Wall-G

(23.04.2009 Toro)

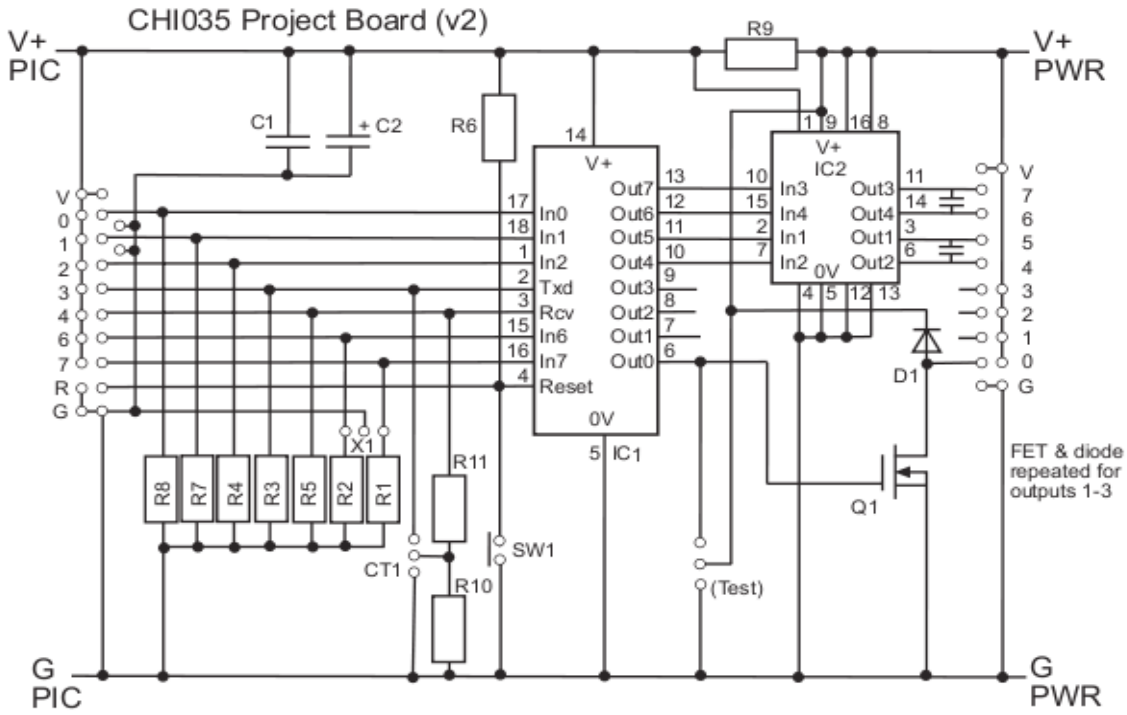
Introduzione

Perché Wall-G ?
Non c'è stata una ragione particolare, ma dopo avergli visto muovere la testolina alla prima accensione, mi è venuto in mente il robottino timido di Walt Disney e lo ho chiamato così. G sta per grey :)
Wall-G non è un robot b.e.a.m. Per questa volta mi sono dedicato alla programmazione, scegliendo però una demo-board ed un processore semplice e programmabile con un linguaggio semplice: il Basic. Sempre per restare fedele alla filosofia b.e.a.m.

Descrizione del comportamento

Wall-G non ha una funzione precisa, almeno per il momento. Il suo scopo è sopravvivere nel suo mondo ed aiutarmi a comprendere con il suo comportamento, quali strategie adottare per permettergli di potersi muovere senza rimanere impigliato in qualche pericoloso (per lui) groviglio di sedie o sotto un divano o incastrato tra due mobili.
Il suo è un carattere mite e curioso allo stesso tempo. Passa il suo tempo a vagare per casa alla ricerca della luce (è fototropico) che è il suo scopo primario. Ma non deve essere troppo intensa, altrimenti si spaventa e si blocca. Cerca con la testa una direzione con meno luce e vi si dirige alla ricerca di un luogo in ombra (diventa fotofobico). Una volta trovato continua a muoversi finché la luce non diventa così bassa da farlo fermare restando in uno stato di quiete. Da questo stato si toglierà solo quando la luce ambiente diventerà sufficientemente alta. In questo caso ritornerà a muoversi, di nuovo alla ricerca di una fonte luminosa. Periodicamente controllerà lo stato della sua batteria e se scarica si porterà in uno stato di freeze in attesa di essere ricaricato. E' anche contento quando trova la luce, infatti suona.

Schema



Wall-G usa come elettronica la Project Board CHI035 della Revolution Education Ltd equipaggiata di un controller Picaxe 18X dotato di 5 ingressi utili di 4 uscite di potenza a Fet e di 4 uscite derivate dal motor controller L293D che vengono usate per pilotare i due motori del robot. I motori sono dei MicroMotors da 75 rpm a 12V con una coppia massima di 2.55 Kg*cm ed un assorbimento massimo di 440 mA a 12V. Essendo però alimentati da una batteria a 6V tramite l'integrato L293D, ai suoi capi arrivano più o meno 4-4,5 Volt che sono però largamente sufficienti a spostare il robot. La velocità di crociera sarà però molto bassa. Sono stati usati perché avanzati dal vecchio chassis del robot Dobby.

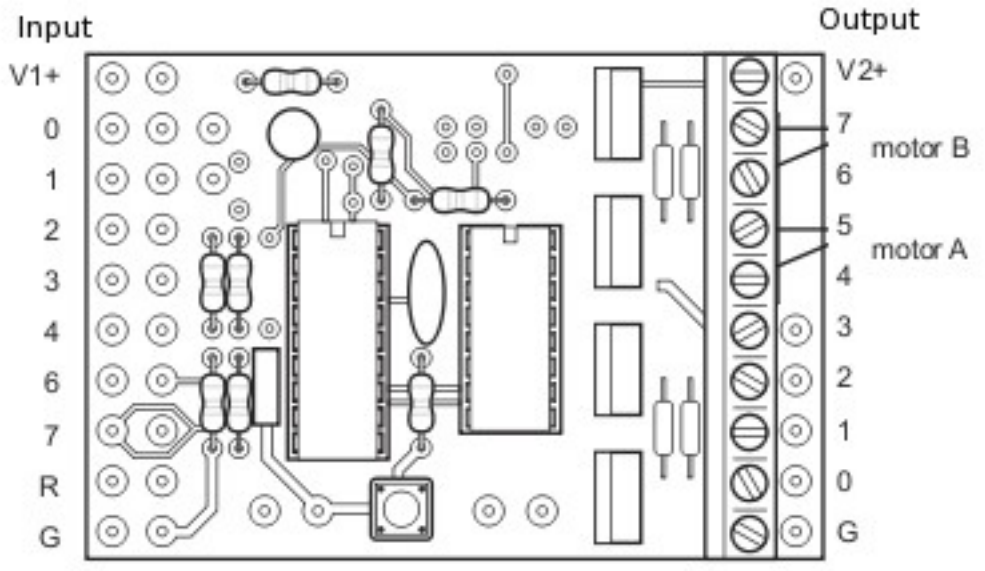
Gli ingressi sono 5 di cui 2 usati in analogico. A questi due (**in0** e **in1**) sono collegate due LDR, gli occhi di Wall-G. Gli altri tre ingressi (**in2, in6, in7**) sono collegati ai bumper ottici sinistro, destro e centrale. Gli ingressi **in3** e **in4** sono riservate alla programmazione del Picaxe 18X.

Le quattro uscite sono utilizzate così:

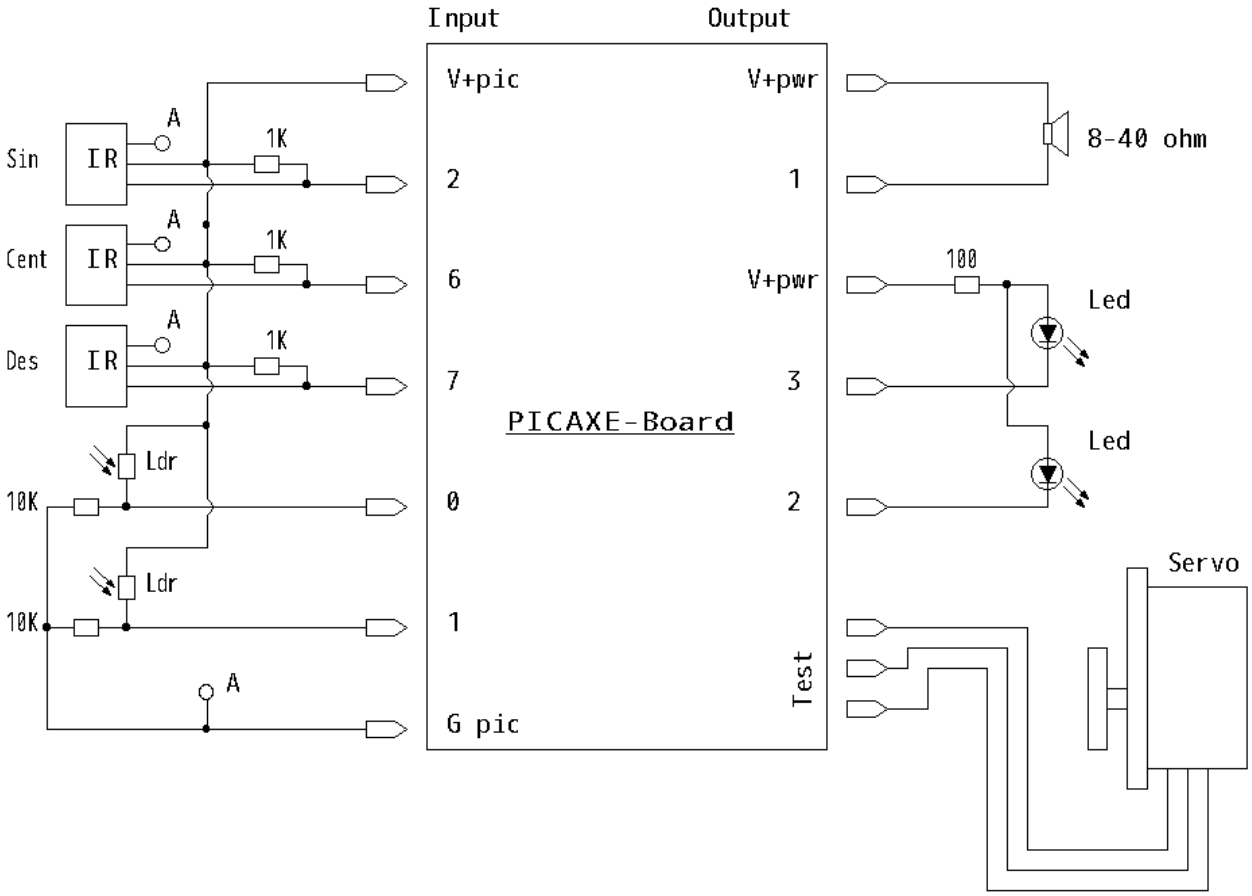
- **Out0** (*conn. Test*) è connessa al servo Hitech 3003 che è la testa del robot su cui sono collegate le LDR per una “visione” differenziale.
- **Out1** è collegata ad un trasduttore acustico e serve ad esternare i propri intenti (la musichetta all'atto dell'accensione) o le proprie emozioni (quando vede troppa luce o quando di sveglia).

- **Out2/out3** sono collegate a due led rossi e danno la misura approssimativa della quantità media di luce catturata dalle due LDR. Serve al solo scopo di calibrazione delle costanti interne al programma. Danno una misura in codice binario su due bit della misura effettuata dagli ADC interni al Picaxe (0-255). Per cui:

  led1 = 0 led0 = 0 => 0-63
  led1 = 0 led0 = 1 => 64-127
  led1 = 1 led0 = 0 => 128-191
  led1 = 1 led0 = 1 => 192-255



Layout della Picaxe Project Board



Schema dei collegamenti delle periferiche

Montaggio

Per l'assemblaggio di Wall-G sono state usate tutte parti riciclate, a cominciare dalla base circolare in legno che è quella del vecchio Dobby.
La base è in legno ed ha il diametro di 25 cm. E' realizzata in legno compensato da 4 mm di spessore. Sotto la base, tra due coppie di listelli da 2x2 cm vengono ospitati due motori della Micromotors da 75 giri al minuto. I collegamenti verso il ponte-H passano attraverso un foro centrale di 1 cm di diametro. Le ruote sono Tamiya e sono montate sui motori tramite due mozzi adattatori in alluminio.
Posteriormente è montato un ruotino pivottante di quelli che si trovano nei vari Brico. Anteriormente è presente uno stabilizzatore in legno ricavato dallo scasso per l'alloggiamento delle ruote. Nella parte superiore della base sono alloggiati la batteria da 6 volt, la Project Board, i sensori e la testa del robot. La testa è realizzata con un servo Hitech S3003 con ai lati due tappi di pennarello bianco opaco contenenti le due LDR per il tracking della luce.
Una nota particolare va fatta su questi due “occhi”. In origine avevo pensato a due tubi nero opaco dove alloggiare le LDR ed avere una forte direzionalità. Fortunatamente non sono riuscito a reperirli di diametro sufficiente a contenere le LDR e così ho ripiegato sui tappi di pennarelli color bianco opaco. Questa è stata una fortuna in quanto danno una “visione laterale” debole ma sufficiente a direzionare preventivamente il robot verso la sorgente luminosa. Questo rende il robot più “soft” nella ricerca della luce e più efficace poiché non ha bisogno di avere la luce perfettamente allineata agli occhi per poterla vedere. Inoltre la direzionalità degli occhi non ne risente in maniera particolare.
Ai lati, a ore 10, ore 12 e ore 2 sono presenti tre bumper a infrarossi dello stesso tipo di quelli

usati su Dobby (IS471F) con un range di rilevazione di circa 5 cm. Successivamente ne verrà applicato uno meccanico nel semicerchio anteriore, per la rilevazione degli ostacoli bassi o invisibili agli infrarossi.

Logica di Programmazione

Il programma scritto in basic è abbastanza semplice da capire e sufficientemente commentato. E' costituito da un ciclo principale contenente tre sottoprogrammi che usano il polling per interrogare le periferiche.
Nell'ordine vengono interrogate le LDR, poi i sensori perimetrali. In apparenza quindi la priorità sembrerebbe del sistema di tracking, invece no. Se un sensore perimetrale viene attivato, il controllo passa immediatamente a questo e dopo ritorna al sistema di tracking.

Da realizzare:
Nel caso la batteria scenda al di sotto di un valore di sicurezza, il robot si fermerà, spegnerà tutti gli attuatori (motori, led, ecc.) e si metterà in modalità basso consumo in attesa di essere ricaricato. Al momento non è presente un sottoprogramma per la guida verso una stazione per la ricarica.

Listato del programma Basic

In testa al programma ci sono i commenti necessari a ricordare le funzioni dei pin di ingresso, uscita e delle variabili (registri) interne al microcontroller visto che le risorse sono limitate e vengono riutilizzate per più funzioni.
Segue il main-program che è composto da alcune istruzioni per la priorità delle azioni e da tre chiamate a sottoprogrammi: tracking, bumper e antistallo. I primi due sottoprogrammi sono abbastanza banali.

- **Il primo esegue il tracking** di una fonte luminosa utilizzando due LDR collegate agli ingressi ADC del controller. Le due intensità vengono misurate e ne viene fatta la somma algebrica. In base al risultato viene decisa la direzione verso cui muoversi. Naturalmente è presente una isteresi variabile per evitare azioni incontrollate del sistema.

- **Il secondo** esegue le operazioni anticollisione rispetto gli oggetti dell'ambiente. A questo scopo sono presenti tre bumper a luce infrarossa.

- **Il terzo verifica che non sia in stallo** controllando che non ci siano state troppe oscillazioni gira-a-destra e gira-a-sinistra. Se ce ne sono state più di 5 suppone di essere andato in stallo ed esegue un dietrofront temporizzato e variabile a caso. Questo per evitare, in casi particolari, di ritornare di nuovo in stallo. La temporizzazione è resa variabile sommando alla parte fissa un valore di tempo determinato dalla istruzione “random”.

A seguire ci sono tutti i sottoprogrammi necessari al funzionamento. Da notare che in alcuni casi i due sottoprogrammi per luce e bumper possono entrare in conflitto tra di loro (es. luce a destra e bumper a destra o viceversa) per cui dopo “n” oscillazioni occorre effettuare una manovra di disimpegno anche se questa va a scapito dello scopo del robot e cioè ricercare la luce. Anche la quantità di luce ambiente varia il comportamento del robot in quanto con troppa luce le LDR vanno in saturazione e non raggiungono il loro scopo che è quello di decidere la direzione di ricerca (tracking).

Qui sotto un esempio di programma sperimentato su Wall-G. Il programma definitivo ovviamente ha subito diverse modifiche e correzioni in fase di debug:

```
rem ********************************************
rem * Programma:  WALL-G                       *
rem * Data:       26-04-2009                   *
rem * Rev.:       0.9                          *
rem * Autore:     Domenico Mancini aka greybear*
rem * Licenza:    GNU-GPL                      *
rem * Note:                                    *
rem * Questo programma per  piattaforma circolare *
rem * mobile simula un comportamento  fototropico *
rem * almeno fino a che la luce non diventa trop- *
rem * po forte. In questo  caso il  comportamento *
rem * si inverte ed il tropismo diventa negativo. *
rem * Una volta  raggiunto  un luogo  in ombra il *
rem * robot si ferma  e l?  permane.  Se il luogo *
rem * ritorna ad una luminosit? accettabile allo- *
rem * ra il robot  ritorna  alle  caratteristiche *
rem * comportamentali precedenti.               *
rem ********************************************
rem
rem
rem ****************************
rem INGRESSI e USCITE
rem ****************************
rem
rem in0 anaA/log1 : (occhio sinistro)
rem in1 anaB/log2 : (occhio destro)
rem in2 log3      : (bumper sinistro)
rem in3 txd       :  n/a
rem in4 rxd       :  n/a
rem in6 log4      : (bumper centrale)
rem in7 log5      : (bumper destro)
rem
rem out0 Fet0     : servo
rem out1 Fet1     : sound/rel?-mis-batt
rem out2 Fet2     : led-lsb
rem out3 Fet3     : led-msb
rem out4 LM298    : motor A+
rem out5 LM298    : motor A-
rem out6 LM298    : motor B+
rem out7 LM298    : motor B-
rem
rem V=volatile P=permanent
rem
rem b0 (P) t1 toggling bit
rem b1 (V) anaA / bumper
rem b2 (V) anaB / bumper
rem b3     *counter1 (sr function)
rem b4 (P) behaviour
rem b5     *math accumulator (eyes function)
rem b6 ( ) not used
rem b7 (V) loop counter
rem b8 (P) bump counter
rem b9     pinout accumulator
rem b10(V) == b1
rem b11(V) == b2
rem b12    *math accumulator (speculatrix function)
rem b13(P) buridano's flag
rem w5 (v) random escape (b10:b11)
```

```
rem ****************************
rem Symbol Table
rem ****************************
rem
      symbol t1 = b0
      symbol LDRs = b1
      symbol LDRd = b2
      symbol flag = b3
      symbol behaviour = b4
      symbol specula = b8
      symbol osci = b13

      symbol toll = 50
      symbol escounter = 12


rem ****************************
rem MAIN PROGRAM
rem ****************************
rem

      setfreq m4 ;            // frequenza a 4 MHz
      pause 5000 ;            // pausa dopo l'accensione
      gosub clearinputs ;     // predispongo uno stato conosciuto

rem ***** MAIN PROGRAM visione e tatto ****************************

main:
      debug
      random w5

      if behaviour = 2 and b7 <= 50 then gosub dormi

      if pin2=1 and pin6=0 and pin7=1 then ;           // no-bumper
            gosub eyes ;      // predil?go la vista
      endif
      gosub bump ;      // e poi il tatto

      gosub escape ;    // verifico se sono in stallo

goto main
end


rem ****************************
rem BUMP FUNCTIONs
rem ****************************
rem

bump:
      if pin6 = 1 then ;            // urto frontale
            gosub alt:pause 1000
            gosub indietro:pause 1000
            if LDRs = 1 then
                  gosub girasinistra:pause 1000 ; // precedentemente a destra
            else
                  gosub giradestra:pause 1000 ;   // precedentemente a sinistra
            endif
```

```
      elseif pin2 = 0 and pin7 = 0 then ;      // sto in un tunnel
            gosub indietro
            let LDRs = 1
            pause 250

      elseif pin2 = 0 then ;        // ostacolo a sinistra
            gosub giradestra ;      // ma involont. inibisco if-pin7
            let LDRs = 1;           // marco la svolta a desta
            pause 250
            inc specula

      elseif pin7 = 0 then ;        // ostacolo a destra
            gosub girasinistra
            let LDRs = 0;                 // marco la svolta a sinistra
            pause 250
            inc specula

      else
            readoutputs b9
            if b9 != %00101000  then gosub avanti ;  // non sta camminando
      endif
return


rem ****************************
rem EYEs FUNCTIONs
rem ****************************
rem

eyes:
      readadc 0,LDRs ;        // read value into b1
      readadc 1,LDRd ;        // read value into b2

      gosub lux_meter

      rem *** La luce ? molto bassa. Faccio un giro di ricognizione.
      if LDRs < 100 and LDRd < 100 and specula >= 5 and behaviour = 1 then
            for b7 = 1 to 10
                  gosub giradestra : pause 300 ; si pu? alluppare
                  ; gosub bump
                  if LDRs > 100 or LDRd > 100 then exit
            next b7
            let specula = 0 ; azzero il contatore di svolte

      rem *** La luce ? troppo forte e scappo diventando fotofobico.
      elseif LDRs > 250 or LDRd > 250 then
            let behaviour = 2
            gosub alt
            gosub suono1
            gosub speculatrix
      else
            rem agisco da fototropico
            if behaviour = 1 then ;
                  if LDRs > LDRd then
                        let b5 = LDRs - LDRd
                        if b5 => toll then gosub girasinistra : pause 100
                  endif
                  if LDRs < LDRd then
                        let b5 = LDRd - LDRs
                        if b5 => toll then gosub giradestra : pause 100
```

```
            endif
      else
            rem agisco da fotofobico
            if LDRs > LDRd then
                  let b5 = LDRs - LDRd
                  if b5 => toll then gosub giradestra : pause 100
            endif
            if LDRs < LDRd then
                  let b5 = LDRd - LDRs
                  if b5 => toll then gosub girasinistra : pause 100
            endif
      endif

endif

return

rem ***** attiva i due led vu-meter *********************************
lux_meter:
      let b7 = LDRs+LDRd/2
            if b7 < 64 then low 3 : low 2 : endif                ; 00
            if b7 > 64 and b7 < 128 then low 3 : high 2 :endif   ; 01
            if b7 > 128 and b7 < 192 then high 3 : low 2 : endif ; 10
            if b7 > 192 then high 3 : high 2 : endif             ; 11
return

rem ***** la luce ? sufficientemente bassa per dormire **************
dormi:
      gosub alt ; // si ferma al buio
      do
            readadc 0,LDRs ;        // read value into b1
            readadc 1,LDRd ;        // read value into b2
            gosub lux_meter
            if pin6 = 1 then exit ;       // se toccato svegliati
            ; if timer = 8ore then exit
      loop while b7 < 64 ;                // se c'? luce svegliati
      let behaviour = 1 ;                 // ritorna fototropico
      gosub suono2
      gosub avanti
return


rem ****************************
rem ESCAPE FUNCTION
rem ****************************
rem

escape:
      if flag = 1 then ;            // c'? stata una oscillazione. Le sommo!
            inc osci
            let flag = 0
      endif

      if osci > escounter then ;          // n/2 oscill -> evado dallo stallo
            let w5 = b10 * 10 + 2000
            gosub indietro: pause 2000
            gosub giradestra
            pause w5
            let osci = 0
            let flag = 0
```

```
      endif
return


rem ****************************
rem MOTOR CONTROL FUNCTIONs
rem ****************************
rem

alt:
      low 4, 5, 6, 7
return

avanti:
      low 5,7,4,6
      high 4,6
return

indietro:
      low 4,6,5,7
      high 5,7
return

girasinistra:
      low 4,7,5,6
      high 5,6
      if t1 = 0 then let flag = 0 : endif ;     // precedente girasinistra
      if t1 = 1 then let flag = 1 : endif ;     // ultimo giradestra
      let t1 = 0 ;                              // marco come girasinistra
return

giradestra:
      low 6,5,7,4
      high 7,4
      if t1 = 1 then let flag = 0 : endif
      if t1 = 0 then let flag = 1 : endif
      let t1 = 1
return


rem ****************************
rem HEAD FUNCTIONs
rem ****************************
rem cw=0,35 cent=0,117 ccw=0,200

no_answer:
      for b7=1 to 3
            servopos 0,137
            pause 100
            servopos 0,97
            pause 100
      next b7
return

speculatrix:
      gosub servoccw ;        // sinistra
            readadc 0,b10 ;         // read value into b1
            readadc 1,b11 ;         // read value into b2

            let b5 = b10+b11/2
```

```
            gosub lm

      gosub servocw ;         // destra
            readadc 0,b10 ;         // read value into b1
            readadc 1,b11 ;         // read value into b2

            let b12 = b10+b11/2
            gosub lm2

      gosub servocenter

      if b5 < b12 then ;            // cerco un luogo in ombra
            gosub indietro : pause 1000
            gosub giradestra : pause 2000
            gosub avanti : pause 2000
      else if b5 > b12 then ;       // cerco un luogo in ombra
            gosub indietro : pause 1000
            gosub girasinistra : pause 2000
            gosub avanti : pause 2000
      else
            gosub indietro : pause 3000
            gosub giradestra : pause 2500 ; // dietrofront
      endif
return

lm:
      if b5 < 64 then low 3 : low 2 : endif                 ; 00
      if b5 > 64 and b5 < 128 then low 3 : high 2 : endif   ; 01
      if b5 > 128 and b5 < 192 then high 3 : low 2 : endif  ; 10
      if b5 > 192 then high 3 : high 2 : endif              ; 11
return

lm2:
      if b12 < 64 then low 3 : low 2 : endif                ; 00
      if b12 > 64 and b12 < 128 then low 3 : high 2 : endif; 01
      if b12 > 128 and b12 < 192 then high 3 : low 2 : endif; 10
      if b12 > 192 then high 3 : high 2 : endif             ; 11
return


rem ****************************
rem Futaba S3003 Servo FUNCTIONs
rem ****************************
rem cw=0,35 cent=0,117 ccw=0,200

servoinit:
      servo 0,122
      pause 800
return

servocenter:
      servopos 0,122
      pause 800
return

servocw:
      servopos 0,40
      pause 800
return
```

```
servoccw:
      servopos 0,208
      pause 800
return


rem ****************************
rem SOUND FUNCTIONs - 0/127
rem ****************************

suono0: ;               // boot
      sound 1, (80,20)
      sound 1, (110,30)
      pause 100
      sound 1, (80,20)
      sound 1, (110,30)
      pause 100
      sound 1, (80,20)
      sound 1, (110,60)
      low 1 ;     // anti classe A
return

suono1: ;               // troppa luce
      for b13 = 100 to 126
            sound 1, (b13,1)
      next b13
      pause 100
      for b13 = 100 to 126
            sound 1, (b13,1)
      next b13
      pause 300
      sound 1, (30,50)
      low 1 ;     anti classe A
      pause 3000
return

suono2: ;               // batteria scarica
      sound 1, (80,200)
return

scala:
      sound 1, (25,50) ; DO4 262
      sound 1, (36,50) ; RE 294
      sound 1, (46,50) ; MI 330
      sound 1, (51,50) ; FA 349
      sound 1, (59,50) ; SOL 392
      sound 1, (67,50) ; LA 440
      sound 1, (73,50) ; SI 494
      sound 1, (77,50) ; DO5 523
return


rem ****************************
rem BATTERY FUNCTIONs
rem ****************************

battlev:
      gosub mis_on ;               // scambio su batteria
      readadc 2,b13
      gosub mis_off ;              // scambio su bumper
```

```
      if b13 < 127 then ;           // equivale a Vcc/2
            for b7 = 1 to 10
                  gosub alt
                  sleep 230 ;       // aspetta 4 min
                  gosub suono2
            next b7
            let pins = %11111101 ;  // spegne definitivamente motori e led
            end ;                   // ibernazione
      endif
return

mis_on:
return

mis_off:
return


rem ****************************
rem RESET FUNCTIONs
rem ****************************
rem

clearinputs:
      let pins = %00000000 ;  // outputs a zero
      let behaviour = 1 ;     // parte fototropico (2=fotofobico)
      let specula = 0 ;       // abilito la 'speculatrix' dopo N-bumping
      let osci = 0 ;          // counter oscillazioni
      let flag = 0
      gosub servoinit : gosub no_answer : gosub servocenter
      gosub suono0
return
```