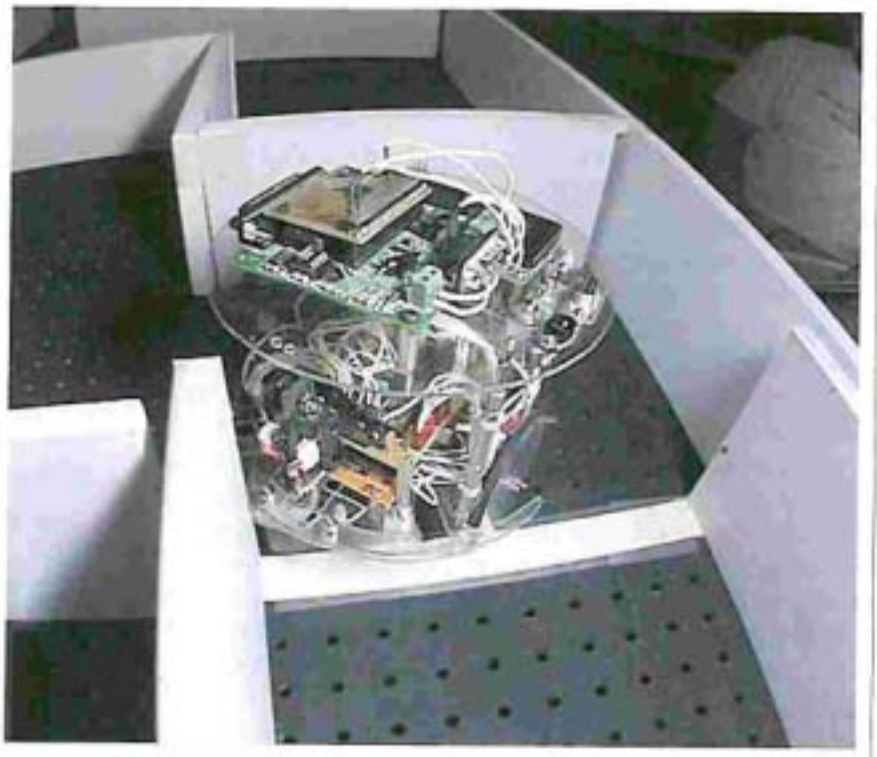
# A partire da qui

Con il secondo sensore IR, allegato a questo fascicolo, puoi completare la dotazione sensoriale posteriore del robot (seguendo le fasi di montaggio a pag. 188): disporrai così della copertura sensoriale necessaria perché il tuo robot possa affrontare la navigazione in ambienti di una certa complessità. Come vedremo nelle prossime pagine, il labirinto rappresenta un'utile astrazione di questo tipo di ambiente.

## DAL PRINCIPIO

Ciò che ci proponiamo ora è di affrontare il tema della navigazione da un punto di vista diverso dal solito. Ovviamente, ci occuperemo anche dei nuovi sensori, non però a partire da un listato finale preconfezionato, bensì dall'inizio, vale a dire dallo studio di un problema complesso, mediante la sua analisi, schematizzazione e suddivisione in sottoproblemi semplici. Dunque, il programma risolutivo, ben strutturato ed efficiente, sarà 'soltanto' il punto di arrivo cui tendere, mentre nei prossimi fascicoli ci si dedicherà allo studio e alla ricerca di algoritmi risolutivi 'intelligenti'. Dato che la maggior parte dei concetti chiave sono ormai noti, ai listati completi, spiegati passo passo, si dedicherà meno spazio; tuttavia non mancheranno, quando necessari, i dettagli su particolari porzioni di codice.



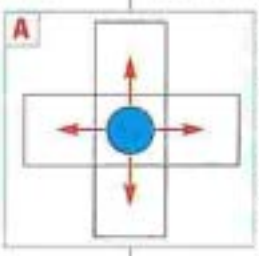
## QUALE APPROCCIO?

Come si diceva, uno dei classici problemi di un robot mobile è costituito dalla navigazione nei labirinti. Questi ultimi, però, costituiscono una vasta e complessa categoria di mondi che, dunque, prevedono diversi approcci risolutivi possibili. Tra questi (di cui si è parlato anche nella sezione Scienza, alle pagg. 165-168) si può istituire una sorta di gerarchia, ai cui estremi si collocano l'approccio *random* ('a caso') e quello 'strategico' (tramite mappatura): si va cioè dalla totale mancanza di una qualsiasi 'intelligenza' (intesa nel senso dell'IA), alla possibilità di elaborare strategie 'sensate', grazie alla costruzione di una mappa dell'ambiente già esplorato. Quest'ultimo è di gran lunga l'approccio preferibile, ma non è sempre possibile. Allora, come vedremo presto, a seconda degli obiettivi e del tipo di labirinto, si potrà optare per approcci intermedi che si avvalgano, cioè, di scelte ora casuali ora strategiche opportunamente alternate. Le une, infatti, offrono la certezza di uscire dal labirinto (prima o poi); le altre, invece, permettono di guadagnare tempo.

**Nella foto. Anche il labirinto più semplice costituisce un'ottima sfida per il programmatore di algoritmi di navigazione.**

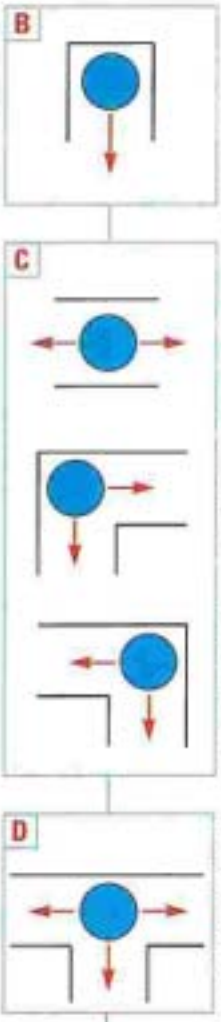
## QUALI LABIRINTI?

Nelle prossime pagine ci occuperemo di un sottoinsieme ristretto di labirinti, aventi caratteristiche ben precise. Sebbene quanto più numerose sono le caratteristiche date, tanto maggiori saranno i vincoli imposti, tali limitazioni si riveleranno utili per affrontare gradualmente lo studio degli algoritmi di navigazione che ci apprestiamo a compiere. Per derivare l'algoritmo, ci serviremo di regole 'euristiche' (ossia regole sensate, ricavate dall'esperienza diretta), ottenute a partire dalle caratteristiche imposte al labirinto: nel nostro caso, sono state scelte le seguenti caratteristiche:
(1) Tutti i varchi presenti nel labirinto sono sufficientemente larghi da permettere il passaggio del robot (baffi compresi).
(2) Il labirinto può essere suddiviso in sezioni o 'celle' di dimensione leggermente superiore a quella del robot.
(3) Gli angoli tra le pareti interne del labirinto sono di 90°.
(4) In ogni punto del labirinto, il robot può scegliere da un minimo di una a un massimo di tre direzioni possibili (rispetto alle quattro totali, perpendicolari tra loro a due a due).
(5) All'interno del labirinto non esistono altri ostacoli, mobili o fissi, oltre alle pareti.
(6) Esiste un'unica uscita, mentre il punto di partenza del robot è all'interno del labirinto.
Come vedremo tra poco, i vincoli imposti da queste caratteristiche sono forti ma utili; inoltre, mentre alcuni di essi dipendono dalla struttura sensoriale del robot e sono dunque imprescindibili, altri potranno essere allentati o 'rilassati' in un secondo momento.

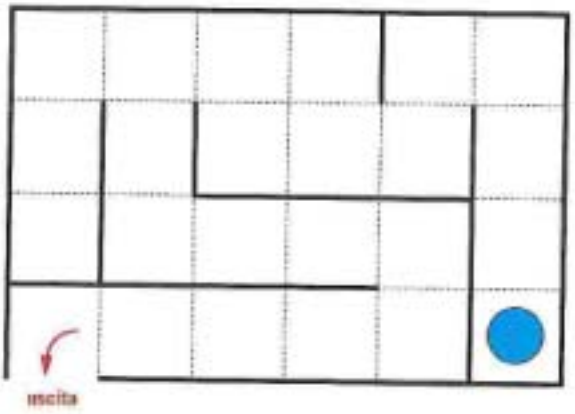



## L'ANALISI

Analizziamo ora le caratteristiche elencate. La (1) implica un vincolo facilmente rilassabile: infatti, anche senza questa caratteristica, cioè in presenza di passaggi ristretti, l'uso dei baffi permetterebbe di rilevare il blocco, cosicché il robot potrebbe poi cambiare direzione di percorrenza. Per semplificare la trattazione, invece, noi consideriamo i muri del labirinto come gli unici ostacoli presenti. La (2) impone alla navigazione uno schema simile a quello di alcuni giochi di ruolo (Dungeons&Dragons, per esempio), in cui il labirinto è diviso in 'celle' tra cui spostarsi lungo quattro direzioni ortogonali (come mostrato nel disegno **A**). La dimensione di ogni cella deve essere sufficiente perché il robot possa ruotare al suo interno ma, allo stesso tempo, non deve essere eccessiva: posizionato il robot al centro, i sensori IR devono poter rilevare un oggetto posto al bordo della cella.
Le caratteristiche (3) e (4), nel loro insieme, definiscono tutte le situazioni in cui il robot potrà trovarsi: in particolare, la tipologia degli incroci. Escluso il quadrivio (**A**), si includono situazioni che prevedono una sola direzione (**B**), oppure due (**C**) o tre (**D**). In seguito, dunque, ci si riferirà a una cella in base al numero di direzioni che ammette.




La (5) e la (6), infine, implicano vincoli intrinseci all'architettura del robot: i sensori IR, infatti, non sono adeguati per rilevare informazioni relative a ostacoli mobili (la cui gestione, inoltre, richiederebbe ulteriori vincoli che esulano dallo studio sui labirinti). D'altro canto, l'assenza di un'entrata è dovuta al fatto che, con la dotazione del robot (baffi e IR), è impossibile distinguere un'entrata da un'uscita: entrambe appaiono infatti sempre e solo come un varco aperto. Il labirinto della pagina accanto è un esempio che soddisfa tutti i vincoli imposti dalle caratteristiche sopra elencate. Tuttavia il nostro obiettivo non è quello di risolvere un preciso problema (vale a dire un particolare labirinto), bensì di analizzare un'intera categoria di labirinti con le stesse caratteristiche, al fine di trovare un metodo per affrontarne la soluzione (ossia la progettazione di un algoritmo di navigazione valido, il più generale possibile). **Il problema generale, allora, consisterà nel trovare l'uscita da un qualunque labirinto avente le caratteristiche discusse.**




**Nei disegni.**
**Un esempio di labirinto (a destra) che soddisfa i vincoli trattati nel testo: suddiviso in celle proporzionate al robot (in blu), ha solo un'uscita e nessun varco stretto. In più ogni cella potrebbe offrire quattro direzioni di percorrenza (A); tuttavia i muri sono stati disposti in modo da ridurle a una (B), due (C) o tre (D).**

## PRIME CONSIDERAZIONI

Date le caratteristiche (2) e (3), prima di iniziare ad affrontare il problema, è necessario fissare le dimensioni delle celle su cui si costruirà il labirinto; così come, si dovrà prevedere fin d'ora l'impiego futuro di routine che permettano al robot di ruotare a destra o a sinistra di 90° e di avanzare e indietreggiare di una distanza tale da spostarsi di cella in cella. Inoltre, queste ultime due azioni potranno rientrare, come subroutine, in un'unica routine di avanzamento dipendente dal valore di una variabile di stato binaria o *flag* (letteralmente 'bandiera', è la spunta, ossia il segno √ usato per contrassegnare i dati che, di un elenco, sono stati controllati): così, una sola variabile di tipo bit (come *direzione*, discussa a pag. 166) potrà gestire due sensi di percorrenza (avanti e indietro) di una stessa traiettoria. Per farlo, dunque, basterà modificare le routine di navigazione usate nei programmi visti finora, calibrando opportunamente i parametri dichiarati per adattarli alle necessità del problema in esame. D'altronde, sappiamo già che i baffi dovranno svolgere la funzione di controllori di rotta (permettendo al robot di correggere la traiettoria in caso di collisione con uno dei muri del labirinto). Andranno perciò previste anche adeguate routine di sterzo (a destra o a sinistra), ossia con parametri che ottengano un angolo di rotazione molto piccolo: la rotta, infatti, si corregge poco per volta; un angolo di sterzo ampio, invece, potrebbe portare il robot a collidere prima con la parete opposta a quella rilevata e poi di nuovo con la prima, comportando cioè uno scomodo effetto 'ping-pong'. Alla luce di quanto detto, si capisce perché **converrà quindi fissare a tutti gli effetti la suddivisione di compiti diversi tra sensori diversi**: infatti, date le caratteristiche dei sensori e degli attuatori e i possibili errori nel posizionamento iniziale, è improbabile che il robot proceda sempre perfettamente al centro delle celle e parallelamente ai muri del labirinto: ecco perché è utile che i baffi correggano la rotta. Delegando invece ai sensori IR la navigazione vera e propria, ci si può permettere di non insistere troppo sulla calibrazione delle routine di sterzo dei baffi: gli errori (purché limitati) verrebbero infatti corretti *runtime* ('durante l'esecuzione' del programma). Ciò che abbiamo dunque scoperto fin qui è che, per progettare l'algoritmo voluto, ci serviranno sicuramente un flag (o variabile binaria), che memorizzi la direzione di percorrenza corrente, le routine di navigazione e quelle di correzione dello sterzo (riassunte nella tabella sotto). Noterai che non è stato previsto il caso in cui entrambi i baffi di una coppia (che va comunque considerata frontale rispetto alla direzione di percorrenza) rilevino un ostacolo: l'errore infatti sarebbe eccessivo rispetto alle premesse (il robot dovrebbe essersi scontrato frontalmente con un muro); volendo, però, si potrà aggiungere una routine che, nel caso, faccia indietreggiare il robot. Da ultimo (per ora), **un'altra considerazione importante riguarda la struttura delle variabili di stato** che permetteranno la memorizzazione e la gestione delle rilevazioni sensoriali, grazie a opportune istruzioni di branch.

**L'analisi di un problema parte dalla definizione dell'ambiente di lavoro**

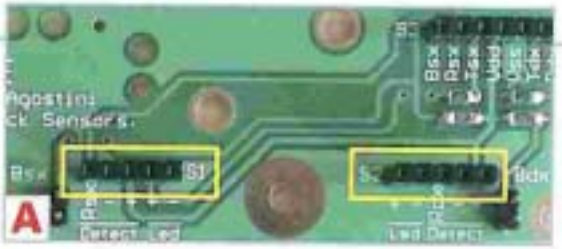
| Nome routine | Rilevazione sensoriale | Dipendenze |
|---|---|---|
| Avanza: | IR | direzione |
| Ruota_Destra: | IR | |
| Ruota_Sinistra: | IR | |
| Correggi_Destra: | baffi | |
| Correggi_Sinistra: | baffi | |

**A sinistra.**
**Puoi riassumere in una tabella ciò che, via via, scopri essere necessario per progettare l'algoritmo futuro.**

Posto che è utile usare baffi e IR in modo diverso, converrà anche separare le rispettive variabili: di solito, infatti, più si raggruppano dati e funzioni, più un codice si complica, fino a diventare ingestibile. Ad esempio, se tutti i sensori venissero raggruppati (bit per bit) in un'unica variabile di stato, un branch dovrebbe poi testare una serie lunghissima di casistiche: la variabile (di tipo byte) ha 8 bit, cioè 256 ($2^8$) condizioni possibili. Sarebbe perciò molto difficile trattare o modificare il branch relativo e, soprattutto, sarebbe inutile ai nostri fini. **Conviene perciò 'dedicare' le variabili di stato, distinguendo le misure dei sensori in base alle loro diverse funzioni: baffi da una parte (in una variabile) e IR dall'altra.** In seconda battuta, possiamo anche considerare che, in un ambiente definito come i labirinti in discussione, non capiterà mai di ottenere rilevamenti, nello stesso istante, sia dai baffi posteriori sia da quelli anteriori: si potrà allora decidere di memorizzarne i rilevamenti in due variabili distinte. Sul fronte dei quattro sensori IR, invece, prevedere il raggruppamento dello stato di ciascuno in un'unica variabile si rivela molto più utile e, come vedremo presto, potrà sveltire la soluzione di molte situazioni.

## Le fasi di montaggio


*Allegati a questo fascicolo ci sono quattro cilindri in plastica nera e due componenti IR, il LED e il ricevitore (nelle foto riquadrate in rosso). I due componenti IR, in particolare, sono identici a quelli allegati al precedente fascicolo (nella foto riquadrata in verde), che dovrai recuperare prima di procedere al montaggio descritto di seguito. Opportunamente assemblati, tutti questi componenti costituiranno due nuovi sensori IR che dovranno essere alloggiati negli slot S1 e S2 del circuito DeA Back Sensors. Per guidarti nel loro corretto posizionamento, ciascuno dei cinque contatti di ogni coppia LED-ricevitore è stato identificato (nelle foto sopra) da una dicitura, corrispondente alle indicazioni presenti sul circuito; inoltre, la freccia rossa indica la parte semisferica di ciascun ricevitore, che dovrà essere rivolta verso l'esterno del robot.*


**A Per eseguire correttamente il montaggio dei due sensori IR e il loro alloggiamento sul circuito posteriore DeA Back Sensors, ti sarà utile consultare le istruzioni fornite alle pagine 59-61 in riferimento agli analoghi sensori posizionati sulla scheda madre. La procedura operativa, infatti, è la stessa. Prima di tutto, individua i due slot S1 e S2 (riquadrati in giallo) interessati da questa fase di montaggio. Di ciascuno slot sono indicati i tre socket (Detect) per il ricevitore e i due (Led) per l'emettitore: il corretto abbinamento di ciascun contatto dei componenti IR con il rispettivo socket è ribadito, come detto, nelle foto in alto a sinistra.**


**B•C Ricorda che, dei due contatti del LED, quello inizialmente più lungo corrisponde all'anodo (+), pertanto controlla di averli piegati in modo corretto prima di accorciarli: sia l'emettitore sia il ricevitore, una volta alloggiati nei rispettivi socket, dovranno infatti essere rivolti verso l'esterno del robot.**