RoboQuaderno

Schema generale di un sistema di controllo
Parte II : Teoria del controllo PID applicata ai robot Lego Mindstorms

Luca e Paolo Capobianco
luglio 2016

Sommario

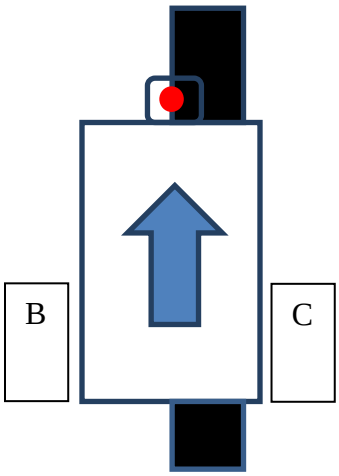
Il quaderno traduce e amplia il testo dell'articolo "*A PID Controller For Lego Mindstorms Robots*" di J. Sluka, il cui testo originale in inglese è reperibile via web all'indirizzo http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html, aggiornando i contenuti originali con trasposizione sul mondo LEGO Mindstorms.

Teoria del controllo PID applicata ai robot Lego Mindstorms

Il controllo PID è una tecnica comunemente utilizzata per il controllo automatico delle apparecchiature e, nonostante non sia immediata la sua spiegazione matematica, il suo uso pratico è abbastanza semplice.

Il controllo PID può essere utilizzato quando abbiamo una grandezza che deve rimanere costante ed uguale a un valore predefinito (riferimento), ed abbiamo la possibilità di misurare la grandezza con un sensore e confrontare la misura con il valore di riferimento. La grandezza da misurare potrebbe essere una temperatura, una distanza, un angolo di rotazione,... Nel seguito consideriamo come esempio di applicazione del controllo PID un inseguitore di linee, ma i concetti potrebbero essere applicati a un robot che deve muoversi mantenendo una certa distanza dalla parete, un certo angolo di rotazione, ...

Immaginiamo di avere un robot con due motori B e C, collegati alle rispettive ruote, e di volere seguire il bordo sinistro di una linea nera di una certa larghezza utilizzando un sensore di luce.



Il cerchio rosso rappresenta la zona illuminata dal sensore di luminosità e la freccia indica la direzione di marcia.

Per seguire il bordo della linea il sensore deve vedere bianco e nero in parti uguali, se il sensore vede più nero che bianco (o tutto nero) è necessario curvare a sinistra mentre se il sensore vede più bianco che nero (o tutto bianco) è necessario curvare a destra.

> *Nota: se si volesse impostare l'inseguimento del bordo destro sarebbe necessario invertire le manovre di correzione (curva a destra se il sensore vede nero, curva a sinistra se il sensore vede bianco. Non sarebbe possibile, almeno con un solo sensore, impostare l'inseguimento del centro della linea nera visto che, nel momento che il sensore vedesse il bianco, non sarebbe possibile sapere se curvare a destra o a sinistra per correggere la posizione.*

Il sensore di luminosità fornisce un valore numerico che in teoria è 0 per il nero e 100 per il bianco. In pratica, il sensore fornirà un valore più basso per la zona scura della linea e un valore superiore per la zona chiara al di fuori della linea.

Consideriamo comunque il caso ideale in cui ho una linea perfettamente nera (lettura 0) e una zona esterna alla linea perfettamente bianca (lettura 100). Quando il sensore sarà perfettamente al centro del bordo della linea vedrà una zona bianca e una zona nera di uguali dimensioni e quindi fornirà un valore numerico pari a 50:

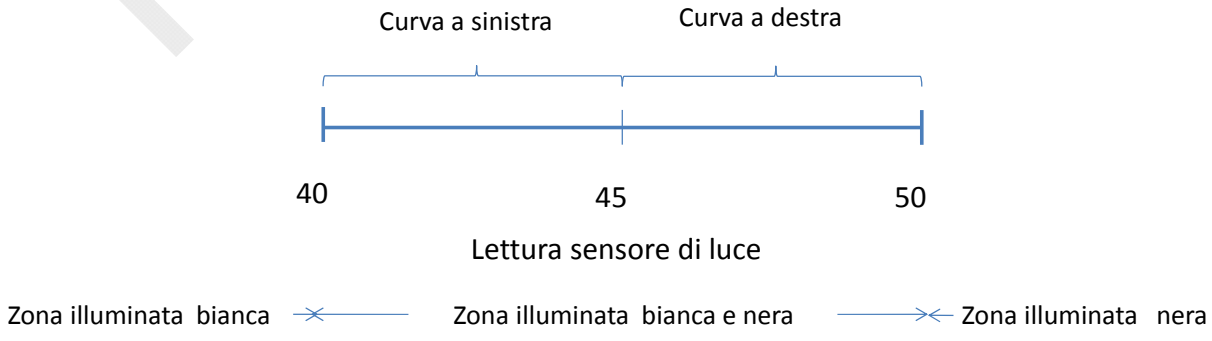
(LivelloNero + LivelloBianco) : 2 = (0+100)/2 = 50 (riferimento)

Spostandosi verso il centro della linea i valori passeranno da 50,49,48,, fino ad arrivare a 0 quando l'area illuminata del sensore sarà tutta nella zona nera, mentre allontanandosi dalla linea i valori passeranno da 50,51,52 fino a arrivare a 100 quando l'area illuminata dal sensore sarà tutta nella zona bianca.

Controllo a tre posizioni

Indicando su un grafico i valori letti, si vede come in pratica si potrebbe impostare un semplice controllo in grado di permettere al robot di seguire il bordo sinistro della linea nera.

Quando il valore letto è 50 il robot dovrà andare dritto mentre per valori compresi tra 50 e 0 dovrà girare a sinistra per allontanarsi dalla linea, e riportare il valore letto a 50, mentre per valori compresi tra 50 e 100 dovrà girare a destra per avvicinarsi alla linea e riportare anche in questo caso il valore a letto 50.



In pratica il controllo avrà tre stati distinti:

Lettura 0-50	Curva a sinistra	Maggiore potenza sulla ruota destra (es.50 destra, 30 sinistra)
Lettura 50	Dritto	Stessa potenza su entrambe le ruote (es.40)
Lettura 50-100	Curva a destra	Maggiore potenza sulla ruota sinistra (es.30 destra, 50 sinistra)

e il robot saprà solo andare dritto, curvare a destra (sempre con la stessa velocità) e curvare a sinistra (sempre con la stessa velocità).

Se la linea nera è dritta, e le differenze tra le potenze dei motori sono impostate per delle curve lente, il robot può riuscire a seguire bene la linea ma se la linea nera curva il controllo mostra i suoi limiti: o il robot non riesce a seguire la linea o, impostando i motori per delle curve veloci, oscilla intorno alla linea nera.

> *Nota: Se invece di un singolo valore di offset si considera un intervallo entro il quale il robot deve andare dritto (es. 45-55), il controllo funziona meglio ma è ancora poco efficace ed è sempre necessario impostare manualmente il tipo di curva in funzione delle caratteristiche della linea nera (dritta, con curve,…)*

Controllo proporzionale (P)

Il limite del controllo a tre posizioni è che corregge la direzione sempre nello stesso modo, mentre sarebbe necessario avere correzioni diverse in funzione di quanto ci siamo spostati dalla zona ottimale di lettura 50; una curva lenta per valori che si discostano poco da 50 e una curva via via più veloce quanto più mi allontano da 50.

Questo concetto introduce l'idea del controllo proporzionale che si basa proprio sul fatto di permettere delle correzioni che varino automaticamente in proporzione diretta con l'aumento o la diminuzione della differenza tra il valore richiesto (nell'esempio illustrato 50) e il valore letto. Piccoli valori della differenza tra il valore richiesto (riferimento) e il valore letto devono determinare piccole/lente correzioni, mentre grandi differenze tra il valore richiesto e il valore letto devono determinare delle grosse/veloci correzioni.

In pratica aggiungo 10 sulla ruota destra e tolgo 10 sulla ruota

Riprendiamo il primo esempio del controllo a tre posizioni:

Lettura 0-50	Curva a sinistra	Maggiore potenza sulla ruota destra (es. 50 destra, 30 sinistra)
Lettura 50	Dritto	Stessa potenza su entrambe le ruote (es.40)
Lettura 50-100	Curva a destra	Maggiore potenza sulla ruota sinistra (es.30 destra, 50 sinistra)

In pratica tolgo 10 sulla ruota destra e aggiungo 10 sulla ruota

Invece di aggiungere/sottrarre sempre 10, posso aggiungere/sottrarre un valore che dipende dalla differenza tra la lettura e il valore di riferimento (50 nel nostro esempio), e quindi se la lettura è ad esempio 55 (il robot si è allontanato di poco dalla linea nera ed è necessaria una curva a destra), aggiungerò sulla ruota sinistra e toglierò alla ruota destra una potenza:

$$T = K_p \times (55 - 50) = K_p \times 5$$

mentre se la lettura è 90 (il robot è più lontano dalla linea nera ed è necessario curvare a destra), aggiungerò sulla ruota sinistra e toglierò alla ruota destra una potenza:

$$T = K_p \times (90 - 50) = K_p \times 40$$

K_p è la costante che mi permette di ottenere in pratica la quantità di potenza da aggiungere/sottrarre e va calcolata in funzione del tipo di linea e della potenza impostata per l'avanzamento rettilineo (nell'esempio 40 su entrambe le ruote).

Ad esempio, se volessi fare in modo che per la lettura 100 (il sensore non vede più la linea nera), si avesse l'aggiunta/sottrazione di potenza 10 alle ruote, il valore di K_p dovrebbe essere

$$10 = K_p \times (100 - 50) = K_p \times 50 \quad \Rightarrow \quad K_p = 10/50 = 0.2$$

Quindi le correzioni di potenza negli esempi precedenti sarebbero:

$$T = 0.2 \times (55 - 50) = 0.2 \times 5 = 1$$

$$T = 0.2 \times (90 - 50) = 0.2 \times 40 = 8$$

fino ad arrivare a una correzione:

$$T = 0.2 \times (100 - 50) = 0.2 \times 50 = 10$$ (max correzione possibile con $K_p = 0.2$)

Quindi maggiore è la distanza del robot dalla linea nera, maggiore è la correzione di potenza sui motori (e quindi maggiormente pronunciata e veloce sarà la curva di rientro). Cambiando il valore di K_p si fisserà in pratica il massimo valore di correzione possibile.

Il ragionamento è analogo se il robot tende ad avvicinarsi alla linea nera, fino ad arrivare alla massima correzione:

$$T = 0.2 \times (0 - 50) = 0.2 \times (-50) = -10$$ (max correzione possibile con $K_p = 0.2$)

quando il sensore vede solo la linea nera.

Se si volesse nell'esempio precedente con una lettura 100 del sensore una correzione pari a 60, in modo tale da portare uno dei due motori da 40 a 100 (potenza massima motore), il valore di K_p da impostare sarebbe pari a 1,2:

$$T = 1.2 \times (100 - 50) = 1.2 \times 50 = 60$$

> *Nota: se il range di variazione della lettura del sensore è diverso da 0-100 (come in pratica succede con un lettore di luminosità che, in funzione del colore della linea "nera" e della zona "bianca" potrebbe avere delle letture vicino a 0 per il nero, es. 10, e vicino a 100 per il bianco, es. 80), è necessario calcolare il riferimento come media del valore minimo e massimo, ma è sempre possibile calcolare il valore di Kp necessario ad avere la correzione massima della potenza dei motori con il massimo errore possibile.*

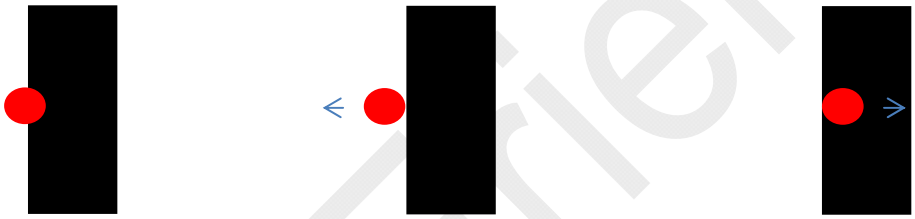
In pratica, il valore K_p deve essere calcolato sperimentalmente iniziando a fare una prova a bassa velocità ed impostando un valore basso di K_p (es. 1): se il robot perde la linea è necessario aumentare il valore di K_p, se il robot oscilla intorno alla linea è necessario ridurre il valore di K_p. Trovato un valore di K_p che ottimizza il comportamento del robot (che sarà il K_p per quella velocità

e per il tipo di linea utilizzata), è possibile provare ad aumentare la velocità e rifare il test di modifica del K_p fino a trovare un altro valore ottimale.

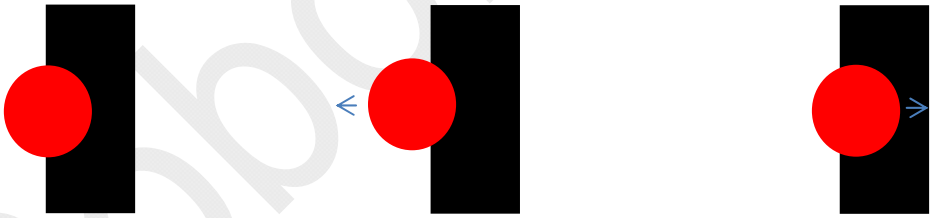
Al di sopra di un certo valore di velocità, soprattutto se la linea ha delle curve, è probabile che non sia possibile avere un comportamento accettabile per qualunque valore di K_p impostato.

Il controllo proporzionale funziona fino a quando la lettura del sensore è all'interno dell'intervallo di lettura: raggiunto e superato il valore minimo o massimo (esempio la zona di lettura del sensore esce completamente dalla linea nera e la lettura diventa costante), la correzione alla potenza dei motori rimane costante e non è più proporzionale alla distanza dalla linea nera.

Nell'esempio dell'inseguitore di linea, per mantenere il più possibile la lettura del sensore di luce all'interno nell'intervallo di lettura (0-100 o in pratica (lettura nero) – (lettura bianco)), è possibile, entro certi limiti, aumentare l'altezza del sensore. Infatti, aumentando l'altezza del sensore di luminosità (vedi figura), aumenta il diametro della zona di lettura del sensore (il cerchio rosso), e il sensore continuerà ad avere una lettura entro l'intervallo (dove il controllo proporzionale funziona) anche con spostamenti maggiori dalla posizione ottimale.



Sensore di luminosità basso



Sensore di luminosità alto

Il sensore di luminosità ha comunque dei limiti di posizionamento in altezza: più è alto e maggiormente è sottoposto all'influenza della luce dell'ambiente (evitabile in parte schermandolo). Inoltre, alzando troppo il sensore si riduce la capacità di distinguere il bianco dal nero (e quindi in pratica si torna a ridurre l'intervallo di lettura).

Un valore ottimale dell'altezza del sensore di luminosità potrebbe essere dell'ordine di 1 cm.

Programmazione controllo proporzionale

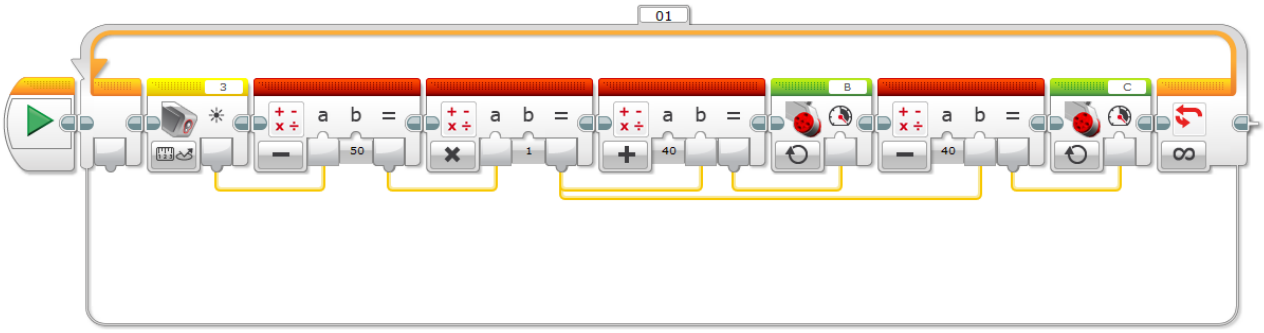
In pratica, il controllo proporzionale può essere impostato con un programma che esegua i seguenti passi:

Kp = 1	Impostiamo il valore di Kp che vogliamo utilizzare (es.1)
riferimento = 50	Impostiamo valore da mantenere costante (media lettura bianco e nero, es.50)
Tp = 40	Impostiamo valore potenza ai motori delle ruote quando il robot non curva (es. 40)
Loop	
LV = lettura sensore	Registriamo la lettura del sensore in una variabile LV (es 55)
errore = LV–riferimento	Calcoliamo l'errore di posizione sottraendo lettura sensore al riferimento (55-50=5)
T = Kp * errore	Calcoliamo la variazione di potenza da applicare ai motori (1x5=5)
PB = Tp + T	Potenza da applicare al motore B a sinistra (40+5=45)
PC = Tp - T	Potenza da applicare al motore C a destra (40-5=40)
end loop	

Mi allontano dalla linea

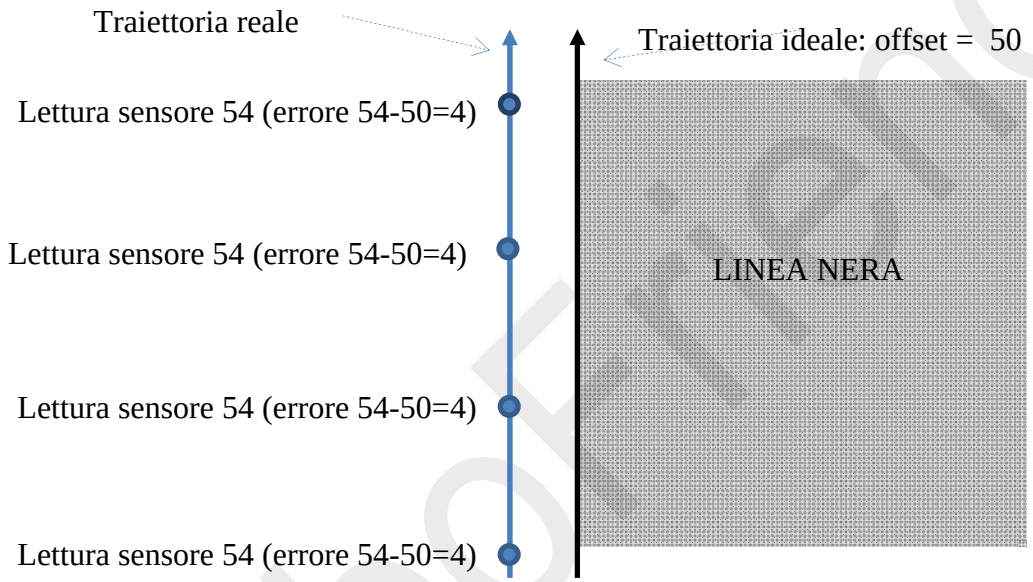
Giro a destra e mi riavvicino alla linea

In pratica, utilizzando il programma Lego Mindstorms il loop (senza definizione delle variabili) potrebbe essere impostato nel seguente modo:



Controllo Proporzionale Integrativo (PI)

Il controllo proporzionale permette di mantenere una piccolo differenza tra il riferimento e il valore letto grazie alla sua capacità di fornire delle correzioni adeguate alla grandezza dell'errore. Se il controllo proporzionale funziona bene, con i corretti valori di Tp e Kp, il valore letto si manterrà in un intorno al riferimento ma quasi sicuramente i due valori non saranno uguali.
Nell'esempio dell'inseguitore di linee, rispetto alla posizione ideale sul bordo della linea (lettura sensore 50, o valore medio tra bianco e nero), il robot potrebbe posizionarsi leggermente a destra o a sinistra (lettura sensore ad esempio 49,48,47,46 o 51,52,53,54)
Supponiamo ad esempio che il robot mantenga una posizione a sinistra rispetto alla posizione ideale: la lettura sarà quindi leggermente superiore a 50 (supponiamo ad esempio 54 e per semplicità costante durante la traiettoria).



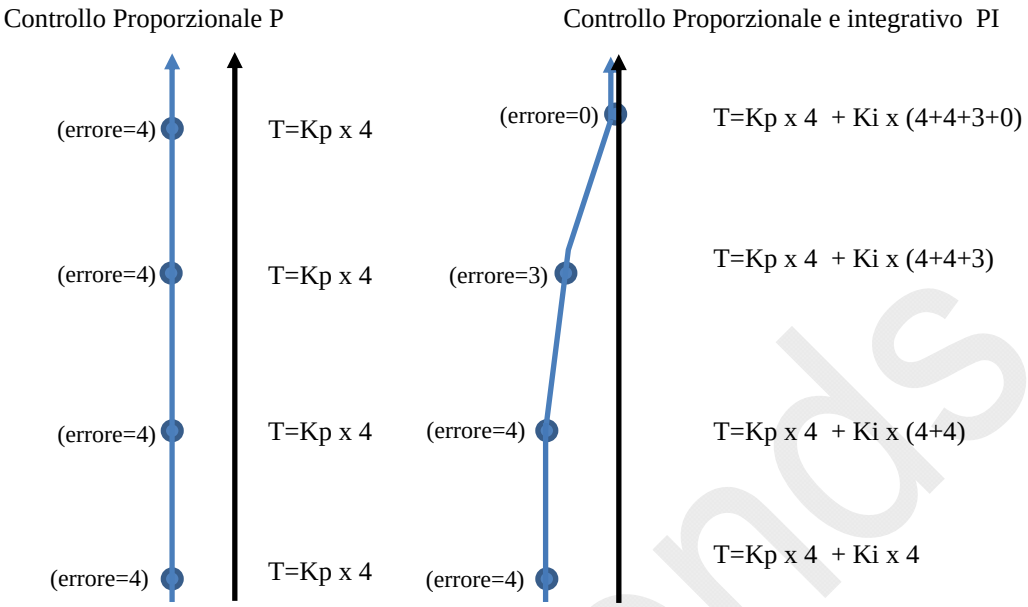
L'errore misurato è costante a ogni ciclo di lettura del controllo e pari a 54-50=4. L'errore è quindi molto piccolo, e probabilmente accettabile nel caso del nostro esempio di inseguitore di linea, ma come si potrebbe ridurre ancora l'errore nel caso avessimo bisogno di mantenere esattamente il valore di offset 50 (e quindi nel nostro esempio la traiettoria ideale)?

Per provare a ridurre ulteriormente l'errore con il controllo proporzionale dovremmo aumentare la costante Kp, ma questo porterebbe ad una traiettoria oscillante dovuta all'incremento eccessivo delle risposte di correzione a parità di errore: in pratica, il controllo proporzionale va benissimo se possiamo accettare un certo errore tra il valore misurato e l'offset ma se l'errore, dopo aver ottimizzato Td e Kp, è ancora troppo alto è necessario aggiungere un altro termine al controllo proporzionale che somma gli errori che vengono misurati in ogni ciclo di lettura e che viene chiamato controllo integrativo:

$$T = K_p \times (\text{errore}) + \mathbf{K_i \times (\text{somma errori})}$$

dove Ki è una costante che ci permette in pratica di calcolare la correzione da apportare ai motori a partire dalla somma degli errori.

Riprendiamo l'esempio precedente in cui l'errore rimane costante e pari a 4.



Se avessimo solo il controllo proporzionale la correzione rimarrebbe costante e insufficiente per correggere l'errore, mentre la correzione del controllo integrale aumenta fino a diventare sufficiente ad annullare anche un piccolo errore.

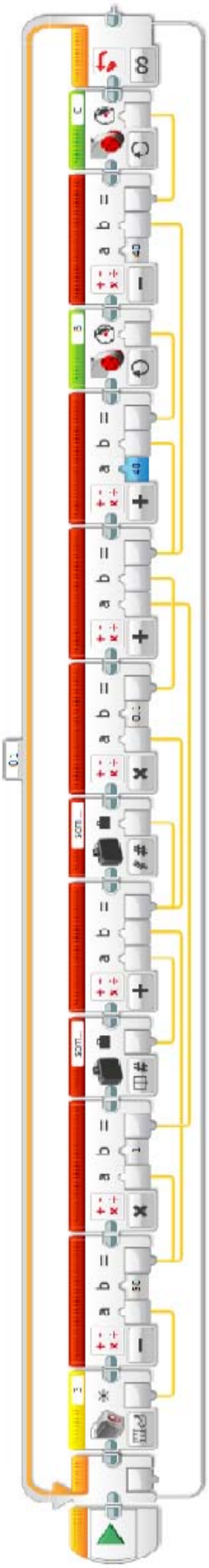
Vediamo in pratica come inserire la parte integrale del controllo in un programma partendo dalla programmazione del controllo proporzionale (in rosso le aggiunte necessarie):

Kp = 1	Impostiamo il valore di Kp che vogliamo utilizzare (es.1)
Ki = 0.1	Impostiamo il valore di Ki che vogliamo utilizzare (es.0.1)
riferimento= 50	Impostiamo valore da mantenere costante (media lettura bianco e nero, es.50)
Tp = 40	Impostiamo valore potenza ai motori delle ruote quando il robot non curva (es. 40)
somma_errori = 0	Definiamo una variabile somma_errori e poniamola uguale a 0

Loop	
LV = lettura sensore	Registriamo la lettura del sensore in una variabile LV (es 55)
errore = LV – riferimento	Calcoliamo l'errore di posizione sottraendo la lettura del sensore all'offset (55 - 50=5)
somma_errori = somma_errori + errore	Calcoliamo la somma degli errori (0+5=5) ← *Mi sto allontanando dalla linea*
T = Kp × errore + Ki × somma_errori	Calcoliamo la variazione di potenza da applicare ai motori (1 × 5 + 0,1 × 5 = 5.5)
PB = Tp + T	Potenza da applicare al motore B sinistra (40+5.5=55.5)
PC = Tp - T	Potenza da applicare al motore C destra (50 - 5.5 = 45.5)
end loop	

Giro a destra e mi riavvicino alla linea

In pratica, utilizzando Lego Mindstorms, il programma potrebbe essere impostato ad esempio nel seguente modo:



Anche il controllo integrativo ha i suoi limiti: oltre al calcolo corretto della Ki, che vedremo dopo, raggiunto errore=0 il controllo integrativo non determina una correzione pari a 0 e continua a dare un correzione fino a quando la somma degli errori non è zero (e quindi fino a quando non legge degli errori negativi che bilanciano gli errori positivi memorizzati). Questo può determinare un'oscillazione intorno alla zona di errore 0, nel nostro esempio la traiettoria ideale dell'inseguitore di linea.

Questo può essere evitato ad esempio:

- inserendo nel programma un reset a zero della somma_errori non appena errore = 0
  (nel loop, *prima* di calcolare il termine Ki × somma_errori, si controlla se errore è zero o molto piccolo, e in questo caso si impone somma_errori = 0),
- rendendo il controllo integrale meno sensibile alla somma degli errori precedenti inserendo nel programma un fattore correttivo (sempre minore di 1) del tipo:

$$\text{somma\_errori} = (2/3) \times \text{somma\_errori} + \text{errore}$$

CONTROLLO PROPORZIONALE INTEGRATIVO DERIVATIVO (PID)

In pratica, il controllo integrativo aiuta il controllo proporzionale a limitare gli errori quando si raggiunge una condizione di equilibrio e rimane un errore residuo costante e non eliminabile con il solo controllo proporzionale.

L'ultimo termine del controllo PID (D=Derivativo), aiuta invece il controllo proporzionale nella fasi di non equilibrio, quando il controllo proporzionale fornisce le correzioni, riducendo la correzione quando l'errore tende a diminuire (per evitare le oscillazioni) e aumentandola quando tende ad aumentare (per mantenere piccolo l'errore).

Il termine che viene aggiunto è:

$$T = K_p \times (\text{errore}) + K_i \times (\text{somma errori}) \mathbf{+ K_d \times (errore - errore\_precedente)}$$

Anche in questo caso è necessaria una costante Kd per ottenere dalla differenza degli errori il valore di correzione ai motori.

Se errore = errore_precedente, il valore di correzione del termine derivativo è 0

Se errore > errore_precedente, il termine derivativo aumenterà la correzione complessiva per aiutare il controllo a limitare l'errore

Se errore < minore di errore_precedente, il termine derivativo diminuirà la correzione complessiva per aiutare il controllo ad evitare fenomeni di oscillazione

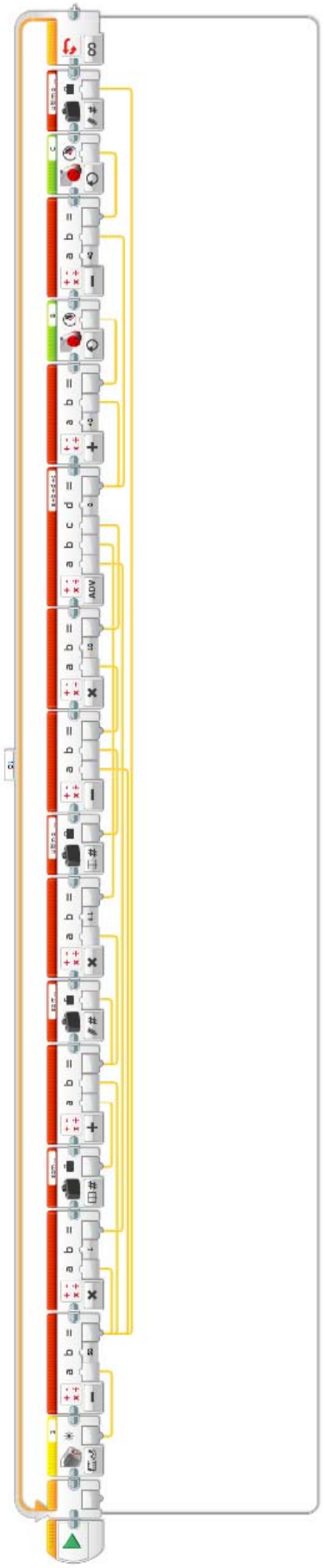
Vediamo coma aggiungere in pratica al controllo Proporzionale e Integrativo la parte Derivativa (in rosso le modifiche):

ALGORITMO COMPLETO CONTROLLORE PID

Kp = 1	Impostiamo il valore di Kp che vogliamo utilizzare (es.1)
Ki = 0.1	Impostiamo il valore di Ki che vogliamo utilizzare (es.0.1)
Kd = 5	Impostiamo il valore di Kd che vogliamo utilizzare (es.5)
riferimento = 50	Impostiamo valore da mantenere costante (media lettura bianco e nero, es.50)
Tp = 40	Impostiamo valore potenza ai motori delle ruote quando il robot non curva (es. 40)
somma_errori=0	Definiamo una variabile somma_errori e poniamola uguale a 0
ultimo_errore=0	Definiamo una variabile ultimo_errore e poniamola uguale a 0
derivata=0	Definiamo una variabile derivata e poniamola uguale a 0
Loop	
LV = lettura sensore	Registriamo la lettura del sensore in una variabile LV (es 55)
errore = LV – riferimento	Calcoliamo l'errore di posizione sottraendo la lettura sensore al riferimento (55 - 50=5)
somma_errori = somma_errori + errore	Calcoliamo la somma integrale degli errori (0+5=5)
derivata = errore – ultimo_errore	Calcoliamo la differenza tra l'errore e l'errore precedente (5 - 0 = 5) ← *Errore in aumento: Mi sto allontanando dalla linea*
T = Kp × errore + Ki × somma_errori + Kd × derivata	Calcoliamo la variazione di potenza da applicare ai motori (1x5+ 0,1x5 + 5 x 5 = 30.5)
PB = Tp + T	Potenza da applicare al motore B a sinistra (40 + 30.5 = 70.5)
PC = Tp – T	Potenza da applicare al motore C a destra (40 - 30.5 = 9.5)
Ultimo_errore = errore	Salvo valore attuale dell'errore per prox iterazione
end loop	

Giro a destra e mi riavvicino alla linea

In pratica, utilizzando Lego Mindstorms, il programma potrebbe essere impostato ad esempio nel modo illustrato nella pagina seguente.

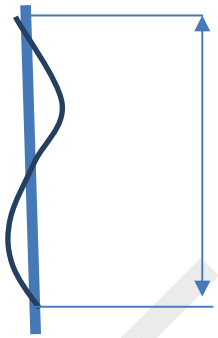


Come calcolare i Guadagni Kp, Ki, Kd

In pratica esistono vari metodi per calcolare il valore ottimale di Kp, Ki, Kd. Uno dei metodi più utilizzati per avere una buona stima dei valori di Kp, Ki, Kd è il metodo *Ziegler–Nichols*.

Applicato al nostro esempio di inseguitore di linea:

1. Fissate Ki e Kd a 0, in modo tale che il vostro controllo funzioni come un semplice controllo proporzionale P
2. Fissate la potenza Tp a un valore molto basso (ad esempio una potenza di 25)
3. Calcolate il valore di Kp critico (indicato con Kc), aumentando Kp se il robot perde la linea o diminuendo Kp se il robot oscilla intorno alla linea, fino a quando il robot segue la linea con delle piccole oscillazioni e misurate con un cronometro il periodo di oscillazione T (il valore di T dovrebbe essere come ordine di grandezza intorno a 0.5-2/3 secondi)



T=Periodo di oscillazione

Calcolate il tempo necessario ad eseguire un loop del vostro programma imponendo un numero di loop fissato (es 10.000) e misurando il tempo necessario ad eseguirli, *t*. Il tempo necessario ad eseguire un singolo loop è pari a:

$$tloop = t / 10.000$$

L'ordine di grandezza di tloop dovrebbe essere intorno a [0.015 - 0.020] secondi

4. Calcolate i valori di Kp, Ki e Kd utilizzando la seguente tabella:

Ziegler–Nichols			
Controllo	**Kp**	**Ki**	**Kd**
P	$0.50 \times K_c$	0	0
PI	$0.45 \times K_c$	$(1.2 \times K_p \times tloop)/ T$	0
PD	$0.80 \times K_c$	0	$(K_p \times T)/(8 \times tloop)$
PID	$0.60 \times K_c$	$(2 \times K_p \times tloop) / T$	$(K_p \times T) / (8 \times tloop)$

La tabella permette di calcolare i valori delle costanti in funzione del fatto che si voglia un controllo solo proporzionale, proporzionale-integrale, proporzionale-integrale-derivativo o proporzionale-derivativo.

Inserite le costanti, è possibile siano necessari degli aggiustamenti per ottimizzare il comportamento del robot.

Per provare a fare delle variazioni delle costanti K può essere utile la tabella riportata qui sotto, che indica quali costanti aumentare o diminuire per avere una diminuzione dei seguenti effetti:

Rise time: indica il tempo che impiega il robot per portarsi nella posizione corretta dopo una perturbazione esterna o dopo la richiesta di una nuova posizione corretta (cambio della linea nel nostro esempio, una nuova direzione,…). Maggiore è il valore di Kp e più veloce è la risposta ma se il valore di Kp è troppo alto il robot può continuare a correggere oltrepassando la posizione corretta (Overshoot)

Overshoot: indica quando il robot raggiunta la posizione corretta continui a mantenere la correzione oltrepassando la posizione corretta

Settling time: indica il tempo in cui il robot oscilla intorno alla posizione corretta prima di stabilizzarsi

Error at equilibrium: indica l'errore residuo che si mantiene quando il robot è stabilmente nella posizione corretta

Come ridurre				
K	**Rise time**	**Overshoot**	**Settling time**	**Error at equilibrium**
K_p	aumentare	diminuire	provare piccoli aumenti/diminuzioni	aumentare
K_i	aumentare	diminuire	diminuire	aumentare
K_d	provare piccoli aumenti/diminuzioni	aumentare	aumentare	nessun effetto