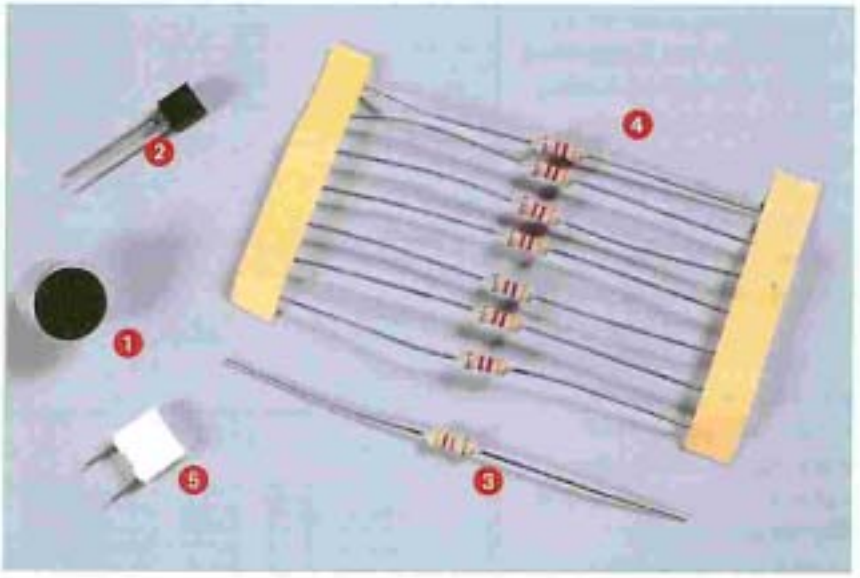
Completando il kit per il microfono

Ecco i componenti che completano il kit relativo al microfono. In particolare, la capsula fornirà, tramite i pin, il segnale in microvolt (espresso in decibel) che dovrà essere amplificato; il transistor è invece l'intermediario tra l'amplificatore operazionale LM 358 e il BS2. I resistori, tra l'altro, fissano il guadagno di amplificazione, mentre il condensatore 'filtra' l'audio captato dal microfono.



A destra. **Gli allegati a questo fascicolo: capsula microfonica 1, transistor 2, otto resistori (di cui uno da 47 kΩ 3 e sette da 4700 Ω 4) e condensatore da 100 nF 5.**

Le fasi di programmazione

Riprendiamo ora il discorso relativo ai sensori a infrarossi (IR), in vista però dell'utilizzo del telecomando (presentato a pagina 65). Grazie a questo dispositivo IR, infatti, è possibile comandare a distanza i movimenti e le azioni del robot. Perché questo sia possibile, però, è necessario programmare in modo opportuno il robot. Ci occuperemo ora della fase preliminare, ossia di decodificare il segnale corrispondente a ciascun tasto del telecomando, in modo tale che il robot possa 'sentire' e 'capire' il comando di volta in volta impartitogli con il telecomando. Solo così, infatti, sarà poi possibile assegnare una funzione precisa a ogni tasto.

Il telecomando IR che ti è stato fornito è un comune telecomando TV: alla pressione dei tasti, in pratica, corrisponde l'invio di particolari segnali IR, ciascuno caratteristico di un singolo tasto. Ogni segnale (costituito da impulsi) deve essere poi decodificato dall'apparecchio ricevente (la TV o, nel nostro caso, il robot), in modo tale da istituire una corrispondenza univoca tra segnale IR, tasto premuto e funzione a esso attribuita. Nel caso della TV, le funzioni specifiche (cambiare canale, regolare il volume ecc.), prestabilite dal produttore dell'apparecchio, sono fisse; nel caso del tuo robot, invece, non esiste alcun legame a priori tra un tasto e una funzione (ossia il comando da impartire al robot). Come vedremo presto, infatti, tale legame dipenderà solo dalle scelte di programmazione. L'aspetto più importante che riguarda l'utilizzo del telecomando, comunque, è la decodifica del segnale inviato mediante la pressione di un tasto: a ciascun tasto, infatti, è abbinato un codice numerico che viene inviato (dall'emettitore del telecomando al ricevitore di uno dei sensori IR del robot) sotto forma di una specifica serie di impulsi (ossia il segnale) che identifica univocamente il tasto e ne consente il riconoscimento all'atto della ricezione.

LE FASI DI PROGRAMMAZIONE

Vediamo allora un programma di test per la decodifica del segnale trasmesso dalla pressione di ciascuno dei tasti del telecomando. Digita il codice (a destra) nell'area di editing, mandalo in esecuzione e, puntando il telecomando verso il robot (che deve rimanere collegato al PC con il cavo seriale), premi uno dei tasti (ad esempio, il 7). Il programma visualizzerà nella finestra di debug (nella pagina accanto) il codice numerico (ad esempio, 116) relativo, di volta in volta, al segnale del tasto premuto (7). Testando il segnale di ogni tasto, otterrai infine gli equivalenti codici reali del segnale, utili per la programmazione degli abbinamenti tasto-funzione. A questo proposito, potrà esserti utile creare una tabella (come quella a pag. 140) ove annotare, per ciascun tasto, il corrispondente codice numerico ottenuto con questo programma. Prima di analizzare il listato in dettaglio, vale la pena fare due premesse. Noterai innanzitutto che, per ricevere i segnali dal telecomando, il programma sfrutterà il ricevitore IR sinistro (IR_SX, collegato a P8). Tale scelta è del tutto arbitraria: a condizione di aggiornare le relative parti del listato, infatti, nulla vieta di utilizzare il ricevitore destro. In secondo luogo, per evitare interferenze è assolutamente necessario che, dei componenti che costituiscono il sensore IR (destro o sinistro che sia), il trasmettitore (LED) non venga mai utilizzato in abbinamento con l'uso del telecomando (dotato, a sua volta di un trasmettitore: sarà quest'ultimo che dovrà 'dialogare' con il ricevitore del robot). Detto questo, torniamo al listato. Il programma è organizzato in tre parti: quella dichiarativa iniziale, il programma principale (che, tramite un ciclo infinito, si occupa di visualizzare il codice del tasto premuto) e una subroutine di ricezione e decodifica del segnale (il cuore del programma).

BASIC Stamp - D:\Documents and Settings\Simone\Desktop\NuoviTelecom\provaart.bs2

File Edit Directive Run Help

provaart.bs2

```
'{$STAMP BS2}

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

codicePulsante    var         byte

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

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

main:

      gosub decodifica
      debug CLS, "Il codice del pulsante premuto è: ", DEC codicePulsante

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
```

1: 1

Nella dichiarazione iniziale troviamo innanzitutto la variabile *codicePulsante*, di tipo byte, in cui verrà memorizzato il valore del segnale ricevuto (cioè il codice numerico del tasto del telecomando); seguono poi altre otto variabili (*segnaleInizio*, *inputIR0*... *inputIR6*), di tipo word, che verranno utilizzate nella subroutine di ricezione e decodifica del segnale. Per quanto riguarda invece le costanti dichiarate, IR_SX contiene la porta logica (8) relativa al ricevitore IR sinistro, mentre la costante *BASSO*, che indica il valore logico basso (0), verrà utilizzata nella parte della subroutine relativa alla ricezione del segnale.

All'interno del programma principale troviamo un ciclo infinito delimitato dall'etichetta *main* e dall'istruzione goto *main*. Tale ciclo infinito assolve a due funzioni: chiama (con l'istruzione di salto gosub) la subroutine *decodifica* per la ricezione e la decodifica del segnale IR; e visualizza (tramite l'istruzione debug) in forma decimale (DEC) il codice numerico (binario) ricevuto. Nella finestra di debug sotto, ad esempio, il codice 116 (che corrisponde al tasto 7 del telecomando), deriva dalla decodifica del numero binario 1110100 (i cui singoli bit, come vedremo, derivano rispettivamente dalla decodifica degli impulsi del segnale inviato). Veniamo allora al cuore del programma: la subroutine *decodifica* che consente di rilevare il segnale proveniente dal telecomando (cioè la serie di impulsi IR relativi alla pressione di un dato tasto) e di decodificarlo, fornendo l'informazione numerica che, infine, contraddistingue univocamente il tasto premuto. La subroutine è infatti organizzata in due parti: una per la ricezione e memorizzazione degli impulsi, l'altra per la decodifica dell'informazione in essi contenuta.

Vediamo anzitutto come è organizzata la prima parte, costituita dalla sequenza di otto istruzioni pulsin e dall'istruzione if... then. La funzione di ricezione del segnale è svolta dal comando pulsin che, collegato a una porta logica, consente di rilevare la durata di un singolo impulso e di memorizzarne il valore in una variabile. La sintassi completa dell'istruzione, infatti, richiede che a pulsin segua l'indicazione (tramite una variabile, una costante o un'espressione numerica) della *porta* interessata dal rilevamento (nel listato, IR_SX, ossia la porta 8); a essa deve seguire poi (sempre tramite una variabile, una costante o un'espressione numerica) lo *stato*, ossia il valore logico (0 oppure 1) a cui l'impulso deve trovarsi per essere interessato dalla rilevazione; nel nostro caso, ad esempio, pulsin andrà a misurare l'intervallo di tempo in cui ogni impulso sarà BASSO (cioè pari a 0): la rilevazione, cioè, comincerà solo quando alla porta 8 si registrerà una transizione di valore logico da alto a basso. Allo *stato* segue, infine, una *variabile*, solitamente di tipo word, nella quale memorizzare il valore della durata dell'impulso (nel listato, la serie di inputIR); come abbiamo già visto (tra l'altro a pag. 113, a proposito del comando pulsout), anche in questo caso il valore memorizzato è espresso in unità di misura di 2 µs. Nel suo complesso, allora, la parte di subroutine dedicata alla ricezione esegue una serie di otto rilevamenti sulla porta (8) del ricevitore IR, alla ricerca di un segnale di tipo basso. Il tuo telecomando, infatti, genera un segnale relativo a ogni tasto composto da otto impulsi. Il primo, come vedremo, va distinto dai successivi; per questo è utile distinguere anche il nome della variabile in cui sarà memorizzata la sua durata (*segnaleInizio*); i successivi sette impulsi, invece, saranno memorizzati nelle sette variabili *inputIR*. Tale distinzione dipende dal tipo di trasmissione del telecomando, che prevede l'invio di un riconoscimento iniziale (è il primo impulso, che avvia la sequenza di impulsi utili al riconoscimento vero e proprio). Questo primo impulso, di fatto, pur non trasportando un'informazione sul tasto che lo ha generato, serve però a segnalare l'arrivo e l'ordine degli impulsi 'utili'. Questi ultimi, sono i sette impulsi successivi che, nel complesso, codificano (sotto forma binaria) il numero contenente l'informazione relativa al tasto premuto (ad esempio, 1110100, decodificato poi nel decimale 116). In particolare, tramite la sua durata, ogni impulso codifica un bit di tale numero binario: se la durata dell'impulso trasmesso è inferiore a un determinato valore di soglia, il corrispondente bit vale 0; viceversa, se la durata supera la soglia, il bit vale 1. Come si è detto, il primo impulso inviato determina l'ordine dei successivi (garantisce cioè la validità dell'intera sequenza di trasmissione, definendo quale sia il secondo, il terzo ecc.): di solito, per distinguerlo dagli altri, ha una durata (memorizzata in *segnaleInizio*) maggiore di quella dei successivi, stimata maggiore di 2 ms. Ed è per questo motivo, dunque, che si ricorre all'istruzione if *segnaleInizio* < 1000, laddove il valore 1000 ha per unità di misura 2 µs e, dunque, equivale a 2 ms. In pratica il programma, identificato il primo impulso dalla durata, potrà 'sincronizzarsi' con la trasmissione dei successivi, potendo poi procedere rilevandone la durata nell'ordine corretto.

LE FASI DI PROGRAMMAZIONE

Passiamo ora alla seconda parte della subroutine, che riguarda la decodifica dei segnali ricevuti. Come si diceva, l'ordine di trasmissione degli impulsi rispecchia, sempre a partire da quello meno significativo (più a destra), l'ordine dei bit del numero binario (ad esempio, 1110100) che andrà a codificare un tasto preciso del telecomando (il 7). Pertanto, il primo impulso utile ricevuto (dopo quello iniziale) trasporta l'informazione relativa al bit0, cioè quello meno significativo (11101000); il secondo al successivo bit1, verso sinistra (11101000) ecc. In particolare, la sequenza di bit completa sarà ricostruita bit per bit nella variabile *codicePulsante*, grazie alla notazione .bit# (vista nel dettaglio a pag. 80, a proposito della fusione sensoriale). Per ottenere questo risultato, però, si è scelto un metodo particolare: il programma, infatti, procede memorizzando nei primi sette bit (da bit0 a bit6) della variabile *codicePulsante* il valore del decimo bit (bit9) delle corrispondenti sette variabili *inputIR* (da *inputIR0* a *inputIR6*). Vediamo perché. Avevamo anticipato la necessità di identificare una soglia che, confrontata con la durata di un impulso utile (comunque inferiore a 1000), permettesse di deciderne il valore (0 oppure 1): nel nostro caso, tale soglia è di circa 500 (sempre in riferimento all'unità di misura 2 µs, dunque pari a 1 ms). Se dunque la durata di un impulso utile risultasse minore di 1 ms (500*2µs), il valore da registrare nel bit corrispondente sarebbe 0; una durata superiore, invece, determinerebbe la memorizzazione di un 1. Per attuare tale indispensabile confronto tra la durata dell'impulso e la soglia, avremmo potuto scegliere una soluzione più intuitiva (ma molto complessa da realizzare), come quella di utilizzare una serie di istruzioni if... then che, confrontando il valore delle rispettive variabili *inputIR* con la soglia, settassero poi il corrispondente bit della variabile *codicePulsante*. Il metodo proposto nel listato, invece, benché meno intuitivo, è molto più compatto. Si tratta, in pratica, di decidere il valore (0 oppure 1) delle variabili *inputIR*, leggendone unicamente il bit9; tale valore, quindi, codificherà via via i bit (da 0 a 6) della variabile *codicePulsante*. Vediamo come. Riprendiamo il discorso della codifica dei numeri binari (riportato alle pagg. 77-79): ricorderai che, di una stringa di bit che codifica un numero decimale nel corrispettivo binario, il bit in posizione *n*-esima corrisponde all'*n*-esima potenza di 2. Il bit in posizione 0 (più a destra) corrisponde cioè a 2^0=1; il bit in posizione 1 corrisponde a 2^1=2 e così via. Dunque il bit in posizione 9 (quello che interessa il nostro caso) corrisponde a 2^9=512. Ed ecco spiegato il motivo della scelta: tale valore, infatti, è pressoché identico alla soglia (500) che ci interessa; inoltre il bit successivo (in decima posizione), avrebbe valore 2^{10}=1024, ossia appena superiore all'altro parametro (1000) valido per distinguere gli impulsi utili da quello iniziale di avvio. Data questa convergenza di valori, dunque, possiamo utilizzare il bit9 come indicatore dell'intero valore delle variabili inputIR: se il bit9 di una delle variabili è 0, il valore in essa memorizzato sarà infatti necessariamente inferiore a 512 (al limite è 511, ovvero 0111111111 in binario) e, dunque, potrà considerarsi inferiore alla soglia e pari a 0; viceversa con il bit9 pari a 1, il valore memorizzato sarà maggiore di 512 (al limite è proprio 512, 100000000 in binario), la variabile dunque sarebbe sopra alla soglia e varrebbe 1. Una volta ricostruiti tutti i bit di *codicePulsante* tramite i bit9 delle variabili inputIR, la subroutine termina con l'istruzione return. Sarà poi l'istruzione di debug a 'tradurre' il suo codice in decimale. Per riassumere, ricostruiamo a ritroso la 'storia' del numero 116, visualizzato alla pressione del tasto 7: di fatto esso corrisponde al numero binario 1110100. Ciascuna cifra, a partire dalla meno significativa, corrisponde al bit9 del valore memorizzato rispettivamente nelle variabili *inputIR* (da *inputIR0* a *inputIR6*). Tale valore, infine, non è che l'esito del rilevamento della durata dei sette impulsi utili, ossia degli intervalli in cui il segnale inviato dal telecomando permane allo stato basso (0). Testando analogamente il segnale di ogni tasto, potrai infine completare con i corrispettivi codici reali la tabella utile per la programmazione del robot (quella qui sotto riproduce la disposizione e le diciture originali dei tasti del telecomando).

20	58	21
63	37	56
0	1	2
3	4	5
6	7	8
29	9	
43	116	124
23	117	97
60	18	16
54	19	17