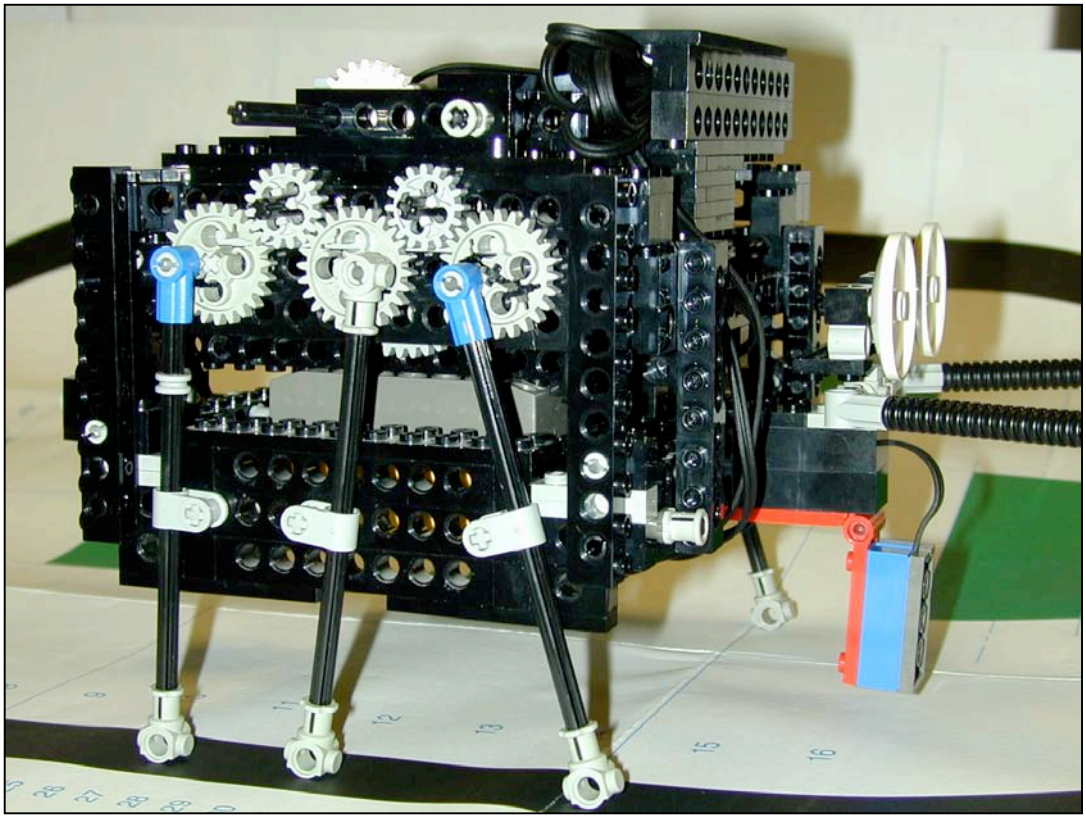
ROBOTICA

Prof.: Riccardo Cassinis

Relazione dell'elaborato
"Zeta: robot a sei zampe che segue una pista"



A cura di:

Luca Lazzeroni Mat.: 024417
Stefano Marpicati Mat.: 016907

Introduzione al Kit Lego® Mindstorm

Lego® Mindstorm è una serie di prodotti di Lego® System orientati alla robotica. Si tratta di un'estensione della serie Technics che affianca alla classica gamma di componenti meccanici ed elettromeccanici nuovi dispositivi con preponderante componente elettronica.
Questi pezzi consentono di controllare con discreta precisione un certo numero di motori elettrici, che varia a seconda del modello di kit in uso, e di rilevare i dati provenienti da sensori, siano essi semplici interruttori oppure sensori analogici.

Il kit in nostro possesso è il "Robotics Invention System 1.5"; incorpora un controllore in grado di gestire tre sensori e tre motori; sono forniti in tutto due motori , due sensori di contatto ed un sensore di luminosità.

Il cuore del sistema è rappresentato dal dispositivo controllore, denominato RCX, così come il linguaggio di programmazione che è in grado di comprendere.
Tale controllore è alimentato da 6 batterie stilo alcaline da 1.5 V che vengono utilizzate sia per l'alimentazione dell'elettronica sia per dare energia ai motori.
Viene fornito un sistema di sviluppo visuale che gira su piattaforma Windows e permette di scrivere il codice da programmare nella memoria non volatile del controllore. La comunicazione tra il PC ed il controllore avviene tramite raggi infrarossi; per permettere a qualunque PC di comunicare col controllore viene fornito a corredo del kit un ricetrasmettitore a raggi infrarossi autoalimentato, tramite una batteria da 9 Volt, che si connette direttamente ad una porta seriale RS-232.

I due motori che abbiamo avuto a disposizione non sono, purtroppo, motori passo-passo; il controllo degli stessi può essere effettuato in due soli modi:

- Variandone la velocità, che, tra l'altro, è anche funzione dello stato di carica delle batterie
- Variandone la direzione

Non è possibile, non esistendo encoder sui motori, imporre loro una rotazione determinata; piuttosto è possibile formulare richieste del tipo "ruota per 5 secondi". La precisione temporale fornita dal sistema è nell'ordine del decimo di secondo.
La velocità dei motori è variabile in un range da 1 a 8; sia la velocità sia il verso sono impostabili in modo indipendente per ciascuno dei motori connessi al sistema.

La connessione tra i motori ed il sistema avviene mediante un cavo bipolare alle estremità del quale sono posti appositi mattoncini Lego® nelle cui cavità sono presenti contatti elettrici; tali mattoncini trovano un corrispondente punto di connessione sull'unità di controllo e possono essere connessi senza preoccuparsi dell'orientamento reciproco del mattoncino-spina rispetto al mattoncino-presa, a parte la variazione del verso di rotazione del motore che ne consegue, essendo i motori in corrente continua.

I sensori a nostra disposizione sono tre: due sono dei banali pulsanti in grado di fornire al controllore un segnale a gradino quando premuti, uno è un sensore costruito mediante un LED rosso ed un sensore di luminosità. Tale sensore è in grado di fornire al sistema un valore percentuale che rappresenta la luminosità percepita.
La luminosità che il sensore rileva è influenzata dalle condizioni ambientali ed obbliga i progettisti di software a ricalibrare le soglie di tolleranza sul campo di lavoro del robot.

Per quanto riguarda la parte meccanica del kit, sono forniti 7500 pezzi Lego® di ogni tipo, prevalentemente pezzi appartenenti alla serie Technics opportunamente arricchita di elementi utili per implementare architetture complesse e trasformazioni di moto; tra i tanti pezzi forniti, in particolare, risultano utili le ruote a cedevolezza selettiva, pezzi completamente nuovi nel panorama Lego® dei quali verrà fornita un'analisi dettagliata nel capitolo dedicato all'implementazione del robot.
Sono anche stati aggiunti alla dotazione Lego® pezzi indispensabili per realizzare manovellismi e alberi a camme, quali quelli utilizzati nel nostro progetto.

Nella nostra relazione descriveremo l'architettura del software di sviluppo fornito a corredo del kit e vedremo come è stato possibile portare a termine il nostro progetto di robot-insetto, sia per quanto riguarda il lato software, sia per quanto riguarda il lato hardware.

Il sistema di sviluppo

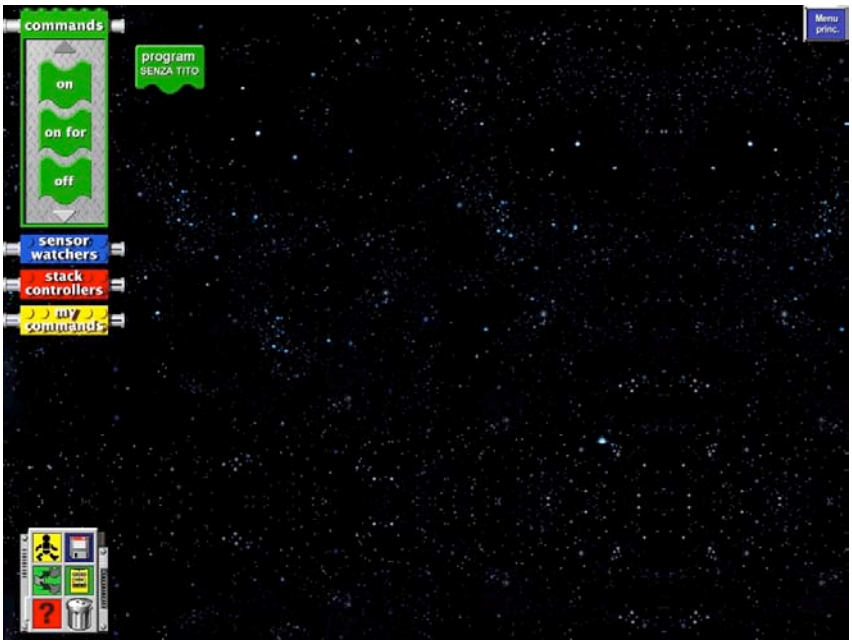
Il sistema di sviluppo fornito a corredo del kit è costituito da un insieme di programmi che illustrano le potenzialità del prodotto, mostrando possibili realizzazioni e proponendo all'utente un breve corso di addestramento, costituito da una parte teorica e una pratica, che lo metta in grado di costruire i suoi primi robot.
Vengono forniti, tramite brevi sequenze video, i primi rudimenti di meccanica applicata ai robot, illustrando come sia possibile realizzare sistemi di moto, interfacciamento meccanico di sensori e programmazione del controllore.
I primi robot realizzati sono veicoli a quattro ruote, due motrici, con differenziale. Sono piccoli robot autonomi in grado di evitare ostacoli muovendosi casualmente sul piano.

Passata la prima, ed inevitabile, fase di addestramento, durante la quale sistema non lascia libertà all'utente di iniziare lo sviluppo autonomo, si ha accesso alla programmazione ed alle sfide.
Le sfide sono proposte di realizzazione di robot che eseguano compiti più complessi, tipo la gestione del ribaltamento, corredate di soluzione e di test per valutare se si è seguita la giusta strada di realizzazione.

L'ambiente di programmazione consiste di un tool grafico che consente, con estrema rapidità, di strutturare programmi in linguaggio RCX. Il linguaggio è rigidamente strutturato; non sono presenti né istruzioni di salto né possibilità di uscita intermedia dai cicli mediante istruzioni tipo "break" tipiche del linguaggio C; purtroppo non è possibile l'uso di variabili dichiarate dal programmatore, costringendo all'uso delle sole variabili implicite rappresentanti lo stato dei sensori.
Qualsiasi evento che condizioni il flusso del programma è esprimibile unicamente con frasi del tipo "il sensore 1 è premuto", oppure "il sensore 2 rileva una luminosità del 15%".
Esiste una sorta di contatore gestito dal software, ma si tratta di un contatore generico che, oltre ad essere incrementato dall'utente tramite un'apposita istruzione, aumenta automaticamente ogniqualvolta uno dei sensori viene sollecitato; questo rende imprevedibile il progresso del contatore, rendendolo inutile per realizzare condizioni di uscita forzata da cicli.
Esiste anche un timer; si può utilizzare per generare una sorta di interrupt a cadenza periodica, chiedendo al controllore di iniziare l'esecuzione di un determinato task con intervallo di ricorrenza fissato dall'utente.
Le istruzioni di verifica di una qualsiasi condizione codificano implicitamente, in forma grafica, l'informazione circa il tipo di sensore da controllare.

Lo stesso discorso si può applicare ai cicli repeat/until e repeat/while, nei quali si può specificare una condizione espressa come “ripeti 10 volte” oppure “ripeti finché il sensore 1 non è schiacciato”.

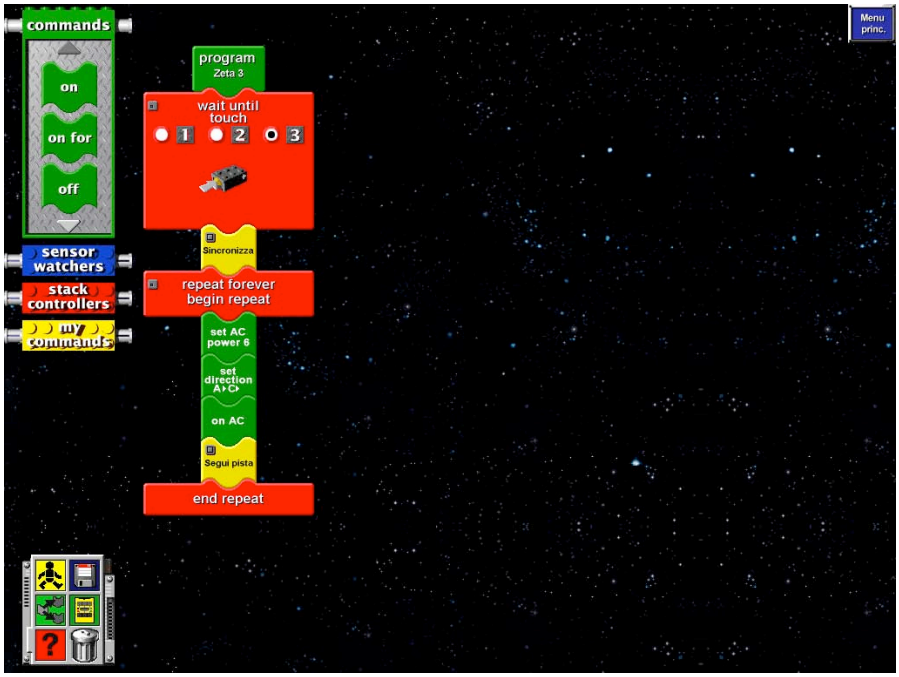
La programmazione avviene in modalità drag & drop. Come si può osservare nella figura successiva, lo schermo è diviso in 3 parti:



1. La barra delle istruzioni, della quale fanno parte i comandi di moto, di attesa,. espressa in decimi di secondo, di emissione di suono e di comunicazione con altre unità RCX, di controllo dei sensori, di controllo di esecuzione e le librerie dei sottoprogrammi creati dall’utente.
2. La barra dei comandi al sistema, caratterizzata da 6 icone che rappresentano il debugging, la gestione dei files su disco, l’azione di copia/incolla, la fase di download del programma sul controllore, l’help ed il cestino per eliminare una o più istruzioni dal programma
3. L’area di programmazione vera e propria, nella quale l’utente può trascinare le istruzioni che vuole inserire nel suo programma prendendole direttamente dalla barra apposita.

La scrittura di un programma avviene proprio trascinando dalla barra delle istruzioni i pezzi di interesse, come se si stesse costruendo un puzzle.
Il sistema obbliga l’utente a modularizzare i programmi onde evitare di avere lo schermo letteralmente riempito di blocchi; la visualizzazione di un programma complesso, così come la sua stesura, avviene per approfondimenti successivi; si parte da un livello di dettaglio basso per andare ad esplorare all’interno ogni singolo blocco, fornendo la possibilità di comprimerlo per ridurre l'occupazione di spazio video.
Un’idea della fase di editing si può avere osservando le due immagini successive.

Nella prima si ha la visione che il sistema ci dà di un programma realizzato in precedenza; si noti come il blocco repeat/end repeat sia chiaramente identificabile e consenta al programmatore di individuare rapidamente il blocco di istruzioni che verranno ripetute.



I blocchi in giallo sono sotto-procedure sviluppate dall'utente, che vengono "sintetizzate" rappresentandone solo il nome all'interno del blocchetto.

Volendo lo stesso discorso è applicabile al blocco repeat/end repeat; cliccando col mouse sul quadrettino in alto a sinistra all'interno del blocco "repeat forever…" se ne provoca l'espansione o la riduzione. Ripetendo il procedimento per tutti i blocchi strutturati presenti nel programma, quasi certamente si presenterà il problema di aver esaurito lo spazio disponibile sullo schermo; in tal caso è sufficiente muovere il mouse nella direzione di interesse per provocare uno scrolling automatico dell'intera videata.

La figura seguente mostra proprio questo aspetto; si è dovuto scrollare lo schermo per creare lo spazio necessario a contenere l'espansione del blocco che rappresenta la procedura "Sincronizza":

Si possono vedere anche 3 blocchi che esprimono condizioni sul sensore di contatto connesso alla porta 1 del controllore. Il primo è un controllo del tipo "attendi che il sensore 1 sia libero da pressioni"; viceversa il secondo è un controllo tipo "attendi che il sensore 1 sia premuto". Il terzo è un controllo identico

al primo.

Durante lo sviluppo del programma è possibile spostare interi blocchi di istruzioni; basta posizionarsi col mouse sulla prima istruzione che si vuole muovere e cliccare su di essa; il software provvederà a "sganciare" dalla struttura l'istruzione selezionata e tutte le sottostanti, consentendo di riagganciarle in un altro punto del programma oppure solo di "parcheggiarle" in una posizione vuota dello schermo.

Quando la fase di editing vero e proprio è terminata, si procede al download del programma appena realizzato nella memoria del robot, previo salvataggio su disco. Questa procedura inizia premendo col mouse sul tasto a forma di controllore, posizionato subito sopra al tasto a forma di cestino.
La comunicazione avviene, come già accennato, tramite raggi infrarossi; il dispositivo di trasmissione offre la possibilità di funzionare in due differenti modalità: bassa ed alta potenza, a seconda della distanza tra il trasmettitore ed il controllore del robot. In modalità a bassa potenza il controllore, e quindi il robot, deve essere praticamente davanti al sensore e a distanza abbastanza limitata, tipicamente nel raggio di 15 cm.
In modalità ad alta potenza, invece, possono esserci anche diversi metri che separano il robot dal PC; questa è una possibilità estremamente utile per il debugging; se infatti si seleziona l'icona a forma di uomo che cammina su fondo giallo, cliccando col mouse in un qualsiasi punto del programma, purché non si tratti di un blocco sotto-procedura, si causa l'esecuzione, da parte del robot, dello step individuato; se l'azione, in particolare, è un controllo sul valore dei sensori, si ha una visualizzazione aggiornata in tempo reale del valore rilevato dal sensore oggetto del controllo.
Questa è, in pratica, la fase di debugging; provare e riprovare i vari passi fino ad ottenere risultati soddisfacenti.

Spesso, infatti, ci è capitato che un algoritmo strutturalmente corretto non funzionasse, qualora implementato nel mondo reale, come aspettato. Le cause sono moltissime; si va dalle condizioni ambientali allo stato delle batterie del robot, dagli attriti sviluppati dalle ruote dentate a particolari condizioni di sfasamento tra le zampe che sostengono il robot. Tutti questi fattori ci hanno provocato diversi problemi che sono stati corretti mediante una verifica sul campo del comportamento della macchina; ecco quindi che, viste le reali condizioni operative, abbiamo corretto le potenze espresse dai motori; abbiamo opportunamente alzato o abbassato le soglie di tolleranza del sensore di luminosità; abbiamo osservato, e compensato, un'iniqua distribuzione dei pesi sui due motori che ci ha costretto ad una altrettanto iniqua distribuzione delle potenze da erogare.

Altre volte, invece di correggere il software, abbiamo apportato modifiche alla struttura hardware del robot, cercando di ridurre attriti, di amplificare coppie, di ridistribuire pesi.
La fase di debugging ha richiesto un tempo considerevole, rapportato al tempo totale trascorso nella realizzazione del progetto; il software e l'hardware sono stati soggetti a continue revisioni; in particolare le revisioni più importanti sono state 3; chiamandosi il nostro robot "Zeta", come la formica del noto film, ed essendo la versione definitiva la terza, abbiamo battezzato "Zeta 3" il progetto.

Il programma Zeta 3

Il codice prodotto graficamente viene pre-compilato in una forma C-like e memorizzato in files con estensione RCX; andando ad analizzare il contenuto di tali files, ed in particolare del file zeta3.rcx, troviamo il codice riportato in appendice a questa relazione che andiamo a commentare.
Il programma è costituito da 3 grandi parti:

- L'inizializzazione, comprendente la fase di sincronizzazione delle 6 zampe onde avere almeno 3 punti d'appoggio.
- Il loop di controllo sullo stato del sensore ottico.
- L'eventuale fase di curva e conseguente risincronizzazione del robot che ha, per qualche motivo, perso di vista la pista nera.

La fase di inizializzazione prevede, essenzialmente, l'allineamento delle zampe in una posizione nota, con 2 zampe a terra da un lato e 1 dall'altro.
La condizione iniziale, per la risoluzione del problema "insetto che cammina" è di importanza preponderante. Infatti il moto del robot è realizzato mediante azionamento sincronizzato, realizzato mediante accorgimenti meccanici, dei 2 motori nella medesima direzione. Perché il robot abbia una traiettoria rettilinea, tuttavia, è indispensabile che lo sfasamento tra le zampe si mantenga costante, onde garantire che, in qualunque momento della "camminata", il robot abbia almeno 3 punti di appoggio, pena l'instabilità.
L'instabilità nella deambulazione si traduce in pericolose oscillazioni ed impuntamenti che tendono a far sbandare il robot o addirittura a ribaltarlo.
Onde evitare questi spiacevoli inconvenienti si è utilizzata una struttura hardware-software basata su 2 camme vincolate agli alberi di rotazione delle ruote dentate che danno il moto alle zampe di destra e di sinistra.

Tale struttura, che verrà descritta con maggiore precisione nella sezione sull'implementazione, aziona, con cadenza opportuna, un sensore di contatto preposto alla rilevazione della condizione di sincronia.

Una volta terminata la sincronizzazione iniziale, che serve per portare il sistema in uno stato noto, il programma attiva entrambi i motori nella medesima direzione e con la medesima potenza; dopodiché entra in un loop di attesa, durante il quale i motori continuano ad essere alimentati ed il robot avanza, dal quale esce solo quando il sensore ottico non rileva che la luminosità è salita oltre un determinato livello, indicando che il robot è uscito di pista o, per qualche motivo, ha perso il contatto con la pista.

Il sensore ottico viene gestito in modalità *polling* con cadenza di mezzo secondo; sarebbe stato possibile gestirlo in modalità interrupt, ma abbiamo preferito la modalità polling per mantenere sempre il controllo del flusso del programma. In modalità interrupt, infatti, non è possibile fissare la cadenza temporale con la quale viene rilevata la condizione di attivazione dell'interrupt; poiché abbiamo notato una certa lentezza a percepire mutamenti delle condizioni quando si utilizza questa modalità, abbiamo deciso di non utilizzarla.

Se dunque il controllo sul sensore di luminosità indica che il sistema sta progredendo senza intoppi seguendo la pista nera, il programma continua imperterrito a ciclare e ad effettuare i suoi controlli.

Se invece il sensore ottico rileva che si è abbandonata la pista nera, il programma entra in una sequenza di azioni volte a far ritrovare la pista al robot:

- Prima di tutto si bloccano tutte le zampe e si inverte il moto di quelle di destra rispetto a quelle di sinistra.
- Si re-imposta la potenza dei motori, aumentandola, onde permettere alla ruota a cedevolezza selettiva, che funge da frizione, di svincolare il moto delle zampe di destra rispetto a quelle di sinistra che, altrimenti, proseguirebbero sincronizzate ad opera del sincronizzatore meccanico.
- Si riavviano i motori e si attende per 1 secondo e mezzo, tempo sufficiente a far curvare il robot verso destra di circa 20 gradi. Si spengono i motori.
- Si rileva nuovamente lo stato del sensore ottico, presentandosi due possibili scelte:
    1. Presenza di linea nera: si procede alla fase di risincronizzazione, si resettano i versi e le potenze dei motori e si riparte in linea retta.
    2. Assenza di linea nera: in tal caso il flusso del programma prosegue.
- Si inverte in modo reciproco il verso dei motori; in questo modo il robot curva verso sinistra. Per poter compensare il diverso sforzo a cui la ruota a contatto con la ruota di frizione è soggetta rispetto all'altra, si attendono 3 secondi e mezzo. A questo punto,

tipicamente, il robot si è girato di cerca 25 gradi verso sinistra rispetto alla direzione iniziale.

- Si controlla nuovamente la presenza di linea nera sotto al sensore ottico:
    1. Presenza di linea nera: risincronizzazione e riavvio del robot in linea retta.
    2. Assenza di linea nera: a questo punto, se la curva non era né a destra né a sinistra, vuol dire che, probabilmente, c'è qualcosa che non ha funzionato adeguatamente; pertanto il robot, che si trova in una condizione in cui non è sulla pista, risincronizza le ruote e ricomincia il ciclo come se si trovasse su una linea nera; non essendovi, rientra quasi immediatamente in questa sequenza di azioni; poiché però la rotazione verso destra non è esattamente uguale a quella verso sinistra, il robot guadagna ad ogni giro un paio di gradi che gli permettono di ruotare di 180 gradi, dopo un numero ragionevole di giri, e recuperare la pista nera.

La disparità tra il tempo assegnato alla rotazione verso destra e quello assegnato alla rotazione verso sinistra è uno di quegli accorgimenti che, sul campo, abbiamo dovuto adottare per permettere al robot di svolgere il suo compito; il nostro modello, infatti, prevedeva una rotazione perfettamente simmetrica verso destra e verso sinistra, ma, all'atto pratico, era ben lontana dall'esserlo.
La fase più critica, dal punto di vista meccanico, è quella in cui il robot curva; in tale momento, infatti, i motori sono soggetti a sforzi molto consistenti perché, per svincolare la ruota motrice di destra da quella di sinistra, devono erogare coppie in grado di far slittare la ruota a cedevolezza selettiva. Tali coppie finiscono col diminuire la potenza trasmessa alle ruote motrici, rendendo il moto del robot più lento.
Inoltre, essendo in curva le zampe necessariamente sfasate tra loro, il robot presenta oscillazioni dovute al fatto che, in determinate condizioni, ha solo 2 punti d'appoggio; in tale contesto il dispositivo "cade" in avanti o all'indietro finché una delle altre zampe non tocca terra. Queste oscillazioni tendono a far spostare leggermente il robot dalla posizione in cui si trova, rendendo talvolta più lunga la fase di ricerca della pista nera.
Un altro approccio alla curva sarebbe stato quello di bloccare le zampe da un lato ed azionare solo quelle dall'altro, garantendo in questo modo sempre almeno 3 punti d'appoggio, ma introducendo attriti dovuti all'inevitabile strisciamento delle zampe ferme. Volendo noi realizzare un robot che ruotasse sul posto, abbiamo dovuto optare per la soluzione leggermente "instabile".
La sincronizzazione dopo la curva ha comportato qualche problema in quanto in tale fase si aziona un solo motore per volta, realizzando in pratica una micro-curva seguita da un'altra contro curva.
Tali lievissime oscillazioni portano il robot a traslarsi leggermente rispetto alla posizione di "pista trovata", portandolo talvolta, in caso di curve troppo strette, a iterare più volte la ricerca della pista.

Le temporizzazioni di rotazione in curva sono opportunamente tarabili sul campo in funzione del tipo di percorso che il robot deve seguire.
Utilizzando un linguaggio di programmazione un po' più versatile, che consenta magari l'uso di variabili, sarebbe certamente possibile passare all'implementazione di una funzione di auto-taratura del robot.

Visione a basso livello

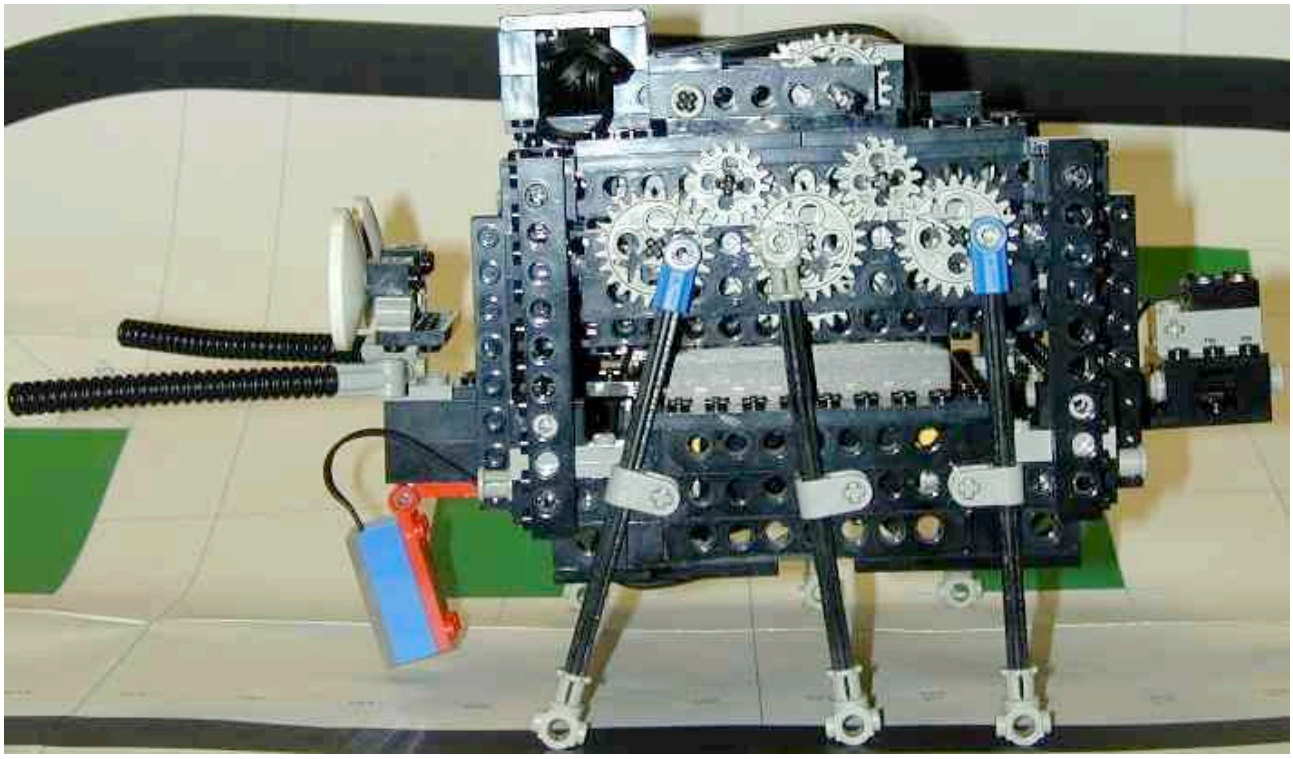
Sbirciando nei files in dotazione al kit, e, nella fattispecie, nel file del firmware da caricare nel controllore al momento dell'inserimento di nuove batterie, abbiamo osservato varie caratteristiche interessanti.
Il firmware in dotazione è un file in formato S19, formato prediletto dai tool di sviluppo di Motorola; presumiamo pertanto che il microcontrollore all'interno del dispositivo RCX sia della famiglia Motorola. Questa nostra supposizione trova riscontro anche in alcuni messaggi letti su vari gruppi di discussione in Internet.
Essendo il firmware in formato S19, riusciamo anche a vedere che la locazione all'interno del controllore nella quale viene memorizzato il firmware è 7800h.
Volendo sarebbe possibile disassemblare il firmware della macchina per apportargli opportune modifiche e comprendere più in dettaglio l'architettura del controllore.

Implementazione del robot

Il progetto è partito con l'idea di creare un robot, costruito utilizzando mattoncini Lego® somigliante ad un insetto, quindi dotato di zampe, che riesca a seguire un percorso.
Abbiamo avuto disposizione

- 2 Motori
- 1 sensore ottico
- 2 sensori di contatto
- 1 ruota dentata a cedevolezza selettiva che permette di svincolare due alberi quando la differenza tra le coppie ad essi applicate superi il valore di 2.5N.

Mediando fra le esigenze di stabilità della macchina e il numero di pezzi che la confezione (pur essendo la più completa nel suo genere) ci metteva a disposizione siamo giunti alla decisione di costruire un robot a sei zampe, tre per lato unite attraverso ruote dentate ad una ruota destinata a ricevere la forza motrice; il robot così fatto poggia a terra con sei zampe, ma dal punto di vista della forza motrice è una macchina a due ruote



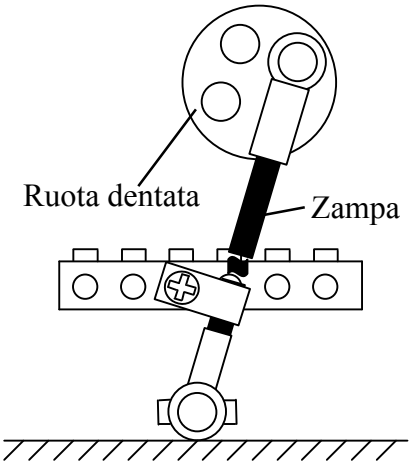
Con una struttura di questo tipo, perché il robot riesca a camminare è necessario che le zampe siano sempre in perfetta sincronia tra loro, in caso contrario il robot oscilla sul posto senza procedere nel verso previsto.

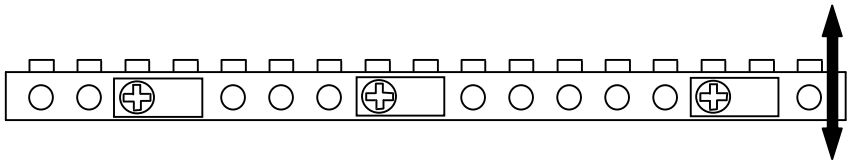
La scelta del numero di motori da utilizzare per la movimentazione della macchina ha visto quindi la sincronia fra le sei zampe e la potenza necessaria a muoverle come obiettivi prioritari.

Abbiamo dapprima esplorato, con vari tentativi, la possibilità di usare un motore solo per la movimentazione di tutte e sei le zampe; questa soluzione ha però evidenziato i seguenti problemi:

- se si uniscono meccanicamente, tramite ruote dentate, il motore con le due bancate di zampe, le zampe sono sempre sincronizzate, ma il robot può procedere solo in linea retta

- per riuscire ad eseguire una curva con questa struttura si potrebbe modificare il punto di attacco del sistema biella manovella che permette di convertire il moto da quello circolare della ruota dentata al lineare della zampa; i tre sistemi sono collegati su un'unica barra, quindi per modificarne il punto di contatto sarebbe necessario applicare alla stessa un sistema che le permetta di sollevarsi e abbassarsi.





È possibile applicare questo procedimento ad entrambi i lati del robot oppure ad un solo lato ottenendo curve di diverso raggio.

Tuttavia, tale soluzione richiederebbe dei materiali tali da assicurare la rigidità necessaria alla struttura portante del robot permettendo, contemporaneamente, di svincolare i due elementi laterali.

Cercando di implementare questa soluzione con dei mattoncini Lego® ci siamo resi conto che il materiale di cui sono costituiti, e il sistema di aggancio di cui dispongono, non permettono una coesione ed un contenimento degli attriti tali da rendere questa soluzione praticabile; in sostanza la struttura del robot diveniva troppo esile e si verificavano diversi episodi di impuntamento dei meccanismi articolati

In alternativa

- si è costruito un cambio a cinque marce assegnate ciascuna alle seguenti funzioni
    1. Moto di entrambe le ruote nello stesso verso
    2. Inversione del moto di una ruota rispetto all'altra
    3. Retromarcia (moto di entrambe le ruote nello stesso verso, opposto a quello della 1à marcia)
    4. Moto sulla ruota di destra, ruota di sinistra ferma
    5. Moto sulla ruota di sinistra, ruota destra ferma

  Questa soluzione si è rivelata presto impraticabile, per le dimensioni, e quindi il peso, che il gruppo cambio veniva ad assumere: eccessive per essere azionate da un motore così piccolo e per la mancanza di una frizione che permettesse di cambiare marcia durante il moto.
- si è pensato quindi di inserire, tra la bancata di destra e quella di sinistra, la ruota dentata a cedevolezza selettiva di cui disponevamo; questa soluzione tuttavia permette solo due direzioni del moto, avanti diritto e curva in un solo verso; come si può intuire questa soluzione permetteva di risparmiare un motore da dedicare ad altri compiti, ma limitava fortemente le possibilità di movimentazione della macchina, che nei nostri intenti invece doveva potersi muovere come un insetto.

Dal momento che queste soluzioni non hanno portato risultati soddisfacenti abbiamo deciso di optare per una struttura che prevedesse un motore per bancata di zampe.

Questa soluzione permette di disporre di potenza sufficiente a muovere il robot, ma pone il problema della sincronia tra le zampe di destra e quelle di sinistra.

I motori a nostra disposizione non sono dotati di un circuito di retroazione; sono comandabili solamente aumentando l'intensità di corrente di alimentazione per cercare di aumentarne la velocità che, naturalmente, a parità di corrente, è variabile in funzione del carico applicato all'albero motore.

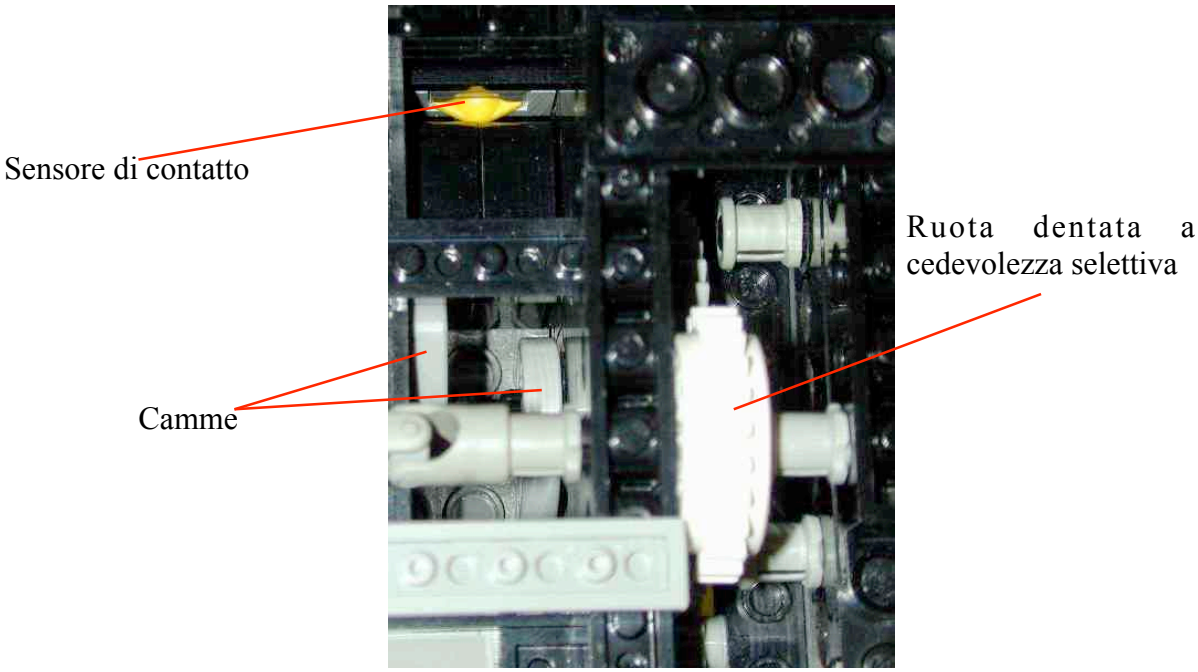
I due motori sono soggetti a carichi variabili e di tipo impulsivo; questo fa sì che non sia possibile far procedere il robot in moto rettilineo semplicemente impostandoli su una determinata velocità nella stessa direzione: a meno che non siano collegati tra loro, dopo pochi passi, le zampe si disallineano.

Si è reso necessario implementare un sistema che consentisse di collegare tra loro le ruote dentate di destra e di sinistra per muoversi di moto lineare, e di poterle svincolare per eseguire curve.

La soluzione adottata è stata di utilizzare la ruota dentata a cedevolezza selettiva per far sì che qualora i due motori esercitassero la loro forza motrice girando entrambi nello stesso verso tutte le zampe risultassero unite, mentre qualora gli alberi motore girassero in versi opposti questo

meccanismo permettesse di disaccoppiare le zampe di destra da quelle di sinistra che in questo modo potrebbero muoversi a velocità diverse o in versi opposti.
Eseguire una curva vuol dire muovere le zampe interne più lentamente di quelle esterne che devono percorrere un percorso più lungo; nel nostro caso significa addirittura che le zampe interne si muovano in verso opposto rispetto alle esterne; questo implica che se le zampe erano sincronizzate all'inizio della curva, non lo sono più alla fine.
Si è reso necessario quindi implementare un sistema che permettesse la risincronizzazione al termine dell'esecuzione di una traiettoria curvilinea.

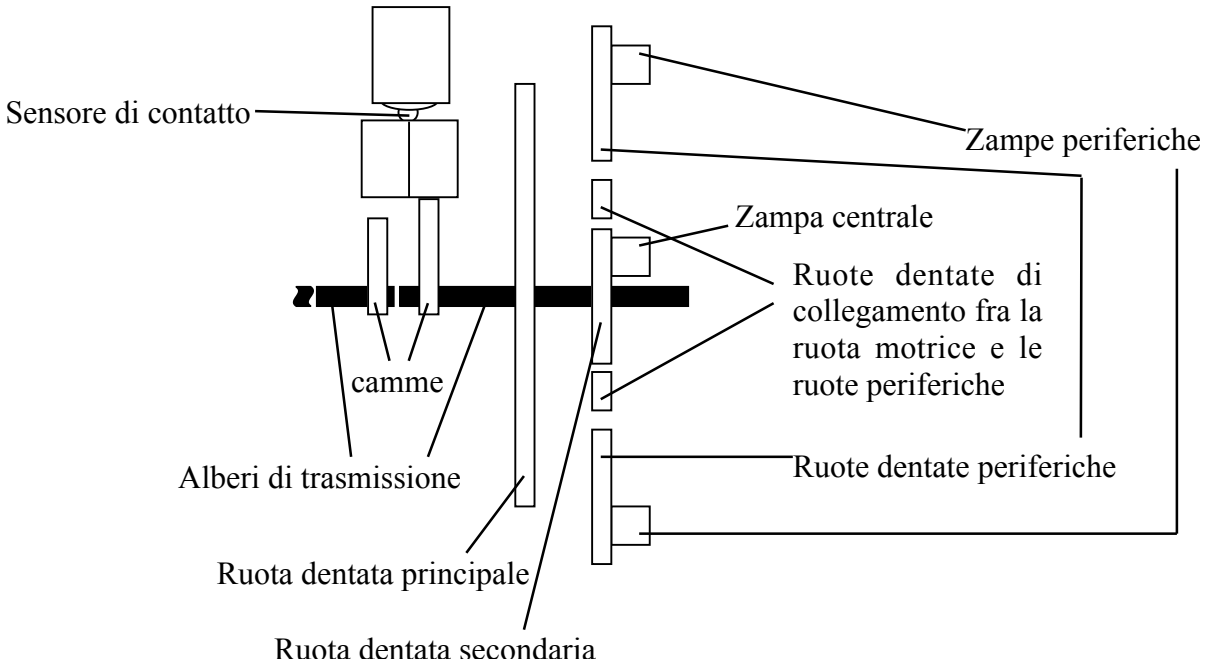
Abbiamo a tal fine inserito due camme sugli alberi di trasmissione; in questo modo, usando un sensore di contatto, è stato possibile determinare la posizione delle ruote su di essi calettate e quindi riportarle in sincronia.



Avendo utilizzato un solo sensore, le ruote vanno azionate in modo opportuno; per sincronizzare le zampe al termine di una curva, prima viene azionata la ruota di destra, finché, attraverso la camma, non provoca una transizione dello stato del sensore, corrispondente al passaggio della camma davanti al sensore e, quindi, al posizionamento delle zampe in modo tale che quella centrale sia a terra e le due laterali siano alzate.
A questo punto, quando la corsa del motore è tale che il sensore non risulta più premuto, il motore viene arrestato; viene quindi attivato il motore corrispondente alle zampe di sinistra e si ripete il

procedimento, considerando che la condizione di sensore premuto corrisponde, in questo caso, a due zampe a terra e quella in mezzo alzata.

Quando il sensore si libera, il robot è in condizioni di stabilità con 3 punti di appoggio; può quindi procedere, avviando entrambi i motori, in linea retta senza eccessivi problemi.



Il robot non compie delle curve solo per seguire il percorso tracciato, ma anche per effettuare delle correzioni di traiettoria quando non "vede" più la pista; per riuscire a seguire un percorso tracciato per terra, la macchina utilizza un'appendice posta sul davanti che contiene un LED e un sensore di luminosità; a questo sensore corrisponde una funzione nel kit di sviluppo che consente di costruire un'istruzione condizionale avente come base il valore acquisito.

La confezione della Lego® fornisce un foglio riportante un circuito che ben si presta ad essere un valido strumento di collaudo delle prestazioni della macchina.

Il sensore di luminosità è stato tarato in una stanza ben illuminata da luce al neon, nella quale, posto sopra alla pista (di colore nero) forniva un valore percentuale di 38.

Dato che dovevamo seguire una pista nera su sfondo bianco, quindi a forte contrasto, sono stati considerati come scuri i valori inferiori a 38 e come chiari i valori superiori riducendo a zero la zona di incertezza: i livelli di luminosità rilevati dal sensore nelle due zone differivano di pochi punti, quindi risultava poco praticabile stabilire una zona di incertezza in un intervallo così limitato.

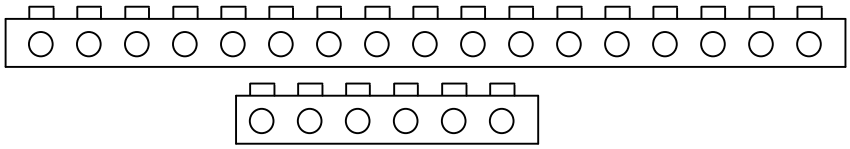
I sensori di luminosità sono purtroppo molto sensibili alle variazioni delle condizioni al contorno della prova; per questo se si trovano in condizioni di illuminazione diverse da quelle nelle quali

sono stati tarati, i livelli di riconoscimento del chiaro e dello scuro cambiano anche di diversi punti e i sensori vengono indotti facilmente in errore

Sarebbe opportuna una taratura sul campo ogni volta che la prova viene eseguita con condizioni di luminosità variate.

Conclusioni

La realizzazione di un robot con in kit di montaggio di questo tipo pone dinanzi a problemi sia di tipo meccanico e cinematico, sia di tipo informatico che sono tipici della robotica.
L'elevata modularità che questo tipo di kit consente, permette di esplorare diverse soluzioni ai problemi suddetti, caratteristica questa che ne mette in risalto la possibilità di utilizzo a fini didattici.
L'unico limite alla creatività dei progettisti è dato dall'esigua disponibilità elementi attivi all'interno della singola scatola; la realizzazione di opere più complesse richiederebbe l'utilizzo di più kit di montaggio, soluzione che , comunque, potrebbe rappresentare una possibilità economica per lo sviluppo di prototipi.



Appendice A - Codice RCX del programma Zeta 3

```
#source RIS1.5
#target RCX

main [
sensor s1 on 1
sensor s2 on 2
sensor s3 on 3
output motorA on 1
output motorB on 2
output motorC on 3
forward [motorA motorB motorC]
power [motorA motorB motorC] 7
s1 is switch as boolean
s2 is light as percent
s3 is switch as boolean
start progTask
wait 10
forever []
]

task progTask [
while s3 is not closed [ ]
MCSincronizza
forever [
power [motorA motorC] 5
forward [motorA motorC]
on [motorA motorC]
MCSegui_pista
]
]

fragment(542,2995) [
if s2 is 0..38 [
][
reverse motorC
MCCerca_nero
]
]

inline MCSincronizza
[
while s1 is not opened [ ]
sound 6
off [motorA motorC]
forward [motorA motorC]
power [motorA motorC] 5
on motorA
while s1 is not closed [ ]
while s1 is not opened [ ]
```

```
wait 10
off motorA
on motorC
while s1 is not closed [ ]
while s1 is not opened [ ]
wait 10
off motorC
power [motorA motorC] 5
]

inline MCSegui_pista
[
if s2 is 0..40 [
][
power [motorA motorC] 5
sound 3
forward motorC
backward motorA
wait 150
if s2 is 0..40 [
MCSincronizza
][
forward motorA
backward motorC
wait 350
MCSincronizza
]
]
]

inline MCCerca_nero
[
if s2 is 0..38 [
wait 100
off [motorA motorC]
][
]
]
```