IL TEMPO DEI ROBOT

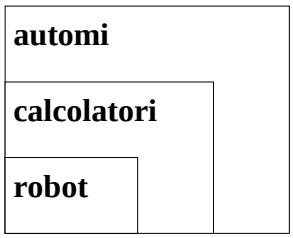
Sinora abbiamo visto come, data una procedura che comporta una elaborazione di informazioni, fosse possibile codificarla in un linguaggio artificiale comprensibile da un calcolatore e, quando necessario, fargliela eseguire.

Con questa unità di apprendimento ci occuperemo di quei calcolatori che, essendo in grado di controllare parti meccaniche, possono operare in modo attivo nell'ambiente spostandosi, muovendo e/o manipolando oggetti, ecc.

Il seguente schema ci aiuterà a riepilogare quanto studiato sinora in merito ai vari tipi di automi.

- eseguono procedure

- memorizzano ed eseguono procedure riguardanti la gestione di informazioni
- memorizzano ed eseguono procedure riguardanti la gestione di informazioni e di materiali



Le parti di un robot

Osservando buona parte dei robot oggi esistenti possiamo individuare le seguenti parti:

- l'**unità di controllo**, che è un vero e proprio calcolatore, riceve i segnali dai sensori e gestisce gli organi attuatori in base al programma precedentemente memorizzato. In questa Unità Didattica ci occupiamo di unità di controllo programmabili.
- i **sensori** hanno lo scopo di raccogliere segnali di variazione di stato esistenti all'esterno e di inviarli all'unità di controllo. Tali variazioni possono riguardare luci, movimento, suoni e per individuarli vi sono sensori ottici, tattili e acustici. Il segnale inviato dal sensore sarà analizzato dal programma attivo nell'unità di controllo.
- gli **organi attuatori** consistono nell'insieme delle parti meccaniche, costituite da elementi rigidi variamente articolati tra di loro e mossi grazie all'impiego di motori elettrici, pneumatici ecc. Tali motori vengono accesi e spenti grazie a segnali provenienti dall'unità di controllo.

Normalmente i robot vengono costruiti appositamente; il collegamento tra l'unità di controllo, sensori e organi attuatori viene già realizzato in fase di costruzione.

Tutti i normali calcolatori presentano però la possibilità di aggiungere alle proprie possibilità di lavoro anche il controllo di motori e sensori. Questo può avvenire utilizzando le **porte** collocate sul retro e destinate proprio a collegare il calcolatore con l'esterno (USB, seriale, parallela). Nella RAM del calcolatore vi sono appositi **byte da 8 bit** (denominati **registri**) destinati ad inviare all'esterno segnali elettrici attraverso le porte (output) oppure a ricevere segnali elettrici provenienti dall'esterno (input).

Dovrà però essere realizzata un'**interfaccia** esterna (con una autonoma fonte di corrente elettrica) nella quale i segnali elettrici provenienti dal calcolatore potranno aprire o chiudere l'alimentazione dei singoli motori. Infatti le correnti che viaggiano nei cavetti in uscita dal calcolatore sono troppo deboli per accendere il motore. Esse però portano il segnale (corrente/non corrente // 0 oppure 1) che attiverà i relè collocati ad aprire o chiudere l'alimentazione dei motori.

D'altra parte compito dell'interfaccia sarà anche indirizzare verso il calcolatore i segnali provenienti dai sensori.

Le schede di robotica: dalla LEGO a MZ

Nel 1986 la ditta Lego fece costruire un'interfaccia collegabile, grazie ad un cavo, ad un calcolatore allora molto diffuso: il Commodore 64. Le nostre classi furono le prime a poterla utilizzare in Italia e tuttora un vecchio Commodore 64 con cavo, interfaccia Lego, motori e sensori è perfettamente funzionante nel nostro laboratorio. Nel Commodore 64 esiste una porta apposita (la porta USER), situata nella parte anteriore, realizzata per collegare il calcolatore ad apparati esterni. Anche in questo caso un registro della RAM del calcolatore (il **56577**) è addetto allo scambio di segnali attraverso questa porta mentre un altro registro (il **56579**) è addetto a stabilire il flusso dei dati: serve cioè a stabilire se i singoli bit del 56577 devono essere utilizzati in uscita (per inviare corrente ai motori) o in entrata (per leggere il segnale proveniente dai sensori.

Da allora sono state realizzate varie altre interfacce tra cui **Andrea C1** (realizzata nel 2004 da Marco Carminati genitore di un alunno della nostra scuola). Essa è collegata al calcolatore grazie alla porta parallela (che un tempo collegava il calcolatore alla stampante) e utilizza i due registri **888** per l'uscita, e **889** per l'ingresso di dati).
Dal 2008 abbiamo in dotazione la NIOM 32, sostituita nel 2011 dalla scheda MZ. Questa interfaccia per comunicare con il calcolatore utilizza le **porte USB** e dunque non vi sono registri fissi per comunicare con essa dal calcolatore. Infatti le porte USB funzionano con un principio simile a quello che abbiamo visto per le variabili. Al momento della loro attivazione il calcolatore cerca dei registri (byte) della memoria RAM liberi, da utilizzare per la gestione dei dati in ingresso e in uscita e memorizza il loro indirizzo. Noi, tramite apposite istruzioni, daremo l'ordine di mettere a 0 o ad 1 i bit che controllano i motori o di leggere lo stato dei bit che ricevono il segnale dai sensori. Ci penserà poi il calcolatore a rintracciare all'interno della RAM i registri e fare il lavoro richiesto.

Come per tutte le periferiche, anche per gestire la scheda MZ il calcolatore ha bisogno di un apposito software (driver) che dovrà essere installato la prima volta che verrà inserito il cavo proveniente dalla scheda in una delle porte USB del computer. In proposito vedere l'apposita scheda.

COME LAVORA UN ROBOT (motori)

La connessione

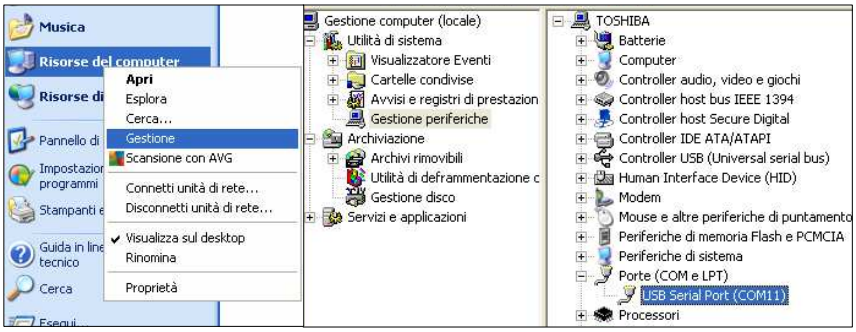
Come si è detto MZ va collegata al calcolatore utilizzando una delle porte USB. E' bene farlo seguendo questa procedura:

- accendere il calcolatore
- inserire nella porta USB il cavo di MZ senza alimentare quest'ultimo
  - sulla piastra centrale di MZ si accende una luce verde. Segnala che sta ricevendo il segnale dal calcolatore tramite il cavo USB
- alimentare MZ
- avviare il programma **libbase**

Se vi sono segnali che indicano problemi nel collegamento togliere e rimettere il cavo USB riavviando il programma.

Nel programma ***libbase.bas*** è già inserito l'ordine:
call nconnect __ che attiva la connessione con MZ tramite la porta USB
questo ordine deve essere seguito dal n° della porta USB che stiamo utilizzando.
Lo si ottiene con:
Risorse del computer *(tasto destro del mouse)* **> *Gestione***

Nell'esempio l'ordine sarà:
call nconnect 11



Mentre con:
call ndisconnect che disattiva la connessione con MZ tramite la porta USB

Dopo l'ordine **nconnect** è normalmente inserito **call nwrite 9,1** di cui ci occuperemo in seguito.
La parte del programma che gestirà l'interfaccia di robotica dovrà essere compreso tra questi ordini.
Gli ordini **nconnect** e **disconnect** sono in realtà nomi di procedure come è evidente visto che sono preceduti dall'ordine di chiamata call.

Il programma ***libbase.bas***

```
global v, s  ' in s c'è il valore analogico e in v c'è il valore binario (0 e 1) / 500 è ….
call nconnect 13 '13 è il n°della porta USB, va letto in risorse de l computer / gesti…..
call nwrite 9,1 'il bit 9 va messo a 1 per abilitare i motori
'---

'---
call ndisconnect ' chiude la connessione on MZ
end
```

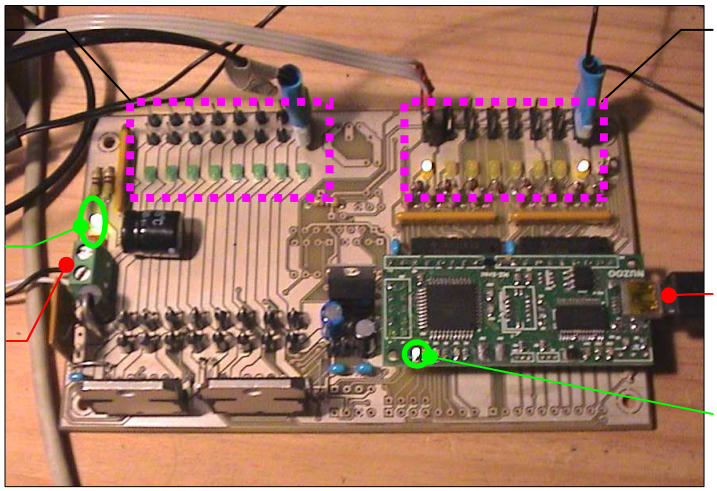
In fondo al programma *libbase*, dopo la scritta NON MODIFICARE, vi è la parte di codice che contiene le procedure che gestiscono la comunicazione con MZ e che servono anche per verificare che i valori provenienti dai sensori non contengano errori. La comunicazione con MZ avviene con procedure simili a quelle che abbiamo già visto per comunicare con l'hard disk e che, in realtà, sono sempre utilizzate quando il calcolatore deve comunicare con una periferica (apertura di un canale di comunicazione con Open / ricezione e trasmissione dei singoli dati con input# e print#).
Per semplificare il nostro lavoro, sono state sudiate queste procedure per rendere più semplice la comunicazione con MZ. Noi potremo utilizzare i nomi delle procedure per controllare i motori e per ricevere i valori dei sensori. Nel caso dei sensori vengono utilizzate due variabili, che dunque sono state rese globali, per ospitare il valore letto dal sensore (s) e per rendere binario tale valore (v) a seconda se il valore letto supera una determinata soglia (che è modificabile).

I nomi delle procedure (che noi utilizziamo come ordini) sono:
nwrite che consente il controllo dei motori
nread che consente la ricezione dei valori provenienti dai sensori
pausa che consente di stabilire un tempo di attesa in millesimi di secondo e che semplifica il controllo del tempo rispetto agli ordini standard di Liberty.
Ad esempio call pausa 1000 provoca un attesa di 1 secondo

zona di output
otto bit controllano l'alimentazione dei motori. Le luci verdi accese indicano che il bit è a 1 e che dunque il motore è alimentato

questa luce indica che la scheda è alimentata

questi due cavi alimentano la scheda la loro posizione non è indifferente e l'accensione della luce ne indica la posizione corretta



zona di input
otto byte raccolgono i valori provenienti dai sensori (da 0 a 1024). L'intensità della luce gialla è proporzionale al valore in arrivo.

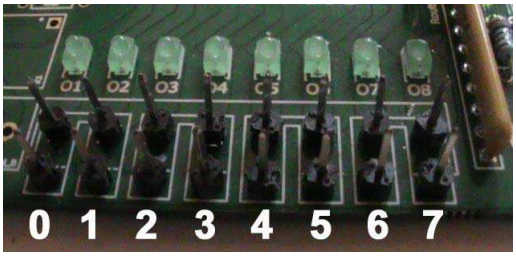
cavo proveniente dalla porta USB del calcolatore

questa luce indica che la scheda è collegata al calcolatore

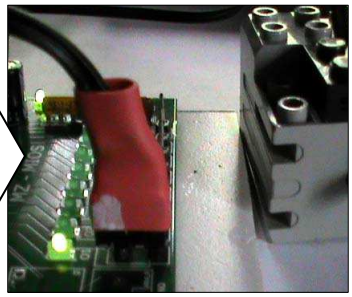
Il controllo dei motori con un byte da 8 bit

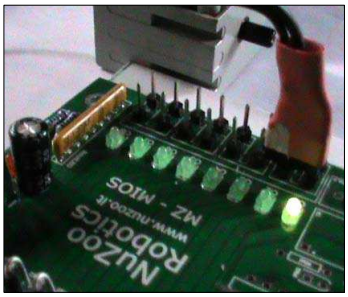
Come sappiamo, nella RAM del computer vi sono otto bit addetti al controllo delle porte che, su MZ, devono fornire corrente ai motori. Questo però non avviene direttamente. Nel chip chiamato "driver output", la debole corrente proveniente dal calcolatore, apre o chiude un altro circuito che alimenta i motori utilizzando la corrente proveniente da un apposito trasformatore. L'avvenuta alimentazione è segnalata dall'accensione di un apposito led di color verde.

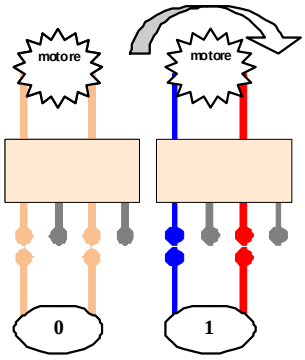
Nell'immagine a fianco è illustrata la numerazione delle porte che controllano i motori. Ogni porta ha due prese destinate all'alimentazione del motore. Quella in alto, se il bit è a 1, viene collegata al polo carico.



A fianco possiamo vedere le due modalità di controllo dei motori. Collocando lo spinotto su una porta è possibile alimentare o non alimentare il motore. Disponendo lo spinotto a cavallo tra due porte è anche possibile invertire il flusso della corrente e dunque il movimento del motore. Osservare con attenzione la disposizione degli spinotti.

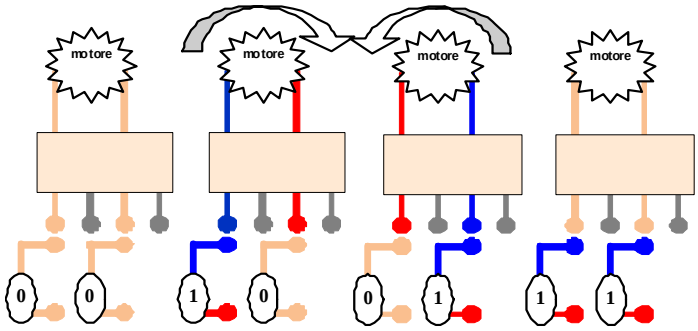






Nelle immagini a fianco abbiamo schemi che illustrano il funzionamento del collegamento tra lo spinotto che alimenta il motore e una porta controllata da un bit. Dei quattro ingressi dello spinotto solo due vengono utilizzati per alimentare il motore (il colore blu indica in collegamento al polo carico e il colore rosso il collegamento al polo meno carico). Il colore rosa indica l'assenza di differenza di potenziale.
Se il bit che controlla la porta è a 0 non vi è differenza di potenziale e il motore resta fermo. Se il bit è a 1 si attiva una differenza di potenziale e il polo carico alimenta il motore.

Ora vediamo la situazione che si crea quando collochiamo lo spinotto a cavallo tra due porte in modo che il motore possa essere controllato da due bit.



- entrambi i bit sono a 0: le prese non sono alimentate dunque il motore resta **fermo**
- il primo bit è a 1, il secondo a 0: dalla presa del primo bit parte un flusso di elettroni diretto alla presa del secondo bit alimentando il motore (**avanti**).
- il primo bit è a 0, il secondo a 1: dalla presa del secondo bit parte un flusso di elettroni diretto alla presa del primo bit. Il flusso di corrente è invertito e dunque anche il movimento del motore (**indietro**).
- entrambi i bit sono a 1: entrambe le prese sono cariche e dunque non vi è differenza di potenziale. Non vi è spostamento di elettroni e il motore resta **fermo**.

L'abilitazione delle porte di uscita

Dopo aver collegato la scheda, dovremo abilitare le porte di uscita. Infatti in MZ, oltre agli 8 bit utilizzati per il controllo dei singoli motori esiste un altro bit (il 9) dove bisogna inserire 1 per poter poi attivare i motori. Di conseguenza, quando uno o più motori sono accesi, mettendo 0 in questo bit si spegne tutto.
Dunque con:

call nwrite 9,1 vengono abilitate le porte di uscita
call nwrite 9,0 vengono disabilitate

L'avvio dei motori

Per mettere in moto un motore bisognerà collocare **1** nel bit ad esso corrispondente. Contrariamente a quanto avveniva nelle schede precedenti, MZ permette l'assegnazione di ogni singolo bit.
In Liberty Basic si utilizzerà l'ordine:

numero della porta da assegnare
numero collocare nel bit (0 oppure 1)

call nwrite __ , __

Ad esempio per alimentare il primo motore (bit 0) si userà call nwrite 0, 1 mentre *per invertirne il movimento bisogna azzerare il bit 0 e mettere uno nel bit 1*

call nwrite 0,0 : call nwrite 1,1

Dopo questa assegnazione avremo:

registro di output

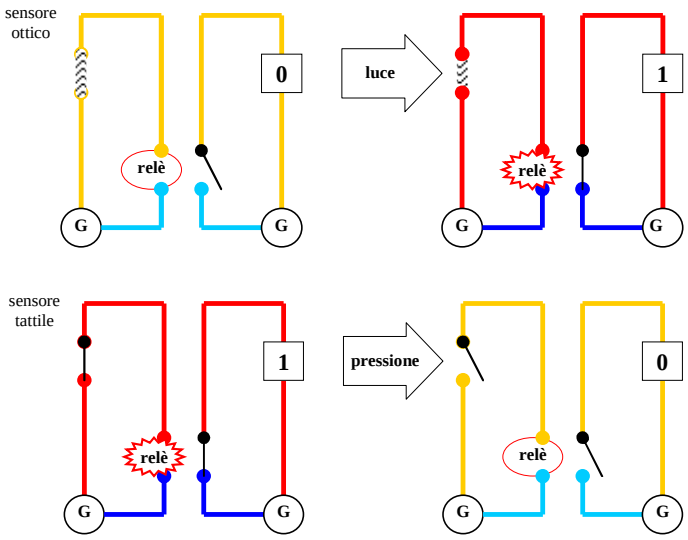
bit	4° motore		3° motore		2° motore		1° motore	
	7 alim	6 inv	5 alim	4 inv	3 alim	2 inv	1 alim	0 inv
	0	**0**	**0**	**0**	**0**	**0**	**1**	**0**

Negli esempi vedi i programmi ***avvia.bas***, ***inverti.bas***

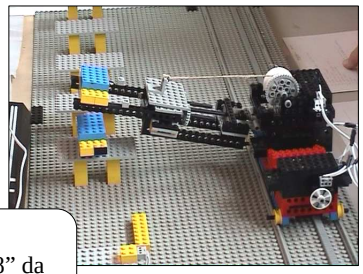
COME LAVORA UN ROBOT (sensori)

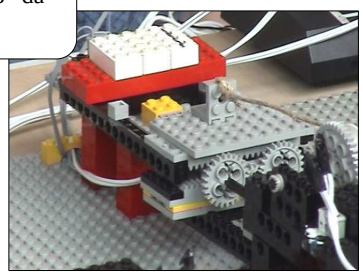
L'uso di sensori digitali (0-1)

I sensori sono delle apparecchiature collocate lungo un circuito elettrico. Queste apparecchiature, a causa di un evento esterno (può essere la pressione di un pulsante, l'arrivo di una fascio di luce intensa, ecc…) possono aprire o chiudere il passaggio di una corrente elettrica diretta verso un relè.
Questo relè, a sua volta, controlla l'apertura o la chiusura del circuito diretto verso il calcolatore e da cui dipende lo stato del bit (0 = non corrente / 1 = corrente).



Il **sensore tattile** funziona come un interruttore che, se premuto, interrompe il contatto dei due fili provenienti dai due poli e dunque disattiva il passaggio di corrente.
Nel **sensore ottico** i fili provenienti dai due poli sono collegati alle opposte estremità di una piastrina realizzata con un materiale sensibile alla luce (fotocellula).
Questo è normalmente un materiale isolante ma se viene colpito da una luce diventa conduttore. Avremo cosi un passaggio di corrente proporzionale alla quantità di luce ricevuta dal sensore .

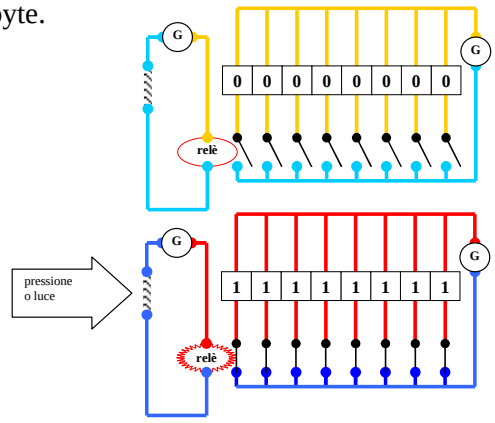


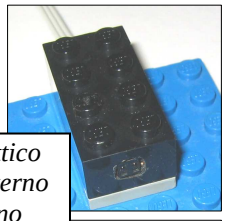


Maggio 2001 - il progetto presentato a "Scienza Under 18" da un gruppo di alunni di terza.

L'uso di sensori analogici (0-255)

Vengono chiamati, anche se impropriamente, ***analogici*** quei sensori che, invece di poter ricevere solo corrente (1) o non riceverla (0), possono ricevere e misurare diverse quantità di corrente che vanno da un minimo (che normalmente è 0) ad un massimo (che in genere è fissato a 255). E' evidente che per poter funzionare questi sensori devono poter disporre non di un solo bit ma di un intero byte.
Essi vengono realizzati con materiali che lasciano passare una quantità diversa di corrente a seconda della situazione in cui si trova ad operare (pressione, quantità di calore, quantità di luce). Gli schemi a fianco ne illustrano sinteticamente il funzionamento.



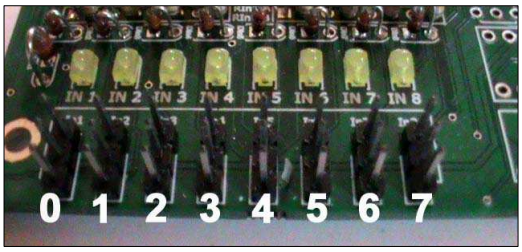


il sensore ottico visto dall'esterno e all'interno

L'uso di sensori con MZ

Le porte utilizzate con le interfacce precedenti permettevano l'utilizzo di un solo byte per la ricezione dei segnali dai sensori. Questo rendeva possibile solo l'utilizzo di sensori digitali del tipo visto sopra.
MZ, grazie all'utilizzo della porta USB, permette l'utilizzo di un byte da 10 bit per ognuna delle 8 connessioni (numerate da 0 a 7) che ricevono i segnali provenienti dai sensori. Esso dunque permette di utilizzare non solo sensori digitali (che daranno solo i due valori estremi) ma anche vari tipi di sensori analogici, come i sensori di luminosità che misurano la quantità di luce ricevuta, quelli termici che misurano la quantità di calore, quelli di prossimità che misurano la distanza di un oggetto.

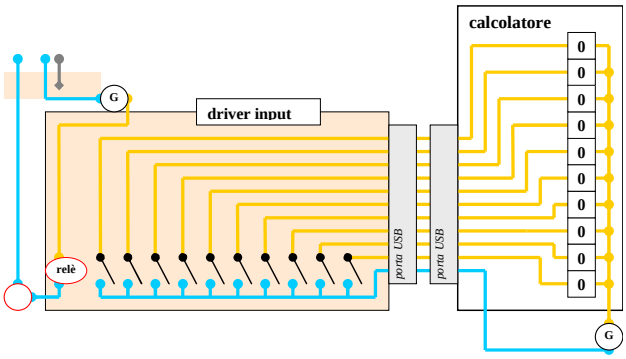
In MZ il chip "driver input" è utilizzato per raccogliere i segnali provenienti da 8 sensori e dirigerli, grazie alla porta USB, verso gli 8 byte dei **registri di input**.



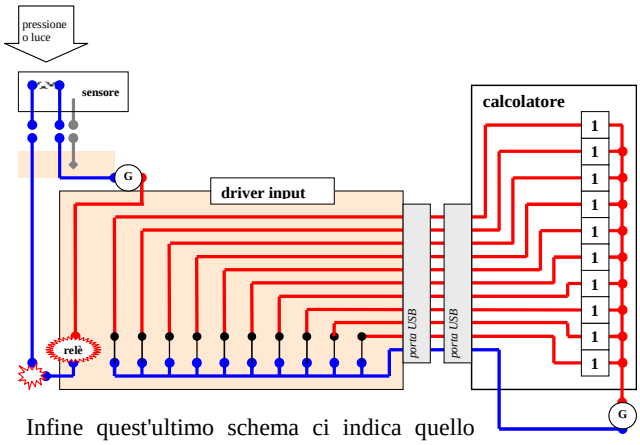
Nell'immagine a fianco è illustrata la numerazione delle porte che controllano i sensori. Ogni porta ha tre prese. Due controllano il flusso di elettroni da e per il sensore, ma è necessaria anche una terza.

Gli schemi che seguono illustrano, con qualche semplificazione, il funzionamento di una delle 8 connessioni addette a ricevere i segnali dai sensori .

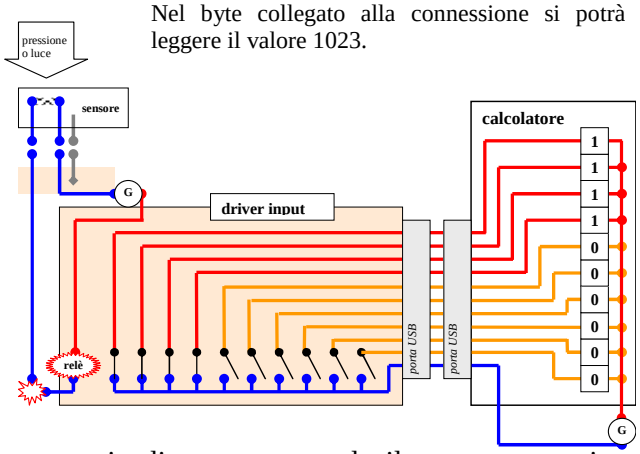
A destra lo schema illustra lo stato di una delle connessioni quando nel circuito iniziale non circola corrente. Il relè del *driver di input* rimane non attivo e dunque non circola corrente neanche nei 10 circuiti diretti verso il calcolatore.



Nella seconda immagine abbiamo un sensore ottico collegato a una delle connessioni. Riceve il massimo della luce. Il relè riesce così ad attivare tutti i circuiti diretti verso il calcolatore.
Nel byte collegato alla connessione si potrà leggere il valore 1023.



Infine quest'ultimo schema ci indica quello che succede nei circuiti se la luce che arriva al sensore ottico è meno intensa.
La minore quantità di elettroni che circola rende meno forte la capacità di attrazione del relè. Solo i ponticelli dei circuiti più vicini verranno attirati chiudendo i loro circuiti. Nel byte collegato a questo sensore si potrà leggere un valore intermedio tra 0 e 1023.



Nel **sensore tattile** che utilizziamo si ha il massimo passaggio di corrente quando il sensore **non** viene premuto. La pressione non interrompe completamente la corrente ed infatti a sensore premuto si possono ottenere valori intorno a 16-18.
Nel sensore ottico il valore massimo si ottiene in presenza di una forte fonte di luce oppure di una superficie bianca, specie se vicina. Si scenderà verso valori più bassi man mano che l'ambiente diventerà più scuro ma, anche in questo caso è difficile arrivare a 0.
Bisogna anche tenere presente che il sensore ottico è anche sensibile al calore e dunque è possibile un aumento dei valori in prossimità di corpi caldi (ad es. un dito).

In Liberty Basic per leggere il contenuto di un registro (o byte) e collocarlo dentro una variabile verrà utilizzato l'ordine:

chiamata di procedura
numero della connessione da leggere

```
call nread
```

ad esempio: con call nread 0: print s
il calcolatore esegue la lettura dalla prima connessione (numero 0) e scrive il valore letto. Il valore analogico letto è memorizzato nella variabile s

Negli esempi vedi il programma ***leggi.bas***

I PROGRAMMI DI ROBOTICA

Controllo dei motori grazie all'utilizzo di sensori

Nella nostra realtà quotidiana siamo oramai circondati da microprocessori che utilizzano i segnali provenienti da sensori per accendere e spegnere i motori (ad esempio le porte del supermercato che si aprono grazie alla variazione di luminosità provocata dalla persona che arriva davanti alla porta o le porte del bus che si aprono alla pressione di un pulsante).
In tutti questi casi l'evento (accensione o spegnimento di un motore) è provocato dalla variazione dello **stato** di un sensore. Per individuare questa variazione (ad esempio un sensore tattile che viene premuto) si potrà leggere il byte che riceve il segnale dal sensore.
Ad esempio con:

```
'---
call nread 0 : a=v : b=a
do  while a=b
     call nread 0: b=v
loop
'---
```

immagine del sensore tattile

il programma ***variazione.bas*** prevede una lettura del sensore (porta 0) e una ripetizione della lettura sino a quando il valore letto non cambia

call nread 0 - viene letto il valore proveniente dal sensore collocato sulla porta 0
a = v - il valore digitale letto viene collocato nella variabile a (che contiene la lettura iniziale)
b = a - il valore presente in a viene anche copiato in b (altrimenti si uscirebbe subito dal ciclo visto che la variabile b ora contiene 0
do while a = b - ciclo: si resta nel ciclo sino a quando i valori presenti in a e in b sono uguali
call nread 0 - ciclo: viene letto il valore proveniente dal sensore collocato sulla porta 0
b = v - ciclo: il valore digitale letto viene collocato nella variabile b (che contiene la lettura ripetuta)
loop - fine del ciclo

Come abbiamo visto nell'esempio, il controllo va effettuato sulla lettura digitale e non su quella analogica visto che quest'ultima è molto sensibile e fornisce piccole variazioni anche se non vi è stata alcuna variazione nello stato del sensore.

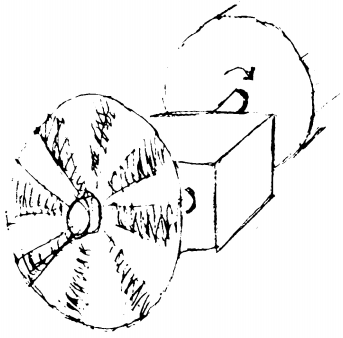
Con il seguente programma (vedi programma ***premi.bas*** negli esempi) il calcolatore accende il primo motore (bit 0) e lo spegne quando vi è una variazione del tattile (porta 0).

```
'---
call nread 0  : a = v : b = a ' lettura iniziale
call nwrite 0,1 'accende il motore della porta 0
do while a = b
      call nread 0 : b = v ' lettura ripetuta
loop
call nwrite 0, 0 'spegne il motore
'---
```

è importante ricordare che la lettura del sensore va collocata prima dell'assegnazione del motore

Stabilire i tempi di accensione dei motori contando le variazioni dei sensori

Il controllo dei tempi di accensione dei motori mediante l'orologio interno del calcolatore (TIME$) presenta sempre molti problemi. Infatti raramente agli stessi tempi di accensione corrisponde un identico numero di giri del motore.
E' molto più efficace utilizzare un'apposita rotellina divisa in fasce bianche e nere collegata all'asse del motore. Il passaggio delle fasce davanti ad un sensore provoca continue variazioni nel bit ad esso collegato. Il programma dovrà contare i passaggi dal bianco al nero e dunque le rotazioni dell'asse del motore.
Visto che il colore bianco ravvicinato dà sempre un valore superiore a 500 mentre una zona più scura dà sempre valori inferiori a 500 ma diversi dovremo rendere binario ciò che non è, visto che, altrimenti, il sensore rischierebbe di variare anche all'interno della fascia nera.



Per risolvere questo problema la procedura nread fornisce, come abbiamo visto, il valore digitale memorizzato dentro la variabile v. In questo modo eseguendo call nread potremo trovare nella variabile v un valore binario (0 oppure 1) più adeguato alle nostre necessità.
Con il seguente programma (vedi programma ***contavar.bas*** negli esempi) il calcolatore chiede il numero di variazioni massimo. Poi accende il motore e lo tiene acceso sino a che il sensore avrà contato quel numero di variazioni:

```
'---
input "scrivi il numero di variazioni"; vmax
call nread 0 :  a = v : b = v ‘ lettura iniziale
call nwrite 0, 1 : var = 0 ‘ accende il motore e azzera il contatore di variazioni
do while var < vmax  ‘confronta il n°attuale di variazioni con il n°ma ssimo
        do while a = b
            call nread 0 : b = v :  ‘ lettura ripetuta
        loop ‘chiude il ciclo
        var = var + 1 ‘incrementa il contatore
        a = b  ‘mette in a (lettura iniziale) il valore appena letto prima di entrare
               nel nuovo ciclo
loop ‘chiude il ciclo
call nwrite 0, 0
'---
```

Programmare utilizzando la programmazione strutturata

Grazie alla programmazione strutturata è possibile scrivere una sola procedura ed attivarla ogni volta che il calcolatore deve azionare uno dei motori collegati a MZ. In essa i dati che possono cambiare (numero da assegnare al driver di output, che indicano il byte del registro di input da leggere o il numero massimo delle variazioni) vengono sostituiti da variabili il cui valore proviene dalla chiamata di procedura.
Le procedure che seguono sono alla base dei programmi di robotica. In un programma complesso, possono essere collocate in un sottoprogramma e chiamate da qualsiasi altra parte del programma.

motori
Ad esempio (vedi negli esempi il programma ***muovi.bas***) con:

```
call muovi 0, 0, 4
```

Viene chiamata la procedura muovi assegnando 0 a pm, 0 a ns, 4 a vmax

```
sub muovi  pm, ns, vmax
call nread ns ' lettura iniziale
a = v : b = v : var = 0
call nwrite  pm, 1 ' corrente alla porta pm
do while var < vmax   'confronto il n°attuale di variazioni con il n°ma ssimo
        do while a = b 'rimane nel ciclo finché i valori delle variabili sono uguali
            call nread ns : b = v ' lettura ripetuta
        loop
        var = var + 1 : a = b
loop
call nwrite pm, 0
end sub
```

nel quale:
pm = ***memorizza il n° della porta del motore***
ns = ***numero della porta del sensore***
vmax = ***numero massimo di variazioni***
var = ***contatore delle variazioni***

solo sensori
Se dobbiamo controllare solo un sensore (vedi negli esempi il programma ***attendi.bas***):

```
call attendi 0, 4
```

Viene chiamata la procedura attendi assegnando 0 a ns, 4 a vmax

```
sub attendi  ns, vmax
call nread ns ' viene letto il sensore ns
a = v : b = v : var = 0
do while var < vmax   'confronto il n°attuale di variazioni con il n°ma ssimo
        do while a = b 'rimane nel ciclo finché i valori delle variabili sono uguali
                call nread ns : b = v ' b=lettura attuale
        loop
        var = var + 1 : a = b 'incremento il contatore e metto in a il valore letto
loop
end sub
```

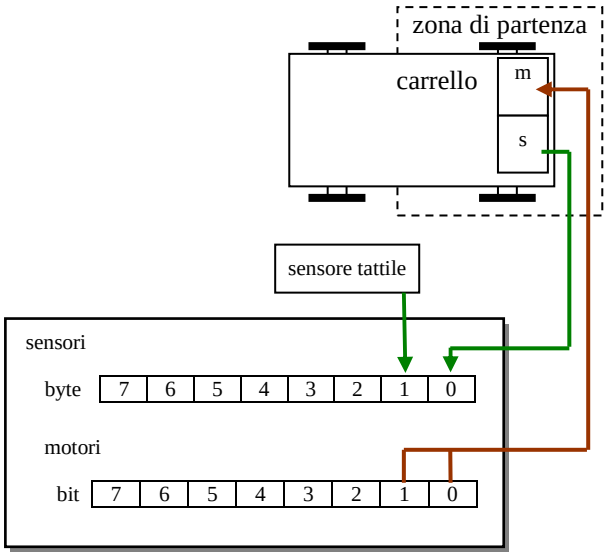
Vediamo ora di applicare le nostre conoscenze ad un problema concreto
(vedi negli esempi il programma ***carrello.bas***).
Abbiamo un carrello con un motore collegato alle porte 0 (avanti) e 1 (indietro) e con un sensore ottico (porta 0) che ne conta le variazioni. Alla pressione di un sensore tattile (porta 1) si avvia verso la zona di arrivo. Qui effettuerà una sosta di 5 secondi per poi tornare alla zona di partenza.

schema di utilizzo

carrello
per raggiungere l'area di arrivo sono necessarie 6 variazioni del sensore ottico. Lo stesso per tornare.

motori		**sensori**	
bit 0	**carrello avanti**	**bit 0**	**variazioni motore**
bit 1	**carrello indietro**	**bit 1**	**tattile avvio**

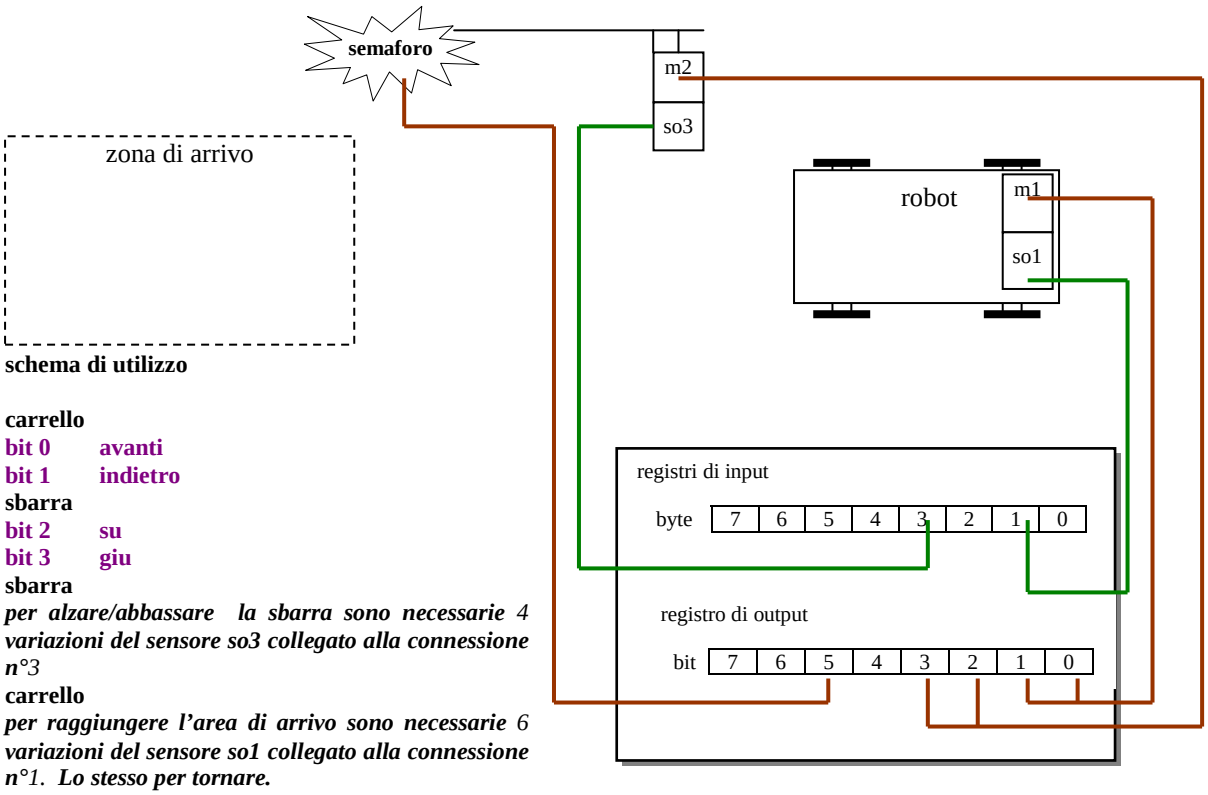
```
global v, s
call nconnect 13
call nwrite 9,1
'---
call attendi 1, 1
call muovi 0, 0, 6
call pausa 5000
call muovi 1, 0, 6
'---
call ndisconnect
end
```




```
' --- muovi
sub muovi  pm, ns, vmax
call nread ns : a = v : b = v : var = 0
call nwrite  pm, 1
do while var < vmax
        do while a = b
                call nread ns : b = v
        loop
        var = var + 1 : a = b
loop
call nwrite pm, 0
end sub
' --- attendi
sub attendi  ns, vmax
call nread ns : a = v : b = v : var = 0
do while var < vmax
        do while a = b
                call nread ns : b = v
        loop
        var = var + 1 : a = b
loop
end sub
```

Problema

*All’avvio del programma si accende il semaforo **(bit 5)** e inizia ad abbassarsi il passaggio a livello (motore **m2** con sensore contagiri **so3)**. Quando si è abbassato parte il treno robot (motore **m1** con sensore contagiri **so1)** e raggiunge la zona di arrivo. Qui si ferma 4 secondi per poi tornare alla zona di partenza. Quando è tornato si rialza il passaggio a livello e si spegne il semaforo.*



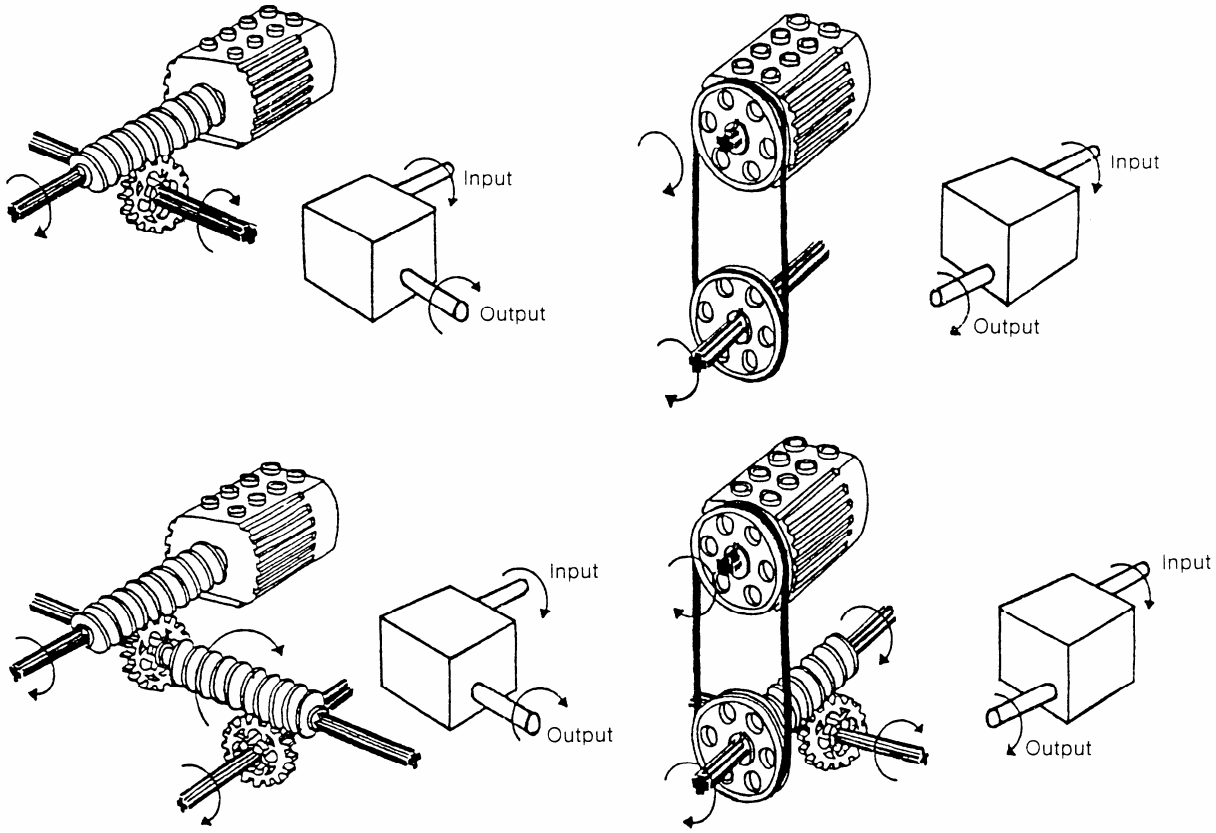
schema di utilizzo

carrello
bit 0 **avanti**
bit 1 **indietro**
sbarra
bit 2 **su**
bit 3 **giu**
sbarra
***per alzare/abbassare la sbarra sono necessarie** 4 **variazioni del sensore so3 collegato alla connessione n°**3*
carrello
per raggiungere l’area di arrivo sono necessarie** 6 **variazioni del sensore so1 collegato alla connessione n°**1. **Lo stesso per tornare.

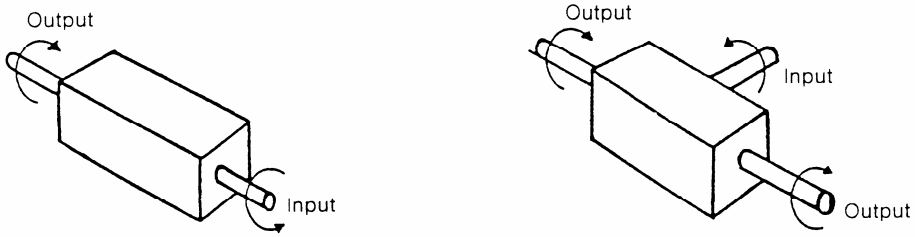
Scrivere qui sotto il programma:

Alcune idee per il lavoro in classe

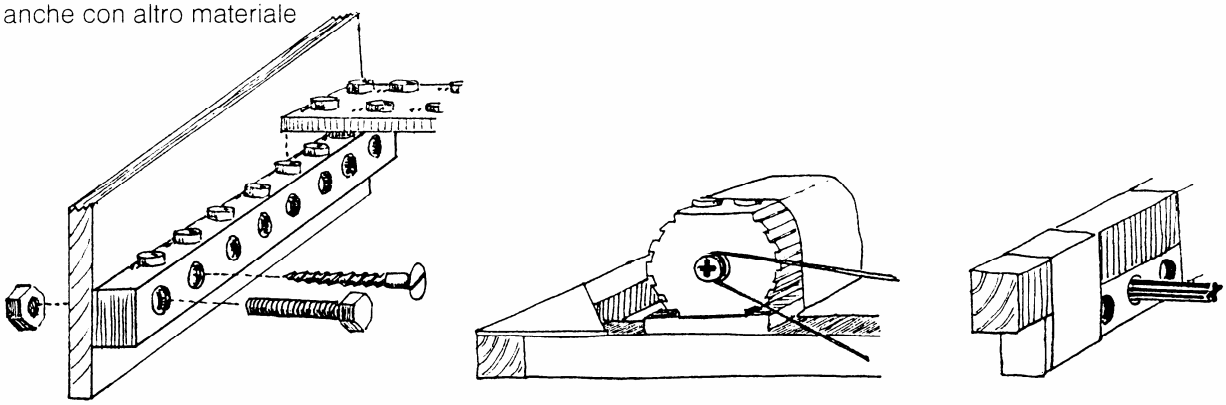
Le scatole degli ingranaggi



Cercate di costruire questi ingranaggi nascosti

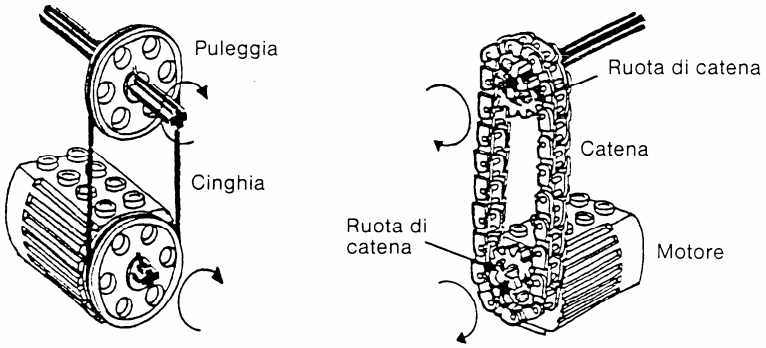


I componenti LEGO Technic possono essere utilizzati anche con altro materiale

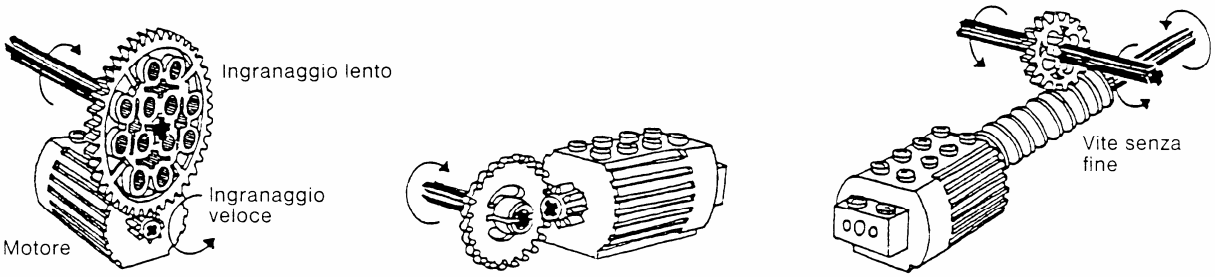


Diversi sistemi di trasmissione del moto

1 senza cambiamento di direzione



2 con cambiamento di direzione



3 con cambiamento di velocità