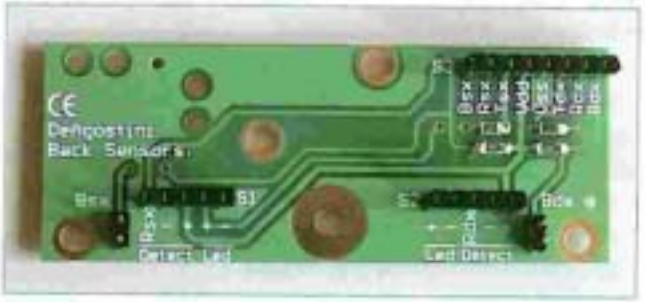
Circuito posteriore

Il circuito allegato a questo fascicolo ti offrirà presto un'interessante alternativa per implementare il robot. Sostituito al manipolatore (la pinza) e opportunamente collegato al microcontrollore, questo circuito potrà di fatto ospitare un secondo set di sensori a raggi infrarossi e di contatto (baffi). Un'opzione utile, ad esempio, per la navigazione del robot in un labirinto.



Nella foto. Il circuito DeAgostini Back Sensors.

Le fasi di programmazione

Come visto a pagina 140, a ogni tasto del telecomando è associato un codice numerico univoco che ne permette il riconoscimento all'atto della ricezione. Grazie alla tabella tasto-codice numerico, disponi ora delle informazioni necessarie per costruire programmi per telecomandare il robot. Il primo passo consiste nell'assegnare ai singoli tasti una funzionalità desiderata. Una volta stabilita l'associazione tasto-funzionalità, potrai poi procedere all'implementazione software per la gestione corretta dei comandi ricevuti.

Come esempio costruiamo un programma di navigazione che consenta al robot di eseguire i movimenti desiderati, comunicatigli tramite il telecomando. A tal fine scegliamo di utilizzare i sei tasti, posti nella parte centrale del telecomando, contrassegnati dalle frecce: queste, infatti, ti aiuteranno a ricordarne la funzione. Qualunque sia la funzione prescelta per ogni tasto, però, va assegnata in fase di programmazione; in particolare, assegneremo ai tasti M l'avanzamento e l'arretramento del robot e ai tasti ms e md la rotazione verso sinistra o verso destra. Consultando la tabella tasto-codice numerico che hai costruito (a pag. 140), noterai che questo insieme di tasti è caratterizzato dall'avere un codice numerico che comprende tutti i numeri interi da 0 a 5: come vedremo a breve, questo ne semplificherà il programma di gestione. Digita nell'area di editing il listato della pagina seguente (modificando i valori delle costanti relative alla calibrazione, come visto alle pagg. 123-124), premi il tasto Run della barra degli strumenti e scollega il cavo seriale: come puoi verificare premendo uno dei sei tasti centrali del telecomando, a ciascuno di essi corrisponde un movimento del robot. Perché tale movimento sia continuo, basterà tener premuto il tasto.

Tornando al listato, noterai che la sua struttura è in gran parte analoga a quella del programma (visto a pag. 138) di test del telecomando. Il codice è organizzato in quattro parti: la parte dichiarativa iniziale, il programma principale, le routine di navigazione e la subroutine di decodifica. Per quanto riguarda la parte dichiarativa, alle variabili e costanti viste per il programma di test sono state aggiunte tre variabili: due (movDx e movSx) di tipo word e una (contatore) di tipo nibble. Per quanto riguarda le costanti sono state aggiunte le informazioni necessarie all'utilizzo dei servo motori (SERVO_DX e SERVO_SX) e i valori relativi alla calibrazione (quelli ideali, non ancora calibrati). Il programma principale è costituito dall'inizializzazione delle porte dei motori (le due istruzioni low) e da un ciclo infinito delimitato dall'etichetta main e dall'istruzione goto main. All'interno del ciclo viene chiamata (gosub) una subroutine (decodifica), identica a quanto visto in precedenza, che consente di memorizzare nella variabile codicePulsante il valore numerico relativo al tasto del telecomando premuto. L'istruzione di salto successiva è un'istruzione branch (vista a pag. 70) applicata alla variabile codicePulsante. Come annunciato, il fatto che i sei tasti utilizzati abbiano come codice numerico tutti i valori interi compresi tra 0 e 5 si rivela utile per gestirli in modo semplice ed elegante, inserendoli opportunamente in un'unica istruzione branch. In questa maniera, a seconda del valore di codicePulsante, è possibile saltare all'etichetta della routine che gestisce la corrispondente funzionalità. Nel nostro caso, le associazioni volute sono riassunte nella tabella a sinistra.

Valore codicePulsante	Etichetta di salto	Funzionalità
0	AvantiSx	Rotazione verso sinistra
1	Avanti	Avanzamento
2	AvantiDx	Rotazione verso destra
3	IndietroSx	Rotazione verso sinistra
4	Indietro	Arretramento
5	IndietroDx	Rotazione verso destra

LE FASI DI PROGRAMMAZIONE

Una soluzione alternativa all'istruzione branch sarebbe una serie di istruzioni if... then, una per ogni valore della variabile codicePulsante: è evidente perciò che la soluzione adottata nel codice è più compatta e, quindi, da preferirsi. Tuttavia, è bene ricordare che l'uso di branch è possibile solo a patto che, come in questo caso, i valori da esaminare costituiscano una successione completa di valori interi a partire dallo 0. Utilizzando i tasti del telecomando corrispondenti ad altri codici numerici, dunque, questa soluzione non sarebbe immediatamente praticabile (tuttavia, come vedremo in seguito, esiste un metodo alternativo per utilizzare branch anche in questi casi). Tornando al listato (a destra), vediamo ora le routine di navigazione. Rispetto ad altri programmi (ad esempio, quello a pag. 123), in cui ogni movimento del robot esigeva la ripetizione delle istruzioni di pulsout (presente quindi in tutte le routine di navigazione), il metodo proposto ora è diverso. Grazie a due variabili di tipo word (movDx e movSx), infatti, non è più necessario che le routine di movimento (Avanti, AvantiDx/Sx, Indietro, IndietroDx/Sx) contengano le istruzioni di pulsout. La gestione dei servomotori, allora, viene affidata a un'unica routine (esegui) cui tutte le altre rimandano. A essa spetta il compito di eseguire (con un ciclo for) una serie di dieci impulsi pulsout su ciascun motore, utilizzando come durata dell'impulso il valore memorizzato nelle due variabili movDx e movSx. Tale valore è settato dalle routine di movimento, il cui compito è quello di associare (=), appunto, movDx/Sx alle opportune costanti (ORARIO_DX/SX, ANTIORARIO_DX/SX) relative ai valori di durata (500 o 1000) necessari per i vari movimenti. La routine Avanti, per esempio, applica l'impulso ORARIO_DX (cioè 500) al motore destro e l'impulso ANTIORARIO_SX (500) al sinistro; memorizzando questi valori nella routine esegui, in movDx e movSx, verrà quindi eseguito l'effettivo avanzamento del robot. Riassumendo, la successione delle operazioni svolte dal main prevede la decodifica del segnale ricevuto (gosub decodifica) e la memorizzazione del corrispondente codice in codicePulsante. Segue un test di branch su codicePulsante che stabilisce la routine di movimento a cui saltare: questa, quindi, setta le variabili movDx e movSx in modo opportuno e delega poi alla ruotine esegui l'effettiva realizzazione del movimento telecomandato.

```
'{$STAMP BS2}

'----------------- Dichiarazione Variabili -----------------

contatore       var     nib

codicePulsante  var     byte

movDx           var     word
movSx           var     word
segnaleInizio   var     word
inputIR0        var     word
inputIR1        var     word
inputIR2        var     word
inputIR3        var     word
inputIR4        var     word
inputIR5        var     word
inputIR6        var     word

'----------------- Dichiarazione Costanti ------------------

IR_SX           con     8
BASSO           con     0

SERVO_DX        con     13
SERVO_SX        con     12

ANTIORARIO_DX   con     1000
ANTIORARIO_SX   con     1000
ORARIO_DX       con     500
ORARIO_SX       con     500

'----------------- Programma Principale --------------------

low SERVO_DX
low SERVO_SX

main:

      gosub decodifica
      branch codicePulsante, [AvantiSx, Avanti, AvantiDx, IndietroSx, Indietro, IndietroDx]

goto main

'---------- Routine di navigazione ----------------------

AvantiSx:
      movDx = ANTIORARIO_DX     : movSx = ANTIORARIO_SX   : goto esegui

Avanti:
      movDx = ORARIO_DX         : movSx = ANTIORARIO_SX   : goto esegui

AvantiDx:
      movDx = ORARIO_DX         : movSx = ORARIO_SX       : goto esegui

IndietroSx:
      movDx = ORARIO_DX         : movSx = ORARIO_SX       : goto esegui

Indietro:
      movDx = ANTIORARIO_DX     : movSx = ORARIO_SX       : goto esegui

IndietroDx:
      movDx = ANTIORARIO_DX     : movSx = ANTIORARIO_SX   : goto esegui

esegui:
for contatore = 1 to 10
            pulsout SERVO_DX, movDx
            pulsout SERVO_SX, movSx
            pause 10
      next
goto main

'---------- Subroutine decodifica impulsi IR ------------

decodifica:
   rileva:
      pulsin IR_SX, BASSO, segnaleInizio
      pulsin IR_SX, BASSO, inputIR0
      pulsin IR_SX, BASSO, inputIR1
      pulsin IR_SX, BASSO, inputIR2
      pulsin IR_SX, BASSO, inputIR3
      pulsin IR_SX, BASSO, inputIR4
      pulsin IR_SX, BASSO, inputIR5
      pulsin IR_SX, BASSO, inputIR6

   if segnaleInizio < 1000 then rileva

      codicePulsante.bit0 = inputIR0.bit9
      codicePulsante.bit1 = inputIR1.bit9
      codicePulsante.bit2 = inputIR2.bit9
      codicePulsante.bit3 = inputIR3.bit9
      codicePulsante.bit4 = inputIR4.bit9
      codicePulsante.bit5 = inputIR5.bit9
      codicePulsante.bit6 = inputIR6.bit9

return
```