# ALCUNI ESEMPI PER RZB-1P

*Dopo aver introdotto le funzioni di base che ci permettono di programmare RZB-1p, passiamo alla scrittura di alcuni esempi.*

Nei due numeri precedenti abbiamo presentato la prima parte di codice che ci permette di sviluppare i firmware per il PIC del nostro robot. Prima di passare ai sorgenti veri e propri, però, spieghiamo come dovranno essere impiegati per compilare gli esempi proposti.

**COME COMPILARE I PROGRAMMI DI ESEMPIO ›››**

Nelle pagine che seguiranno è mostrata una serie di firmware di esempio realizzati a partire dalle funzioni e dalle definizioni elencate nei Workshop 71 e 72. **Per poter compilare correttamente tali esempi**, di conseguenza, **è indispensabile che all'interno del progetto siano presenti anche tutte le righe di codice necessarie a definire i valori simbolici e a implementare le procedure di base.** Dovrai quindi copiare il codice sorgente mostrato nei due fascicoli precedenti e aggiungere di seguito la funzione 'main', che implementa il 'comportamento' desiderato. Fatto ciò, potrai procedere con la compilazione del progetto, generando così il file binario da caricare sul PIC.

**UN FIRMWARE DI TEST DEI MOTORI ›››**

Il primo firmware che analizziamo è pensato per svolgere un test elementare che **verifichi il movimento dei motori e il funzionamento dei LED.** Questo programma potrà essere eseguito anche senza collegare la scheda sensori alla scheda PIC di RZB-1p. Il firmware in questione, che abbiamo già anticipato alla fine del Workshop 72, è fondamentalmente una sequenza temporizzata delle principali istruzioni di movimento del robot. Associeremo, inoltre, allo stato di attività dei motori l'accensione del LED verde, mentre allo stato di inattività l'accensione del LED rosso. La sequenza di operazioni prodotta dal programma è descritta nello schema sottostante.

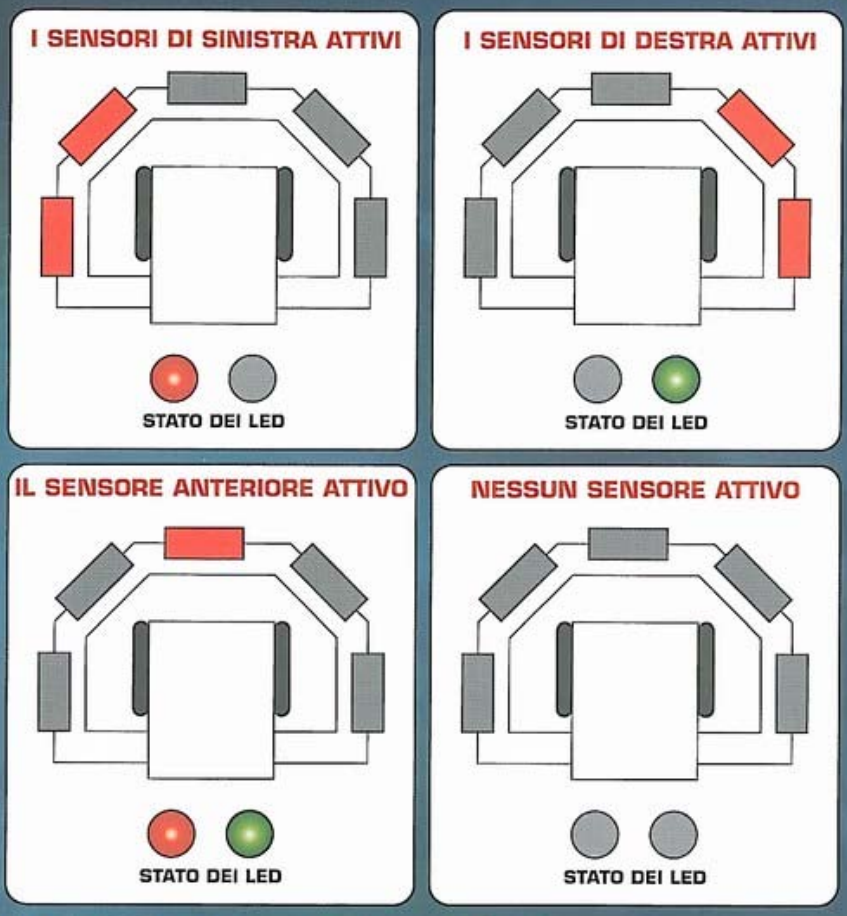

*Nello schema è mostrata la sequenza di movimenti che realizzeremo con la scrittura del nostro primo firmware di collaudo di RZB-1p. La sequenza sarà ripetuta in maniera ciclica.*

*Il firmware di test dei motori*

```
/* Firmware di test del movimento dei motori. Richiede le definizioni dei valori simbolici
e delle funzioni base del robot contenute nei fascicoli 71 e 72. */
int main()
{
     /* Inizializziamo il PIC del robot */
     InizializzaRobot();

    /* iniziamo il ciclo di funzionamento del firmware */
    /* ciclo while infinito di ripetizione  */
    while(1)
    {

     /* Accendiamo i motori */
     AccendiMotori(ENTRAMBI);

     /* Accendiamo il led verde e spegniamo quello rosso */
     LED_Verde(ACCESO);
     LED_Rosso(SPENTO);

     /* Impostiamo i motori del robot in configurazione di avanzamento */
     Avanza();
     /* Inseriamo un ritardo di  3 secondi (ossia 3000 msec) */
     Delay_ms(3000);


     /* Impostiamo il robot per ruotare verso destra */
     RuotaDestra();
     /* Inseriamo un ritardo di  2 secondi (ossia 2000 msec) */
     Delay_ms(2000);


     /* Impostiamo i motori del robot in configurazione di avanzamento */
     Avanza();
     /* Inseriamo un ritardo di  3 secondi (ossia 3000 msec) */
     Delay_ms(3000);


     /* Impostiamo il robot per ruotare verso sinistra */
     RuotaSinistra();
     /* Inseriamo un ritardo di  2 secondi (ossia 2000 msec) */
     Delay_ms(2000);


     /* Impostiamo i motori del robot in configurazione di avanzamento */
     Retrocedi();
     /* Inseriamo un ritardo di  5 secondi (ossia 5000 msec) */
     Delay_ms(5000);


     /* Spegniamo i motori per 3 secondi e riconfiguriamo i LED */
     SpegniMotori(ENTRAMBI);
     LED_Rosso(ACCESO);
     LED_Verde(SPENTO);
     Delay_ms(3000);
     }
}
```

**TESTIAMO I SENSORI DI RZB-1P ›››**

Il primo firmware di test ci è servito per testare il corretto interfacciamento della scheda PIC con i due servocomandi. Non abbiamo, però, ancora impiegato le funzioni di acquisizione dati dai sensori. Proprio queste procedure sono al centro di questo secondo esempio, con il quale **collauderemo le 'abilità' sensoriali del robot, illuminando i LED in funzione dello stato di attività dei sensori**. In particolare, il firmware attiverà il **LED rosso** nel caso in cui la scheda sensori rilevi un **ostacolo alla sinistra del robot** (sensori Ovest o NordOvest), mentre accenderà quello **verde** nel caso in cui vi sia un **ostacolo alla destra** (sensori Est o NordEst). Il **sensore Nord**, infine, accenderà **entrambi i LED** della scheda PIC (vedi schema a destra). Entrambi i motori rimarranno disattivati.




*Nel grafico è mostrato lo schema di funzionamento del secondo firmware che realizzeremo in questo Workshop, che opera attivando opportunamente i LED in risposta alle stimolazioni sensoriali provenienti dall'esterno (i sensori di sinistra accendono il LED rosso, quelli di destra il LED verde, mentre quello centrale entrambi).*

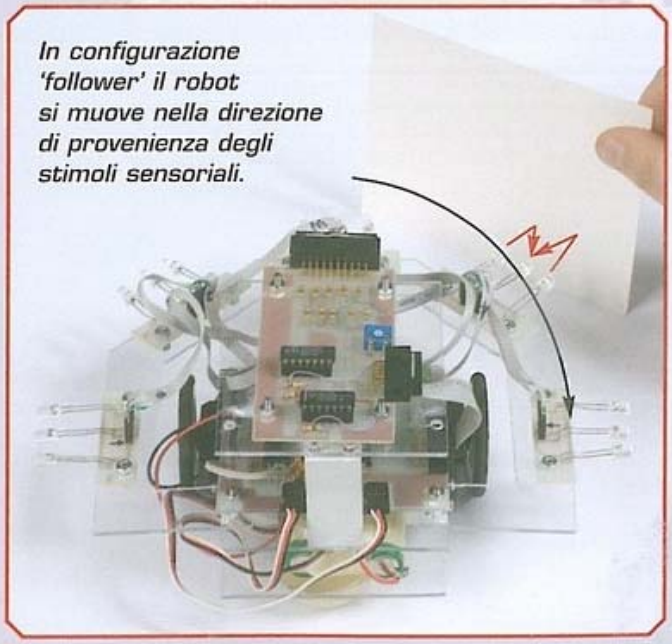
*Il firmware di test dei sensori*

```
/* Firmware di test del movimento dei motori. Richiede le definizioni dei valori simbolici
e delle funzioni base del robot contenute nei fascicoli 71 e 72. */
int main()
{
    /* Inizializziamo il PIC del robot */
    InizializzaRobot();

    /* iniziamo il ciclo di funzionamento del firmware */
    /* ciclo while infinito di ripetizione  */
    while(1)
    {
    /* Verifichiamo la presenza di ostacoli alla sinistra del robot */
    /* Se è rilevato un ostacolo dal sensore ovest o da quello nordovest */
    if(Ostacolo_A_Ovest() || Ostacolo_A_NordOvest())
    {
       /* accendiamo il LED rosso */
       LED_Rosso(ACCESO);
    }
    else
```

```
    {
        /* altrimenti spegniamo il LED rosso */
        LED_Rosso(SPENTO);
    }

    /* Se è rilevato un ostacolo dal sensore est o da quello nordest */
    if(Ostacolo_A_Est() || Ostacolo_A_NordEst())
    {
        /* accendiamo il LED verde */
        LED_Verde(ACCESO);
    }
    else
    {
        /* altrimenti spegniamo il LED verde */
        LED_Verde(SPENTO);
    }

    /* Se è rilevato un ostacolo dal sensore nord */
    if(Ostacolo_A_Nord())
    {
        /* accendiamo entrambi i LED della scheda PIC */
        LED_Verde(ACCESO);
        LED_Rosso(ACCESO);
    }
  } /* chiusura ciclo while */
}
```

## PROGRAMMIAMO RZB-1P COME 'FOLLOWER'

In quest'ultima parte del Workshop ci dedichiamo a realizzare un **firmware che trasformerà RZB-1p in un 'follower'**, ossia un 'inseguitore'. Ma entriamo più nel dettaglio, spiegando in cosa consiste questa nuova modalità di funzionamento. Come è facile intuire dal nome che abbiamo utilizzato, **il software in questione consente a RZB-1p di inseguire un elemento presente nei suoi paraggi,** purché collocato a una distanza tale da essere rilevato dai suoi sensori. Avremo, in sostanza, un comportamento 'complementare' a quello studiato nella versione originale del robot, nel quale la direzione di movimento sarà condizionata dai dati ambientali. La prima differenza rilevabile dipende dal fatto che, dovendo seguire un 'ostacolo', il robot non ruoterà più in direzione opposta alle rilevazioni sensoriali, ma seguirà gli stimoli. Partiamo dal presupposto, quindi, che la presenza di **uno stato alto su uno dei due sensori di sinistra (Ovest e NordOvest) imponga al robot di ruotare verso sinistra** e che **lo stato alto di almeno uno dei due sensori di destra porti il suo sistema di controllo a attivare una rotazione verso destra.** Nel caso in cui, infine, venga attivato **solo (e soltanto) il sensore anteriore, facciamo in modo di far avanzare il robot, per produrre l'inseguimento.** E se nessuno dei sensori risulta attivo? Possiamo, semplicemente, spegnere i motori e rimanere in attesa di un 'bersaglio'.



*In configurazione 'follower' il robot si muove nella direzione di provenienza degli stimoli sensoriali.*

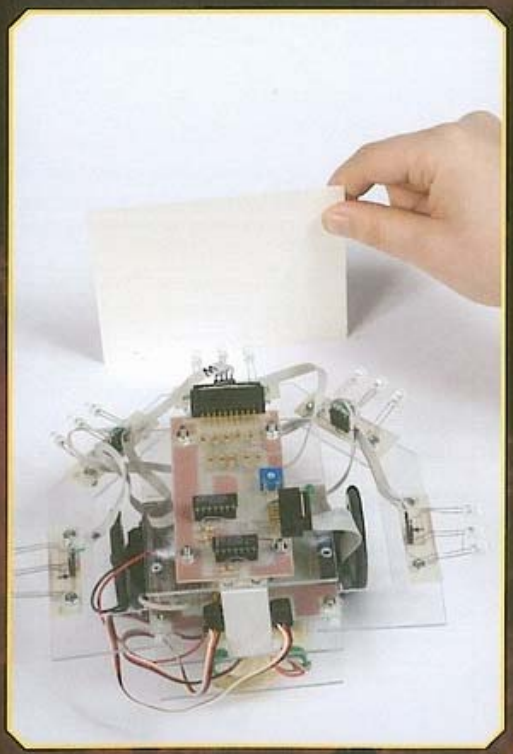
Il sorgente del 'follower'

```
/* Funzione main del firmware 'follower' di RZB-1p */
void main()
{
    /* Inizializziamo il PIC in modo da predisporlo al controllo del robot */
    InizializzaRobot();

    /* Attiviamo il LED rosso che identificherà la condizione di 'attesa'
       del robot */
    LED_Rosso(ACCESO);
    LED_Verde(SPENTO);

    /* ciclo di funzionamento del robot che viene ripetuto dal PIC */
    while(1)
    {

        /* se nessuno dei sensori rileva la presenza di un ostacolo,
            disattiva i motori */
        if (!(Ostacolo_A_Ovest() || Ostacolo_A_NordOvest() || Ostacolo_A_Nord()
      || Ostacolo_A_NordEst()  || Ostacolo_A_Est()))
        {
            SpegniMotori(ENTRAMBI);
            /* Accendiamo il LED rosso per segnalare lo stato di attesa */
            LED_Rosso(ACCESO);
            LED_Verde(SPENTO);
        }


        /* Verifico la presenza di un ostacolo esattamente frontale al robot
            (solo il sensore Nord attivo) */
        if (!Ostacolo_A_Ovest() && !Ostacolo_A_NordOvest() && Ostacolo_A_Nord()
     && !Ostacolo_A_NordEst()  && !Ostacolo_A_Est())
        {
            /* Se solo il sensore nord è attivo, facciamo avanzare il robot */
            AccendiMotori(ENTRAMBI);
            Avanza();
            /* Accendiamo il LED verde per segnalare lo stato di movimento */
            LED_Rosso(SPENTO);
            LED_Verde(ACCESO);
         }


        /* altrimenti, se è attivo ALMENO UNO dei sensori di sinistra e NESSUNO
            di quelli di destra */
        else if( (Ostacolo_A_Ovest() || Ostacolo_A_NordOvest()) &&
            !Ostacolo_A_NordEst() && !Ostacolo_A_Est())
        {
            /* Imponiamo la rotazione la del robot verso sinistra */
            AccendiMotori(ENTRAMBI);
            RuotaSinistra();
            /* Accendiamo il LED verde per segnalare lo stato di movimento */
            LED_Rosso(SPENTO);
            LED_Verde(ACCESO);
         }


        /* altrimenti, se è attivo ALMENO UNO dei sensori di destra e NESSUNO
            di quelli di sinistra */
        else if( (Ostacolo_A_Est() || Ostacolo_A_NordEst()) &&
    !Ostacolo_A_Ovest() && !Ostacolo_A_NordOvest())
```

```
        {
          /* Imponiamo la rotazione la del robot verso destra */
          AccendiMotori(ENTRAMBI);
          RuotaDestra();
          /* Accendiamo il LED verde per segnalare lo stato di movimento */
          LED_Rosso(SPENTO);
          LED_Verde(ACCESO);
        }


      /* Altrimenti, nel caso in cui vi siano casi di ambiguità
       (ossia siano attivi sia sensori di destra, sia sensori di sinistra
       imponiamo al robot di fermarsi in attesa che venga risolta l'ambiguità. */
         else
         {
           SpegniMotori(ENTRAMBI);
          /* Accendiamo il LED rosso per segnalare lo stato di attesa */
          LED_Rosso(ACCESO);
          LED_Verde(SPENTO);
         }

    } /* Termine del ciclo 'while' */
}
```

**IL COLLAUDO DI RZB-1P IN MODALITÀ FOLLOWER ›››**

Non ti resta che sperimentare il codice mostrato nelle pagine precedenti, **trascrivendolo** e **compilandolo** all'interno dell'ambiente **mikroC**. Come già detto all'inizio di questo articolo, ti ricordiamo che, affinché il firmware binario possa essere generato correttamente, è indispensabile che prima della funzione 'main' siano presenti tutte le funzioni e le definizioni contenute nei due fascicoli precedenti a quello corrente. Una volta ottenuto il file '.hex', non ti resta che caricarlo nella memoria istruzioni del PIC 16F628 alla base della scheda di controllo del robot, per poter poi testare il suo funzionamento. **Per eseguire il test ti consigliamo di utilizzare un cartoncino di colore chiaro.** Come accennato nel fascicolo 40, in cui abbiamo introdotto il funzionamento dei sensori di prossimità a infrarossi, infatti, i colori chiari tendono a riflettere meglio la radiazione emessa dai LED IR del robot, permettendo a RZB-1p di identificare gli ostacoli a distanza maggiore. Inoltre, poichè i sensori di prossimità che abbiamo realizzato operano 'a soglia', cerca di muovere il cartoncino mantenendo una velocità costante, in modo da mantenerlo sempre nel range 'visivo' dei sensori. Ora che hai visto come utilizzare le funzioni base di RZB-1p puoi provare a sperimentare programmi personalizzati, realizzando, ad esempio, un comportamento analogo a quello che abbiamo implementato nella prima versione del robot, ma ottenuto usando un PIC e il linguaggio C.



*La configurazione 'follower' (ossia 'inseguitore') di RZB-1p, che realizziamo in questo Workshop, fa si che il robot avanzi quando il sensore di prossimità anteriore del robot è l'unico a essere attivato dalla presenza di un oggetto.*