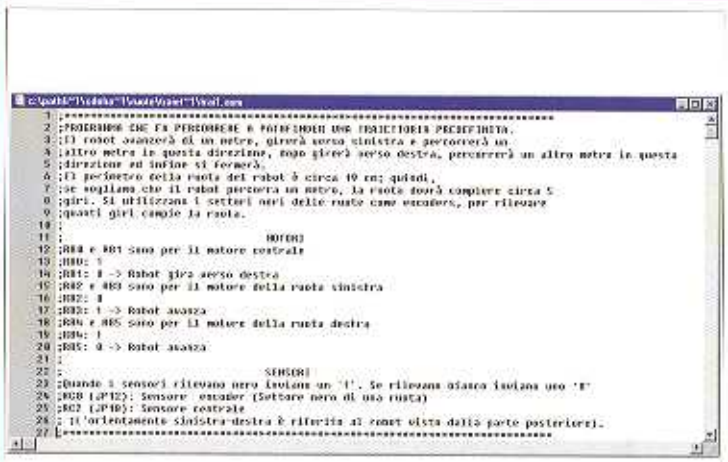
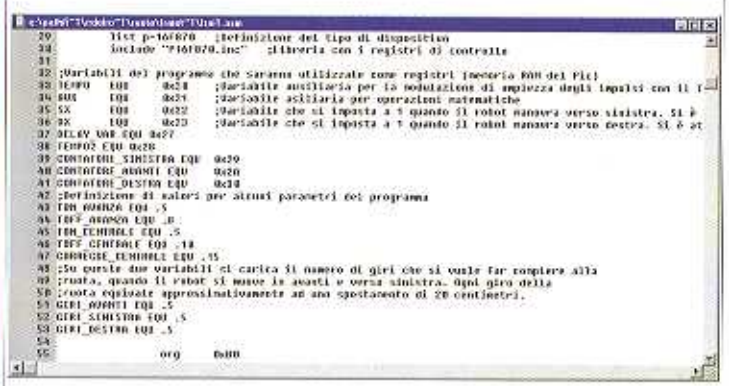
Modo ruote:
Traiettorie



```
;*****************************************************************
;PROGRAMMA CHE FA PERCORRERE A POLYBOT UNA TRAIETTORIA PREDEFINITA.
;IL robot avanzerà di un metro, girerà verso sinistra e percorrerà un
;altro metro in questa direzione, dopo girerà verso destra, percorrerà un altro metro in questa
;direzione ed infine si fermerà.
;Il perimetro della ruota del robot è circa 40 cm; quindi,
;se vogliamo che il robot percorra un metro, la ruota dovrà compiere circa 5
;giri. Si utilizzano i settori neri delle ruote come encoders, per rilevare
;quanti giri compie la ruota.
;
                        MOTORI
;RB4 e RB1 sono per il motore centrale
;RB4: 1
;RB1: 0 -> Robot gira verso destra
;RA2 e RB3 sono per il motore della ruota sinistra
;RB2: 0
;RB2: 1 -> Robot avanza
;RB4 e RB5 sono per il motore della ruota destra
;RB4: 1
;RB5: 0 -> Robot avanza
;
                        SENSORI
;Quando i sensori rilevano nero inviano un '1'. Se rilevano bianco inviano uno '0'
;RB18 (JP12): Sensore  encoder (Settare nero di una ruota)
;RC2 (JP18): Sensore centrale
; Il'orientamento sinistra-destra è riferito al robot vista dalla parte posteriore).
;*****************************************************************
```

Nella cartella Traiettorie, all'interno della directory Ruote, troviamo l'ultimo programma di esempio di Pathfinder in modo ruote. Questo programma mostra come possiamo programmare il robot per largli realizzare traiettorie predefinite. Per realizzare programmi di traiettorie abbiamo bisogno di sapere quanta distanza ha percorso il robot. Utilizzeremo un sensore ottico CNY70 montato nella parte interna delle ruote. Questo sensore rileverà il passaggio della striscia nera presente sugli adesivi incollati all'interno delle ruote. Grazie a esso, controlleremo la rotazione delle ruote e di conseguenza l'avanzamento che realizza il robot.

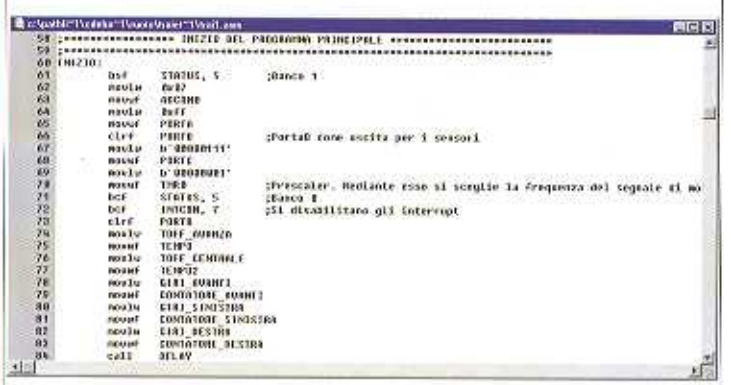


```
        list p=16F870   ;Definizione del tipo di dispositivo
        include "P16F870.inc"   ;Libreria con i registri di controllo

;Variabili del programma che saranno utilizzate come registri (memoria RAM del Pic)
TEMPO   EQU     0x28    ;Variabile ausiliaria per la modulazione di ampiezza degli impulsi con il T
AUX     EQU     0x29    ;Variabile ausiliaria per operazioni matematiche
SX      EQU     0x22    ;Variabile che si imposta a 1 quando il robot manovra verso sinistra. Si è
DX      EQU     0x23    ;Variabile che si imposta a 1 quando il robot manovra verso destra. Si è at
DELAY_UNO EQU 0x27
TEMPO2 EQU 0x26
CONTATORE_SINISTRA EQU  0x25
CONTATORE_AVANTI EQU    0x24
CONTATORE_DESTRA EQU    0x18
;Definizione di valori per alcuni parametri del programma
TEMP_AVANZA EQU .5
TOFF_AVANZA EQU .8
TEMP_CENTRALE EQU .5
TOFF_CENTRALE EQU .10
CORREGGE_CENTRALE EQU .15
;Su queste due variabili ci carica il numero di giri che si vuole far compiere alla
;ruota, quando il robot si muove in avanti e verso sinistra. Ogni giro della
;ruota equivale approssimativamente ad uno spostamento di 20 centimetri.
GIRI_AVANTI EQU .5
GIRI_SINISTRA EQU .5
GIRI_DESTRA EQU .5

                org     0x00
```

Per provare l'esercizio sono necessari solamente due sensori ottici. Il sensore ottico che rileva l'encoder delle ruote sarà collegato su JP12 della scheda di controllo, e il sensore ottico per il controllo di rotazione delle ruote anteriori rimarrà collegato su JP18, come per il resto degli esercizi che abbiamo realizzato. Questo programma realizza una traiettoria definita in cui si avanza di un metro in avanti e uno a sinistra, e infine un altro a destra. Nell'intestazione del programma disponiamo di alcune variabili che possiamo modificare per cambiare la distanza percorsa dal robot in ogni direzione.



```
;****************** INIZIO DEL PROGRAMMA PRINCIPALE ********************
;************************************************************************
INIZIO:
        bsf     STATUS, 5       ;Banco 1
        movlw   0x07
        movwf   ADCON0
        movlw   0xff
        movwf   PORTB
        clrf    PORTB           ;PortaB come uscita per i sensori
        movlw   b'00000011'
        movwf   PORTE
        movlw   b'00000001'
        movwf   TMR0            ;Prescaler. Mediante esso si sceglie la frequenza del segnale di mo
        bcf     STATUS, 5       ;Banco 0
        bcf     INTCON, 7       ;Si disabilitano gli interrupt
        clrf    PORTB
        movlw   TOFF_AVANZA
        movwf   TEMPO
        movlw   TOFF_CENTRALE
        movwf   TEMPO2
        movlw   GIRI_AVANTI
        movwf   CONTATORE_AVANTI
        movlw   GIRI_SINISTRA
        movwf   CONTATORE_SINISTRA
        movlw   GIRI_DESTRA
        movwf   CONTATORE_DESTRA
        call    DELAY
```

All'inizio del programma si realizzano le configurazioni dei pin di ingresso e uscita. In questo caso dobbiamo leggere lo stato dei due sensori ottici e inviare il segnale di controllo ai motori collegati sulla porta B. Si gestisce anche il Timer 0 con il prescaler, dato che il controllo della velocità di avanzamento del robot si realizza mediante modulazione di ampiezza degli impulsi, con le stesse funzioni che sono già state utilizzate nel resto degli esercizi di controllo di Pathfinder in modo ruote.

```
c:\pathfi~1\codice~1\ruote\traiet~1\trai1.asm
 95 ;*****************************************************************************
 96 ;Ciclo principale del programma
 97 CICLO:  bcf     PORTB, 0 ;Si ferma il motore centrale
 98         bcf     PORTB, 1
 99         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
100         btfsc   PORTC, 0
101         goto    CICLO
102 CICLO2: bcf     PORTB, 0 ;Si ferma il motore centrale
103         bcf     PORTB, 1
104         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
105         btfss   PORTC, 0
106         goto    CICLO2
107         decfsz  CONTATORE_AVANTI,1
108         goto    CICLO
109 CENTRALE_SU_BIANCO_SX:
110         call AVANZA_CENTRALE_SINISTRA
111         btfss   PORTC, 2
112         goto    CENTRALE_SU_BIANCO_SX
113 SU_NERO_SX:
114         call AVANZA_CENTRALE_SINISTRA
115         btfsc   PORTC, 2
116         goto    SU_NERO_SX
117 CICLO3: bcf     PORTB, 0 ;Si ferma il motore centrale
118         bcf     PORTB, 1
119         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
120         btfsc   PORTC, 0
121         goto    CICLO3
```

Qui comincia il ciclo principale del programma. In esso possiamo vedere che il robot inizia a muoversi con le ruote dritte fino a percorrere la distanza di un metro, che viene definita tramite la variabile CONTATORE_AVANTI. Dopo aver percorso un metro il robot girerà le ruote verso sinistra, e percorrerà un altro metro in questa direzione. In questo caso la distanza che percorre verso sinistra si controlla con il valore della variabile CONTATORE_SINISTRA.

```
c:\pathfi~1\codice~1\ruote\traiet~1\trai1.asm
122 CICLO4: bcf     PORTB, 0 ;Si ferma il motore centrale
123         bcf     PORTB, 1
124         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
125         btfss   PORTC, 0
126         goto    CICLO4
127         decfsz  CONTATORE_SINISTRA,1
128         goto    CICLO3
129 CENTRALE_SU_BIANCO_DX:
130         call AVANZA_CENTRALE_DESTRA
131         btfss   PORTC, 2
132         goto    CENTRALE_SU_BIANCO_DX
133 SU_NERO_DX:
134         call AVANZA_CENTRALE_DESTRA
135         btfsc   PORTC, 2
136         goto    SU_NERO_DX
137 CICLO5: bcf     PORTB, 0 ;Si ferma il motore centrale
138         bcf     PORTB, 1
139         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
140         btfsc   PORTC, 0
141         goto    CICLO5
142 CICLO6: bcf     PORTB, 0 ;Si ferma il motore centrale
143         bcf     PORTB, 1
144         call    MOTORI_TRAZIONE ;Il robot comincia a muoversi
145         btfss   PORTC, 0
146         goto    CICLO6
147         decfsz  CONTATORE_DESTRA,1
148         goto    CICLO5
149         clrf    PORTB
150 FINE:   goto    FINE
```

Quando il robot avrà percorso un altro metro verso sinistra girerà completamente verso destra per terminare il programma, percorrendo un altro metro in questa direzione. In questo caso il controllo dell'avanzamento verso destra si realizza tramite la variabile CONTATORE_DESTRA. Il valore di queste tre variabili CONTATORE viene inizializzato tramite le costanti GIRI_AVANTI, GIRI_SINISTRA e GIRI_DESTRA, che si trovano nell'intestazione del programma. Ogni unità di valore contenuta in una delle variabili equivale a un percorso di 20 cm nella direzione corrispondente.

Il file che contiene il codice sorgente si chiama trai1.asm e il programma esadecimale trai1.hex. Scriveremo questo file sulla scheda di memoria e la inseriremo sul robot per iniziare l'esecuzione del programma. I sensori ottici inseriti nel lato interno delle ruote ci forniranno il controllo sulla distanza che percorre il robot, quindi potremo realizzare applicazioni in cui il robot segue delle traiettorie definite e assolve compiti specifici.

Modo zampe: Esapodo (I)

```
c:\pathfi~1\cdelro~1\zampe\esap1.asm
  1         list p=16F876
  2
  3         include "P16F876.inc"
  4
  5 CONTATORE       EQU     0x20
  6 TEMP            EQU     0x30
  7 v_STATUS        EQU     0x25
  8
  9 ;RB2 e RB3 -> ZAMPA1 (JP4) (sinistra) con Sensore RC0 (JP12)
 10 ;RB4 e RB5 -> ZAMPA2 (JP2) (destra) con Sensore RC1 (JP17)
 11 ;RB0 e RB1 -> ZAMPA3 (JP3) (Centrale) con Sensore RC2 (JP18)
 12 ;Quando i sensori rilevano nero inviano un '1'. Se rilevano bianco inviano uno '0'
 13 ; 1 -> NERO
 14 ; 0 -> BIANCO
 15 ;RA4: Finecorsa sinistro (JP7)
 16 ;RA2: Finecorsa destro (JP8)
 17
 18 ;STATI IN CUI PASSA IL ROBOT NEL SUO MOVIMENTO
 19 ;STATUS 1
 20 ;       OFF_ZAMPA1
 21 ;       OFF_ZAMPA2
 22 ;       DX_ZAMPA3
 23 ; Sensore RC2 su bianco
 24
 25 ;STATUS 2
 26 ;       OFF_ZAMPA1
 27 ;       OFF_ZAMPA2
 28 ;       DX_ZAMPA3
 29 ; Sensore RC2 su nero
```

Inizieremo l'analisi dei tre programmi per il controllo di Pathfinder in modo esapodo, che si trovano sotto la directory Zampe del secondo CD-ROM. Per provare questi esercizi Pathfinder deve essere configurato con le zampe montate in modo esapodo. Dobbiamo collegare il sensore ottico con funzione di encoder della ruota destra sul connettore JP17 della scheda di controllo, il sensore ottico con funzione di encoder della ruota sinistra sul connettore JP12 e il sensore ottico di controllo delle sei zampe centrali con il connettore JP18. Per realizzare questi collegamenti dobbiamo scollegare i sensori ottici anteriori del robot che si utilizzano solamente nella configurazione con le ruote.

```
c:\pathfi~1\cdelro~1\zampe\esap1.asm
 91 ;MACRO UTILIZZATE NEL PROGRAMMA
 92 ;^^^^^^^^^^^^^^
 93 SX_ZAMPA3:      macro
 94                 bcf     PORTB, 0
 95                 bsf     PORTB, 1
 96                 ENDM
 97 DX_ZAMPA3:      macro
 98                 bsf     PORTB, 0
 99                 bcf     PORTB, 1
100                 ENDM
101 OFF_ZAMPA3:     macro
102                 bcf     PORTB, 0
103                 bcf     PORTB, 1
104                 ENDM
105 ;^^^^^^^^^^^^
106 AV_ZAMPA2:      macro
107                 bsf     PORTB, 5
108                 bcf     PORTB, 4
109                 ENDM
110 DX_ZAMPA2:      macro
111                 bcf     PORTB, 5
112                 bsf     PORTB, 4
113                 ENDM
114 OFF_ZAMPA2:     macro
115                 bcf     PORTB, 5
116                 bcf     PORTB, 4
117                 ENDM
118 ;^^^^^^^^^^^^
119 AV_ZAMPA1:      macro
```

Il primo programma che analizzeremo sarà esap1.asm. Questo programma permetterà al robot di camminare in avanti in modo esapodo. Per facilitare la lettura del programma, all'inizio dello stesso sono state generate nuove istruzioni (mediante macro) che serviranno per fermare ognuna delle zampe o attivarle in un senso o nell'altro. Osservando il robot dalla parte posteriore, chiameremo ZAMPA1 la zampa sinistra, ZAMPA2 quella destra e ZAMPA3 quella centrale. Le zampe sinistra e destra possono realizzare movimenti di avanzamento o retromarcia, e la zampa centrale è quella che si muove verso i lati per permettere l'avanzamento e la rotazione del robot.

```
c:\pathfi~1\cdelro~1\zampe\esap1.asm
135 ;*****************************************************************
136 ;Inizio del programma principale
137 INIZIO:
138         bcf     STATUS, RP1
139         bsf     STATUS, RP0
140         movlw b'00000111'
141         movwf ADCON0
142         movlw   b'00001111'
143         movwf   PORTA
144         clrf    PORTB
145         movlw   b'00000111'
146         movwf   PORTC
147         bcf     STATUS, RP0
148         clrf    PORTB
149         movlw   b'00100000'
150         movwf   PORTC
151
152 ;Routine di inizializzazione che si fa prima di iniziare il movimento. Tutti
153 ;i sensori stanno vedendo il settore nero, cioè sono in una zona corretta
154 INIZ:
155         call    LEGGERE
156         movf    v_STATUS, W
157         xorlw   .1
158         btfsc   STATUS, 2
159         goto    AV_GIRA_DX
160         movf    v_STATUS, W
161         xorlw   .2
162         btfsc   STATUS, 2
163         goto    AV_GIRA_DX2
```

All'inizio del programma configureremo la porta B come uscita, per poter gestire i motori. Prima di arrivare al ciclo principale troviamo una funzione di inizializzazione, che utilizza una variabile della memoria EEPROM del PIC.
Questa memoria EEPROM ha la caratteristica di non essere volatile, in quanto il dato inserito si mantiene anche quando si toglie l'alimentazione al robot.
Scriveremo in questa memoria l'ultimo stato delle zampe del robot prima di fermarsi, poiché ogni volta che si inizia il programma, il robot dovrà sapere in che posizione si trovano le zampe.

Modo zampe: Esapodo (I)

```
207 ;*****************************************************************************
208 ;Routine che fa camminare il robot in modo esapodo in avanti
209 CICLO:
210         OFF_ZAMPA1
211         OFF_ZAMPA2
212         OFF_ZAMPA3
213         movlw   .1
214         movwf   v_STATUS
215         call    SCRIVERE
216 ;STATUS 1
217 AV_GIRA_DX:
218         OFF_ZAMPA1
219         OFF_ZAMPA2
220         DX_ZAMPA3
221         btfss   PORTC,2
222         goto    AV_GIRA_DX
223         call    DELAY
224         btfss   PORTC,2
225         goto    AV_GIRA_DX
226         movlw   .2
227         movwf   v_STATUS
228         call    SCRIVERE
229 ;STATUS 2
230 AV_GIRA_DX2:
231         OFF_ZAMPA1
232         OFF_ZAMPA2
233         DX_ZAMPA3
234         btfsc   PORTC,2
235         goto    AV_GIRA_DX2
236         call    DELAY
```

Qui possiamo vedere l'inizio del ciclo principale del programma, che ha il compito di generare un movimento di avanzamento continuo del robot nella configurazione con le zampe. Per realizzare questo movimento di avanzamento, dobbiamo eseguire i passi già spiegati nella meccanica di Pathfinder in modo zampe. Ogni volta che eseguiamo un nuovo movimento delle zampe, lo stato attuale delle stesse viene scritto nella prima posizione della memoria EEPROM, per fare in modo che il robot all'inizio del programma possa conoscere l'ultimo stato all'interno della sequenza del movimento in cui si trovava.

```
386 ;*****************************************************
387 ;Routine per la scrittura nella memoria EEPROM.
388 ;Si scrive all'indirizzo 0 il valore della variabile v_STATUS
389 SCRIVERE:
390         bcf     STATUS, 5
391         bsf     STATUS, 6       ;Banco 2
392         movlw   0x00
393         movwf   EEADR
394         bcf     STATUS, 6
395         movf    v_STATUS, w
396         bsf     STATUS, 6
397         movwf   EEDATA
398         bsf     STATUS, 5
399         bcf     EECON1, EEPGD
400         bsf     EECON1, WREN
401         bcf     INTCON, GIE
402         movlw   0x55
403         movwf   EECON2
404         movlw   0xAA
405         movwf   EECON2
406         bsf     EECON1, WR
407         bcf     STATUS, 5
408         bcf     STATUS, 6
409         return
410 ;*****************************************************************************
411 ;LEGGERE: Routine che legge l'indirizzo 0 della memoria EEPROM e lascia il r
412 ;nella variabile v_STATUS
413 LEGGERE:
414         bsf     STATUS, 6
415         bcf     STATUS, 5       ;Banco 2
416         movlw   .0
417         movwf   EEADR
418         bsf     STATUS, 5       ;Banco 3
```

Nella parte finale del programma si trovano le routines che servono per scrivere e leggere la memoria EEPROM del microcontroller. Si tratta di due funzioni generiche che possiamo utilizzare in qualsiasi programma e che scrivono nella posizione 0 della memoria EEPROM il contenuto della variabile della memoria RAM, chiamata v_STATUS. A differenza della memoria RAM la cui scrittura è immediata, la memoria EEPROM impiega 10 ms per scrivere un dato, per questo ci dobbiamo assicurare che tra scrittura e scrittura trascorra un certo tempo. Nel caso specifico di questo programma, fra i diversi movimenti delle zampe del robot si supera sempre questo minimo tempo di scrittura.

In questa immagine possiamo vedere Pathfinder configurato in modo esapodo, mentre realizza un movimento di avanzamento con le zampe. Per fare questo dobbiamo scrivere il contenuto del file esap1.hex nella Smartcard del robot e inserirla nella scheda di alimentazione. Dopo qualche secondo il robot inizierà l'esecuzione del programma avanzando in modo esapodo. La prima volta che eseguiremo il programma, le zampe dovranno essere perfettamente centrate e i sensori ottici dovranno vedere la banda nera degli encoder, altrimenti i movimenti che eseguirà il robot non saranno corretti.

Modo zampe: Esapodo (II)

```
c:\pathli~1\codeto~1\zampe\esap2.asm
   1        list p=16F870
   2
   3        include "P16F870.inc"
   4
   5 CONTATORE       EQU     0x20
   6 TEMP            EQU     0x30
   7 v_STATUS        EQU     0x25
   8
   9 ;RB2 e RB3 -> ZAMPA1 (JPA) (sinistra) con Sensore RC0 (JP12)
  10 ;RB4 e RB5 -> ZAMPA2 (JP2) (destra) con Sensore RC1 (JP17)
  11 ;RB0 e RB1 -> ZAMPA3 (JP3) (Centrale) con Sensore RC2 (JP18)
  12 ;Quando i sensori rilevano nero inviano un '1'. Se rilevano bianco inviano uno '0'
  13 ; 1 -> NERO
  14 ; 0 -> BIANCO
  15 ;RA1: Finecorsa sinistro (JP7)
  16 ;RA2: Finecorsa destro (JP8)
  17
  18 ;STATI IN CUI PASSA IL ROBOT NEL SUO MOVIMENTO
  19 ;STATUS 1
  20 ;       OFF_ZAMPA1
  21 ;       OFF_ZAMPA2
  22 ;       DX_ZAMPA3
  23 ; Sensore RC2 su bianco
  24
```

Il secondo programma di gestione di Pathfinder in modo zampe si chiama esap2.asm e si trova sotto la directory Zampe del secondo CD-ROM. Grazie a questo programma Pathfinder potrà muoversi in modo esapodo nelle quattro direzioni. Il robot rileva l'attivazione dei finecorsa. Se i finecorsa sono disattivati il robot avanza, se sono attivati retrocede e se si attiva solo il finecorsa destro o sinistro, il robot gira verso destra o verso sinistra.

```
c:\pathli~1\codeto~1\zampe\esap2.asm
 207 ;*************************************************************************
 208 ;Inizializzazione
 209 CICLO:
 210         movf    PORTA, W
 211         movwf   TEMP
 212         movf    TEMP, W
 213         andlw   b'00000110'
 214         xorlw   b'00000000'
 215         btfsc   STATUS, Z
 216         goto    AVANTI
 217         movf    TEMP, W
 218         andlw   b'00000110'
 219         xorlw   b'00000010'
 220         btfsc   STATUS, Z
 221         goto    DESTRA
 222         movf    TEMP, W
 223         andlw   b'00000110'
 224         xorlw   b'00000100'
 225         btfsc   STATUS, Z
 226         goto    SINISTRA
 227         movf    TEMP, W
 228         andlw   b'00000110'
 229         xorlw   b'00000000'
 230         btfsc   STATUS, Z
 231         goto    INDIETRO
```

Questo è il ciclo principale del programma. In esso si testa lo stato dei due sensori tipo finecorsa montati sulla parte anteriore del robot. In funzione dello stato di questi sensori si accederà a una delle quattro routines di direzione di cui dispone il robot. Ogni routine farà eseguire al robot una diversa sequenza con le zampe che lo porterà ad avanzare, retrocedere o girare verso uno dei lati.

```
c:\pathli~1\codeto~1\zampe\esap2.asm
 233 ;*************************************************************************
 234 ;Routine da eseguire quando il robot cammina in avanti
 235 AVANTI:
 236         OFF_ZAMPA1
 237         OFF_ZAMPA2
 238         OFF_ZAMPA3
 239         movlw   .1
 240         movwf   v_STATUS
 241         call    SCRIVERE
 242 ;STATUS 1
 243 AV_GIRA_DX:
 244         OFF_ZAMPA1
 245         OFF_ZAMPA2
 246         DX_ZAMPA3
 247         btfss   PORTC,2
 248         goto    AV_GIRA_DX
 249         call    DELAY
 250         btfss   PORTC,2
 251         goto    AV_GIRA_DX
 252         movlw   .2
 253         movwf   v_STATUS
 254         call    SCRIVERE
 255 ;STATUS 2
 256 AV_GIRA_DX2:
 257         OFF_ZAMPA1
 258         OFF_ZAMPA2
```

La routine di avanzamento del robot, a cui si accede quando i due finecorsa sono disattivati, è simile a quella già utilizzata nel programma esap1.asm, che serviva a fare avanzare il robot in modo zampe. Questa routine di gestione di Pathfinder in modo esapodo utilizza le stesse istruzioni tipo macro generate nel programma esap1.asm, in modo che risulti più semplice eseguire le rotazioni e le fermate di ognuna delle zampe.

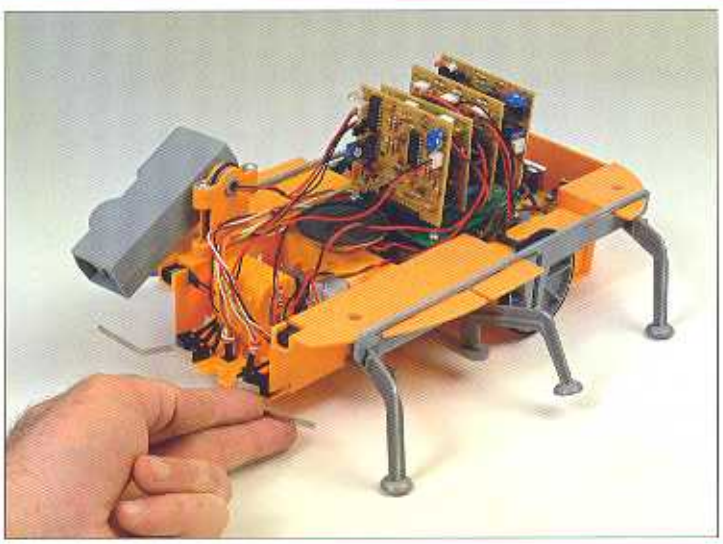
Modo zampe: Esapodo (II)

```
398 ;*****************************************************************************
399 ;Routine da eseguire quando il robot retrocede
400 INDIETRO:
401         OFF_ZAMPA1
402         OFF_ZAMPA2
403         OFF_ZAMPA3
404         movlw   .1
405         movwf   v_STATUS
406         call    SCRIVERE
407 ;STATUS 1
408 DX_GIRA_DX:
409         OFF_ZAMPA1
410         OFF_ZAMPA2
411         DX_ZAMPA3
412         btfss   PORTC,2
413         goto    DX_GIRA_DX
414         call    DELAY
415         btfss   PORTC,2
416         goto    DX_GIRA_DX
417         movlw   .2
418         movwf   v_STATUS
419         call    SCRIVERE
420 ;STATUS 2
421 DX_GIRA_DX2:
422         OFF_ZAMPA1
423         OFF_ZAMPA2
```

Questa funzione fa in modo che il robot retroceda, si attiva quando i finecorsa sono premuti.
Cosi come la routine di avanzamento, è una funzione generica che potremo utilizzare in altri programmi che progetteremo. Questa funzione di retromarcia realizza gli stessi passi di quella di avanzamento, però con la sequenza delle zampe basculanti invertita, l'unica variazione necessaria per fare in modo che il robot retroceda invece di avanzare.

```
565 DESTRA:
566         OFF_ZAMPA1
567         OFF_ZAMPA2
568         OFF_ZAMPA3
569         movlw   .1
570         movwf   v_STATUS
571         call    SCRIVERE
572 ;STATUS 1
573 DES_GIRA_DES:
574         OFF_ZAMPA1
575         OFF_ZAMPA2
576         DX_ZAMPA3
577         btfss   PORTC,2
578         goto    DES_GIRA_DES
579         call    DELAY
580         btfss   PORTC,2
581         goto    DES_GIRA_DES
582         movlw   .2
583         movwf   v_STATUS
584         call    SCRIVERE
585 ;STATUS 2
586 DES_GIRA_DES2:
587         OFF_ZAMPA1
588         OFF_ZAMPA2
```

La funzione DESTRA serve a fare in modo che il robot giri in modo zampe verso questa direzione, e la funzione SINISTRA realizza lo stesso lavoro però nell'altro senso.
A questa routine si arriva quando si attiva solamente uno dei due finecorsa posteriori. Il robot ruoterà sempre su se stesso, utilizzando le zampe e tornerà a riprendere la marcia in avanti quando cesserà l'attivazione del finecorsa.



Per provare questo programma dobbiamo scrivere il file esap2.hex sulla Smartcard del robot. Dobbiamo disporre della stessa configurazione dei sensori ottici, utilizzata nel programma precedente, esap1.asm.
I finecorsa devono essere collegati sulla scheda di interfaccia del robot.
Dopo aver alimentato il robot, attenderemo qualche secondo fino a quando il microcontroller leggerà il programma e inizierà a funzionare.
A partire da questo momento attiveremo i finecorsa per provare i diversi movimenti del robot in modo esapodo.

Modo zampe: Esapodo (III)

```
c:\pathfi~1\cdelro~1\zampe\esap3.asm
  1         list p=16f876
  2
  3         include "P16F876.inc"
  4
  5 CONTATORE       EQU     0x20
  6 AUX             EQU     0x21
  7 W_STATUS        EQU     0x25
  8 TEMP            EQU     0x30
  9 ;La variabile retromarcia indica quanti passi indietro farà il robot prima di girare a destra
 10 RETROMARCIA EQU .5
 11 ;le variabili gira_sinistra e gira_destra indicano quanti passi esegue il robot per girare ver
 12 ;o verso destra dopo una collisione.
 13 GIRA_SINISTRA EQU .4
 14 GIRA_DESTRA EQU .4
 15
 16 ;RB2 e RB3 -> ZAMPA1 (JP4) (sinistra) con Sensore RC0 (JP12)
 17 ;RB4 e RB5 -> ZAMPA2 (JP2) (destra) con Sensore RC1 (JP17)
 18 ;RB0 e RB1 -> ZAMPA3 (JP3) (Centrale) con Sensore RC2 (JP18)
 19 ;Quando i sensori rilevano nero inviano un '1'. Se rilevano bianco inviano uno '0'
 20 ;RA1: Finecorsa sinistro (JP7)
 21 ;RA2: Finecorsa destro (JP8)
 22
 23 ;STATI IN CUI PASSA IL ROBOT NEL SUO MOVIMENTO
 24 ;STATUS 1
 25 ;       OFF_ZAMPA1
 26 ;       OFF_ZAMPA2
 27 ;       DX_ZAMPA3
 28 ; Sensore RC2 su bianco
```

Vediamo l'ultimo esercizio di esempio della gestione di Pathfinder in modo esapodo. Si chiama esap3.asm e si trova nella stessa directory Zampe del programma precedente. Con questo programma, Pathfinder funziona come un esploratore, avanzerà e ogni volta che toccherà un ostacolo con i finecorsa anteriori, retrocederà e girerà a destra o a sinistra per schivare l'ostacolo e proseguire il suo percorso.

```
c:\pathfi~1\cdelro~1\zampe\esap3.asm
 95
 96 ;MACRO UTILIZZATE NEL PROGRAMMA
 97 ;-------------
 98 SX_ZAMPA3:      macro
 99                 bcf     PORTB, 0
100                 bsf     PORTB, 1
101                 ENDM
102 DX_ZAMPA3:      macro
103                 bsf     PORTB, 0
104                 bcf     PORTB, 1
105                 ENDM
106 OFF_ZAMPA3:     macro
107                 bcf     PORTB, 0
108                 bcf     PORTB, 1
109                 ENDM
110 ;-------------
111 AV_ZAMPA2:      macro
112                 bsf     PORTB, 5
113                 bcf     PORTB, 4
114                 ENDM
115 DX_ZAMPA2:      macro
116                 bcf     PORTB, 5
117                 bsf     PORTB, 4
118                 ENDM
119 OFF_ZAMPA2:     macro
120                 bcf     PORTB, 5
121                 bcf     PORTB, 4
122                 ENDM
123 ;-------------
```

La configurazione dei sensori ottici e dei finecorsa è la stessa di quella utilizzata nel precedente esercizio, esap2.asm; anche le configurazioni dei pin di ingresso e di uscita sono le stesse. Le macro che si trovano all'inizio del programma sono state generate per facilitare la gestione delle zampe del robot e sono identiche a quelle utilizzate nei due programmi precedenti.

```
c:\pathfi~1\cdelro~1\zampe\esap3.asm
211
212 ;*****************************************************************
213 ;Inizializzazione
214 CICLO:
215         movf    PORTA, W
216         movwf   TEMP
217         movf    TEMP, W
218         andlw   b'00000110'
219         xorlw   b'00000000'
220         btfsc   STATUS, Z
221         goto    AVANTI
222         movf    TEMP, W
223         andlw   b'00000110'
224         xorlw   b'00000010'
225         btfsc   STATUS, Z
226         goto    FINECORSA_SINISTRO
227         movf    TEMP, W
228         andlw   b'00000110'
229         xorlw   b'00000100'
230         btfsc   STATUS, Z
231         goto    FINECORSA_DESTRO
232
```

Questo è il ciclo principale del programma, la cui struttura è simile a quella degli altri esercizi che abbiamo già realizzato. Mediante una maschera si testano i finecorsa del robot e in funzione del loro stato si entra nelle diverse routines. Quando i due sensori sono disattivati il robot avanzerà, ma se qualcuno di essi si attiva, si passerà alla routine che farà retrocedere e girare a destra o a sinistra per schivare l'ostacolo.

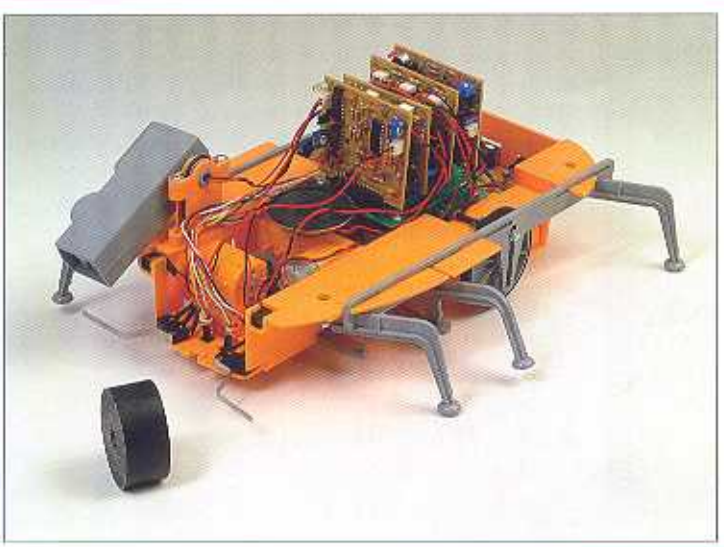
Modo zampe: Esapodo (III)

```
c:\pathfi~1\codeio~1\zampe\esap3.asm
232
233 FINECORSA_SINISTRO:
234         movlw   RETROMARCIA
235         movwf   AUX
236 DX_CICLO:
237         call    INDIETRO
238         decfsz AUX, 1
239         goto    DX_CICLO
240         movlw   GIRA_SINISTRA
241         movwf   AUX
242 DX_CICLO_SINISTRA:
243         call    SINISTRA
244         decfsz AUX, 1
245         goto    DX_CICLO_SINISTRA
246     goto CICLO
247
248 FINECORSA_DESTRO:
249         movlw   RETROMARCIA
250         movwf   AUX
251 SX_CICLO:
252         call    INDIETRO
253         decfsz AUX, 1
254         goto    SX_CICLO
255         movlw   GIRA_DESTRA
256         movwf   AUX
257 DX_CICLO_DESTRA:
258         call    DESTRA
259         decfsz AUX, 1
260         goto    DX_CICLO_DESTRA
261     goto CICLO
```

Queste sono le funzioni che realizza il robot quando si attiva il finecorsa sinistro o destro. Mediante la variabile RETRO_MARCIA, dichiarata all'inizio del programma, possiamo controllare quanti passi indietro vogliamo fare eseguire al robot prima di iniziare la rotazione a destra o a sinistra. Con le variabili GIRA_SINISTRA e GIRA_DESTRA controlliamo quanti passi esegue il robot quando gira in entrambe le direzioni. Questa funzione permetterà a Pathfinder di adattarsi a terreni diversi.

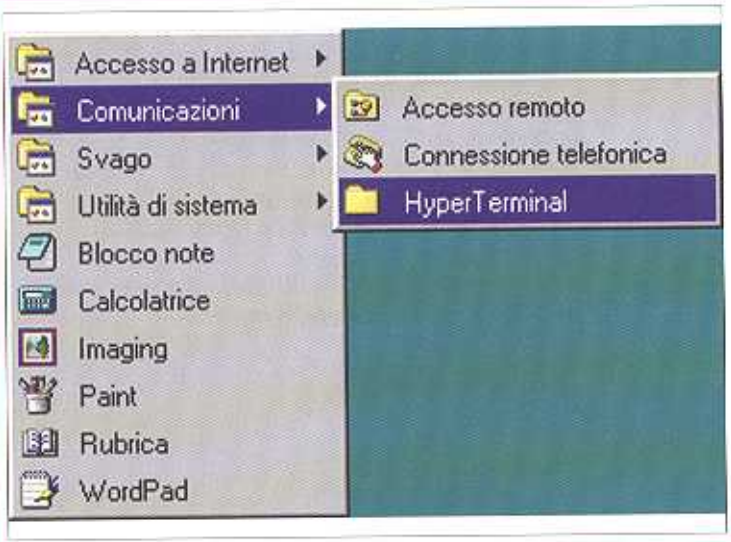
```
c:\pathfi~1\codeio~1\zampe\esap3.asm
263 ;*****************************************************************
264 ;Routine da eseguire quando il robot cammina in avanti
265 AVANTI:
266         OFF_ZAMPA1
267         OFF_ZAMPA2
268         OFF_ZAMPA3
269         movlw   .1
270         movwf   v_STATUS
271         call    SCRIVERE
272 ;STATUS 1
273 AV_GIRA_DX:
274         OFF_ZAMPA1
275         OFF_ZAMPA2
276         DX_ZAMPA3
277         btfss   PORTC,2
278         goto    AV_GIRA_DX
279         call    DELAY
280         btfss   PORTC,2
281         goto    AV_GIRA_DX
282         movlw   .2
283         movwf   v_STATUS
284         call    SCRIVERE
285 ;STATUS 2
286 AV_GIRA_DX2:
287         OFF_ZAMPA1
288         OFF_ZAMPA2
289         DX_ZAMPA3
290         btfsc   PORTC,2
291         goto    AV_GIRA_DX2
292         call    DELAY
```

Le funzioni di avanzamento, retromarcia e rotazione del robot a sinistra o a destra sono le stesse che abbiamo già utilizzato nel programma esap2.asm, dato che le sequenze da realizzare con le zampe per ottenere questi movimenti nel robot sono sempre le stesse. Possiamo utilizzare le stesse funzioni per qualsiasi altra applicazione per la quale vorremo programmare il robot.



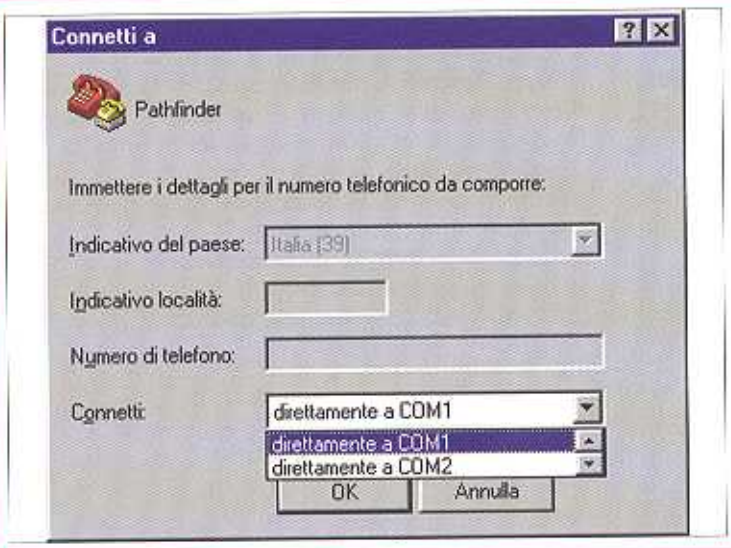
Per provare l'esercizio dobbiamo scrivere il programma esap3.hex sulla Smartcard e inserirla sulla scheda di alimentazione di Pathfinder. Dopo aver alimentato il robot, quest'ultimo inizierà la sua marcia e rileverà la presenza di ostacoli con i suoi finecorsa anteriori. Ogni volta che un finecorsa si attiva, il robot inizierà la sequenza per schivare l'ostacolo e continuare la sua marcia.

Controllo tramite il PC (I)

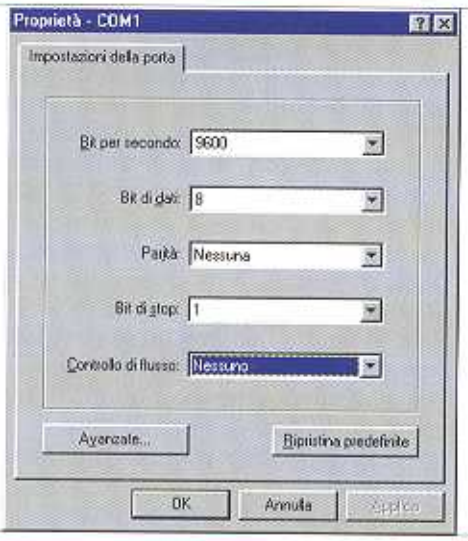


Oltre a programmare Pathfinder mediante la sua Smartcard per fargli realizzare applicazioni in modo autonomo, possiamo anche controllare i movimenti del robot tramite il PC. Per questo dovremo utilizzare la Smartcard con i programmi di controllo remoto contenuti sul secondo CD-ROM di Pathfinder, e inviare una serie di comandi tramite la porta seriale di un computer. Qualsiasi programma che gestisce la porta seriale può servire per controllare Pathfinder. Un comune programma di comunicazione è HyperTerminal, che è un'applicazione contenuta nel sistema operativo Windows. Per eseguire questo programma accederemo al menù Inizio, Programmi, Accessori, Comunicazioni e HyperTerminal.

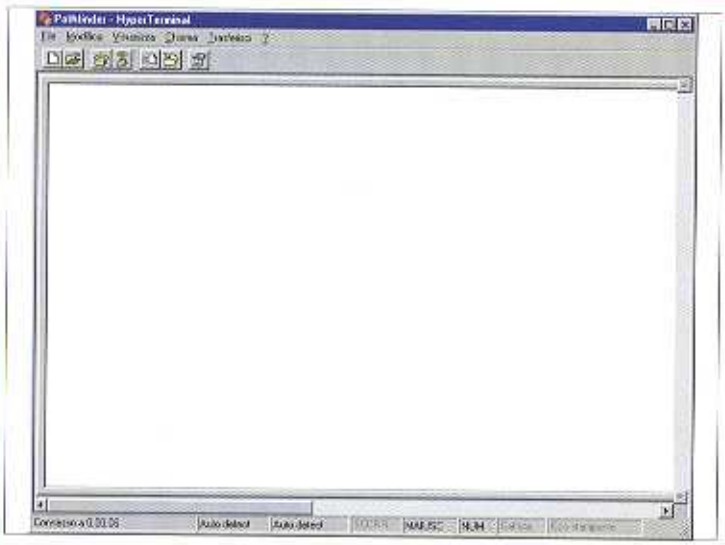
Quando eseguiamo il programma HyperTerminal.exe, ci verrà chiesto di dare un nome e scegliere un'icona per l'applicazione di comunicazione che stiamo per generare. Potremo dare un nome qualsiasi e scegliere un'icona qualsiasi. Dopo aver realizzato questa configurazione premeremo il pulsante OK.



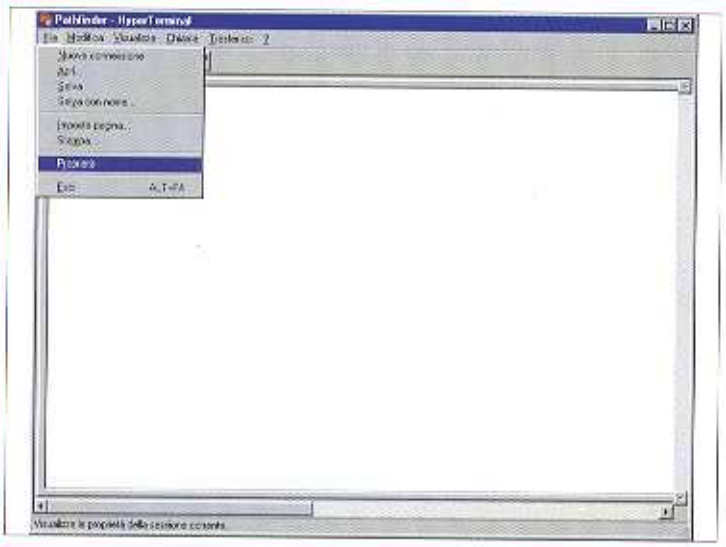
Nella videata successiva di HyperTerminal ci verrà chiesto di scegliere o la porta o il modem installato sul computer con il quale vogliamo stabilire una comunicazione. Dobbiamo scegliere l'opzione "Diretto a COM", per inviare i comandi direttamente a una delle porte seriali del computer. In funzione della porta seriale che abbiamo libera sul nostro PC e che vogliamo utilizzare per gestire Pathfinder, sceglieremo COM1, COM2, COM3 o COM4.



Dopo aver scelto la porta di comunicazione che abbiamo libera, HyperTerminal aprirà una nuova finestra, in cui ci viene chiesto di configurare il modo di comunicazione. Dobbiamo scegliere la velocità standard di 9.600 baude, 8 bit di dati, Nessuna Parità, 1 bit di stop e nessun controllo di flusso. Questa è la configurazione standard del programma di comunicazione di Pathfinder ed è necessario che sia la stessa utilizzata nel PC.



Il passo successivo costituisce l'ultima configurazione necessaria prima di poter iniziare a gestire la comunicazione con HyperTerminal. La videata mostrata nella figura è quella di controllo. Cliccheremo l'icona del telefono per iniziare la comunicazione. Quando la porta sarà aperta, sulla barra di programma posizionata in basso a sinistra apparirà la scritta "collegato". Per terminare la comunicazione selezioneremo l'icona che ha un telefono scollegato. Per inviare un dato sulla porta seriale dobbiamo solo aprire la comunicazione e premere un pulsante. Ogni pulsante premuto invierà il suo corrispondente codice ASCII sulla porta seriale, e sarà il modo di inviare comandi al robot.



Nel caso si volesse modificare la configurazione della porta seriale o un'altra funzione di HyperTerminal, dobbiamo accedere al menù File e scegliere l'opzione Proprietà. Proseguendo torneremo ad aprire la finestra di selezione della porta e di tipo di comunicazione. Prima di realizzare qualsiasi modifica sulla configurazione della porta, è indispensabile fermare la comunicazione attuale, cliccando sull'icona del telefono scollegato.