I gusci posteriori

I due gusci plastici del blocco posteriore, insieme agli allegati del prossimo fascicolo, ti permetteranno di completare il montaggio del manipolatore, per poi fissarlo al corpo del robot. Potrai quindi effettuare il corretto cablaggio dei cavi dei due motori a spazzola e programmare i movimenti della pinza.

Nella foto. **I due gusci, destro ❶ e sinistro ❷, del blocco posteriore del manipolatore.**

A SECONDA DELLA LUCE

Nelle prossime pagine, invece, preciseremo il discorso sulla programmazione del robot: in particolare, torneremo sull'uso dei fotoresistori e sulla calibrazione dei servomotori. Come abbiamo già avuto modo di vedere (alle pagine 101-104), grazie ai fotoresistori il robot può rilevare la presenza di luce. In particolare, sappiamo che l'informazione relativa alla luminosità si ricava dal valore della costante di tempo τ del circuito RC (formato cioè da un fotoResistore e un Condensatore) tramite l'utilizzo del comando **rctime**. Finora abbiamo sempre fatto riferimento a circuiti RC aventi condensatori da 10 nF. A robot spento, prova invece a sostituirli con quelli da 100 nF (cioè 0,1 µF). Mandando in esecuzione il programma di test (visto a pag. 108), noterai che i due valori delle costanti τ dei sensori (visualizzati nella finestra di debug) sono ora di un ordine di grandezza 10 volte maggiore. La costante di tempo τ, infatti, è direttamente proporzionale sia a R sia a C (infatti è uguale a R*C): assumendo per semplicità che R sia fissa (cioè che non ci siano brusche variazioni di luminosità nell'ambiente), aumentando C di 10 volte (da 10 a 100 nF), il valore di τ aumenta dello stesso fattore (10). In tal caso, però, il condensatore impiega più tempo a caricarsi (il transitorio è più lungo): a parità di illuminazione, quindi, se si usa la coppia di condensatori da 100 nF si otterranno valori di τ maggiori e, dunque, anche il tempo misurato da **rctime** sarà maggiore.

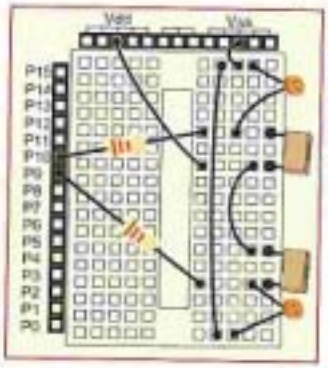
Nei sensori di luce, la scelta del condensatore dipende dalla prestazione richiesta al robot

Tale prestazione può essere più o meno vantaggiosa, a seconda delle circostanze e dei compiti assegnati al robot, e la scelta del condensatore più adatto dipende dal grado d'illuminazione dell'ambiente in cui agirà il robot. Una maggiore capacità dei condensatori, ad esempio, è svantaggiosa nel caso in cui sia richiesta un'elevata velocità di navigazione e di percezione delle variazioni dell'intensità luminosa. Più in generale, **all'aumentare della luminosità dell'ambiente è preferibile aumentare la capacità del condensatore; in ambienti più bui, invece, è meglio utilizzare condensatori con capacità minore.** Poniamo il caso di operare in un ambiente buio: il valore R del fotoresistore sarà molto grande (in base alle specifiche del costruttore, è dell'ordine dei mega-ohm), il che significa che anche il valore di τ è molto grande. Con un condensatore a bassa capacità (10 nF), dunque, sarà possibile ridurre il valore di τ e avere così dei tempi di risposta migliori. Al contrario, in ambienti molto illuminati il valore di R sarà relativamente piccolo (qualche decina di chilo-ohm), con il rischio di avere a che fare con valori di τ troppo piccoli e poco precisi. In tal caso, dunque, sarà preferibile una maggiore capacità (100 nF): aumenta così il valore di τ, mantenendo al contempo una buona sensibilità.

Le fasi di programmazione

Proviamo ora ad applicare quanto detto con un programma che faccia muovere il robot a seconda dei dati percepiti tramite i sensori fotosensibili.

L'intento è quello di costruire un programma che 'attiri' il robot nella direzione di maggior intensità di una fonte luminosa (ad esempio il fascio di luce di una torcia elettrica). Ovviamente, per far questo è necessario implementare sulla breadboard il doppio circuito RC (a destra) con i due fotoresistori collegati a P9 e P10 e, in serie, i due condensatori. Da quanto detto, la scelta dei condensatori dipende dalla luminosità dell'ambiente; in questo caso si è optato per quelli da 10 nF. Per quanto riguarda invece i fotoresistori, puoi scegliere se 'puntarli' verso l'alto (in modo da ricevere direttamente la luce), oppure verso il basso (così da ricevere la luce riflessa dal terreno). Queste due configurazioni non comportano alcuna variazione nel listato: si tratta infatti di una scelta puramente strategica, che dipende da come si vuole indirizzare il robot (se puntando il fascio di luce direttamente sui sensori, oppure sul pavimento). Volendo, puoi anche piegare leggermente ciascuno dei due fotoresistori verso l'esterno del robot. Per quanto riguarda poi la locomozione, utilizzeremo i servomotori che dovresti già aver montato (in caso contrario, puoi procedere anche con i motori a spazzola, con l'accortezza, però, di sostituire le parti di listato relative ai movimenti dei servomotori con gli equivalenti movimenti dei motori a spazzola).



Dopo aver digitato il listato (nella pagina accanto) nell'area di editing, collega il robot al computer attraverso il cavo seriale; quindi accendilo e premi il pulsante Run della barra degli strumenti per caricare il programma in memoria. Poi scollega il robot e posizionalo sul pavimento della stanza. Premendo il tasto start/stop della scheda madre, il programma verrà mandato in esecuzione. Analizziamo ora il listato nel dettaglio. Innanzitutto sono state dichiarate due variabili di tipo word, RCfoto_SX e RCfoto_DX; come già nel programma di test (visto a pag. 108), tali variabili sono destinate a contenere il valore della costante di tempo τ calcolato con rctime. Noterai che i nomi delle due variabili contengono il carattere (_), detto *underscore*: poiché, come sai, i nomi di variabili e costanti devono essere stringhe di caratteri uniche (senza spazi), tale carattere consente di avere lo stesso effetto visivo dello spazio, senza però incorrere in errori di sintassi. In pratica, al pari dell'alternanza tra maiuscole e minuscole, l'uso dell'underscore migliora la leggibilità del listato, soprattutto nel caso in cui presenti nomi 'complessi', che accostano più elementi (in questo caso, il rimando RC al circuito, la tipologia foto del sensore e il lato DX/SX della breadboard sul quale si trova). Dopo la dichiarazione delle due variabili, è stato inoltre dichiarato l'aliasing PBin che ti consentirà di usare il tasto start/stop della scheda madre. Segue poi la dichiarazione di cinque costanti (con), di cui due relative alle porte dei fotoresistori (FOTO_SX con 10 e FOTO_DX con 9) e due a quelle dei servomotori (SERVO_SX con 12 e SERVO_DX con 13); il significato della quinta costante (SOGLIA), infine, sarà chiarito tra poco. Nella fase di inizializzazione, attraverso il comando low, le porte relative ai due servomotori vengono settate al valore logico basso: ricorderai infatti che l'istruzione low (analogamente all'istruzione high) setta la porta dapprima come output e, quindi, le assegna il corrispondente valore logico (basso per low e alto per high). Come abbiamo già visto in altre occasioni, prima del programma principale (main), è stata introdotta la verifica (Aspetta: if PBin = 1 then Aspetta) dello stato (1; 0) del tasto start/stop: fino a quando questo non verrà premuto (0), il programma non andrà in esecuzione. Per quanto riguarda invece il programma principale, esso è costituito da un ciclo infinito (main... goto main) che può essere suddiviso in tre fasi. Inizialmente, tramite il comando rctime, viene calcolato il valore della costante di tempo di ognuno dei due circuiti RC, memorizzato poi nelle due variabili RCfoto_SX e RCfoto_DX. La seconda fase riguarda invece il confronto tra le due costanti di tempo. Poiché vogliamo che il robot si orienti verso la fonte luminosa, occorre valutare la diversa luminosità percepita dai due fotoresistori e, quindi, stabilire il verso di rotazione del robot: si tratta quindi di confrontare i valori delle costanti di tempo. In particolare, delle due costanti di tempo τ, quella minore indica che il corrispondente sensore ha ricevuto una maggiore quantità di luce (a parità di C, il valore di R è minore). Prima di procedere al vero e proprio confronto, tuttavia, occorre valutare l'entità della differenza tra i due valori calcolati: difficilmente, infatti, i due fotoresistori avranno caratteristiche fisiche ed elettriche identiche; sottoposti alla stessa intensità luminosa, perciò, molto probabilmente tenderanno ad avere valori di resistenza leggermente diversi. Inoltre la stima del valore di τ, eseguita tramite rctime, può introdurre a sua volta un piccolo errore: anche se l'intensità luminosa è uguale e uniforme per entrambi i sensori, i valori calcolati per le due costanti di tempo potranno essere leggermente diversi. Valutato dunque lo scarto tra i due valori assoluti (vale a dire il risultato della sottrazione tra i due moduli, privi di segno), è opportuno stabilire una soglia oltre la quale tale differenza sia da considerarsi rilevante. La costante SOGLIA dichiarata all'inizio serve esattamente a questo scopo: indica cioè il valore massimo (5) che la differenza dei due valori può assumere senza essere considerata. Più tale valore è alto, meno sensibile alle variazioni di luminosità sarà il robot; e viceversa. In termini di programmazione, nel listato questa serie di operazioni si traduce in tre salti condizionati if...then.

```
' {$stamp BS2}

'---------- Variabili ----------

RCfoto_SX   var   word
RCfoto_DX   var   word

PBin        var   in2

'---------- Costanti ----------

SOGLIA      con   5

FOTO_SX     con   10    ' Costante relativa alla porta del fotoresistore sinistro
FOTO_DX     con   9     ' Costante relativa alla porta del fotoresistore destro

SERVO_SX    con   12    ' Costante relativa alla porta del servo di sinistra
SERVO_DX    con   13    ' Costante relativa alla porta del servo di destra

'---------- inizializzazione ----------

' imposta contrarbe le porte servo come output e a valore logico basso
low SERVO_DX
low SERVO_SX

'---------- check tasto di reset ----------

Aspetta:
      if PBin = 1 then Aspetta

'---------- Programma principale ----------

main:


' Misura la costante di tempo relativa al fotoresistore di destra

      high FOTO_DX
      pause 3
      rctime FOTO_DX, 1, RCfoto_DX

' Misura la costante di tempo relativa al fotoresistore di sinistra

      high FOTO_SX
      pause 3
      rctime FOTO_SX, 1, RCfoto_SX

' Controlla la differenza tra le due costanti di tempo; se il valore assoluto (abs)
' della differenza è maggiore della soglia procede al confronto

      if abs(RCfoto_SX - RCfoto_DX) < SOGLIA then main
      if RCfoto_SX > RCfoto_DX then deviaDestra
      if RCfoto_SX < RCfoto_DX then deviaSinistra

deviaDestra:
      ' esegue una rotazione sul posto verso destra
      pulsout SERVO_DX, 1000
      pulsout SERVO_SX, 1000
      pause 20
goto main


deviaSinistra:
      ' esegue una rotazione sul posto verso sinistra e
      pulsout SERVO_DX, 500
      pulsout SERVO_SX, 500
      pause 20
goto main
```

Il primo, tramite l'operatore matematico abs(...) (contrazione di *absolute*, cioè "assoluto"), calcola la differenza tra i valori assoluti delle costanti rapportandola poi al valore di soglia (tramite l'operatore <). In pratica, calcola la differenza senza segno, (ad esempio abs(5 – 6) = 1, non -1): finché tale differenza è minore del valore di SOGLIA, non si effettuano altre operazioni e si riprende saltando all'etichetta main. In caso contrario, si procede invece al confronto tra le due costanti di tempo (if RCfoto_SX > o < RCfoto_DX...) eseguendo le successive istruzioni di salto condizionato (then deviaDestra o deviaSinistra). Se la variabile RCfoto_SX è maggiore (>) di RCfoto_DX, allora il sensore sinistro riceve meno luce; per inseguire la fonte luminosa, dunque, il robot dovrà ruotare verso destra, come indicato dalle istruzioni contenute nella rispettiva subroutine (deviaDestra). Nel caso opposto, invece, il robot dovrà girare a sinistra: si salta perciò alla subroutine deviaSinistra. La terza e ultima fase del main riguarda infatti le operazioni di movimento. Nel nostro caso bastano due rotazioni sul posto: a sinistra e a destra. Consideriamo la rotazione verso destra (deviaDestra: ... goto main), che si ottiene facendo ruotare i due servomotori in senso antiorario. In termini di programmazione, si sa, questo significa impartire alle rispettive porte il comando pulsout 1000. Viceversa, per ottenere la rotazione a sinistra (deviaSinistra: ... goto main) è necessario far ruotare i servomotori in senso orario, con un impulso di 500*2 µs (cioè pulsout 500).

Come ricorderai, nel fascicolo 32 avevamo parlato di calibrazione dei servomotori. Torniamo ora sull'argomento, per creare delle routine di moto ottimizzate in base alla calibrazione reale.

Digita il listato di pag. 124 (a destra) che richiama, in sequenza, una serie di subroutine di moto tali per cui, con i motori calibrati, il robot dovrebbe avanzare, ruotare in senso orario, antiorario e poi tornare indietro, esattamente nello stesso punto da cui è partito. Tra ogni routine (o subroutine: i due termini sono quasi sinonimi, giacché una subroutine è una routine rapportata a un'altra che la chiama; tutte le subroutine sono dunque routine, ma non viceversa) di movimento ne intercorre una che blocca il robot, per evidenziare la separazione tra le azioni. Il listato prevede, anzitutto, la dichiarazione di alcune costanti: le prime (MotoreSx con 12 e MotoreDx con 13) identificano le porte cui sono collegati i motori; le successive (StopSx con 750; OrarioSx con 500 e AntiorarioSx con 1000) riguardano i valori di stop e di massima rotazione (nei due sensi) per il motore sinistro. Seguono poi analoghe costanti per i valori di stop e massima rotazione del motore destro; poi, la costante relativa al tempo che intercorre tra due impulsi successivi (tempoBasso con 20). I valori di stop e di massima rotazione dichiarati per i due motori sono quelli standard che, dunque, non tengono conto della calibrazione (se i motori fossero identici, non sarebbe necessario calibrarli, né distinguerli: basterebbe dichiarare tre costanti valide per entrambi, una di stop e due per le massime rotazioni). Prova ora a mandare in esecuzione il programma: nel migliore dei casi, vedrai il robot comportarsi come predetto; probabilmente, però, non sarà così. Potrebbe infatti accadere che il robot, invece di stare fermo, ruoti leggermente a piccoli scatti; oppure, in fase di avanzamento o indietreggiamento, potrebbe deviare in una direzione. Per quanto riguarda il primo caso, come visto a pag. 116, devi procedere alla calibrazione del robot rispetto alla media dei valori reali di arresto, più o meno distanti dal centro di simmetria dato come valore di default (750). Ricavati tali valori, dovrai sostituirli a 750 nella dichiarazione delle costanti.

LE FASI DI PROGRAMMAZIONE

Consideriamo ora il secondo caso, per cui il robot, invece che procedere dritto, devia di lato. Analizziamo il listato in basso, una semplificazione del precedente, in cui compare solo la routine di avanzamento, inclusa in un ciclo infinito (loop: ... goto loop). Puoi notare che il corpo del ciclo è analogo alla routine avanti del programma precedente. Vediamo ora un modo per calibrare i motori sull'avanzamento del robot. Mandando in esecuzione il programma, se il robot non è calibrato, virerà in una direzione. Se gira a destra, significa che il motore sinistro gira un po' più velocemente: in tal caso, visto che non si può aumentare la velocità massima del motore destro (pulsout deve essere compreso tra 500 e 1000), si dovrà rallentare il sinistro, diminuendo il valore della costante AntiorarioSx. Si procede, dunque, diminuendo di poche unità alla volta, fino a che il robot procede in linea retta (e, dunque, i due motori girano alla stessa velocità). Se invece il robot girasse a sinistra, sarebbe il motore destro a girare più velocemente; stavolta, quindi, dovrai incrementare di qualche unità il valore di OrarioDx fino a ottenere un moto perfettamente rettilineo. Ottenuti questi valori reali, dovrai sostituirli a quelli di default presenti in entrambi i listati.
Se poi procedi alla calibrazione per il movimento di indietreggiamento (intervenendo allo stesso modo sulla routine indietro, con l'accortezza però di dichiarare, al posto delle variabili AntiorarioSx e OrarioDx, quelle OrarioSx e AntiorarioDx), disporrai di tutti i parametri reali necessari per determinare l'effettiva durata degli impulsi di comando dei tuoi servomotori. Memorizza dunque i valori che hai ottenuto in fase di calibrazione, sia per i programmi che vedremo in seguito sia per quelli che creerai tu stesso. D'ora in poi, infatti, ogni volta che incontrerai i valori di default (cioè 500, 750 e 1000) in un programma per gestire i servomotori, dovrai sostituirli con quelli reali, da te ottenuti grazie alla calibrazione. Ora, per ottenere routine di moto personalizzate, non ti resta che fare un ultimo passo.
Può infatti darsi che, in seguito alla calibrazione, il robot non percorra la stessa distanza applicando prima la subroutine avanti e poi quella indietro; allo stesso modo, le rotazioni in senso orario e antiorario, associate alle subroutine destra e sinistra, potrebbero coprire un angolo diverso. Per ovviare a questi inconvenienti, si può intervenire sul comando for..., In tutte le routine, infatti, ricorre lo stesso ciclo for (for contatore=0 to 100).

prog2.bs2

```
'{$STAMP BS2}

MotoreSx     con    12
MotoreDx     con    13

AntiorarioSx       con    1000

OrarioDx     con    500

tempoBasso   con    20

contatore    var    word
PBin   var   in2                ' Pulsante Restart

'inizializzo i motori

low MotoreSx
low MotoreDx

'---------Start: controlla pressione tasto avvio----------

start:
      if PBin = 1 then start


loop:
      pulsout MotoreSx,AntiorarioSx
      pulsout MotoreDx,OrarioDx
      pause tempoBasso
      goto loop
```

prog1.bs2

```
'{$STAMP BS2}

MotoreSx     con    12
MotoreDx     con    13

StopSx       con    750
OrarioSx     con    500
AntiorarioSx       con    1000

StopDx       con    750
OrarioDx     con    500
AntiorarioDx       con    1000

tempoBasso   con    20

contatore    var    word
PBin   var   in2                ' Pulsante Restart

'inizializzo i motori

low MotoreSx
low MotoreDx

'---------Start: controlla pressione tasto avvio----------

start:
      if PBin = 1 then start
main:
      gosub avanti
      gosub fermo
      gosub destra
      gosub fermo
      gosub sinistra
      gosub fermo
      gosub indietro
      gosub fermo
      goto start

avanti:

      for contatore=0 to 100
      pulsout MotoreSx,AntiorarioSx
      pulsout MotoreDx,OrarioDx
      pause tempoBasso
      next
      return


indietro:

      for contatore=0 to 100
      pulsout MotoreDx,AntiorarioDx
      pulsout MotoreSx,OrarioSx
      pause tempoBasso
      next
      return

destra:

      for contatore=0 to 100
      pulsout MotoreSx,AntiorarioSx
      pulsout MotoreDx,AntiorarioDx
      pause tempoBasso
      next
      return

sinistra:

      for contatore=0 to 100
      pulsout MotoreSx,OrarioSx
      pulsout MotoreDx,OrarioDx
      pause tempoBasso
      next
      return

fermo:
      for contatore=0 to 100
      pulsout MotoreSx,StopSx
      pulsout MotoreDx,StopDx
      pause tempoBasso
      next
      return
```

Variando a piacere il valore finale (100), puoi differenziare le varie subroutine, in modo da fare compiere al robot movimenti ad hoc. Per quanto riguarda invece la velocità dei movimenti, bisogna cambiare il valore della costante tempoBasso (ora uguale per tutti i movimenti): nulla vieta, infatti, di avere pause diverse per ogni movimento, in modo tale da ottenere movimenti eseguiti a velocità diverse.