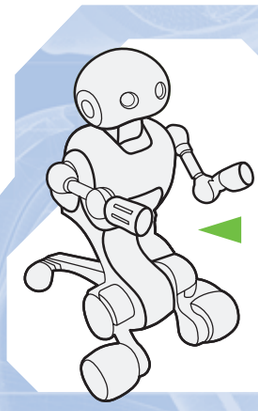
IL GOMITO PER LA PINZA DI I-DO1



In allegato a questo fascicolo trovi quattro elementi: essi comporranno il gomito dell'avambraccio 'potenziato', quello, cioè, connesso alla pinza di I-Droid01.

Con il fascicolo precedente hai ricevuto, tra l'altro, le due metà della struttura connessa alla pinza di I-Droid01, ossia l'avambraccio che andrà sostituito a quello destro che il robot presenta in questo momento. Per essere montato al braccio del robot, però, questo avambraccio 'potenziato' necessita anche del giunto del gomito. I componenti allegati a questo fascicolo realizzano proprio tale giunto: suddiviso in due metà, esso sarà agganciato all'estremità del braccio destro del robot. Una volta montato, il gomito sarà mobile, nel senso che potrà essere fatto ruotare agendo direttamente su di esso (proprio come accade per quelli attuali). Oltre alle due metà del giunto, tra gli allegati trovi anche due elementi neri: serviranno a ricoprire i fori del gomito riservati all'albero filettato e al dado, che fisseranno l'avambraccio al braccio.

COMPONENTI

1. Parte interna del gomito per la pinza
2. Parte esterna del gomito per la pinza
3. Coperchio interno del gomito per la pinza
4. Coperchio esterno del gomito per la pinza

Gli elementi allegati a questo fascicolo realizzeranno il nuovo gomito, che sostituirà quello dell'avambraccio al momento installato (immagine a destra).

I COPERCHI

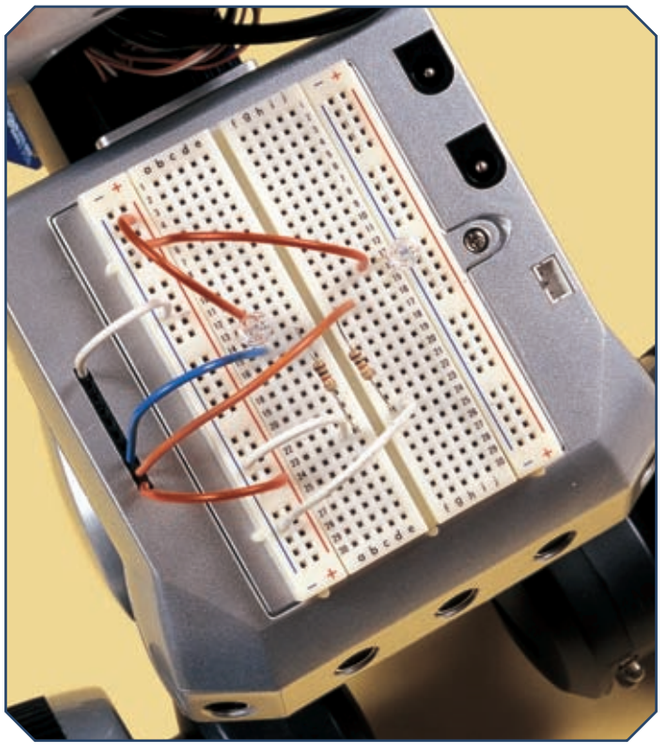
DATI

Come detto, tra i componenti allegati si trovano anche due coperchi di colore nero. Essi non hanno forma uguale, come è facile vedere osservandone la parte interna (immagini a destra). Uno di essi servirà a ricoprire il gomito nel suo lato esterno rispetto al robot (immagine qui a destra): esso presenta una cavità circolare all'interno di un profilo anch'esso circolare. L'altro coperchio (immagine sotto a destra), invece, permetterà di ricoprire il lato interno del gomito: esso presenta una cavità circolare (simile in dimensioni a quella dell'altro coperchio), inclusa però in un profilo non perfettamente circolare, in cui compare una parte sagomata (indicata nell'immagine a lato).

FUGGI-SEGUI LUCE IN VISUAL C-LIKE

La programmazione di I-Droid01 può avvenire non solo usando Java, ma anche il C-like e il Visual C-like, in quest'ultimo caso grazie all'editor a blocchi. Nelle prossime pagine vediamo un esempio di programma scritto proprio in Visual C-like, grazie al quale I-D01 potrà seguire o fuggire da una fonte luminosa.

Abbiamo avuto modo, con i fascicoli 76, 77 e 78, di vedere alcuni esempi di programmazione in Java del robot, in particolare per quanto riguarda la connessione e la disconnessione da PC e la gestione via interfaccia grafica dei LED della testa (occhi e orecchie). Ma Java non è il solo linguaggio di programmazione 'inteso' da I-Droid01. Come noto, infatti, il robot può eseguire istruzioni scritte in C-like e in Visual C-like. E, sebbene Java offra maggiori potenzialità, a fronte però di una complessità anch'essa maggiore, nella sua semplicità di utilizzo anche la programmazione visuale a blocchi può realizzare comportamenti strutturati e tutt'altro che banali. È il caso del programma che viene mostrato in queste pagine: esso definisce i comportamenti per seguire una fonte luminosa e per fuggire da essa sfruttando, dal punto di vista 'hardware', il kit di sensori di luce per la breadboard allegato al fascicolo 66. Prima di realizzare il codice del programma, quindi, è necessario montare sulla breadboard di I-Droid01 il sistema sensoriale.



Il programma in Visual C-like illustrato in queste pagine permette di usare il kit di sensori di luce, allegati al fascicolo 66. Sopra, il circuito (con i relativi cavi di collegamento, allegati al fascicolo 67), montato sulla breadboard del robot.

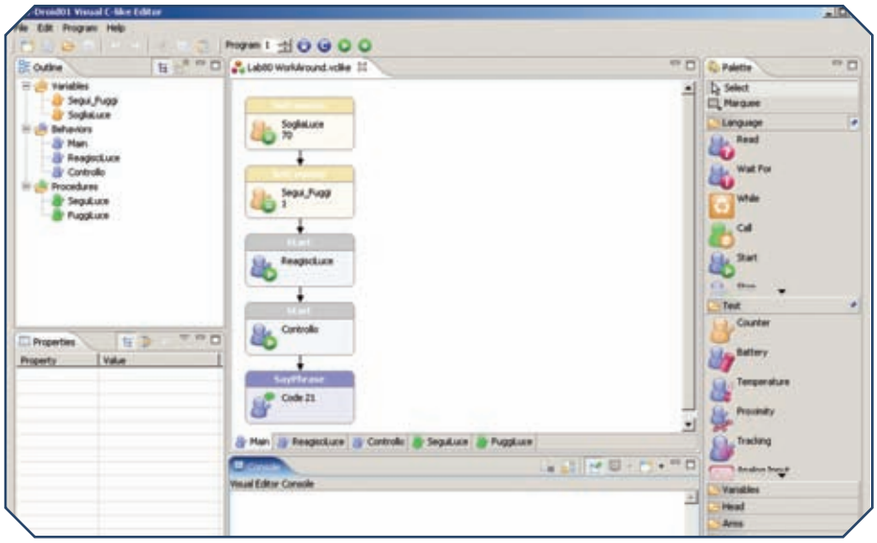
IL KIT DEI SENSORI

Innanzitutto, quindi, vanno recuperati gli elementi allegati al fascicolo 66 e i cavetti per la breadboard, allegati al fascicolo 67. In seguito, a robot spento, va realizzato il circuito del sistema sensoriale, come mostrato nelle istruzioni dei fascicoli succitati. Nel posizionamento dei vari elementi, va ricordato di prestare particolare attenzione al 'verso' dei sensori di luce, che presentano due contatti metallici (più precisamente anodo e catodo) con ruolo molto diverso: inserire nella breadboard le estremità dei sensori in modo errato non permetterà al circuito di funzionare e, in più, potrebbe comportare danneggiamenti anche seri ai sensori stessi.

GLI ELEMENTI DEL PROGRAMMA

Una volta realizzato il circuito, si può passare alla composizione del programma in Visual C-like. Tale programma ha bisogno di due variabili (Variables, nel 'gergo' del Visual C-like Editor), tre comportamenti (o Behaviors) e due procedure (Procedures). Le due variabili Segui_Fuggi e SogliaLuce permettono al robot rispettivamente di memorizzare qual è l'azione che gli si richiede (seguire la fonte luminosa oppure

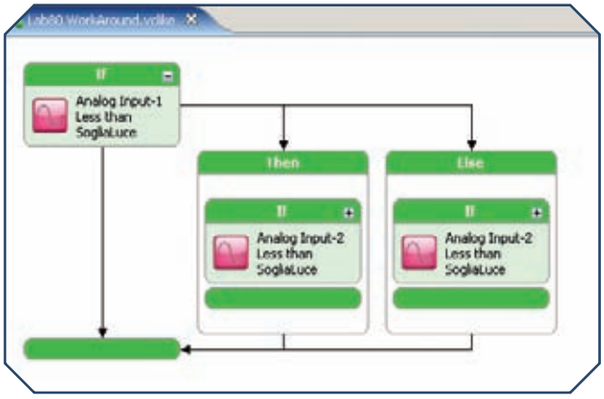
fuggire da essa) e qual è la soglia di intensità luminosa (rilevata dai sensori) che faccia reagire il robot.
I comportamenti, invece, indirizzano i 'flussi' principali di esecuzione del programma. Il comportamento principale (Main), ovviamente, è quello che gestisce l'esecuzione nel suo inizio, richiamando al suo interno gli altri comportamenti. Da parte sua, invece, il comportamento ReagisciLuce impartisce le direttive principali nella reazione alla presenza della fonte luminosa, invocando a proposito le procedure apposite. Infine, il comportamento Controllo gestisce la selezione tra 'inseguimento' della fonte luminosa o 'fuga' da essa, selezione che l'utente può effettuare tramite appositi comandi vocali. Le procedure, infine, sono SeguiLuce e FuggiLuce; come è facile intuire dai loro nomi, esse indicano al robot le azioni concrete da effettuare per inseguire la fonte luminosa rilevata e per fuggire da essa. Vediamo ora nel dettaglio proprio tali procedure, non prima però di aver parlato delle variabili. Infine, concluderemo con la descrizione dei tre comportamenti.



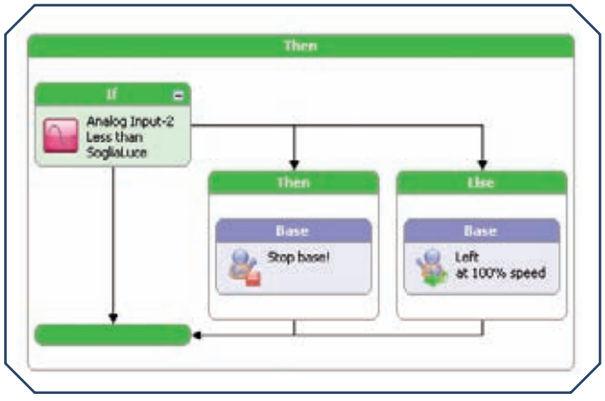
Sopra, una schermata del Visual C-like Editor che mostra, tra l'altro, l'area di Outline (in alto a sinistra), quella di Properties (in basso a sinistra) e quella principale (al centro), dove è visualizzato il comportamento Main del programma.

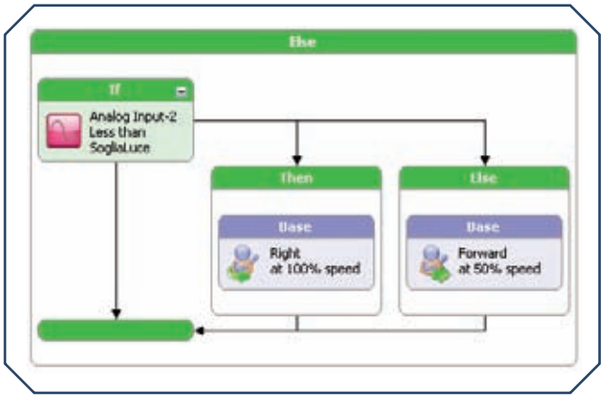
VARIABILI E PROCEDURE

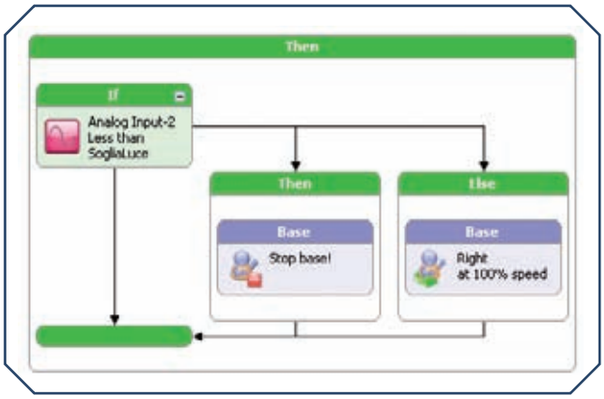
Come detto, il programma qui mostrato utilizza due variabili. In Visual C-like le variabili sono essenzialmente contatori, ossia quantità numeriche che possono essere settate, confrontate,

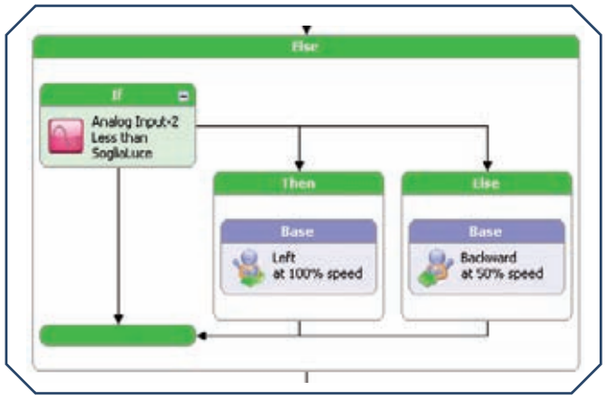


Alcune schermate del Visual C-like Editor che illustrano la procedura SeguiLuce. Essa indica le azioni che il robot intraprende quando deve seguire una fonte luminosa. La procedura è organizzata in un If generale (qui a destra) all'interno del quale si trovano altri due test: il primo (qui sotto) distingue tra la situazione di nessuna fonte luminosa rilevata e di fonte luminosa a sinistra; il secondo (sotto a destra) la situazione di fonte luminosa a destra e di fronte.





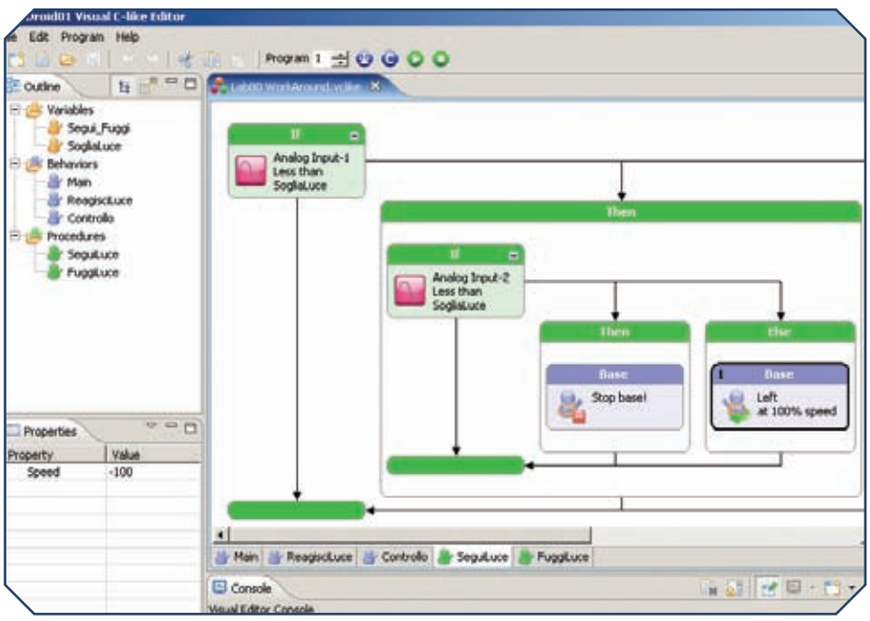




Sopra, due schermate relative alla procedura FuggiLuce. Vengono mostrati i due If interni, quello per nessuna fonte luminosa e fonte luminosa a sinistra (qui sopra) e di fonte luminosa a destra e di fronte (immagine sopra a destra). L'If esterno è del tutto identico a quello della procedura SeguiLuce.

incrementate e decrementate. All'interno del Visual C-like Editor, la definizione di una variabile avviene cliccando con il tasto destro del mouse sulla piccola icona, rappresentante le variabili stesse, posta all'interno della zona chiamata 'Outline' (in alto a sinistra nell'interfaccia grafica dell'Editor). Eseguendo il clic con il mouse, viene mostrato un menu, tra le cui voci appare anche Add Variable: utilizzando tale funzione è possibile creare una nuova variabile, alla quale viene dato un nome di default, comunque modificabile (area Properties dell'Editor, in basso a sinistra). Nella scrittura del programma, le prime operazioni da compiere consistono proprio nella creazione delle variabili Segui_Fuggi e SogliaLuce, di cui abbiamo già illustrato i ruoli all'interno dell'applicazione.

Passando alle procedure, esse sono alla base del funzionamento del programma. Cominciamo con SeguiLuce. Per realizzarla, è sufficiente cliccare col tasto destro del mouse sull'icona Procedures della zona Outline del Visual C-like Editor, per poi accedere alla funzione Add Procedure. Una volta cambiato il nome di default (zona Properties, in basso a sinistra dell'interfaccia dell'Editor) si può passare alla definizione del 'corpo' della procedura. Essa è composta da tre test, uno 'ad alto livello' e gli altri due 'innestati' all'interno di quello principale. Grazie a tali test, è possibile definire determinate situazioni 'ambientali' (in questo caso riferite alla presenza o meno di una fonte luminosa), alle quali il robot dovrà reagire. Così, il test ad alto livello comincia col discriminare i casi in cui l'ingresso analogico numero 1 del marsupio (collegato alla breadboard) rileva un valore minore o maggiore della soglia di luce impostata nella variabile SogliaLuce. In particolare, l'ingresso analogico è associato al sensore di destra (rispetto al robot) della breadboard, quindi riguarda l'intensità di luce rilevata alla destra del robot. All'interno di questo test, come detto, se ne trovano altri due, a pari livello. Essi, come facile immaginare, non fanno altro che determinare la situazione, in base al valore dell'ingresso analogico numero 2, relativo al sensore di sinistra della breadboard. Complessivamente, quindi, si vengono a distinguere quattro casi particolari, frutto

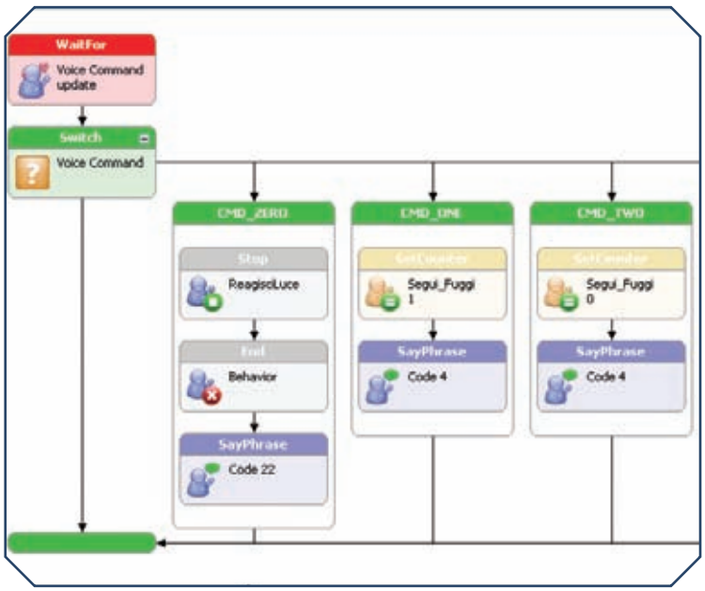


A sinistra, il Visual C-like Editor in cui viene mostrata la procedura SeguiLuce. In particolare, è stato selezionato il blocco Base->Left at 100% speed. La velocità può essere modificata agendo sull'apposito campo dell'area Properties, in basso a sinistra dell'interfaccia.

delle quattro possibili combinazioni:
– ingresso 1 minore della soglia e ingresso due minore della soglia;
– ingresso 1 minore della soglia e ingresso 2 maggiore della soglia;
– ingresso 1 maggiore della soglia e ingresso 2 minore della soglia;
– ingresso 1 maggiore della soglia e ingresso 2 maggiore della soglia.

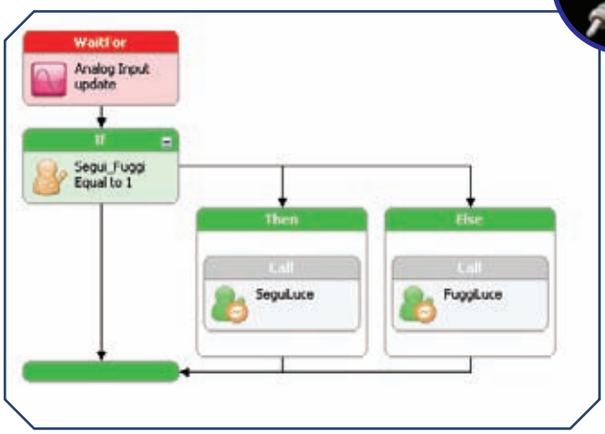
Tali casi rappresentano le quattro situazioni possibili, e cioè, rispettivamente:
– non ci sono fonti luminose significative rilevate dal robot;
– c'è una fonte luminosa alla sinistra del robot;
– c'è una fonte luminosa alla destra del robot;
– c'è una fonte luminosa di fronte al robot.

Le azioni indicate dalla procedura SeguiLuce in risposta a tali situazioni sono semplici: se non



Sopra, una parte del comportamento Controllo, in cui vengono gestiti i comandi vocali. Non sono mostrati i comandi oltre quello 'due', dato che non comportano nessuna azione.

ci sono fonti, il movimento delle ruote viene interrotto (Base->Stop base!); se la fonte è a sinistra, il robot gira a sinistra (Base->Left at 100% speed); se la fonte è a destra, I-D01 si volta verso destra (Base->Right at 100% speed); infine, se la fonte si trova di fronte al robot, esso si muove in avanti (Base->Forward at 50% speed). La procedura FuggiLuce è del tutto simmetrica. Anche in questo caso, grazie a opportuni test sugli input analogici, vengono individuati gli stessi quattro possibili casi, ai quali il programma fa corrispondere le opportune azioni, opposte a quelle indicate nella procedura di inseguimento. Così, se non c'è luce,

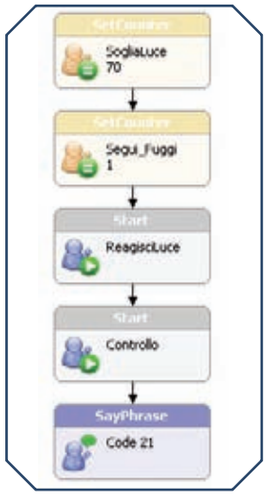


Sopra, i blocchi del comportamento ReagisciLuce, che invoca le procedure SeguiLuce e FuggiLuce sulla base del valore della variabile di test Segui_Fuggi.

il robot si ferma (unico caso in cui l'azione intrapresa è uguale a quella della procedura SeguiLuce); se la luce è sulla sinistra, il robot gira a destra; se la luce è a destra, I-D01 si volta a sinistra; infine, se la luce è di fronte, il robot si sposta all'indietro.

COMPORTAMENTI

Le procedure, quindi, sono usate per indicare le azioni che il robot deve intraprendere in determinati casi. I comportamenti, invece, operano a un livello più astratto, 'convogliando' il flusso dell'esecuzione e organizzando il tutto. Cominciamo illustrando il comportamento ReagisciLuce. Il suo compito è semplice: chiamare la giusta procedura (SeguiLuce o FuggiLuce) sulla base della variabile Segui_Fuggi, che permette di distinguere se l'utente desidera che I-Droid01 insegua una fonte luminosa oppure fugga da essa (vedremo più avanti come avviene nel concreto tale scelta). Come è facile immaginare, quindi, il comportamento ReagisciLuce è costituito essenzialmente da un test sulla variabile Segui_Fuggi: se essa vale 1 (valore scelto per indicare che l'utente richiede l'inseguimento della fonte luminosa) viene invocata SeguiLuce, altrimenti viene lanciata l'esecuzione



A destra, un'immagine che mostra il comportamento principale (Main). Esso inizializza le due variabili utilizzate nel programma (SogliaLuce e Segui_Fuggi), poi lancia l'esecuzione dei comportamenti ReagisciLuce e Controllo e fa emettere un suono a I-D01.

di FuggiLuce. Il test viene preceduto da un blocco di tipo WaitFor, che indica quando il comportamento deve 'valutare' la situazione. Nel nostro caso, ReagisciLuce riesamina la situazione quando avviene un cambiamento negli ingressi analogici. Anche il comportamento Controllo fa uso di un test. Stavolta, però, a differenza degli If impiegati nelle procedura SeguiLuce e FuggiLuce e nel comportamento ReagisciLuce, qui si utilizza un blocco Switch. Quest'ultimo permette di distinguere fra più casi diversi, proprio come richiesto in Controllo. Tale comportamento, infatti, ha il compito di permettere all'utente di effettuare la scelta tra l'inseguimento e la fuga dalla fonte luminosa, tramite comandi vocali. Più nello specifico, vengono usati i comandi vocali 'zero', 'uno' e 'due' del Word Set 11 (accessibile dal Word Set 2 pronunciando la parola 'comando'). Il comando 'zero' è associato alla conclusione del programma: viene interrotto il comportamento ReagisciLuce e terminata l'esecuzione. Il comando 'uno', invece, permette di selezionare l'inseguimento della fonte luminosa, impostando il valore della variabile Segui_Fuggi a 1. Il comando vocale 'due', infine, permette di scegliere la fuga dalla luce, tramite il settaggio della variabile Segui_Fuggi con il valore 0. In questi due ultimi casi, il robot pronuncia anche un suono di conferma. Non rimane che organizzare l'esecuzione di ReagisciLuce e Controllo. Ciò viene fatto nel Main. In esso vengono definiti i valori di SogliaLuce (in questo esempio impostato a 70, ma che può essere variato per adattare il programma a diverse situazioni ambientali, magari trovando la soglia migliore tramite la funzione Test GPIO del menu del display) e di Segui_Fuggi (impostata di default con il valore 1). Poi viene lanciata l'esecuzione dei comportamenti ReagisciLuce e Controllo, che in questo modo saranno attivati.

ESEMPI DI PROGRAMMAZIONE

Codice C-like corrispondente al programma descritto nelle pagine precedenti. Tale codice può essere visualizzato a partire dal diagramma a blocchi, tramite la funzione 'Show source...' del menu Program nel Visual C-like Editor. Anche nel codice in C-like sono descritti i comportamenti (Main, ReagisciLuce e Controllo) e le procedure (SeguiLuce e FuggiLuce).

CODICE C-LIKE DEL PROGRAMMA

```
#include "c-like.h"
#include "robot.h"


counter Segui_Fuggi = new(counter);
counter SogliaLuce = new(counter);

declare( behavior(ReagisciLuce) );
declare( behavior(Controllo) );

void SeguiLuce();
void FuggiLuce();


define( behavior(Main))
{
      set(SogliaLuce,70);
      set(Segui_Fuggi,1);
      start(ReagisciLuce);
      start(Controllo);
      say_phrase(21);
}


define( behavior(ReagisciLuce) )
{
      local(analog_in) = wait_for (analog_in, update);
      if (get(Segui_Fuggi) == 1)
      {
            SeguiLuce();
      }
      else
      {
            FuggiLuce();
      }
}

define( behavior(Controllo) )
{
      local(voice_cmd) = wait_for (voice_cmd, update);

      switch (local(voice_cmd))
      {

      case CMD_ZERO:
            stop(ReagisciLuce);
            end();
            say_phrase(22);
            break;

      case CMD_ONE:
            set(Segui_Fuggi,1);
            say_phrase(4);
            break;

      case CMD_TWO:
            set(Segui_Fuggi,0);
            say_phrase(4);
            break;

      case CMD_THREE:
            break;

      case CMD_FOUR:
            break;

      case CMD_FIVE:
            break;

      case CMD_SIX:
            break;

      case CMD_SEVEN:
            break;

      case CMD_EIGHT:
            break;

      case CMD_NINE:
            break;

      case CMD_TEN:
            break;

      default:
            break;

      }
}

void SeguiLuce()
{
      if (local(analog_in).pin_1 < get(SogliaLuce))
      {
            if (local(analog_in).pin_2 < get(SogliaLuce))
            {
                  base_stop();
            }
            else
            {
                  rotate(-100);
            }
      }
      else
      {
            if (local(analog_in).pin_2 < get(SogliaLuce))
            {
                  rotate(100);
            }
            else
            {
                  move_speed(50);
            }
      }
}


void FuggiLuce()
{
      if (local(analog_in).pin_1 < get(SogliaLuce))
      {
            if (local(analog_in).pin_2 < get(SogliaLuce))
            {
                  base_stop();
            }
            else
            {
                  rotate(100);
            }
      }
      else
      {
            if (local(analog_in).pin_2 < get(SogliaLuce))
            {
                  rotate(-100);
            }
            else
            {
                  move_speed(-50);
            }
      }
}
```