Vedere l'obiettivo per raggiungerlo

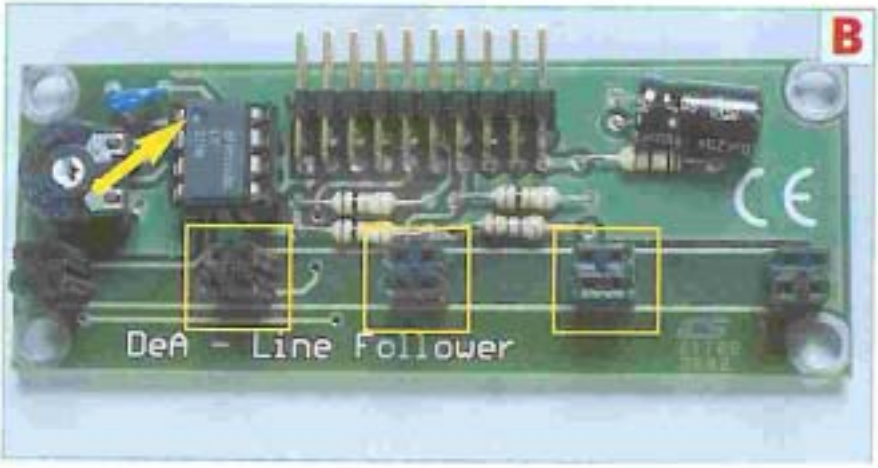
Gli allegati a questo fascicolo completano il cosiddetto 'inseguitore di linea', già descritto nei fascicoli precedenti. Prima ancora di metter mano ai componenti del dispositivo, tuttavia, è necessario scollegare i motori a spazzola dalla scheda di controllo motori, rimuovendo le coppie di cavetti dei due motori dai corrispondenti morsetti della scheda. Come spiegato a pagina 171, infatti, non si possono usare il DeA Line Follower e i motori a spazzola contemporaneamente.

Le fasi di montaggio

Per eseguire le operazioni di montaggio illustrate di seguito, oltre ai pezzi allegati a questo fascicolo, è necessario recuperare il comparatore LM 311N allegato al fascicolo 45 (il chip raffigurato nella foto sotto, a sinistra), i tre sensori monoblocco a raggi infrarossi forniti nei fascicoli 46, 47 e 48 (uno dei quali è rappresentato sotto, nella foto a destra) e la scheda DeA Line Follower, allegata al fascicolo 49. Ti serviranno inoltre una vite da 16 mm e un dado M3 (allegati al fascicolo 23).

L'ELENCO DEI PEZZI

1. staffa di supporto della scheda DeA Line Follower
2. slitta di supporto della scheda DeA Line Follower
3. cavo piatto
4. connettore maschio a 20 pin
5. n. 2 viti a stella da 4 mm
6. n. 2 viti a stella da 25 mm
8. n. 2 dadi M3



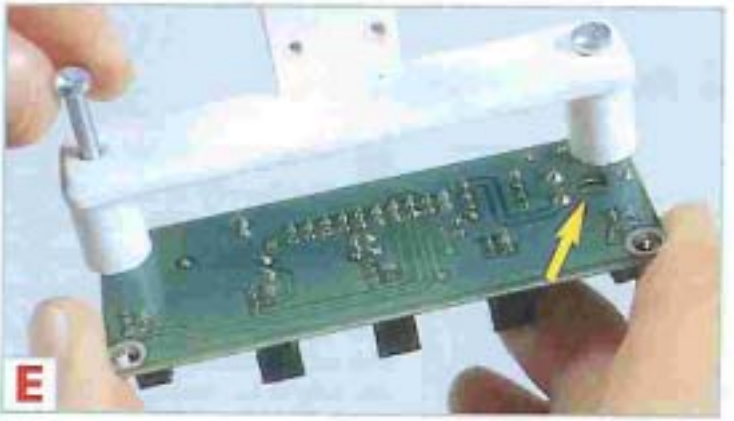
A Prima di tutto rimuovi la coppia anteriore di baffi (che, data la posizione e l'uso di alcune porte del microcontrollore, intralcia l'impiego del DeA Line Follower) e la vite (corchiata) che fissa il paraurti al telaio.

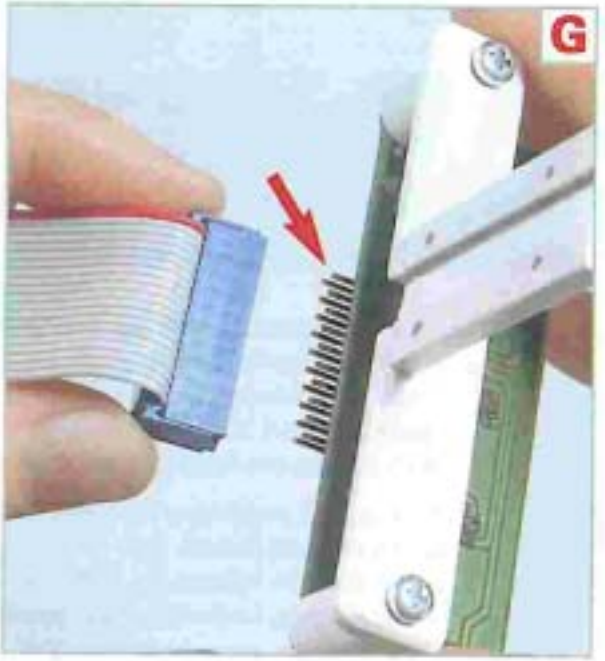
B Alloggia il comparatore LM 311N nella sede predisposta (indicata dalla freccia) sulla scheda DeA Line Follower: per orientarlo in modo corretto, fai riferimento alla posizione della tacca circolare presente su di esso (indicata dalla stessa freccia). Ora individua sulla scheda le tre sedi (riquadrate) per l'alloggiamento dei sensori monoblocco

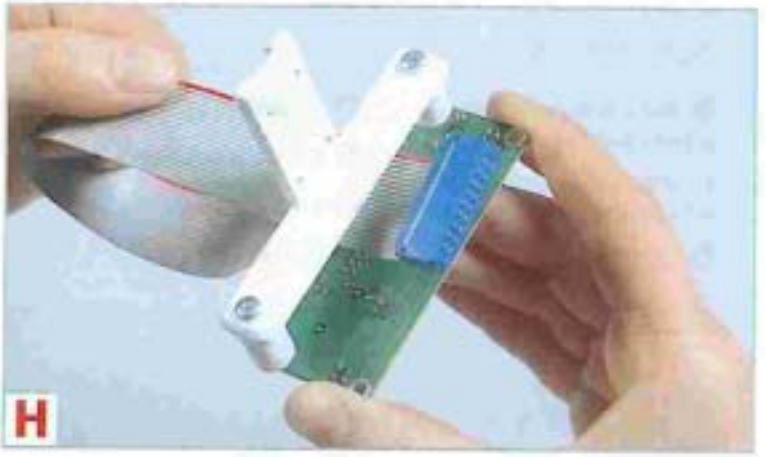
C Con un comune paio di forbici taglia i quattro contatti di ciascuno dei tre sensori monoblocco, in modo che la lunghezza dei contatti risulti di 5 mm.

D È il momento di alloggiare i sensori nelle sedi individuate nella foto **B**, ma fai attenzione al loro orientamento: i LED emettitori (riquadrati in giallo) devono essere rivolti verso la dicitura DeA - Line Follower.

E•F Capovolgi la scheda e posiziona due dei suoi fori in corrispondenza di quelli per le viti presenti sulla staffa di supporto: per orientare correttamente i due pezzi l'uno rispetto all'altro, fai riferimento all'apertura circolare (indicata dalla freccia) posta sulla scheda in corrispondenza del trimmer. Fissa la scheda e la staffa l'una all'altra avvitando le due viti da 25 mm ai dadi M3.



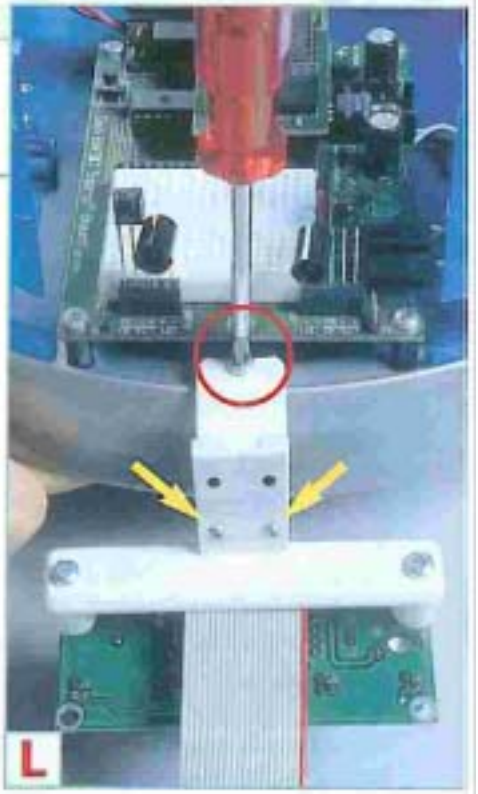




G Collega una delle due prese femmine del cavo piatto al connettore a 20 pin (indicato dalla freccia) presente sulla scheda. L'orientamento della presa rispetto al connettore non è significativo.

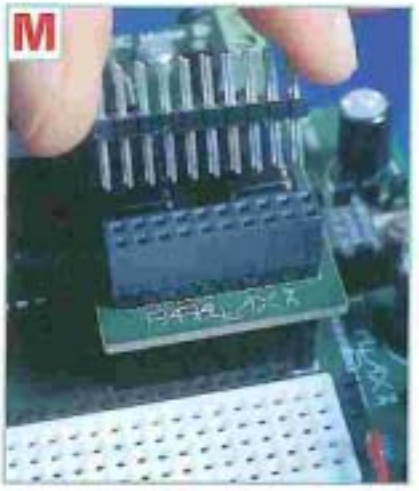
H Senza attorcigliare su se stesso il cavo piatto, fanne passare la presa rimasta libera nello spazio vuoto tra la scheda e la staffa.

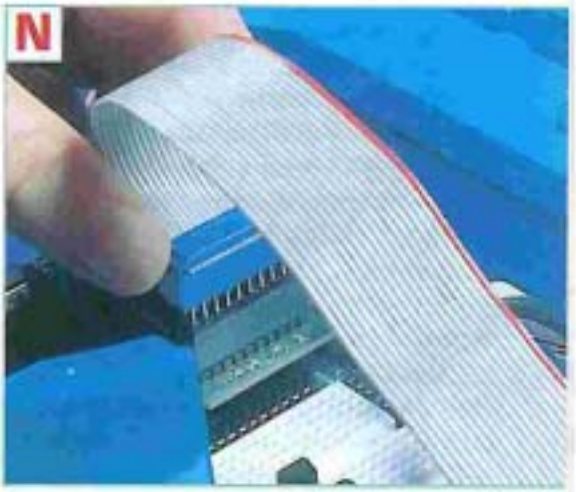
I Inserisci la slitta di supporto nella guida predisposta sulla staffa. Nota che la staffa presenta due coppie di fori per le viti che la fissano alla slitta: ciò permette di regolare la reciproca posizione dei due pezzi e, quindi, la distanza dei sensori monoblocco dalla superficie di appoggio del robot (la distanza ottimale è di circa 3 mm). **L** Inserita la slitta, in tutta la sua lunghezza, nella guida della staffa, fissa l'uno all'altro i due pezzi, avvitando le due viti da 4 mm nella coppia inferiore di fori (indicati dalle frecce) presenti sulla staffa. Posiziona il pezzo così ottenuto sul paraurti del robot e fissalo, avvitando una vite da 16 mm a un dado M3 nella sede (cerchiata in rosso) da cui avevi inizialmente rimosso la vite di fissaggio del paraurti al telaio.



M Inserisci il connettore maschio a 20 pin nel bus di espansione X1 della scheda di controllo motori (il lato di inserimento non è significativo).

N Facendo di nuovo attenzione a non attorcigliare su se stesso il cavo piatto, posiziona la presa femmina, posta alla sua estremità, sul connettore maschio che hai appena montato sulla scheda di controllo motori. Esercita una leggera pressione per completare l'inserimento dei pezzi l'uno nell'altro.





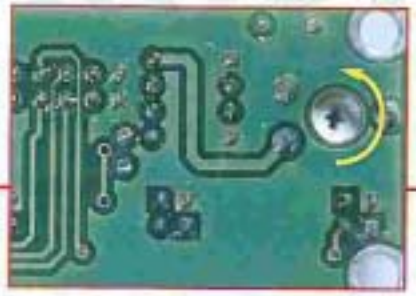
O Ecco il risultato delle operazioni di montaggio appena eseguite: la scheda DoA Line Follower, che hai fissato al telaio mediante la staffa e la slitta di supporto, risulta collegata alla scheda di controllo motori dal cavo piatto.

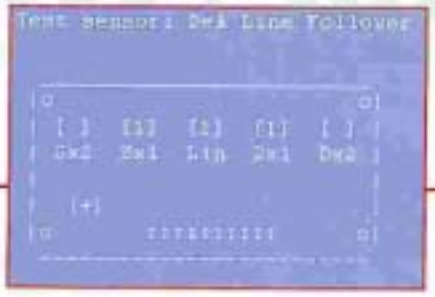
Le fasi di programmazione

Dopo aver montato il DeA Line Follower e averne visto gli aspetti hardware (alle pagg. 169-171), occupiamoci di quelli software, legati alla programmazione del tuo robot per la navigazione automatica (per cui trovi fin da ora un programma d'esempio disponibile sul sito ***www.robot.deagostini.it****) e alla preventiva calibrazione dei sensori monoblocco (in queste pagine). A proposito dei sensori, occorre ricordare anzitutto che, sebbene in un solo blocco, sono tuttavia costituiti da due componenti diversi: un emettitore (LED) e un ricevitore (transistor) di raggi infrarossi (IR). Inoltre, la loro prestazione dipende dal materiale della superficie riflettente: come vedremo, è proprio in base alle caratteristiche di quest'ultima che dovrai calibrare i sensori, intervenendo sul trimmer (nella foto, cerchiato) del Line Follower (per brevità, da ora DeA LF) e, dunque, sul valore di soglia del suo circuito comparatore.*

Per calibrare i sensori useremo un programma (a destra) che offre un riscontro immediato delle operazioni eseguite sulla scheda, visualizzandone i risultati nella finestra di debug (per questo, digitato e mandato in esecuzione il programma, è necessario mantenere il robot collegato al PC tramite il cavo seriale). Prima di tutto, però, dovrai aver completato il montaggio e il collegamento del DeA LF, come illustrato nelle pagine precedenti; dal canto suo, il programma può essere utilizzato con qualsiasi configurazione di sensori ma, nel nostro caso, utilizzeremo quella che sfrutta le tre sedi centrali (nella foto: Sx1, Line e Dx1). Per eseguire il test, inoltre, è necessario tracciare una linea, un percorso (per ora, basterà una linea disegnata su un foglio di carta, da posizionare sotto il DeA LF), con l'accortezza che la larghezza della linea sia uguale o, meglio, maggiore della dimensione di un sensore. Ovviamente, poiché il DeA LF è in grado di distinguere i contrasti, potrai scegliere se tracciare una linea bianca su sfondo scuro o una linea scura su sfondo bianco. Quest'ultimo è il caso proposto nel listato ma, come vedremo, una semplice modifica nel codice ti permetterà di usare lo stesso programma con un percorso tracciato in chiaro su uno sfondo nero. Come si diceva, nella finestra di debug verranno visualizzate le informazioni rilevate dai sensori. In particolare e fin da subito, verrà visualizzata una rappresentazione della scheda vista dall'alto ('generata' dalla lunga serie di istruzioni di debug del listato).
Le effettive misure ottenute dai sensori, invece, verranno visualizzate in seguito, sotto forma di 1 o 0 (a seconda che la linea tracciata sia presente o meno nel raggio d'azione del sensore), nelle posizioni via via corrispondenti ([ ]): ad esempio, un 1 posto in seconda posizione indicherà la presenza della linea sotto il sensore Sx1. Muovendo il foglio e, quindi, la linea da un sensore all'altro, puoi verificare la corrispondente variazione di lettura nella finestra di debug. Se invece non osservi alcun cambiamento, oppure la stringa di valori visualizzati si compone di tutti e solo 1, occorrerà procedere alla calibrazione. Con un cacciavite, porta al minimo il valore di resistenza del trimmer girando la vite in senso antiorario (con la convenzione di guardare il robot dall'alto e, dunque, con la scheda rivolta verso il basso, come illustrato nella foto della pagina accanto); quindi posiziona il foglio in modo che la linea sia posta in corrispondenza del sensore centrale (Line). Avendo selezionato il valore minimo di resistenza, la finestra di debug visualizzerà la stringa di tre 1 (in alto al centro, nella pagina accanto): portato al valore di tensione di massa l'ingresso indicato con il meno (-) del chip LM 311N, infatti, qualsiasi lettura dei sensori è superiore al valore di soglia impostato; in uscita dal circuito, perciò, sarà riportato il valore alto (1).


```
'(48STAMP BS2)

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

LED_ON          con     0               ' Valore logico accensione/spegnimento
LED_OFF         con     1               ' LED IR

LINEA_BIANCA    con     0               ' Linea bianca su fondo nero
LINEA_NERA      con     1               ' Linea nera su fondo bianco

MODALITA        con     LINEA_NERA      ' Tipo di percorso scelto

Sx1             con     6
LINE            con     4
Dx1             con     3

SpostatiIn      con     7

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

ledPos          var     nib
count           var     nib

ifBits          var     nib

' ---------- Inizializzazione ----------

outl = %01011000
dirl = %01011000

debug "Test sensori DeA Line Follower", CR
debug CR
debug " ,-------------------------. ", CR
debug " |o                       o| ", CR
debug " | [ ]   [ ]   [ ]   [ ]   | ", CR
debug " |  Sx2   Sx1   Line  Dx1  | ", CR
debug " |                         | ", CR
debug " |   (+)                   | ", CR
debug " |o                       o| ", CR
debug " `-------------------------' ", CR


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

Main:

      gosub Lettura_Sensori
goto Main
end

' ---------- Subroutine ----------

Lettura_Sensori:
      ifBits = 0
      for count = 0 TO 2
            lookup count, [Dx1, LINE, Sx1], ledPos
            outl.lowbit(ledPos) = LED_ON
            pause 1
            ifBits.lowbit(count) = in9 ^ MODALITA
            outl.lowbit(ledPos) = LED_OFF
            debug SpostatiIn, (9+((2-count)*5)), 4, DEC1 ifBits.lowbit(count)
      next
return
```





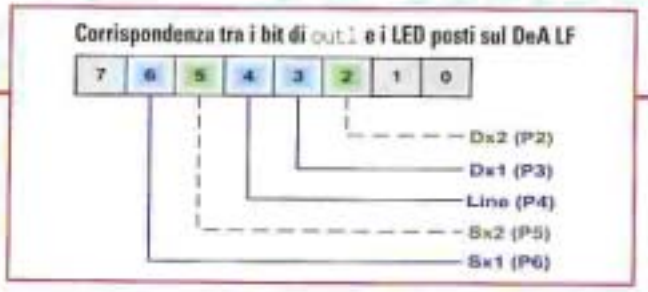
Sempre con il cacciavite, gira ora lentamente la vite del trimmer in senso opposto (orario) fino a quando nella finestra di debug compare un solo 1 in posizione centrale (in alto a destra): con questa operazione il DeA LF è calibrato e il robot potrà rilevare correttamente la linea. Per sicurezza, puoi ripetere l'esperimento con gli altri sensori e verificare la prestazione con lo stesso valore di soglia del trimmer. Vediamo ora il listato del programma, tenendo a mente le porte utilizzate dal DeA LF (riassunte nella tabella qui sotto). Per quanto riguarda le costanti dichiarate: LED_ON e LED_OFF contengono il valore per accendere (0) e spegnere (1) i LED dei sensori; mentre il valore di LINEA_BIANCA e LINEA_NERA identifica la modalità di test: l'una si riferisce al caso in cui si abbia un tracciato bianco su sfondo scuro, viceversa l'altra.

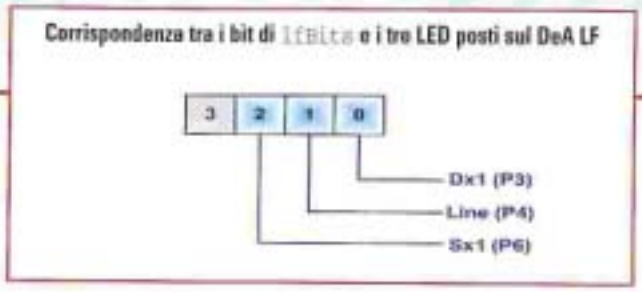
Porta	Tipo funzionamento	Componente DeA LF	Esito con valore 0	Esito con valore 1
P2	output	Sensore destro esterno Dx2	Led Acceso	Led Spento
P3	output	Sensore destro interno Dx1	Led Acceso	Led Spento
P4	output	Sensore centrale (Line)	Led Acceso	Led Spento
P5	output	Sensore sinistro esterno Sx2	Led Acceso	Led Spento
P6	output	Sensore sinistro interno Sx1	Led Acceso	Led Spento
P9	input	Uscita circuito comparatore (IR_Detect)	Rileva zona scura	Rileva zona chiara

Segue un nuovo tipo di dichiarazione di costante, MODALITÀ, che non si riferisce a un valore, bensì a un'altra costante; tale dichiarazione fa sì che la costante dichiarata assuma il valore di quella cui rimanda: MODALITÀ dunque avrà il valore di LINEA_NERA (1). Come avrai intuito, è proprio qui che dovrai intervenire (sostituendo LINEA_NERA con LINEA_BIANCA) se e quando vorrai testare il robot su un percorso bianco tracciato su sfondo nero. Procedendo nella dichiarazione, le costanti Sx1, LINE e Dx1 fanno riferimento alle porte (6, 4 e 3) relative ai corrispettivi sensori, mentre il valore (2) della costante Spostatile verrà utilizzato come codice speciale in una successiva istruzione di debug. Sul fronte delle variabili, invece, ne sono state dichiarate tre, tutte di tipo nibble: *ledPos* e *cont*, il cui significato sarà chiarito poi; e la variabile *IRBits* che conterrà i valori misurati dai sensori. Il programma presenta una prima fase di inizializzazione cui compete anche la visualizzazione nella finestra di debug dello schema (grazie alla serie di istruzioni debug). Il programma principale prevede solo un ciclo infinito (Main... goto Main) contenente la chiamata alla subroutine (Lettura_Sensori) che si occuperà della gestione dei sensori, della lettura delle misurazioni e della visualizzazione dei valori forniti dai sensori. Prima di scendere nel dettaglio del listato, vale la pena riprendere e ampliare alcuni concetti riguardanti la gestione dei registri INS, OUTS e DIRS della EEPROM (di cui si è detto alle pagg. 63 e 94-95).

Registro	Word	Byte	Nibble	Bit
INS	ins	inl - inh	ina, inb, inc, ind	in0, in1...in15
OUTS	outs	outl - outh	outa, outb, outc, outd	out0, out1...out15
DIRS	dirs	dirl - dirh	dira, dirb, dirc, dird	dir0, dir1...dir15

Ricorderai che i registri sono composti da 16 bit ciascuno: il registro INS contiene le informazioni in ingresso alle varie porte logiche del microcontrollore, cui si può accedere con le rispettive variabili speciali (in0, in1... in15). Il registro OUTS, invece, consente di settare il valore logico delle porte logiche nella modalità output (in questo caso le variabili speciali sono out0, out1... out15). Il registro DIRS, infine, determina la modalità di effettivo utilizzo delle porte: come output (valore 1) o come input (0). Facciamo ora un passo avanti: oltre alle variabili speciali individuali (come le precedenti) che consentono di accedere alla singola porta, esistono anche variabili speciali di gruppo che, invece, fanno riferimento a gruppi di più bit consecutivi di un registro, raggruppati secondo dimensioni di word (16 bit), byte (8), nibble (4) e bit (1, nel qual caso, in realtà, basta una variabile speciale individuale). Considerando la tabella qui sotto, vediamo come esempio il registro OUTS e le possibili variabili speciali di gruppo per accedervi (il discorso vale però anche per gli altri registri): con *outs* (di dimensione word) è possibile accedere per intero al registro di 16 bit; le variabili *outl* e *outh*, invece, di dimensione byte, fanno riferimento rispettivamente, l'una agli 8 bit (da 0 a 7) meno significativi del registro (*outl* sta per *low*) e l'altra ai restanti 8 bit (da 8 a 15), cioè i più significativi, (*outh* sta per *high*). Le variabili nibble (*outa*, *outb*, *outc* e *outd*), infine, fanno riferimento a gruppi di 4 bit, sempre a partire dai meno significativi (*outa*, da 0 a 3; *outb*, da 4 a 7 ecc.). Come ricorderai, avevamo già avuto modo di parlare della possibilità di accedere a ogni singolo bit di una variabile, grazie al costrutto *variabile*.bit# (spiegato a pag. 80). Vediamo ora un'altra possibilità, offerta da un nuovo costrutto (presente nel listato), ossia *variabile*.lowbit(*indice*). Se *indice* è un numero compreso tra 0 e *n*-1 (ove *n* è la dimensione in bit della variabile), l'espressione risultante (*variabile*.lowbit(#)) ha di fatto lo stesso significato di *variabile*.bit#; se invece *indice* è una variabile o una costante, e qui sta il vantaggio di questa nuova espressione, l'accesso al singolo bit di *variabile* non sarà prefissato, bensì regolato da un parametro variabile (ossia proprio il contenuto di *indice*): in informatica, infatti, tale tipo di accesso si definisce 'parametrico'. Tornando al listato, entriamo ora nel dettaglio della fase di inizializzazione. Con la prima istruzione (*outl* = %01011000, per la cui grafia vedi a pag. 79), le porte relative ai sensori (che in totale sono cinque, ma di cui noi ne stiamo usando solo tre, ossia P3, P4 e P6) vengono settate in modalità output e i LED dei sensori spenti.





Questo avviene perché la stringa binaria 01011000 di *outl*, corrisponde bit per bit alle porte (da P7 a P0) del microcontrollore (come mostrato qui sopra): il valore logico alto (1), in corrispondenza delle porte 3, 4 e 6, setta come spenti i LED (che, in quanto emettitori, sono l'output del sensore). La modalità output, invece, viene settata in seguito ma in modo analogo con *dirl* = %01011000 (in cui gli 1 ricalcano le stesse posizioni viste prima). L'ordine di queste due operazioni (*outl*... e *dirl*...) è importante: settando prima la direzione (*dirl*...), infatti, di default si avrebbero tutte le porte (in output) settate a 0 e, di conseguenza, tutti i LED accesi contemporaneamente, almeno fino alla loro successiva inizializzazione (*outl*...), il che, come detto a pag. 171, non è auspicabile. Proseguendo nel listato, ecco la lunga serie di istruzioni di debug che 'disegnano' (una volta per tutte, giacché precedono il Main) lo schema della scheda nella finestra di debug; come vedremo fra poco, invece, le informazioni relative ai sensori verranno sovrascritte di volta in volta.
Il Main non fa altro che chiamare la subroutine Lettura_Sensori che deve, nell'ordine: accendere un singolo LED, leggere e memorizzare il valore in uscita dal circuito comparatore su P9, spegnere il LED, visualizzare nella finestra di debug il valore letto (nella posizione corrispondente dello schema) e, infine, ripetere l'operazione con gli altri sensori, sempre e solo uno alla volta. I valori letti su P9, in particolare, vengono memorizzati nei 3 bit meno significativi della variabile *lfBits*, mantenendo l'ordine delle rilevazioni e delle posizioni dei sensori (come mostrato nella figura in alto a destra).
Vediamo ora come tutto ciò venga effettivamente realizzato.
Per prima cosa, imposto il valore 0 alla variabile *lfBits* (vale a dire a tutti e otto i suoi bit), si cancellano eventuali precedenti misure.
Il successivo ciclo for, utilizzando la variabile *cont* come contatore da 0 a 2, esegue le tre iterazioni necessarie per operare sui tre sensori; in seguito, la stessa variabile *cont* verrà utilizzata per indicare la posizione del bit di *lfBits* in cui memorizzare il valore letto dal sensore in questione. La variabile *ledPos*, invece, serve per accedere alla posizione del bit di *outl* che accende il corrispondente LED. Attraverso l'istruzione lookup (la cui sintassi è a pag. 147), infatti, il valore di *cont* viene rimappato su *ledPos*, cosicché quest'ultima variabile contenga di volta in volta il valore della porta (3, 4 o 6) corrispondente alla posizione indicata da *cont* (0, 1 o 2): valore necessario per intervenire sul corrispondente bit (singolo) di *outl*. Quindi, per accendere il LED corrispondente, si setta tale bit di *outl* allo stesso valore di LED_ON (0), secondo il costrutto visto prima (*outl*.lowbit(*ledPos*)). Segue poi una pausa di 1 ms che permette al ricevitore IR del sensore in questione (di fatto un transistor) di leggere in modo corretto la precedente emissione del LED.

Con l'istruzione *lfBits*.lowbit(*cont*) = *in9* ^ MODALITÀ si legge il valore in entrata su P9 e lo si memorizza in *lfBits* (sempre con lowbit, che usa *cont* come indice). Di fatto, se si rileva una zona scura, il valore restituito da P9 (ottenuto con *in9*) è basso; in caso contrario è alto: tale informazione, però, non è sufficiente, giacché in questo caso noi vogliamo che venga memorizzato un 1 quando il sensore percepisce la presenza della linea (che è scura) e uno 0 altrimenti, il che significa integrare l'informazione di *in9* con quella relativa al tipo di percorso scelto (cioè MODALITÀ). Per farlo, si utilizza l'operatore booleano XOR ('or esclusivo'), indicato con il simbolo ^: dalla tabella di verità illustrata a destra (analoga a quella di pag. 66) si sa che, dati due valori booleani A e B (detti operandi), A XOR B restituisce il valore 1 se e solo se uno solo degli operandi ha valore 1. Se dunque gli operandi sono *in9* e MODALITÀ, l'operazione otterrà i risultati riassunti nella tabella in basso.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Memorizzato il valore opportuno, occorre spegnere il LED, settando il corrispondente bit di *outl* al valore LED_OFF (1). L'istruzione debug, infine, visualizza il valore della lettura effettuata dal sensore. Il primo 'argomento' che viene passato a debug è la costante Spostatilo che vale 2: si tratta di un valore speciale che debug interpreta come istruzione per spostare il cursore in una posizione qualsiasi della finestra di debug, indicatagli dalle 'coordinate' che seguono, cioè due numeri (o espressioni matematiche con variabili e/o costanti), separati da una virgola (,). Il primo (nel listato, l'espressione (9+((2-*cont*)*5))) si riferisce alla colonna, mentre il secondo (nel listato, 4) alla riga. Perché i valori siano sovrascritti al posto giusto, infatti, il cursore dovrà essere portato sempre sulla riga numero 4 (che corrisponde alla quinta istruzione debug, dato che il conteggio comprende anche lo 0) e, quindi, spostarsi tra le colonne, fino alla posizione corrispondente al sensore operativo, scopo che si ottiene con l'espressione parametrica (9+((2-*cont*)*5)). Infatti, 9 è la colonna assunta come 'iniziale' (in cui è visualizzata la lettura del sensore Sx1); mentre il moltiplicatore 5 quantifica la distanza che separa le posizioni ([ ]) di visualizzazione di un sensore dal successivo; (2-*cont*), infine, è il parametro variabile che, a seconda del valore di *cont* (0, 1 o 2), stabilisce di 'quante volte 5' spostarsi (2, 1 o 0) e, di conseguenza, se sovrascrivere il valore relativo alla misurazione, rispettivamente, in Dx1, Line o Sx1.

	MODALITÀ	in9	in9 ^ MODALITÀ	
LINEA_BIANCA, percorso costituito da linea bianca su sfondo scuro	0	0	0	Nessuna linea rilevata
	0	1	1	Linea rilevata
LINEA_NERA, percorso costituito da linea nera su sfondo chiaro	1	0	1	Linea rilevata
	1	1	0	Nessuna linea rilevata