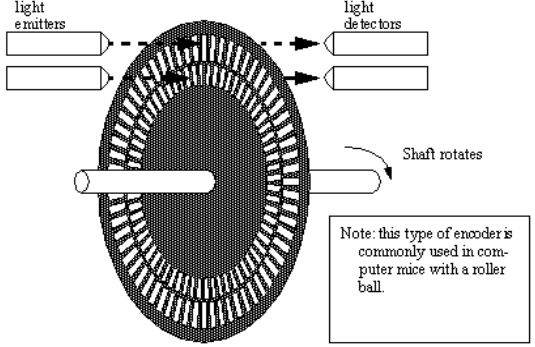
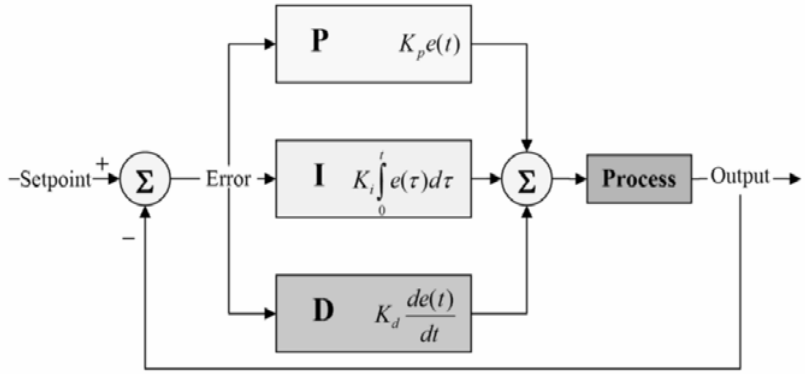


Il PID in breve

Il controllo di velocità di motori DC per la robotica mobile



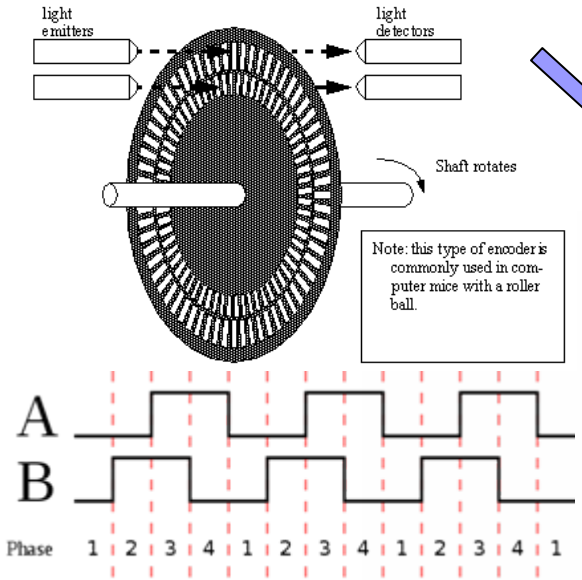
Giovanni De Luca

Laboratorio progettazione elettronica

Sistema di controllo standard

Un sistema di controllo fornisce un comando in tensione o in corrente agli attuatori (motori) in modo da far assumere ai giunti una configurazione desiderata

Elementi di un Sistema di controllo



uControllore

Posizione

Unità di controllo

Segnale di riferimento

Encoder

Motore

Riduttore

mot +

mot -

amplificatore di potenza

Alimentazione

GIUNTO

Ruota

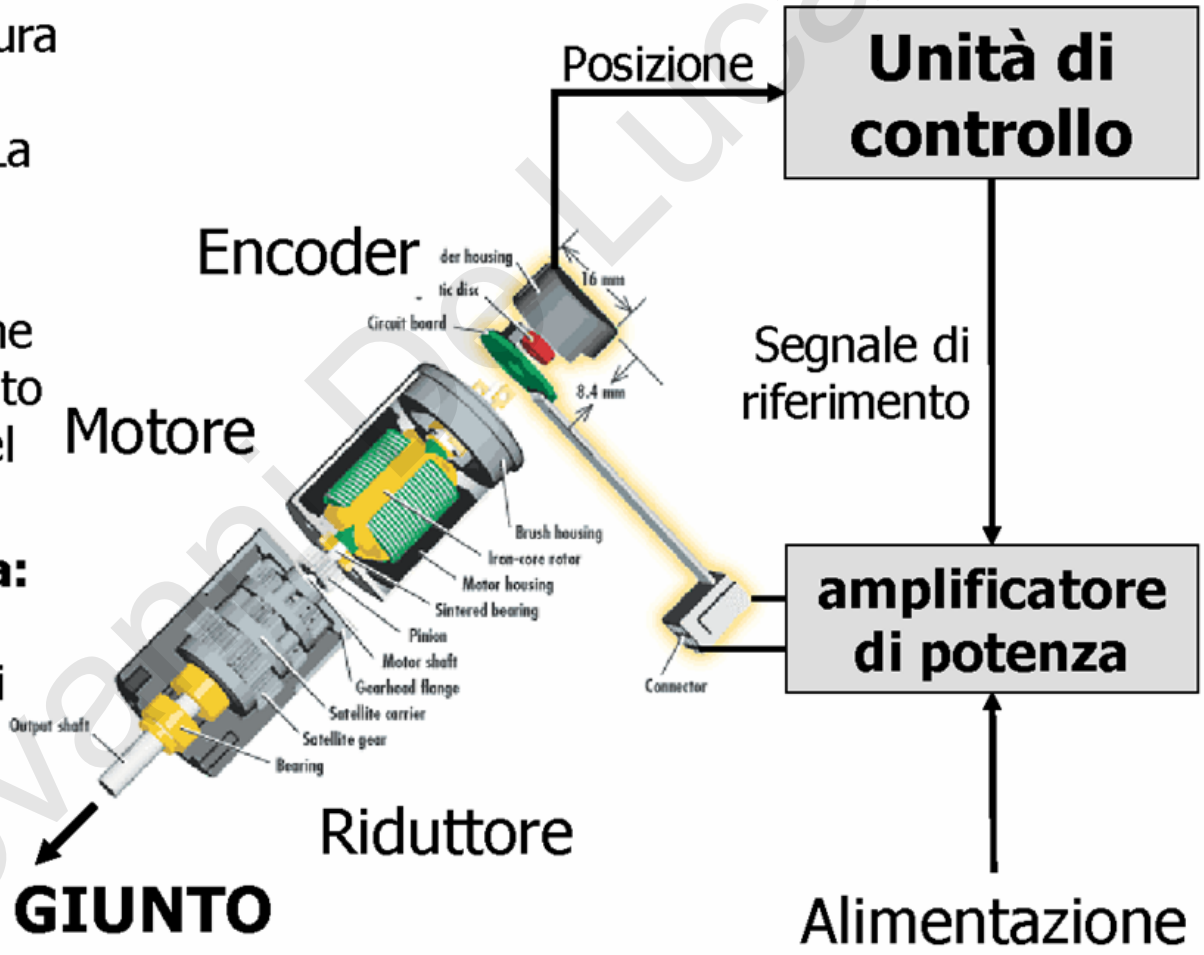
Elementi unità di controllo

Encoder: sensore che misura la rotazione dei giunti in valore relativo o assoluto. La misurazione avviene in "*tacche di encoder*"

Riduttore: meccanismo che riduce i giri dell'asse montato sul giunto rispetto ai giri del motore (es. riduzione 1:N)

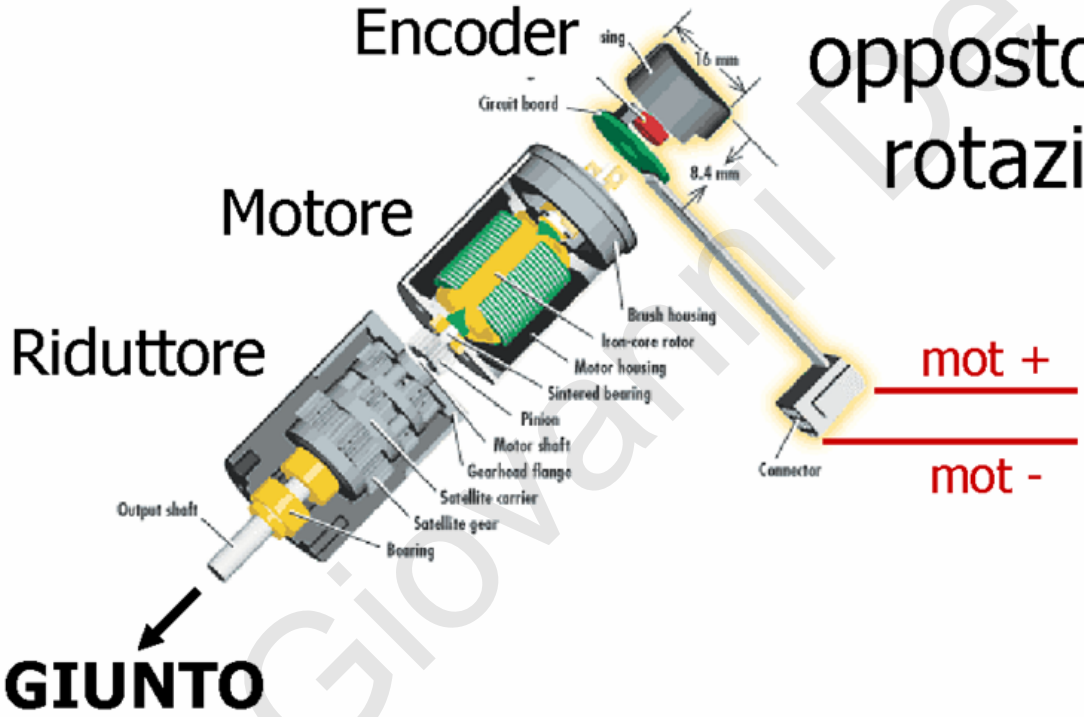
Amplificatore di potenza: amplifica un segnale di riferimento in un segnale di potenza per muovere il motore

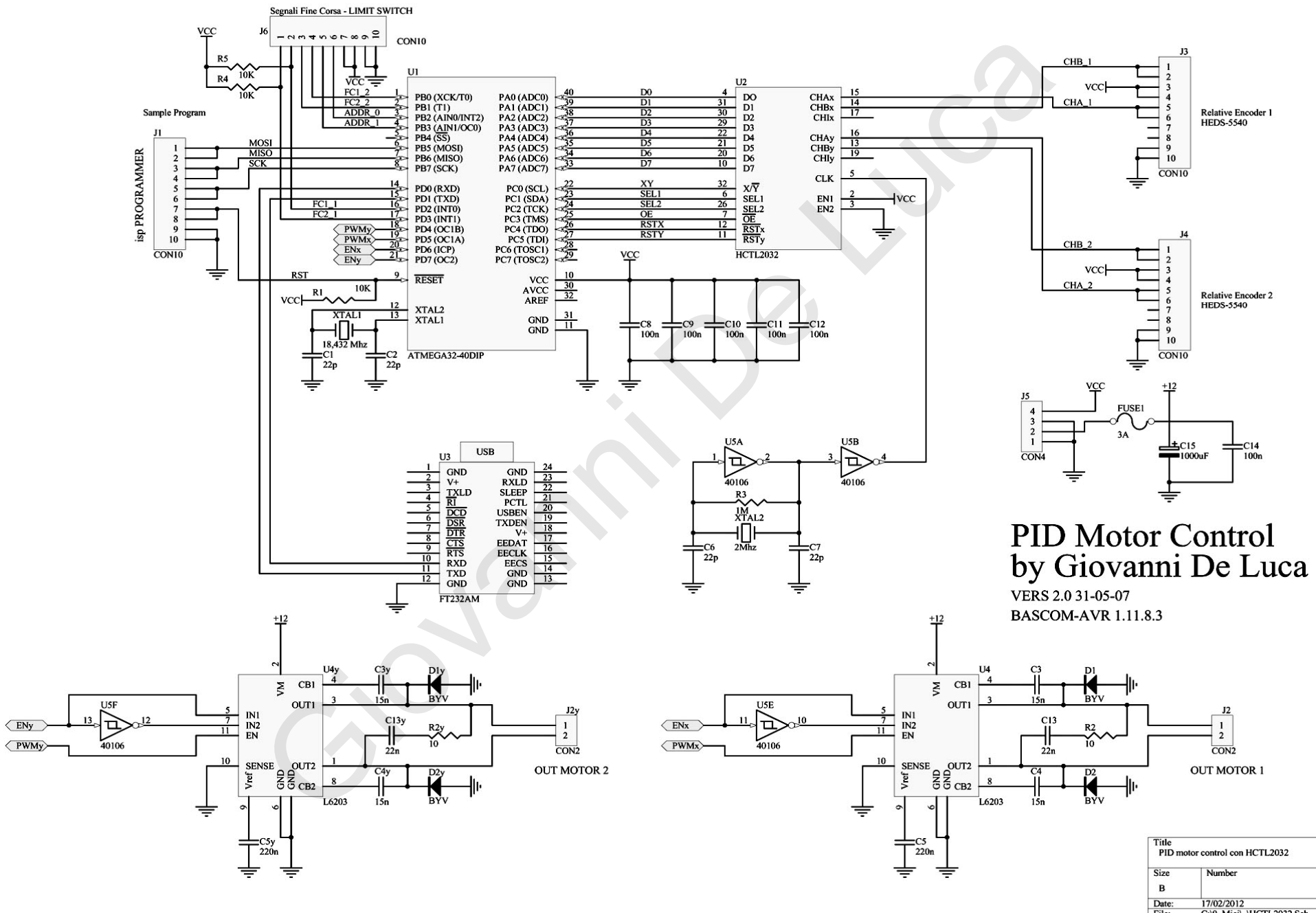
Unità di controllo: unità che produce un segnale di riferimento per il motore



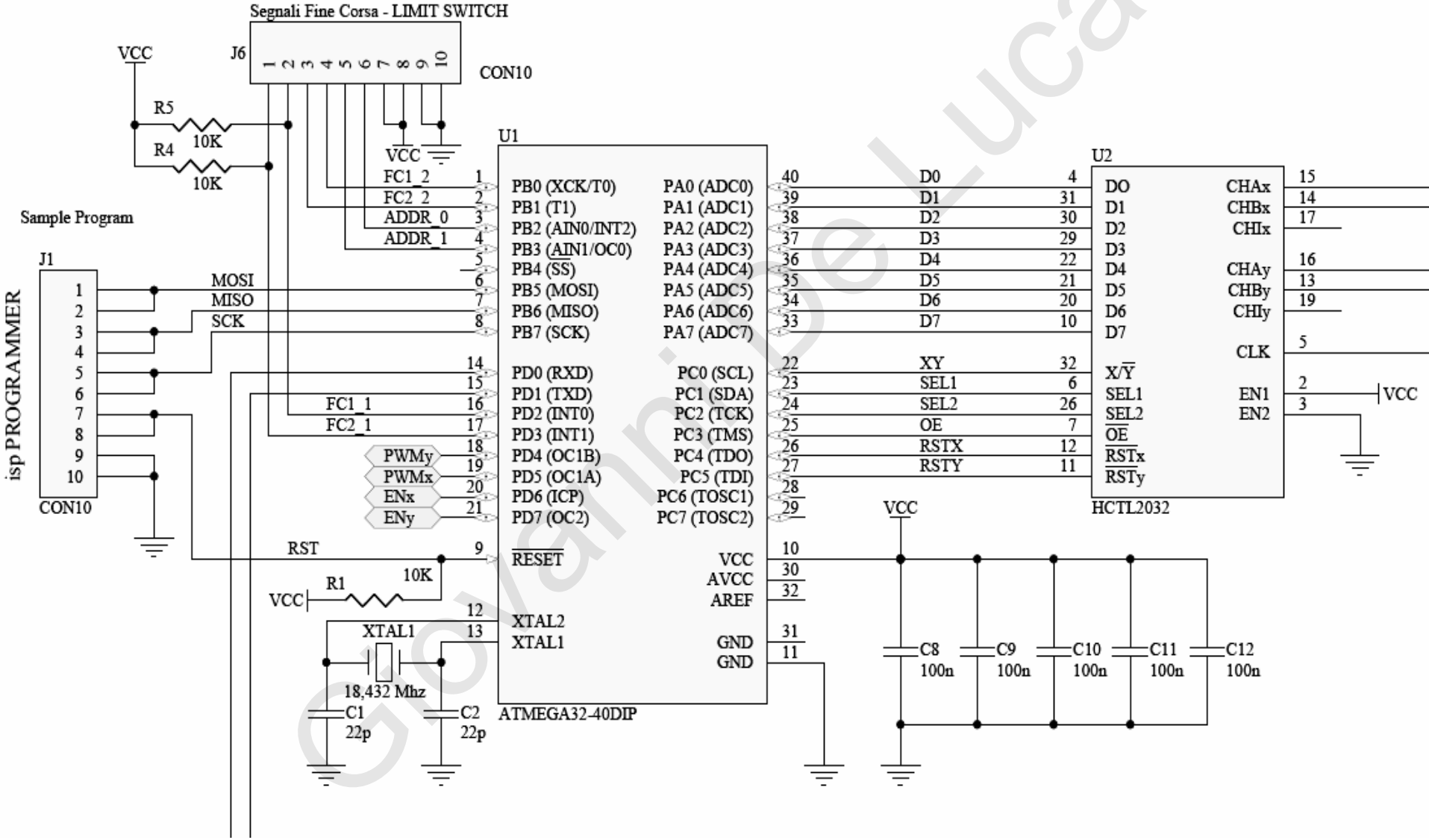
Elementi unità di controllo

A tensioni in ingresso al motore di segno opposto corrispondono rotazioni opposte del motore

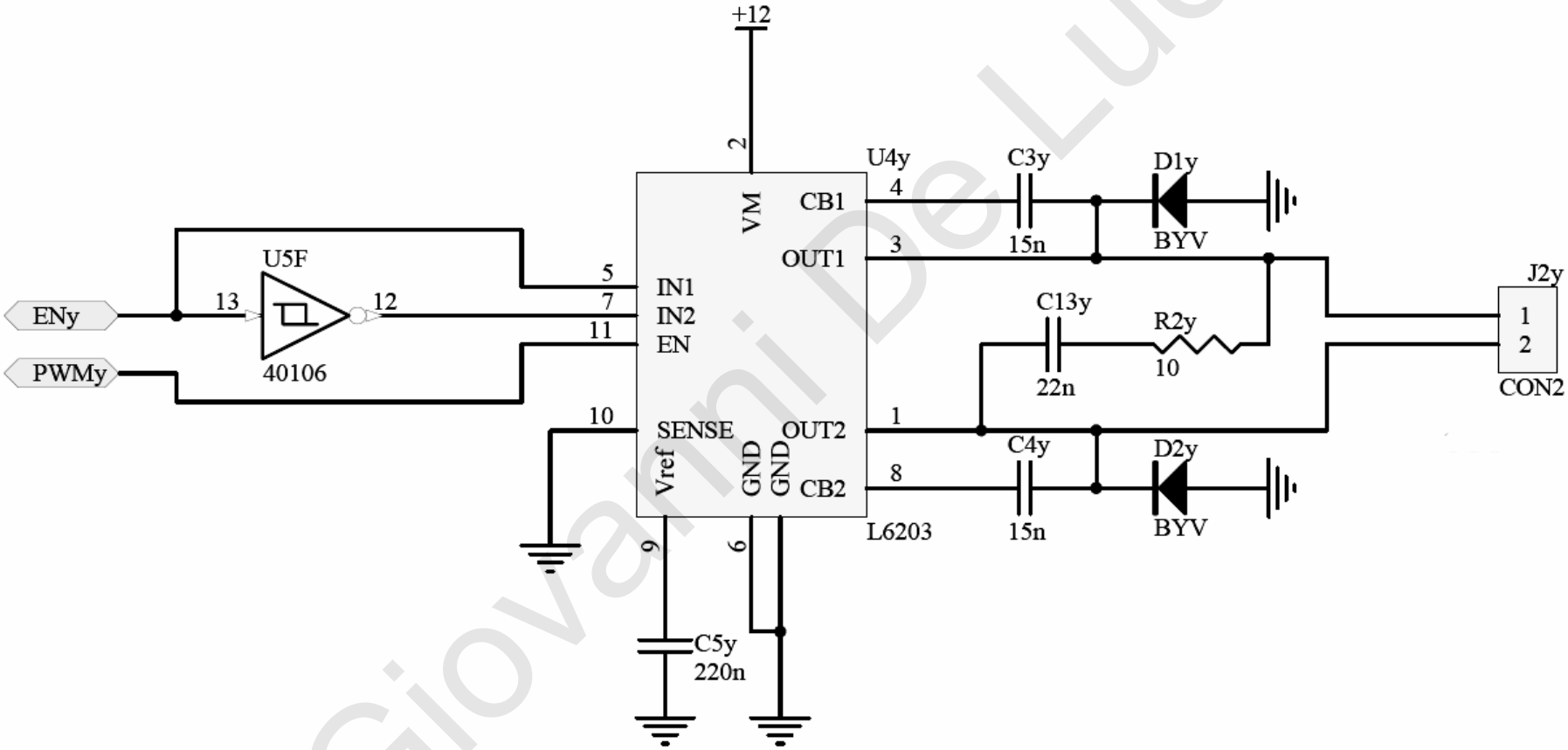




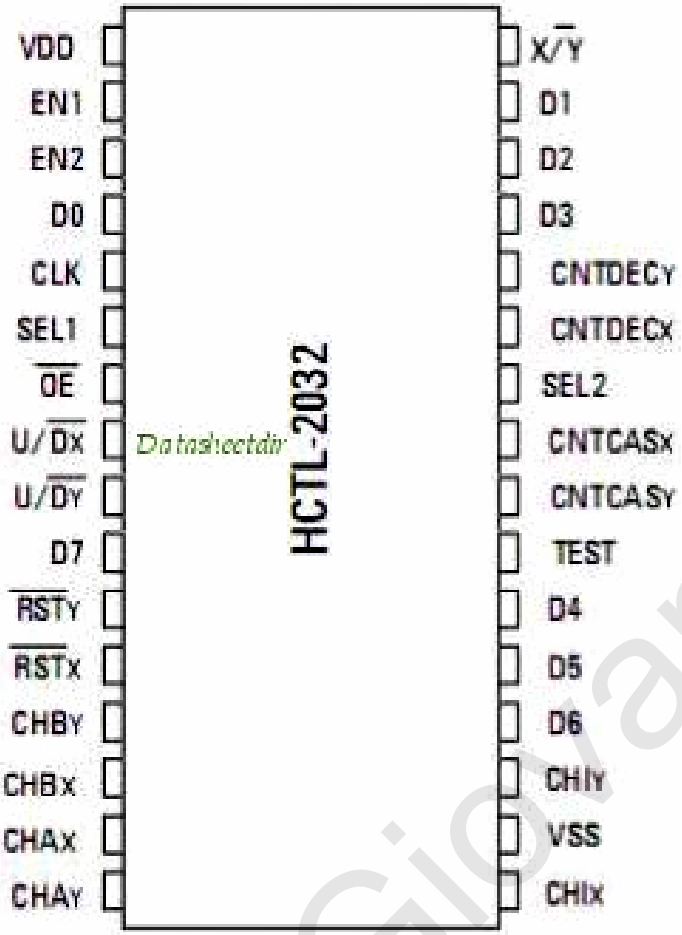
Collegamenti con il uC



Power con L6203



HCTL 2032



The HCTL-2032 consists of a quadrature decoder logic, a binary up/down state counter, and an 8-bit bus interface.
The HCTL-2032 contains 32-bit counter and provides LSTLL compatible tri-state output buffers.
The HCTL-2032 have dual-axis capability and index channel support.
Both devices can be programmed as 4x/2x/1x count mode.
The HCTL-2032 also provides quadrature decoder output signals and cascade signals for use with many standard computer ICs.

Pins description

Table 4. Functional Pin Descriptions

Symbol	Pin: HCTL 2032/ 2032-SC	Pin: HCTL 2022	Description
V_{DD}	1	1	Power Supply
V_{SS}	18	12	Ground
CLK	5	3	CLK is a Schmitt-trigger input for the external clock signal.
CHA_X	15	10	CHA_X, CHA_Y, CHB_X, and CHB_Y are Schmitt-trigger inputs that accept the outputs from a quadrature-encoded source, such as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required. CHA_X and CHB_X are the 1st axis and CHA_Y and CHB_Y are the 2nd axis.
CHA_Y	16	NC	
CHB_X	14	9	
CHB_Y	13	NC	
CHI_X	17	11	CHI_X and CHI_Y are Schmitt-trigger inputs that accept the outputs of Index channel from an incremental optical shaft encoder.
CHI_Y	19	NC	
$RSTN_X$	12	8	This active low Schmitt-trigger input clears the internal position counter and the position latch. It also resets the inhibit logic. RST_X/ and RST_Y/ are asynchronous with respect to any other input signals. RST_X/ is to reset the 1st axis counter and RST_Y/ is to reset the 2nd axis counter.
$RSTN_Y$	11	NC	
OEN	7	5	This CMOS active low input enables the tri-state output buffers. The OE/, SEL1, and SEL2 inputs are sampled by the internal inhibit logic on the falling edge of the clock to control the loading of the internal position data latch.
SEL_1	6	4	These CMOS inputs directly controls which data byte from the position latch is enabled into the 8-bit tri-state output buffer. As in OE/ above, SEL_1 and SEL_2 also control the internal inhibit logic.
SEL_2	26	17	

Pins description

EN_1	2	NC	These CMOS control pins are set to high or low to activate the selected count mode before the decoding begins.
EN_2	3	NC	
X/Y	32	NC	Select the 1st or 2nd axis data to be read. Low bit enables the 1st axis data, while high bit enables the 2nd axis data.
$CNTDEC_X$	27	NC	A pulse is presented on this LSTTL-compatible output when the quadrature decoder (4x/2x/1x) has detected a state transition. CNTDECX is for 1st axis and CNTDECY is for 2nd axis.
$CNTDEC_Y$	28	NC	
U/Dx	8	6	This LSTTL-compatible output allows the user to determine whether the IC is counting up or down and is intended to be used with the CNTDEC and CNTCAS outputs. The proper signal U (high level) or D/ (low level) will be present before the rising edge of the CNTDEC and CNTCAS outputs.
U/Dy	9	NC	

		BYTE SELECTED			
SEL1	SEL2	MSB	2ND	3RD	LSB
0	1	D4			
1	1		D3		
0	0			D2	
1	0				D1

		Count Modes		
EN1	EN2	4x	2x	1x
0	0	Illegal Mode		
1	0	On		
0	1		On	
1	1			On

Altri pins

$CNTCAS_X$	25	NC	A pulse is presented on this LSTTL-compatible output when the HCTL-2032 / 2032-SC internal counter overflows or underflows. The rising edge on this waveform may be used to trigger an external counter.
$CNTCAS_Y$	24	NC	
TEST	23	16	This pin is used for internal testing. Tied it to ground or leave it floating for normal operation.
D0	4	2	These LSTTL-compatible tri-state outputs form an 8-bit output ports through which the contents of the 32-bit position latch may be read in 4 sequential bytes. The MSB is read first followed by the rest of the bytes with the LSB is read last.
D1	31	20	
D2	30	19	
D3	29	18	
D4	22	15	
D5	21	14	
D6	20	13	
D7	10	7	

Schema a blocchi

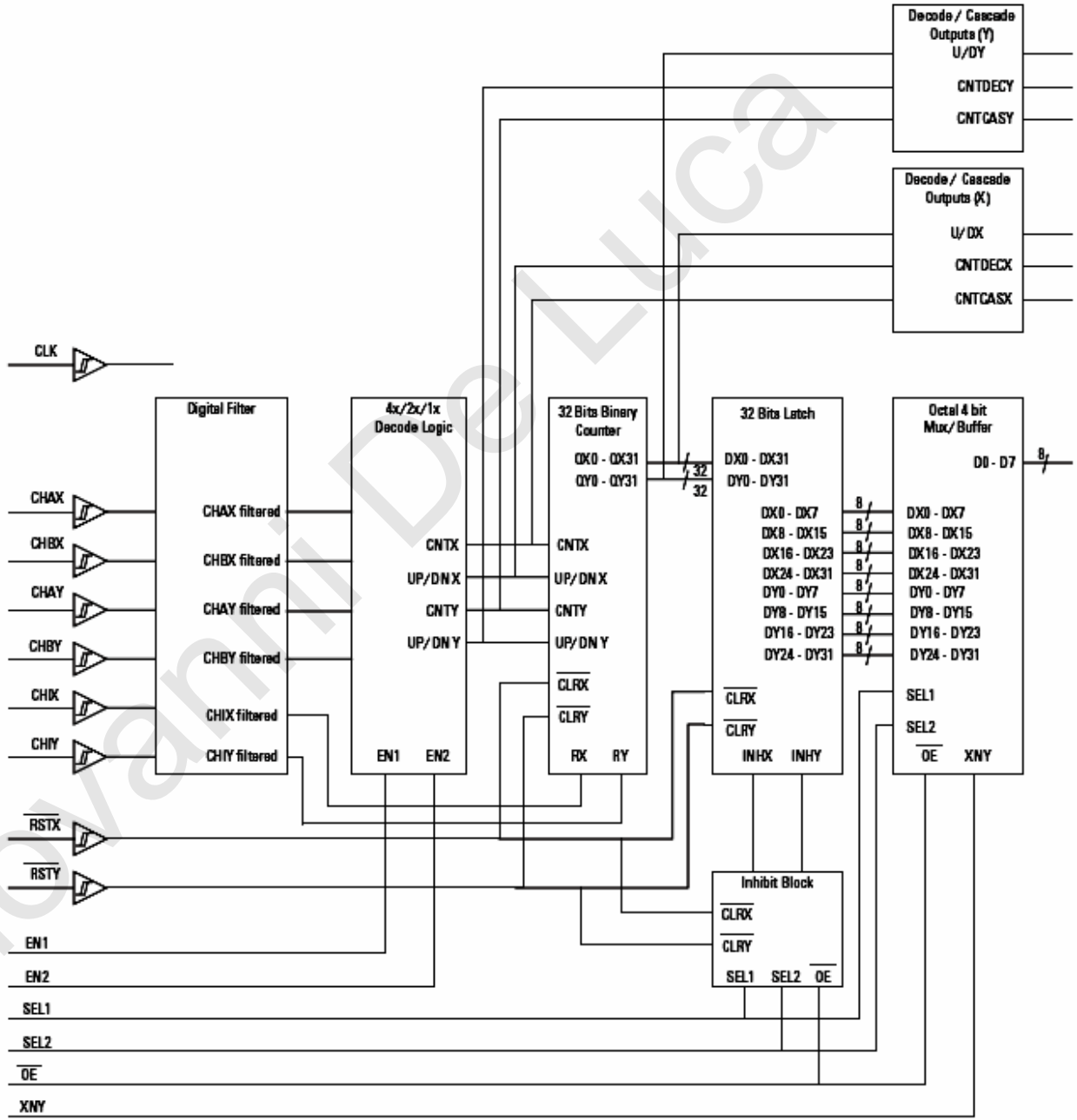


Figure 10. Simplified Logic Diagram

Bus control timing

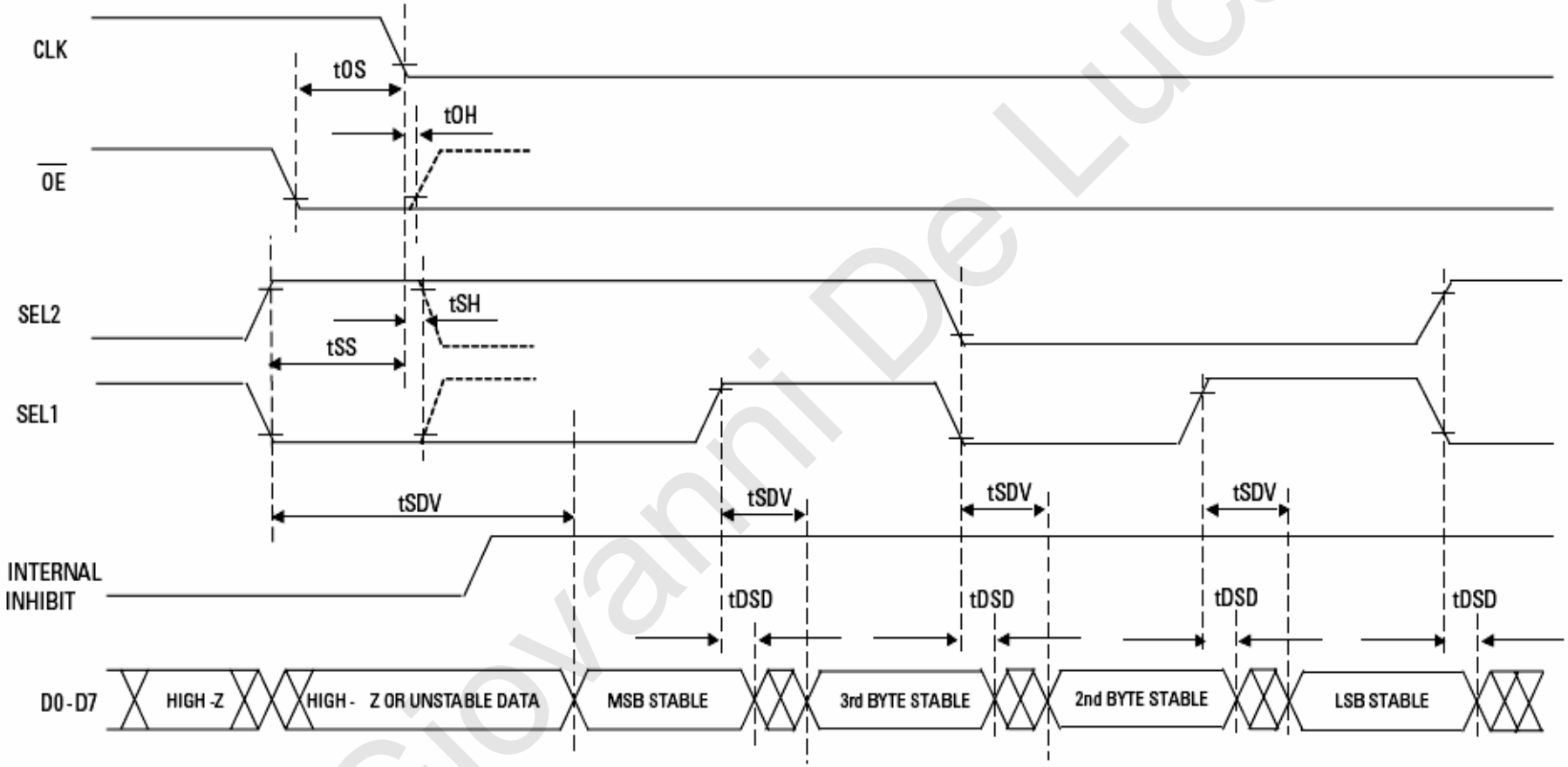
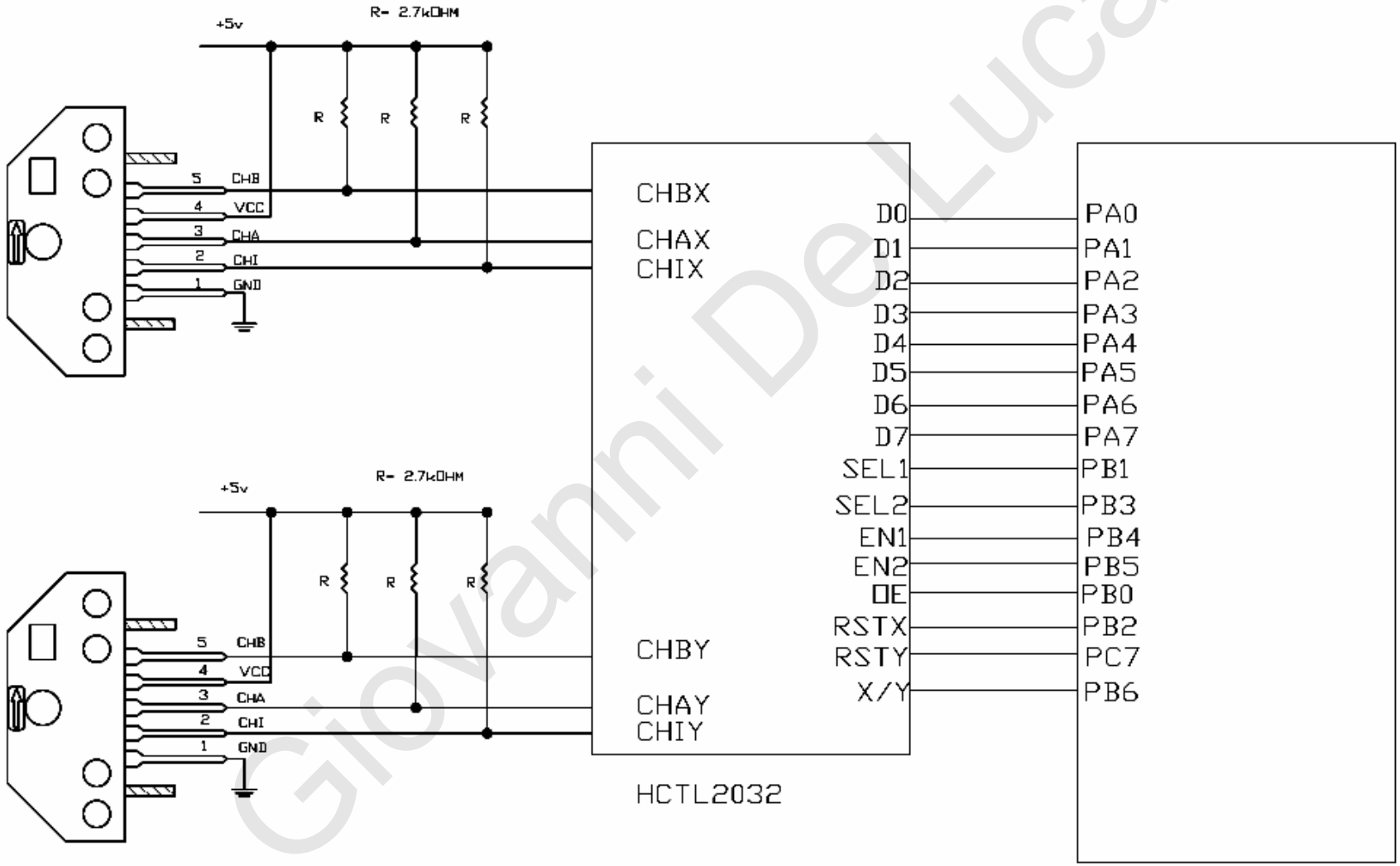


Figure 4. Bus Control Timing

Atmega Interface



Il PID in breve

Semplice descrizione dell'algoritmo PID ampiamente applicato nei controlli, in particolare riferimento:

- Regolazione di portata
- Regolazione di velocità nei motori elettrici
- Regolazione di temperatura
- Tanto altro ….

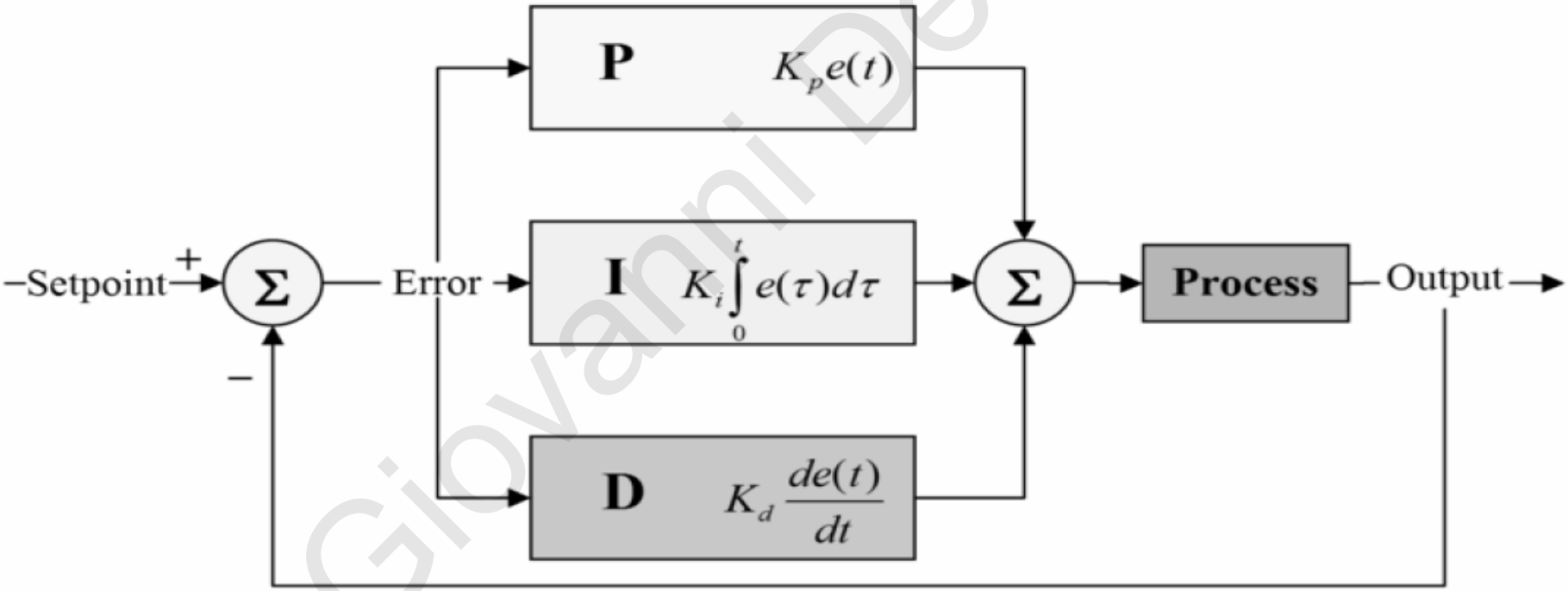
A cosa serve il PID

- Ogni volta che un dispositivo deve mantenere costante un determinato valore, ad esempio una velocità, una temperatura, un livello, una rotta ecc. serve un regolatore. Ci serve qualcosa che corregga eventuali ed inevitabili errori rispetto al valore di consegna.

- Se pensiamo al pilota automatico di una nave che deve lavorare per mantenere la rotta, ci è facile comprendere come venti, correnti e onde siano fonti di errore che il regolatore deve contrastare. Non è detto che sia solo così, ci può essere anche il caso in cui si debba seguire un valore che cambia come ad esempio seguire un percorso che cambia come succede ad un robot tipo line-follower o se vogliamo ad un missile che segue l'aereo da colpire o il valore della temperatura in un forno per la diffusione nei wafer per la costruzione dei chip.

Il controllo PID

Il regolatore PID (Proporzionale - Integrale - Derivativo) fornisce le tre azioni di controllo in grado di inseguire un setpoint e contrastare una grande varietà di disturbi



Controllo Proporzionale

- Nel caso della regolazione della temperatura tale sistema elimina il problema delle fluttuazioni di temperatura utilizzando una regolazione continua della potenza erogata dal riscaldatore: questa è proporzionale alla entità della differenza tra la temperatura reale ed il set-point. Così ad esempio una grosso errore negativo produrrà una grande tensione al riscaldatore per correggere l'errore.
- Se la potenza in uscita fosse proporzionale all'errore nell'intero range dello strumento, sarebbe necessario un errore negativo pari alla metà del range per ottenere il massimo della potenza del riscaldatore. L'accuratezza sarebbe quindi molto insoddisfacente.
- Si rimedia a ciò introducendo il parametro di **banda proporzionale** (negli USA è più comune il suo reciproco, il guadagno GAIN=1/banda proporzionale). La banda proporzionale è espressa normalmente in frazione percentuale dell'intervallo di funzionamento dello strumento: all'interno della banda proporzionale la potenza in uscita sarà proporzionale all'errore, all'esterno di questa banda la potenza sarà la massima oppure zero.
- Riducendo la banda proporzionale (i.e. aumentando il guadagno) l'accuratezza del controllore migliora, visto che basta un errore più piccolo per avere una data modifica della potenza in uscita.

La P di PID

- Per il pilota automatico, detto in modo molto banale può sembrare sufficiente qualcosa che legge la bussola e giri il timone dal lato giusto e dell'angolo giusto per correggere; con un comportamento quindi Proporzionale (più la nave è sbandata più intensa sarà la correzione). Un comportamento quindi legato ad una relazione tipo: **correzione=K*errore** dove K è ovviamente da impostare in funzione dei parametri meccanici del timone e della velocità di risposta che vogliamo.
- Beh, in effetti questo va già bene, è un regolatore di tipo proporzionale e come credo sia facile intuire la P di PID sta proprio per proporzionale. Sfortunatamente un regolatore che preveda una correzione solo proporzionale non è sempre ciò che ci serve. Ritornando al nostro esempio sarebbe facile verificare quanto i nostri passeggeri siano in preda ad attacchi di mal di mare e ciò a causa del fatto che la nave continua a virare attorno alla rotta assegnata. Senza inoltrarci nella teoria del perché proviamo con l'intuito. Intanto è evidente che il regolatore è attivo solo quando c'è un errore e quindi salvo casi eccezionali o brevi transitori avremo sempre un errore inoltre, se si eccede nella intensità della correzione il sistema diventa instabile e inizia a pendolare attorno al valore giusto con oscillazioni sempre più ampie. Giusto per non fare solo chiacchiere ritorniamo alla formula citata prima che mettiamo in forma più seria:

$$P = K_p * \varepsilon$$

La P di PID

- **P** è il valore della correzione che vogliamo dare al nostro sistema.
- **ε** è il valore dell'errore tra il valore della grandezza che stiamo controllando e il valore che desideriamo o valore di consegna
- **Kp** è la costante che imposteremo per regolare la risposte del sistema.
- Chi ha fatto un minimo di matematica riconoscerà nella formula l'equazione tipica di una retta dove **Kp** rappresenta la pendenza della retta. In pratica maggiore sarà **Kp** maggiore sarà "l'intensità" della correzione e più veloce il ritorno verso il valore di consegna e maggiore sarà però il rischio di instabilità. In sostanza la parte proporzionale è fondamentale per un regolatore in quanto lega la correzione in modo diretto all'errore, da sola però questa parte non riesce a dare stabilità al sistema e se vogliamo nemmeno a prevenire l'errore.

Controllo Integrale

- Per migliorare l'accuratezza del controllo proporzionale si introduce il controllo integrale.
- Consideriamo un sistema controllato col sistema proporzionale, con la banda proporzionale grande a sufficienza per non indurre auto-oscillazioni. Il risultato è un sistema stabile ma non eccessivamente accurato. Supponiamo di mandare il segnale di errore residuo ad un integratore, la cui uscita è sommata a quella del proporzionale. Il risultato sarà che la potenza in uscita aumenta fintanto che la temperatura non eguaglia il set-point. A questo punto l'uscita dell'integratore si annulla e così si mantiene una potenza costante. L'integratore potrebbe però indurre oscillazioni. Ciò è evitato dalla presenza del proporzionale.
- Il controllo integrale è caratterizzato dal **tempo di integrazione** (integral action time, negli USA più comunemente il RESET), definito come il tempo necessario perché l'uscita vari da zero al suo massimo in presenza di un errore fisso pari alla banda proporzionale.
- Il RESET può essere specificato come un tempo o come una frequenza (ripetizioni per minuto).
- Per evitare che il controllo integrale induca oscillazioni nel sistema è bene porre il tempo di integrazione pari almeno alla costante di tempo di risposta del sistema.
- Se il set-point viene variato considerevolmente è probabile che nel tempo che il sistema impiega ad avvicinarsi al nuovo set-point l'integratore venga saturato, risultando poi in un overshoot quando la temperatura raggiunge finalmente il set-point. E' perciò conveniente mantenere a zero l'integratore fintanto che la temperatura non rientra all'interno della banda proporzionale.

La I di PID

- L'integrale altro non è se non la somma di tutti gli errori nel tempo e in pratica si ottiene semplicemente facendo:

$$I = I + Ki * \varepsilon$$

- Questo parametro "ricordando" gli errori precedenti produce l'effetto di smorzare i penzolamenti tipici del sistema proporzionale. Se prima il sistema ci forniva in uscita una variabile che era: **OUT=Kp*ε** e che avevamo chiamato P ora abbiamo:

$$OUT = P + I$$

- Con I calcolata come nella formula precedente.
- L'effetto di rendere più stabile il sistema è dato da una sorta di ritardo con la quale la variabile out viene integrata. Questo ritardo dipende dalla **Ki**, costante che dovremo scegliere e bilanciare affinché l'accoppiata **P** e **I** sia efficace.
- Questo ritardo ha però un prezzo, rende l'insieme meno reattivo e veloce. Per molte applicazioni comunque il tutto potrebbe essere già sufficiente così.

Controllo Derivativo

- La combinazione di controllo proporzionale e integrale garantisce un controllo stabile ed accurato, tuttavia se il **set-point** viene variato è probabile che il sistema approssimi il nuovo **set-point** con poca prontezza od alternativamente con buona prontezza ma producendo un overshoot. Ciò è curato dal **controllo derivativo**. Come suggerisce il nome, il controllo derivativo misura la derivata temporale del segnale di errore del sistema, e modifica la potenza in uscita in modo da ridurre la velocità della variazione.
- Anche il controllo derivativo è caratterizzato da un tempo caratteristico, il **tempo di derivata**, (derivative action time, negli USA il **RATE**, che può essere dato indifferentemente come un tempo o come una frequenza). Se il segnale d'errore sta variando velocemente, al tasso di una banda proporzionale in un tempo di derivata, allora l'uscita del derivatore è sufficiente a portare a zero la potenza di uscita.
- In molti casi il controllo **PI** è sufficiente, in altri si rende necessario anche il termine derivativo.
- La determinazione dei tre parametri di guadagno del **PID** può essere fatta empiricamente nell'ordine logico in cui sono stati descritti i controlli, oppure negli strumenti moderni spesso c'è la possibilità di lasciare che sia il controllore stesso a determinare i tempi di risposta del sistema, applicando una serie di impulsi deltiformi di potenza e misurando il tempo di risposta proprio del sistema.
- Molto spesso infine i parametri **PID** ottimali dipendono anche della temperatura di lavoro di un dato sistema. Il processo di ottimizzazione dei parametri del **PID** quindi viene ripetuto, con piccoli aggiustamenti continui.

La D di PID

- La derivata è una funzione matematica che credo conosciate, diversamente vi rimando sui libri di scuola anche se per come la analizzeremo qui non è strettamente necessario conoscerla in modo approfondito.
- Viene usata la derivata per ottenere una risposta più rapida e se vogliamo una previsione di quanto sta accadendo. La derivata infatti lavora sulla tendenza dell'errore e non sull'errore assoluto. Questo significa che ad esempio sapendo che l'errore sta crescendo possiamo dare più energia alla correzione di quanto previsto dal solo parametro proporzionale mentre possiamo ridurre la correzione se l'errore sta diminuendo segno che se siamo in fase di arrivo. La derivata per così dire accelera o decelera l'intervento del regolatore in modo dinamico seguendo la tendenza dell'errore e permettendoci di prevedere che, senza altri interventi, alla prossima lettura l'errore sarà minore o maggiore in base appunto alla tendenza.
- A livello di formule la derivata la calcoliamo così:

$$D=Kd*\delta\varepsilon/\ \delta t$$

La D di PID

- Qui abbiamo che **D** è il valore della correzione, **Kd** è la solita costante che andremo a settare noi in funzione del peso che vogliamo dare alla parte derivativa del nostro regolatore.
- **δε** è la differenza tra i due errori ossia, è chiaro che il nostro sistema dovrà controllare la nostra grandezza ad intervalli regolari e confrontarla con il valore che dobbiamo mantenere per cui ad ogni controllo avremo un errore, se fissiamo in t il momento dell'ultimo controllo avremo che il controllo precedente sarà stato effettuato in t-1, avremo quindi due letture di errore, **εt εt-1** la differenza tra questi due valori ci dice se
- l'errore sta aumentando o diminuendo, avremo quindi un risultato positivo (l'errore aumenta) se l'ultimo errore è maggiore del precedente, negativo al contrario, noterete quindi che **D** assume un valore positivo o negativo in funzione della tendenza e non del valore assoluto dell'errore.
- **δt** se ripetiamo la misura ad intervalli regolari viene fissato ad 1 non è quindi influente nel nostro calcolo.

Finalmente il PID

- Abbiamo finalmente i nostri tre elementi che costituiscono il nostro PID, non ci resta che combinarli.
- Beh, dal punto di vista matematico è molto semplice:

$$OUT = P + I + D$$

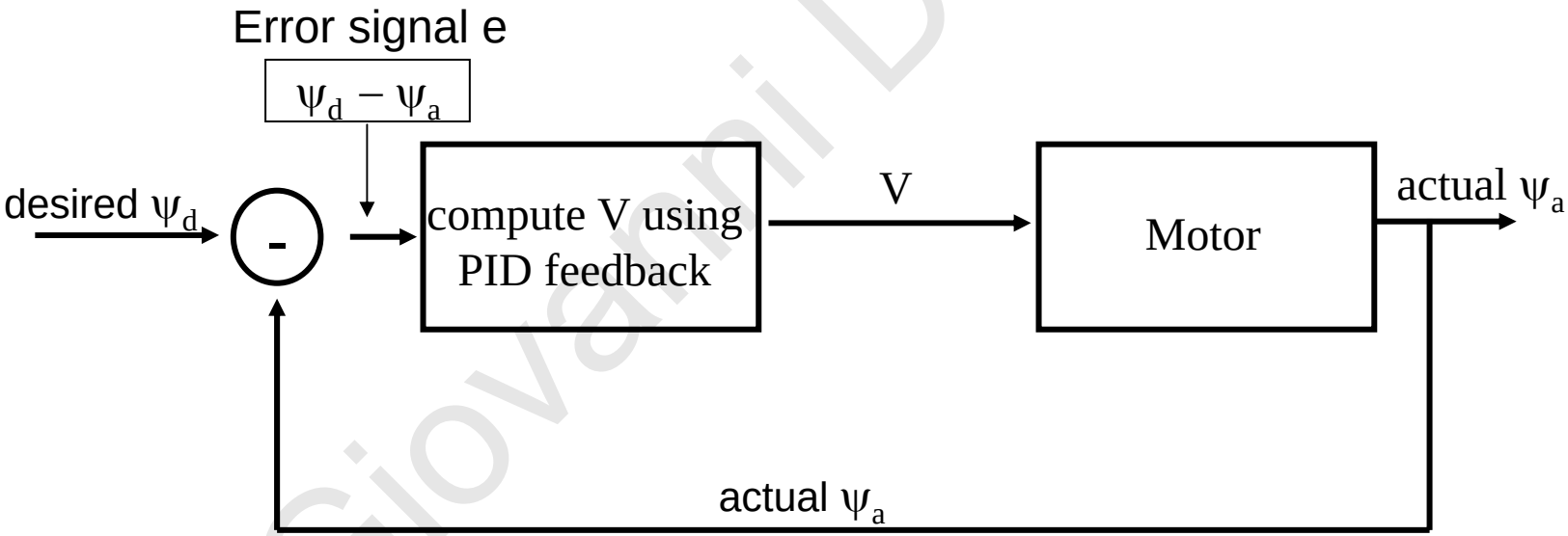
- In effetti è tutto quello che serve, i calcoli dei tre parametri li abbiamo fatti prima, li riassumiamo aggiungendo che ovviamente vanno fatti ad ogni ciclo di misura della nostra grandezza da regolare.

$$P = K_p * \varepsilon$$

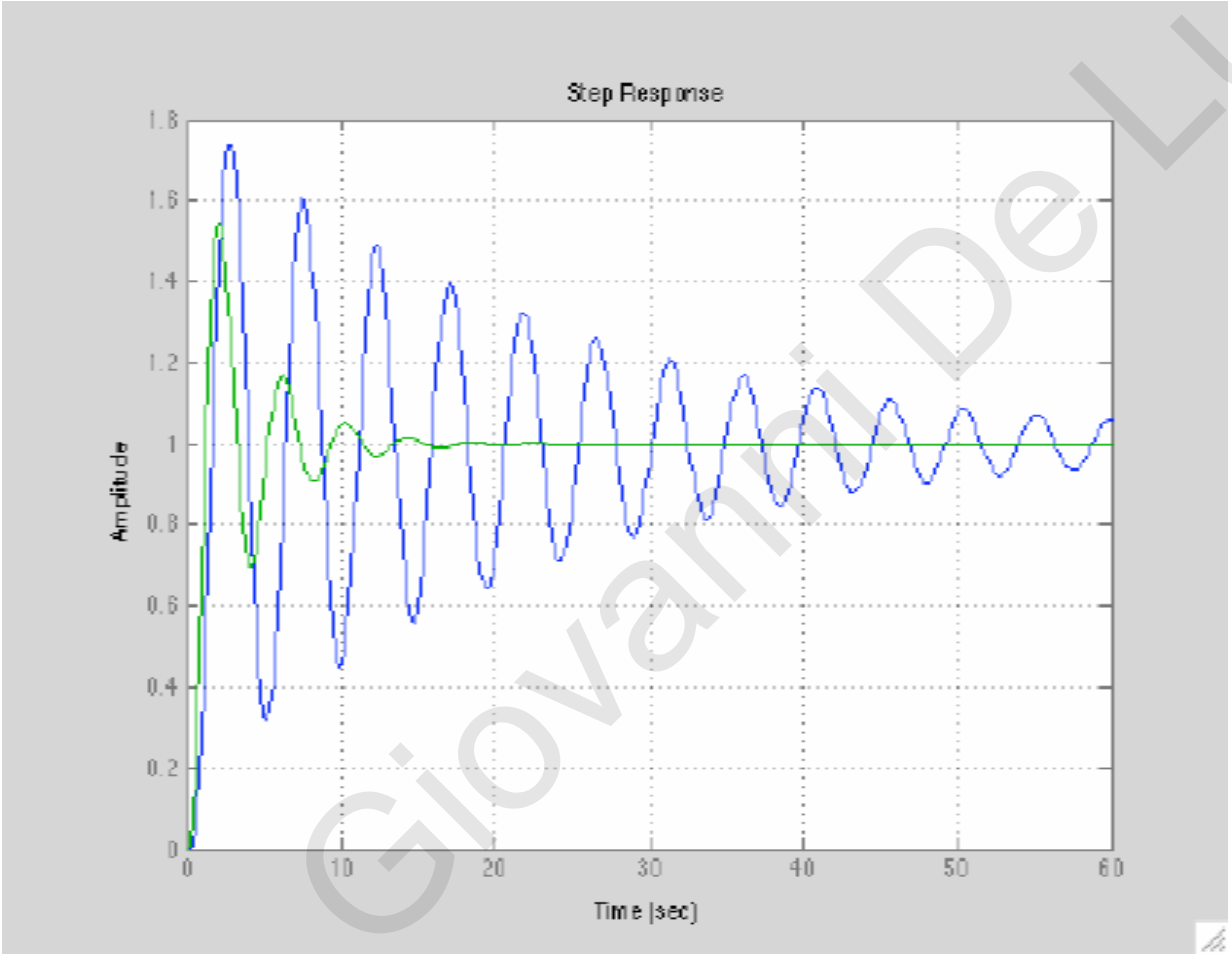
$$I = I + K_i * \varepsilon$$

$$D = K_d * \delta\varepsilon / \delta t$$

Regolazione velocità



Confronto uscita PI e PID



PID

PI

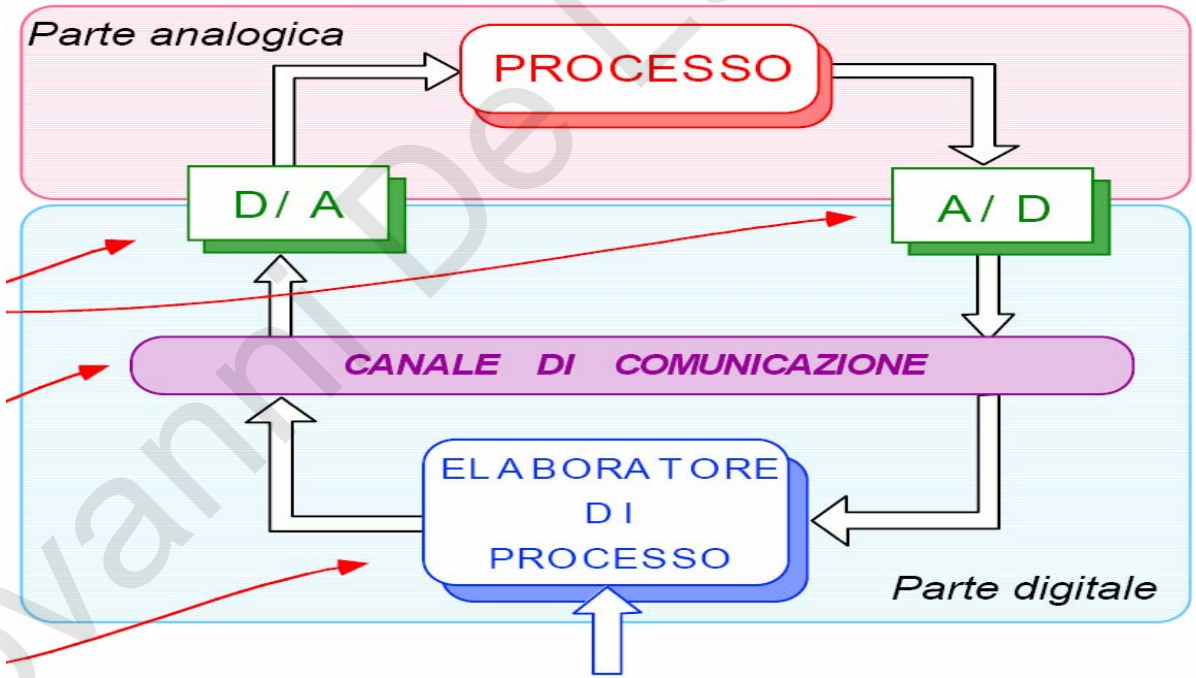
IL regolatore PID ha un margine di fase maggiore, quindi la risposta avrà Sempre una minore oscillazione.

Struttura di un regolatore digitale

Sistemi Hardware/Software di interfacciamento

Protocolli

Algoritmi



Specifiche di processo

Settaggio PID

- Il difficile in tutto questo è regolare le tre **K**. Queste determinano infatti il peso dei tre parametri sul totale della nostra regolazione, per fare una cosa scientifica dovremmo avere i parametri meccanici, di risposta e forse anche le previsioni del tempo, ovviamente a livello hobbistico dove spesso si lavora con materiali di recupero non è cosa fattibile, non ci resta che il metodo empirico e cioè la sperimentazione e le prove.
- Seppur empirico è un metodo e va fatto non per tentativi casuali (che può funzionare ma si tratta di altro … metodo) ma con cognizione di causa e criterio.
- IL suggerimento è quello di impostare un parametro per volta ovvero disabilitare le regolazioni **I** e **D** e cercare di ottenere il massimo dell'efficienza dalla sola regolazione proporzionale, aggiungere poi la regolazione **I** e poi la **D** cercando sempre di fare piccole variazioni.
- Siccome iniziamo dal parametro proporzionale potrebbe risultare necessario tenere questo un po' più basso rispetto alla prova iniziale e questo per il semplice motivo che comunque andiamo ad aggiungere gli altri parametri e ciò potrebbe portare il tutto oltre le possibilità del regolatore o di risposta della macchina.
- A questo punto un **line-follower** sarebbe veramente utile per vedere se ciò che abbiamo fatto ha senso, in particolare ci servirebbe una scheda con un microcontrollore tipo **AVR** dove scrivere due righe di programma e infilare le formule che abbiamo visto, ah sì, ci serve anche il programma.

Il codice PID

Riporto di seguito il codice in pseudo C.

```
int PID (int val_cons)
{
/*La funzione PID è richiamata dall'interrupt del timer di sistema; la chiamata avviene ogni 10 msec., pertanto z-1 = 10 msec. La funzione PID riceve un valore intero che rappresenta il valore di consegna "val_cons ", la funzione restituisce un valore intero "DA_conv" che rappresenta il valore di riferimento per l'attuatore.
Le variabili:
float Upper_P_limit, Upper_I_limit, Upper_D_limit, Upper_Total_limit
float Lower_P_limit, Lower _I_limit, Lower _D_limit, Lower _Total_limit
float Kp, Ki, Kd
sono globali; sono introdotte e modificate tramite interfaccia HMI
*/
static int AD_Conv = 0; /*Lettura convertitore A/D: acquisisce la variabile di ingresso*/
static int DA_Conv = 0; /*Scrittura convertitore D/A: scrive la variabile di uscita*/
static int error = 0; /*differenza tra valore di consegna e valore reale */
static int old_error = 0; /*differenza tra valore di consegna e valore reale @ z-1 */
float P=0; /* componente proporzionale */
float I=0; /* componente integrale */
float D=0; /* componente differenziale */
float i_inst = 0; /* parte istantanea del processo di integrazione*/
float Out = 0; /* Totale regolazione */
error = val_cons - AD_Conv;
P = error * Kp;
if (P > Upper_P_Limit) P = Upper_P_Limit;
if (P < Lower_P_Limit) P = Lower_P_Limit;
if Ki > 0 {
        i_inst = error * Ki;
        I = I + i_inst;
```

… continua

```
if (I > Upper_I_Limit) I = Upper_I_Limit;
if (I < Lower_I_Limit) I = Lower_I_Limit; }
else
I = 0;
if Kd > 0 {
        D = Kd * (error – old_error);
        old_error = error;
                if (D > Upper_D_Limit) D = Upper_D_Limit;
                if (D < Lower_D_Limit) D = Lower_D_Limit;}
else
        D = 0;
        Out = P + I + D;
                if ( Out > Upper_Total_limit) Out = Upper_Total_limit;
                if (Out < Lower_Total_limit) Out = Lower_Total_limit;
                DA_Conv = Out;
                Return (DA_Conv);
}
```

La routine di gestione dell'interrupt, legata alla scadenza del System Timer, che richiama PID sarà simile a:

```
void RT_G( )
{
        static int Out_DA = 0; /*uscita D/A per riferimento)
        static int valore_di_consegna = 0;
        valore_di_consegna = act_val;
        Out_DA = act_ref_val;
        act_ref_val = PID(valore_di_consegna);
}
```

Il PID in BASCOM

```
Sub Pid(pid_setpoint , Pid_actual)

   Pid_error = Pid_setpoint - Pid_actual        'errore = posizione finale – posizione attuale
   Pid_out = Kp * Pid_error                     ‘uscita= Kp * errore
   Ptemp = Pid_error - Pid_prev_error           'variabile appoggio = dE / dt
   Pid_prev_error = Pid_error                   'calcolo errore precedente
   Ptemp = Kd * Ptemp                           'calcolo derivata
   Pid_out = Pid_out + Ptemp                    'uscita=P + D
   Ptemp = Ki * Pid_integral_error              'calcolo integrativa
   Pid_out = Pid_out + Ptemp                    ‘uscita=P+D+I
   Pid_out = Pid_out / Pid_scale                'aggiusta il fondo scala

   If Pid_out > 255 Then                        'windup superiore e inferiore
     Pid_out = 255
   Elseif Pid_out < -255 Then
     Pid_out = -255
   Else
      Pid_error = Pid_error + Pid_integral_error
      If Pid_error > 255 Then
         Pid_error = 255
      Elseif Pid_error < -255 Then
         Pid_error = -255
      End If
      Pid_integral_error = Pid_error
   End If


    If Pid_out => 0 Then Motor_dir_1 = 0                 'calcolo direzione
    If Pid_out < 0 Then Motor_dir_1 = 1                  'se si pilota un motore
    Pid_out = Abs(pid_out)                               'ed è necessario il segnale
    If Pid_out => Max_pwm Then Pid_out = Max_pwm         'di direzione
    Motor_pwm_1 = Pid_out

End Sub
```

Conclusione

- Come potete vedere vengono fissati dei valori massimi e minimi entro i quali ogni parametro può muoversi, viene anche inserita una istruzione **if** che controlla se la **K** relativa è impostata a zero, in questo caso il parametro non viene calcolato e impostato a zero, in questo modo è possibile disabilitare le correzioni singolarmente **P**, **I** e **D** in modo da fare la taratura.
- IL codice è facilmente convertibile anche in altri linguaggi se escludiamo l'assembler, e ci si può risparmiare anche la gestione dell'interrupt se questo codice diventa un loop infinito che si ripete e all'interno del quale mandiamo le variazioni all'hardware che regola ad esempio i motori.
- In questo modo commettiamo un errore che è quello di non rendere esattamente preciso l'intervallo di tempo tra una lettura e la successiva, regolatori particolarmente sofisticati e precisi potrebbero risentire molto di questa incoerenza.

Riassumendo (P) di PID

- Componente più sensibile al valore attuale dell'errore
- Un valore elevato di **Kp** comporta una reazione pesante anche per variazioni modeste del valore dell'errore
- Un valore basso di **Kp** trasferirà sulle variabili di controllo variazioni contenute anche in presenza di errori rilevanti
- Aumenta la banda passante (sistema più pronto)

(I) di PID

- Tiene conto della storia dell'andamento avuto dall'errore in passato
- Introduce un polo nell'origine
- L'errore a regime per ingresso a gradino diventa nullo
- Aumenta il ritardo della catena diretta di controllo e determina una riduzione dei margini di fase ed ampiezza del sistema retroazionato

(D) di PID

- Varia linearmente con la derivata dell'errore
- Azione anticipatrice, migliora i margini di stabilità
- Amplifica i segnali con contenuto armonico a frequenze elevate (può amplificare il rumore)
- Realizzazione fisica diversa dalla realizzazione teorica (zero nell'origine):
- Una variazione a gradino del riferimento genera una componente impulsiva sulla componente derivata.
- Si può filtrare il set-point o derivare solamente la variabile di processo al posto dell'errore