalmente dall'istruzione successiva. L'espressione contenuta in *condizione*, detta espressione booleana, è di solito costituita dal confronto tra più valori legati da operatori, detti operatori condizionali. Il confronto ha la seguente forma: Valore1 *Operatore* Valore2. Valore1 e Valore2 possono essere variabili, numeri o espressioni algebriche; *Operatore*, invece, deve essere uno degli operatori condizionali (riportati nella tabella in alto a destra). Ad esempio, supponiamo di avere, all'interno di un programma, una variabile *x* (di qualunque tipo) e la seguente istruzione: if (x <> 1) then Procedi. L'espressione completa significa: 'se il valore della variabile *x* è diverso da *1*, allora salta all'etichetta *Procedi*'. In particolare, *(x<>1)*

Operatore condizionale	Significato
=	Uguale a
<>	Diverso da
<	Minore di
>	Maggiore di
<=	Minore o uguale a
>=	Maggiore o uguale a

è la condizione (*x* e *1* sono il valore1 e il valore2, <> è l'operatore che li confronta) e *Procedi* è l'etichetta. Un confronto analogo, tramite operatori condizionali, può avvenire anche tra due variabili quali, ad esempio, *x* e *y* (anche di tipo diverso); in tal caso un'istruzione come if (x <= y) then Procedi significa: 'se il valore della variabile *x* è minore o uguale (<=) a quello della variabile *y*, allora salta all'etichetta *Procedi*'. Come avrai notato, il simbolo uguale (=) ha di fatto un duplice utilizzo possibile. In precedenza abbiamo già visto che, in un'istruzione di assegnamento, può essere usato per memorizzare il valore di una variabile: l'istruzione x = 1, infatti, assegna alla variabile *x* il valore 1. All'interno di un'espressione booleana, invece, è utilizzato come operatore condizionale di confronto: l'istruzione if (x = 1) then Procedi, per esempio, confronta il valore della variabile *x* con il valore 1 per poi decidere se effettuare il salto.

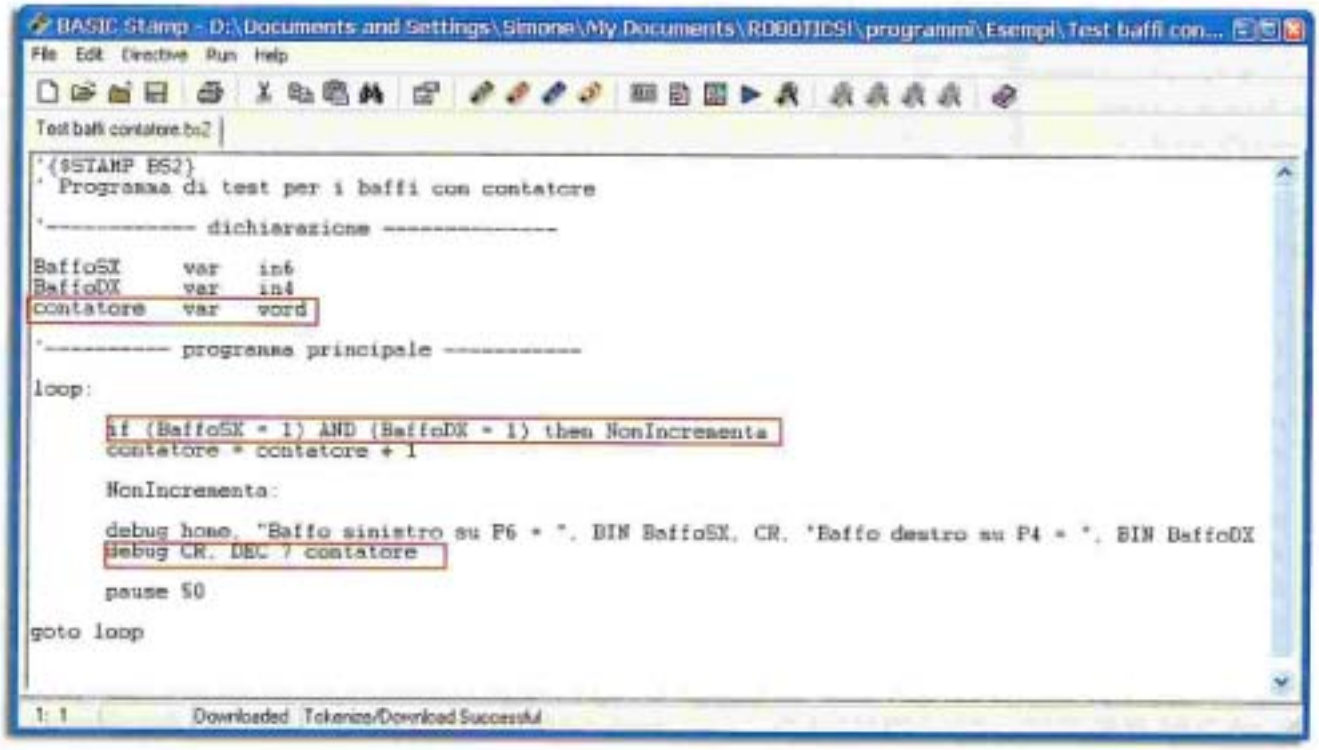
Operatore logico	Forma simbolica	Forma abbreviata
AND	AND	&
OR	OR	|
NOT	NOT	~
XOR	XOR	^

In un'espressione booleana è possibile inserire, oltre agli operatori condizionali, anche i cosiddetti operatori logici, ossia AND, OR, NOT e XOR, utilizzati per costruire espressioni logiche che contengano più di un confronto. Questi operatori possono comparire sia in forma simbolica sia in forma abbreviata (secondo la notazione riportata nella tabella sopra). Il carattere (~) si ottiene attraverso la scorciatoia da tastiera Alt+0126 (ossia tenendo premuto il tasto Alt e digitando, in sequenza, i numeri 0, 1, 2, 6); il carattere (|), invece, si ottiene con Alt+0124. Degli operatori logici utilizzabili nel linguaggio PBASIC, il NOT è unario (cioè applicabile a un solo valore logico); gli altri tre operatori sono invece binari (cioè applicabili a due valori logici). Secondo le tabelle di verità (sotto), l'operatore NOT inverte (nega) il valore logico in ingresso (A); un confronto tramite l'operatore AND (che corrisponde alla congiunzione italiana 'e') risulta vero solo quando entrambi i valori in ingresso (A e B) sono veri; invece, un confronto con OR (che corrisponde alla 'o', intesa in senso inclusivo), risulta vero quando almeno uno dei valori in ingresso (A o B) è vero. Infine, con XOR (che corrisponde alla 'o', in senso esclusivo) un confronto risulta vero quando uno solo dei due valori in ingresso (o A o B) è vero.

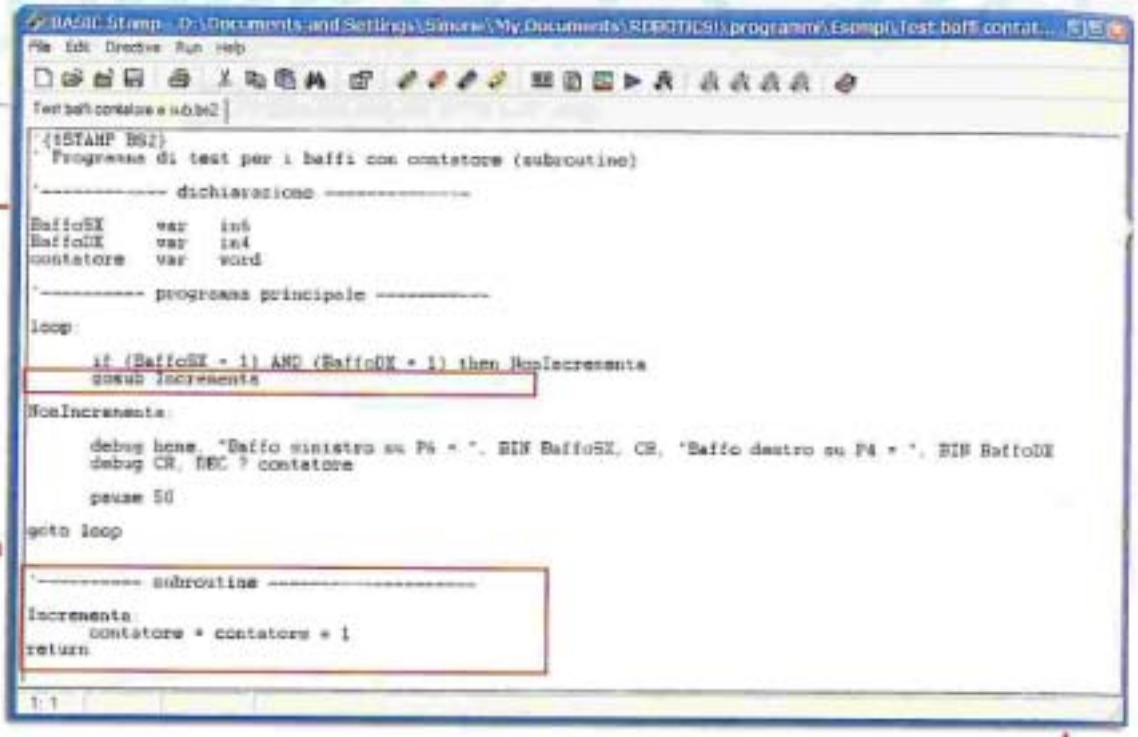
A	NOT A
vero	falso
falso	vero

A	B	A AND B	A OR B	A XOR B
falso	falso	falso	falso	falso
falso	vero	falso	vero	vero
vero	falso	falso	vero	vero
vero	vero	vero	vero	falso

Una condizione può anche utilizzare entrambi i tipi di operatori. L'istruzione if (x >= 1) AND (x < 23) then Procedi significa: 'se il valore della variabile *x* è maggiore o uguale a (operatore condizionale) 1 e (operatore logico) minore di (condizionale) 23, allora salta all'etichetta Procedi'. Nell'espressione più complessa: if ((x <> y) | (z < 16)) & (y > z) then Procedi, ai due risultati dei confronti (x <> y) e (z < 16), racchiusi in un'unica parentesi, viene dapprima applicato l'operatore OR (|); il risultato viene quindi valutato in AND (&) con il terzo confronto (y > z). In questo caso è bene aiutarsi con le parentesi (utilizzandole anche doppie ((e)) per distinguere la successione dei confronti operati e per rendere meno ambigua l'interpretazione dell'espressione.



Proviamo ora a utilizzare l'istruzione if... then in un esempio. In particolare, modifichiamo l'ultima versione del programma di test per i baffi al fine di stampare a video lo stato delle porte corrispondenti, ma anche il numero di volte in cui i baffi sono stati toccati. Per fare questo è necessario dichiarare una nuova variabile che, di volta in volta, memorizzi il numero di 'tocchi'. Digita nell'area di editing il listato (sopra), collega il robot al computer con il cavo seriale, accendilo e premi il pulsante Run della barra degli strumenti. Nella finestra di debug (in basso a sinistra) compare ora, nell'ultima riga, il nome (contatore) e il valore (8) della nuova variabile che verrà incrementata di una unità ogni volta che uno dei baffi verrà toccato. Vediamo le modifiche (sopra, riquadrate) apportate al programma per ottenere tale risultato. Innanzitutto è stata dichiarata, in word, la variabile contatore; inoltre ci sono nuove istruzioni. La prima è l'istruzione di salto condizionato if (BaffoSX = 1) AND (BaffoDX = 1) then NonIncrementa che verifica lo stato dei due baffi per decidere se saltare all'etichetta *NonIncrementa*: se le porte relative ai due baffi hanno entrambe (AND) valore 1 (se cioè i baffi non sono stati toccati) avviene il salto all'etichetta specificata. In caso contrario (se cioè uno dei baffi è stato toccato), non c'è alcun salto e viene eseguita l'istruzione successiva (contatore = contatore + 1) che incrementa di un'unità il valore della variabile (ad esempio, il valore di contatore verrà portato da 8 a 9, cioè 8+1). Infine è stata aggiunta l'istruzione per stampare a video (debug), a capo (CR), il nome (?) e il valore decimale (DEC) della variabile contatore.



Come si è visto, all'interno del programma le istruzioni goto e if... then impongono all'esecuzione un salto definitivo in un'altra porzione di codice. Esiste però un'altro tipo di istruzione di salto, detta gosub (contrazione di *GO to SUBroutine*, ossia 'Vai alla SUBroutine'), che mantiene in memoria la posizione da cui parte il salto cosicché, una volta incontrata l'istruzione return (cioè 'ritorna'), l'esecuzione del programma possa tornare esattamente all'istruzione immediatamente successiva a gosub. Rispetto al listato del programma principale o main, una subroutine è una porzione di codice (di solito scritta dopo il main) destinata all'esecuzione di alcune istruzioni a cui si accede solo in caso sia necessario svolgere un particolare lavoro. La struttura di una subroutine è etichetta: (a capo) istruzione1 (a capo) istruzione2 (a capo)... istruzioneN (a capo) return. Tutto ciò che è compreso tra i due punti (:) e il comando return (fondamentale perché l'esecuzione torni all'istruzione succesiva al salto) viene detto codice di una subroutine. Come detto, per chiamare una subroutine, ossia per saltare alla porzione del listato relativa a essa, si utilizza l'istruzione gosub, seguita dall'etichetta (ossia il nome simbolico) che contraddistingue la subroutine che si vuole chiamare.
La sintassi del comando gosub è molto simile a quella di goto: a partire dall'etichetta specificata, infatti, manda in esecuzione particolari istruzioni; tuttavia, attraverso il comando return, può riprendere l'esecuzione del programma principale dal punto in cui lo aveva lasciato per il salto. Vediamo ora un semplice esempio di utilizzo di una subroutine. Riprendendo il programma di pagina 67, vediamo come sia possibile inserire una subroutine per ottenere il medesimo risultato (visualizzato nella finestra di debug). Digita nell'area di editing il listato (sopra), collega il robot al computer con il cavo seriale, accendilo e premi il pulsante Run della barra degli strumenti. Come puoi verificare toccando i baffi del robot, questo programma produce lo stesso effetto del precedente. Le modifiche apportate riguardano l'inserimento dell'istruzione gosub e della subruotine etichettata Incrementa che, in questo caso, incrementa il valore della variabile contatore.

La subroutine è stata inserita esternamente al ciclo infinito (loop), in modo che non possa essere eseguita di seguito al normale procedere del programma (sfruttando semplicemente il fatto che il loop, di per sé, non permette di procedere oltre). Tutte le subroutine di un programma, infatti, vengono scritte dopo il listato principale, con l'accortezza che siano raggiungibili solamente da un'istruzione gosub. Nel listato, l'istruzione di salto gosub è stata inserita subito dopo if... then, sostituendo cioè l'istruzione contatore = contatore + 1 che, nella versione precedente del programma, aggiornava il valore della variabile; quest'ultima, infatti, è stata inserita nella subroutine. Durante l'esecuzione del programma, la chiamata della subroutine Incrementa viene eseguita solo se la condizione che segue if non è soddisfatta (dunque se almeno uno dei due baffi viene toccato e, quindi, una delle porte vale 0). In tal caso, la subroutine incrementa il valore di contatore e, attraverso return, riporta l'esecuzione del programma alla riga successiva a gosub, etichettata NonIncrementa (l'andamento dell'esecuzione del programma relativo alla subroutine è indicato dalle frecce sotto). Una stessa subroutine può essere chiamata in più punti del listato, come fosse un piccolo programma a sé stante. In questo modo si risparmia spazio in memoria e si facilita l'eventuale modifica del codice. Ad esempio, in un programma che prevede la ripetizione di una stessa sequenza di tre istruzioni in posizioni diverse, è utile creare una sola subroutine contenente quella sequenza. Il codice, infatti, sarà semplificato e anche meglio organizzato; inoltre, volendo modificare la sequenza, basterà farlo una volta nel codice della subroutine, anziché eseguire più volte l'operazione nel main.